Departament d´Informàtica

DISEÑO E IMPLEMENTACIÓN DE UNA ARQUITECTURA MULTIAGENTE PARA LA AYUDA A LA TOMA DE DECISIONES EN UN SISTEMA DE CONTROL DE TRÁFICO URBANO

TESIS DOCTORAL

MEMORIA PARA OPTAR AL GRADO DE DOCTOR
PRESENTADA AL
DEPARTAMENT D'INFORMÀTICA
E.S.T.C.E.
UNIVERSITAT JAUME I
POR LUIS AMABLE GARCÍA FERNÁNDEZ
DIRIGIDA POR DR. FRANCISCO TOLEDO LOBO

CASTELLÓN DE LA PLANA, 2000

DON FRANCISCO TOLEDO LOBO, Catedrático del Área de Conocimiento de Ciencias de la Computación e Inteligencia Artificial de la Universitat Jaume I

CERTIFICA:

> Que la presente memoria "**DISEÑO E IMPLEMENTACIÓN DE UNA ARQUITECTURA MULTIAGENTE PARA LA AYUDA A LA TOMA DE DECISIONES EN UN SISTEMA DE CONTROL DE TRÁFICO URBANO**", ha sido realizada bajo mi dirección en el Departament d'Informàtica de la Universitat Jaume I por D. Luis Amable García Fernández, Licenciado en Informática, y constituye su tesis para optar al grado de Doctor.

Y para que así conste a todos los efectos oportunos, se extiende la presente certificación, en el lugar y la fecha indicados.

Castellón a 25 de mayo de 2000

Fdo. Dr. D. Francisco Toledo Lobo

A mis padres,

por todo el tiempo dedicado que no se puede compensar

AGRADECIMIENTOS

A mi director, Paco Toledo, por brindarme su confianza, tiempo y apoyo para poder realizar esta tesis doctoral.

A Salvador Moreno y Enrique Bonet del LISITT por las concisas respuestas y la excelente disposición con las que atendieron mis peticiones de ayuda.

A los componentes del grupo de investigación “Computación Científica y Paralela” por facilitarme la utilización del cluster de PCs en el que se han implementado parte de los prototipos propuestos en esta tesis.

A aquellas instituciones y organismos oficiales que han ayudado en la realización de esta tesis doctoral: el Departament d’Informàtica de la Universitat Jaume I de Castellón, el Departament d’Informàtica i Electrònica de la Universitat de Valencia (LISITT) y la Fundació Caixa-Castellò.

A mis compañeros de trabajo y amigos en el Departament d’Informàtica de la Universitat Jaume I por tener la paciencia de escucharme y apoyarme en los momentos difíciles.

A mi familia por su apoyo y ánimo incondicional y por haber pinchado lo justo en los momentos adecuados.

A mis amigos mediterráneos y transmontanos.

ÍNDICE DE LA MEMORIA DE TESIS DOCTORAL

Introducción

PARTE I: ANÁLISIS COMPARATIVO DE LOS SISTEMAS DE CONTROL DE TRÁFICO URBANO

Capítulo 1. Sistemas de Control de Tráfico Urbano

Capítulo 2. Sistemas Basados en el Conocimiento Aplicados al Control de Tráfico Urbano

PARTE II: ARQUITECTURAS MULTIAGENTE PARA LOS SISTEMAS DE CONTROL DE TRÁFICO URBANO

Capítulo 3. El Prototipo MASHGREEN SM 94-97

Capítulo 4. El Prototipo MASHGREEN DM 97-99

INTRODUCCIÓN

Índice de la introducción

Carácter de la Tesis

La investigación y desarrollo de nuevas herramientas y métodos aplicables a los denominados Sistemas de Tráfico Inteligentes (*ITS*) ha promovido una fuerte inversión en Europa durante la última década. Programas tales como *DRIVE I* y *II* (1989-1991, 1992-1994, *Dedicated Road Infrastructure for Vehicle safety in Europe*) o *PROMETHEUS* (*PROgramMme for European Traffic with the Highest Efficiency and Unprecedented Safety*) son una muestra de tales iniciativas. La situación, no obstante, tendió a cambiar, y las inversiones realizadas fueron cada vez menores[1]. Uno de los factores que explican este comportamiento se relaciona con la falta de un mercado dinámico y abierto que saque provecho de los resultados obtenidos en la ejecución de estos proyectos.

La situación actual en el mercado de los Sistemas de Control de Tráfico Urbano (SCTU) es que un pequeño número de empresas domina este sector. Además, los SCTU actuales son sistemas de *llave en mano*, es decir, son sistemas propietarios de las empresas que los comercializan, por lo que sus posibilidades de ampliación están seriamente amenazadas por altos costes económicos.

Sin embargo, el gran desarrollo que actualmente se está obteniendo en las áreas de la Tecnología de la Información y de las Comunicaciones está provocando un cambio de mentalidad respecto al desarrollo de los ITS. Este cambio está siendo impulsado por la Unión Europea para abrir los monopolios a la competencia, tanto en el suministro de la infraestructura como en la provisión de servicios.

Estos acontecimientos están volviendo a relanzar las iniciativas orientadas a la gestión del tráfico, como se puede observar al analizar el contenido del directorio general VII[2] - Transporte de la Comisión Europea. Entre otros proyectos de tráfico y transporte este directorio contiene proyectos recientemente finalizados relacionados con el control de tráfico urbano como son los proyectos *MUSIC* (1996-1999, *Management of traffic Using flow Control and other measures*), *INCOME* (1996-1999, *Integration of traffic Control with other Measures*) o *SMARTEST* (1997-1999, *Simulation Modelling Applied to Road Transport European Scheme Tests*).

En la reciente *Workshop on Models in Support ADVANCED TRAFFIC MANAGEMENT SYSTEMS (ATMS)* celebrada en Florida en Mayo de 1999[3] se

[1] *Los programas DRIVE y PROMETHEUS dejaron de financiarse, pero también bajo la responsabilidad de la DG XIII y dentro del Programa de Aplicaciones Telemáticas, Área A: Telemática para Servicios de Interés Público, surgió el Sector 2: Telemática para el Transporte, al que se suma, bajo la responsabilidad de la DG VII, el programa de Transporte.*

[2] *http://www.cordis.lu/transport/home.html.*

[3] *Exponsorizada por US Department of Transportation y la Federal Highway Administration. http://mctrans.ce.ufl.edu/TransportationTopics/ATMSWorkshopProceedings.pdf.*

identificaron una serie de problemas, con las correspondientes indicaciones para su solución, asociadas a los actuales ATMS:

- Dificultades asociadas con el desarrollo de los modelos de tráfico así como con los derechos de uso del software empleado para su implementación. Para abordar este problema se propone utilizar software de libre distribución suficientemente contrastado (*Free Software Foundation*) como herramienta con la que promover y desarrollar tanto los modelos de tráfico propuestos como nuevo software de investigación.

- Pérdida de datos significativos para la toma de decisiones sobre el tráfico. Se propone continuar soportando la investigación sobre el comportamiento del flujo de tráfico, así como ampliar las fuentes de datos que permitan monitorizarlo de una forma eficiente. Otra tarea a desarrollar es la investigación en cómo identificar los datos perdidos que resultan críticos para la correcta interpretación del comportamiento del tráfico.

- Problemas asociados al desarrollo de modelos y algoritmos para ATMS en tiempo real. Se trata de establecer cuales son las características fundamentales que han de ser consideradas: identificación de las entradas disponibles y de cuando lo están, los resultados que se pueden esperar de su análisis, los objetivos de control en el sistema, la cantidad de recursos (temporales, recursos de cómputo, de comunicaciones, etc.) disponibles para resolver cada tarea y las restricciones operativas asociadas tanto a los usuarios como al propio sistema ATMS.

- Los problemas asociados a la asignación dinámica de tráfico. Se propone el desarrollo y empleo de modelos avanzados de tráfico para analizar los datos, especialmente los asociados a las tareas de observación del tráfico en tiempo real y de la determinación de incidentes mediante su análisis. Otro factor relacionado consiste en determinar la información consistente con la que poder conseguir los objetivos asociados a la gestión de tráfico, operación y control de acuerdo a estos modelos.

El planteamiento de la presente memoria de tesis trata de abordar, y proponer soluciones, a algunos de los ítems anteriores. En particular, debido a la fuerza impulsora que la Telemática está recibiendo, se propone el estudio de una arquitectura funcional que pueda ser capaz de integrar distintas funcionalidades, e incluso poder aumentarlas mediante la adición sencilla de nuevos componentes en el SCTU propuesto.

La ejecución de los proyectos *ESPRIT II EQUATOR* y *CHIC* [EQUATOR 90, 91, 94] permitió elaborar una serie de técnicas para el razonamiento sobre el comportamiento temporal de sistemas basados en procesos. El conjunto de las técnicas desarrolladas debían de servir como herramientas básicas con las que construir sistemas de gran escala basados en el conocimiento aplicables a distintas áreas de comercio e industriales.

La presente tesis continua con la dirección apuntada por los resultados de estos proyectos mediante la aplicación de algunas de sus técnicas a la tarea de control del tráfico urbano [Moreno, 93], [Moreno, 96], [Toledo, 96]. Con este propósito, se persiguen tres objetivos iniciales:

- La identificación e implementación de aquellas tareas asociadas al control de tráfico urbano que puedan utilizar los métodos desarrollados en los proyectos ESPRIT II EQUATOR y CHIC. De esta forma se pretende conseguir un modelo en el que se puedan integrar, por ejemplo, datos procedentes de diferentes tipos de sensores (lazo cerrado[4], imágenes de vídeo[5], telefonía móvil[6], etc.). De esta forma (mediante la incorporación del transductor adecuado) se consigue un modelo independiente de los diferentes tipos de sensores disponibles en el sistema y de las diferentes medidas y/o magnitudes que pueden proporcionar.

- La definición de un prototipo de arquitectura funcional en la que integrar el conjunto de las tareas desarrolladas. En esta arquitectura se debe de cumplir que las prestaciones en la ejecución de cada una de las tareas no se tienen que ver comprometidas por la incorporación de nuevas tareas, o por las interacciones entre las tareas disponibles.

- La implementación del prototipo en una arquitectura con altas prestaciones computacionales (inicialmente el modelo propuesto es un multiprocesador con memoria compartida).

Cada uno de estos objetivos se han intentado conseguir utilizando diversas aproximaciones. La tendencia general que ha guiado este proceso ha sido siempre la búsqueda de un equilibrio entre los costes asociados a realizar las tareas respecto a la calidad de la respuesta obtenida. Algunos de los proyectos desarrollados en el ámbito de la Ingeniería de Tráfico priman la obtención del máximo número posible de medidas de tráfico respecto del tiempo empleado para calcularlas (como es el caso al emplear simuladores de tráfico de tipo microscópico[7]). La mejora de prestaciones, en este tipo de sistemas, se consigue mediante el empleo de mecanismos de programación paralelos. No obstante, estos proyectos todavía se encuentran en fase de investigación y desarrollo no encontrándose productos comerciales disponibles actualmente que los soporten.

---

[4] *proyecto ESPRIT II EQUATOR*

[5] *proyecto CICYT TI95-0704 "Técnicas de visión artificial cualitativas e integración con un sistema experto de segunda generación para el control de tráfico urbano en tiempo real"*. Periodo de desarrollo 1995-1998

[6] Tesis Doctoral *Estudio de una nueva arquitectura para la monitorización del tráfico por carretera basada en el sensor móvil apoyado en la telefonía celular digital* presentada por José Miguel Castellet Martí, Noviembre 1999.

[7] *Un simulador microscópico considera cada vehículo que compone el tráfico urbano de forma individual, calculando para cada instante de tiempo todos los parámetros que definen su comportamiento.*

El enfoque seleccionado en esta tesis para realizar estas tareas precisa de una aproximación diferente. En vez de conseguir el máximo de medidas posibles, todas ellas valores numéricos que hay que interpretar, se trata de establecer un modelo que permita diferenciar entre datos, información y conocimiento. De esta forma, mediante el empleo de un modelo que explique el comportamiento del tráfico urbano se obtienen menos medidas, pero capaces de representar de una forma significativa la evolución del tráfico urbano. El modelo seleccionado para explicar el comportamiento del tráfico es un modelo cualitativo. Como consecuencia, se consiguen unas medidas cuyo proceso de generación presenta un bajo coste computacional. Resulta evidente que con este enfoque no se pueden obtener todas las medidas que puede proporcionar el uso de un modelo cuantitativo. No obstante, la rapidez con que se obtienen estas medidas permite incorporar al Sistema de Control de Tráfico Urbano otras habilidades como puede ser, por ejemplo, la predicción del estado del tráfico a medio plazo.

Adicionalmente a estas características funcionales, en el proceso de desarrollo del prototipo se propone valorar el uso de la *programación lógica basada en restricciones de dominio finito* [Hentenryck, 89], [Marriot, 98] como herramienta que permita desarrollar prototipos en un corto periodo de tiempo.

Todas las tareas desarrolladas deben estar integradas en una arquitectura funcional que proporcione un marco donde las prestaciones de cada una de esas tareas ejecutadas de forma individual sean equiparables a sus prestaciones cuando se ejecutan de forma conjunta. Una de las técnicas más utilizadas para conseguir estos propósitos consiste en emplear programación paralela y/o distribuida.

La búsqueda de una arquitectura computacional en la que implementar la arquitectura funcional propuesta se realiza buscando el equilibrio de tres factores:

- facilidad de adquisición (a ser posible que no se trate de un modelo particular de computador que limite la capacidad de implementación de la arquitectura funcional propuesta a otros modelos de computadores),

- disponibilidad de herramientas de programación adecuada (que permitan la implementación de los mecanismos de coordinación entre las tareas de una forma eficiente), y

- bajo coste económico.

Todas estas características se dan en los sistemas tipo *COTS* (*Mass-Market Commodity off-the-self*). Estos sistemas se caracterizan por estar compuestos de elementos de bajo coste y disponibles como productos de consumo masivo (amplia disponibilidad). El término procede de la descripción dada por Thomas Sterling (*Caltech*) para describir el cluster de PC que permitió alcanzar potencias sostenidas de 10 Gigaflops. En particular, estos clusters reciben el nombre de sistemas *Beowulf*.

El desarrollo de la presente tesis se ha realizado mediante los resultados obtenidos por su autor en la participación en los siguientes proyectos de investigación financiados por diversos organismos e instituciones oficiales:

Título proyecto:	*Diseño e Implementación de una Arquitectura Paralela con Memoria Compartida para la Construcción de un Sistema Basado en el Conocimiento Aplicado al Control de Tráfico Urbano.*
Financiación:	*Fundació Caixa-Castelló*.
Periodo de desarrollo:	*1994-1997*.
Coste:	*3.710.000 pts*.
Demostraciones:	*Castellón de la Plana*.
Breve descripción:	*Diseño de una arquitectura funcional basada en el modelo de pizarra como medio en el que integrar diferentes tareas cualitativas con las que gestionar el control de trafico urbano.*

Este proyecto fue evaluado por la ANEP (Agencia Nacional de Evaluación y Prospectiva) recibiendo la calificación de *Bueno* (segunda máxima calificación que este organismo oficial puede otorgar).

Título proyecto:	*Plan Director para el análisis del tráfico y transporte de Castellón de la Plana*.
Financiación:	*Excmo Ayuntamiento de Castellón de la Plana*.
Periodo de desarrollo:	*1998*.
Coste:	*5.000.000 pts*.
Demostraciones:	*Castellón de la Plana*.
Breve descripción:	*Recogida de datos de tráfico con el que evaluar los esquemas de gestión utilizados por el actual sistema de control de tráfico urbano en Castellón de la Plana*.

Organización de la Memoria de Tesis

La presente memoria de tesis describe la evolución en la definición, análisis e implementación de cada uno de los objetivos inicialmente propuestos. Para ello define dos bloques. La primera parte se denomina *Análisis Comparativo de los Sistemas de Control de Tráfico Urbano*. De esta forma, en el capítulo 1, *Sistemas de Control de Tráfico Urbano*, se realiza un análisis de los principales resultados de la Ingeniería de Tráfico en el desarrollo de SCTU. El capítulo comienza identificando los objetivos y beneficios operacionales que pueden ser conseguidos mediante el uso de los Sistemas de Control de Tráfico Urbano así como la definición de sus componentes básicos y los conceptos clave para su utilización. El capítulo finaliza mediante la exposición de tres de los sistemas de control de tráfico urbano con más implantación en el mercado (SCOOT, SCATS y RT-TRACS).

Una vez establecida cual es la situación actual en la aplicación de estos sistemas, el capítulo 2, *Sistemas Basados en el Conocimiento Aplicados al Control de Tráfico*

Urbano, realiza un análisis de cuales son los principales problemas asociados a los actuales SCTU. Posteriormente a este análisis, se identifican cuales han sido las principales contribuciones del campo de la Inteligencia Artificial asociadas a la definición de prototipos "inteligentes" para el control de tráfico urbano. Este estudio destaca el uso de los modelos de razonamiento *profundo* como herramienta fundamental para la construcción de estos prototipos. Como consecuencia se expone los prototipos desarrollados aplicando los mecanismos de razonamiento *profundo*. En esta exposición se destaca el papel importante que desempeña la forma en que se relacionan los componentes de este prototipo con el propósito de conseguir mejores prestaciones (respuestas en tiempo real y aplicabilidad en dominios con un gran número de componentes).

Estos dos capítulos comprenden lo que se puede denominar "*estado del arte*" referente a la presente memoria de tesis. Una vez establecidos, comienza la parte segunda: *Arquitecturas Multiagente para los Sistemas de Control de Tráfico Urbano*.

El primero de los prototipos desarrollados se expone en el capítulo 3, *El Prototipo MASHGREEN*[8] *SM 94-97* (a ***M***ulti***A***gent ***S***ystem for helpin***G*** u***R***ban traffic d***E***cision tak***EN*** with ***S***hared ***M***emory). Este prototipo sigue un modelo general de arquitectura de pizarra [Nii, 86], [Velthuijsen, 92] ampliada mediante el paso directo de mensajes entre sus componentes imperativos. La primera cuestión que se plantea es la identificación de las principales funciones que debe ser capaz de realizar, así como la organización funcional de cada uno de sus componentes principales. Estos componentes son analizados minuciosamente, exponiendo los métodos que utilizan para resolver cada una de las tareas de las que son responsables. Cada uno de ellos utiliza el proceso de simulación temporal cualitativa expuesto en el capítulo 2 como herramienta fundamental con la que realizar el análisis profundo del estado del tráfico urbano. Este capítulo finaliza con la descripción de los principales problemas detectados durante el desarrollo secuencial de este primer prototipo.

Como consecuencia de este análisis se define e implementa otra arquitectura funcional que permite resolver parte de los problemas identificados. Esta nueva arquitectura se define en el capítulo 4, *El Prototipo MASHGREEN DM 97-99* (a ***M***ulti***A***gent ***S***ystem for helpin***G*** u***R***ban traffic d***E***cision tak***EN*** with ***D***istributed ***M***emory), de la presente memoria de tesis doctoral. Los métodos descritos en el capítulo anterior para realizar cada una de las tareas de control de tráfico urbano siguen siendo utilizados en el nuevo prototipo, pero se cambia la forma en que estos se ejecutan y de cómo organizan sus datos. La nueva arquitectura funcional utilizada responde a un modelo multiagente [Durfee, 94], [Russell, 95]. De esta forma, se intenta conseguir un prototipo cuyo comportamiento sea distribuido y que permita la gestión dinámica de sus componentes imperativos.

[8] *El nombre se podría traducir como pulpa verde. El significado que se le ha querido dar hace referencia a la distribución del tiempo de verde disponible en cada cruce de forma que se minimice el grado de congestión global de la red urbana.*

Este nuevo prototipo define una entidad imperativa, un *agente*, para realizar cada tarea relevante en el control del tráfico urbano. Cada uno de estos agentes comparte la misma organización funcional, así como el conjunto de métodos mediante el cual establecer el intercambio de datos e información. El conjunto de agentes implementado comparte un mismo objetivo común, la resolución de la tarea de gestión de tráfico urbano mediante la monitorización, detección y cálculo de las mejores acciones de control a ejecutar para evitar las potenciales congestiones de tráfico urbano.

Una vez realizado el proceso de definición de cada agente en el nuevo prototipo, la siguiente tarea a desarrollar consiste en buscar un sistema de computadores en el que realizar su implementación. Esta tarea se describe en el capítulo 5, *Resultados de la Implementación*. Un análisis de las perspectivas actuales respecto a las arquitecturas computacionales utilizadas para realizar la denominada *computación de altas prestaciones*, determina que una buena elección consiste en la utilización de un cluster de PC tipo *Beowulf*. Por lo tanto, se plantea cómo implementar el prototipo *MASHGREEN DM* en este sistema. Los resultados del comportamiento en ejecución de cada agente del prototipo, así como algunas de las incidencias de este proceso, se exponen al final de este capítulo.

El último capítulo de la memoria de tesis expone las conclusiones finales respecto del comportamiento del prototipo *MASHGREEN DM* así como el trabajo futuro que queda por desarrollar.

La memoria contiene dos apéndices. El primero de ellos especifica los conjuntos de datos utilizados para evaluar los prototipos desarrollados. Estos datos fueron el resultado de uno de los proyectos ejecutados durante el desarrollo de esta tesis doctoral. La falta de datos reales procedentes de los sensores de tráfico del escenario seleccionado para su ejecución determinó la creación de varias alternativas de asignación de datos cualitativos iniciales para cada uno de estos conjuntos. Una de las situaciones representa un tráfico ligero, mientras que la otra situación representa un tráfico ligeramente congestionado.

El segundo apéndice contiene imágenes del interfaz de usuario captadas durante la ejecución del prototipo *MASHGREEN DM*. Estas imágenes proporcionan una idea del modo de relación que se puede establecer entre el operador del prototipo y los resultados proporcionados por su ejecución.

PARTE I

ANÁLISIS COMPARATIVOS DE LOS SISTEMAS DE CONTROL DE TRÁFICO URBANO

CAPÍTULO 1
SISTEMAS DE CONTROL DE TRÁFICO URBANO

Índice del capítulo 1

1.1 Introducción

El constante incremento en la cantidad del número de vehículos en circulación así como la creciente complejidad estructural de la red urbana junto con otras importantes consideraciones de tipo económico y social exigen el establecimiento de algún mecanismo que sea capaz de gestionar el tráfico urbano. El empleo de señales de tráfico en las intersecciones permite controlar los movimientos de vehículos mediante la asignación de un tiempo de uso para cada una de las direcciones que confluyen en una intersección.

La tarea de distribuir tanto el tiempo como el espacio físico disponible, para optimizar el movimiento de vehículos de una manera coordinada en una intersección se extiende de forma natural a la inclusión de intersecciones adyacentes. Esta es la base de la mayoría de sistemas de control de tráfico urbano (CTU).

1.1.1 Objetivos Operacionales de un Sistema CTU

Los sistemas de control de tráfico urbano se han popularizado fundamentalmente por dos razones [RTUA, 87]; pueden hacer un mejor uso de la capacidad de la red urbana existente y reducir, de esta forma, los tiempos de viaje sin crear efectos adversos en el entorno (mediante la reducción de las situaciones de congestión de tráfico y del tiempo necesario para realizar un recorrido se pueden llegar a conseguir reducir tanto la contaminación acústica como la ambiental); y proporcionan el mecanismo básico para la incorporación de otras características tales como la monitorización de las congestiones de tráfico, el establecimiento de prioridades para vehículos del servicio de urgencias o la inclusión de señales de tráfico con paneles informativos cuyo contenido varía con el tiempo.

Un sistema de CTU puede ser empleado para influir en el patrón de tráfico de un área con el propósito de conseguir determinados objetivos (por ejemplo, ayudar o fomentar el comercio en una determinada área de la ciudad). Los resultados se concentran principalmente en conseguir la mejor prestación de tráfico posible en cada momento mediante la minimización del retraso de los vehículos debido al número de veces que estos tienen que detenerse y de la velocidad máxima que estos pueden conseguir en su trayecto. Otras medidas para la evaluación de las prestaciones son la reducción en el consumo de combustible o la minimización de la contaminación provocada por los vehículos, aunque todos estos objetivos están fuertemente ligados [RTUA, 87].

Otros objetivos pueden ser el atraer o expulsar tráfico de determinadas áreas o rutas, dar prioridad a ciertas categorías de usuarios para algunos tramos de la red o dirigir las congestiones a ciertas áreas en las que sus efectos sean menos perjudiciales, por ejemplo, por disponer de una mayor longitud de vía, o por ser una zona no residencial [RTUA, 87].

1.1.2 Los Beneficios Potenciales en el Uso de Sistemas CTU

Entre los beneficios más importantes se destacan los siguientes [RTUA, 87]:

- La reducción en el tiempo empleado para realizar rutas, número de paradas obligatorias, consumo de combustible y contaminación ambiental.
- La creación de prioridades para los servicios de transporte público de viajeros y la asignación de rutas fijas para los autobuses.
- La asignación de prioridades para vehículos de emergencias en respuesta a incidentes mediante la asignación de valores óptimos para las señales que favorezcan determinadas rutas (por ejemplo hacia los hospitales).
- La implementación de diferentes esquemas para tratar con emergencias o con eventos especiales (deportivos, festivos, ...)
- La mejora en la determinación de fallos en los sensores de detección del tráfico.
- El mejor aprovechamiento de los aparcamientos públicos y la mejora en la reducción de tráfico en circulación mediante la disposición de sistemas de información del estado de los aparcamientos públicos como parte integrante del sistema CTU.
- La creación de una base de datos centralizada sobre información de tráfico actualizada constantemente.

1.2 Elementos Básicos

Una red urbana se compone de distintos elementos cuyas características e interacción determinan completamente el comportamiento global de la red. Los componentes básicos son los siguientes:

- *segmentos*: caracterizados por su longitud, número de carriles, elemento de la red en el que comienza y elemento en el que finaliza. Sólo se considera un sentido de tráfico por segmento, por lo que una calle, tal como habitualmente se entiende, suele estar constituida por varios segmentos, tanto para expresar los dos sentidos de circulación (cuando ambos estén presentes) como las intersecciones que atraviesan.
- *intersecciones*: son los elementos que principalmente definen los comienzos y finales de los segmentos. Su principal función consiste en conceder derecho de paso a unos sentidos respecto de otros sentidos de circulación, permitiendo realizar los cambios de dirección en el flujo de tráfico de una forma eficiente y segura.

Para gestionar el flujo de tráfico en las intersecciones se suelen utilizan distintos métodos:

- Regla implícita de prioridad de sentido de circulación del tráfico que circula por la derecha.

- Señalización de la prioridad de paso por medio de señales de tráfico no luminosas, por ejemplo, señales verticales (stop, ceda el paso, vía con preferencia,...)
- Distribución del derecho de paso entre las distintas direcciones de una intersección por medio de señales luminosas.

Los sistemas de regulación del tráfico basados en semáforos se instalan cuando el tráfico en una intersección resulta demasiado denso para que los vehículos circulen de una forma eficiente o bien cuando se debe de mejorar la seguridad del tránsito de vehículos y peatones por la misma.

La instalación de semáforos es una forma más restrictiva de control que las intersecciones no reguladas por semáforos. Por este motivo, los sistemas de semáforos están considerados como el método más drástico a aplicar para actuar sobre el control de tráfico en una intersección.

El funcionamiento de los semáforos está gobernado por el regulador que es un mecanismo que ordena, cada cierto tiempo, cambios en las luces de los semáforos que gestiona, de forma que periódicamente todas las direcciones de la intersección dispongan de derecho de paso durante un intervalo finito de tiempo.

Asociados a estos reguladores se definen una serie de conceptos que caracterizan a su funcionamiento [HCM, 85]:

- *ciclo*: es el tiempo necesario para que se dé una sucesión completa de indicaciones en los semáforos conectados a un mismo regulador. Cada una de estas combinaciones de indicaciones permiten la realización de uno o varios movimientos simultáneos a través de la intersección. Cada una de estas combinaciones recibe el nombre de *fase*.
- *reparto*: es la distribución del tiempo de duración del ciclo entre todas las fases asociadas a una intersección. Los tiempos dedicados a cada una de las fases se especifica por medio de un vector, *v*, tal que la componente *v[i]* indica el tiempo dedicado a la ejecución de la fase *i*.
- *Factor de pico en una hora*: la relación entre el número de vehículos que entran en una intersección en la hora pico respecto 4 veces el número de vehículos que entran en la intersección en el periodo de 15 minutos de la hora pico de mayor flujo. Si no se dispone de esta información entonces se asume un valor de *0.85*, por lo que el flujo de tráfico durante el periodo de quince minutos será *N / (4 x .85)*, es decir *0.3N* donde *N* es el flujo de pico en una hora.
- *Intervalo de fluidez (Interverde)*: el tiempo que transcurre entre el final de la indicación de verde para una fase y el comienzo de la indicación de verde para la siguiente fase en la misma intersección. Suele mostrarse en las intersecciones como una señal ámbar intermitente, y su duración debería de aproximarse a un valor que redujese el área de incertidumbre en las cercanías de la intersección.

- *Intervalo todas-en-rojo:* el tiempo en el que todas las direcciones de la intersección están simultáneamente en rojo.

La decisión de qué *fases* se han de definir para cada cruce no puede sujetarse a reglas fijas, sino que dependerá, en general, de las características del tráfico y del trazado de la intersección. Los criterios que deben presidir el estudio de las fases son los siguientes [HCM, 85]:

1. El número de fases debe ser el menor posible. Con ello se reducen al mínimo los tiempos perdidos en cada ciclo.
2. El número de movimientos simultáneos –sin conflictos comunes– debe ser el máximo posible.
3. El recorrido dentro de la intersección se procurará que sea lo más corto posible, para así lograr unos tiempos de desalojo de la intersección más cortos.
4. Cuando un ciclo se divida en dos o más fases será necesario considerar el orden en que se suceden, ya que ello influye en la seguridad y rendimiento de la intersección.
5. Cuando durante la fase de movimiento de peatones, tanto el número de éstos como el de vehículos que tratan de salir de la intersección no sea muy elevado, puede permitirse el paso simultaneo de ambos asignando la preferencia de paso a los peatones. Si uno de los dos movimientos es relativamente importante puede ser necesario establecer unas fases que los separen totalmente, ya que en caso contrario podría resultar peligroso para los peatones o podrían producirse colas de vehículos que llegasen a bloquear la intersección.

La figura 1.1 muestra la sucesión de fases y los sentidos de circulación permitidos en cada una de ellas para una intersección ejemplo.

Un semáforo, una vez instalado, puede funcionar de forma constante o flexible. En la práctica la instalación en intersecciones de semáforos con un mecanismo de funcionamiento constante resulta cada vez menos habitual:

1. Operación constante. Las indicaciones de rojo y verde son temporizadas a valores constantes calculados mediante el análisis del comportamiento histórico del tráfico en la intersección. Este tipo de operación asume que los patrones de tráfico pueden predecirse según sea la hora del día y, por lo tanto, no precisa de detectores de tráfico en la intersección que regulan. Este tipo de operación sólo suele ser utilizada cuando no se dispone de un presupuesto suficiente para implementar una operación flexible.
2. Operación flexible. Las intersecciones que operan de esta manera consisten de controladores de tráfico y *detectores de vehículos* colocados en las vías próximas a la intersección. En una operación flexible se tiene que calcular la duración de los intervalos de verde. Los intervalos de verde pueden finalizar en una de las siguientes cuatro formas:

1. Alcance del tiempo máximo de tiempo verde (*timing out*): El intervalo se acaba cuando se alcanza un tiempo máximo de verde previamente establecido.
2. Disminución sensible del flujo de tráfico en las proximidades de la intersección (*gapping out*): cuando se presenta un flujo ligero de tráfico que es menor que un cierto valor umbral previamente establecido.
3. Finalización por orden del sistema (*force-off*): Cuando un sistema de semáforos es parte de un sistema coordinado, el sistema mantiene la señal en espera con la operación indicada.
4. La señal es expulsada. Cuando un vehículo prioritario, por ejemplo una ambulancia o un coche de bomberos, se aproxima a la intersección, los intervalos de verde asociados a movimientos no prioritarios deben de terminar en favor de los que están asociados a esos movimientos prioritarios.

En la mayoría de los casos cuando se tratan de regular intersecciones aisladas se utilizan los dos primeros métodos, El tercer método se emplea en operaciones coordinadas, y el último método, expulsión, es un caso especial.

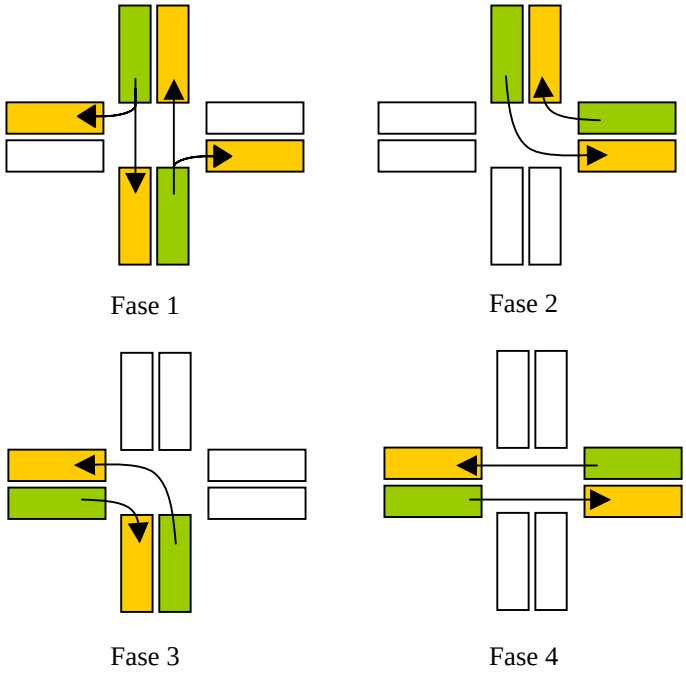


Fig 1-1. *Esquema de las fases de una intersección clásica. Para especificar el tiempo asignado para la ejecución de cada una de ellas se utiliza un vector de reparto. Por ejemplo, sea* t_{ciclo} *= 100, entonces el vector de reparto (30, 20, 20, 30) representa que para la ejecución de la Fase 1 se van a emplear 30 segundos, 20 para la Fase 2, 20 para la Fase 3, y 30 para la última fase.*

Los detectores de vehículos son los dispositivos necesarios para implementar en un regulador el tipo de operación flexible. Estos elementos se activan, normalmente, con el

paso o la presencia de un vehículo por la zona de detección. Existe una gran variedad de principios físicos utilizados para su implementación: presión, distorsión de un campo magnético, interrupción de un foco de luz, un cambio en la frecuencia de onda del radar, un cambio en la inductancia de una espira, detección de vídeo mediante el empleo de técnicas de procesado de imágenes,... Los detectores difieren respecto a su coste, mantenimiento, medidas suministradas, así como en su fiabilidad y serán objeto de estudio en el siguiente apartado.

Dos conceptos básicos en el análisis del tráfico para cada intersección son su *capacidad* y *nivel de servicio*. La definición de estos conceptos precisa del establecimiento de los valores asociados al *flujo de saturación* y a las *relación del flujo*. El *flujo de saturación* es una medida que representa básicamente la tasa de servicio: el número máximo de vehículos que puede circular por un grupo dado de carriles asumiendo que para este grupo de carriles el 100% de tiempo real disponible es tiempo de verde efectivo y se expresa como número de vehículos por hora de tiempo verde efectivo. La figura 1.2 muestra la forma ideal que tiene la curva del flujo de vehículos en una intersección.

La *relación del flujo* se define como el cociente del flujo actual o previsto para cada grupo de carriles, *V*, y el valor del flujo de saturación, *s*, es decir, el cálculo de esta medida para el grupo de carriles *i* es $(V / s)_i$.

La *capacidad* para un grupo de carriles *i*, denotado por c_i, se puede expresar mediante la fórmula siguiente: $c_i = s_i(g_i / C)$ donde

c_i se mide en vehículos por hora

s_i es el flujo de saturación para el grupo de carriles *i*

g_i es el tiempo de verde efectivo

C es la duración del ciclo

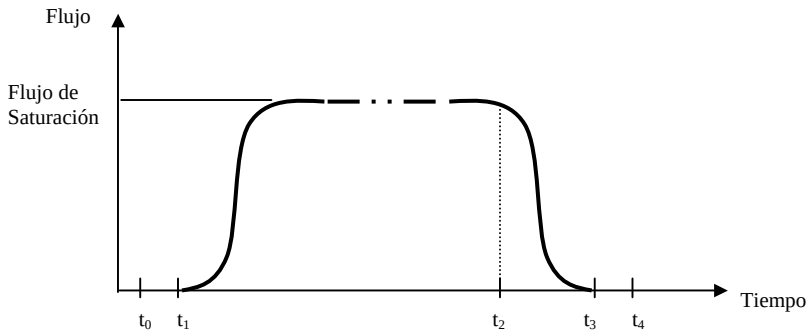


Fig 1-2. *Gráfica del concepto de flujo de saturación. En t_0 comienza la luz verde; en t_1 el primer vehículo cruza la intersección; la cantidad $t_1 – t_0$ representa el tiempo de respuesta del conductor; en t_2 comienza la luz ámbar (final de luz verde); en t_3 el último vehículo cruza la intersección; en t_4 comienza la luz roja; la cantidad $t_2 – t_1$ representa el tiempo de verde y la cantidad $t_3 – t_1$ representa el tiempo de verde efectivo.*

La relación entre el flujo actual o medido para un grupo de carriles *i* y la capacidad nos define el *grado de saturación* y, por lo tanto, su cálculo se puede realizar por medio de la siguiente expresión:

$$X_i = \left(\frac{V}{C}\right)_i = \frac{V_i}{s_i\,(g_i/C)} = \frac{V_i C}{s_i g_i} = \frac{(V/s)_i}{g_i/C}$$

Los valores para X_i van desde *1.0* cuando el valor del flujo actual o medido iguala a la capacidad, hasta *0.0*, cuando el valor del flujo medido o actual es 0.0. Otro concepto de capacidad de gran utilidad en el análisis de intersecciones reguladas por semáforos consiste en el cálculo de la relación crítica *V / c*, denotada por X_c. Esta medida consiste en calcular la relación *V / c* para cada fase, considerando en cada una de las mismas aquel grupo de carriles que presente la mayor relación de flujo, *V / s*. Es decir, si durante una fase se permiten que grupos de carriles distintos tengan acceso a la intersección sólo se selecciona la mayor relación *V / s* de entre todas las presentes. Entonces, cada fase tendrá un grupo de carriles críticos que determinan los requisitos de tiempo verde de la fase.

La relación crítica *(V / c)* para la intersección, como se ha indicado, se define en términos de grupos de carriles críticos

$$X_c = \sum_i \left(\frac{V}{s}\right)_{ci} \frac{C}{C - L}$$

donde

$\sum_i (V/s)_{ci}$ es la suma de las relaciones de flujo para todos los grupos de carriles críticos

C es la longitud del ciclo

L es el tiempo perdido por ciclo, calculado como la suma del tiempo perdido (normalmente es igual a la suma de las duraciones de las luces ámbar y roja).

El cálculo de la relación X_c es particularmente útil en intersecciones con algunos grupos de carriles sobresaturados, puesto que en el caso de que existan y de que el valor de X_c sea menor que *1.0* (lo que indica que la intersección no se está utilizando de forma óptima) basta con realizar una redistribución de los tiempos de verde durante la misma duración de ciclo para mejorar las condiciones de funcionamiento de la intersección.

El *nivel de servicio* es un parámetro con el que se trata de describir las condiciones de tráfico en una vía o intersección. En el caso de intersecciones reguladas este término se define por medio de unidades de retraso. El retraso es una medida del disgusto de un conductor, su frustración, consumo de combustible, e incremento de tiempo que emplea en la realización de un trayecto. Los criterios de nivel de servicio se establecen en términos del retraso medio de parada por vehículo durante un periodo de análisis de 15 minutos y se corresponden con los valores de la siguiente tabla [HCM, 85]

Nivel de servicio	*Retraso de parada por vehículo (seg.)*
A	≤ 5.0
B	5.1 – 15.0
C	15.1 – 25.0
D	25.1 – 40.0
E	40.1 – 60.0
F	> 60.0

El nivel *A* se corresponde con el concepto de *circulación fluida* en el que el flujo de tráfico es bajo y la velocidad alta, por lo tanto, el conductor elige libremente la velocidad de circulación sin recibir restricciones a causa del resto de tráfico.

El nivel *B* indica una *circulación estable con alta velocidad* aunque ésta comienza a depender del tráfico.

El nivel *C* representa el concepto de *circulación estable* con más dependencia del tráfico que en el nivel anterior.

El nivel *D* se caracteriza por representar una *circulación casi inestable* es la que se producen cambios bruscos en la velocidad aunque durante cortos periodos de tiempo.

El nivel *E* indica una *circulación inestable* en la que el flujo de tráfico se aproxima rápidamente a la capacidad, que es el límite superior de este nivel. En este nivel se producen paradas frecuentes y la velocidad aunque reducida es constante.

El nivel *F* se corresponde con el concepto de *circulación forzada* es el que la velocidad es muy baja y el flujo no llega a la capacidad, careciendo de representatividad debido a las frecuentes colas y colapsos de circulación con frecuentes situaciones de parada y arrancada.

1.3 Detectores de Tráfico

Un detector es –en el ámbito de la ingeniería de tráfico– cualquier dispositivo capaz de registrar y transmitir los cambios que se producen, o los valores que se alcanzan, de una determinada característica del tráfico o de la vía. Existe un gran número de detectores de tráfico dependiendo de la tecnología utilizada y del tipo de medidas que pueden proporcionar [DT, 95]

DETECTORES DE PRESIÓN

Consisten en una plancha de caucho en cuyo interior se sitúan dos láminas metálicas, muy cercanas entre sí, que establecen contacto cuando pasa un vehículo que supera un cierto peso umbral sobre la plancha. Todo este mecanismo está ubicado en la parte superior de una plataforma de hormigón o metálica que se empotra en el pavimento. Puede conseguirse una detección direccional si en vez de las dos láminas se ponen cuatro, enfrentadas dos a dos.

DETECTORES MAGNÉTICOS

Detectan la distorsión del campo magnético producida por el paso sobre ellos de una masa metálica. Están formados por un tubo metálico en cuyo interior hay un núcleo de hierro con una bobina conectada a un amplificador. Los detectores más habituales que emplean esta tecnología no son capaces de detectar la dirección del movimiento, por lo que se fueron incorporando mejoras en su diseño dando lugar a los llamados detectores magnéticos compensados, formados por cuatro núcleos, que permiten distinguir el sentido de la marcha de la masa metálica que circula sobre ellos.

DETECTORES DE LAZO

Constituyen el tipo de detector más utilizado en las vías públicas actuales. Su principio de funcionamiento se basa en emplear las características de un lazo magnético situado sobre la superficie de la carretera y las fluctuaciones eléctricas producidas por la aproximación de un objeto metálico (que en este caso es un vehículo) para detectar su presencia y su paso. Efectivamente, mientras circula una corriente alterna por el lazo metálico situado sobre la carretera, se crea un campo magnético de la misma frecuencia cerca de la superficie de la carretera. Si un objeto metálico entra en este campo magnético, entonces la inducción magnética causa corrientes sobre el objeto metálico y como resultado se produce una variación en la impedancia a la salida del lazo magnético. Cuando se detecta un cambio de impedancia se detecta un vehículo. En este caso se puede decir que el lazo y el vehículo actúan como un transformador, si el lazo es la bobina primaria, la parte metálica del coche constituye la bobina secundaria, la cual es acortada por la resistencia metálica. El bobinado principal se combina con el secundario mediante una inductancia mutua extremadamente baja, que llega a tener el valor cero mientras un vehículo está presente sobre el lazo. Cuando el coche se aproxima al lazo metálico, la inductancia mutua se eleva, provocando una modificación de la impedancia en el bobinado primario del transformador. Este cambio de impedancia se manifiesta como un cambio en la resistencia, inductancia y fase.

Todos estos parámetros se pueden emplear para la detección de vehículos. Sin embargo, la mayor parte de los detectores de lazo que están actualmente en funcionamiento se basan en el cambio de inductancia. La inductancia del lazo también puede ser provocada por cambios de temperatura, lluvia y transformaciones en la superficie de la vía. Tras largos periodos de funcionamiento, los cambios causados por estos factores pueden ser mayores que los causados por la aparición de un vehículo sobre el lazo. Por lo tanto, estos factores deben de ser compensados en el análisis de los datos suministrados por este tipo de detector.

El cambio de inductancia provocado por el paso de vehículos varía según el tipo de vehículo. Este tipo de detectores basados en la medida de la inductancia son más sensitivos a vehículos pequeños que a vehículos de gran volumen. La forma que pueden tener los lazos es muy amplia y cada país tiene sus propias preferencias. Entre las medidas que este tipo de detector de tráfico puede proporcionar destacan las siguientes:

- Presencia de un vehículo.

- Tipo de vehículo (mediante el empleo de técnicas de reconocimiento de patrones es posible diferenciar entre seis o más tipos de vehículos).
- Velocidad del vehículo (mediante el uso de lazos dobles).
- Ocupación de los lazos.
- Intervalo de tiempo entre vehículos.

La siguiente figura, 1.3, muestra la ubicación y utilización de distintos detectores de lazo que realizan distintas funciones en una intersección [Papa, 93]

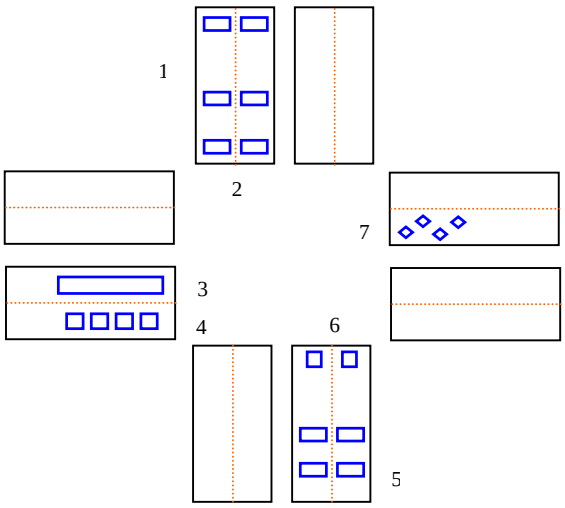


Fig 1-3 *Ejemplos de utilización de detectores de lazo según su ubicación y forma. Los cuatro detectores (1) se utilizan para medir velocidades de paso. Los detectores (2) se utilizan para detectar el número de vehículos que cambian de dirección en el segmento que es entrada a la intersección. El detector (3) se emplea para detectar la presencia de vehículos al igual que los detectores (4). Los detectores (5) se emplean para medir velocidades máximas de paso superiores a la medida por los detectores (1) –debido a la menor distancia que los separa –. Los detectores (6) anotan el paso de vehículos, y los detectores de lazo en forma de diamante (8) señalan la presencia de vehículos.*

Los datos suministrados por este tipo de detectores pueden presentar una serie de inconsistencias o inexactitudes debido a una variedad de factores no asociables a su mal funcionamiento (es decir, a que estos estén estropeados, o exista un fallo en la conexión entre los detectores y los mecanismos que anotan sus medidas o entre estos y el computador central que anota todas las medidas de todos los detectores). Algunas de estas inconsistencias pueden conducir a errores de la siguiente naturaleza [Petty, 96]:

- **Cuentas:** Los lazos pueden no ser suficientemente sensibles a todos los vehículos que pasen sobre ellos y por tanto pueden estar contando menos vehículos que los que realmente pasan. Además, los vehículos que no circulen por el centro del carril pueden no ser contados como sucede ante

vehículos que intentan cambiar de carril en el mismo segmento. Por otra parte, los lazos pueden detectar vehículos que circulen en carriles vecinos y por lo tanto, realizar cuentas de más.

- **Velocidades:** La resolución con la que el lazo recupera la situación inicial después del paso de un vehículo limita la exactitud con la que se pueden medir velocidades de vehículos. Más aún, cuando sólo uno de los dos detectores empleados en la medición de la velocidad del paso de vehículos aporta datos, bien sea esporádicamente o bien de forma permanente, entonces se producen errores que no pueden ser compensados en un análisis posterior.
- **Presencia:** Los umbrales de funcionamiento del detector pueden no estar correctamente ajustados o pueden desajustarse con el tiempo y, por lo tanto, puede afectar a la tarea de detección de presencia.

Por lo tanto, resulta necesaria una tarea de preprocesado de datos que intente minimizar, en la medida de lo posible estos tipos de inconsistencias.

Entre las ventajas de este tipo de detector, y a modo de resumen (dado que este es el tipo de detector que se utiliza más frecuentemente) se pueden apuntar las siguientes [Valdes, 88]:

- Muy sencillos de instalar.
- Pueden hacerse pruebas previas colocando la espira sobre el pavimento.
- El cable que forma el lazo tiene un valor casi despreciable.
- Pueden usarse varios lazos con el mismo amplificador.
- Admiten diversas formas y tamaños.
- Pueden emplearse para la detección de un gran número de características, incluyendo la duración del tiempo de parada si ésta no es muy larga (del orden de 20 minutos como máximo).
- Pueden emplearse para detectar colas de vehículos.
- Se autocompensan para "*ignorar*" la presencia de un vehículo estacionado o parado encima de ellos.

DETECTORES DE RADAR

Constan de un aparato emisor y otro receptor de ondas electromagnéticas y generalmente se suspenden sobre la vía o se colocan lateralmente a ella.

En la actualidad se emplean dos tipos de detectores de radar de microondas en las aplicaciones de gestión del tráfico. El primero transmite energía electromagnética a una frecuencia constante midiendo la velocidad de los vehículos dentro de su campo de visión usando el principio Dopler, en el que la diferencia de frecuencia entre las señales transmitidas y recibidas es proporcional a la velocidad del vehículo. Por lo tanto, la

detección de una variación en la frecuencia denota el paso de un vehículo. Este tipo de sensor no puede detectar vehículos parados y, por lo tanto, no es adecuado para aplicaciones que precisan detectar la presencia de vehículos tales como regulación de semáforos o líneas de parada obligatoria.

El segundo tipo de detector de radar de microondas transmite una onda en forma de diente de sierra, también denominada onda continua modulada en frecuencia, que varía la frecuencia transmitida de forma continua en el tiempo. Los vehículos parados se detectan midiendo el rango desde el detector hasta el vehículo y también calculando la velocidad del vehículo midiendo el tiempo que le lleva al vehículo viajar entre dos marcas internas que representan distancias conocidas para el radar. Al disponer de la característica de detección de vehículos parados, este detector suele denominarse radar de microondas de presencia real.

DETECTORES PASIVOS DE INFRARROJOS

Este tipo de dispositivo es capaz de detectar el paso y la presencia de vehículos pero no su velocidad. Su método de funcionamiento se basa en un detector sensitivo a la energía de fotones colocado en un plano focal para medir la energía infrarroja emitida por los objetos en el campo de visión del detector. Los detectores pasivos no transmiten energía por si mismos. Cuando un vehículo entra en la zona de detección produce un cambio en la energía medida normalmente desde la superficie de la vía en la ausencia de vehículos. El cambio en la energía es proporcional a la temperatura absoluta del vehículo y la emisividad de la superficie metálica del vehículo (la emisividad es el cociente de la energía emitida respecto al radiante perfecto de energía a la misma temperatura). La diferencia de energía que es capaz de detectar este detector se reduce ante condiciones meteorológicas adversas (lluvia, nieve, niebla, ...).

DETECTORES ACTIVOS DE INFRARROJOS

Su funcionamiento es similar al de los detectores de radar por microondas. Los más comunes utilizan un diodo láser para emitir energía en el espectro cercano al infrarrojo, una porción del cual vuelve al receptor del detector desde el vehículo de su campo de visión. Los detectores basados en el radar láser pueden suministrar la presencia, el paso y la velocidad de vehículos. La medición de la velocidad se realiza anotando el tiempo que le lleva a un vehículo cruzar dos haces de infrarrojos que están ubicados a una distancia conocida. Algunos de estos detectores son capaces de clasificar los vehículos contrastando las mediciones con unos ficheros modelos.

DETECTORES ULTRASÓNICOS

Los detectores ultrasónicos emiten sonidos a una frecuencia entre los 25 KHz a los 50 KHz (según sea el fabricante). Estas frecuencias están en la franja audible. Una porción de la energía transmitida se refleja desde la carretera o la superficie del vehículo de nuevo al detector y se procesa para dar el paso y presencia de vehículos. Un detector típico de presencia ultrasónico emite energía ultrasónica en forma de pulsos. El tiempo que le lleva al pulso dejar el detector, chocar contra la superficie y regresar al detector

es proporcional al rango del detector a la superficie. Cuando un vehículo se introduce en su campo de visión se mide el rango desde el detector hasta el vehículo, obteniéndose un rango menor que el producido sobre la vía lo que produce en el detector una señal de detección de vehículo.

DETECTORES ACÚSTICOS PASIVOS

El tráfico de vehículos produce una energía acústica o sonido audible desde una variedad de fuentes dentro del vehículo y desde la interacción de los neumáticos del vehículo con la superficie de la vía. Cuando un vehículo pasa por la zona de detección, el algoritmo de procesado de señales detecta un incremento respecto a la energía del sonido y se genera una señal de presencia de vehículo. Cuando el vehículo abandona la zona de detección, la energía del sonido decrece hasta por debajo de un nivel de detección umbral terminando la señal de presencia de vehículo.

PROCESADORES DE IMÁGENES DE VÍDEO

Estos detectores identifican los vehículos y sus parámetros de flujo de tráfico asociados mediante el análisis de las imágenes suministradas por cámaras de vídeo. Estas imágenes se digitalizan y se analizan para identificar los cambios observables entre imágenes sucesivas, es decir, los cambios de los niveles de contraste entre píxeles adyacentes. Por lo tanto estos detectores pueden suministrar información sobre el paso, presencia, velocidad, longitud y cambios de carriles de vehículos según sea el tipo de técnica de procesado de imágenes utilizada.

1.4 Sistemas de Control de Señales de Tráfico Urbano

Los sistemas de control de señales de tráfico coordinan señales individuales de tráfico, generalmente semáforos, para conseguir unos ciertos objetivos operacionales en la red urbana. Estos sistemas integran los siguientes componentes básicos: intersecciones reguladas por semáforos, una red de comunicaciones para permitir el envío o intercambio de información, y un computador central, o una red de computadoras, para gestionar el funcionamiento de todo el sistema. La tarea de gestionar los valores y duraciones de las distintas señales en las intersecciones se puede implementar de una forma muy variada, siendo los métodos más habituales el uso de técnicas temporales o el de métodos de interconexión hardware. Otra característica a tener en cuenta es que los datos de tráfico son utilizados por varias agencias públicas, por lo que es preciso desarrollar estándares que permitan compartir los datos y el control de las señales de tráfico. Por lo tanto, un componente institucional crítico del control de señales de tráfico consiste en el establecimiento de métodos, bien sean formales o informales, para compartir tanto información de control del tráfico como de las operaciones que se llevan a cabo en cada momento en las intersecciones reguladas por semáforos.

El objetivo fundamental de los sistemas de coordinación de señales principalmente es el proporcionar *acceso*. Un sistema de control de señales no tiene otra finalidad que suministrar unas buenas asignaciones temporales de señales a los vehículos. El sistema

proporciona unas características de funcionamiento que mejoran la capacidad del ingeniero para conseguir este objetivo.

Además del control de señales de tráfico, los sistemas actuales también proporcionan otras funciones adicionales para la detección de tráfico y el uso de imágenes de vídeo. Estos sistemas implementan algoritmos de gestión del tráfico más potentes, permitiendo realizar tareas tales como implementar un control adaptativo e incluso de realizar una previsión sobre el estado del tráfico a corto plazo.

1.4.1 El Concepto de Coordinación

La coordinación entre señales permite controlar los comienzos y duraciones de los periodos de luz verde de los semáforos pertenecientes a intersecciones adyacentes con el propósito de que, en la medida de lo posible, se produzcan el mínimo número posible de interrupciones del flujo de tráfico a lo largo de una ruta o en unas determinadas zonas urbanas.

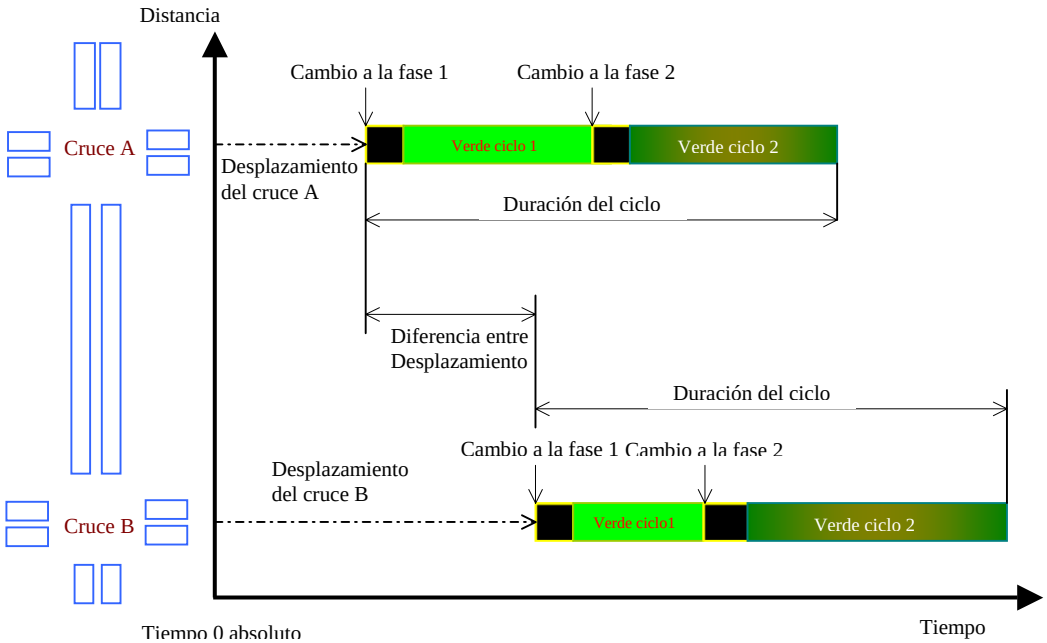


Fig 1-4. *En la figura se observa que para los cruces A y B la duración del ciclo es la misma. Ambos cruces están compuestos de dos fases y tienen un valor distinto de coordinación con respecto al tiempo cero absoluto (el cruce A tiene un valor de coordinación equivalente a "Desplazamiento del cruce A" mientras que el valor para el cruce B es "Desplazamiento del cruce B")*

Para mantener la coordinación entre señales de ciclo a ciclo cada intersección debe de operar con un mismo ciclo de tiempo o con un múltiplo del mismo. Los periodos de luz verde en las intersecciones son desplazados entre sí mediante la especificación de un

tiempo de *desplazamiento*. Esta medida es el tiempo de comienzo del ciclo en cada intersección con respecto al comienzo del ciclo absoluto (figura 1.4)

Estos mecanismos de coordinación entre intersecciones permiten establecer "*ondas verdes*" cuya finalidad consiste en combinar la coordinación entre intersecciones adyacentes para sincronizar los ciclos de luz verde en la misma dirección.

Uno de los primeros métodos empleados para calcular el desplazamiento que se debería establecer para cada una de las intersecciones a lo largo de cada ruta consiste en la generación de su *diagrama de espacio-tiempo*. En este tipo de diagramas el eje de abscisas representa la línea temporal mientras que el eje de ordenadas representa la distancia entre cada uno de los cruces de la ruta. En aquellos puntos que se corresponden con la ubicación de un semáforo se indican con un trazo oscuro los periodos en que el semáforo está en rojo y con un trazo claro los periodos en que el semáforo está en verde. La trayectoria de un vehículo que avanza con velocidad constante estará representada por una línea recta, cuya inclinación variará con la velocidad.

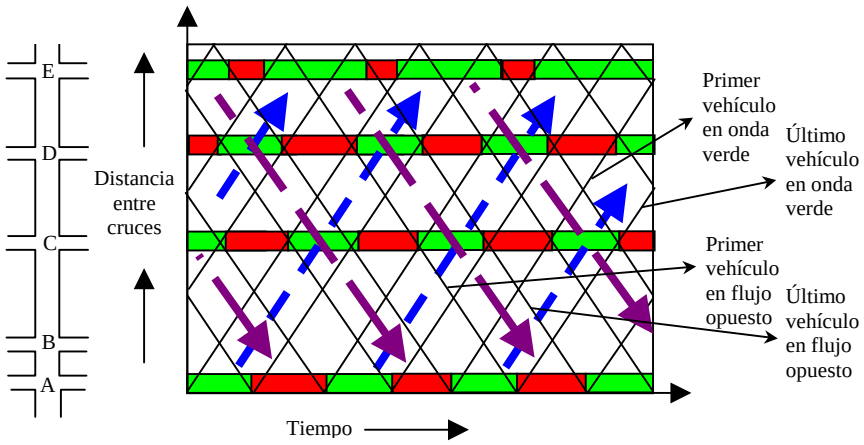


Fig 1-5. *Los rectángulos horizontales verdes representan luz verde en el cruce, los rojos, luz roja (el cruce B no está regulado por semáforos). Las diagonales este-oeste (en azul) representan el flujo de vehículos en el sentido de la onda verde que se quiere establecer; las diagonales oeste-este (en violeta) representan el flujo de vehículos en el sentido contrario al de la onda verde.*

En vez de representar una por una las trayectorias de los vehículos, lo que puede resultar en un diagrama confuso, se suele representar solamente las del primer y del último vehículo. De esta forma, en lugar de una línea, se obtiene una banda (denominada onda verde). El objetivo a conseguir con este diagrama es el intentar implementar la banda más ancha de las posibles. Por esta razón se denomina eficacia de la onda verde a la relación que existe entre su anchura (medida en segundos) y la duración del ciclo. Un ejemplo de este tipo de diagrama se muestra en la figura 1.5.

El establecimiento de los valores adecuados para los desplazamientos mediante el uso de este tipo de diagramas no produce siempre los mejores resultados y cuando se

desean optimizar dos o más flujos de tráfico conflictivos entonces el problema resulta demasiado complicado por lo que resulta necesario aplicar técnicas computacionales.

2.4.2 Identificación de las Áreas Adecuadas para la Coordinación de Señales

Una de las divisiones más habituales en la organización de un sistema CTU consiste en la división en subáreas de la red urbana. Esta división se considera cuando se presentan las siguientes situaciones [RTUA, 87]:

- Se pueden identificar grupos de señales adyacentes que precisan de diferentes estrategias o planes de funcionamiento.
- Normalmente, existe una gran distancia entre grupos de señales.
- Hay establecidas unas vías con una significativa carga de tráfico.
- El espacio disponible para que los vehículos se encolen es escaso en determinadas intersecciones.
- Movimientos de tráfico complejos han de ser ubicados en un área relativamente pequeña.

La coordinación entre las distintas subáreas se puede ajustar para que cumpla unas ciertas demandas.

1.4.3 Técnicas/Estrategias para el Control de Señales

Se establecen fundamentalmente dos tipos de topologías de funcionamiento para los Sistemas de Control de Tráfico Urbano: Sistemas de Control Distribuidos y Sistemas de Control Centralizados. Esta clasificación no es estanca, pues cada vez son más los sistemas que combinan ambos tipos de topologías.

1.4.3.1. Sistemas de Control Distribuidos

Los sistemas distribuidos son aquellos en los que, generalmente, el controlador de la intersección es el responsable de las decisiones de control sobre la misma. Existen distintos tipos de sistemas distribuidos, desde los primeros sistemas de lazo cerrado hasta los más modernos sistemas de gran escala[1].

Los ingenieros de tráfico intentan coordinar las señales de tráfico de un grupo de intersecciones de forma que el flujo de tráfico se comporte de una manera sistematizada y reconocible de una intersección a la siguiente. La coordinación de señales precisa que los valores temporales de los semáforos en las distintas intersecciones se ajusten de forma que se cumplan una serie de objetivos del flujo de tráfico en la red urbana. Esta finalidad requiere que todas las señales funcionen a la misma longitud o una longitud

[1] Las diferencias que se pueden establecer entre los sistemas centralizados y los sistemas distribuidos se han diluido en los sistemas más modernos.

compatible de la duración del ciclo. En el cálculo de estos valores debe de prestarse una atención especial a las relaciones espacio-temporales existentes entre las intersecciones.

En el nivel más básico para realizar esta sincronización, cada intersección regulada por semáforos puede contener un reloj y todas las intersecciones se coordinan según los valores de sus relojes internos. Las señales de las intersecciones se mantendrán coordinadas si lo están sus relojes. Esta forma de coordinación se denomina *coordinación basada en el tiempo* y se emplea generalmente cuando no se dispone de una infraestructura de comunicaciones suficiente para todas las intersecciones bajo control. Las señales se pueden coordinar de una forma más efectiva si se dispone de tal infraestructura de comunicaciones.

Los sistemas distribuidos tienen las siguientes características [TCSH, 96]:

- Se basan en potentes controladores locales. Cada uno de estos controladores se encarga del funcionamiento de una intersección, o de un número pequeño de ellas que estén localmente próximas, mediante el establecimiento de todas las características necesarias para el control de señales en cada intersección.
- Son inherentemente robustos. Si las comunicaciones fallan, o el computador central queda inutilizado, el sistema sigue funcionando debido a que los controladores generalmente funcionan en coordinación basada en el tiempo y, por lo tanto, las comunicaciones se utilizan para mantener ajustados los valores de los relojes internos.
- Son sistemas que pueden ser ampliados sin asumir grandes costes. Cuando se introduce una nueva intersección en el sistema lo único que ha de añadirse es su infraestructura de comunicaciones.
- Bajo coste. Al no precisar de una infraestructura de comunicaciones que permita el intercambio de información en tiempo real la red de comunicaciones necesaria resulta ser más sencilla, y por lo tanto, económica. El ahorro conseguido en el coste de la infraestructura de comunicaciones compensa el alto coste de los controladores locales (entre $10000 y $30000 por intersección).
- Generalmente los sistemas distribuidos no proporcionan la función de observación del tráfico en las intersecciones por medio de cámaras de vídeo (debido al tipo de infraestructura de comunicaciones disponible).
- No disponen de algoritmos de control adaptativos centralizados al no disponer de comunicaciones de control fiables en tiempo real.
- El procesador central en el sistema realiza las funciones de interfaz con el operador.

Los sistemas distribuidos son susceptibles de ser aplicados cuando se cumpla alguna de las siguientes condiciones:

- Se desea instalar un potente control local en cada intersección.

- No existe posibilidad de crear nueva infraestructura de comunicaciones ni de disponer de redes de computadores tolerantes a fallos.
- Se deben de utilizar las infraestructuras actuales.
- El control de señales que se desea implementar tiene pocas características a controlar.
- Deben utilizarse tecnologías de comunicaciones inalámbricas.
- No se precisa de un control ni de una observación en tiempo real.
- No se precisa de la ejecución de un control adaptativo óptimo centralizado.

1.4.3.2. Sistemas de Control Centralizados

En los sistemas centralizados un computador central toma las decisiones de control y dirige de forma individual las acciones a cada uno de los controladores en las intersecciones. Cada intersección, por lo tanto, se compone de un controlador estándar y de un interfaz que, en principio, no precisa de la implementación de ninguna función software.

Los sistemas centralizados tienen las siguientes características [TCSH, 96]:

- Dependen de una red de comunicaciones fiable. Cualquier interrupción en la red de comunicaciones fuerza al controlador local a operar sin el control centralizado en tiempo real y, por lo tanto, les obliga a actuar según su plan de emergencia. En los primeros sistemas centralizados implementados el plan de emergencia consistía en programar en el controlador una actuación aislada de la intersección respecto al resto de intersecciones. Los sistemas más modernos emplean como plan de emergencia un mecanismo de coordinación basada en el tiempo, que reduce el grado de exigencia respecto a la red de comunicaciones. Este es el motivo por el que en las redes de comunicaciones para sistemas centralizados se incluye algún sistema que permite establecer de forma fija e independiente un nivel mínimo de comunicaciones.
- Dependen de computadores centrales fiables. Cuando el computador central queda fuera de servicio el problema afecta a todas las intersecciones de la red urbana que gestiona y no sólo a unas pocas como sucede cuando es la red la que falla. Para evitar esta situación los sistemas centralizados más modernos incluyen un mecanismo de funcionamiento tolerante a fallos.
- No son fácilmente ampliables. Normalmente los sistemas centralizados establecen un tamaño máximo de red que son capaces de gestionar. Incrementar este tamaño máximo provoca grandes costes económicos al tener que introducir modificaciones en todos los componentes del sistema: el computador central, el mecanismo de control y la red de comunicaciones.

- Son caros. Las redes de comunicaciones consumen al menos dos tercios del coste del sistema (entre $40000 y $80000 por intersección)
- Proporcionan un excelente tiempo de respuesta tanto en el envío de comandos como en la observación y supervisión del sistema.
- Permiten el uso de algoritmos adaptativos de control óptimo.

Se puede considerar instalar un sistema centralizado cuando se cumple una o algunas de las siguientes condiciones:

- Se dispone tanto de una buena infraestructura de comunicaciones como de un sistema de redes de computadores centrales tolerantes a fallos.
- Se desea aplicar tecnologías centralizadas óptimas de control adaptativo.
- El sistema general debe de compartir recursos (por ejemplo, la red de comunicaciones) con otros sistemas (por ejemplo, sistemas de gestión de emergencias).

1.4.4 Arquitectura de la Red de Comunicaciones

Una de las características fundamentales para un sistema CTU es su arquitectura de comunicaciones, es decir, la forma en que la información circula entre los distintos componentes del sistema. Cabe destacar los siguientes tipos:

- **Arquitectura Centralizada**. Es representativa del sistema de control centralizado en el que el computador central controla directamente la duración de cada fase. El computador transmite órdenes que se emplean para finalizar la fase actual del controlador.
- **Arquitectura Distribuida con Gestores Locales**. Es representativa de la mayoría de sistemas de lazo cerrado en los que las órdenes de control se transmiten desde los gestores locales a los controladores de las intersecciones. Los gestores locales se comunican con el procesador central cuando ocurre un fallo o cuando así lo ordene el procesador central.
- **Arquitectura Distribuida sin Gestores Locales**. Es similar a la arquitectura centralizada con menores requisitos respecto a la potencia de la red de comunicaciones.

1.4.5 Controles Especiales: Sistemas de prioridad/expulsión

Los sistemas con expulsión anulan la operación del controlador cuando se presentan ciertas condiciones especiales. Los sistemas con prioridad proporcionan la capacidad de asignar una consideración especial a una clase de vehículos sin expulsar completamente la operación actual. Las condiciones especiales que pueden activar el uso de tales características se pueden agrupar en tres categorías:

- Expulsión por paso a nivel con barrera. La operación actual se expulsa como resultado de la proximidad de un tren.

- Expulsión por vehículo de emergencia. Aproximación estática: se asigna verde a toda la ruta (es el mecanismo frecuentemente empleado en sistemas centralizados) o asignación por intersecciones: se asigna verde intersección a intersección, según el vehículo de emergencia va avanzando por la ruta (es el mecanismo más frecuentemente empleado en sistemas distribuidos).
- Expulsión y prioridad por flujo denso. La tendencia actual consiste en utilizar esquemas de prioridad que trabajen conjuntamente con el patrón de coordinación de señales en vez de utilizar esquemas de expulsión para eliminar la señal que está en operación. De este modo se evita que el sistema sea muy sensible a cambios de pequeña importancia respecto al flujo de tráfico.

1.5 Tecnologías Actuales

Para el establecimiento de la coordinación de las distintas intersecciones en una red urbana se aplican fundamentalmente alguna de las siguientes estrategias:

1. *Sistemas de Control de Tiempo Fijo*
2. *Sistemas de Control Adaptativos al Tráfico*: por selección de planes preestablecidos o totalmente reactivos.

Para la correcta selección de la estrategia a adoptar en cualquiera de los sistemas anteriores es preciso disponer de una gran cantidad de datos:

- Referentes a la representación de la red urbana a controlar
- Longitudes de las secciones de calles
- Flujos de tráfico en las intersecciones
- Flujos de saturación de tráfico en cada sección de calle
- Datos sobre las diferentes opciones de duración temporal de cada una de las señales de cada intersección.

La correcta asignación de valores a las señales que constituyen la red urbana depende tanto de la fiabilidad de los datos de entrada al sistema como de la adecuación del modelo de tráfico suministrado al sistema CTU.

Ambos sistemas presentan un gran inconveniente: la gran cantidad de datos que se han de procesar, bien sea para escoger el mejor plan fijo disponible a una situación determinada, o para determinar la mejor asignación libre de valores a señales para minimizar los problemas de tráfico detectados.

1.5.1 Sistemas de Control de Tiempo Fijo.

Un plan de asignación de valores a señales de un grupo de intersecciones es una colección de asignaciones coordinadas a las señales en la red urbana. Cada uno de los planes pueden ser implementados en cualquier momento mediante la emisión del comando correspondiente desde el computador central. Los valores que componen el

plan son determinados por medio de datos históricos, experiencia del ingeniero de la sala de control de tráfico, o bien mediante la utilización de programas de optimización del flujo de tráfico.

La asignación de valores a señales en cada plan es fija en lo referente a la duración del tiempo de verde y de los tiempos de coordinación (desplazamiento) entre las distintas intersecciones. Estos sistemas, por lo tanto, no responden a la demanda de flujo de tráfico sino más bien a conocidos patrones de tráfico.

Un sistema de control de tiempo fijo dispone de planes diferentes para la mañana, tarde, noche, picos de tráfico y fines de semana. Resulta bastante común que existan también planes para condiciones nocturnas y para acontecimientos específicos tales como procesiones o eventos deportivos. Si la red urbana posee un equipamiento de detección de vehículos aislados entonces el sistema puede cambiar su modo de funcionamiento a ser reactivo al paso de estos vehículos (sobre todo en condiciones de flujo de tráfico ligero o bien porque exista algún fallo de conexión entre los reguladores y el computador central de tráfico. Sin embargo, tales sistemas no suelen ser utilizados debido a su escaso beneficio relativo a coste/vehículos implicados.

Los beneficios obtenidos por la implementación de este tipo de planes se van decrementando a medida que las condiciones del tráfico van cambiando y el plan inicial se hace menos apropiado. Se ha estimado [RTUA, 87] que los planes de señales fijos se degradan un 3% anual y, por lo tanto, el beneficio actual de su aplicación puede perderse en 5 años. Además, este tipo de planes tienen una vigencia temporal bastante corta debido tanto al continuo incremento del número de vehículos en circulación como a las constantes modificaciones de la red urbana. Se cumple también, que cuando se implementa por primera vez un plan se produce una apreciable reducción del grado de congestión lo que motiva a otros vehículos a alterar su ruta con el fin de aprovechar aquellas rutas menos congestionadas y por lo tanto alterar la distribución del flujo de tráfico dentro de la red urbana.

Otro de los inconvenientes de este tipo de sistemas consiste en determinar cual es el mejor momento para proceder al cambio de plan. Estos cambios se suelen producir de 4 o 8 veces durante un día laborable. En ciertos sistemas estas decisiones las realizan los operarios de la sala de control de tráfico en respuesta a la información suministrada por lo sensores del sistema (imágenes de vídeo en tiempo real, sensores de cuenta de vehículos,...). No obstante, la situación más habitual consiste en cambiar de plan a determinadas horas fijas según un análisis de datos histórico referente a las condiciones de tráfico en la red urbana.

Uno de los principales modelos desarrollados para realizar el cálculo de los planes de señales disponibles es el sistema TRANSYT.

1.5.1.1 Generación de Planes Fijos de Señales: El Modelo TRANSYT

En el Reino Unido, la técnica que se emplea más frecuentemente [RTUA, 87] para realizar el cálculo de planes de señales consiste en el uso del programa TRANSYT [McDonald, 91], acrónimo de TRAffic Network StudY Tool, desarrollado por Dennis Robertson para el Transport Road Research laboratories en 1967. Este programa modela el comportamiento del tráfico, realiza mecanismos de optimización y predice cuales son

las mejores asignaciones de valores a las señales. Este programa también suministra una información exhaustiva sobre las prestaciones de la red, incluidos los retrasos estimados, el número de paradas, las velocidades medias en rutas origen/destino y el consumo de combustible. La estructura del programa TRANSYT es la indicada en la figura 1.6

El sistema TRANSYT modela el comportamiento del tráfico mediante el uso de histogramas que representan a los patrones de tráfico de llegada. Estos histogramas reciben el nombre de modelos de flujo cíclicos debido a que representan el patrón medio del flujo de tráfico durante un ciclo. Una tarea posterior a la obtención de estos modelos consiste en comprobar que proporcionan una representación lo suficientemente ajustada del comportamiento real del tráfico y, por lo tanto, los parámetros de funcionamiento del sistema se van modificando con el propósito de conseguir este objetivo.

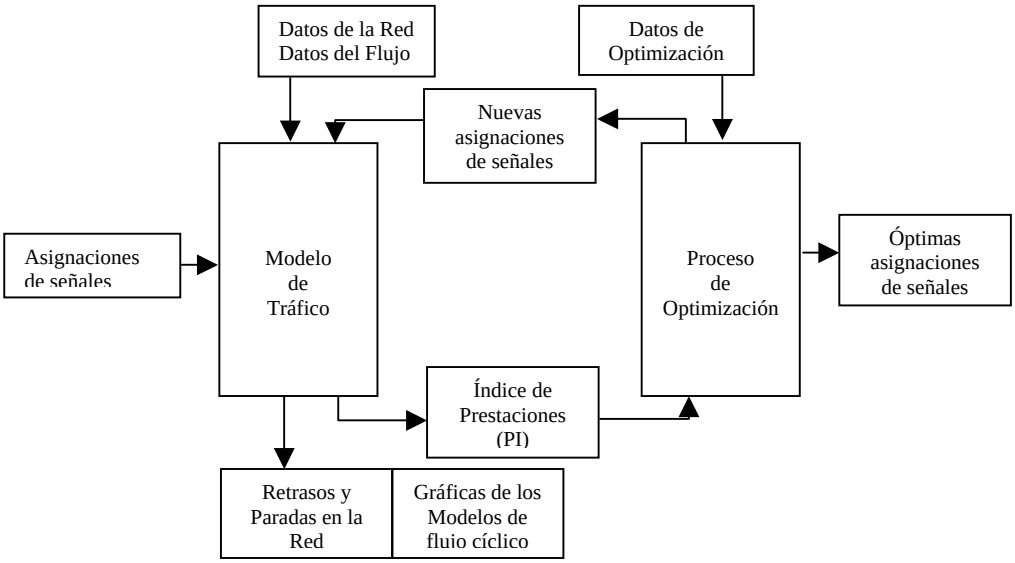


Fig 1-6*. Estructura del programa TRANSYT (TRAffic Network StudY Tool).*

La parte del sistema que se ocupa de la optimización de señales trata de establecer un buen plan fijo de señales que mantenga bajo el nivel de congestión mediante la minimización del siguiente índice de prestaciones (PI) sobre la red:

$$PI = \sum_{i=1}^{n} d_i + kS_i$$

donde

d_i representa el retraso total del cruce i,

S_i es el número total de paradas en el cruce i,

n es el número de intersecciones en la red, y

k es un parámetro definido por el usuario para determinar la importancia de las paradas en relación a los retrasos (es decir, un mayor énfasis en las paradas implica a un mayor énfasis en el tráfico continuo).

Como este índice de prestaciones es la suma de factores “negativos” (es decir, grandes retrasos y un gran número de paradas denotan una pobre asignación de valores a las señales), el sistema trata de minimizarlo.

1.5.1.2 Sistemas de Lazo Cerrado

El sistema de control de lazo cerrado es un sistema distribuido de control de tráfico compuesto de tres niveles. El primer nivel está compuesto por los controladores locales, el segundo por los controladores maestros y en el tercer nivel se ubica el computador central. Una de las características básicas de estos sistemas consiste en que disponen de un mecanismo de comunicación bidireccional tanto entre los controladores locales y los controladores maestros como entre los controladores maestros y el computador central.

Estos sistemas aplican principalmente tres modos de control: control por franja horaria, control manual y control reactivo al tráfico. En el modo de control por franja horaria, la unidad de control selecciona e implementa automáticamente un plan de señales previamente establecido y ejecuta su secuencia de ejecución según sea la franja horaria, día de la semana y/o día del año actual. En el modo de control manual, el operador del sistema especifica el plan de señales a aplicar así como su secuencia de ejecución desde la consola del computador central. Finalmente, en el modo de control reactivo al tráfico, el computador selecciona automáticamente el plan de señales a ejecutar que mejor se adapta a las actuales condiciones del tráfico. Esta selección se realiza por medio de una técnica de contraste de los datos actuales del flujo de tráfico respecto a unos patrones de comportamiento del flujo de tráfico previamente almacenados en el computador central. Esta tarea se realiza de forma periódica cada cinco minutos.

Los sistemas de lazo cerrado constan de seis componentes:

- detectores del sistema
- equipo de control local
- comunicaciones controlador local – controlador maestro
- controlador maestro
- comunicaciones entre el controlador maestro y el computador central
- computador central.

Detectores del sistema

Los detectores del sistema se emplean para detectar el paso de vehículos. Se debe tener un especial cuidado en la ubicación de estos detectores para garantizar que proporcionen las mediciones deseadas. Estas medidas se calculan tomando como referencia el valor o valores suministrado por cada detector o por una combinación de valores asociados a distintos detectores.

Equipo de control local

El soporte físico en el que se instala el controlador local determina en gran medida como va a ser su comportamiento. Por este motivo, se distinguen dos tipos de configuración hardware para los controladores locales: configuración externa y configuración interna. La versión externa integra las funciones de supervisión y de comunicaciones en un soporte distinto que el que almacena el controlador local mientras que la versión interna lo integra todo en un único soporte. Los controladores internos pueden realizar tres funciones: comunicaciones, coordinación basada en el tiempo y modificación de la base de datos del controlador local, lo que les hace altamente dependientes del fabricante del controlador, mientras que los controladores externos, al no disponer de estas funciones, permiten que se conecten cualesquiera controladores disponibles en el mercado (del mismo o de distinto fabricante).

Comunicaciones controlador local – controlador maestro

El controlador maestro debería, en un sistema ideal, acceder una vez por segundo a cada controlador local. Los datos transmitidos suelen ser órdenes para cambiar de un plan a otro plan determinado de señales, o para realizar cambios en la base de datos de un controlador específico. Existen tres categorías de mensajes de retorno: datos de estado, datos de señales y datos de detectores.

Controlador maestro

La función principal de un controlador maestro consiste en seleccionar los planes de señales a implementar para cada grupo de intersecciones, procesar y almacenar información de los detectores y supervisar la operación del equipo

Comunicaciones entre el controlador maestro y el computador central

Suelen utilizarse redes de telefonía para realizar la comunicación entre el controlador maestro y el computador central, normalmente una línea para cada controlador maestro. El contenido de la información que se gestiona en esta comunicación pertenece a uno de los tres tipos siguientes: sincronización de tiempo, órdenes del sistema o transmisión de datos.

Computador central

El computador central se emplea para:

- Asignación hora y fecha
- Observación y seguimiento del estado de alguna intersección
- Modificación de la base de datos del controlador maestro: Los datos se organizan en cinco categorías: el planificador de eventos, parámetros

reactivos al tráfico, parámetros de procesamiento de los detectores, resumen de datos y resumen de eventos acontecidos.

- Modificación de los datos del controlador y de coordinación.
- Modificación de los parámetros del sistema.
- Monitorización del sistema
- Realización de informes.

Uno de los inconvenientes de estos sistemas es su incapacidad para gestionar de una forma unificada y eficiente intersecciones que estén conectadas a diferentes controladores maestros.

UN CASO PRÁCTICO: EL SISTEMA MATS[2] (PEEK TRAFFIC – TRANSYT CORPORATION)

El sistema centralizado de lazo cerrado MATS es un sistema comercializado por la empresa Peek Traffic especialmente diseñado para cubrir el espacio existente entre los sistemas de lazo cerrado pequeños y los potentes sistemas de control centralizados. El sistema proporciona tres niveles de operación: central, lazo cerrado local y coordinación local basada en el tiempo. El modo de operación central proporciona al sistema una amplia coordinación que no está limitada a las conexiones físicas a los controladores maestros. El sistema funciona como un sistema de lazo cerrado cuando así se disponga o cuando la red de comunicaciones con el computador central esté inactiva. Los controladores locales actuarán en coordinación basada en el tiempo cuando ocurra una pérdida de comunicaciones entre el controlador local y el controlador maestro.

El sistema MATS permite almacenar hasta 420 patrones de comportamiento del tráfico, y su selección se puede realizar según sea el volumen de tráfico, la ocupación de las vías urbanas, una medida ponderada entre volumen y ocupación, la concentración de tráfico, su densidad, velocidad o colas de vehículos que se produzcan en las inmediaciones de las intersecciones.

1.5.2 Sistemas de Control de Señales Adaptativos al Tráfico

Los sistemas de control de tráfico adaptativo (CTA) difieren de los sistemas de control de planes fijos en que los tiempos para las señales se adaptan a un corto periodo de tiempo con el fin de ajustarse a las necesidades de los volúmenes y patrones del flujo de tráfico actual. Esta capacidad produce una serie de beneficios tales como la reducción en el tiempo de trayectos, número de paradas, consumo de combustible y contaminación atmosférica. Además, debido a que los sistemas CTA reaccionan a las condiciones de tráfico en tiempo real, los planes de señales temporales están permanentemente actualizados y, por lo tanto, no quedan obsoletos. Los ejemplos más representativos de este tipo de sistemas son los modelos SCOOT, SCATS y los desarrollados en RT-TRACS.

[2] http://www.peek-traffic.com

1.5.2.1 El Sistema SCOOT (Split, Cycle & Offset Optimization Technique)

El sistema SCOOT (Técnica de Optimización de los valores de Desplazamiento, Ciclo y Giro) [SCOOT, 98] se fundamenta en la experiencia aportada en la definición, desarrollo e implementación de la Herramienta para el Estudio de la Red de Tráfico TRANSYT. El objetivo del sistema, tal como se indica en su nombre, consiste en realizar una optimización a tres niveles: *giro*, *ciclo* y *desplazamiento*.

El sistema SCOOT está instalado y controlando tanto configuraciones de vías principales como mallas de calles en diversas ciudades del mundo, entre éstas las más relevantes son: Toronto (vías principales – Lake Shore Boulevard- y mallas –distrito comercial centro), Madrid (malla representando el área centro) y Londres (regulando la vía principal Cromwell Road) [SCOOT, 98].

Para poder llevar a cabo esta optimización el sistema SCOOT toma como datos de partida los datos suministrados por detectores de tráfico, generalmente detectores de lazo enterrados en el pavimento, situados a una distancia de ocho segundos de tiempo de viaje de la línea de parada de la intersección.

El programa central predice segundo a segundo la secuencia de llegadas a la intersección mediante la construcción de un modelo de flujo de demanda a la intersección basándose en los datos medidos por cada detector. Esta secuencia se contrasta con la secuencia de salida basada en la saturación de la ocupación desde semáforo en rojo (vehículos encolados) hasta el despeje de la cola de vehículos (situación que sucede con semáforo en verde). La diferencia entre los modelos de salida de la intersección y de demanda a la misma (la entrada medida por el detector) representa a los vehículos retrasados en la cola. Estos modelos, denominados *link profile units*, representan medidas que relacionan ocupación de los detectores y huecos en el flujo de tráfico, y su determinación es uno de los secretos de funcionamiento del sistema SCOOT.

Los ajustes temporales implementados en SCOOT son pequeños con la intención de evitar sobre-reaccionar a picos en el flujo de tráfico (y por lo tanto el sistema se comporta de forma poco sensible a cambios esporádicos en el flujo), pero estos ajustes son realizados frecuentemente por lo que, si las variaciones de flujo persisten en el tiempo, el sistema se va adaptando paulatinamente a las condiciones reales del tráfico.

El optimizador de giro evalúa cada segundo los modelos de salida y de demanda en cada intersección. Cinco segundos antes de que se produzca un cambio de fase durante el ciclo actual, el sistema SCOOT analiza si añade un retraso o provoca un adelanto a las señales que van a comenzar o que están finalizando. Aquella asignación de valores que proporcione la mejor relación posible respecto al retraso para todos los movimientos posibles en la fase actual será implementada. La evaluación de esta relación se realiza teniendo en cuenta las preferencias y restricciones establecidas por el ingeniero de la sala de control de tráfico (por ejemplo, el establecimiento de unas duraciones máximas y mínimas para cada fase o bien establecer una duración fija para cada fase).

Al comienzo de cada ciclo el optimizador de desplazamiento calcula el retraso asociado a paradas y congestiones para todos los movimientos de la intersección tomando como referencia los modelos obtenidos en el ciclo anterior. El optimizador de desplazamiento compara esta situación con las que se podrían dar ante un comienzo del ciclo cuatro segundos antes o después, implementando aquella situación que proporcione un retraso mínimo.

El optimizador de ciclo observa los niveles de saturación de todos los movimientos de las intersecciones una vez en cada periodo de control del ciclo (cada 2'5 ó 5 minutos). La intersección crítica será aquella que presente un mayor nivel de saturación. Si la saturación en la intersección en los movimientos más densos excede el 90%, entonces el controlador de ciclo añadirá 4, 8 o 16 segundos al ciclo dependiendo de cual sea la longitud del ciclo actual (4 segundos para los ciclos más cortos y 16 para los más largos). Si la saturación es menor del 90% entonces la duración del ciclo se mantiene constante o bien se reduce.

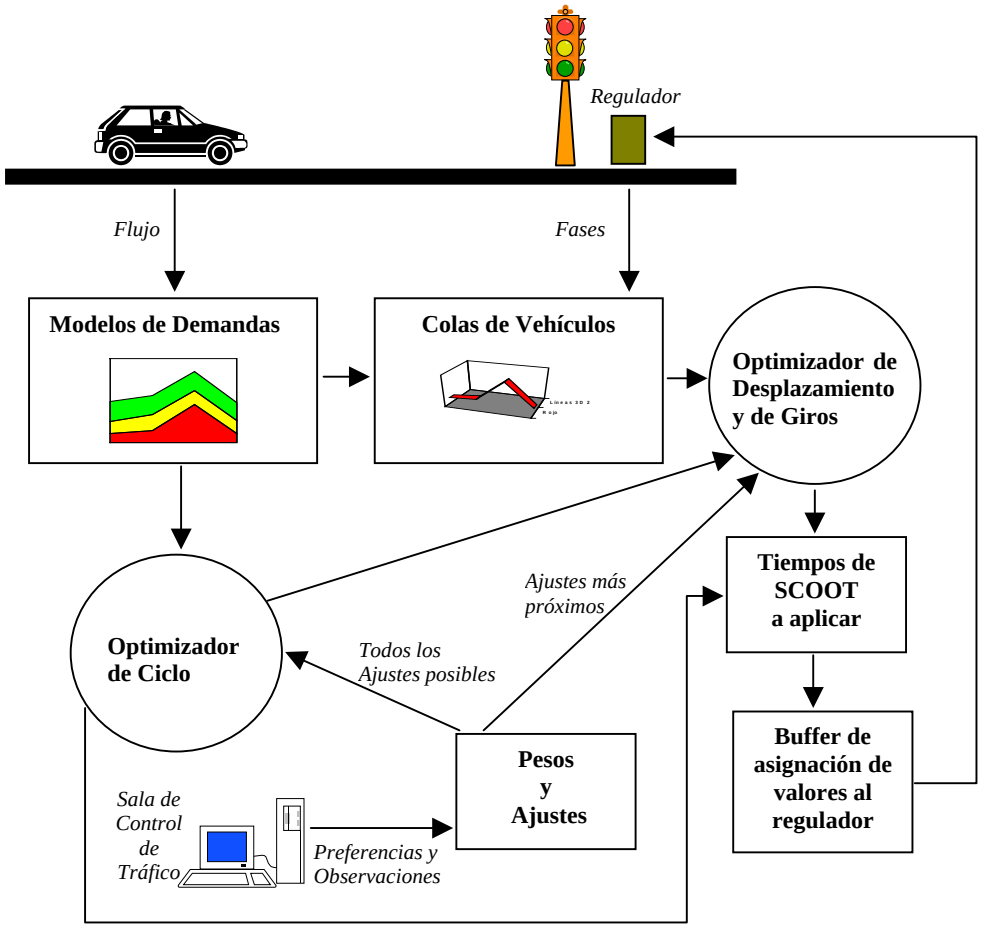


Fig 1-7. *Esquema general de funcionamiento del sistema SCOOT*

La implementación de un sistema SCOOT precisa del calibrado de un amplio rango de parámetros para cada movimiento de tráfico en la red para asegurar que el modelo SCOOT funciona adecuadamente. El sistema SCOOT no proporciona una optimización

de la secuencia de fase, es decir, no puede alterar el orden en que se suceden las fases ni la composición de sus movimientos permitidos, aunque si que se permite cambiar la duración temporal de la fase actual o bien, de forma general, el cambio de fase mediante la asignación a la intersección de un nuevo plan. El optimizador de desplazamiento puede variar en cada ciclo. Además de estas funciones el sistema SCOOT dispone de planes adicionales para implementar la prioridad de tránsito, y la prioridad para vehículos de emergencia.

SCOOT es un algoritmo centralizado basado en la comunicación de órdenes una vez por segundo con el controlador local de intersecciones. En este controlador local no se realiza ninguna medida autónoma de optimización, por lo que la optimización que proponga e implemente el sistema SCOOT depende totalmente de la fiabilidad tanto de la red de comunicaciones como del computador central.

1.5.2.2 El Sistema SCATS (Sydney Coordinated Area Traffic System)

Los objetivos fundamentales del sistema SCATS [SCATS, 98] son la reducción del número de paradas debido a una mala regulación de las señales de los semáforos en las intersecciones, los retrasos provocados al flujo de tráfico denso y la optimización del tiempo necesario para realizar una serie de rutas origen-destino previamente establecidas.

El sistema SCATS está instalado y controlando cerca de 5000 intersecciones en diversas ciudades del mundo, entre estas las más relevantes son: Sydney (en la que se regula cerca de 2000 intersecciones), Melbourne (regulando 2000 intersecciones) y Hong Kong (regulando aproximadamente 600 intersecciones) [SCATS, 98].

El sistema SCATS es un sistema distribuido jerárquico en tres niveles. La arquitectura del sistema se compone de un computador central para realizar la monitorización del sistema global, computadores regionales remotos o locales y controladores locales de señales de tráfico.

El computador central permite el acceso a los computadores regionales para la recogida de datos de tráfico, datos de entrada y acciones de monitorización. Es el componente encargado de realizar las siguientes tareas sin influir en la operación de gestión del tráfico:

- Mostrar el estado del tráfico y del equipo para la posible corrección de errores.
- Producir y almacenar datos de tráfico.
- Mantener una copia de los datos de cada computador regional.
- Establecer un mecanismo de control central para monitorizar el sistema (subsistema o intersección), modificar los parámetros de control, redefinir manualmente funciones dinámicas o realizar diagramas de espacio-tiempo.

Cada computador regional puede gestionar hasta 128 controladores locales y su principal función consiste en controlar de forma autónoma las intersecciones de su área mediante el análisis de la información de los datos de los detectores preprocesados por

medio de microprocesadores locales, implementando, por lo tanto, la estrategia de control sobre esa región. La arquitectura de comunicaciones utilizada en el sistema SCATS se muestra en la figura 1.8.

Las variables empleadas para realizar la estrategia de control sobre una región calculan el tiempo que tarda en recorrer un determinado espacio un vehículo según sea el estado del flujo de tráfico actual (*tiempo de espacio*) y el tiempo total disponible de ese espacio asociado a verde en cada fase (*tiempo total de espacio*).

Estas medidas se emplean para computar para cada intersección los valores asociados a los conceptos de "*grado de saturación*" (*GS*) y "*flujo equivalente de vehículos*" (*VK*). Estos valores se calculan como la media aritmética de cada uno de ellos durante tres ciclos consecutivos.

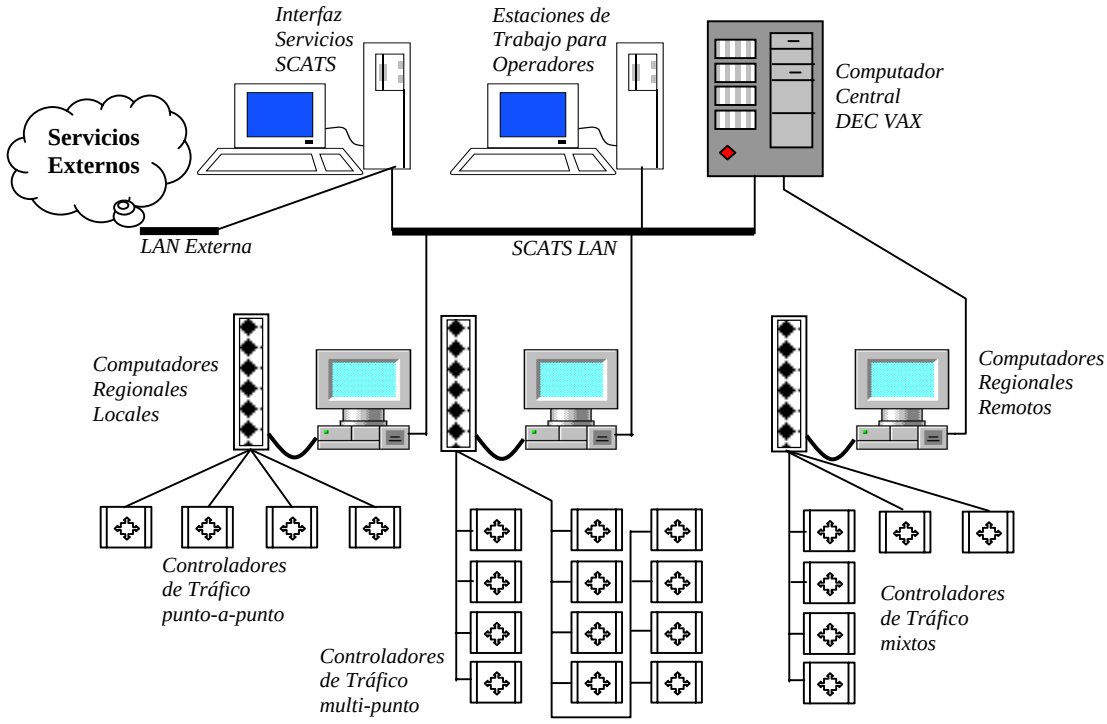


Fig 1-8. *Arquitectura del sistema SCATS. Cada computador regional puede gestionar hasta 128 controladores de tráfico mediante líneas de conexión punto-a-punto, multi-punto o conexiones mixtas. El computador central puede gestionar hasta 32 computadores regionales. El sistema SCATS también proporciona un interfaz de comunicación con otros sistemas externos para permitir el intercambio de datos entre distintas agencias.*

El "*grado de saturación*" es una medida de eficiencia entre el tiempo empleado en la fase respecto a su tiempo disponible. Para calcular esta medida se utiliza una fórmula en la que se incluyen los datos asociados al concepto de tiempo de "verde no utilizado" (tiempo total de espacio menos el número de tiempos de espacio que se pueden realizar

en un tiempo de espacio estándar asociado al valor de flujo máximo de vehículos) y al tiempo de verde disponible (tiempo de verde durante la fase en la que se recogen datos por los detectores): *GS = [verde – (verde no utilizado)] / verde disponible*. Un valor nulo del GS indica que se obtiene el valor asociado al flujo de saturación, un valor positivo que se está en un régimen inferior al de saturación y un valor negativo que se está en un régimen superior al de saturación). Se observa que el GS puede ser superior al 100% en el caso que el tiempo de espacio sea mayor que el tiempo estándar asociado al flujo máximo.

El valor de GS se utiliza para seleccionar la longitud del ciclo y el plan de giro a implementar. El efecto de los posibles valores de la duración del ciclo respecto al retraso producido en el tráfico debido a su ejecución responde a los valores indicados en la siguiente gráfica (obtenida mediante resultados experimentales):

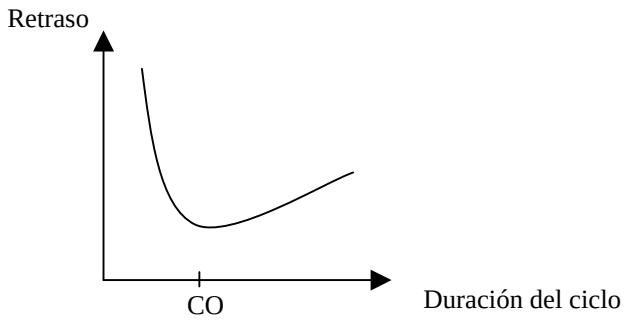


Fig 1-9. *Relación entre los valores de duración de ciclo y retraso asociado*

El valor asociado al retraso se incrementa rápidamente para duraciones de ciclo menores que la duración del ciclo óptimo (CO), mientras que para valores de ciclo superiores al óptimo este incremento se realiza más lentamente. La duración del ciclo que se establece en el sistema SCATS se determina tomando como referencia el valor máximo del GS en el subsistema para el que se está calculando. El método se basa en determinar la duración del ciclo que debería aplicarse según sean los valores del grado de saturación medidos. Esta medida se compara con la última duración del ciclo realizada. La diferencia entre el ciclo realizado y el medido nos proporciona una medida intermedia que se calculará para cada tres ciclos consecutivos. El valor final al que tendría que converger la duración del ciclo se calcula realizando la media ponderada de estas tres medidas intermedias. La duración del ciclo se modificará en ± 6 segundos hacia este valor final, excepto cuando las medidas intermedias para los últimos dos ciclos fuesen mayores a 6 segundos, en cuyo caso se permite un cambio de hasta 9 segundos. Estas suaves variaciones en la longitud del ciclo permiten al sistema irse adaptando paulatinamente al comportamiento persistente del estado del tráfico evitando actuar sobre picos esporádicos del mismo.

El GS también se utiliza para seleccionar el plan de giro. Para ello se examinan los planes posibles de giro en cada ciclo calculando su GS proyectado máximo. Aquel plan de giro que presente el menor máximo del grado de saturación será el seleccionado.

El "*flujo equivalente de vehículos*" es una medida derivada del grado de saturación y del flujo de saturación por carril en cada segmento. Para calcular esta medida se emplea la siguiente fórmula: *VK = GS* x *Tiempo de verde* x *vehículos/s asociados al flujo máximo*. Es una medida independiente del tipo de vehículos del que se compone el tráfico y se utiliza para seleccionar el plan de desplazamiento. Para ello se especifican para cada segmento unos valores de sentido de dirección, posteriormente se multiplican estos valores por los volúmenes ponderados de *VK* y el resultado se añade a cada plan. El plan que resulte con un mayor valor recibe un voto. Cuando 4 de los últimos 5 votos han sido para un mismo plan de desplazamiento entonces este plan es seleccionado para su aplicación.

La calibración de estas medidas durante años de experiencia es la clave para conseguir unas prestaciones efectivas.

Todas las intersecciones pertenecientes a un mismo subsistema operan a la misma duración del ciclo y se coordinan mediante desplazamientos. También se permite compartir la duración del ciclo entre distintos subsistemas mediante el uso de un conjunto separado de valores de desplazamientos

La comunicación entre los controladores locales y el computador regional se realiza por medio de líneas dedicadas punto a punto o bien por medio de una única línea a la que se conectan varios controladores de un mismo computador regional. El controlador local procesa los datos recogidos por los detectores de tráfico, realiza decisiones tácticas respecto a la operación de las señales en la intersección e incorpora un mecanismo de funcionamiento de emergencia para el caso de que se produzcan anomalías en la red de comunicaciones o en su computador regional asociado. Las variables empleadas para realizar el control táctico calculan la fase actual (bloqueándola o no) y el tiempo restante para la terminación de la fase.

Los sensores de tráfico utilizados en el sistema SCATS son detectores de lazo o bien, cámaras de vídeo ubicados en la línea de parada de cada carril en cada segmento y son empleados para computar medidas referentes a la longitud de las colas de vehículos en las inmediaciones de las intersecciones, la discriminación del giro en segmentos con varios carriles, o el porcentaje de giro del volumen de tráfico en las intersecciones.

El sistema SCATS permite establecer hasta 5 clases de prioridad, una para trenes y cuatro para vehículos. También dispone de un sistema que permite realizar un control automático de los valores de las señales de tráfico sobre una ruta de emergencia previamente establecida.

1.5.2.3 El Sistema RT-TRACS

El sistema de control adaptativo de tráfico en tiempo real RT-TRACS [RT-TRACS, 98] es un proyecto para desarrollar y evaluar sistemas de control de señales de tráfico adaptativo en tiempo real similares a los sistemas SCOOT y SCATS. Estos sistemas proporcionan cuatro niveles de control: sistema, distrito, sección e intersección. Cada

nivel debería ser capaz de conseguir un funcionamiento óptimo según las restricciones impuestas por los niveles superiores. El *nivel de sistema* es el nivel superior y contiene todas las intersecciones conectadas en la red. Este nivel proporcionará los medios para calcular las funciones de control globales. El siguiente nivel es el *nivel de distrito* asignado a distritos geográficos y las intersecciones que contiene de forma que estas no se compartan entre distintos distritos. El *nivel de sección* consiste en agrupaciones lógicas de intersecciones en las que una misma intersección puede pertenecer a diferentes secciones. El nivel más bajo es el *nivel de intersección*.

El sistema RT-TRACS se compone de cinco componentes básicos: el primero predice las condiciones del tráfico según las condiciones atmosféricas, incidentes, y factores de tráfico; la segunda componente define las secciones en la red; la tercera componente selecciona la estrategia de control adecuada para cada sección según sean las condiciones de tráfico que se han previsto; el cuarto componente implementa esta estrategia y el último componente la evalúa. El sistema RT-TRACS implementa tres estrategias de control:

- Estrategias de control de primera generación: usan un catálogo de planes temporales fijos, previamente establecidos, para controlar una red de intersecciones.
- Estrategias de control de segunda generación: usan planes temporales que se generan en tiempo real según sean las predicciones realizadas basándose en los datos proporcionados por los detectores.
- Estrategias de control de tercer nivel: ajustan los tiempos de duración de las señales en cada intersección según sean los valores de los datos detectados en tiempo real.

1.6 Comparativa entre los Tipos de Sistemas CTU

Los sistemas de control de señales de tráfico utilizados más comúnmente en las ciudades funcionan según un conjunto predeterminado de planes temporales de señales almacenados en el computador central, es decir, son sistemas de control de tiempo fijo (también conocidos como *sistemas de primera generación*). Los planes son desarrollados por personal especializado según mediciones realizadas manualmente. Posteriormente, a estas medidas se les aplica un proceso intensivo de análisis de señales de tráfico (por ejemplo, mediante el sistema TRANSYT) y, como resultado, se obtienen los planes temporales de señales que finalmente son introducidos en la base de datos del computador central.

A pesar de que este método es sencillo y razonablemente económico, los planes de señales temporales adolecen de la capacidad de adaptación ante variaciones en las condiciones del tráfico. Recientes resultados de la investigación sobre ingeniería de tráfico [Green, 98] confirman que la efectividad de los planes de señales temporales fijos se degradan en un mínimo de un 6% o más debido tanto a las modificaciones de los patrones de comportamiento del flujo de tráfico como a la variabilidad respecto al volumen de tráfico. Si estos planes de señales fijos no se revisan de una forma

periódica, el impacto sobre las condiciones del tráfico puede irse deteriorándose progresivamente en años sucesivos.

Tipo de Sistema CTU	Ventajas	Desventajas
Tiempo Fijo	• Más económico de instalar y de mantener. • Puede implementarse sin utilizar un sistema de control centralizado. • Facilidad en el ajuste de "ondas verdes". • Facilidad en la gestión de vehículos prioritarios.	• Necesita de enormes cantidades de datos que deben actualizarse de forma continua. • Precisa de la constante actualización de los planes de señales. • Precisa de la interacción del operador ante la presencia de incidentes. • No puede reaccionar ante fluctuaciones a corto plazo.
Selección Adaptativa del Plan	• Pueden gestionar diariamente las pequeñas fluctuaciones en el flujo. • La selección de la hora de implementación del plan es más flexible. • Requiere menos detectores que los sistemas completamente adaptativos al tráfico.	• Precisa de tantos o más datos que el sistema de tiempo fijo. • Posible fallo en las mediciones de los detectores. • Dificultad en la previsión de las necesidades para todos los planes. • El cambio de plan puede no ser aceptado, dependiendo del valor umbral que se haya establecido.
Sistemas Completamente Adaptativos	• Precisa de la toma de una menor cantidad de datos que los otros sistemas. • Se evitan los problemas asociados a la actualización de los planes y a su proceso de selección. • Gestionan las fluctuaciones de tráfico a corto y a largo plazo. • Reaccionan de forma automática ante incidentes. • Proporcionan una monitorización de todo el sistema.	• Precisa de un gran desembolso económico para su instalación y mantenimiento. • Precisa de un mayor número de detectores. • Muy dependiente de la calidad de las medidas proporcionadas por los detectores. • Su mantenimiento resulta una tarea crítica.

Tabla 1.1 *Tabla resumen comparación sistemas control de tráfico urbano.*

Sin embargo, la actualización de todos los planes de señales es muy costosa, por ejemplo, actualizar todos los planes temporales de señales de forma anual en el área metropolitana de Toronto, que contiene unas 1760 intersecciones reguladas por semáforos, mediante la toma de una única medición por cada intersección, precisaría de

los servicios de 30 personas en dedicación exclusiva al año [Green, 98]. Un resumen de las ventajas y desventajas de cada uno de los tipos de sistemas se expone en la tabla 1.1.

Debido a las evidentes ventajas que proporcionan los sistemas de control de tipo adaptativo (ya sean completamente adaptativos o por selección del mejor plan) el número de implementaciones de tales sistemas crece constantemente. Un reciente estudio [Green, 98] sobre el impacto de una implementación del sistema SCOOT respecto a un sistema de planes fijos en la ciudad de Toronto (Canadá), proporciona unos resultados interesantes. Para realizar este estudio se seleccionaron 75 intersecciones divididas en tres áreas complejas operacionalmente distintas: Distrito Comercial Central de Toronto (una malla de 42 intersecciones), el boulevar del lago Shore (ruta de acceso limitado consistente de 20 intersecciones) y la calle Yonge (red arterial comercial de acceso no limitado de 13 intersecciones, con enlace a una autopista). Para implementar el sistema se precisó instalar 350 detectores de lazo cerrado.

El sistema SCOOT se comparó respecto a una mezcla de planes antiguos y actualizados diariamente. La mayor parte de las señales temporales consistían de tres planes que cubrían los picos del mediodía, los picos de la tarde y el resto del día. El sistema se evalúo durante 6 semanas intercambiando ambos métodos en días consecutivos e incluso durante el mismo día.

Con el sistema SCOOT se consiguieron mejoras en prestaciones respecto a los tiempos de trayecto de vehículos, paradas y retrasos, así como en la seguridad de tráfico para los motoristas y unas estimables reducciones en impactos de salud pública asociadas a la reducción del consumo de combustible y de los niveles de contaminación. Como contrapartida, se obtuvieron leves incrementos en los retrasos asociados a los peatones, así como en ciertos trayectos que contenían paradas no controladas por semáforos (pasos de peatones, señales de stop, ..), pero todos estos inconvenientes podrían ser fácilmente resueltos mediante el ajuste adecuado de los parámetros operacionales del sistema SCOOT para estas calles (es decir, realizando un ajuste fino de los mismos).

Los sistemas CTU, como se observa en la exposición realizada, evolucionan muy rápidamente, incorporando paulatina y sistemáticamente nuevas funcionalidades y prestaciones en cada nueva versión. Entre las nuevas características que, a corto y medio plazo, estos sistemas van a proporcionar pueden destacarse las siguientes: la interconexión de las señales de tráfico mediante comunicaciones por radio de amplio espectro, el control de los valores de las señales en tiempo real de forma reactiva al tráfico, el mantenimiento de una base de datos con información común y una integración más intensa entre los sistemas de monitorización y de control.

CAPÍTULO 2

SISTEMAS BASADOS EN EL CONOCIMIENTO APLICADOS AL CONTROL DE TRÁFICO URBANO

Índice del capítulo 2

2.1 Introducción

La enorme incidencia, tanto en aspectos económicos como sociales, que para un entorno urbano presenta el establecimiento y correcto funcionamiento de un sistema de control de tráfico urbano (CTU) ha provocado que, durante las últimas décadas, se estén dedicando una gran cantidad de recursos en investigación aplicada a tales sistemas. Este importante interés explícito en la mejora de sistemas CTU es compartido entre un gran número de países de nuestro entorno debido, fundamentalmente, a que los problemas que afectan al tráfico urbano –niveles crecientes de congestión y de degradación ambiental, altas tasas de accidentes e importantes incrementos en el consumo de energía- son comunes en sociedades que han alcanzado un cierto grado de desarrollo económico.

Además de los beneficios apuntados, el objetivo de obtener un mejor control de la infraestructura urbana disponible tiene otros muchos beneficios explícitos. Por ejemplo, al conseguir una mejor utilización de la red existente, se evita o suaviza la necesidad de construcción de nuevas infraestructuras de transporte. Además, como ya se ha indicado anteriormente, al mejorar la calidad del tráfico se consigue una reducción en los tiempos de realización de trayectos urbanos, lo que resulta en un flujo de tráfico más continuo, reduciéndose, por lo tanto, la contaminación acústica, la contaminación ambiental y el consumo de combustible. No obstante, la mejora en los servicios suministrados por los sistemas CTU tienen que realizarse de una forma continua, puesto que al variar la oferta se produce como efecto asociado una mayor demanda por parte de los usuarios. Por lo tanto, no existen soluciones óptimas permanentes, sino que éstas han de ser actualizadas permanentemente.

Como resultado de estas fuertes inversiones realizadas en la investigación de sistemas CTU se han producido un gran número de prototipos y de sistemas aplicados cuyo proceso de análisis y puesta a punto han contribuido tanto a una mejor comprensión del problema del tráfico urbano como a un aporte de las tendencias más prometedoras a las empresas dedicadas a la construcción de tales sistemas. El constante abaratamiento en el coste de los sistemas hardware y el gran incremento en prestaciones desarrollado en los últimos años han permitido disponer de un equipo más asequible y potente con el que poder afrontar la búsqueda de soluciones para problemas de tráfico más complejos.

Iniciativas estratégicas tales como DRIVE (1989-1991, 1992-1994) (*D*edicated *R*oad *I*nfraestructure for *V*ehicle safety in *E*urope), PROMETHEUS (*PRO*gra*M*me for *E*uropean *T*raffic with a *H*ighest *E*fficiency and *U*nprecedented *S*afety) y, más recientemente, SMARTEST (*S*imulation *M*odelling *A*pplied to *R*oad *T*ransport *E*uropean *S*cheme *T*ests) en Europa, IVHS (*I*ntelligent *V*ehicle-*H*ighway *S*ystems) en Estados Unidos junto con los programas RACS y VICS (*V*ehicle *I*nformation and *C*ommunication *S*ystem center) en Japón son una muestra de los principales proyectos de investigación financiados por instituciones nacionales o supranacionales. Todas estas iniciativas son sólo una muestra de una gran cantidad de proyectos de investigación ya

finalizados o en fase de ejecución. Basta con visitar la página WEB de la Unión Europea correspondiente al directorio general DGVII referente a la tecnología del transporte (http://www.cordis.lu/transport/home.html) como muestra de la importancia concedida al desarrollo e investigación en tales sistemas en el contexto europeo.

Todos estas iniciativas apuntadas comparten el interés y uso de las denominadas *tecnologías de la información* y de las *telecomunicaciones* aplicadas al problema del transporte. Dentro de esta amplia área de investigación y desarrollo, distintas técnicas de Inteligencia Artificial y de Sistemas Basados en el Conocimiento han sido, y siguen siendo, sucesivamente aplicadas, con el fin de ayudar al proceso de toma de decisiones en la regulación del tráfico. Los sistemas obtenidos mejoran las prestaciones de los actuales sistemas CTU en uso y, además, introducen nuevas características adicionales que permiten mejorar la calidad global del sistema.

2.2 Problemas Asociados a los Actuales Sistemas CTU

Los sistemas que actualmente se están aplicando en el control de tráfico urbano difieren en varios aspectos:

- La organización espacial de los componentes sobre los que se realiza el control de la aplicación, pudiendo esta responder a un esquema de control centralizado o distribuido.

- Los distintos enfoques disponibles en los que se puede realizar el control del sistema. Por ejemplo, los tres enfoques más comunes para realizar la coordinación de señales son: la coordinación aislada de las intersecciones, la coordinación de las intersecciones que forman parte de vías de importancia relevante (en terminología de tráfico urbano estas vías se denominan *arterias principales*) y la coordinación por áreas urbanas (por ejemplo, una malla urbana correspondiente a una zona comercial).

- Los distintos modos en los que se puede gestionar la regulación del flujo de tráfico. Como ya se ha apuntado en el capítulo anterior, esta gestión puede realizarse bien sea por la selección y posterior aplicación de planes de tráfico previamente calculados, o por medio de su continua construcción atendiendo a la demanda de flujo de tráfico detectada.

- La gran cantidad de combinaciones posibles de políticas y criterios de control a adoptar en el sistema para poder alcanzar unos objetivos operacionales adecuados. Por ejemplo, la asignación de distintos grados de prioridad para establecer cual va a ser el orden de actuación en la consecución de objetivos conflictivos entre sí en tareas tales como la optimización del flujo medio del tráfico, la minimización de las longitudes de las colas de vehículos en las intersecciones o la reducción de los niveles de contaminación acústica o ambiental.

A pesar de estas diferencias, como describe Ambrosino en [Ambrosino, 97], los actuales sistemas CTU comparten un mecanismo de funcionamiento basado en algoritmos y procedimientos cuantitativos. Este funcionamiento interno permite obtener un comportamiento correcto ante condiciones de flujo ligero de tráfico en un escenario de tráfico con escasa o nula saturación, pero no es capaz de gestionar adecuadamente, y mucho menos de anticiparse, a situaciones de saturación media o alta. Más aún, el uso de estos algoritmos y procedimientos cuantitativos en estas situaciones de saturación puede resultar en su empeoramiento [Bretherton, 82], [Robertson, 87], [Lear, 87].

Otro problema más sencillo y que estos sistemas tampoco son capaces de resolver de una forma satisfactoria consiste en la selección y ejecución de las acciones de control adecuadas que permitan actuar sobre situaciones no críticas inesperadas. Estas situaciones inesperadas generan cambios en el flujo normal de tráfico, lo que provoca alteraciones en rutas de tráfico o en elementos cercanos que resultan difíciles de explicar, como sucede en el caso de la presencia de accidentes o de vehículos parados que estén obstaculizando carriles.

Debido a que estos problemas no pueden ser resueltos mediante el exclusivo uso de modelos y métodos cuantitativos, los sistemas de control de tráfico urbano modernos aplican, sobre una primera etapa de cálculo cuantitativo, una segunda etapa en la que se realiza un uso intensivo de la experiencia del ingeniero de la sala de control de tráfico [Ambrosino, 91b], [Bell, 91], [Sybille, 87], [Barceló, 91a], [Toledo, 90b]. Esta persona en la encargada de tomar las decisiones de control adecuadas con el fin de suavizar, e incluso eliminar, los problemas actuales detectados. En estas condiciones críticas el ingeniero de la sala de control de tráfico, al poseer una visión global y profunda de la ciudad, está en condiciones de tomar decisiones más coherentes que las tomadas por los sistemas algorítmicos (que por lo general responden a una buena estrategia de administración del cruce, pero que no suelen manejar una información global, lo que puede originar una traslación del problema a otra u otras zonas).

Otra de las tareas que más frecuentemente ha de realizar el ingeniero de la sala, debido fundamentalmente a la dificultad asociada al proceso de selección de las acciones de control adecuadas que pueden resolver una situación problemática del tráfico, consiste en prever cuales van a ser éste tipo de situaciones y en determinar dónde van a manifestarse con el propósito de llevar a cabo las acciones de control anticipadas que eviten, en la medida de lo posible, que estas “futuras” situaciones lleguen finalmente a producirse.

Como consecuencia de la complejidad y el tamaño del problema a tratar en cada momento, las actuaciones y supervisión de los ingenieros han de ser continuas, lo que debido a una serie de factores [Toledo, 90b], [Bell, 91], hace que el sistema de CTU basado en dos capas sea insuficiente para obtener un mecanismo de control adecuado.

De una forma general, y atendiendo a una clasificación basada en los distintos niveles funcionales que pueden establecerse en un sistema CTU clásico, las características y métodos que pueden ser objeto de mejora forman parte de alguno de los siguientes niveles [Ambrosino, 97]

- **Nivel de recogida de datos**: Debido tanto a razones técnicas como económicas, las tareas de análisis y de toma de decisiones se realizan basándose en una visión incompleta de la composición del tráfico. Las causas más comunes que producen esta situación son la disponibilidad de un número limitado de sensores (debido tanto a restricciones operacionales como de coste), el funcionamiento defectuoso de los sensores (bien sea porque estén estropeados o bien por razones operacionales, como por ejemplo, los desajustes en sus niveles de sensibilidad debido a efectos climatológicos o a un mal calibrado), los fallos que se producen en la transmisión de datos,... etc. Los aspectos a mejorar en este nivel se orientan fundamentalmente hacia la consecución de una mejora en la calidad de los datos de tráfico almacenados en la base de datos del sistema CTU. Para realizar esta tarea resulta muy importante conseguir modelos o datos con las siguientes características:

  1. Métodos que permitan deducir datos perdidos –es decir, datos que han sido tomados por los sensores pero que no han llegado al sistema CTU–, y para poder proporcionar valores aceptables de la circulación de vehículos allí donde no existan sensores de tráfico o bien donde estos no funcionen adecuadamente. También resulta necesario disponer de métodos que nos permitan deducir cuando los datos que proporciona un sensor se corresponden con un comportamiento defectuoso del sensor, es decir, métodos que permitan identificar los sensores que estén estropeados.

  2. Métodos para poder estimar cual es el porcentaje de giro en cada intersección de la zona urbana bajo análisis.

  3. Una estimación del comportamiento del tráfico respecto a itinerarios origen/destino en ciertas áreas de la zona urbana.

  4. Métodos para integrar en el sistema CTU datos procedentes de distintos tipos de sensores. Tal y como se describió en el capítulo 1 existen una gran variedad en sensores de tráfico. Por lo tanto, resulta necesario establecer procesos de conversión de cada posible formato en el que un sensor puede proporcionar sus datos a un único formato estándar que permita realizar una gestión uniforme y más sencilla al sistema (métodos conocidos como *data fusion*).

  Estos requisitos constituyen la referencia sobre la que construir una mejor comprensión respecto a la evolución del tráfico y sobre la que poder realizar una mejor toma de decisiones de control.

- **Nivel de Monitorización**: La finalidad principal de este nivel consiste en realizar una interpretación correcta de cual es la situación global del tráfico en cada instante mediante el uso de un conjunto limitado de indicadores de tráfico locales, es decir, mediante el empleo de la información accesible del nivel de datos. En los actuales sistemas de control de tráfico urbano ya se emplean procesos para realizar un análisis del patrón de tráfico actual. En este

nivel también se incluyen aquellas tareas que precisan de una correcta interpretación del estado del tráfico, como son la selección de planes, el análisis de las congestiones o la facultad de predecir el estado del tráfico futuro.

Efectivamente, la correcta interpretación del estado del tráfico actual permite seleccionar el plan de señales a aplicar en cada instante que mejor se adapte a la situación detectada. Los métodos más comunes para realizar esta selección son mediante una descripción basada en un análisis del espacio de estados (es decir, empleando métodos numéricos) o bien mediante la aplicación de ciertas reglas de medición de umbrales de tráfico ajustables empíricamente. Sin embargo, tales métodos algorítmicos precisan de ser mejorados para que puedan adaptarse a la variación continua del entorno urbano y a los imprevisibles incidentes que puedan surgir. Además, estos métodos también deberían hacer uso de información respecto a la evolución temporal del comportamiento del tráfico, como puede ser el empleo de información histórica y de un análisis de las tendencias apreciables en la evolución de su flujo.

Otro aspecto a considerar consiste en cómo realizar el análisis de las congestiones y de las causas que las han producido. Para realizar este análisis se definen unos niveles de congestión de acuerdo a ciertos criterios empíricos que son utilizados para ayudar a la identificación de aquellas secciones congestionadas en la red urbana y de las interacciones producidas entre estas secciones de acuerdo a un criterio cronológico. Como consecuencia de este análisis se debe de identificar las causas que las han provocado además de establecer unos valores de prioridad a cada congestión detectada para seleccionar el orden en el que se van a solucionar.

Dada la dificultad intrínseca del proceso de cálculo del conjunto de órdenes de control que permiten resolver una congestión, un método alternativo consiste en incluir en el sistema CTU un nuevo componente especializado en calcular predicciones respecto a cual va a ser la situación del tráfico a corto y medio plazo (5-10 minutos). De esta manera, y apoyándose en estas predicciones se podrían establecer estrategias para la gestión de las acciones de control a tomar, incluyendo, por ejemplo, el poder anticiparse o retrasarse en la aplicación de un determinado plan de señales. Dadas las dificultades que presentan los modelos cuantitativos para poder realizar este tipo de predicciones a corto y medio plazo, especialmente cuando se analizan redes urbanas en el que se manejan un gran número de componentes, un método alternativo para conseguir esta predicción consiste en la definición, desarrollo e implementación de modelos cualitativos capaces de predecir únicamente aquellos cambios en el estado del tráfico que resulten significativos en el análisis de la red urbana.

- **Nivel de Control**: La decisión sobre que acciones de control se han de ejecutar en cada instante se realiza tomando como referencia los resultados

proporcionados por los niveles inferiores. Por este motivo, la calidad y efectividad de las acciones de control ejecutadas depende en gran medida de la calidad y fiabilidad de los datos proporcionados por estos niveles. No obstante, para conseguir una gestión mas "experta" sobre las tareas de control, también resulta adecuado acompañar estos datos con unos mecanismos adicionales de control, como puede ser por ejemplo, el añadir una capacidad adicional que permita evaluar de forma continua las prestaciones del sistema de tráfico según sean las estrategias de control que estén en curso actualmente. De esta forma, se pueda realizar un análisis individual de la necesidad y urgencia de cada intervención, y se puede calcular, por lo tanto, la ideonidad y los posibles efectos que pueden producirse al ejecutar las alternativas acciones posibles de control durante cada intervalo temporal.

Entre los aspectos más importantes que cabría esperar de un sistema de gestión de las acciones de control a ejecutar en cada instante para la consecución de un control efectivo del sistema se destacan los siguientes:

1. La capacidad para poder gestionar simultáneamente múltiples problemas detectados en la red urbana.

2. La habilidad para poder gestionar objetivos que resulten conflictivos entre sí. Por ejemplo, el favorecer el tráfico en unas determinadas áreas respecto del objetivo adicional de reducir sus niveles de contaminación.

3. La capacidad para poder tomar decisiones sobre la base de un análisis temporal del comportamiento del tráfico respecto de la repercusión que la ejecución de tales decisiones pueda ocasionar en toda la red urbana.

4. La capacidad para poder gestionar las congestiones no recurrentes, es decir, aquellas congestiones que, mediante la ejecución de las acciones de control adecuadas pueden ser evitadas, y también la capacidad para poder responder ante incidentes y eventos inesperados que puedan aparecer en la red urbana.

El análisis de los problemas identificados en cada uno de estos niveles, junto con las deficiencias detectadas en la organización estructural del sistema CTU en dos capas, ha llevado a diversos autores [Ambrosino, 91a], [Forasté, 87], [Bell, 90,91], [Moreno, 90a], [Cuena, 89], a considerar la posibilidad de introducir sistemas basados en el conocimiento dentro de la arquitectura de los sistemas de CTU, con el objeto de introducir en ellos conocimiento que el experto maneja para su toma de decisiones.

Las características y evolución de los prototipos que han sido desarrollados mediante el empleo de esta tecnología pueden plantearse con relación a la forma en que se realiza la integración tanto del conocimiento disponible del tráfico como del mecanismo empleado para realizar el proceso de inferencia. De esta forma, se suelen agrupar los SBC como pertenecientes a una de las siguientes categorías: Primera Generación de

Sistemas Basados en el Conocimiento, y Segunda Generación de Sistemas Basados en el Conocimiento.

2.3 Primera Generación de SBC

Los primeros sistemas basados en el conocimiento desarrollados para ser aplicados al control de tráfico urbano intentaban modelar la experiencia del ingeniero de la sala de control de tráfico mediante el empleo de enormes conjuntos de reglas empíricas *causa-efecto*, es decir, se representaban relaciones directas conocidas entre abstracciones de datos y abstracciones de soluciones. El conocimiento almacenado en estos primeros sistemas se denomina *superficial* puesto que no contiene información que explique los mecanismos que han provocado la existencia de estas relaciones *causa-efecto*.

2.3.1 Definición del Modelo

Los sistemas basados en reglas, también conocidos como sistemas de producción, constituyen uno de los tipos de sistemas expertos más extendidos. Esto es debido a las características inherentes de las reglas:

- Naturaleza modular. Resulta relativamente sencillo encapsular el conocimiento en bloques de pequeño tamaño y, por lo tanto, incrementar paulatinamente el conjunto de reglas disponibles en el sistema experto según se esté utilizando.

- Facilidad de explicación. La propia sintaxis de las reglas permite construir explicaciones sobre las acciones tomadas sin más que guardar las condiciones que han permitido la ejecución de cada una de las reglas aplicadas.

- Similaridad con el proceso de razonamiento humano. Las reglas representan una forma natural de modelar el mecanismo por el que un experto humano resuelve un problema de un dominio concreto [Hayes-Roth, 85].

Cada una de las reglas de un sistema experto convencional exhibe un formato consistente de un *antecedente (LHS)* – que establece las condiciones necesarias para que se pueda ejecutar la regla – y de un *consecuente (RHS)* – que ejecuta las acciones sobre la memoria de trabajo del sistema modificando su estado interno y permitiendo que algunas otras reglas puedan activarse.

En general, en un sistema basado en reglas se pueden identificar los siguientes componentes:

- Interfaz de usuario. Es el mecanismo por el que se comunican el usuario y el sistema.

- Facilidad de explicación. Almacena el proceso de razonamiento, es decir la secuencia de reglas, utilizado en el proceso de búsqueda de la solución.

- Memoria de Trabajo. Almacena los hechos, conocimiento fijo, del sistema.

- Base de Conocimientos (Memoria de Producciones). Almacén de las reglas que expresan las relaciones conocidas entre los hechos y las consecuencias, acciones que se derivan de los hechos que puedan existir en la memoria de trabajo.

- Motor de Inferencia. Realiza el proceso de decisión para determinar que reglas pueden ejecutarse en cada instante, establece un orden entre ellas y ejecuta la regla que resulte más importante.

- Agenda. Almacén del conjunto de reglas que puede ejecutarse en cada instante. El algoritmo RETE [Forgy, 82] proporciona un método eficiente para calcular este conjunto.

- Facilidad de incorporación de nuevo conocimiento. Mecanismo por el que se permite realizar altas, bajas y modificaciones en el conjunto de reglas almacenadas en la Base de Conocimientos.

El proceso de inferencia se realiza básicamente aplicando un mecanismo de *encadenamiento hacia delante* – de los hechos a las conclusiones - o *hacia atrás* – de las hipótesis a los hechos. Otros métodos de inferencia también se utilizan aunque con menos frecuencia [Giarratano, 98].

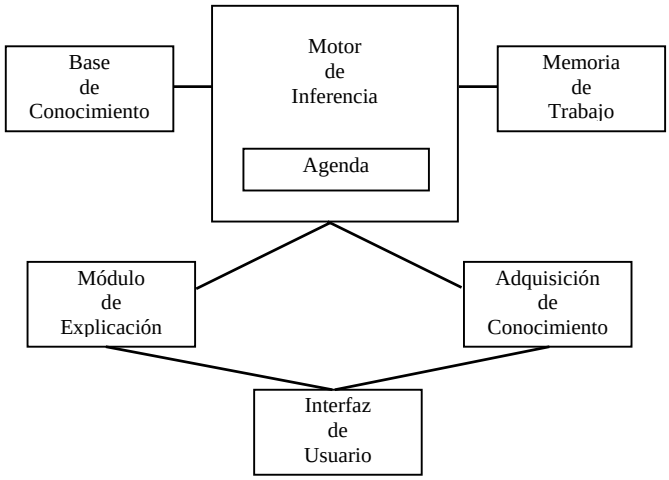


Fig. 2-1 *Estructura de un Sistema Experto clásico*

El motor de inferencia funciona ejecutando ciclos. En cada uno de ellos se realizan una serie de etapas características, que para el caso concreto del sistema experto OPS5, uno de los más conocidos y utilizados, son las siguientes [Giarratano, 98]:

MIENTRAS no fin *HACER*
Resolución de conflictos: Seleccionar la regla con mayor grado de prioridad
Acción: Ejecutar el consecuente de la regla seleccionada. Eliminar la activación de esta regla del conjunto de reglas ejecutables.
Búsqueda de reglas ejecutables: Actualizar la agenda con las nuevas reglas que se pueden ejecutar y eliminar las activaciones de las que ya no sean ejecutables
Comprobación de si se ha obtenido la solución.
FINMIENTRAS

2.3.2 Sistemas CTU Basados en Reglas

Los sistemas basados en reglas constituyeron los primeros sistema expertos aplicados a la ayuda en la toma de decisiones para sistemas CTU. Estos sistemas tenían su principal área de aplicación en situaciones de pre-congestión, es decir, antes de que en la red urbana se produzcan situaciones de congestión. Un buen número de proyectos se desarrolló persiguiendo esta finalidad, entre otros SAGE (*S*ystème d'*A*ide à la *G*estion des *E*mbouteillages) [Foraste, 96].

EL SISTEMA EXPERTO SAGE

El objetivo de este sistema de ayuda a la toma de decisiones en el sistema de control de tráfico urbano de París consiste en detectar el origen de las congestiones y de diagnosticarlas, calculando su importancia y evolución, y avisando al operador de tráfico de su presencia. Para realizar esta tarea el sistema dispone de los datos acumulados por los sensores de sistema (sensores de lazo cerrado) recogidos en periodos de integración de tres minutos.

Las congestiones sólo son detectadas y diagnosticadas cuando ya se han producido, es decir, el sistema no tiene la capacidad de poder anticiparse a su aparición debido, fundamentalmente, a que SAGE no contiene un modelo que explique el proceso de evolución de las congestiones.

El sistema experto SAGE está formado por tres componentes principales:

- El *motor de inferencia*. El mecanismo básico de inferencia es un proceso de encadenamiento hacia delante, en el que se han realizado ciertas modificaciones para adaptarlo a un funcionamiento sometido a restricciones débiles en tiempo real.

- La *base de hechos*. Contiene tanto la descripción de la red urbana – sus componentes, características y relaciones – junto con los objetos creados durante la ejecución del sistema. Los objetos se pueden agrupar mediante la definición y uso de las relaciones adecuadas.

- La *base de conocimientos*. Este componente contiene las reglas que expresan el conocimiento que se posee sobre el sistema. Por ejemplo, la siguiente regla localiza y realiza una diagnosis sobre aquellas rutas en la red que están saturadas:

```
Rule: new_chain
      properties;
            priority: 0
            comment: Generation of saturated chain in a route
      end properties
      {     (new_state(!link1) = congested
            successor(!link1) = ?link2
            state(?link2) ≠ congested        }
      THEN
            create[fact_base, %chain]
            first(%chain) ← !link1
            starting-junction(%chain) ← end(!link1)
            belongs(!link1) ← %chain
end Rule
```

  En esta regla *!link1* y *?link2* representan, respectivamente, variables cuantificadas universal y existencialmente, mientras que *%chain* identifica a una variable local.

Las acciones a tomar en cada instante las realiza el operador de tráfico. El sistema experto, por lo tanto, se limita a señalar los problemas detectados y sus causas probables. No obstante, el operador tiene la posibilidad de proporcionar información al sistema experto. Por ejemplo, puede forzar que el estado de un segmento de la red que no contenga detectores de tráfico tenga un determinado valor si así lo estima al valorar la información proporcionada por algún otro medio en el sistema (por ejemplo, por cámaras de vídeo). Adicionalmente, el operador puede indicar al sistema las causas que han provocado alguna de las congestiones, actualizando de esta manera la base de hechos del sistema.

2.3.3 Limitaciones de los SBC de Primera Generación

Resulta difícil para los sistemas basados en el conocimiento pertenecientes a la primera generación el poder deducir dónde se producen los problemas en el dominio de la aplicación o bien cuáles son las causas que los han producido. Esto es debido a que este tipo de sistemas basados en el conocimiento realiza el proceso de diagnosis mediante la aplicación de técnicas de clasificación heurística. Estas técnicas representan el conocimiento sobre la diagnosis del sistema mediante el uso de reglas heurísticas, las cuales representan relaciones entre abstracciones de datos (síntomas) y abstracciones de soluciones (enfermedades). Esta representación del conocimiento se denomina *superficial* puesto que no contienen información acerca de los mecanismos causales que soportan a las relaciones establecidas entre síntomas y enfermedades. Las reglas reflejan, por lo tanto, una asociación empírica derivada de la experiencia más que una teoría bien fundada de cómo funciona el sistema sobre el que se está actuando (denominado *conocimiento profundo*) [Jackson, 90].

Este conocimiento *profundo* no está completamente ausente en los sistemas de conocimiento *superficial*, sino que suele estar ya sea integrado en las reglas de una manera que resulta difícil separarlo del conocimiento *superficial*, o incluso puede estar formando parte de otras estructuras de datos que se utilizan en el funcionamiento del motor de inferencia del sistema. Ante cualquier situación que se presente en el sistema, el conocimiento *superficial* determina el espacio de búsqueda de las soluciones, ya que es éste el conocimiento que caracteriza las relaciones entre los datos y las soluciones [Jackson, 90].

Además, los sistemas que utilizan este tipo de conocimiento *superficial* son muy sensibles a todo tipo de cambios en la disponibilidad y funcionamiento de los elementos del sistema [Ambrosino, 91b], [Ambrosino, 97], [García, 99]. Por ejemplo, si en el sistema se añaden, modifican o eliminan sensores entonces la base de conocimiento también debe de modificarse, puesto que algunas reglas del sistema dejarán de ser aplicables. El proceso de determinar cual es este conjunto de reglas no es sencillo, por lo que en el sistema se debe de ejecutar un proceso para mantener la integridad de los datos de la memoria de trabajo y del conjunto de reglas aplicables en cada momento.

Estas razones han promovido la investigación y desarrollo de sistemas que integren de una forma explícita conocimiento *profundo*. Este conocimiento se suele incorporar en los sistemas basados en el conocimiento aplicados al control de tráfico urbano, además de con la información heurística del experto en tráfico, por medio de un proceso simulador de tráfico urbano que permita explicar la estructura, comportamiento y evolución del tráfico urbano.

2.3.4 Integración de un Simulador en un Sistema CTU

La simulación es un proceso consistente en la construcción de un modelo computerizado que represente de una forma adecuada a un sistema real o ficticio. Tal modelo debe de permitir extraer unas inferencias y resultados similares a los que se extraerían del proceso real. Este proceso de simulación ha llegado a convertirse en una de las herramientas más usadas y poderosas para la realización y análisis de un gran número de sistemas. El uso principal que se les da consiste en analizar cual sería la situación y evolución del sistema ante diferentes alternativas de asignación de valores iniciales o intermedios, lo que sirve de una gran ayuda en el proceso de diseño de nuevos sistemas o en la estimación de cual puede ser el impacto sobre el sistema si se realizan los cambios propuestos.

Un modelo de simulación siempre consiste en una representación simplificada de la realidad que destaca aquellos aspectos que resultan más importantes para los propósitos de su análisis. Un modelo de simulación es, por lo tanto, específico tanto para el problema al que se aplica como al modelo desarrollado. El estudio de la simulación de un sistema proporciona un mejor entendimiento de su funcionamiento, evaluando de una forma continua el impacto de los cambios que se producen y valorando las diferentes estrategias de control que se puedan aplicar.

Esta utilidad general del proceso de simulación como herramienta para ayuda al análisis de sistemas se consigue también cuando se aplica al estudio de sistemas de tráfico y transporte. Resulta fundamental para una gran cantidad de tareas importantes – planificación del transporte, observación por vídeo y monitorización, detección de incidentes, diseño de estrategias de control, evaluación del estado del tráfico en cada instante, predicción de tráfico y evaluación de los objetivos perseguidos en el sistema – el disponer de un modelado del comportamiento del flujo de tráfico [Lieberman, 97].

2.3.4.1 Tipos de Simulación

Existe una gran variedad respecto a posibles clasificaciones de modelos de simulación según sea la característica principal seleccionada como referencia para realizar esa simulación [Lieberman, 97]:

- Representación de los cambios de estado: *continuos* y *discretos*.
- Grado de detalle en la representación: *microscópicos* y *macroscópicos*.
- Representación de los procesos en el sistema: *deterministas* y *estocásticos*.

Cuando la característica fundamental que se destaca es el modo en el que se producen los cambios de estado en los elementos del sistema, los simuladores se clasifican en *continuos* y *discretos*. Un simulador *continuo* es aquel en el que los elementos cambian de estado de una forma continua en el tiempo en respuesta a un estímulo también continuo. Un modelo de simulador *discreto* es aquel en el que los elementos del sistema cambian de una forma brusca de un estado a otro en ciertos instantes temporales. Existen dos tipos fundamentales de modelos discretos:

- Modelos discretos de tiempo discreto
- Modelo discretos de eventos

El primero de estos modelos segmenta la línea temporal en una sucesión de intervalos de tiempo conocidos. Dentro de cada intervalo, el modelo de simulación computa las actividades que cambian de estado en ciertos elementos seleccionados del sistema. Esta aproximación resulta análoga a representar una ecuación diferencial de valor inicial en forma de una expresión en diferencias finitas con una variable independiente Δt.

Los sistemas CTU están compuestos generalmente de entidades que están “ociosas” la mayor parte del tiempo. Por ejemplo, el estado de una señal de tráfico (por ejemplo, un semáforo en luz verde) permanece constante durante bastantes segundos hasta que su estado cambia instantáneamente a otro estado (continuando con el ejemplo anterior, el cambio de luz verde a luz roja en el semáforo). Este cambio abrupto en el estado del semáforo se denomina un *evento*. Al resultar posible describir de una forma ajustada el funcionamiento de una señal por medio de almacenar sus cambios de estado como una sucesión de *eventos temporales*, se puede conseguir un considerable ahorro en tiempo

de computación atendiendo únicamente a la ejecución de estos eventos en vez de computar el estado de la señal segundo a segundo. Existen ventajas e inconvenientes con respecto al uso de cualquiera de estas aproximaciones. En aquellos sistemas cuya potencia de cálculo sea seriamente limitada o que representen fundamentalmente entidades cuyos estados varían poco frecuentemente, los modelos discretos basados en eventos resultan ser más apropiados que los modelos discretos de tiempo discreto. Además los modelos basados en eventos proporcionan los resultados de la simulación bastante más rápidamente que los modelos de tiempo discreto. No obstante, los modelos de tiempo discreto son más frecuentemente utilizados en aquellos sistemas en los que la mayor parte de entidades que lo componen experimentan cambios continuos de estado (por ejemplo, el estudio del estado del tráfico en un segmento) y en aquellos en los que los objetivos del sistema precisan de unas descripciones del sistema más detalladas.

Otra clasificación diferente de los tipos de simulación puede realizarse si se destaca el nivel de detalle con el que se estudia el sistema[1]:

- Microscópico (alto nivel de fidelidad)
- Macroscópico (bajo nivel de fidelidad)

Un modelo de simulación *microscópico* describe tanto las entidades que componen el sistema como sus interacciones con un alto nivel de detalle. Se considera cada uno de los elementos del sistema y todas sus interacciones incluyendo cada una de sus duraciones. Un sistema de este tipo modela cada uno de los vehículos en circulación, su categoría y características de tráfico, incluyendo por ejemplo, la duración de los tiempos de reacción de cada conductor.

Un modelo de simulación *macroscópico* describe las entidades y sus actividades e interacciones con un bajo nivel de detalle. Por ejemplo, el estado del tráfico en un segmento se puede representar basándose en alguna de sus características mediante el uso de valores escalares de tasas de flujo, densidad y velocidad media. Algunas de las acciones de los elementos del sistema no pueden ser representadas, por ejemplo, las maniobras individuales de cambio de carril.

Los modelos de menor nivel de detalle son más sencillos y menos costosos de desarrollar, ejecutar y mantener. Pero estos modelos contienen el riesgo de que su representación del mundo real sea menos ajustada, menos válida o quizás inadecuada. Por lo tanto, resulta imprescindible para la realización de estos sistemas el proceso de selección y posterior aplicación de una teoría de tráfico bien fundamentada que los sustente. Se puede establecer que, de forma general, el empleo de simuladores de bajo nivel de detalle puede resultar adecuado en dominios de problemas de tráfico que presenten las siguientes características:

[1] Algunos autores también identifican simuladores *mesoscópicos* como un grupo intermedio entre los simuladores *microscópicos* y los macroscópicos. Los simuladores *mesoscópicos* se caracterizan como aquellos en los que las entidades se describen con un alto grado de detalle pero sus interacciones son descritas a menor grado de detalle.

- Los resultados no son sensibles a detalles microscópicos.
- El tamaño de la escala de la aplicación no puede ajustarse a la gran cantidad de tiempo de ejecución preciso por un modelo microscópico.
- Tanto el tiempo disponible para el desarrollo del modelo como los recursos están fuertemente limitados.

Una última clasificación identifica los posibles tipos de simulación con relación al modo en que se representan los procesos del sistema:

- Modelos de simulación determinista
- Modelos de simulación estocástica.

Los modelos determinístas no poseen variables aleatorias, todas las interacciones entre sus entidades se definen por medio de relaciones exactas (bien sean éstas matemáticas, estadísticas o lógicas). Los modelos estocásticos, por otra parte, implementan procesos que incluyen funciones de probabilidad. Por ejemplo, un modelo de seguimiento de la trayectoria de un vehículo puede formularse de una manera determinística o estocástica según se defina el tiempo de reacción del conductor por medio de un valor constante o de una variable aleatoria.

A pesar de que cada modelo de simulación de tráfico desarrollado implementa características de cada una de las clasificaciones expuestas, la mayor parte de estos simuladores suelen catalogarse como correspondientes a modelos microscópicos o macroscópicos.

2.3.4.2 Simulación Microscópica de Tráfico

Los modelos de simulación microscópicos están compuestos, fundamentalmente, de dos componentes, una descripción ajustada y lo más completa posible de la topología de la red urbana (incluyendo, por ejemplo, la posición y tipo de semáforos, detectores, paneles informativos variables, etc.); y un modelado lo más fiable posible del comportamiento del tráfico que reproduzca la dinámica de cada vehículo en circulación de forma individual, estableciendo diferencias entre los distintos tipos de vehículos e incorporando en los cálculos de la simulación los distintos posibles comportamientos de los conductores.

Los modelos de simulación microscópica son los más cercanos a la realidad respecto de la fidelidad al sistema de tráfico pudiendo analizar un amplio rango de escenarios de tráfico posibles en las que se pueda incluir descripciones precisas de esquemas de gestión de tráfico y de operaciones de control.

Cada uno de los comportamientos de los vehículos se simulan de forma individual mediante el uso de modelos de seguimiento de vehículos. Estos modelos consisten de ecuaciones de diferencias finitas que representan la aceleración que sufre un vehículo

con respecto al comportamiento de los vehículos que le preceden en la vía. Una exposición teórica de este tipo de modelos se puede encontrar en [Gabard, 91]. La forma general de los modelos de seguimiento de vehículos se corresponde con la siguiente expresión en donde *T* representa el tiempo de reacción de la combinación vehículo-conductor:

$$respuesta(t+T) = sensitividad \times estímulo(t)$$

En la mayoría de los modelos esta respuesta siempre será la aceleración (o deceleración) del coche que le sigue, y el estímulo será la diferencia de velocidad entre el coche que le precede y el que le sigue.

Esta expresión suele implementarse de forma lineal mediante el uso de la siguiente ecuación diferencial en diferencias de segundo orden. Esta expresión se define como función de *T* y de λ (coeficiente de sensitividad estimado de forma experimental):

$$\ddot{x}_{n+1}(t+T) = \lambda\left[\dot{x}_n(t) - \dot{x}_{n+1}(t)\right]$$

Los modelos que implementan esta expresión lineal para el seguimiento de vehículos no son muy estables, en el sentido de que al cambiar la velocidad del vehículo que encabeza un conjunto de vehículos se amplifica la velocidad de cada uno de los vehículos que le siguen. Para solucionar este problema se han propuesto modelos no lineales, como por ejemplo el modelo de Gipps [Gipps, 86]. Este modelo se basa en la suposición de que cada conductor define sus propios límites respecto a la aceleración y al frenado de su vehículo.

Existe un gran número de simuladores microscópicos de tráfico. A pesar de que algunos son productos comerciales (AIMSUN2 [Barceló, 94], PARAMICS [Duncan, 96]), casi todos ellos se corresponden con productos de investigación continuamente en desarrollo. En [SMARTEST, 99] se realiza un análisis comparativo de los 17 simuladores microscópicos más representativos.

Las principales deficiencias en el uso de este tipo de simuladores se centran en los siguientes apartados:

- Dificultad en modelar las congestiones debido a que los modelos implementados de seguimiento de vehículos y de cambio de carril no reflejan de una forma real el comportamiento de los conductores.

- Falta de uniformidad respecto al formato de los datos de entrada para todos los modelos integrados en el simulador, lo que produce un esfuerzo extra al tener que introducir los datos varias veces o al tener que ejecutar programas de conversión.

- Necesidad de calibrar constantemente entre grado de detalle de la simulación y tiempo de ejecución.

- Dificultad en desarrollar modelos de comportamiento y de reacción para todos los posibles usuarios de la red urbana, como pueden ser los ciclistas o los peatones. Lo que además afecta a la medición del grado de seguridad del sistema.

A pesar de estas deficiencias, este tipo de simuladores son ampliamente utilizados en las siguientes situaciones:

- Contribuir al proceso de asignación de valores de tráfico para cada uno de los esquemas de transporte a analizar. Esta aportación resulta muy importante debido a la gran dificultad que plantea el poder disponer de suficientes datos para poder producir conclusiones estadísticas satisfactorias sobre el comportamiento del tráfico día a día.

- Realización de predicciones a corto plazo para poder evaluar el conjunto posible de intervenciones a realizar debido a un incidente o para reducir el número de emisiones contaminantes.

- Ayudar a evaluar los posibles modos en los que se verá modificado el tráfico si se producen cambios en la red urbana.

- Como medio de proporcionar datos lo más cercanos posible a la realidad para los simuladores de conducción de vehículos en condiciones de total seguridad.

2.3.4.3 Simulación Macroscópica de Tráfico

Los modelos de simulación de tráfico macroscópicos proporcionan un análisis de cómo se comporta el flujo dinámico de vehículos. Normalmente están basados en una analogía hidrodinámica al considerar el flujo de tráfico como un proceso particular del comportamiento de un fluido cuyo estado se caracteriza mediante la adición al sistema de variables macroscópicas como pueden ser la densidad de tráfico (medido en vehículos/Km.), el volumen de tráfico (medido en vehículos/hora) y la velocidad media (medida en Km./hora) [Michalopoulos, 91].

Uno de los primeros sistemas que hacían un uso de un simulador macroscópico fue AURA, desarrollado por J. Cuena [Cuena, 88]. Este simulador utiliza métodos de modelado cualitativos aplicados al modelado de tráfico. El sistema AURA implementa una descripción cualitativa del flujo de tráfico en autopistas de acceso a entornos urbanos con el propósito de analizar situaciones cercanas a niveles de congestión. Las acciones de control posibles que podía ejecutar este sistema consistían en imponer cambios de dirección en aquellos carriles definidos como reversibles, y la selección de los mensajes más adecuados a mostrar en los paneles informativos variables. La tarea del simulador en este sistema consistía en la realización de predicciones a corto y medio plazo para identificar cuales pueden ser los futuros incidentes que se puedan deber a la actual estrategia de control de tráfico.

Este modelo de simulador no es adecuado para su aplicación en control de tráfico urbano debido a que no considera intersecciones ni semáforos. La introducción de estos conceptos hace necesario definir e introducir información temporal sobre conceptos tales como el ciclo, el reparto y la coordinación.

Estas características temporales no pueden ser gestionadas por el simulador integrado en el sistema AURA debido a la limitada potencia de su tratamiento temporal. Efectivamente, el algoritmo de simulación implementado sólo proporciona los posibles estados que siguen al estado actual, pero no proporciona información sobre el momento en que esta transición de estado se va a realizar. Como consecuencia, para calcular este instante de cambio surge un problema de búsqueda con un espacio de soluciones exponencial [Toledo, 90].

Otros simuladores macroscópicos han sido desarrollados para tratar con el problema de control de tráfico urbano. En particular, el simulador desarrollado por Moreno [Moreno, 93] en el entorno del proyecto EQUATOR [EQUATOR, 91]. En este simulador se realiza un proceso de razonamiento temporal basado en eventos. Los nuevos eventos del sistema se calculan por medio de reglas de deducción de eventos. Estos eventos representan tanto acciones externas (por ejemplo, el cambio de rojo a verde en un semáforo) como acciones internas (por ejemplo, una modificación significativa de la densidad de tráfico en algún segmento o cruce del sistema).

Como característica adicional de este simulador resalta la simplicidad de su base de datos temporal. En un simulador microscópico, se almacena en la base de datos una gran cantidad de valores para representar al sistema completo en cada instante temporal. Como casi toda esta información representa información no relevante, es decir, con poca o nula expresividad cognitiva, se precisa de un proceso adicional que compute el significado de todo este conjunto de datos. En el simulador desarrollado por [Moreno, 93], sólo se almacena en la base de datos unos pocos eventos, cada uno de ellos asociado a un evento significativo que ha sucedido en la red. Por lo tanto, estos datos pueden ser directamente analizados sin tener que precisar del proceso de un análisis posterior con un alto coste computacional.

En el proceso de analizar las ventajas e inconvenientes que plantea el uso de un simulador macroscópico respecto de uno microscópico, además de las ya apuntadas anteriormente en el apartado de clasificación de simuladores, cabe destacar las siguientes:

- Precisa de un modelo teórico muy ajustado para conseguir una fiable representación de la realidad.

- Requiere un proceso de razonamiento adicional sobre los datos proporcionados por los sensores debido a la incertidumbre asociada a sus valores. Por ejemplo, un valor de 30 cuentas por hora puede representar tanto una situación correspondiente a un flujo altamente congestionado como a un flujo ligero.

- Presentan un coste computacional sensiblemente más reducido que los simuladores microscópicos y, por lo tanto, pueden proporcionar una predicción del tráfico a más largo plazo.

- Proporcionan pocas medidas pero con un alto grado de significación cognitiva (conocimiento) en contraposición a los simuladores microscópicos que proporcionan muchos datos, que en sí mismos, poseen nula o casi nula significación.

2.4 Segunda Generación de SBC

La gran sensibilidad de los sistemas basados en el conocimiento de la primera generación respecto de las características de cada uno de los objetos del sistema así como su limitado conocimiento respecto de las causas que motivan el comportamiento del sistema en cada instante (como ya se apuntó en el apartado anterior), provocaron el desarrollo de nuevos sistemas que implementasen conocimiento *profundo*.

Este conocimiento, para que sea realmente útil, debería de incorporar las estructuras de razonamiento correspondientes a lo que podríamos llamar las teorías sobre el dominio de aplicación [Forbus, 88]: conocimiento sobre la estructura interna, funcionamiento y/o composición de los objetos de estudio (lo que se ha dado en llamar conocimiento profundo). Este conocimiento se estructura en dos niveles [Salmerón, 92]: un primer nivel con conocimiento que explica el comportamiento del sistema sobre el que se razona, y un segundo nivel con conocimiento sobre la búsqueda de soluciones a un problema dado (en nuestro caso esto es lo que podríamos llamar conocimiento de control).

Esta estructura de dos niveles ha sido implementada en varios sistemas CTU, por ejemplo, los sistemas IUTCS y KITS desarrollados en los proyectos DRIVE I y II, en el sistema AURA [Cuena, 88]) y en el marco del programa ESPRIT II EQUATOR [EQUATOR, 91,94].

La exposición de los principales prototipos de sistemas basados en el conocimiento de segunda generación aplicados al control de tráfico urbano se va a realizar atendiendo a la forma en que cada uno de sus componentes interaccionan entre sí, es decir, de acuerdo a su arquitectura funcional. De esta forma se definen los prototipos con planificación de ejecución *estática* para sus módulos como aquellos en los que estos componentes se ejecutan siguiendo siempre una misma secuencia fija. Consecuentemente, los prototipos con planificación de ejecución *dinámica* se definen como aquellos en los que sus componentes interactuan entre sí sin tener que seguir una secuencia de ejecución fija.

El estudio de estos sistemas estructurados en dos niveles ha proporcionado dos resultados fundamentales. El primero es una mejor identificación de los problemas asociados al proceso de control de tráfico urbano y la realización de una clasificación de los mismos con el propósito de determinar las teorías y técnicas más adecuadas para

construir componentes software que o bien traten de solucionarlos, o bien de reducir su grado de dificultad en alguno de sus aspectos [Ambrosino, 97][García, 99].

El segundo resultado se relaciona con la importancia que para un sistema CTU tiene el disponer de un proceso simulador que determine la naturaleza y generación de todos los cambios detectados por el tráfico urbano. Algunos de los sistemas desarrollados optan por implementar simuladores de tráfico microscópicos, mientras que en otros sistemas el simulador utilizado es macroscópico. La elección del tipo de simulador a utilizar en cada sistema es una decisión que depende de la filosofía de trabajo para la que se destine el sistema CTU, y que determina, debido a las ventajas e inconvenientes asociados a cada tipo de simulación el tipo de resultados y de acciones que el sistema podrá realizar.

Uno de los métodos aplicados para estructurar el conocimiento *profundo* sobre el comportamiento del tráfico urbano es el método desarrollado en el entorno del proyecto ESPRIT II EQUATOR. Este método se expone ampliamente en el siguiente apartado.

2.4.1 Representación Temporal-Cualitativa del Tráfico Urbano.

El *razonamiento cualitativo* [Kuipers, 86, 94], [De Kleer, 84], [Forbus, 84] consiste en una serie de métodos que permiten manipular sistemas de los que se posee un conocimiento incompleto. El uso de modelos cualitativos permite expresar estados de conocimiento incompleto en sistemas en los que se produce un cambio de estado continuo de una forma más adecuada que los métodos tradicionales. Además, el uso de simulación cualitativa permite determinar todos los comportamientos que resultan consistentes con la información que se posee del sistema. Esta potencia expresiva sumado al amplio rango de sistemas a las que estos métodos pueden aplicarse, han propiciado que se aplique de forma intensa en tareas de diagnosis, diseño, monitorización, procesos de explicación y otras aplicaciones de la inteligencia artificial [Kuipers, 94].

Los primeros sistemas desarrollados mediante modelos cualitativos compartían una serie de características [EQUATOR, 90]

- El mundo real se descompone en una colección de objetos simples que se pueden agrupar para construir objetos más complejos.
- El estado de cada objeto está determinado por un conjunto de parámetros.
- Los parámetros son funciones temporales que toman valores en un espacio continuo, por lo que se suelen representar como funciones $F_i(t)$. A cada valor cualitativo representativo de cada parámetro se le asocia un espacio de valores cuantitativos.
- La evolución del sistema se puede representar mediante una cadena de sucesión de estados, donde cada cambio de estado representa el cambio de algún valor cualitativo asociado a un parámetro del sistema.

- La evolución temporal del sistema está determinada por medio de una serie de restricciones entre los posibles valores de sus parámetros.

Sin embargo, estos sistemas adolecían de un paradigma de razonamiento cualitativo que fuese capaz de tratar con la simulación cualitativa de sistemas físicos en los que los parámetros pudiesen ser funciones de dos o más variables. Esta limitación llevó al desarrollo del formalismo (λ, t) [EQUATOR, 90], [Toledo, 91], [Toledo, 93].

2.4.1.1 El formalismo (λ, t)

El formalismo (λ, t) se desarrolló para tratar con sistemas en los que alguno de sus parámetros fuese una función bidimensional en los que uno de sus elementos fuese el tiempo (*t*) y el otro fuese distinto del tiempo (denotado de forma general por λ). Este formalismo aporta dos herramientas fundamentales: Un método que permite establecer una representación jerárquica del conocimiento del sistema físico y un motor de inferencia capaz de gestionar parámetros unidimensionales y bidimensionales.

La representación jerárquica del conocimiento se puede dividir en cinco partes [Toledo, 93]:

- Taxonomía de objetos: Consiste en la especificación de todos los objetos físicos relevantes para la aplicación. El formalismo establece tres clases básicas: *Dominio de la aplicación* (superclase de todos los objetos), *externa* (que recoge todos los objetos que representan condiciones frontera del sistema físico) e *internas* (agrupa al resto de objetos que no pertenecen a la clase externa).

- Taxonomía de relaciones: Consiste en la especificación de todas las clases que agrupan relaciones causales entre los objetos del sistema. Una relación entre dos objetos indica que la evolución de uno depende directamente del otro.

- Taxonomía de parámetros: Consiste de cuatro clases: *Parámetros* (superclase de todos los parámetros); *F* (parámetros constantes); *F(t)* (parámetros función del tiempo) y *F(λ, t)* (parámetros función del tiempo y de algún otro parámetro) (figura 2-2).

- Comportamiento de los objetos: Está determinado por ecuaciones entre los parámetros de cada objeto y pueden ser de dos tipos:

  - *Acción*: si un parámetro es una función unidimensional de otros.

  - *Restricción*: el resto de ecuaciones.

- Comportamiento de las relaciones: Actúan como entidades con capacidad de proceso autónomo, realizando tareas como canales de comunicación entre los objetos.

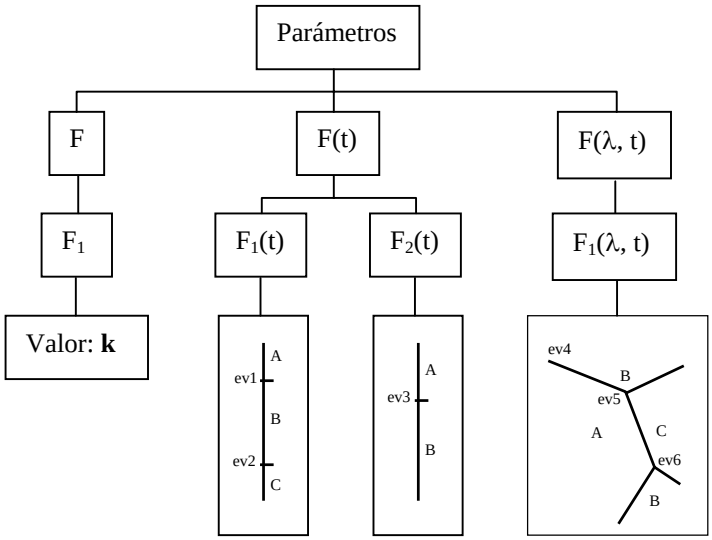


Fig 2-2 *Taxonomía de parámetros en el formalismo (λ, t). Un parámetro de la clase F es un valor constante para el sistema. Un parámetro de la clase F(t) representa la evolución temporal lineal de sus valores cualitativos. En la figura el parámetro $F_1(t)$ representa que el estado del sistema era A antes de la aparición del evento ev1, inmediatamente después el estado es B hasta que sucede ev2 que provoca un nuevo cambio de estado al valor cualitativo C. El parámetro $F_1(\lambda, t)$ representa la evolución temporal y espacial de sus valores cualitativos.*

2.4.1.2 Aplicación del formalismo (λ, t) al modelado de tráfico urbano

La caracterización del estado del tráfico se puede realizar por medio de dos parámetros: *densidad* y *velocidad* de los vehículos en cada instante temporal y espacial [ETRA, 90].

No obstante, bajo ciertas condiciones, se puede establecer una relación directa entre ambos parámetros, y por lo tanto, sólo puede ser necesario conocer el valor de uno de ellos para caracterizar el tráfico urbano en cualquier momento (figura 2-3). Sea ese parámetro la *densidad* de vehículos. Mediante la identificación de aquellas secciones de la calle que presentan un valor de densidad constante se puede caracterizar de forma cualitativa el estado del trafico. Por ejemplo, el uso de este parámetro permite representar la evolución del tráfico en una calle, en la que el semáforo que se encuentra en su extremo final cambia de rojo a verde, de una forma *cuantitativa*, contando la densidad de vehículos en cada instante temporal y espacial (figura 2-4 izquierda) o bien de una forma *cualitativa*, representando aquellos puntos del espacio y del tiempo en que alguna de las zonas de densidad constante aparece o desaparece (figura 2-4 derecha).

Para representar de una forma cuantitativa este proceso se necesita conocer exactamente el valor de la función de densidad en el espacio y en el tiempo, es decir, el valor de *densidad(x,t)*. Mientras que en la descripción cualitativa, que aplica el

formalismo (λ, t), sólo es necesario conocer los instantes temporales y espaciales en que aparece o desaparece una de las regiones del espacio que tiene densidad constante.

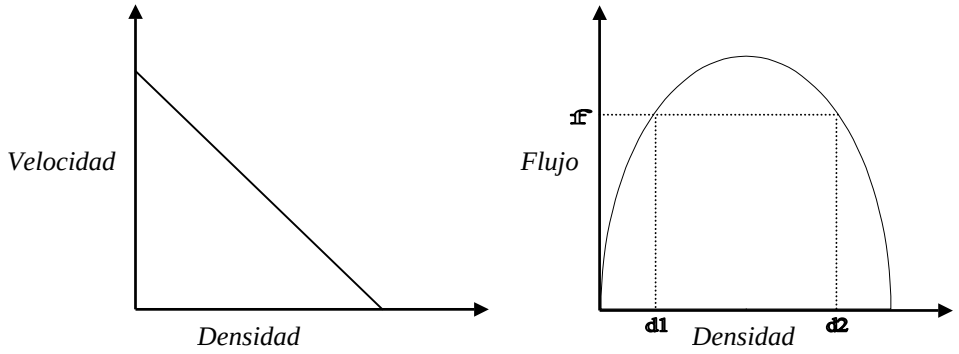


Fig 2-3 *Representación de las funciones bidimensionales del flujo y velocidad en función de la densidad de vehículos. En la figura de la izquierda se observa que a mayor densidad de vehículos la velocidad del tráfico disminuye. En la figura de la derecha se representa el carácter parabólico del flujo respecto de la densidad, en el que un único valor de flujo puede estar asociado a dos valores diferentes de densidad.*

Mediante el uso de la descripción cualitativa sólo se tienen en cuenta los eventos que resultan significativos en la descripción de la evolución del tráfico en la calle. En el caso representado en la figura 2-4, la descripción cualitativa de la calle representa el proceso de desaparición de una cola de vehículos. Durante este proceso se generan varios eventos. Los eventos *e1* y *e2* representan la aparición de nuevas regiones cualitativas, mientras que los eventos *e3*, *e4* y *e5* representan la desaparición de regiones cualitativas. Como se observa en la figura según pasa el tiempo se generan una serie de líneas rectas que representan la evolución temporal del tamaño de cada región cualitativa y que pueden ser interpretadas como “frentes de ondas”.

La base de datos temporal cualitativa sólo representa estos cinco eventos. Con lo que se consigue representar la evolución del sistema de una forma muy compacta y con un alto nivel de conocimiento.

Este comportamiento cualitativo del tráfico puede ser explicado mediante el análisis temporal y espacial en cada punto de una calle de las siguientes ecuaciones de modelado del tráfico [ETRA, 90]:

Ecuación 1 $$flujo(x,t) = velocidad(0,0) - K \cdot densidad(x,t)$$

Ecuación 2 $$flujo(x,t) = velocidad(x,t) \cdot densidad(x,t)$$

Ecuación 3 $$flujo(x,t) = velocidad(0,0) \cdot densidad(x,t) - K \cdot densidad(x,t)^2$$

La **ecuación 1** no se corresponde con una ley física sino que es una relación empírica que explica como la velocidad de un vehículo en un momento y en un punto del espacio

determinado depende de la velocidad del vehículo al inicio de la calle y de la densidad de vehículos en ese mismo punto de la calle y en el mismo instante.

La **ecuación 2** se corresponde con la definición física del concepto de flujo, mientras que la **ecuación 3** es una deducción de las dos ecuaciones anteriores.

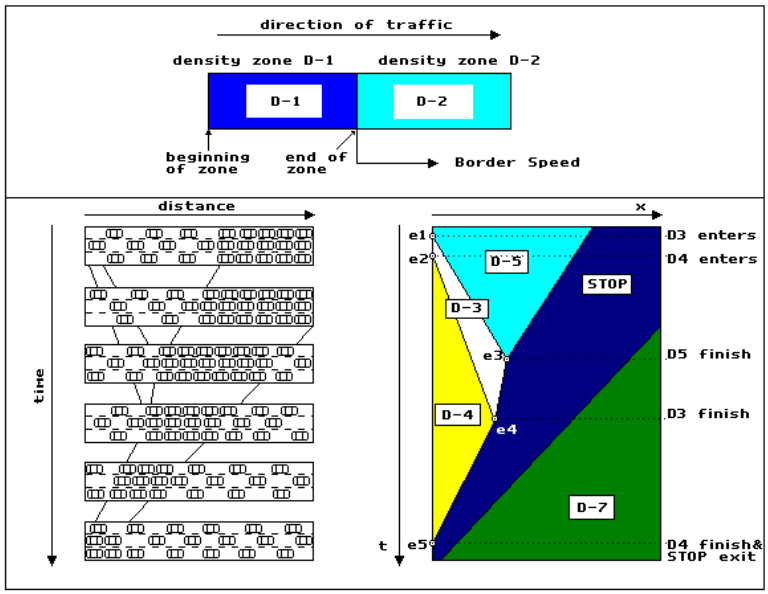


Fig 2-4 *Representación temporal-espacial del proceso de generación de una cola de vehículos como consecuencia del cambio de rojo a verde en un semáforo. La figura de la izquierda utiliza un método cuantitativo, mientras que la de la derecha un método cualitativo.*

El borde entre dos regiones vecinas de una calle se desplaza a una velocidad constante que puede ser calculada desde una ecuación de conservación de la masa aplicada al borde:

Ecuación 4 $$v = \frac{f_1 - f_2}{d_1 - d_2}$$

El conjunto de ecuaciones anterior muestra que la velocidad y el flujo pueden ser calculados a partir de la densidad. Por lo tanto, el comportamiento dinámico del tráfico puede ser caracterizado por la densidad de vehículos.

Este último valor de velocidad expuesto (ecuación 4), denominado *velocidad de frontera*, puede ser positivo o negativo. Un valor negativo indica que la dirección de avance del borde es contraria a la del movimiento de los vehículos entre ambas zonas. Resulta adecuado destacar que la velocidad de desplazamiento del borde no es la velocidad de movimiento de los vehículos de alguna de las dos regiones que forman el borde.

Una vez identificada la densidad como la herramienta básica para modelar el comportamiento del tráfico la siguiente tarea es identificar cuales de sus valores nos resultan más significativos. Para realizar esta tarea se establece un árbol jerárquico de valores, tal y como se muestra en la figura 2.5. Estos valores se asocian a unos determinados intervalos cuantitativos de densidad, velocidad y flujo, tal y como se muestra en la tabla 2.1.

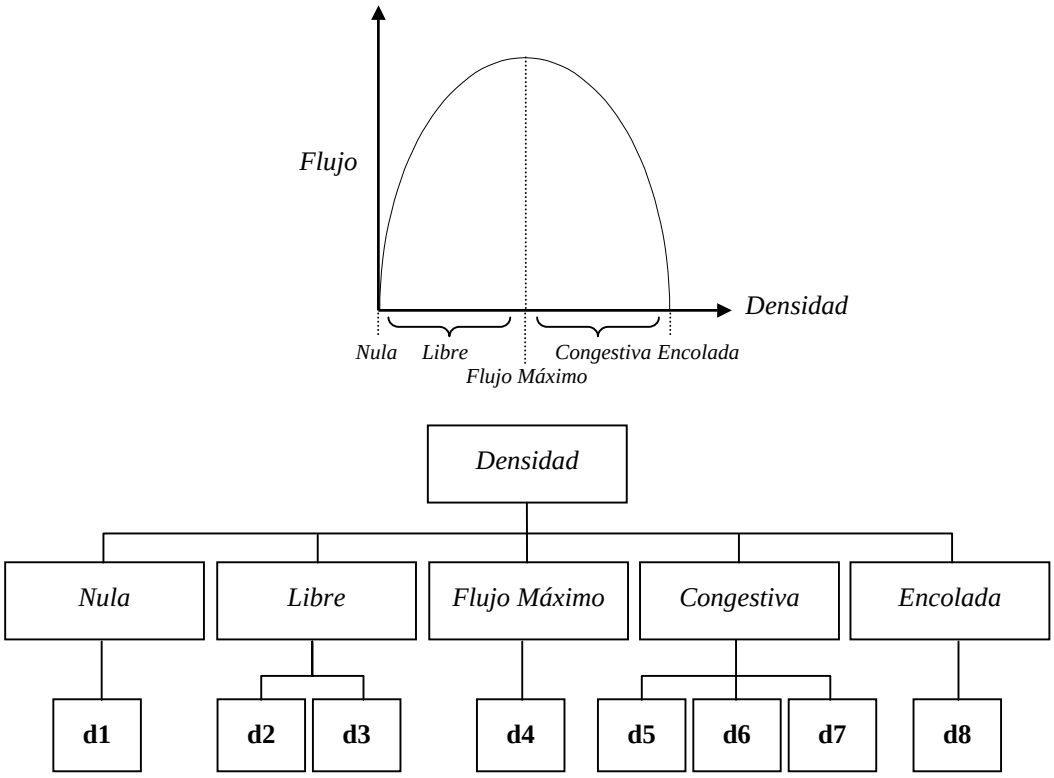


Fig 2-5 *Árbol jerárquico de valores cualitativos asociados a la densidad de vehículos*

En resumen, el comportamiento dinámico del tráfico se puede representar mediante la evolución de los bordes entre regiones limítrofes. Cuando dos de estos bordes colisionan se genera uno o más nuevos bordes y, por lo tanto, esta es una situación que produce un evento con un alto grado de significado. El conocimiento de la evolución del tráfico, por lo tanto, se puede representar como un conjunto de reglas que describen las clases diferentes de bordes que pueden generarse desde los estados cualitativos actuales.

Los valores de las velocidades de frontera se muestran en la tabla 2.2 (han sido calculados según la ecuación 4 y con los datos de la tabla 2.1)

Valor Cualitativo	*Velocidad Km/hora*	*Densidad vehículos/m*	*Flujo vehículos/seg*
d1	**[70, 65]**	**[0.000, 0.015]**	**[0.0, 0.3]**
d2	**[65, 54]**	**[0.015, 0.045]**	**[0.3, 0.7]**
d3	**[54, 44]**	**[0.045, 0.075]**	**[0.7, 0.9]**
d4	**[44, 35]**	**[0.075, 0.100]**	**[0.9, 1.0]**
d5	**[35, 26]**	**[0.100, 0.125]**	**[1.0, 0.9]**
d6	**[26, 16]**	**[0.125, 0.155]**	**[0.9, 0.7]**
d7	**[16, 5]**	**[0.155, 0.185]**	**[0.7, 0.3]**
d8	**[5, 0]**	**[0.185, 0.200]**	**[0.3, 0.0]**

Tabla 2.1 *Intervalos de valores de velocidad, densidad y flujo de vehículos asociados a cada valor cualitativo de la densidad.*

	d1	d2	d3	d4	d5	d6	d7	d8
d1	0	16.6667	13.8889	11.1111	8.3333	5.5555	2.7777	0
d2	16.6667	0	11.8056	8.7301	5.8333	2.7777	-0.1984	-2.9411
d3	13.8889	11.8056	0	4.6296	1.8518	-1.7361	-5.0	-7.4786
d4	11.1111	8.7301	4.6296	0	-0.9259	-5.5555	-9.127	-11.111
d5	8.3333	5.8333	1.8518	-0.9259	0	-12.5	-15.277	-15.476
d6	5.5555	2.7777	-1.7361	-5.5555	-12.5	0	-18.055	-16.666
d7	2.7777	-0.1984	-5.0	-9.127	-15.277	-18.055	0	-15.740
d8	0	-2.9411	-7.4786	-11.111	-15.476	-16.666	-15.740	0

Tabla 2.2 *Valores asociados a la velocidad de frontera entre dos intervalos consecutivos. Como se observa estos valores se pueden representar en forma de matriz cuadrada simétrica.*

2.4.1.3 Modelo cualitativo del tráfico urbano

En toda red de tráfico urbano se pueden identificar las siguientes clases de objetos:

- *Segmentos*, representan calles de sentido único sin intersecciones que las atraviesen. El comienzo de un segmento puede venir determinado por una

entrada o por un cruce, y su final puede ser una salida u otra intersección. Si una calle posee doble sentido de circulación, entonces esta calle se representa como dos objetos segmentos, uno para cada sentido permitido de la circulación.

- *Entradas*, los objetos de esta clase se consideran como objetos adyacentes a los comienzos de los segmentos con los que están conectados.

- *Salidas*, cumplen la función de punto de salida del tráfico y se consideran como objetos adyacentes a los finales de los segmentos con los que están conectados.

- *Cruces*, son los puntos de confluencia de dos o más segmentos, se corresponden con el concepto habitual de intersección física, e incluyen a los mecanismos que pueden alterar el modo de funcionamiento de la intersección mediante la modificación de los tiempos de verde permitidos a cada dirección

- *Controlador de idéntico ciclo*, especifican el tiempo total dedicado a conceder derecho de paso a todos los segmentos que confluyen en cada intersección, para todas las intersecciones a las que este controlador tiene acceso.

- *Control*, esta no es una clase, sino un objeto que representa el sistema de control y que almacena todas las órdenes que han sido enviadas a cada uno de los objetos del sistema.

Los objetos del sistema se corresponden con instancias de estas clases. Así, por ejemplo, en una red urbana se pueden identificar entradas (e_1, e_2, ..e_n), salidas (s_1, s_2, ..s_m), segmentos (l_1, l_2, ..l_k), intersecciones ($ifis_1$, $ifis_2$, ..$ifis_f$) y controladores de idéntico ciclo (cci_1, cci_2, ..cci_t).

Cada uno de los objetos del sistema posee una descripción estática y otra dinámica. La representación estática almacena la información que es constante para el objeto, por ejemplo, en un objeto de la clase segmento, la información referente al nombre del segmento, su longitud y su número de carriles (entre otras cosas) pertenece a su descripción estática. La representación dinámica se construye mediante el almacenamiento de los eventos significativos de su evolución. En los siguientes apartados se expone como desarrollar un modelo cualitativo de tráfico para ser aplicado a cada uno de estos objetos.

2.4.1.4 Representación dinámica cualitativa de un segmento

En un segmento el estado del tráfico en cada momento se puede describir como una colección de intervalos espaciales, en los que en cada intervalo la densidad de tráfico es constante. Según pasa el tiempo, las fronteras de cada intervalo se pueden ir desplazando, los intervalos pueden desaparecer cuando su longitud se va decrementando paulatinamente y llega a alcanzar el valor 0, y también pueden aparecer nuevos intervalos en las regiones fronteras de un segmento, como consecuencia de las

condiciones impuestas por objetos vecinos o bien por el propio comportamiento de los conductores.

Para calcular la velocidad de desplazamiento de cada borde se utiliza la ecuación 4, que está basada en la condición de equilibrio respecto a la densidad de vehículos en cada una de las regiones. Pero esta ecuación no representa de una forma completa todas las opciones posibles de desplazamiento. Por ejemplo, si la ecuación se aplica a un segmento en el que la densidad es máxima y el segmento siguiente presenta densidad nula, entonces la **ecuación 4** dará un valor nulo de velocidad de frontera. Este valor es correcto desde el punto de vista de la conservación de vehículos en ambas regiones, pero no es correcto desde el punto de vista del comportamiento de los conductores. Por lo tanto, esta ecuación debe de complementarse con un conjunto de reglas que modelen el comportamiento de los conductores.

Como norma general se genera una región intermedia entre dos regiones contiguas si se satisfacen las siguientes condiciones:

- La región intermedia debe de tener el valor d4 como estado de densidad cualitativa.
- La longitud de esa nueva región intermedia debe de crecer (en caso contrario esa región desaparecerá tan pronto como haya sido creada)
- El valor de la velocidad de frontera entre las dos regiones primitivas debe de estar entre los valores de las dos velocidades de frontera asociadas a la nueva región intermedia.

Estas condiciones se reflejan en el cumplimiento de las siguientes reglas:

Regla 1:

nuevo_evento(X, T) :-
    intervalo(l_j),
    velocidad_de_frontera_inicio(V_1, l_j),
    velocidad_de_frontera_fin(V_2, l_j),
    V is $V_2 - V_1$,
    V < 0
    calcular(X, T, l_j, V).

En esta regla el parámetro *X* representa una coordenada espacial, *T* es el intervalo temporal que existe entre el momento actual y el momento en el que sucederá el siguiente evento de este intervalo espacial, V_i son las velocidades de frontera y l_j representa el intervalo espacial para el que se calcula el siguiente evento. Esta regla sólo sucederá (tendrá éxito) si el intervalo espacial l_j desaparece después de un periodo finito de tiempo (figura 2-6)

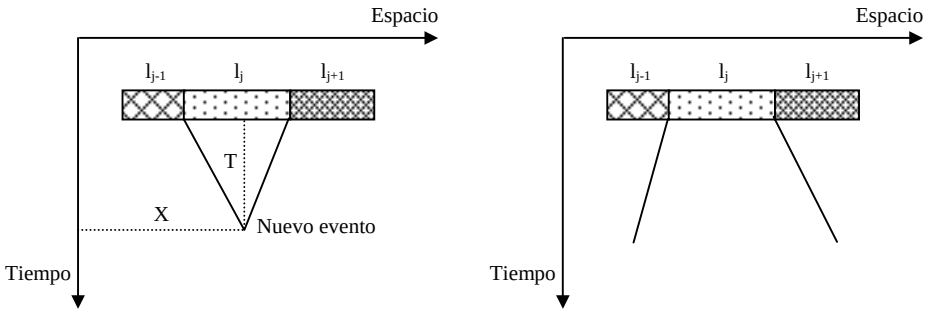


Fig 2-6 *La figura de la izquierda representa la situación en la que la regla nuevo_evento(X, T) se ejecuta con éxito, mientras que en la figura de la derecha la regla fracasa.*

La siguiente regla refleja el proceso de creación de nuevos intervalos cuando la velocidad de frontera no está ajustada al comportamiento de los conductores.

Regla 2:

nuevo_intervalo(l_k) :-

intervalo(l_j),

intervalo(l_t),

proximo_a(l_j, l_t),,

velocidad_de_frontera_inicio(V_1, l_t),

velocidad_de_frontera_fin(V_2, l_t),

velocidad-de_frontera(V_3, l_k),

$V_1 < V_3$,

$V_3 < V_2$,

calcular(l_j, l_t, l_k).

Esta regla sólo tendrá éxito cuando el comportamiento de los conductores no es consistente con las velocidades de frontera entre dos intervalos vecinos. En este caso se calcula un nuevo intervalo, l_k, a insertar entre los dos intervalos que hacen frontera, l_j y l_t, cuando l_k tiende a crecer y no desaparece instantáneamente.

Mediante el uso de ambas reglas se pueden deducir todos los posibles eventos futuros que van a suceder en un segmento mediante el conocimiento del estado inicial del segmento, tal y como se refleja en la figura 2-7

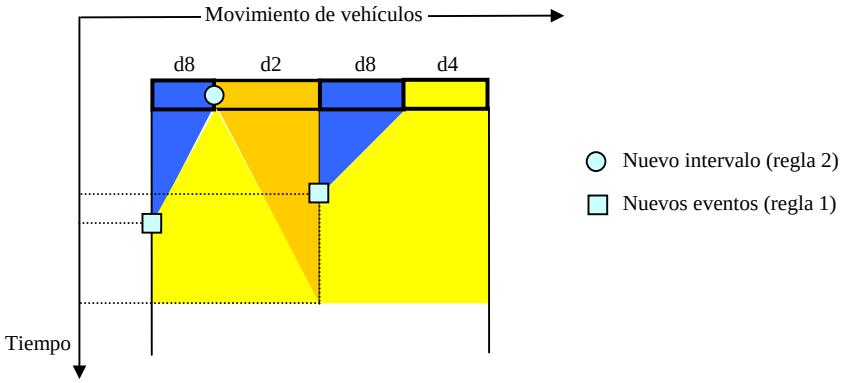


Fig 2-7 *Representación de la evolución de la densidad del tráfico en un segmento a partir de su estado inicial*

2.4.1.5 Representación dinámica cualitativa de una intersección

Las intersecciones son entidades cuyo estado en un cierto instante de tiempo está definido por un conjunto de estados de tráfico. Cada estado de tráfico representa el estado cualitativo final de un segmento de entrada a la intersección o el estado cualitativo inicial de un segmento de salida de la intersección. La siguiente figura (figura 2-8) representa el estado de una intersección ejemplo en dos instantes temporales donde cada uno de ellos está asociado a una fase diferente.

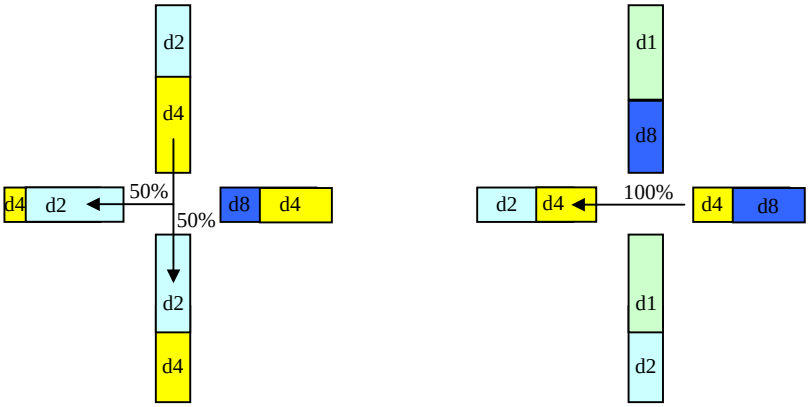


Fig 2-8 *Especificación de la densidad cualitativa en una intersección en dos instantes temporales diferentes correspondientes a la ejecución de fases diferentes. El porcentaje representa el tanto por ciento de vehículos que a partir de una entrada de la intersección se dirigen a una de sus salidas. Estos porcentajes se suelen agrupar en matrices denominadas de entrada-salida.*

Las intersecciones sólo pueden modificar su estado interno mediante acciones externas. Los estados de las luces de los semáforos y los valores cualitativos del extremo de cada segmento de la intersección son las entradas con las que trabaja el modelo cualitativo aplicado a la intersección. Cuando se recibe una nueva entrada se

ejecuta un proceso que determina el conjunto de posibles estados de tráfico futuro que son consistentes con las condiciones actuales de la intersección. De este conjunto se selecciona un elemento en función de dos heurísticas: el deseo de incremento de velocidad de los conductores y la selección de aquel elemento que minimice el número de cambios a realizar en el conjunto de estados del tráfico con relación a su estado inicial.

Los valores del conjunto de estados de tráfico que determinan el estado cualitativo de una intersección son linealmente dependientes. Esta relación se expresa por medio de la matriz entrada-salida que expresa el porcentaje de flujo de vehículos que se dirigen de una particular entrada a una particular salida en la intersección. Para cada posible fase en cada intersección existe una matriz de datos entrada-salida, como por ejemplo se muestra en la figura 2-8

El flujo de salida se calcula utilizando la matriz de entrada-salida correspondiente junto con los flujos de entrada de todas los segmentos de entrada en la intersección. No obstante, esta tarea no es tan directa puesto que el estado del tráfico no puede ser determinado únicamente con el flujo, debido a que cada valor del flujo de tráfico puede corresponderse con dos o más estados cualitativos de la densidad del tráfico. Esto es debido a que el flujo tiene una relación parabólica con la densidad (véase figura 5.2). Por lo tanto, este cálculo se tiene que completar con otro proceso que elimine esta ambigüedad.

El proceso que permite calcular cual va a ser la densidad cualitativa de vehículos en la intersección impone las siguientes restricciones:

- Para delimitar el número de posibles estados de tráfico se incluyen las relaciones de flujo indicadas en la matriz entrada-salida como restricciones.

- Los estados de las luces de los semáforos se incluyen como restricciones. Si una entrada a la intersección tiene su semáforo en rojo entonces se impone un flujo de entrada nulo para esa entrada, lo que implica que en el caso de que existan vehículos en ese segmento de entrada se empiece a generar una cola de vehículos.

- Se eliminan todos aquellos estados que son inestables, es decir, aquellos estados que provocan que la frontera formada por el estado cualitativo de algún segmento de la intersección y el nuevo estado calculado invada el espacio asociado a la intersección, provocando otro cambio de estado en la intersección. Por lo tanto el primero de los estados calculados para la intersección no era firme, sino inestable.

Después de aplicar estas reducciones en el conjunto de estados posibles para la intersección sólo quedan valores que son:

- consistentes con la matriz de entrada-salida

- consistentes con los estados de las luces de los semáforos de la intersección

- se expanden de la intersección hacia los segmentos

Para seleccionar de entre todos estos estados cual va a ser el que represente el estado de la intersección se aplican dos heurísticas:

- el flujo de vehículos siempre tiende a ser el mayor posible debido al comportamiento habitual de los conductores

- se debe de cumplir la condición de inercia, es decir, el número de cambios posibles en los estados de la intersección debe de ser el menor que se pueda.

Después de aplicar estas heurísticas sólo debe de quedar un conjunto de valores que deben de ser casi equivalentes entre sí, por lo tanto, se puede seleccionar cualquiera de ellos como valor a asignar a la intersección.

2.4.1.6 Representación dinámica cualitativa del control del sistema

La red urbana se particiona en áreas diferentes denominadas CCI (*C*ontroladores de *C*iclo *I*déntico). Cada controlador gestiona un conjunto de intersecciones de la red estableciendo para cada una de ellas sus tiempos de rojo y de verde para cada sentido de la circulación permitido en cada instante.

Todas las intersecciones de un CCI funcionan a un mismo ciclo de reloj, y cada CCI se controla enviándole, desde el control, los datos asociados a cual es su ciclo de trabajo, y los valores de coordinación y reparto actual para cada una de sus intersecciones.

Una de las direcciones de cada intersección se define como la *dirección principal*, y el comienzo del ciclo en cada intersección se establece que comienza cuando el semáforo de su dirección principal cambia a verde. Los valores del vector de repartos modifican el tiempo dedicado a verde en cada una de las direcciones de una intersección (es decir, modifican la duración de cada fase de la intersección), pero no pueden cambiar su instante de comienzo de ciclo (es decir, el instante de comienzo de la fase 1). Mediante este funcionamiento se puede establecer valores de coordinación entre intersecciones independientemente de los repartos que se estén utilizando en cada instante.

La finalidad de asignar valores de coordinación a cada intersección, que no es más que un valor de desplazamiento respecto del momento en que el CCI envía su señal de comienzo de ciclo y el comienzo del ciclo en la intersección, es el conseguir *ondas verdes*, es decir, que los semáforos que forman una determinada ruta cambien secuencialmente de rojo a verde según se van acercando los vehículos a cada una de las intersecciones, con lo que se busca conseguir un flujo de tráfico fluido en esas rutas.

En el modelo propuesto para el trafico sólo se permiten tres valores posibles para el CCI: *80*, *100* y *120* segundos; y el valor actual del ciclo sólo puede modificarse en el momento en que comienza. Por lo tanto, al discretizar los posibles valores del ciclo se puede modelar cada CCI como un objeto cuya línea temporal sólo almacena instantes de cambio de ciclo.

Las acciones que se pueden ejecutar sobre cada intersección son de dos tipos. Para cada intersección se define un conjunto de ocho posibles vectores de reparto ordenados del *1* (que representa aquel conjunto de valores que proporciona el tiempo máximo de verde permitido a la dirección principal) al *8* (conjunto de valores para dedicar el tiempo mínimo posible a la dirección principal). La coordinación se establece por medio de la asignación de un número entre *0* y *999* que representa el desplazamiento a realizar, en tantos por mil, del valor actual del tiempo de ciclo al que está funcionando la intersección.

La modificación de los valores asignados a cada intersección sólo se puede realizar en momentos concretos. En particular, si se desea modificar el valor de la coordinación en una intersección, entonces el controlador de la intersección debe de incrementar o decrementar su valor de tiempo de ciclo para esa intersección de una forma paulatina, evitando cambios bruscos que producirían un comportamiento anómalo en el tráfico. Una acción de este tipo suele emplear de *1* a *5* minutos en ejecutarse.

Un caso similar sucede cuando lo que se desea cambiar es el vector de reparto a aplicar en una intersección. Este cambio sólo se puede realizar al comienzo del ciclo en la intersección, pues en otro caso, se podría llegar a ejecutar tiempos de ciclos de mayor o menor duración que los que el CCI ha dado orden de utilizar para esa intersección.

2.4.1.7 Clasificación de Problemas de Tráfico.

El modelo cualitativo de tráfico urbano expuesto está desarrollado con el objetivo de servir de modelo *profundo* de razonamiento aplicable en sistemas basados en el conocimiento. Por esta razón, el modelo cualitativo se extiende con otro formalismo que permite la descripción e identificación de los problemas que pueden presentarse en el tráfico urbano. Este formalismo, el formalismo de representación general (GRF), es otro de los resultados del proyecto EQUATOR.

EL FORMALISMO DE REPRESENTACIÓN GENERAL (GRF)

El modelo GRF [Moreno, 93] está basado en el Cálculo de Eventos de Kowalski [Kowalski, 86]. Este formalismo permite realizar razonamientos sobre el tiempo mediante el uso de un subconjunto de las Cláusulas de Horn de la lógica de primer orden aumentadas con el mecanismo de “negación como fallo”. El formalismo GRF proporciona varias extensiones con respecto al Cálculo de Eventos, principalmente la temporalidad granular y el cambio continuo [Barber, 94] [Toledo, 96].

En este mecanismo, el conocimiento temporal del dominio se expresa por medio de ocurrencias de eventos. Cada uno de estos eventos se asocia de forma general a un

dominio temporal relacionado con su granularidad intrínseca. De esta forma, la aparición de un evento puede ser especificada explícitamente por medio de un intervalo acotado, cuyos extremos pueden no estar inicialmente definidos.

Los eventos se asocian a las propiedades mediante el establecimiento de unas relaciones de dominio intencionadamente definidas. Estas relaciones se representan por medio de los predicados[2] *inicia(Evento, Propiedad)* y *termina(Evento, Propiedad)* que definen los efectos de los eventos sobre las propiedades. Las instancias de los eventos y las propiedades se consiguen asociando a cada uno de los tipos de eventos y propiedades un punto o intervalo temporal. De forma coherente con las propiedades de persistencia de GRF, el predicado *max_cumple(Propiedad, [Comienzo, Fin])* busca los mayores intervalos temporales disjuntos en la que se cumple una determinada propiedad (que a su vez fue iniciada y terminada por dos eventos). El predicado *cumple_en(Propiedad, Tiempo)* permite realizar preguntas sobre las propiedades que son estables en un cierto tiempo.

El mecanismo GRF permite la especificación de restricciones métricas y de ordenación con eventos y propiedades. El predicado *antes_que(Evento1, Evento2)* actualiza las restricciones asociadas con los eventos especificados para especificar que *Evento1* debe de producirse antes que *Evento2*. La introducción de restricciones de tipo métrico se puede realizar mediante el predicado *distancia(E1, E2, Distancia)* donde *Distancia* representa la diferencia entre los tiempos en que aparecen los eventos *E1* y *E2*.

Los predicados *contradice(P1, P2)* y *antes(P1, P2)* establecen restricciones sobre las propiedades. El predicado *contradice(P1, P2)* establece que las propiedades *P1* y *P2* no pueden suceder simultáneamente, es decir, sus intervalos de validación deben ser disjuntos. El predicado *antes(P1, P2)* establece que la propiedad *P1* se debe de cumplir antes que la propiedad *P2*.

Según se ejecuta el modelo cualitativo de tráfico se van generando una serie de eventos asociados a situaciones singulares del tráfico como son:

- Eventos debidos a que una cola de vehículos alcanza un cruce.
- Eventos debidos a que una cola de vehículos desaparece de un cruce.
- Eventos asociados al comienzo de luz roja en un semáforo.
- Eventos asociados al comienzo de luz verde en un semáforo.
- Eventos de comienzo de nuevo ciclo para un CCI.

[2] En su nomenclatura original inicia/2 es initiates/2, termina/2 is terminates/2, max_cumple/2 es mholds_for/2, cumple_en/2 es holds_at/2, antes_que/2 es earlier_than/2, distancia/3 is distance/3, contradicts/2 es contradice/2 y antes/2 es before/2.

Estos eventos se pueden utilizar como datos básicos con los que construir información temporal de más alto nivel que representen situaciones problemáticas de tráfico.

Las distintas situaciones problemáticas del tráfico urbano que se pueden identificar con estos eventos se clasifican en diferentes niveles de abstracción [Moreno, 93]:

- Situaciones problemáticas elementales.
- Situaciones problemáticas temporales.
- Situaciones problemáticas espacio-temporales.

Las situaciones problemáticas elementales se corresponden a aquellos instantes en que una cola de vehículos en un segmento alcanza el cruce de su extremo inicial y algunos segmentos de entrada a ese cruce que se dirigen al segmento bloqueado tienen su semáforo en verde. Esta situación se denomina *cruce_cerrado* y el conjunto de propiedades y eventos que la constituyen son los siguientes (I denota el identificador de un cruce y Segmento es el identificador de un segmento):

```
inicia( alcanza_cola(I, Segmento), cruce_bloqueado(I, Segmento) ).
termina( desaparece_cola(I, Segmento), cruce_bloqueado(I, Segmento) ).
inicia( cambio_verde(Segmento), estado_verde(Segmento) ).
termina( cambio_rojo(Segmento), estado_verde(Segmento) ).
inicia ( (nuevo_ciclo(I), ciclo(I) ).
termina( (nuevo_ciclo(I), ciclo(I) ).
```

En este conjunto de propiedades se debe de satisfacer la restricción *contradice(estado_rojo(Segmento), estado_verde(Segmento))*.

La propiedad asociada a la situación *cruce_cerrado* se define por el siguiente axioma donde *SegmentoE* es cualquier segmento de entrada al cruce que sea perpendicular a *Segmento*

```
max_cumple(cruce_cerrado(I), [Com, Fin]) :-
        max_cumple(cruce_bloqueado(I, Segmento), [Com1, Fin1]),
        perpendicular(I, Segmento, SegmentoE),
        max_cumple(estado_verde(SegmentoE), [Com2, Fin2]),
        intersección([Com1, Fin1], [Com2, Fin2], [Com, Fin]).
contradice(cruce_cerrado(I), cruce_nocerrado(I)).
```

El segundo nivel de abstracción consiste en reconocer situaciones problemáticas temporales. Estas situaciones se corresponden con la persistencia en el tiempo de

situaciones problemáticas elementales. Para cada intersección se calcula en cada ciclo el intervalo más largo en que se cumplió la propiedad *cruce_cerrado*. Las siguientes reglas expresan como capturar estas propiedades

```
max_cumple(congestión_nula(I), [Com, Fin]) :-
        max_cumple(ciclo(I), [Com, Fin]),
        max_cumple(cruce_nocerrado(I), [Com, Fin]).

max_cumple(congestión_media(I), [Com, Fin]) :-
        max_cumple(ciclo(I), [Com, Fin]),
        cumple_en(cruce_cerrado(I, Tiempo1),
        Tiempo1 > Com, Tiempo1 < Fin,
        cumple_en(cruce_nocerrado(I), Tiempo2),
        Tiempo2 > Com, Tiempo2 < Fin.

max_cumple(congestión_completa(I), [Com, Fin]) :-
        max_cumple(ciclo(I), [Com, Fin]),
        max_cumple(cruce_cerrado(I), [Com, Fin]).
```

Las situaciones problemáticas espacio-temporales se definen sobre un conjunto de objetos, sus relaciones espaciales y sus situaciones problemáticas temporales detectadas. El problema de este tipo más importante es la situación denominada *bucle_cerrado* que consiste de un circuito de segmentos bloqueados entre sí.

Las reglas que expresan esta propiedad son las siguientes

```
max_cumple(bucle_cerrado(Segmentos), [Com, Fin]) :-
        circuito(Segmentos),
        max_cumple(todos_bloqueados(Segmentos), [Com, Fin]).

max_cumple(todos_bloqueados([]), _).
max_cumple(todos_bloqueados([Segmento|Rsegmentos]), [Com, Fin]) :-
        intersectados([Com, Fin], [Tiempo1, Tiempo2]),
        max_cumple(cruce_bloqueado(I, Segmento), [Tiempo1, Tiempo2]),
        max_cumple(todos_bloqueados(Segmentos), [Com, Fin]).
```

En estas reglas *Segmentos* representa a todos los segmentos que forman el *bucle_cerrado,* el predicado *circuito/1* obtiene los segmentos que forman un circuito en la red de tráfico urbana e *intersectados/2* es una restricción que establece que su primer

intervalo debe estar incluido en el segundo intervalo ya que lo que se busca es la situación en la que todos los segmentos están bloqueados al mismo tiempo.

Otra situación problemática, no contemplada en la clasificación original del proyecto EQUATOR, sucede cuando existe un camino abierto de segmentos bloqueados simultáneamente. Esta situación se denomina *camino_cerrado*, y las reglas que expresan esta propiedad son las siguientes [García, 00a]:

```
max_cumple(camino_cerrado(Segmentos), [Com, Fin]) :-
        no_circuito(Segmentos),
        max_cumple(todos_bloqueados(Segmentos), [Com, Fin]).


max_cumple(camino_cerrado(Segmentos), [Com, Fin]) :-
        max_cumple(bucle_nocerrado(SegmentosNB),
        no_circuito(Segmentos, SegmentosNB)
        max_cumple(todos_bloqueados(Segmentos), [Com, Fin]).
```

En la primera regla *Segmentos* representa a todos los segmentos que forman el *camino_cerrado*, el predicado *no_circuito/1* obtiene los segmentos que forman un camino abierto, es decir que no forman bucle, en la red de tráfico. Este predicado debe garantizar que proporciona los Segmentos que forman un camino de longitud máxima, puesto que, en caso contrario, se daría el caso de detectar muchas situaciones problemáticas con la propiedad *camino_cerrado* donde sólo hay una. De esta forma se garantiza que un *camino_cerrado* se reconoce como tal y no como la concatenación de varias situaciones de *camino_cerrado*.

La segunda regla expresa que todo circuito cerrado formado por segmentos en los que se detectan situaciones problemáticas temporales, pero que no comparten un mismo intervalo temporal de presencia, debe ser particionado en caminos no cerrados de longitud máxima que compartan un mismo intervalo temporal de congestión.

2.4.1.8 Simulación Cualitativa del Tráfico Urbano

El simulador de tráfico urbano desarrollado en el marco del proyecto EQUATOR integra el conjunto de reglas, heurísticas y de funcionamiento, expuestas en el apartado anterior para cada uno de los objetos representados en la red urbana. Este simulador ejecuta un ciclo de trabajo similar al ejecutado por el motor de inferencia del formalismo (λ, t):

1) Localizar el evento (o conjunto de eventos) más cercano al momento actual. Resulta sencillo de comprobar que antes de este instante todos los objetos tienen su línea temporal completamente determinada, y que después de ese instante existe al menos un objeto que tiene su línea temporal completamente libre.

2) Sólo los eventos del paso 1) pueden producir nuevos eventos. Estos nuevos eventos pueden ser de dos tipos: eventos internos, en los que un evento en un objeto producirá otro evento en el mismo objeto, y eventos externos, en los que un evento en un objeto producirá otro evento en otro objeto.

3) Se analiza cada uno de los eventos producidos en el paso anterior para determinar el conjunto de hipótesis de posibles estados futuros.

4) Se eliminan del conjunto de hipótesis aquellas que no pueden convertirse en nuevos estados. Las hipótesis restantes se convierten en nuevos eventos y se almacenan en la base de datos.

5) Se analizan el nuevo estado del tráfico obtenido para comprobar si existen situaciones problemáticas elementales o temporales.

Este ciclo de trabajo se ejecuta hasta que el evento asociado en el paso 1) tenga asociado un tiempo mayor que el tiempo final asignado para la simulación.

Para hacer que la ejecución de este ciclo de trabajo sea lo más rápido posible, sólo se considera la identificación de aquellos problemas que resulten sencillos de realizar según se vaya calculando la evolución cualitativa de cada objeto del sistema. Una característica importante de la clasificación de problemas de tráfico expuesta anteriormente es que resulta intrínsecamente constructiva, es decir, los problemas de una clase superior se construyen a partir de problemas de una clase inferior. Por este motivo resulta imprescindible que su evaluación sea realizada de abajo arriba, es decir, de las clases inferiores a las superiores.

Esta característica constructiva permite que en el ciclo de trabajo sólo se identifiquen las situaciones problemáticas elementales y temporales, puesto que las situaciones problemáticas espacio-temporales pueden ser analizadas posteriormente mediante el análisis conjunto de las situaciones problemáticas temporales identificadas durante el intervalo de ejecución del ciclo de trabajo.

El simulador cualitativo de tráfico urbano desarrollado implementa este esquema de funcionamiento como ciclo de trabajo básico. Este simulador precisa de una serie de datos de entrada relativos a su intervalo de ejecución [Tinicial, Tfinal] que son calculados por medio del siguiente predicado:

```
preparar_simulación(Tinicial, Tfinal, TDB) :-
  asignar_tiempos_fases(Tinicial, Tfinal),
  recoger_estado_segmentos(Tinicial, TDB1),
  recoger_estado_cruces(Tinicial, TDB2),
  append(TDB1, TDB2, TDB).
```

El resultado de este predicado es una lista, TDB, que representa el estado de cada objeto del sistema en el instante Tinicial. El predicado asignar_tiempos_fases(Tinicial,

Tfinal) inserta en la base de datos los tiempos de comienzo de cada fase para cada cruce, que se cumplen dentro del intervalo [Tinicial, Tfinal], en función de su reparto actual y del valor del ciclo del CCI al que pertenece.

El cuerpo principal del simulador es el siguiente:

```
simulador_tráfico(TDB, P, Tinicial, Tfinal, TDBfinal, Pfinal) :-
    acciones_internas(TDB, Tfinal, Tinterna, Hevent),
    acciones_externas(TDB, Tinicial, Tinterna, Tfinal, Hevent),
    primera_acción(Tacción, Tinterna),
    consolidar(Tinicial, Tacción, TDB, TDBaux, P, Pnew),
    append(TDB, TDBaux, TDBnew),
    ( Tacción ≥ Tfinal, !, TDBfinal = TDBnew, Pfinal = Pnew ;
      Tacción < Tfinal, !,
        simulador_tráfico(TDBnew, Pnew, Tacción, Tfinal, TDBfinal, Pfinal) ).
```

El simulador recibe una lista con el estado temporal cualitativo de cada objeto en el instante inicial, TDB, y como resultado devuelve la lista TDBfinal, que almacena la evolución temporal cualitativa para el intervalo [Tinicial, Tfinal] de todos los objetos del sistema como consecuencia de la ejecución de todos los eventos significativos surgidos en ese intervalo.

Los eventos se clasifican en función de las causas que los han producido. Si un evento se produce como consecuencia de la propia evolución de un objeto, y sin recibir la intervención externa de otros objetos, entonces ese evento se corresponde con una acción interna del sistema. Este es el caso de los eventos producidos en los segmentos como consecuencia del éxito de las reglas de las figuras 2-6 y 2-7. Si, por el contrario, el evento se produce como consecuencia de la ejecución de alguna acción de control, o por la intervención de otro objeto, entonces este evento se corresponde con una acción externa en el sistema. Por ejemplo, cuando un cruce va a cambiar de fase, o cuando el sistema de control ejecuta un cambio en el valor de ciclo de un CCI.

Los predicados acciones_internas(TDB, Tfinal, Tinterna, Hevent) y acciones_externas(TDB, Tinicial, Tinterna, Tfinal, Hevent) realizan estas tareas, almacenando en la lista Hevent los posibles cambios cualitativos que se pueden producir en los objetos asociados a los eventos de acciones internas del sistema.

La siguiente tarea que realiza el simulador, mediante la ejecución del predicado primera_acción(Tacción, Tinterna), es definir Tacción como el tiempo asociado a la aparición del evento, o eventos, más cercano.

Una vez identificados cuales van a ser los próximos eventos a ejecutar en el sistema, el simulador calcula los nuevos valores cualitativos para cada uno de los objetos asociados a esos eventos. Además, como consecuencia de la existencia de restricciones entre los objetos, también se deben de modificar todos aquellos objetos que se ven

indirectamente afectados por esta ejecución. El predicado consolidar(Tinicial, Tacción, TDB, TDBaux, P, Pnew) realiza esta tarea, desestimando todos aquellos valores que no pueden convertirse en nuevos valores cualitativos válidos. Por ejemplo, desestimando aquellos valores de estado inestables para una intersección (tal como se expuso en el apartado anterior).

Como consecuencia de la ejecución de este último predicado, se construye una nueva lista, TDBaux, que representa los nuevos valores cualitativos para todos los objetos del sistema que han sufrido alguna modificación en este ciclo.

Además, este predicado también calcula la nueva lista Pnew de problemas detectados hasta el instante Tacción a partir de la lista P que almacena los problemas identificados en los instantes anteriores a Tinicial.

Los problemas elementales de tráfico se calculan dentro del predicado consolidar/6 por medio del predicado problemas_elementales(LL, Lsn, Prob). Los parámetros LL, Lsn y P identifican los siguientes datos:

- LL: Porcentajes de vehículos que se dirigen de cada entrada a cada salida para la fase actual en el cruce que se está evaluando.

- Lsn: Estados cualitativos de los extremos finales de las calles de salida del cruce que se está evaluando para el momento actual.

- Prob: Lista de un único elemento de la forma [blocked([l_{i1},l_{i2},..l_{in}])] tal que cada l_{ij} es una calle de entrada al cruce evaluado en el que su calle de salida asociada para la fase actual está bloqueada (es decir, tiene d8 como valor cualitativo en su extremo final)

Los problemas temporales de tráfico se calculan también en el predicado consolidar/6 por medio del predicado problemas_temporales(PH,1,P,A,P1,Pnew). Los parámetros A, PH, P, P1 y Pnew identifican los siguientes datos:

- A: Cruce a evaluar.

- PH: Fase anterior del cruce A.

- P: Lista de los problemas detectados elementales y temporales anteriores al instante actual. Es una lista ordenada cronológicamente de los más cercanos a los más lejanos, P = [pevent(A_1,T_1,C_1), pevent(A_2,T_2,C_2),…,pevent(A_n,T_n,C_n)] / $T_1 \geq T_2 \geq ... \geq T_n$. Cada C_i es una lista de la forma [valor_congestión(VAL), blocked([l_{i1},..,l_{in}])] o bien de la forma [blocked([l_{i1},..,l_{in}])].

- P1: Problemas elementales detectados en este instante.

- Pnew: Lista que puede tener uno o dos componentes. Si el cruce A cambia a fase 1 entonces Pnew = [valor_congestión(VAL),P1], en caso contrario Pnew = P1.

La tarea a realizar consiste en analizar la evolución de la situación *cruce_cerrado* en el cruce *A* durante el ciclo inmediatamente anterior. Este análisis se resume en el valor asignado a la variable *VALOR*:

- VALOR = 0. En todo el ciclo anterior no ha habido calles bloqueadas, por lo tanto se corresponde con la propiedad *congestión_nula*.

- VALOR = 1. Resume varias situaciones: 1) en todo el ciclo anterior no ha habido cruces cerrados, 2) en T_i no hay cruces cerrados pero si en T_j, $T_i > T_j$, 3) en T_j hay cruces cerrados pero no en T_i, $T_j > T_i$, y 4) en el instante de comienzo del ciclo anterior no había cruces cerrados y ahora si los hay. Estas situaciones se corresponden con la propiedad *congestión_media*.

- VALOR = 10. En el instante de comienzo del ciclo anterior había cruces cerrados y ahora sigue habiéndolos. Esta situación se corresponde con la propiedad *congestión_completa*.

Por último, el simulador une las listas TDB y TDBaux para formar la nueva lista que representa la evolución del sistema desde el instante Tinicial. Si se ha alcanzado el tiempo final de la simulación, Tfinal, entonces el simulador devuelve TDBnew como lista final. En caso contrario, realiza otra iteración recursiva con la lista TDBnew como lista inicial y reduciendo el intervalo temporal inicial al nuevo intervalo [Tacción, Tfinal].

Para realizar todas estas operaciones temporales el simulador precisa implementar los siguientes métodos: comparaciones temporales (determinar si un instante de tiempo es mayor, menor o igual a otro instante de tiempo), pertenencia a un intervalo temporal, búsqueda del último evento que se ha ejecutado y el cálculo de eventos para el cambio continuo [Shanahan, 90] (para realizar la inferencia temporal).

Este simulador se ha implementado en Sicstus Prolog 3.7.1 y su ejecución en un PC con procesador Pentium II a 300 MHz y sistema operativo linux RedHat 5.1 con cuatro conjuntos diferentes de datos iniciales ha proporcionado los resultados mostrados en la figura 2-9.

Todos estos conjuntos de datos compartían el mismo número de elementos: 72 segmentos dirigidos, 22 cruces, 11 entradas, 17 salidas y 1 o 2 ccis. Las diferencias entre estos conjuntos se establecen variando los valores cualitativos iniciales de cada elemento junto con la modificación de alguna de las características ajustables de cada cruce: su vector de reparto a aplicar y su valor de coordinación.

En la gráfica de la derecha se observa que la unidad de medida del eje horizontal es en segundos, mientras que en el eje vertical es de milisegundos. Por lo tanto, el comportamiento del simulador con estos conjuntos de datos es lineal muy próximo al eje horizontal.

Int. de simul.	*datos1 (mseg)*	*datos2 (mseg)*	*datos3 (mseg)*	*datos4 (mseg)*
100	4148	5201	3705	4251
180	6924	7639	6421	6935
200	7696	8463	7060	7727
240	9367	9916	8641	9461
300	12203	12576	11216	12085
400	17593	17464	16298	16933
500	23362	22883	21837	22720

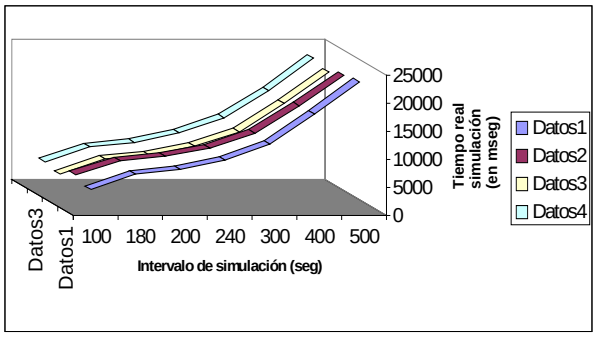


Fig 2-9 *Tabla de tiempos de la ejecución del simulador con cuatro conjuntos de valores iniciales diferentes y gráfico asociado a la misma.*

El simulador cualitativo de tráfico urbano expuesto, por lo tanto, presenta un tiempo de ejecución muy rápido. Un análisis de cual es el motivo de este comportamiento se puede realizar contando el número de potenciales estados cualitativos que el simulador calcula durante su ejecución.

Sea N el número de segmentos de la red, C el número de cruces, MaxFases el número de fases del cruce que tiene un mayor número de fases, MaxEnt el número de entradas del cruce que tiene un mayor número de entradas y MaxSal el número de salidas del cruce que tiene un mayor número de salidas. Supongamos que el intervalo de ejecución del simulador va a ser de 100 segundos, que todos los cruces de la red pertenecen al mismo CCI que funciona a un periodo de 100 segundos, y que no se realizan acciones de control externas en el sistema, es decir, no se cambia de valor de coordinación, ni de vector de reparto ni de tiempo de ciclo para el CCI.

En el caso de que en cada segundo se generase un evento en cada segmento (situación bastante remota y que se corresponde con el número máximo de eventos que pueden ocurrir), el número total de estados cualitativos calculados por el predicado acciones_internas(TDB, Tfinal, Tinterna, Hevent) sería de 100N. Como en 100 segundos cada cruce ejecuta 1 sóla vez cada una de sus fases, entonces el predicado acciones_externas(TDB, Tinicial, Tinterna, Tfinal, Hevent) calcula C × MaxFases cambios de fase como cota superior.

El predicado consolidar(Tinicial, Tacción, TDB, TDBaux, P, P2) calcula los nuevos estados cualitativos surgidos como consecuencia de la ejecución de las acciones internas (que ya se encuentran calculadas en la lista Hevent) y de las acciones externas, que en este supuesto se corresponden con cambios de fase. Por lo tanto se calculan, en el caso peor, C × MaxFases × (MaxEnt + MaxSal). El resultado de este análisis muestra, que aunque el tiempo de ejecución del simulador depende en gran medida de los datos iniciales de la red, se puede establecer la siguiente cota superior polinómica para cada 100 segundos de ejecución:

$$100N + C \times MaxFases + C \times MaxFases \times (MaxEnt + MaxSal)$$

Este resultado explica porqué es tan rápida la ejecución del simulador.

2.4.2 Prototipos con Planificación Estática en SBC de Segunda Generación.

Los primeros SBC de segunda generación desarrollados mantenían una arquitectura funcional en la que la ejecución de cada uno de sus componentes se realizaba siguiendo una misma secuencia fija preestablecida (como se puede observar en [Jackson, 90]). Entre los SBC de segunda generación con este tipo de arquitectura aplicados al control de tráfico urbano destaca el prototipo desarrollado en el proyecto EQUATOR.

EL DEMOSTRADOR DEL PROYECTO EQUATOR

El proyecto ESPRIT II EQUATOR (Environment for Qualitative and Temporal Reasoning), desarrollado desde el año 1989 hasta el año 1994, produjo una serie de métodos y módulos software para dar soporte a procesos de razonamiento cualitativo y temporal en aplicaciones industriales. La aplicación de Control de Tráfico Urbano se utilizó como demostrador, permitiendo validar el funcionamiento de los distintos módulos y la integración en un sistema CTU de aspectos funcionales no disponibles en aplicaciones prácticas.

Uno de los resultados fundamentales del sistema demostrador es la definición del modelo cualitativo que soporta la estructura de dos niveles del sistema CTU (expuesto en el apartado 2.4.1).

La arquitectura funcional del demostrador desarrollado en el proyecto EQUATOR se muestra en la figura 2-10. A continuación se expone una breve explicación de la función que desarrolla cada módulo.

El módulo de *preprocesamiento* convierte los datos provenientes de los sensores recogidos cada cinco minutos, en la descripción temporal/cualitativa basada en acontecimientos previamente descrita. La falta de datos para determinar completamente el estado del tráfico en cada instante hace que la construcción de este módulo no sea tan sencilla como pudiera parecer (que, como ya se ha apuntado en el capítulo anterior, es uno de los problemas más importantes con los que tiene que adaptarse un sistema CTU). En este módulo la conversión de la información se realiza tomando en cuenta los datos actuales, los anteriores, las acciones de control tomadas, y utilizando el modelo para generar la información que haga consistente toda la información anterior.

El módulo de *detección de problemas* analiza el estado actual del mundo para reconocer la existencia de problemas de tráfico. Esta tarea la realiza usando conocimiento de los que es un problema, codificado en reglas que se aplican para detectar el problema a diferentes niveles de granularidad espacial y temporal [Martin, 93]. Un problema muy puntual en el tiempo puede no considerarse como tal, y por otra parte, un problema que afecte a una zona se ha de detectar como tal, y no como una suma de problemas sobre los componentes espaciales de la zona.

El *módulo de diagnosis* realiza un análisis para distinguir si la presencia de un problema se debe a malas regulaciones de tráfico o si por el contrario, puede ser originado por un incidente (situaciones tales como un mal estacionamiento de un vehículo, un accidente, etc.) La distinción es importante, puesto que las acciones a tomar en cada caso son diferentes (en la actualidad un sistema algorítmico reacciona absurdamente ante una cola de vehículos provocada por un accidente dando más tiempo de luz verde en esa dirección, situación absurda dado que pueden incluso no poder salir por ese punto).

El *módulo de predicción* utiliza el modelado cualitativo del tráfico para predecir problemas futuros que pudieran producirse. Esta funcionalidad es muy importante para un sistema CTU dado que cuando se detectan los problemas éstos ya llevan un cierto tiempo de existencia y hasta que se toman las acciones de control adecuadas y éstas entran en acción hay un desfase temporal importante durante el cual el problema sigue creciendo y extendiéndose a cruces vecinos.

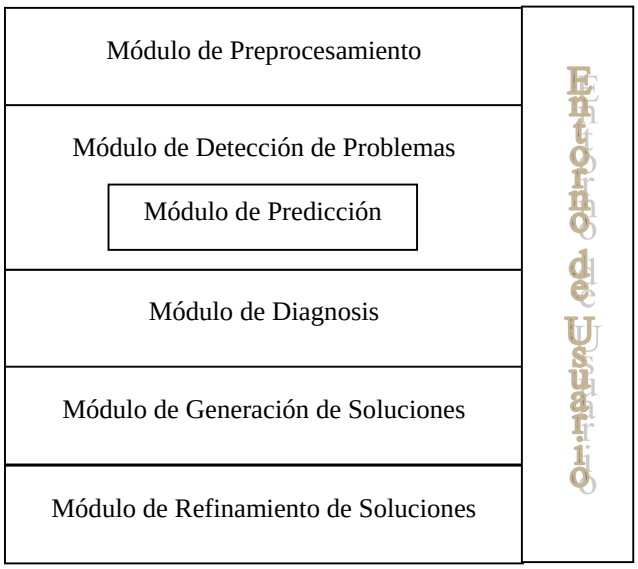


Fig. 2-10 *Estructura del demostrador del proyecto EQUATOR*

El *módulo de generación de soluciones* calcula cambios en las regulaciones de los semáforos como respuesta a los problemas actuales detectados, y de forma que estos problemas no se trasladen a otras zonas de la ciudad. Este módulo contiene una gran cantidad de reglas y está construido sobre una herramienta EQUATOR, el TTMS (Temporal Truth Maintenance System) que presenta un bajo rendimiento.

El módulo de *refinamiento de soluciones* actúa sobre la solución generada anteriormente para mejorarla. Por ejemplo, si como respuesta a un problema dado se ha reducido el tiempo de verde en un trozo de un itinerario o la entrada al mismo, esto se puede aprovechar para incrementar el tiempo de verde en ciertas calles transversales, no como respuesta al problema dado sino como mejora de la solución obtenida.

Todos estos módulos actúan de forma secuencial, y el usuario recibe una muestra de la evolución del sistema mediante el *interfaz de usuario*.

La velocidad de proceso del sistema, en su momento sobre un 20% de la ciudad de Valencia, da respuestas a mitad del tiempo real sobre una estación de trabajo HP9000/730. Este nivel de prestaciones hizo necesario plantearse como nueva línea de trabajo la definición e implementación de los elementos teóricos y funcionales desarrollados en el EQUATOR en una arquitectura funcional en la que se permita romper la secuencialidad en la ejecución de cada uno de los módulos descritos anteriormente, así como su adaptación a su desarrollo y programación en arquitecturas paralelas en las que todos los módulos puedan acceder a una base de datos temporal común en la que residen todos los datos de la aplicación.

2.4.3 Prototipos con Planificación Dinámica en SBC de Segunda Generación.

Los dos modelos con planificación dinámica más frecuentemente utilizados[3] para construir prototipos de SBC de segunda generación son los modelos de pizarra y, más recientemente, los modelos que implementan sistemas multiagentes.

2.4.3.1 Definición del Modelo de Pizarra.

La arquitectura de pizarra [Engelmore, 88], [Corkill, 91], [Carver, 94], [Garcia, 96a] es una arquitectura de sistemas basados en el conocimiento que fue descrita inicialmente por Newell e implementada por vez primera en el sistema de reconocimiento del habla *Hearsay II* [Erman, 80], [Nii, 86], [Nii, 88], [Velthuijsen, 92]. La mejor forma de definir el concepto que soporta a la arquitectura de pizarra consiste en describir la metáfora en la que se basa [Newell, 62] “se dispone de varios especialistas, cada uno de ellos con un determinado conocimiento específico, para resolver un problema cooperativamente. Los especialistas se disponen alrededor de una pizarra que utilizan para el intercambio de hipótesis y de datos. Cuando algún especialista escribe algún dato sobre la pizarra, el resto de especialistas deciden si pueden utilizar esa información de una manera que contribuya al proceso de resolución del problema. Este proceso se sigue realizando mientras no se obtenga una solución del problema, siempre y cuando existan especialistas activos, es decir, el proceso no quede parado debido a que los especialistas no puedan contribuir al proceso de solución.”

La arquitectura de pizarra implementa este comportamiento como paradigma general de resolución de problemas. El papel de los especialistas se asigna a distintos programas denominados *fuentes de conocimiento*. La *pizarra* consiste en una base de datos estructurada y global sobre la que se disponen las hipótesis y datos activos en cada momento. Otro componente típico de la arquitectura de pizarra es su elemento *controlador*; su papel consiste en determinar que peticiones de ejecución procedentes de diferentes, o de las mismas, fuentes de conocimiento tienen permiso de ejecución en cada instante.

[3] Estos modelos son utilizados como arquitecturas funcionales en otras muchas aplicaciones, no solamente en SBC de segunda generación.

Las entradas almacenadas en la pizarra representan a los resultados intermedios generados durante el proceso de resolución del problema. Estas pueden tener conexiones con cualesquiera otras entradas formando lo que se denomina *islas de solución*, representando soluciones parciales del problema. La presencia de estas islas de solución y la prioridad de su utilización en el proceso de resolución del problema permite reducir drásticamente el tamaño del espacio de búsqueda de las soluciones.

La pizarra, como ya se ha comentado, es una base de datos estructurada y global. Esta estructura se realiza fundamentalmente por dos razones [Carver, 94]: para mejorar los mecanismos de acceso de la pizarra y las prestaciones globales del sistema. La pizarra realiza básicamente dos funciones. La primera consiste en servir como almacén para representar el estado de la solución en cada instante. La segunda función que realiza consiste en servir como medio de comunicación entre los distintos componentes del sistema. Las entradas se pueden disponer en la pizarra siguiendo distintos tipos de organizaciones, como por ejemplo, los ya clásicos niveles y paneles. El grado más alto de organización de las entradas consiste en definir y utilizar más de una pizarra para tratar con los diferentes problemas que se pueden distinguir en la aplicación concreta. Por ejemplo, con este tipo de organización se puede utilizar una pizarra para tratar con el dominio de aplicación del problema concreto mientras que otra pizarra se puede emplear para tratar con el problema del control de la aplicación.

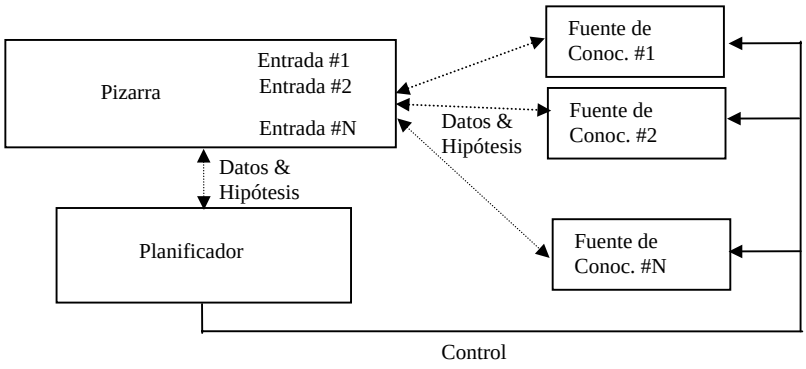


Fig. 2-11 *Estructura de un Sistema Experto basado en el modelo de pizarra*

Las fuentes de conocimiento modelan el conocimiento del experto. Generalmente están compuestas de dos partes: condición y acción. La parte condición especifica las condiciones de trabajo en las que se pueden ejecutarla fuente de conocimiento, es decir, los datos que deben estar presentes en la pizarra para que pueda ejecutarse la parte acción. La componente acción decide que acciones puede realizar la fuente de conocimiento. Este conocimiento puede estar implementado utilizando cualquier método de procesamiento: basado en reglas, orientado a objetos, simbólico, declarativo, etc. Una de las principales características de la arquitectura de pizarra es la facilidad con que diferentes técnicas de razonamiento pueden ser integradas en un único sistema.

El *planificador* en un sistema de pizarra se emplea para dirigir el modo en que va a trabajar el sistema. Fundamentalmente ha de cumplir dos objetivos: la construcción incremental de soluciones (basándose en la generación de islas y su empleo preferente) y la implementación de un comportamiento oportunístico. Este último objetivo implica que en cada ciclo de ejecución del sistema, el planificador debe de seleccionar, basándose en un conocimiento heurístico, para ejecución la fuente de conocimiento que más aporte al proceso de resolución.

El modelo de pizarra es un modelo en constante evolución en el que destacan tres líneas principales de investigación [Shapiro, 92]: control, prestaciones y distribución. Varios sistemas también emplean arquitecturas específicas de control para implementar métodos de resolución de problemas en tiempo real [BarberF, 94]. Las principales características de diseño en tales sistemas son la ejecución concurrente de fuentes de conocimiento, la ejecución interrumpible de la ejecución de tareas de baja prioridad, y el diseño de un planificador adaptativo que permita equilibrar el proceso de resolución atendiendo tanto a la calidad de la solución buscada como al tiempo necesario para su obtención.

2.4.3.2 Sistemas CTU Basados en el Modelo de Pizarra

El modelo de arquitectura de pizarra ha sido empleado con bastante frecuencia para el desarrollo de prototipos de SBC para la ayuda a la toma de decisiones en un sistema de control de tráfico urbano. Un gran número de estos, todos ellos basados en la selección y posterior ajuste ligero del plan de señales que mejor se adapte a la situación del tráfico detectada, han sido definidos e implementados en el marco del proyecto DRIVE. Como ejemplo característico de este tipo de sistemas se procede a describir las principales características del modelo desarrollado por [Ambrosino, 97].

IUTCS (PROYECTO DRIVE)

Este prototipo de SCTU, desarrollado bajo el proyecto DRIVE, realiza una división de las distintas operaciones de control de tráfico que tiene que realizar el sistema. Cada una de estas operaciones se implementa en un módulo (fuente de conocimiento) distinto. La comunicación de información entre estos módulos se realiza a través de una pizarra, que también sirve como medio de almacenamiento para representar la situación actual del tráfico de forma accesible a todos ellos.

La arquitectura del sistema IUTCS consta de los siguientes elementos:

- Una fuente de conocimiento, denominada *fuente de conocimiento de datos*, cuya función consiste en realizar una visión más completa de los datos proporcionados por el sistema (es decir, de los datos proporcionados por los sensores de tráfico)

- Una fuente de conocimiento, denominada *fuente de conocimiento de análisis de datos*, que evalúa las características del estado del tráfico según la información de la que disponga –es decir, por los datos del sistema, los datos inferidos a partir de estos y de un conjunto de indicadores del comportamiento del tráfico en la red– para reconocer cual es la situación de tráfico, para detectar eventos críticos en el sistema y para realizar la diagnosis de la situación actual del tráfico.

- Una fuente de conocimiento, *fuente de conocimiento de predicción del tráfico*, que permite realizar una predicción del estado del tráfico a corto y a medio plazo.

- Una fuente de conocimiento de control, *fuente de conocimiento de control del tráfico*, que evalúa de forma continua el nivel de prestaciones obtenidos con la actual estrategia de control y que decide qué acciones de control son las más adecuadas para aplicar en cada instante (es decir, esta fuente de conocimiento intenta implementar un comportamiento oportunístico).

- Una memoria de trabajo compartida, la *pizarra central*, que resulta accesible por todas las anteriormente expuestas fuentes de conocimiento. En la pizarra se almacenan todos los datos referentes al dominio del problema (la representación del estado del tráfico urbano) así como la información precisa para tratar con el problema del control de la aplicación, es decir, las acciones de control que se están ejecutando junto con su evaluación respecto a un modelo cualitativo del comportamiento actual y previsto para el tráfico.

El prototipo IUTCS realiza un proceso de razonamiento basándose en un modelo teórico del mundo real. Este modelo se mantiene en memoria central y se actualiza utilizando la información procedente de los sensores en cada ciclo de adquisición de datos junto con el resultado del proceso de razonamiento realizado por las distintas fuentes de conocimiento del sistema.

Cada una de las fuentes de conocimiento del sistema realiza las siguientes funciones:

FUENTE DE CONOCIMIENTO DE DATOS

Las tareas fundamentales que realiza esta fuente de conocimiento consisten en suministrar datos de tráfico procedentes de los bucles de espira disponibles en la red urbana y en intentar inferir datos de flujo, es decir volúmenes de tráfico y tasas de ocupación, para aquellos segmentos en los que no haya disponibilidad de datos de sensores. Esta fuente de conocimiento proporciona los siguientes servicios:

- Comprobación de la validez de los datos recogidos de los sensores activos; y

- La deducción de los valores de flujo y de ocupación en el caso de sensores defectuosos o de segmentos en la red sin sensores.

Para poder realizar estas tareas, esta fuente de conocimiento incluye un modelo de la propagación del flujo de tráfico a través de la red. Este modelo integra datos tanto sobre el conocimiento topológico del entorno urbano como conocimiento histórico estadístico sobre el comportamiento del tráfico en intersecciones y en segmentos. Por ejemplo, datos como los correspondientes a los porcentajes de giro en cada uno de los sentidos posibles en una intersección, los patrones de comportamiento del tráfico en los detectores, etc.

La tarea de deducción de volúmenes de tráfico se lleva a cabo mediante el establecimiento de una red de interdependencias entre todos los puntos de la red para los que se poseen datos reales (sensores activos) y los puntos para los que se tiene que deducir esos valores (sensores virtuales).

Las intersecciones proporcionan los elementos básicos con los que modelar las interacciones del flujo por medio de la definición y establecimiento de un conjunto de restricciones. Los valores medidos se propagan a través de este conjunto de restricciones mediante una aproximación basada en restricciones para poder realizar el proceso de deducción de los valores desconocidos. También se emplean heurísticas para poder deducir posibles valores de ocupación partiendo de los valores de flujo actuales, la capacidad de las calles y el último valor medido (o deducido) de la tasa de ocupación.

FUENTE DE CONOCIMIENTO DE ANÁLISIS DE DATOS

La tarea principal consiste en detectar condiciones de tráfico significativas y en realizar estimaciones sobre la situación global del tráfico. Esta fuente de conocimiento contiene dos pasos de razonamiento básicos:

- La evaluación cualitativa del conjunto de indicadores del comportamiento del tráfico; y

- la diagnosis de los estados de congestión, es decir, la identificación de cuales pueden ser los elementos que provocan la congestión y su posterior expansión.

Para realizar esta evaluación de la situación del tráfico en la red se implementa un sencillo sistema de razonamiento basado en reglas con encaminamiento hacia delante. Esencialmente, para evaluar las condiciones del tráfico a un nivel local, es decir en segmentos e intersecciones, se utilizan dos indicadores básicos..

Los flujos de tráfico en cada segmento se clasifican de acuerdo a su *grado de saturación v/c*, donde *v* es el flujo actual de la calle y *c* es su capacidad. Este parámetro indica lo cerca que está el segmento de alcanzar su capacidad máxima y proporciona, por lo tanto, una medida del tiempo disponible de verde.

Para calcular las condiciones de tráfico en una intersección se emplea un indicador global denominado *factor de capacidad (CF)*.

Ambos parámetros, el grado de saturación v/c y el factor de capacidad, utilizan unos indicadores cualitativos que se definen por medio de niveles de carga en un espacio cuantitativo para el correspondiente parámetro.

Un ejemplo de los valores definidos para cada uno de estos parámetros, ajustados a las características de la ciudad Alemana de Hamburgo, se muestran en las tablas siguientes:

Flujo	Ocupación	Nivel de carga
v < 20% c	or < 0.2	muy bajo
20% < v < 50% c	or < 0.2	bajo
50% < v < 80% c	or < 0.2	infrasaturado
80% c < v	or < 0.2	saturación
	0.2 < or < 0.3	saturación
	0.3 < or < 0.4	precongestión
	or < 0.4	congestión

Factor de Capacidad	Nivel de carga
CF > 5	muy bajo
5 > CF > 2	bajo
2 > CF > 1.25	infrasaturado
1.25 > CF > 1	precongestión
1 > CF	congestión

Además de determinar estos valores también resulta interesante calcular cual está siendo la evolución de los cambios en estos parámetros. Por este motivo se definen las tendencias de flujo basándose en tres niveles de distinción: incremento, decremento, estable. Por ejemplo, la descripción cualitativa completa del flujo en un segmento, en un determinado instante de tiempo, puede presentar la siguiente forma: <bajo,+> lo que significa “flujo de tráfico bajo con tendencia a incrementarse”

Fuente de Conocimiento de Predicción del Tráfico

Esta fuente de conocimiento realiza una predicción cualitativa del tráfico a corto plazo (5-10 minutos). Para realizar esta predicción esta fuente de conocimiento se basa en: (1) el conocimiento de un gran volumen de información histórica (patrones de detección) en cualquier situación del tráfico del día; y (2) en las series temporales de flujo de datos medidas por los sensores del sistema. Los volúmenes de datos actuales son contrastados con una base de datos de información histórica siguiendo una descripción del comportamiento habitual de la calle con el fin de identificar el mejor modelo que permita deducir volúmenes de información. La predicción del tráfico presenta, por lo tanto, dos pasos. El primero consiste en una tarea de interpretación, y el

segundo en una tarea de razonamiento simbólico sobre una identificada tendencia de flujo, es decir, el sistema calcula:

- La identificación de la mejor tendencia histórica, y
- la extrapolación del volumen de tráfico previsto

El prototipo IUTCS realiza un proceso de razonamiento superficial. Los criterios para evaluar y razonar sobre patrones históricos son básicamente heurísticos y son establecidos según sea el conocimiento específico del área bajo observación. La suposición general en el sistema es que el tráfico evoluciona típicamente, lo que a veces no resulta muy correcto. Sin embargo, la debilidad intrínseca del método se compensa por el requisito de operar en un nivel cualitativo; es decir, en un primer nivel, los flujos de tráfico previstos son asumidos por los niveles de carga definidos cualitativamente y resultan mantenidos por la fuente de conocimiento de análisis de datos.

La Fuente de Conocimiento de Control de Tráfico

La base de conocimientos de control de tráfico modela el proceso de decisión que gestiona dinámicamente la toma de acciones de control para poder actuar ante condiciones de tráfico cambiantes. Las acciones de control se disponen en forma de una biblioteca predefinida de planes de señales para intersecciones. Los servicios que aporta son los siguientes:

- evaluar las prestaciones de la red de tráfico bajo las actuales directrices de control
- identificar en las intersecciones críticas los posibles cambios locales para realizar su control
- evaluar los efectos de los cambios locales propuestos en intersecciones adyacentes y en grupos de intersecciones
- seleccionar y proponer aquellos cambios que resulten compatibles con el control actual para tratar con problemas locales que han sido identificados.

El proceso de decisión se realiza teniendo en cuenta distintos elementos, algunos de los cuales están disponibles para las demás fuentes de conocimiento. Entre estos elementos destacan:

- una medida de prestaciones, que se evalúa continuamente para monitorizar las prestaciones del control actual con respecto a la presente situación de tráfico;
- un conjunto de parámetros cuantitativos y cualitativos de parámetros de tráfico, que conforman un dibujo del comportamiento del tráfico a medio

plazo (proporcionado por las fuentes de conocimiento de análisis de datos y de conocimiento de datos);

- la focalización de las acciones de control sobre segmentos críticos e intersecciones, es decir, distribuir la capacidad en las intersecciones, niveles de flujo, etc. (proporcionado por la fuente de conocimiento de análisis de datos);

- la predicción de la situación del tráfico a medio plazo (proporcionada por la fuente de conocimiento de predicción de tráfico).

La fuente de conocimiento de control de tráfico se basa en la evaluación de una *medida de prestación*, proporcionando una medida del éxito de los valores actuales del plan de señales con respecto a la actual situación del tráfico.

El proceso de búsqueda del mejor plan de señales a aplicar en un momento determinado es un proceso con un alto coste computacional, se trata de un problema de búsqueda en un espacio de soluciones exponencial. Para intentar conseguir reducir el tamaño de ese espacio se establecen una serie de restricciones heurísticas. Ejemplo de tales heurísticas son las siguientes:

```
Regla 1:   SI    CF > 2 en todas las intersecciones claves
           ENTONCES   restringir   los   planes   de   señales
                      candidatos a aquellos con menor tiempo de
                      ciclo
Regla 2:   SI    CFi < 2 en todas las intersecciones clave
           ENTONCES   restringir los planes de señales
                      candidatos a aquellos con mayor tiempo de
                      ciclo

Regla 3:   SI    CFi < 2 en la intersección i con t > n*T
                      (T =tiempo de muestreo; n=3)
                 Y    dirección  de  coordinación    actual  =
                      dirección  de  coordinación  del  plan
                      candidato
                 Y    CFi > 2
           ENTONCES SPk es un candidato
           SINO  restringe la selección de planes de señales a
                 aquellos con mayor tiempo de ciclo
```

REPRESENTACIÓN DEL CONOCIMIENTO

Para implementar la pizarra se ha utilizado un mecanismo de representación orientado a objetos junto con un proceso de razonamiento, proporcionando un modelo del entorno de tráfico accesible por las fuentes de conocimiento. La pizarra, básicamente implementa una estructura a tres niveles:

- Estructura de red, incluyendo todos los parámetros físicos y topológicos relevantes

- El estado del tráfico en la red urbana en diferentes instantes temporales, incluyendo indicadores cualitativos y datos de tráfico cuantitativos

- El estado del control, es decir, los parámetros de las señales de tráfico activas y de los planes de señales disponibles

Mediante el acceso a elementos individuales de la red se puede obtener información sobre el estado local del tráfico y sobre el estado de las señales. Las representaciones de los segmentos y de las intersecciones contienen también, en una memoria local, los eventos de tráfico que han sucedido en cada uno de ellos. Por lo tanto, mediante el acceso a un elemento, se puede obtener una historia (de longitud variable) de las condiciones de tráfico locales.

El conocimiento superficial se integra en el entorno IUTCS por medio de una representación basada en reglas. Un lenguaje de reglas bastante simple, basado en la habitual sintaxis antecedente/consecuente, se utiliza como medio para expresar el conocimiento empírico empleado por las distintas fuentes de conocimiento:

```
(defrule MEDIUM-OUTBOUND
  (:agent DA)
  (:ruleset TRAFFIC-SITUATION)
     (if
        (and(loadlevel j32-j42(on-of ‘precongested’congested)
        (eq (congestion-cause j32-j42) j42-j8042)
        (load-trend (all j22-j32 j13-j14) ‘increasing)
        (load-level (all-others (dlinks envir)
                    ‘(j13-j14 j14-j15 j22-j32))
                         (lower= ‘undersaturated)))
     (then
        (change-traffic-situation (traf-state envir)
                                      ‘MEDIUM-OUTBOUND))))
```

2.4.3.3 Definición del Modelo de Sistemas Multiagentes.

El término multiagente es aplicado generalmente a un sistema que está, o puede considerarse que está, compuesto de múltiples agentes interactivos. Los sistemas multiagente desarrollados utilizan una definición de agente muy amplia que comprende desde agentes inteligentes completamente autónomos (tales como usuarios) hasta entidades de programación relativamente simples (tales como reglas o conjuntos de reglas) [Gerhard, 99]. Por lo tanto, a medida que el término multiagente ha venido siendo empleado, su significado ha resultado más y más confuso, incorporando a su significado una gran variedad de resultados, aproximaciones y fenómenos, hasta el punto en el que no hay conferencia de sistemas multiagente en que no se trate la cuestión de cómo definir el término multiagente [Durfee, 94], [Müller, 97], [Singh, 98].

Russell y Norving en [Russell, 95] proponen una definición de agente de acuerdo a la siguiente fórmula: *agente = arquitectura + programa*. En su misión como *programa* un

agente establece relaciones entre percepciones y acciones exhibiendo las siguientes características:

- Autonomía: Un agente debe de poder tomar la iniciativa y de ejercer un grado de control no trivial sobre sus acciones.

- Colaboración: Un agente debe de colaborar con otros agentes para mejorar la calidad de sus decisiones.

- Flexibilidad y versatilidad: En función del estado detectado en el entorno el agente debe de seleccionar qué acciones ejecutar.

- Historia temporal: En entornos que cambian dinámicamente, el agente debe de almacenar una representación de su evolución como información adicional a aplicar con la que conseguir sus objetivos.

- Representación del conocimiento: Un agente debe de utilizar el modelo de representación del conocimiento más adecuado en cada instante de su ejecución.

- Comunicación: El mecanismo básico de inferencia contiene como tarea fundamental el intercambio de información con otros agentes, construyendo la solución o la interpretación del estado del sistema de forma continua y distribuida.

Cada agente en un sistema multiagente representa un determinado conjunto de experiencia y técnicas de resolución de problemas. El propósito del sistema multiagente consiste en coordinar las habilidades, conocimiento, planes y experiencia de distintos agentes para conseguir un objetivo común de alto nivel.

El otro componente del agente, su *arquitectura*, se define como [Maes, 94] una metodología particular para la construcción de agentes. La arquitectura expresa el mecanismo por el que un agente se divide en un conjunto de módulos y el modo en el que estos módulos deben de interaccionar entre si. El conjunto de todos estos módulos junto con sus interacciones tiene que proporcionar una respuesta a la cuestión de cómo determinar las acciones y el estado interno futuro del agente partiendo de su actual estado interno junto con los datos proporcionados por los sensores del sistema. La definición de una arquitectura de agente incluye la definición e implementación de las técnicas y algoritmos que sirven de soporte a esta tecnología.

La definición e implantación de sistemas multiagente es un área que, desde hace ya unos años, está generando unas amplias expectativas de éxito y en el que se está realizando una extensa y productiva investigación.

2.4.3.4 Sistemas CTU Basados en Modelos de Sistemas Multiagente

La aplicación de sistemas multiagente en sistemas de control de tráfico urbano es un campo de aplicación relativamente reciente que ha proporcionado, hasta el momento, un conjunto relativamente reducido de prototipos [Irgens, 97]. Uno de estos prototipos es el sistema KITS (*K*nowledge based and *I*ntelligent *T*raffic control *S*ystem) [KITS, 96]. Su definición surgió como un proceso de refinamiento natural del sistema IUTCS desarrollado en la primera fase del proyecto DRIVE (explicado en el apartado anterior). Este prototipo ha sido aplicado y evaluado en cuatro ciudades europeas con distintas características de tráfico para las que el prototipo IUTCS no resultaba adecuado debido al tamaño de las redes urbanas implementadas.

KITS (PROYECTO DRIVE II)

El prototipo KITS implementa una arquitectura denominada "*Agentes Cooperativos Supervisados*" [Irgens, 97] que está construida aplicando características de prototipos previamente desarrollados. El aspecto clave que se introduce en este nuevo sistema consiste en la división de la red urbana en distintas áreas, cada una de las cuales está compuesta de elementos que precisan de un mismo análisis y proceso de actuación.

Este prototipo surgió como resultado del análisis de la imposibilidad de tratar de proporcionar una solución global a cada posible escenario de tráfico en cada instante. A medida que se gestionaban áreas cada vez de mayor tamaño las características de las soluciones propuestas para tareas similares en distintas áreas precisaban de la ejecución de distintos métodos. Por ejemplo, la predicción del tráfico no se podía realizar de la misma manera en una arteria principal que en un segmento aislado de la red urbana.

Este análisis provocó la introducción de dos nuevos conceptos: los *actores* y los *supervisores*. El prototipo KITS hereda del sistema IUTCS la definición del modelo de tráfico y de sus componentes funcionales, es decir, sus fuentes de conocimientos (denominados en este nuevo contexto como *agentes*).

Un *actor* es una entidad imperativa – entendiendo como tal a una entidad encargada de realizar el proceso de razonamiento y que, por lo tanto, ejecuta acciones- que tiene como objetivo principal gestionar el tráfico de un área. Un *actor* tiene bajo sus órdenes a un conjunto completo de agentes cuyas funciones y definiciones se corresponden con las fuentes de conocimiento utilizadas en el prototipo IUTCS.

Un *supervisor* es también una entidad imperativa cuya función consiste en controlar y gestionar el tráfico en un área mayor que la gestionada por un *actor*. Un *supervisor* tiene bajo sus órdenes a un conjunto de actores que gestionan el tráfico en cada una de las sub-áreas en que se puede dividir el área gestionada por el supervisor. También se definen e implementan supervisores que gestionan a otros supervisores.

Las entidades imperativas en el sistema se organizan en una jerarquía que resulta determinada por la división en áreas y sub-áreas definida por el modelo de tráfico.

Estas entidades imperativas, a diferencia de lo que ocurría en el prototipo IUTCS, ya no trabajan forma aislada realizando modificaciones en el estado del modelo de tráfico, sino que además implementan un mecanismo de colaboración más directo que el proporcionado por el mecanismo de pizarra. Para realizar este proceso de colaboración se define e implementa un protocolo de comunicación específico junto con unos canales por los que realizar esta comunicación.

La definición de este protocolo recoge algunas de las características descritas en el proceso de especificación de las tareas principales del sistema: el mantenimiento del modelo de tráfico, el control de su flujo y la gestión del sistema. Estas tareas se organizan en una jerarquía debido a que las entidades imperativas también lo están al establecer que los nodos más importantes gestionan problemas generales de tráfico en áreas mayores que los nodos menos importantes, cuya gestión está dedicada a problemas específicos de tráfico en áreas pequeñas.

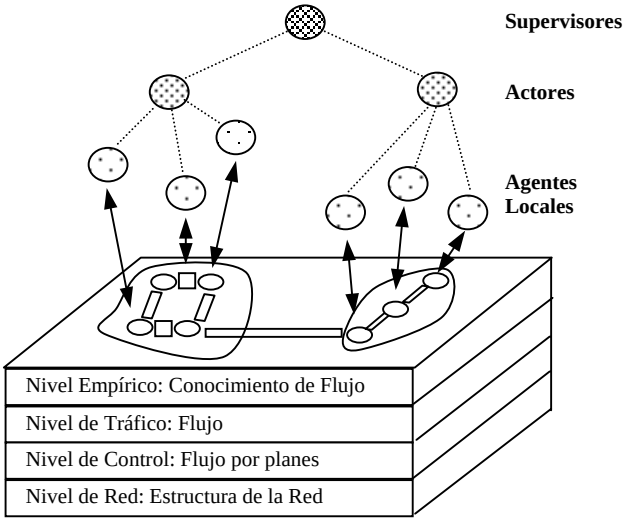


Fig. 2-12 *Arquitectura del sistema KITS*

Las tareas a ejecutar pueden ser enviadas desde las entidades de nivel inferior (agentes) hasta entidades de nivel superior (actores), siendo tarea de éstas últimas reenviarlas hasta las adecuadas entidades de nivel inferior para su adecuada ejecución.

Además de estas entidades imperativas, el sistema trata también con entidades declarativas que contienen la descripción del modelo implementado de tráfico. Estas entidades declarativas se clasifican según una jerarquía de cuatro niveles:

- Nivel de Red: En este nivel, el más bajo de la jerarquía, se especifican los elementos del sistema que componen la topología de la red. Estos elementos se clasifican en tres grupos: objetos de área, primarios y secundarios. Se

utilizan para expresar restricciones de flujo fuertes, es decir, la disposición y relación topológica entre los distintos componentes de la red urbana.

- Nivel de Control: En este nivel se especifica la biblioteca de planes de señales disponible, y que sirve como referencia para la construcción de nuevos planes de señales adecuados a cada situación respecto del estado del tráfico.

- Nivel del Comportamiento del Tráfico: Se especifican las representaciones cuantitativas y cualitativas con las que deducir los niveles de flujo de tráfico en cada elemento del sistema.

- Nivel Empírico: En este nivel, el más alto en la jerarquía, se especifican las reglas de conocimiento del experto en tráfico. Estas reglas contienen la expresión del conocimiento empírico de una forma jerárquica y distribuida entre los diferentes objetos en función de sus características y comportamiento. Esta organización del conocimiento permite además razonar sobre metaconocimiento, como por ejemplo ocurre en la regla siguiente:

SI La regla #A especifica que una intersección tiene alguna carencia Y
Existe un conflicto en el área que contiene a esa intersección
ENTONCES Eliminar la regla #A del conjunto de reglas aplicables.

2.5 Conclusiones

Los sistemas basados en el conocimiento que han sido sucesivamente aplicados al control de tráfico urbano explotan de una forma directa dos factores: la experiencia aprendida en la implementación y evaluación de prototipos previos (tanto respecto a las técnicas que utilizan para gestionar las funcionalidades del sistema como a la representación utilizada para expresar el conocimiento sobre la evolución del tráfico urbano) y la, cada vez mayor, disponibilidad de computadores más rápidos (por lo que se pueden obtener las respuestas en un menor periodo de tiempo así como ampliar sus capacidades mediante la incorporación de nuevas funciones).

Los primeros SBC desarrollados utilizaban un mecanismo de razonamiento *superficial*, grandes conjuntos de reglas causa/efecto implementadas en una arquitectura funcional clásica de sistemas de producción. Esta forma de representar el conocimiento pronto evidenció sus limitaciones (como así quedó expuesto en el apartado 2.3.3). Como consecuencia, diversos autores propusieron la incorporación de un modelo del comportamiento del tráfico en los SBC aplicados a un sistema CTU, es decir, la incorporación de un mecanismo de razonamiento *profundo*.

El modelo temporal-cualitativo de tráfico urbano desarrollado en el proyecto EQUATOR (expuesto en el apartado 2.4.1) presenta una serie de características que lo hacen adecuado para ser adoptado como mecanismo de razonamiento profundo:

- El tiempo de ejecución del motor de inferencia del formalismo (λ, t) es muy bajo y con crecimiento lineal, lo que lo hace aplicable en problemas que

precisen respuestas en cortos periodos de tiempo y en dominios de aplicación compuestos por un gran número de componentes.

- La base de datos temporal generada por su ejecución presenta dos características importantes: es compacta (puesto que el simulador trabaja sólo sobre acontecimientos significativos) e incorpora etiquetas temporales con las que eliminar parte de la indeterminación existente en una simulación cualitativa.

- Junto con el mecanismo de representación general (GRF) permite definir e identificar lo que es una situación problemática de tráfico urbano a distintos niveles de granularidad espacial y temporal.

Este modelo fue incorporado en el demostrador del proyecto EQUATOR evaluándose muy positivamente [Moreno, 93]. No obstante, los diferentes módulos del demostrador actuaban de una forma secuencial por lo que era necesario implementar algún tipo de arquitectura funcional con la que se pudiera obtener una respuesta en tiempo real y en la que sus componentes pudiesen relacionarse de una forma menos estática.

Otros autores comenzaron a desarrollar prototipos de SBC de segunda generación aplicados al control de tráfico urbano utilizando arquitecturas con planificación de ejecución dinámica para sus componentes. La primera de estas arquitecturas funcionales adoptada seguía un modelo de pizarra (apartado 2.4.3.2). En este tipo de sistemas toda la información de tráfico es accesible por cualquiera de sus componentes. Por lo tanto, estos prototipos están orientados hacia la implementación en computadores paralelos con memoria compartida.

Sin embargo, la mayor parte de los prototipos desarrollados se ejecutaban en computadores monoprocesadores, puesto que, aunque existía una buena disponibilidad de computadores paralelos con memoria compartida como productos comerciales, las herramientas necesarias para su programación no se habían desarrollado al mismo nivel.

Al mismo tiempo que se dedicaban grandes esfuerzos en desarrollar mejores herramientas de programación paralelas fueron cogiendo más importancia los sistemas distribuidos en los que se disponía de un espacio de memoria distribuida virtualmente compartida. Como consecuencia de estos esfuerzos, se desarrolló un conjunto de paradigmas de programación que servían tanto para los computadores paralelos con memoria compartida como con memoria distribuida (como por ejemplo los modelos de programación *MPI* o *linda*). Además, los computadores con memoria distribuida tenían unos costes de adquisición y mantenimiento sustancialmente inferiores a sus "equivalentes" computadores con memoria compartida.

Otro inconveniente de los prototipos basados en el modelo de pizarra es la fuerte dependencia de todos los componentes respecto de la integridad y fiabilidad de la información almacenada en la pizarra, así como su gran generalidad, lo que impide predecir cual puede ser la secuencia de ejecución de los componentes del sistema en

cualquier momento [Irgens, 97]. Por lo tanto, se hacía necesario aplicar una nueva arquitectura funcional en la que se redujese de una forma significativa las dependencias de datos y de resultados de ejecuciones entre sus componentes principales.

La siguiente arquitectura funcional con planificación dinámica recientemente aplicada en el desarrollo de sistemas CTU sigue un modelo de arquitectura multiagente (apartado 2.4.3.3). Sus componentes funcionales, *agentes*, poseen un alto grado de independencia y la comunicación entre ellos se realiza mediante mecanismos de paso de mensajes. Además, no se ha de olvidar que los sistemas CTU deben de cumplir con la finalidad de servir como soporte en el que poder integrar nuevas funcionalidades, por lo que este modelo, al permitir integrar componentes con una mayor independencia respecto a su ejecución, facilita la consecución de este objetivo.

A estos factores hay que añadir la creciente disponibilidad de sistemas de computadores distribuidos con buenas prestaciones y coste razonable junto con la enorme y creciente incidencia de todos los aspectos relativos a las tecnologías de la información y de las telecomunicaciones. Como consecuencia se puede pronosticar que este modelo de arquitectura funcional todavía puede proporcionar nuevos prototipos más potentes a corto plazo.

PARTE II

ARQUITECTURAS MULTIAGENTE PARA LOS SISTEMAS DE CONTROL DE TRÁFICO URBANO

CAPÍTULO 3
EL PROTOTIPO MASHGREEN SM 94-97

Índice del capítulo 3

3.1 Breve Descripción del Sistema Secuencial

El prototipo MASHGREEN SM surge como una evolución del demostrador desarrollado en el proyecto EQUATOR [EQUATOR, 94]. Una primera fase en la construcción de este nuevo prototipo es la conversión de los módulos del demostrador en agentes inteligentes. Por lo tanto, las tareas que realizaba cada uno de estos módulos se ve alterada con la introducción de nuevas tareas y el establecimiento de nuevos modelos de coordinación. La finalidad del prototipo MASHGREEN SM es, por lo tanto, el desarrollo de un sistema de ayuda a la toma de decisiones en control de tráfico urbano capaz de ser ejecutado en un computador paralelo con memoria compartida.

Las principales funciones que este sistema tiene que realizar son las siguientes:

- Visualización y cálculo de la evolución del sistema de forma continua. Los datos procedentes de los sensores del sistema se van acumulando y sólo resultan disponibles para el sistema cada cinco minutos, lo que hace que durante cada intervalo de cinco minutos la evolución del sistema se represente por medio de los datos proporcionados por el simulador cualitativo de tráfico. Cuando los datos del sistema están disponibles entonces se analizan y se ajusta el estado del sistema a los resultados de este análisis, procediendo de nuevo a otro nuevo bloque de simulación de cinco minutos.

- En el supuesto de que no existan incidentes cual puede ser la evolución del estado del tráfico a corto y medio plazo.

- Detección e identificación de problemas de tráfico actuales y futuros (previstos ante la ausencia de incidentes). Durante el proceso de simulación (del estado presente o futuro), el sistema puede detectar problemas debido a una mala regulación de los semáforos. En el proceso de análisis de los datos de los sensores el sistema puede determinar incidentes contrastando estos datos con los que ha obtenido del proceso de simulación.

- Búsqueda de los mejores vectores de reparto que minimicen el número de problemas detectados para cada cruce en el supuesto de que no existan incidentes de tráfico.

El sistema también debe permitir la ejecución de acciones de control sobre los cruces o sobre el CCI. Estas acciones pueden proceder de dos fuentes diferentes: El usuario determina por su cuenta la ejecución de una acción de control, o el sistema implementa los vectores de reparto más apropiados cada vez que estos se calculan. Estos modos de ejecución son ajustables dinámicamente por medio de una interfaz de usuario. En resumen, el sistema presenta dos modos de funcionamiento:

- Interno al sistema, en el que desde la interfaz se decide si se implementa o no de forma automática las propuestas calculadas para el estado actual del tráfico.

Estas acciones sólo se ejecutan en determinados momentos, en particular, en el momento en el que el agente de solución ha terminado de calcular la propuesta que, debido a la gran cantidad de operaciones que conlleva, sólo se realiza una vez cada cinco minutos.

- Externo, en el que el usuario realiza peticiones de ejecución de acciones independientemente de las propuestas del sistema. Por ejemplo, el usuario puede decidir si cambia el valor de coordinación de un cruce en función de su propio conocimiento sobre la red urbana que esta controlando.

Las acciones de control internas se ejecutan en un solo paso, es decir, si un cruce va a cambiar de vector de reparto, acción que sólo se puede realizar en el instante en el que el cruce va a comenzar su primera fase, entonces este cambio se realiza de forma inmediata. En el caso de las acciones de control externas, se puede decidir si estas acciones de control se realizan en un solo paso o de forma gradual, es decir, mediante el incremento o decremento paulatino de la actual acción de control para conseguir ese valor final deseado. Las acciones de control externas pueden ser de tres tipos: 1) modificación del vector de reparto de un cruce; 2) modificación del valor de coordinación de un cruce; y 3) modificación del valor de ciclo para cada CCI del sistema.

Todas estas funciones las realizan unos componentes especializados denominados *agentes* [García, 96]. El agente de *preproceso* es el encargado de que en todo momento la representación de la evolución del tráfico en el sistema sea coherente con su evolución real. Para ello utiliza principalmente dos métodos: la conversión de datos de flujo y ocupación de sensores de espira en datos cualitativos de la densidad, y la simulación cualitativa del tráfico para mantener una visión lo más ajustada posible del mundo real cuando no se dispone de estos datos. Los datos proporcionados por este agente resultan fundamentales para la ejecución del resto de agentes del sistema, por lo que es muy importante que estos datos sean calculados muy rápidamente. Por otra parte, el sistema debe de mantener el máximo de tiempo posible coherencia de la representación del estado del tráfico respecto del estado de tráfico real y, por lo tanto, cuando el usuario ejecuta una acción de control externa el sistema debería ejecutarla simultáneamente con la del mundo real. Estos dos objetivos resultan contradictorios entre sí, por lo que para buscar cual debe de ser el tiempo de ejecución más adecuado para el agente de preproceso hay que buscar una solución de equilibrio. Además, esta solución de equilibrio también servirá para deducir cuál será el tiempo dedicado a cada ciclo de ejecución básico en todo el sistema.

El agente de *predicción* realiza una estimación de cual puede ser la evolución del tráfico futuro a partir del estado del tráfico actual. Este agente abre una ventana temporal cuya longitud es un parámetro acotado superiormente por petición del usuario del sistema. Es decir, se fija el valor máximo deseado para la estimación, y en función de la cantidad de tiempo asignado para este proceso el agente de *predicción* estima hasta donde pueda llegar.

El agente de *detección* analiza el estado del tráfico actual (proporcionado por el agente de preproceso) y futuro (proporcionado por el agente de predicción) para determinar cuales son las situaciones debidas a malas regulaciones en los cruces. Estos análisis los suministra al agente de *diagnosis* (para que los compare con los detectados por medio de los datos proporcionados por los sensores) y al agente de *solución* (para que calcule la mejor asignación posible de vectores de reparto a cada cruce mediante la minimización del número de situaciones problemáticas detectadas). Por último, el agente de *refinamiento de soluciones* recoge la solución proporcionada por el agente de solución para intentar mejorarla en sus efectos laterales.

El agente de *interfaz* es el encargado de visualizar el estado del tráfico en cada momento y de recibir las peticiones de acciones de control (internas y externas) del usuario.

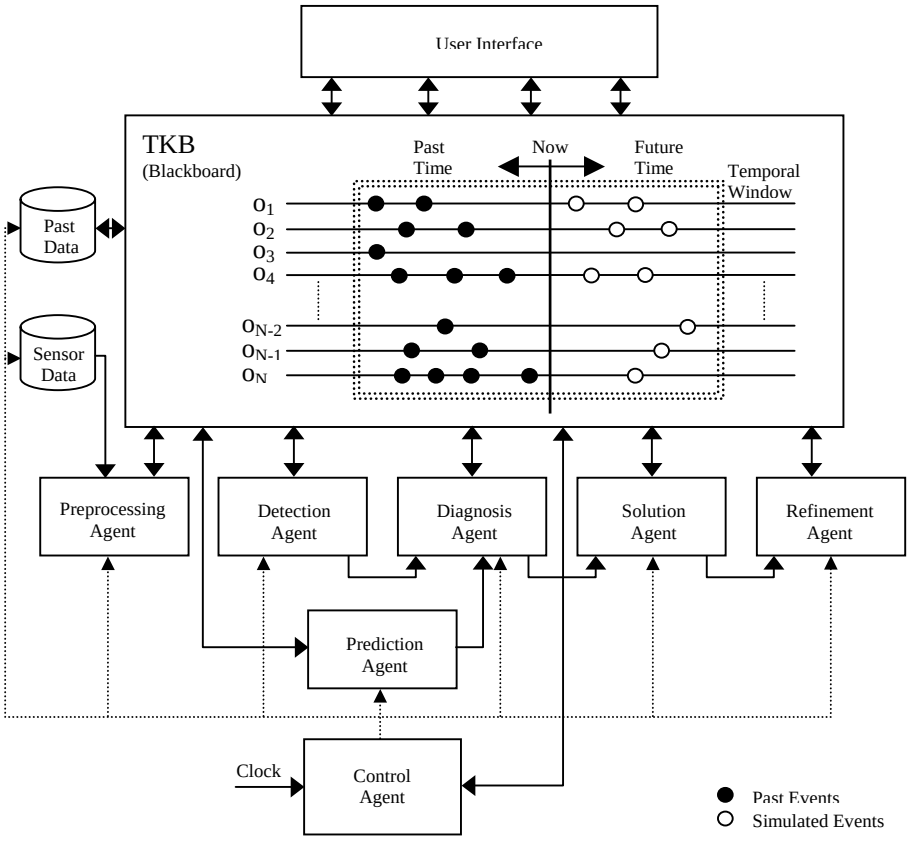


Fig. 3-1 *Arquitectura Funcional del sistema MASHGREEN SM. Las líneas punteadas representan acciones de control mientras que las líneas planas representan intercambio de mensajes*

La idea fundamental en este prototipo consiste en desarrollar estos agentes de forma que trabajen en paralelo concurrentemente sobre una base de conocimientos temporal que realizará las funciones de pizarra. Esta base de conocimientos contendrá, para cada objeto del sistema, los acontecimientos pasados, presentes y futuros (previstos) ocurridos durante su evolución. Esta información estará representada utilizando el modelo temporal cualitativo desarrollado en el proyecto EQUATOR.

En la base de conocimientos temporal se abre una ventana dividida en tres secciones: pasado, momento actual y simulación de futuro. Esta ventana se irá desplazando temporalmente según la dirección de la línea temporal, e irá depositando en disco aquellos datos que ya no son necesarios por ninguno de los componentes del sistema y que, por lo tanto, deben de quedar fuera de la actual ventana temporal. El contenido de estas tres secciones reflejará el conjunto de datos necesarios para que cada agente del sistema pueda realizar su trabajo [García, 96].

De esta forma el agente de preproceso trabajará con los datos provenientes de los sensores y con la parte del pasado para generar el estado actual del tráfico. El agente de detección de problemas actuales y el agente de diagnosis trabajarán sobre la parte del pasado y la parte actual. El agente de predicción trabajará sobre toda la ventana, en su fase de generación de futuro sobre la parte actual y futura, y en la fase de detección de problemas futuros sobre toda la ventana temporal. Los agentes de generación de soluciones y de refinamiento trabajan sobre la parte actual y futura.

Además de esta comunicación a través de los datos de la pizarra, los diversos agentes se intercambian mensajes entre ellos con las conclusiones que se vayan obteniendo, siendo este intercambio de alto nivel, en el sentido que se comunica información mucho más abstraída que la de la pizarra, que recordemos también tiene toda ella un significado cognitivo, como por ejemplo los problemas presentes o futuros o los planes que se vayan generando.

El sistema utiliza los recursos de un agente de *control* para mediar entre las diferentes peticiones de ejecución de distintos, o incluso del mismo, agentes. Este agente de control decide la duración del ciclo de ejecución básico al que el resto de agentes en el sistema (excepto el agente de interfaz) tienen que adaptarse.

En los siguientes apartados se describen cada uno de los componentes del prototipo MASHGREEN SM. Se observa, de la exposición anterior, que estos componentes están íntimamente relacionados entre sí, por lo que para poder especificar de una forma completa a cada uno de ellos se precisa del conocimiento completo del resto. Por este motivo se especifica primero el agente de control, puesto que en su definición se establecen los criterios generales respecto a la ejecución y descripción del resto de componentes del prototipo.

3.2 El Agente de Control

La tarea principal del agente de control es permitir la ejecución del número máximo de agentes de forma que la diferencia que exista entre la evolución del tráfico real y la representada en el sistema sea lo menor posible [García, 98].

Como ya se ha indicado previamente conseguir estos dos objetivos simultáneamente no es sencillo puesto que son contradictorios. Por lo tanto, se ha buscado una solución de compromiso que permitiese una ejecución adecuada en el sistema. Esta solución consiste en ejecutar ciclos de simulación en el sistema durante el tiempo en el que no se poseen datos de los sensores. Al utilizar este método los datos proporcionados por el agente de preproceso están rápidamente disponibles para su uso por otros agentes, pero durante el tiempo en el que el sistema está realizando este ciclo, el estado del tráfico real puede no coincidir con el estado del tráfico representado en el sistema si el usuario ha ejecutado alguna acción de control externa. No obstante, estas acciones de control no se realizan de una forma continua, sino que son acciones esporádicas del usuario. Este es el motivo por el que se decide realizar esta ejecución a intervalos, cuya longitud, denominada t_{ciclo}, establecerá el tiempo máximo de espera del sistema hasta que se ajusten las evoluciones de tráfico real y simulada [García, 98].

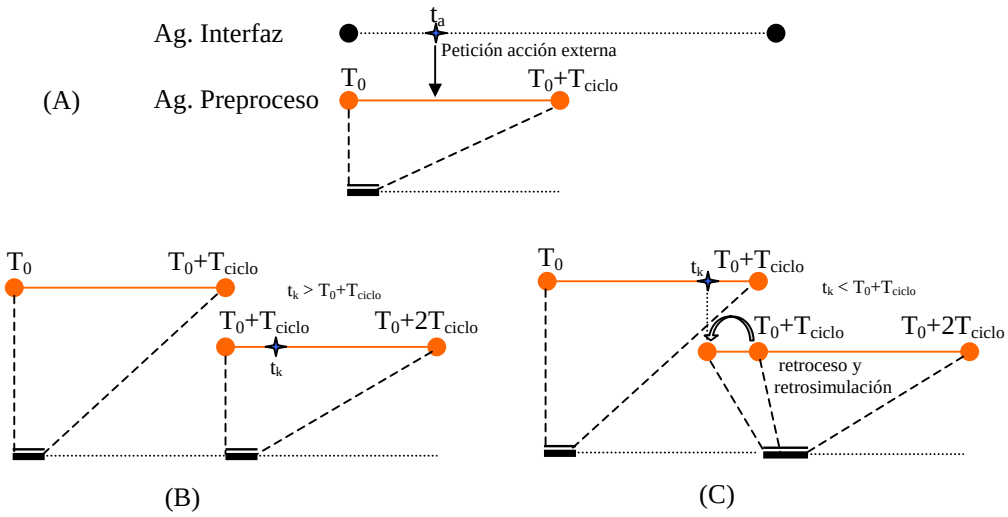


Fig 3-2 *Evolución del agente de preproceso ante la petición de una acción de control externa. Las líneas planas delgadas (limitadas con círculos) representan al intervalo temporal de simulación a calcular, mientras que las lineas gruesas inferiores representan el tiempo real empleado para el cálculo de cada simulación.*

Por ejemplo, si la longitud del intervalo temporal se define como 100 segundos, entonces el agente de preproceso simula 100 segundos (tarea que como se mostró en el capítulo anterior se realiza en un tiempo mucho menor, aproximadamente 6 segundos para los datos de las gráficas) y cede el derecho de ejecución al resto de agentes. Si posteriormente el usuario ejecuta una acción de control externa, entonces en el comienzo del siguiente ciclo de ejecución el agente de preproceso se entera de esta nueva situación y deduce el instante en el que ese cambio ha tenido lugar en el tráfico

real. Si todavía no se ha alcanzado ese instante, entonces empieza a simular su siguiente ciclo y ejecuta esta acción de control. Si por el contrario, esta acción ya fue ejecutada en el tráfico real, entonces el agente de preproceso retrocede hasta el instante en el que se efectuó y simula el resto de tiempo que le queda hasta alcanzar su tiempo final de simulación.

En la figura 3-2 se observa que si el usuario realiza una acción de control externa en el instante t_a entonces en el comienzo del siguiente intervalo de simulación el agente de preproceso determina t_k (se observa que $t_k \geq t_a$), es decir, el instante en el que esa acción va a empezar a ejecutarse en el tráfico real. Si $t_k > T_0 + T_{ciclo}$, figura 3-2 (B), entonces esa acción todavía no se ha ejecutado en el tráfico real, por lo tanto, cuando el simulador llegue a calcular el estado del tráfico para el instante t_k ejecutará esa acción. El caso (C) representa la situación en la que $t_k < T_0 + T_{ciclo}$, situación en la que el agente de preproceso debe de retroceder y repasar toda la simulación realizada desde el instante t_k hasta T_0+T_{ciclo}, para posteriormente ejecutar el simulador hasta el instante T_0+2T_{ciclo}.

Para definir cual va a ser el comportamiento del agente de control y, por lo tanto, el ciclo de funcionamiento básico del sistema se utilizan dos variables, t_{actual} y $t_{simulación}$, y dos parámetros, t_{ciclo} y $t_{integración}$.

La variable t_{actual} representa al valor del reloj interno del sistema y sirve como valor de referencia para el resto de medidas temporales a realizar. La otra variable temporal, $t_{simulación}$, representa el tiempo calculado por el simulador, que como ya se ha indicado avanza mucho más rápidamente que t_{actual}. Estas dos variables se sincronizan periódicamente cada t_{ciclo} segundos.

La selección del valor de t_{ciclo} se realiza en función de las características de la red urbana a controlar. Por ejemplo, si todos los cruces a controlar están agrupados en un único CCI, entonces una buena asignación para t_{ciclo} es hacerlo coincidir con el valor del ciclo activo en cada momento de ese CCI. Si, por el contrario, hubiese varios CCIs en el sistema, y cada uno tuviese un valor de ciclo diferente entonces se podrían seleccionar con valor para t_{ciclo} el ciclo del CCI que tuviese asociado un mayor número de cruces, o por ejemplo, el valor del mínimo tiempo de ciclo entre todos los CCIs. Una condición necesaria que debe de cumplir la selección que se realice para el valor de t_{ciclo} es que debe de ser un múltiplo del valor de $t_{integración}$. El parámetro $t_{integración}$ almacena la longitud del intervalo en el que el sistema tiene que realizar la simulación debido a la falta de datos procedentes de los sensores del sistema.

El agente de control clasifica las peticiones de ejecución del resto de agentes del sistema en una estructura jerárquica de dos niveles.

El nivel superior establece cuatro clases de agentes ordenadas de mayor a menor prioridad de ejecución. La primera clase está formada por aquellos agentes cuya ejecución periódica resulta fundamental para la aplicación, como es el caso del agente de preproceso que, además de proporcionar datos para la ejecución del resto de agentes, se encarga de sincronizar el modelo del mundo real con el almacenado en el sistema cada $t_{integración}$ segundos. Los agentes de detección, diagnosis y predicción pertenecen a

la segunda clase, mientras que el agente de solución y de refinamiento de soluciones pertenecen respectivamente a la tercera y cuarta clase. La definición de estas clases indica las dependencias de datos iniciales para cada uno de los agentes de la aplicación. Los agentes de detección, predicción y diagnosis precisan de los datos del agente de preproceso para poder ejecutarse. El agente de solución precisa de la identificación de los problemas detectados por el agente de detección y el de diagnosis para su ejecución, y el agente de refinamiento de soluciones precisa de la solución calculada para poder refinarla.

Debido a las frecuentes interacciones que realiza el agente de interfaz ya que, además de mostrar el estado del sistema cada vez que se genera un evento, recoge peticiones de información y de control por parte del usuario, este agente se excluye de esta clasificación y se trata como un agente especial al que se le dedica tiempo compartido con el resto de agentes. Más concretamente, este agente se implementa como un proceso aparte del resto de agentes del sistema y es el procesador el que distribuye el tiempo disponible de ejecución entre estos dos procesos.

El segundo nivel de la jerarquía se define para realizar una ordenación de las peticiones de ejecución para cada una de las clases definidas de agentes. Esta ordenación se realiza aplicando un conjunto de heurísticas almacenadas en la pizarra del sistema.

Todas las peticiones de ejecución se almacenan en una *agenda* que contiene tantas secciones como clases de agentes se hayan definido en el sistema. En cada sección se almacenan y se ordenan las peticiones de ejecución realizadas por todos los agentes de una misma clase.

El agente de control realiza el siguiente esquema de ejecución, que también determina el modelo de ejecución de todo el sistema (fig 3-3) [García, 98].

La primera acción que realiza el agente de control consiste en sincronizar los valores de todas las variables temporales del sistema. La siguiente acción es dedicar $t_{integración}$ segundos para ejecutar todas las peticiones del resto de agentes. Este periodo de tiempo se ejecuta en bloques de t_{ciclo} segundos. Por lo tanto el círculo interior se ejecuta un total de $(t_{integración}\ DIV\ t_{ciclo})$ veces. Este bloque de tiempo se utiliza para ejecutar el mayor número posible de peticiones de ejecución de los agentes de la agenda según sea su clasificación jerárquica y el valor calculado de su ordenación dentro de cada clase. Una vez consumido el tiempo disponible para la iteración, el sistema interrumpe la ejecución de la tarea actual, actualiza la agenda y comienza la ejecución del siguiente ciclo. Cuando se ha acabado de ejecutar el último ciclo, entonces el agente de control borra la agenda de peticiones de ejecución, lee los datos de los sensores y comienza la ejecución de otro nuevo ciclo de $t_{integración}$ segundos.

Como resultado de la ejecución del agente de preproceso se modifica el estado del tráfico representado en el sistema, bien sea por la ejecución del proceso de simulación, o bien por su actualización con los datos de los sensores. Cada una de las situaciones singulares que se generan provoca la aparición de una serie de eventos en el sistema,

que el agente de control recoge y convierte en peticiones de ejecución de cada uno de los agentes que les puedan interesar.

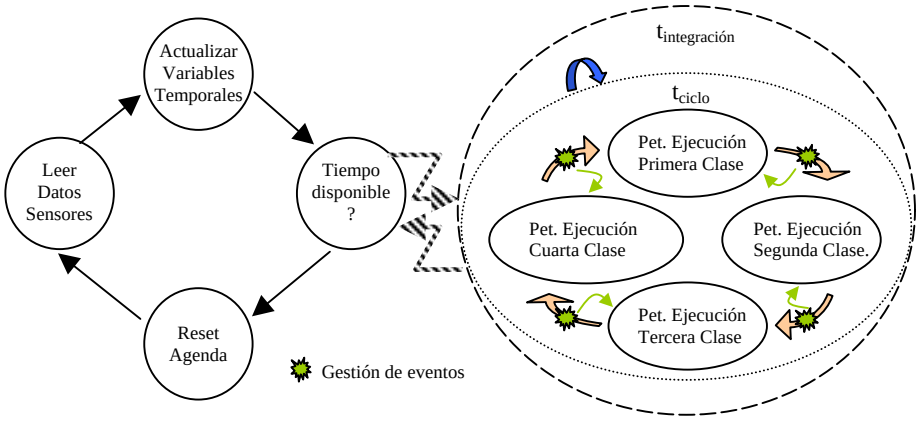


Fig 3-3 *Esquema de ejecución del agente de control. El círculo exterior de la derecha indica el tiempo máximo dedicado a cada una de las ejecuciones del círculo interior. Cada iteración del círculo interior dura t_{ciclo} segundos.*

Esta conversión de eventos en peticiones de ejecución la realiza un componente especializado del agente de control que recibe el nombre de *gestor de eventos* (figura 3-4). Este componente conoce cada uno de los tipos de eventos que el sistema puede producir, sus características y requisitos. Con este conocimiento se construye para cada evento que se produzca en el sistema una petición de ejecución asociada a un *valor de calidad*, que representa las expectativas de beneficios asociadas a su posterior ejecución.

El agente de control selecciona peticiones de ejecución de la clase más prioritaria que tenga peticiones de ejecución todavía no satisfechas. Cuando se acaban las peticiones de una clase el gestor de eventos revisa si existen nuevos eventos para esta clase. En el caso de que hubiesen surgido, entonces el constructor de tareas los convertirá en nuevas peticiones de ejecución que serán satisfechas mientras quede tiempo disponible. Si no existen nuevos eventos para esta clase, entonces el constructor de tareas recoge todos los eventos para la clase siguiente, construye las nuevas peticiones de ejecución y las inserta en la agenda.

Una vez realizados estos cálculos, se seleccionan peticiones de ejecución de la siguiente clase en prioridad (de la clase *i* se pasa a ejecutar peticiones de la clase *i+1* si hay tiempo disponible). Además, una característica de esta clasificación de agentes es que la última clase de agentes, la que agrupa al agente de solución y al de refinamiento de soluciones, es una clase muy voraz, en el sentido que cuanto más tiempo se le conceda para ejecutarse mejor será la calidad del resultado devuelto. Por este motivo, el tiempo dedicado por el sistema a ejecutar cada ciclo básico, es decir, los t_{ciclo} segundos se consumen totalmente en ejecutar tareas en el sistema. Por lo tanto, el sistema nunca permanece ocioso.

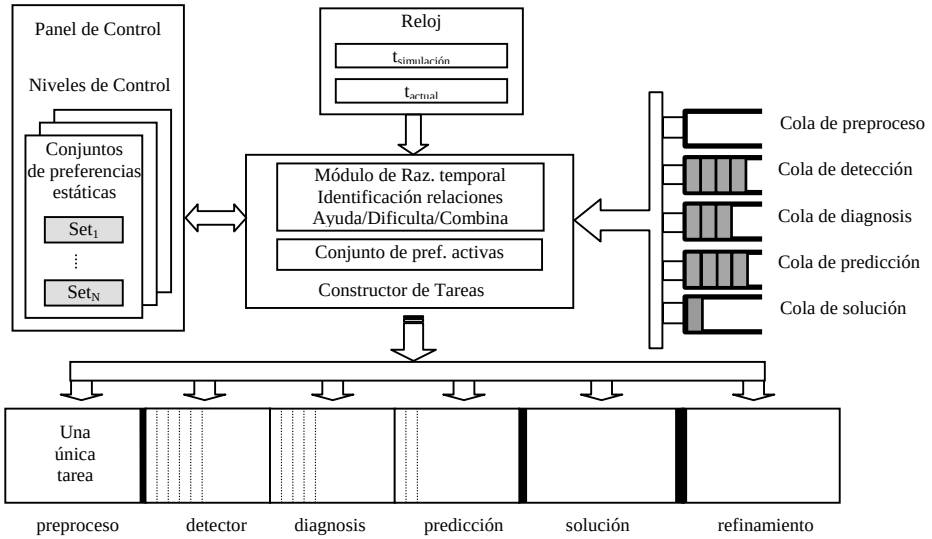


Fig 3-4 *Arquitectura de la Agenda (implementada como una cola con prioridades) y del Gestor de Eventos.*

El sistema utiliza dos tipos de eventos: internos a cada agente, y eventos de control. Los eventos internos a cada agente son lo surgidos como consecuencia de la evolución temporal cualitativa de cada objeto del sistema, y es el propio agente que los ha generado el que se encarga de su gestión (son los asociados al proceso de simulación cualitativa descrita en el capítulo 5). Los eventos de control son los surgidos como consecuencia de situaciones singulares que precisan de la próxima ejecución de algún agente. Estos eventos son los que posteriormente se pueden convertir en peticiones de ejecución para agentes del sistema.

Los eventos de control tienen todos el mismo formato básico: ***evento(Categoría, Intervalo, Tipo, Datos)*** donde **Categoría** identifica a la característica de la aplicación que lo ha generado, **Intervalo** es el intervalo temporal en el que esa característica se ha presentado o se ha mantenido, **Tipo** indica el valor concreto dentro de la categoría al que pertenece el evento, y **Datos** almacena el conjunto de datos asociados a la característica que lo motivó. Este conjunto de eventos se puede clasificar en los siguientes tipos:

- Identificación de problemas en simulación: son generados por el agente de preproceso de datos en la fase de simulación del estado del tráfico ante la ausencia de datos de sensores. Los eventos pertenecientes a este tipo tienen el valor *'simprob'* para el campo **categoría**, el campo **Tipo** puede ser *'elemental'* (en cuyo caso el campo **intervalo** es de la forma *'val(T_i, T_i)'*) o *'temporal'* (en cuyo caso el campo **intervalo** es *'val(T_i, T_f)'*). Por último, el campo **Datos** almacena los identificadores de cruces y calles involucrados en el evento.

- Identificación de problemas en predicción: son generados por el agente de predicción. El campo **categoría** tiene el valor *'futprob'*, el resto de campos tienen un contenido semejante al de los eventos de identificación de problemas en simulación.

- Identificación de problemas en reconstrucción: son generados por el agente de preproceso de datos en la fase de reconstrucción del modelo cualitativo inicial con los datos de sensores recién recibidos. Los eventos pertenecientes a este tipo tienen el valor *'diagprob'* para el campo **categoría**, el resto de campos tienen un contenido semejante al de los eventos de identificación de problemas en simulación.

- Petición de más ciclos de predicción: son generados al finalizar la ejecución de un ciclo del agente de predicción cuando todavía no ha alcanzado el valor de predicción máximo establecido en el sistema. El campo **categoría** tiene el valor *'másciclos'*, el campo *Tipo* puede contener el valor *'completo'* (indicando que la anterior ejecución del agente de predicción acabó satisfactoriamente) o *'parcial'* (indicando que la ejecución anterior del agente de predicción fue abortada por algún motivo, por ejemplo, que se agotó el tiempo t_{ciclo}), el campo **intervalo** contiene el próximo intervalo a calcular, y el campo **datos** contiene para cada cruce y calle del sistema el último instante para el que se calcularon datos de predicción.

El constructor de tareas recoge los eventos pendientes de tratar y construye cada una de las peticiones de ejecución. Estas peticiones una vez construidas se evalúan utilizando dos herramientas: el conjunto de preferencias de actuación estáticas activo en cada momento y la búsqueda de relaciones de simplificación entre todas las peticiones. Estas herramientas permiten la ejecución "oportunista" típica de las arquitecturas de pizarra. Aunque este concepto se emplea en un sentido ligeramente distinto de su significado habitual. En su sentido tradicional [Nii, 86], [Shapiro, 92], [Craig, 89], [Velthuijsen, 92], el concepto de comportamiento "oportunista" en arquitecturas de pizarra implica la selección como fuente de conocimiento a ejecutar en cada momento de la que mejor colabore en el proceso de resolución del problema. Este modo de funcionamiento implica la resolución compuesta de dos problemas distintos: el propio de la aplicación, conocido con el nombre del *problema del dominio*, y el adicional de búsqueda de aquellas secuencias de ejecuciones más beneficiosas para la resolución del sistema, lo que se conoce con el nombre del *problema del control* [Hayes-Roth, 84], [Carver, 94]].

En el caso de la arquitectura que se propone, y al tratarse de una aplicación que requiere de un proceso continuo de monitorización sujeto a unas ciertas restricciones de tiempo real suaves, el carácter oportunista se interpreta como aquella ejecución que deja más tiempo de cómputo libre para las ejecuciones más pesadas, que fundamentalmente en esta aplicación es la asociada al agente de solución.

Cada conjunto de preferencias estáticas se define al inicio de la ejecución del sistema y se almacena en la pizarra. Cada definición consiste de un conjunto de reglas que al ser

aplicadas sobre cada nueva petición de ejecución, genera para cada una de ellas un valor de calidad. Por ejemplo, el siguiente conjunto de reglas implementa la asignación de un valor de calidad a cada petición pendiente de ejecución según la zona de la red urbana a la que afecte:

```
asignación_pref(Zona, Peticiones, PeticionesPeso) :-
        cruces_zona(Zona, Cruces),
        links_zona(Zona, Links),
        zona(Zona, Indice),
        zona_peso_pet(Peticiones, PeticionesPeso, Cruces, Links).

zona_peso_pet([], [],_,_,_) :- actualiza_índice(Zona).
zona_peso_pet([Petición | RP], [Petición-Peso | RPP], Indice, Cruces, Links) :-
        Petición =.. [Agente, Tipo, Intervalo, Categoría, CrucesP, LinksP],
        ( ( Agente == detector ; Agente == diagnosis ) ->
            ( comunes(Cruces, CrucesP, Links, LinksP, Ncruces, Nlinks),
              Peso is Ncruces * Indice + Nlinks
            ) ; Peso is 0
        ),
        zona_peso_pet(RP, RPP, Indice, Cruces, Links).
```

Los conjuntos de preferencias definidos para esta aplicación son los siguientes:

- Preferencias de zonas: se divide la red urbana en zonas, y a cada una de ellas se le asigna un valor de importancia. Este valor se modifica dinámicamente de acuerdo a las veces que se accede a cada zona para determinar un problema. Un ejemplo de tal conjunto se corresponde con el ejemplo de reglas anterior.

- Preferencias secuencial agentes: las peticiones del agente de detección tiene menor prioridad que las del agente de diagnosis, pero mayor que las del agente de predicción.

- Preferencias anulación predicción: en la situación de que el tiempo disponible para ejecutar tareas no sea suficiente para todas ellas, se escoge eliminar aquellas tareas asociadas al agente de predicción.

Estos conjunto de preferencias se almacenan en la pizarra y la decisión sobre cuál aplicar en cada momento se puede realizar tanto estáticamente como dinámicamente. Hay que tener en cuenta que si esta selección se realiza dinámicamente, este proceso consume un tiempo de ejecución que puede ser necesario para la posterior ejecución del resultado de la selección. Por lo tanto, habría que buscar una solución de equilibrio entre tiempo dedicado al proceso de selección y tiempo dedicado a la ejecución de las tareas de la agenda.

Para buscar esta situación de equilibrio se escoge la asignación estática del conjunto de preferencias y se la complementa con la introducción de otra herramienta con la que

poder afinar este proceso de evaluación: la identificación de relaciones *ayuda/dificulta* (facilities/hinders) [Long, 95] y la relación *combina* entre las diferentes peticiones de ejecución activas.

La relación *ayuda* entre dos peticiones de ejecución *A* y *B* indica que la ejecución de la petición *A* permitirá que la posterior ejecución de la petición *B* requiera menos tiempo, mejore el valor de su solución, o que se elimine alguna otra petición de la agenda.

La relación *dificulta* entre dos peticiones de ejecución *A* y *B* es la forma contraria de la relación *ayuda* en el sentido que la ejecución de la petición *A* va a producir que la posterior ejecución de la petición *B* requiera más tiempo, empeore el valor de su solución, o que introduzca nuevas y redundantes peticiones de ejecución en el sistema, en el sentido que estas peticiones respondan a situaciones que ya se encuentren representadas en el actual conjunto de peticiones de ejecución.

La relación *combina* entre dos peticiones de ejecución *A* y *B* de un mismo agente consiste en su eliminación de la agenda y en la introducción de una nueva petición *C* que integre las características de estas peticiones eliminadas. Esta relación se puede realizar en el caso de que las peticiones *A* y *B* requieran de muy poco tiempo de ejecución, y por lo tanto, sea más interesantes ejecutarlas a la vez que lanzar ejecuciones independientes.

Un ejemplo de dos peticiones relacionadas mediante la relación *dificulta* sucede cuando la petición *A* está asociada a la identificación de un problema en el segmento l_j, y la petición *B* identifica a un problema en un cruce $ifis_k$ que contiene al segmento l_j como calle de entrada. En esta situación la ejecución de la petición *A* podría resultar en cálculo inútil, puesto que la posterior ejecución de la petición *B* podría invalidar los resultados de *A* (al considerar no solamente el segmento l_j sino el cruce $ifis_k$ lo que permitiría explicar mejor la causa del problema). En este caso, por lo tanto, se cumple que *A dificulta B* [García, 96].

Otra situación parecida sucede cuando la petición *A* implica el cálculo de la evolución del sistema para un intervalo temporal $[t_i, t_f]$, mientras que la petición *B* consiste en la ejecución del agente de preproceso sobre ese mismo intervalo temporal $[t_i, t_j]$. Como la petición *B* no puede ser satisfecha hasta el inicio del próximo ciclo de ejecución, la ejecución de la petición *A* implicará que cuando se ejecute *B* su duración sea sustancialmente reducida, puesto que le basta con acceder a los datos que haya calculado el agente de predicción mediante la petición *A*. En esta situación se cumple que *A ayuda B*.

La situación en la que las peticiones *A* y *B* estén asociadas, respectivamente, a la identificación de problemas de tráfico en los instantes t_i y t_j, y se cumpla que ambos instantes pertenecen al mismo intervalo temporal de simulación o de predicción de futuro, es una situación candidata para unirlas mediante la relación *combina*.

Mediante la utilización de estas dos herramientas en la asignación de valores de calidad para cada una de las peticiones pendientes de ejecución se consigue la integración de dos factores distintos: *factores dirigidos por los datos* y *factores dirigidos por los* objetivos. Los *factores dirigidos por los datos* informan acerca de lo mejor que se puede hacer con los datos disponibles (los eventos surgen como consecuencia de la aparición de datos singulares en la pizarra). Los *factores dirigidos por los objetivos* informan sobre qué es lo más aconsejable hacer para resolver el problema (la selección de peticiones de ejecución que optimice el tiempo de ejecución restante). Sin un control dirigido por los objetivos se puede emplear mucho esfuerzo en trabajar sobre datos e hipótesis poco importantes para el sistema. Sin un control dirigido por los datos se puede emplear mucho esfuerzo persiguiendo objetivos que no pueden ser fácilmente alcanzados [Carver, 94].

3.3 Integración y Cooperación de los agentes en el sistema

Todos los agentes de la aplicación utilizan una base de conocimientos estructurada común, la pizarra, que utilizan tanto para dejar resultados (intermedios o finales) o hipótesis de trabajo, como un medio mediante el cual poder intercambiar información. Esta estructuración se realiza fundamentalmente por dos razones: 1) mejorar el acceso a cada uno de sus componentes, y 2) mejorar la eficiencia del sistema, en el sentido que todos los agentes del sistema tienen acceso en todo momento a aquellos datos que precisan para su ejecución. En las siguientes secciones se expone el tipo de información que se almacena en la pizarra, así como la estructura de datos utilizada en cada implementación.

3.3.1 Almacén de Información Cognitiva. El TKB

Como ya se ha indicado, en la base de datos temporal común (en adelante TKB) se almacenan tanto las entradas generadas durante la resolución del problema, como cualquier otro tipo de conocimiento que pueda ser considerado útil en ese proceso. El TKB se organiza en dos bloques (denominados *paneles* en la nomenclatura habitual en sistemas de pizarra). Un panel está dedicado a tratar cuestiones del control de la aplicación, mientras que el segundo panel almacena el estado de resolución del sistema en cada momento.

El panel de control se encarga de almacenar las heurísticas de control hábiles en el sistema. Como ya se ha indicado previamente, sólo una de estas heurísticas estará activa en cada momento, y la selección de cuál activar depende directamente del usuario. Aunque este es el modo de funcionamiento habitual, también se podría emplear un modo de funcionamiento dinámico, en el que el propio sistema decide que heurística de control aplicar en cada momento en función del número de tareas pendientes de ejecución, del tiempo estimado para realizar cada una de ellas y del tiempo que resta hasta el comienzo del próximo ciclo de integración de datos de los sensores. No obstante, como ya se ha apuntado, este segundo modo de funcionamiento plantea la discusión entre el tiempo que dedicar a tareas de control respecto del tiempo que hay que dedicar a resolver el problema de la aplicación. La solución adoptada en este trabajo

consiste en desarrollar en una primera fase el modo estático de funcionamiento, y con relación a los resultados obtenidos al implementar y experimentar con este modelo decidir si es razonable implementar el modo de selección dinámica de la heurística de control a aplicar.

Para ayudar en el proceso de selección del orden en que ejecutar las tareas pendientes se utiliza una segunda herramienta: la búsqueda de las relaciones *ayuda*/*dificulta*. Esta búsqueda se realiza sobre el panel de control mediante la construcción de un grafo de dependencias temporales/topológicas entre las tareas y los datos requeridos por cada una de ellas. Por este motivo, en la construcción de este grafo se tiene acceso a los datos del panel de la aplicación, es decir, se establecen *enlaces* entre ambos paneles.

El panel de la aplicación almacena todas las características de los elementos significativos de la red urbana bajo control. Estos elementos se clasifican de acuerdo a la descomposición orientada a objetos establecida por el formalismo *(λ, t)* (figura 3-5). Este formalismo establece una superclase denominada *universo* que agrupa al resto de clases. De esta superclase cuelgan otras tres clases: *parámetros*, *valores dominio*, *dominio*.

La clase *parámetros* establece los diferentes tipos de evolución que se soportan en el sistema: constante (clase *f*), función unidimensional del tiempo (clase *ft*) o función bidimensional del tiempo y de otra componente que en esta aplicación es el *espacio* (clase *flt*).

La clase *valores dominio* establece los distintos conjuntos de valores que pueden utilizar cada objeto del sistema. Estos valores pueden ser valores *cualitativos* que pertenecen a la clase *fluido* o a la clase *congestión*, o valores *cualitativos* asociados a sus correspondientes *intervalos cuantitativos*, como son las clases *flujo fluido*, *flujo congestión* y *flujo nulo*.

Los objetos de la aplicación del sistema pertenecen a la clase *dominio*. Esta clase es subdividida en dos clases más. La clase *externa* se corresponde con aquellos elementos que representan condiciones frontera en el sistema, como son los objetos agrupados en las clases *control* y *entradas*. La clase *interna* se corresponde con todos los elementos que no son parte de la clase externa, como son los objetos agrupados en las clases *cruces* y *segmentos*.

Cada objeto final de la clase *dominio* está compuesto de una parte estática y de una parte dinámica. La componente estática almacena sus características físicas y topológicas, mientras que la componente dinámica almacena su evolución temporal cualitativa. Se observa que en esta clasificación no se han tenido en cuenta los objetos *salida* en la red urbana puesto que se les identifica con objetos sumideros que consumen ininterrumpidamente todo el flujo que son capaces de absorber.

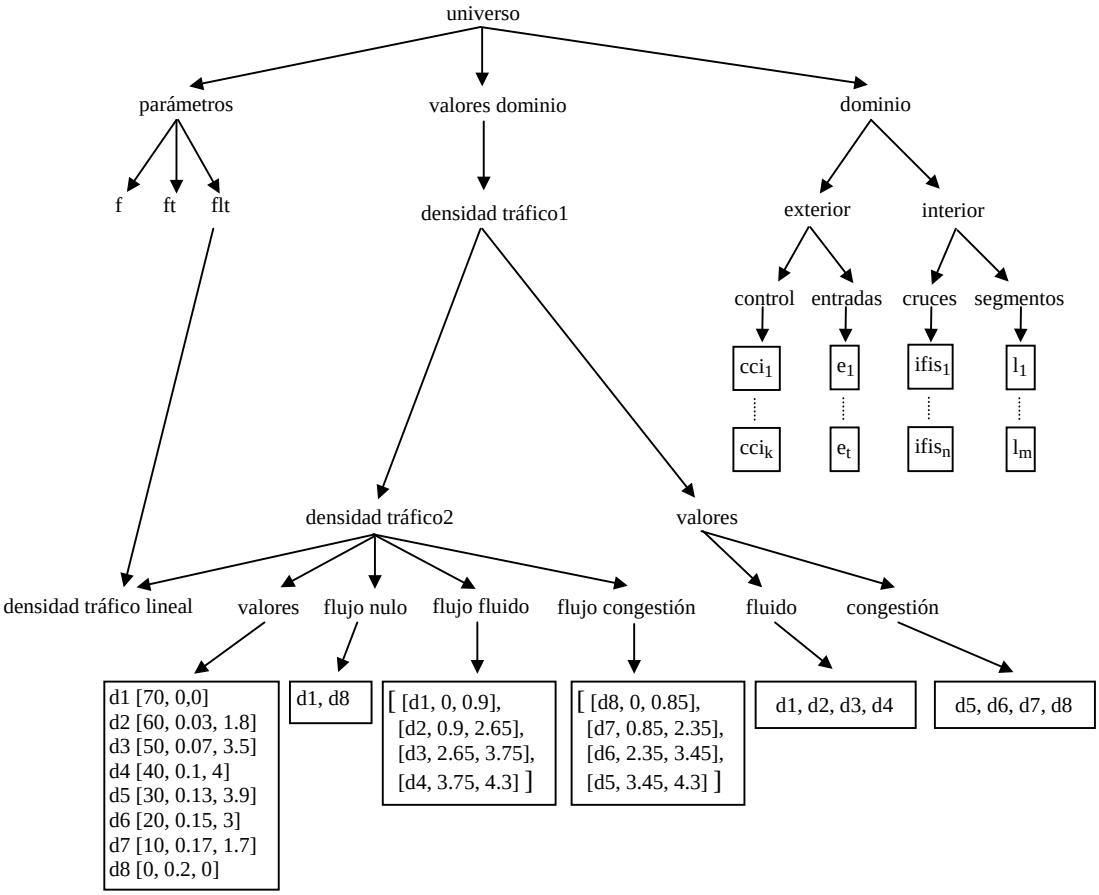
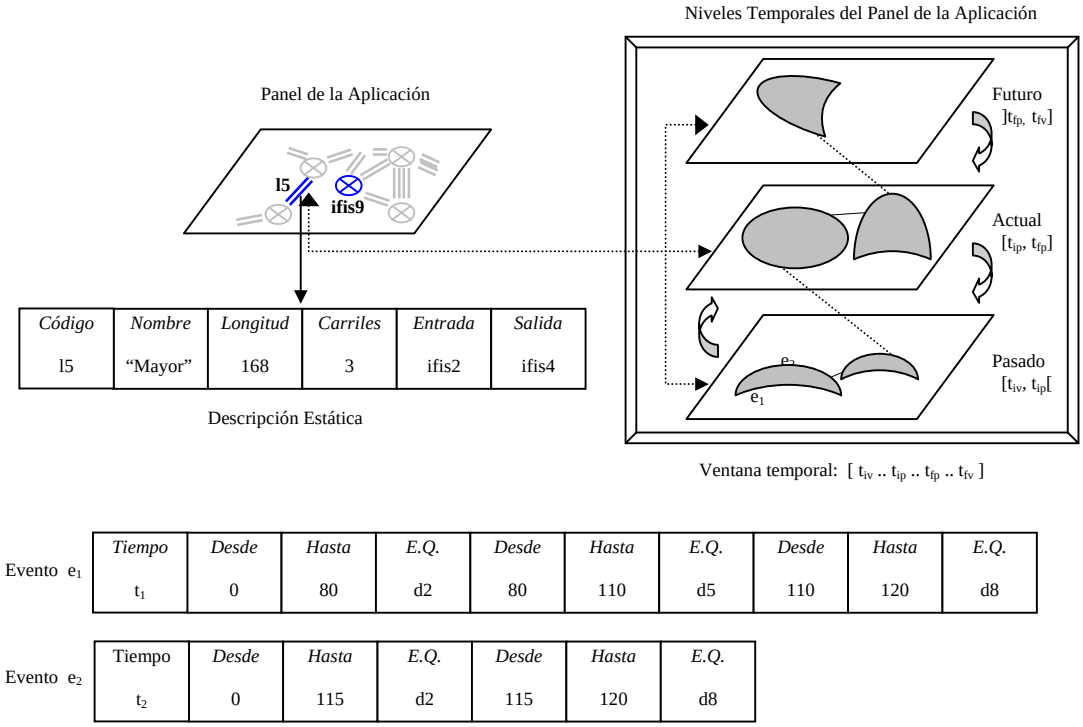


Fig 3-5 *Descomposición orientada a objetos de todos los elementos del panel de la aplicación en el TKB*

Los valores correspondientes a la descripción dinámica de un objeto de la clase *dominio* se agrupan en intervalos temporales de longitud t_{ciclo} debido fundamentalmente a dos razones. La primera es que cuando se precisa analizar algún comportamiento cualitativo de un objeto del sistema, se necesita un intervalo de esos valores para poder realizar ese análisis. Por este motivo la tarea será más sencilla de realizar si se dispone de todo este conjunto de datos en un solo acceso. La segunda razón que motiva este agrupamiento es por razones de eficiencia en el acceso. Si se desea acceder a un determinado valor cualitativo de un objeto, resulta más rápido insertar todos los datos de un intervalo en una lista en solo paso y realizar una búsqueda posterior sobre esa lista que realizar directamente esta búsqueda sobre toda la pizarra.

Para tratar con todos estos conjuntos de datos el panel de la aplicación consta de tres niveles adicionales en los que se distribuyen estos conjuntos [García, 97]. Todos los conjuntos con valores temporales finales inferiores a un determinado valor t_{ip} forman parte ordenada del nivel *pasado*. Aquellos valores cuyos intervalos estén entre dos marcas temporales t_{ip} y t_{fp} pertenecen al nivel *actual*. El tercer nivel, el nivel *futuro*, almacena los valores cuya marca temporal es superior a t_{fp}.



	Código	*Nombre*	*Longitud*	*Carriles*	*Entrada*	*Salida*
	l5	"Mayor"	168	3	ifis2	ifis4

	Tiempo	*Desde*	*Hasta*	*E.Q.*	*Desde*	*Hasta*	*E.Q.*	*Desde*	*Hasta*	*E.Q.*
Evento e_1	t_1	0	80	d2	80	110	d5	110	120	d8

	Tiempo	*Desde*	*Hasta*	*E.Q.*	*Desde*	*Hasta*	*E.Q.*
Evento e_2	t_2	0	115	d2	115	120	d8

Fig. 3-6 *Organización del panel de aplicación. Las áreas sombreadas en cada nivel representa un conjunto de datos asociado a un intervalo temporal de longitud* t_{ciclo}

Los valores t_{ip} y t_{fp} son comunes a todos los objetos de la aplicación en el sistema (cruces, segmentos, cruces y entradas). Si se denomina por t_{iv} al instante más antiguo que se almacena en el nivel *pasado* y por t_{fv} al instante de predicción de futuro que el sistema es capaz de alcanzar, entonces se define que el sistema tiene abierta la siguiente ventana temporal:

$$[t_{iv} .. t_{ip} .. t_{fp} .. t_{fv}]$$

Aquellos conjuntos de valores cuyo intervalo temporal asociado tenga una marca temporal superior que sea inferior a t_{iv} se guardan en memoria externa. Estos datos podrían ser útiles para su posterior análisis como datos de entrada para otros programas (no desarrollados en esta tesis) que permitieran calcular nuevos casos de planes de señales más ajustados a cada hora y día.

```
identificador:
    ifis9
fases:
    fase(1, [[l35, l31, 21], [l35, l36, 62], [l35, l34, 17]])
    fase(2, [[l32, l36, 60], [l32, l34, 40], [l33, l31, 100]])
    fase(3, [[l33, l36, 100], [l35, l31, 100]])
```

```
        fase(4, [[]])
    repartos:
        reparto(1, [500, 200, 200, 100])
        reparto(2, [450, 250, 200, 100])
        reparto(3, [400, 250, 250, 100])
        reparto(4, [350, 300, 250, 100])
        reparto(5, [300, 300, 300, 100])
        reparto(6, [250, 350, 300, 100])
        reparto(7, [200, 350, 350, 100])
        reparto(8, [150, 400, 350, 100])
    entradas:
        [l32, l33, l35]
    salidas:
        [l31, l34, l36]
    cci:
        cci1
```

Tabla 3.1 *Descripción estática del cruce ifis9. Todos los objetos de la clase cruce tienen 8 vectores de reparto ordenados de mayor a menor tiempo de verde en la dirección principal. Para este objeto hay definidas 4 fases. En cada fase se representa el sentido de tráfico permitido y el porcentaje de vehículos que lo utilizan. También se representan los segmentos de entrada y de salida del cruce, así como el CCI al que pertenece.*

El contenido de estos tres niveles no es permanente, sino que en función de la ejecución del sistema, conjuntos de datos pueden cambiar de nivel. La situación habitual es que los datos pasen del nivel *actual* al nivel *pasado*, y del nivel *futuro* al nivel *actual*. Pero dependiendo de lo ajustado que sea la simulación de tráfico calculada, de los incidentes que sucedan en su evolución y del momento en que se detecten, también puede darse el caso de que datos del nivel *pasado* pasen al nivel *actual* para volver a ser analizados, y una vez finalizado este análisis vuelvan a incorporarse al nivel *pasado*.

La descripción estática de un objeto de la clase *cruce* incluye información tal como su identificador único para el sistema de control, sus fases, repartos, cci al que pertenece y los identificadores de sus calles de entrada y de sus calles de salida. La tabla 3.1 muestra la descripción estática del objeto *ifis9* que es un objeto de la clase *cruce*.

La descripción dinámica de un objeto de tipo cruce está determinada por los valores cualitativos representativos de su evolución y de los instantes en que éstos suceden.

En la tabla 3.2 se observa que en cada conjunto de datos se recogen todos los eventos que han resultado significativos en la evolución del objeto *ifis9*. Cada elemento consta de tres datos.

El primero es la identificación del tiempo asociado al evento. El segundo es una lista que contiene la nueva descripción cualitativa del cruce *ifis9* en este instante que consta

de la fase actual en ejecución y de los estados cualitativos finales de sus segmentos de entrada y de los estados cualitativos iniciales de sus segmentos de salida. Esta lista está formada por combinaciones de algunas de estas cinco posibilidades:

1) [phase, NuevaFase, control]: el cruce cambia a la fase siguiente,

2) [outstart, NuevoEQ, IdSegmento]: el segmento IdSegmento tiene NuevoEQ como nuevo valor cualitativo en su tramo inicial,

3) [outend, NuevoEQ, IdSegmento]: el segmento IdSegmento tiene NuevoEQ como nuevo valor cualitativo en un tramo final,

4) [instart, NuevoEQ, IdSegmento]: el segmento IdSegmento, que es una de las calles de salida del cruce, tiene NuevoEQ como nuevo valor cualitativo en su tramo inicial, y

5) [inend, NuevoEQ, IdSegmento]: el segmento IdSegmento, que es una de las calles de entrada al cruce, tiene NuevoEQ como nuevo valor cualitativo en su tramo final.

Además de estos tres niveles, el panel de la aplicación también almacena los valores actuales de los parámetros ajustables para cada objeto de la clase cruce, como son su valor actual de coordinación respecto al inicio del ciclo asociado al cci al que pertenece, y el identificador del vector de reparto que está ejecutando en cada momento. La evolución temporal de cada uno de estos parámetros se almacena en este panel formando una lista para cada objeto.

La descripción estática de un objeto de la clase *segmento* incluye información relativa a su identificador dentro del sistema, su nombre real, longitud, número de carriles y los identificadores de los objetos conectados a su tramo inicial y a su tramo final. La figura 3-6 muestra esta descripción estática para el objeto *l5*.

La tabla 3.3 muestra la evolución temporal cualitativa del objeto *l5*. Estos datos, como sucede para todos los datos de la clase *dominio*, se agrupan en bloques de t_{ciclo} segundos, y cada elemento de este bloque tiene la siguiente información. En un primer lugar se almacena la etiqueta temporal asociada a ese evento.

La segunda parte está formada por una lista de dos componentes. En una de ellas se almacena el estado cualitativo actual del segmento. En la otra componente se almacena información sobre que tipo de cambio significativo se ha realizado en el segmento. Esta información se establece mediante la presencia de algunas de estas marcas: 1) [instart, NuevoEQ, IdSegmento]: el segmento IdSegmento, cuyo extremo inicial está conectado a un cruce, tiene NuevoEQ como nuevo valor cualitativo en su tramo inicial, y 2) [inend, NuevoEQ, IdSegmento]: el segmento IdSegmento, cuyo extremo final está conectado a un cruce, tiene NuevoEQ como nuevo valor cualitativo en su tramo final.

```
[[[ 36738,
  [ [state,2,[[l32,d4],[l33,d1],[l35,d1]],[[l31,d1],[l34,d2],[l36,d2]]],
    [instart,d2,l34],[instart,d2,l36],[inend,d4,l32],[phase,2,control1]] ],
[ 36746,
  [ [state,2,[[l32,d1],[l33,d1],[l35,d1]],[[l31,d1],[l34,d1],[l36,d1]]],
    [instart,d1,l34],[instart,d1,l36],[inend,d1,l32],[outend,d1,l32]] ],
[ 36768,
  [ [state,2,[[l32,d2],[l33,d1],[l35,d1]],[[l31,d1],[l34,d1],[l36,d2]]],
    [instart,d2,l36],[inend,d2,l32],[outend,d2,l32]] ],
[ 36772,
  [ [state,2,[[l32,d1],[l33,d1],[l35,d1]],[[l31,d1],[l34,d1],[l36,d1]]],
    [instart,d1,l36],[inend,d1,l32],[outend,d1,l32]] ],
[ 36773,
  [ [state,3,[[l32,d1],[l33,d1],[l35,d1]],[[l31,d1],[l34,d1],[l36,d1]]],
    [phase,3,control1]] ],
[ 36781,
  [ [state,3,[[l32,d8],[l33,d1],[l35,d2]],[[l31,d2],[l34,d1],[l36,d1]]],
    [instart,d2,l31],[inend,d8,l32],[inend,d2,l35],[outend,d2,l32],[outend,d2,l35]] ],
[ 36781,
  [ [state,3,[[l32,d8],[l33,d1],[l35,d2]],[[l31,d2],[l34,d1],[l36,d1]]] ],
[ 36787,
  [ [state,4,[[l32,d8],[l33,d1],[l35,d8]],[[l31,d1],[l34,d1],[l36,d1]]],
    [instart,d1,l31],[inend,d8,l35],[phase,4,control1]] ],
[ 36797,
  [ [state,1,[[l32,d8],[l33,d1],[l35,d4]],[[l31,d1],[l34,d1],[l36,d2]]],
    [instart,d2,l36],[inend,d4,l35],[phase,1,control1]] ],
[ 36806,
  [ [state,1,[[l32,d8],[l33,d1],[l35,d1]],[[l31,d1],[l34,d1],[l36,d1]]],
    [instart,d1,l36],[inend,d1,l35],[outend,d1,l35]] ],
[ 36825,
  [ [state,1,[[l32,d8],[l33,d1],[l35,d2]],[[l31,d1],[l34,d1],[l36,d2]]],
    [instart,d2,l36],[inend,d2,l35],[outend,d2,l35]] ]
],
[[ 36841,
  [ [state,2,[[l32,d4],[l33,d1],[l35,d8]],[[l31,d1],[l34,d2],[l36,d2]]],
    [instart,d2,l34],[inend,d4,l32],[inend,d8,l35],[phase,2,control1]] ],
[ 36849,
  [ [state,2,[[l32,d1],[l33,d1],[l35,d8]],[[l31,d1],[l34,d1],[l36,d1]]],
    [instart,d1,l34],[instart,d1,l36],[inend,d1,l32],[outend,d1,l32]] ],
[ 36876,
  [ [state,3,[[l32,d1],[l33,d1],[l35,d4]],[[l31,d4],[l34,d1],[l36,d1]]],
    [instart,d4,l31],[inend,d4,l35],[phase,3,control1]] ],
                                        ...
],
                                        ...
].
```

Tabla 3.2 *Contenido del nivel actual para el cruce ifis9. En esta lista sólo se almacenan los valores significativos de la evolución de ese cruce: cambios de fase y cambios cualitativos en sus entradas o salidas.*

En el caso de los objetos de las clases *entrada* y *control*, el panel de aplicación almacena los datos de su evolución temporal siguiendo el mismo criterio que para los objetos de las clases *cruces* y *segmentos*. No obstante, estos objetos sólo modifican su

comportamiento mediante acciones externas, por lo que requieren de muy poco espacio para poder representar su evolución.

La descripción estática de cada objeto de la clase *control* consta de un identificador propio, los posibles valores para la longitud del ciclo y los cruces asociados a este objeto. Su descripción dinámica se construye formando una lista de parejas *(t_i, Nvciclo)* que indica que en el instante t_i este cci ha cambiado al nuevo valor de ciclo *Nvciclo*.

```
[[ [36738,[[instart,d2],[state,[d2,0,0],[d1,0,120]]]],
   [36745,[[inend,d8],[state,[d2,0,120],[d8,120,120]],[outend,d2]]],
   [36746,[[instart,d1],[state,[d1,0,0],[d2,0,117],[d8,117,120]]]],
   [36752,[[state,[d1,0,99],[d8,99,120]]]],
   [36768,[[instart,d2],[state,[d2,0,0],[d1,0,99],[d8,99,120]]]],
   [36772,[[instart,d1],[state,[d1,0,0],[d2,0,67],[d1,67,99],[d8,99,120]]]],
   [36774,[[state,[d1,0,32],[d2,32,99],[d8,99,120]]]],
   [36777,[[state,[d1,0,89],[d8,89,120]]]],
   [36777,[[state,[d1,0,89],[d8,89,120]]]],
   [36794,[[inend,d4],[state,[d1,0,89],[d8,89,120],[d4,120,120]]]],
   [36797,[[instart,d2],[state,[d2,0,0],[d1,0,89],[d4,89,120]]]],
   [36800,[[state,[d2,0,47],[d1,47,120]],[outend,d1]]],
   [36804,[[state,[d2,0,120]],[outend,d2]]],
   [36806,[[instart,d1],[state,[d1,0,0],[d2,0,120]]]],
   [36813,[[state,[d1,0,120]],[outend,d1]]],
   [36825,[[instart,d2],[state,[d2,0,0],[d1,0,120]]]],
   [36832,[[state,[d2,0,120]],[outend,d2]]]                          ],

[  [36839,[[inend,d8],[state,[d2,0,120],[d8,120,120]]]],
   [36849,[[instart,d1],[state,[d1,0,0],[d2,0,91],[d8,91,120]]]],
   [36854,[[state,[d1,0,77],[d8,77,120]]]],
   [36878,[[state,[d1,0,91],[d8,91,120]]]],
                                    ...
                                                                     ],

                                    ...

].
```

Tabla 3.3 *Contenido del nivel actual para el segmento l5. En esta lista sólo se almacenan los valores significativos de la evolución de este segmento, sus cambios cualitativos y sus intervalos de comienzo y fin.*

La descripción estática de cada objeto de la clase *entrada* consta de un identificador propio (qué está relacionado con el identificador del segmento al que cada entrada está conectada). La descripción dinámica se construye formando una lista de parejas *(t_i, EQentrada)* que indica que en el instante t_i (que coincide con el momento de integración de datos en el sistema) esta entrada ha modificado su valor cualitativo persistente durante todo el periodo inmediatamente anterior de $t_{integración}$ segundos al nuevo valor *EQentrada* para el próximo periodo de simulación.

3.3.2 Intercambio de Información

Una función importante de la pizarra en cualquier arquitectura de pizarra consiste en proporcionar un medio de comunicación unificado por el que intercambiar datos entre los distintos agentes del sistema. Este intercambio de información se realiza de forma

asíncrona fundamentalmente por dos motivos. El primero reside en que los agentes del sistema son independientes entre sí, y por lo tanto, no tienen porqué saber qué otro experto va a operar con los resultados que hubiesen calculado. El segundo es debido a la posible falta de procesadores para ejecutar simultáneamente todas las tareas pendientes, y por lo tanto, que un agente no se quede esperando por datos de otro agente que esté en espera de ser ejecutado.

No obstante, debido a las características especiales del sistema de control de tráfico desarrollado, esta arquitectura de pizarra se aumenta con la capacidad de emisión y de recepción de mensajes entre los agentes de la aplicación. En el momento en que un agente entre en su fase de ejecución la primera tarea que realiza es comprobar si tiene mensajes pendientes de lectura. En el caso de que así suceda, entonces el agente receptor determinará que hacer con ese mensaje, y si debe enviar un acuse de recibo o no al agente emisor.

Los mensajes tienen cinco campos. El primero identifica al agente que ha enviado el mensaje. El segundo el receptor al que el mensaje va dirigido. El tercer campo contiene una etiqueta que identifica al tipo de mensaje del que se trata. El conjunto de etiquetas posibles es un conjunto finito y conocido por todos los agentes del sistema. El cuarto campo sirve para determinar el instante temporal en que se envió el mensaje, y el último campo contiene los datos que han motivado el envío del mensaje.

La siguiente figura muestra un ejemplo de un mensaje enviado por el agente *Detector* para ser recibido por el agente *Solución* que le informa de la presencia de una situación de congestión circular entre los cruces *ifis3, ifis5, ifis6, ifis3*. Este problema ha sido detectado en el intervalo temporal $[t_j, t_i]$.

Emisor	*Receptor*	*Tipo Msg*	*Tiempo*	*Datos*
Detector	Solución	Deadlock	t_i	[ifis3, ifis5, ifis6]-[t_j, t_i]

Fig. 3.7 *Ejemplo de mensaje dirigido directamente al agente Solución desde el agente Detector.*

La comunicación, en todo caso, sigue siendo asíncrona, puesto que una de las limitaciones iniciales de esta arquitectura es que el número de procesadores disponibles en el sistema no tiene por que ser mayor o igual al número de peticiones pendientes de ejecución para cada agente. Mediante el empleo de este tipo de mensajes se implementan mecanismos de comunicación por multidifusión (broadcasting) como es el caso en el que se haya realizado alguna acción de control externa sobre el sistema. Esta nueva situación debe de ser comunicada al resto de agentes para que actualicen sus datos de la forma adecuada.

3.4 El Agente de Preproceso

El agente de preproceso tiene el objetivo de conseguir que en todo momento la representación de la evolución de tráfico en el sistema sea coherente con su evolución real. Con este fin el agente de preproceso utiliza fundamentalmente dos métodos: la conversión de datos de flujo y ocupación proporcionados por los sensores del sistema (que en el momento actual han sido considerados todos como sensores del tipo espira magnética, aunque no son los únicos tipos de sensores que se podrían utilizar [Moreno, 96]) en datos cualitativos de la densidad, y la simulación cualitativa del tráfico para mantener una visión lo más ajustada posible de la evolución real del tráfico cuando no se poseen estos datos.

El modo de funcionamiento de este agente consiste en generar el estado actual del sistema desde el estado cualitativo previo utilizando simulación cualitativa en vez de emplear un análisis directo de los datos de los sensores. El proceso de simulación cualitativa desde el estado previo proporciona una visión global del estado del sistema sujeto a tres suposiciones:

- El estado cualitativo previo es correcto, es decir, representa de forma fiel al sistema real.

- El proceso de predicción cualitativa es correcto, es decir, el modelo implementado del comportamiento de cada objeto del sistema responde a su comportamiento real.

- Ausencia de incidentes en el próximo paso de simulación.

Los datos de los sensores, cuando están disponibles, se utilizan para verificar los estados cualitativos calculados. El estado actual del sistema obtenido mediante el proceso de simulación cualitativa proporciona más información que la que se puede obtener directamente de los datos de los sensores. Esto es debido a que para calcular el estado actual del tráfico se ha utilizado información del estado previo y este estado previo ha sido calculado mediante la combinación de datos procedentes de los sensores en el pasado y del conocimiento del sistema sobre la evolución del tráfico.

El agente de preproceso realiza esta tarea de forma periódica al menos una vez cada $t_{integración}$ segundos, y por este motivo, se define como una tarea de la primera clase (según la organización expuesta en la descripción del agente de control). Esta tarea es fundamental para el sistema, puesto que existe una gran dependencia de datos entre los resultados que proporciona este agente y los datos iniciales precisos para la ejecución del resto de agentes del sistema.

No obstante, la ausencia de valores actuales de los sensores del sistema durante periodos de $t_{integración}$ segundos plantea la discusión de cómo calcular de la forma más eficiente posible para el sistema este periodo de simulación. La solución adoptada, que busca un equilibrio entre rapidez de ejecución y fidelidad respecto de la evolución real

del tráfico, consiste en ejecutar intervalos de simulación de duración t_{ciclo} hasta completar el tiempo de integración.

3.4.1 Esquema de Funcionamiento

Cuando al agente de preproceso se le concede derecho de ejecución (T_{sim}), lo primero que comprueba es si existen acciones de control externas que se hayan solicitado en el ciclo anterior (y que, por lo tanto, pueden haber sido ejecutadas en el tráfico real y no haber sido tenidas en cuenta en este sistema).

En el caso de que existan acciones de control externas el sistema debe de recogerlas, analizarlas y determinar para cada una de ellas cuando se empiezan a ejecutar. Todas estas acciones se almacenan en dos listas. La lista *Val* almacena todos las acciones que se han debido de empezar a ejecutar en el intervalo anterior. La lista *NoVal* almacena todas las acciones que se van a empezar a ejecutar en el intervalo actual de simulación.

Si la lista *NoVal* no está vacía entonces la evolución del tráfico almacenada en la pizarra no es coherente con la evolución real por lo que el sistema debe de realizar un proceso de actualización de sus datos. Este proceso implica la localización de la acción que más pronto se ejecutó en el tráfico real.

Sea T_{old} el instante de comienzo de la ejecución de esa acción. El sistema calcula el estado del tráfico para el instante T_{old} y elimina todos los datos cualitativos asociados a un tiempo posterior a T_{old}. La evolución de cada objeto del sistema se vuelve a simular teniendo en cuanta la ejecución de estas acciones hasta que se alcance el instante de comienzo del nuevo ciclo, T_{sim}. Una vez finalizada esta “retrosimulación”, el sistema debe de depositar estos datos en el TKB para que el resto de agentes puedan utilizarlos cuando lo consideren oportuno.

Esta acción lleva asociada el envío al resto de agentes del sistema un mensaje con el tipo de acción ejecutada y como consecuencia estos agentes deben de actualizar sus datos internos, y revisar sus peticiones de ejecución puesto que algunas de estas ya no serán válidas.

Aquellas acciones externas de control que deban ejecutarse en el intervalo actual de simulación (contenido de la lista *NoVal*) deben de indicarlo a cada uno de los objetos relacionados para que llegado el instante de ejecución esta acción se realice.

Las acciones de control externas que puede ejecutar el sistema son las ya descritas como *modificación del vector de reparto* de un cruce, *modificación del valor de coordinación* de un cruce, y *modificación de la longitud del ciclo* para un CCI.

Las acciones de control sobre cruces sólo pueden ejecutarse en el instante de comienzo de la primera fase, mientras que las acciones de control sobre CCIs sólo se pueden ejecutar en el comienzo del ciclo. Una limitación impuesta en este prototipo es que sólo se permite ejecutar una acción de control externa por objeto del sistema en cada momento, y hasta que esa acción finalice no se permite ejecutar otras acciones

sobre ese objeto. Además, también se ha añadido la opción de escoger el modo en que estas acciones se ejecutan: múltiples pasos (en incrementos de cinco, diez o quince segundos) o en un sólo paso.

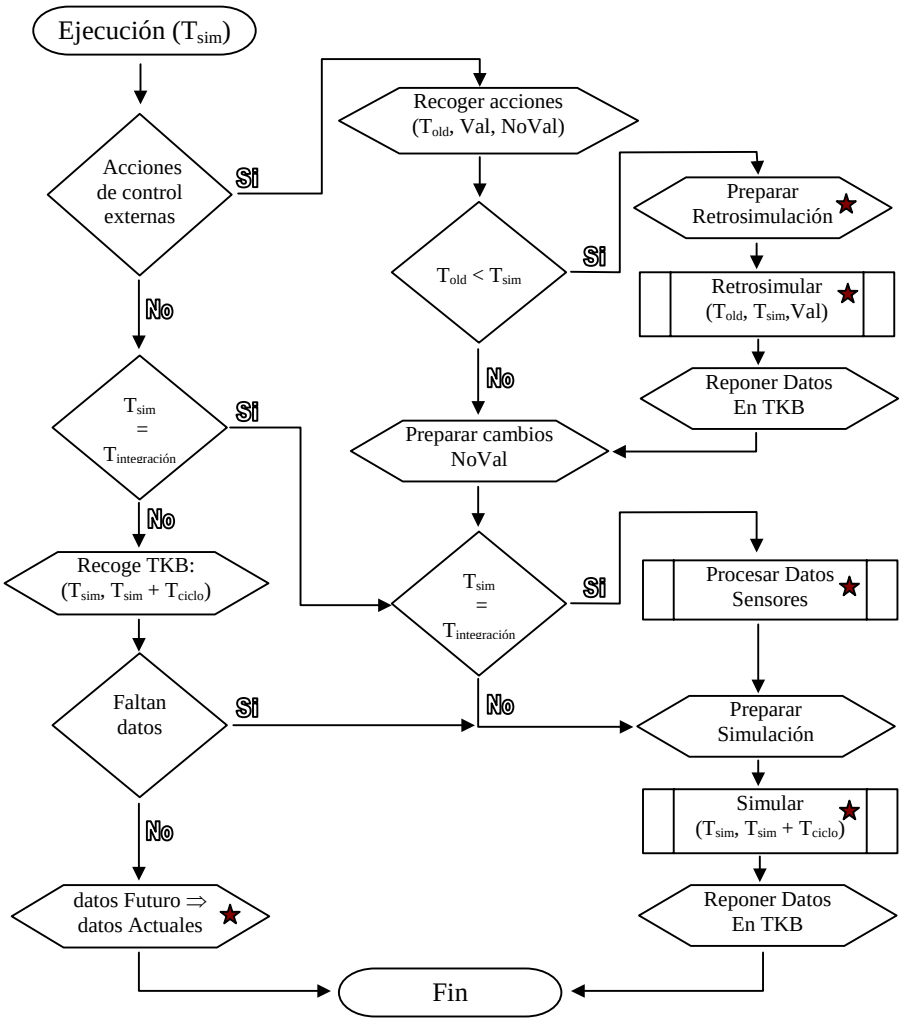


Fig 3-8 *Diagrama de flujo de la ejecución del agente de preproceso. Los elementos marcados con una estrella producen eventos (a ser tratados por el agente de control) o envían mensajes a otros agentes con información que les pueda ser útil en su posterior ejecución.*

En el caso de que T_{sim} coincida con el primer ciclo de un nuevo periodo de integración, el sistema debe de procesar los datos de los sensores, pues estos ya estarán disponibles, y en función de ese análisis modificará el estado inicial de cada objeto para adaptarse mejor a estos datos [García, 97].

Si ha habido acciones de control externas y T_{sim} no es el primer ciclo de un nuevo periodo de integración, el agente de preproceso debe de calcular la evolución para este

ciclo, puesto que la ejecución de estas acciones ha invalidado los estados cualitativos que hubieran sido calculados por el agente de predicción.

En el supuesto que no haya habido acciones de control externas, y que T_{sim} no sea el primer ciclo de un nuevo periodo de integración, el sistema busca en el panel de la aplicación si el nivel *futuro* contiene los datos de este nuevo ciclo. En el caso de que estén todos estos datos ya calculados, el agente de preproceso los cambia al nivel *actual*, indica al resto de agentes esta nueva situación y finaliza su ejecución.

En la situación en la que los datos del nivel *futuro* no estén calculados para el intervalo de simulación actual, el agente de preproceso debe calcularlos directamente, y para ello calcula el estado del tráfico en el instante T_{sim}, realiza la simulación para todo este intervalo y finalmente vuelca los resultados en el nivel *actual* del panel de la aplicación.

Cada cambio cualitativo que se genere en el sistema, en cualquiera de sus procesos de simulación, retrosimulación, o transferencia de datos del nivel futuro al nivel actual, provoca el envío de un mensaje al agente interfaz para que este cambio sea visualizado en la ventana que representa el estado actual del tráfico.

Las acciones de control propuestas por el propio sistema tienen un tratamiento diferente al de las acciones de control externas. Estas últimas son ejecutadas de forma poco frecuente por el usuario del sistema, mientras que las acciones de control propuestas por el sistema, en el caso que el sistema esté funcionando en el modo de ejecución automático se ejecutan de forma periódica cada $t_{integración}$ segundos; siempre y cuando el sistema disponga de tiempo suficiente para poder calcular estas propuestas.

Cuando el sistema funciona en el modo *automático* todas las acciones propuestas se ejecutan durante el próximo ciclo de simulación de forma simultanea a como se ejecutan en el tráfico real. Para realizar esta tarea se almacena en cada objeto una marca que indica que tiene una acción de control a ejecutar pendiente, y que cuando sea el momento adecuado, ésta debe de ejecutarse.

El agente de preproceso también realiza una primera tarea de identificación de situaciones problemáticas elementales y temporales de tráfico a medida que el simulador calcula la evolución cualitativa de cada objeto del sistema. De esta forma el agente de Detección tendrá identificadas antes de su ejecución estas situaciones problemáticas, puesto que se les serán comunicadas mediante eventos al agente de control y, por lo tanto, su ejecución será más rápida que si tuviera que rastrear toda la pizarra en busca de todas las situaciones potenciales de problemas de tráfico.

3.5 El Agente de Predicción

El agente de predicción realiza una estimación de cual puede ser la evolución del tráfico futuro en relación con el estado del tráfico actual proporcionado por el agente de preproceso. El núcleo fundamental de este agente consiste en ejecutar el proceso de

simulación cualitativa sobre intervalos de duración T_{ciclo} hasta alcanzar el tiempo máximo de predicción posible.

Esta tarea se realiza bajo la suposición de que no se ejecutan acciones de control externas y no se generan incidentes durante el actual periodo de integración. El sistema cuando recibe los datos de los sensores, los analiza y elimina los datos dinámicos del nivel de futuro en el panel de la aplicación, puesto que la situación habitual es que la evolución del tráfico en el sistema se haya desviado ligeramente del modelo de tráfico implementado en el simulador, y para evitar que estas desviaciones crezcan se parte de un nuevo conjunto de datos iniciales (asociados a estos datos de los sensores) cada periodo de integración.

Esta desviación puede hacerse mucho mayor, e incluso invalidar absolutamente toda la simulación realizada, en el caso que se presenten incidentes en el sistema.

3.5.1 Esquema de Funcionamiento

La tarea de predicción se define como perteneciente a la segunda clase. Esta propiedad indica la dependencia de datos iniciales para la ejecución de este agente respecto de los resultados proporcionados por el agente de preproceso (primera clase). Por lo tanto, este agente, en su primera ejecución, lee del panel de la aplicación en su nivel actual los datos cualitativos finales para cada objeto del sistema, y a partir de éstos calcula cual puede ser su evolución para los próximos T_{ciclo} segundos en el supuesto que no se ejecuten acciones de control externas ni ocurran incidentes en el sistema.

En el caso de que se hubiesen ejecutado acciones de control externas, el agente de predicción elimina del panel de la aplicación el nivel futuro, puesto que esta acción externa ha sido ejecutada por el agente de preproceso, y los datos correctos para empezar la nueva predicción de futuro están en el nivel actual.

Si estas acciones de control externas ya han comenzado, pero todavía no han finalizado, entonces este agente desestima calcular la predicción de futuro hasta que estas acciones de control externas finalicen de ejecutarse. Esta decisión la comunica al agente de control mediante el envío de un mensaje.

Si la ejecución actual no se corresponde con la primera ejecución del agente de predicción (es decir, si $T_{pred} > T_{sim}$) entonces este agente busca cuales han sido los últimos datos generados en simulación de futuro para cada uno de los objetos del sistema. A partir de éstos calcula los eventos asociados a cambios de fase y de ciclo en cada cruce para el tiempo de simulación que resta hasta alcanzar el tiempo máximo de simulación para este ciclo (T'_{ciclo}).

El valor de la variable T_{pred} indica el instante de comienzo de la simulación de futuro y está comprendido en el intervalo $[T_{sim} + NT_{ciclo}, T_{sim} + (N+1)T_{ciclo}]$, mientras que la variable T'_{ciclo} tiene el valor $(N+1)T_{ciclo}$, e indica el valor final que debería de alcanzarse en la presente ejecución del agente de predicción.

La siguiente tarea que realiza el agente de predicción es realizar la simulación del sistema en el intervalo $[T_{pred}, T'_{ciclo}]$. Durante esta simulación el agente de predicción busca situaciones problemáticas elementales y temporales. Cada una de estas situaciones provoca la generación de un evento que será tratado posteriormente por el agente de control. Estos eventos incluyen los datos asociados a cada problema detectado y serán convertidos en peticiones de ejecución del agente de Detección para así facilitarle su trabajo (al igual que sucedía con el agente de preproceso).

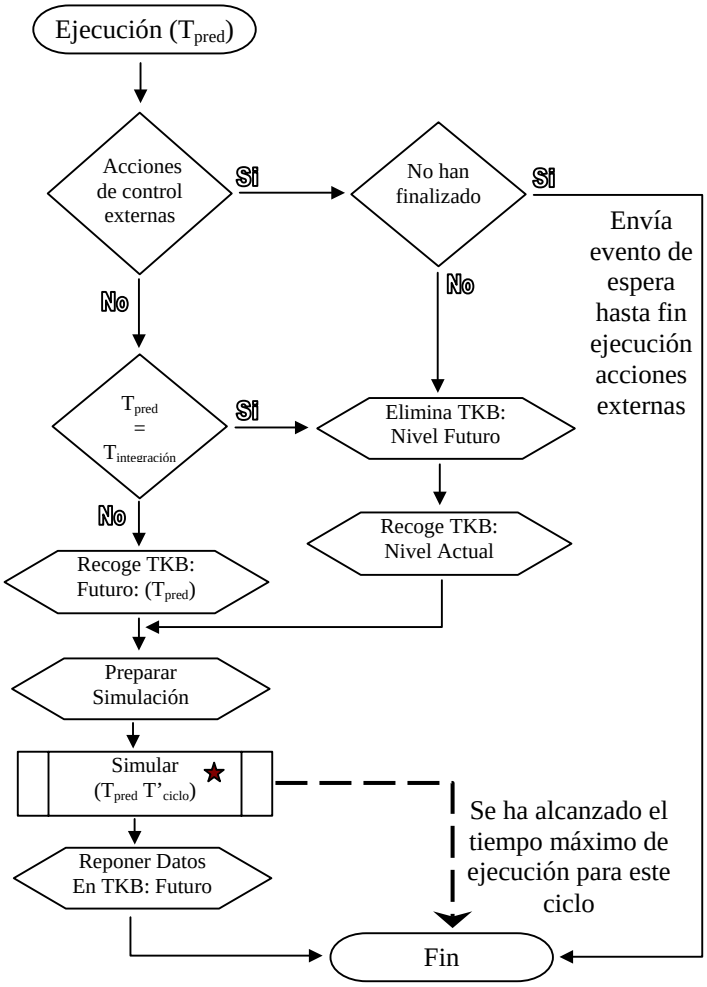


Fig 3-9 *Diagrama de flujo de la ejecución del agente de predicción. El elemento marcado con una estrella produce eventos (a ser tratados por el agente de control) o envía mensajes a otros agentes.*

La tarea de simulación en este agente se puede interrumpir debido a que se alcance el tiempo máximo dedicado a las tareas de un ciclo, es decir, cuando el tiempo real alcance al tiempo de simulación de futuro. Esta situación provoca que se acabe prematuramente la simulación y se guarde en el nivel futuro los datos cualitativos calculados hasta ese instante.

Por último, el agente de predicción en el caso que todavía no haya alcanzado el tiempo máximo de predicción de futuro, construye un evento de petición de ejecución para el cálculo del próximo ciclo de simulación de futuro y lo envía para ser tratado por el agente de control.

Cada cambio cualitativo que se genere en el sistema durante el proceso de simulación de futuro provoca el envío de un mensaje al agente interfaz para que este cambio sea visualizado en la ventana que representa el estado futuro del tráfico.

3.6 El Agente de Detección

El agente de detección analiza cual es el estado del tráfico actual (proporcionado por el agente de preproceso) y futuro (proporcionado por el agente de predicción) para determinar cuales son las situaciones problemáticas debidas a malas regulaciones en los cruces.

Las situaciones problemáticas asociadas a problemas elementales y temporales (apartado 3.4) son suministradas a este agente como datos de entrada. A partir de estos datos el agente identifica las situaciones problemáticas de la tercera clase, los problemas espacio-temporales, en la representación almacenada de la red urbana del sistema. Más concretamente, este agente calcula los objetos del sistema en que se cumplen las propiedades *bucle_cerrado* y *camino_cerrado* [García, 00a].

3.6.1 El Grafo de Dependencias Espacio-Temporales.

Para realizar estas tareas el agente construye dos grafos dirigidos con pesos en los que se representan tanto las dependencias temporales como las espaciales: un grafo está asociado al nivel actual del panel de la aplicación (y que, por lo tanto, sus datos serán suministrados por el agente de preproceso), y otro grafo se asocia al nivel futuro del panel de la aplicación (en el que sus datos son proporcionados por el agente de predicción de futuro).

Además de estos dos grafos el agente de detección también almacena todos los problemas detectados en el intervalo de integración anterior en un grafo de dependencias pasado.

Los nodos de cada grafo están asociados a situaciones problemáticas temporales. Cada uno de ellos almacena la siguiente información:

- Identificador del cruce en el que se detecta una situación problemática temporal.
- Valor de congestión del cruce. Puede tomar los valores: *0* indica congestión nula, *1* indica congestión media y *10* indica congestión completa.
- Intervalo temporal continuo asociado a ese valor de congestión: $[T_i, T_f]$.

- Los identificadores de los segmentos de salida en el cruce para los que se han detectado que se cumple la propiedad *cruce_cerrado* en algún subintervalo del intervalo $[T_i, T_f]$.

- Todos los identificadores de los segmentos de entrada al cruce.

Los arcos de cada grafo indican las dependencias espaciales entre los nodos que conectan. Es decir, para todo arco A_j, tal que este arco conecta dos nodos N_1 y N_2 en el grafo, se cumple que existe un segmento l_1 de salida del cruce asociado a N_1 que es segmento de entrada en el cruce asociado al nodo N_2 y, además, l_1 es uno de los segmentos que han motivado que la propiedad *cruce_cerrado* se cumpla en el nodo N_1. Cada arco del grafo está etiquetado con un peso que indica el valor de congestión del cruce asociado al nodo origen.

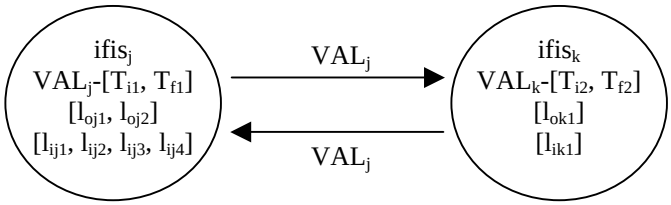


Fig 3-10 *Grafo empleado para representar las dependencias espacio-temporales de las situaciones problemáticas elementales y temporales detectadas por otros agentes de la aplicación. Se ha de cumplir que* l_{oj1} *o* l_{oj2} *sea un segmento de entrada para el cruce* ifis$_k$. *Además,* $l_{oj1} = l_{ik1}$ *o* $l_{oj2} = l_{ik1}$. *El segmento* l_{ok1} *debe de estar contenido en el conjunto* $[l_{ij1}, l_{ij2}, l_{ij3}, l_{ij4}]$. *Los valores* VAL$_j$ *y* VAL$_k$ *representan, respectivamente, el grado de congestión de las intersecciones* ifis$_j$ *e* ifis$_k$.

3.6.2 Búsqueda en el Grafo de Dependencias Espacio-Temporales.

El agente de detección construye cada grafo de forma incremental. Según recibe nuevos datos asociados a situaciones problemáticas elementales y temporales el agente de detección actualiza el grafo. Esta actualización implica la ejecución de algunas de estas tareas [García, 00a]:

- Eliminar nodos del grafo cuando sus cruces asociados tienen un valor de congestión nulo para el último intervalo temporal de duración T_{ciclo} ejecutado. También se han de eliminar sus arcos de entrada y de salida.

- Añadir nodos al grafo cuando se recibe un problema temporal para el que no existe nodo en el grafo. Esta actualización se completa mediante el cálculo de aquellos arcos de entrada y de salida a los nuevos nodos asociados a nodos existentes en el grafo.

- Modificar nodos del grafo cuando se reciben para éstos nuevos problemas temporales cuyo valor de congestión es medio o completo. Estas

modificaciones pueden ser de ampliación del intervalo temporal de congestión, cuando hay una continuidad entre los intervalos temporales anterior y actual, o de inserción de nuevo intervalo temporal de congestión en el caso que el intervalo temporal anterior y el recibido sean discontinuos. El valor del peso de cada arco de salida del nodo modificado pasa a ser el último valor de congestión recibido.

Cuando el agente de detección acaba de actualizar el grafo comienza a realizar la búsqueda de aquellos caminos que representen las propiedades *camino_cerrado* y *bucle_cerrado*. Estos caminos estarán formados por nodos del grafo unidos mediante arcos que conserven un intervalo temporal de congestión común a todos ellos.

Además, para evitar que un único problema no sea considerado como la suma de varios subproblemas, esta búsqueda deberá de considerar sólo caminos de longitud máxima sujetos a las restricciones temporales de sus intervalos; es decir, un bucle en el grafo cuyos nodos no tengan un intervalo temporal de congestión común no debe de ser considerado como un bucle, sino como una sucesión de propiedades *camino_cerrado* discontinuas temporalmente.

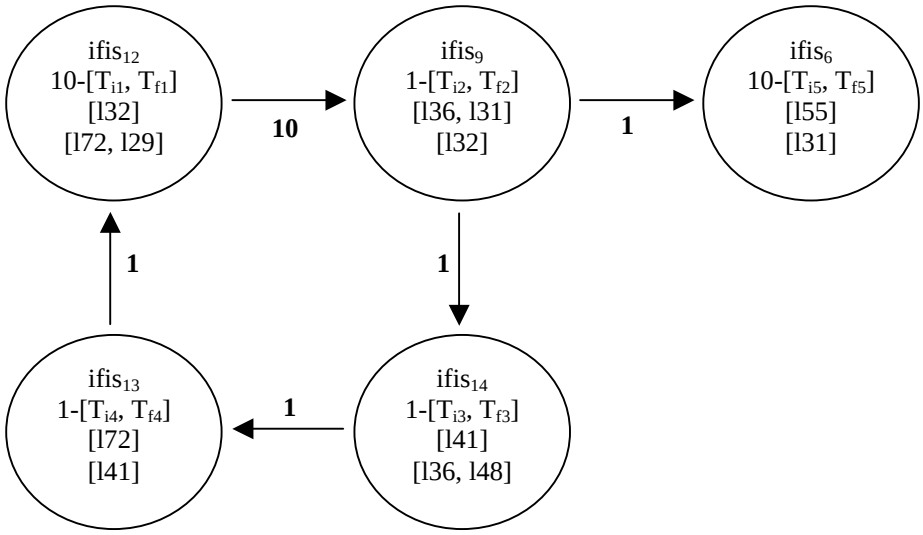


Fig 3-11 *Bucle en el grafo de dependencias espacio-temporales. Para que este bucle sea considerado como tal es necesario que* $[T_{i1},T_{f1}] \cap [T_{i2},T_{f2}] \cap [T_{i3},T_{f3}] \cap [T_{i4}, T_{f4}] = [T_{ini}, T_{fin}] \neq \varnothing$. *Aunque el cruce* $ifis_6$ *no forma parte propia de este bucle, el mecanismo de búsqueda de bucles lo incluye si se cumple que* $[T_{ini}, T_{fin}] \cap [T_{i5}, T_{f5}] \neq \varnothing$, *es decir, se considera que el problema temporal detectado en el cruce* $ifis_6$ *es consecuencia del propio bucle.*

Por otra parte, un bucle en el que exista ese intervalo común para todos sus nodos, que tenga adyacente otro nodo que no forme bucle, pero cuyo intervalo temporal de congestión tenga intersección no nula con el del bucle, debe de ser considerado como integrante del bucle, aunque físicamente no lo sea. Esto es así debido a que el problema

de congestión del nodo adyacente al bucle sea consecuencia lateral del propio problema detectado en el bucle.

La suma de todos los pesos de los arcos pertenecientes a un camino del grafo proporciona una medida de su grado de congestión. Para el bucle del grafo de la figura 3-10 se considera que el grado de congestión de todo el bucle es de **14** durante el intervalo $[T_{ini}, T_{fin}] \cap [T_{i5}, T_{f5}]$ si este intervalo no es nulo, en caso contrario se considera que el grado de congestión es **13** en el intervalo $[T_{ini}, T_{fin}]$.

Todas estas tareas son realizadas por los predicados *analizar_problemas1y2/3* y *analizar_problemas3/1*. Estos predicados se ejecutan cada vez que el agente de control concede derecho de ejecución al agente de detección con nuevos datos asociados a problemas elementales y temporales proporcionados por el agente de preproceso y el agente de predicción.

```
analizar_problemas1y2([], Gnew, Gnew).
analizar_problemas1y2([pevent(Cruce, T, Tipo)|Ps], G, Gnew ):-
   ( Tipo = [blocked(Segmentos)] ->  %% Se ha detectado un prob. elemental
      ( Segmentos == [] ->      %% No hay problemas en el cruce
            Gnueva = G ;
            ( envía_msg(interfaz, elemental, T, [Cruce,Segmentos])),
              Gnueva = G )
      ) ;
      ( Tipo = [nivel_congestión(VAL), blocked(Segmentos)], %% Prob. temp.
        ccis(LCcis),
        busca_ccis(LCcis, Cruce, Tciclo),
        Tini is T - Tciclo,
        cruces(Cruces),
        memberchk(Cruce-SSal, Cruces),
        NodoVacio =.. [Cruce, Info, _, _],
        envía_msg(interfaz,  temporal, T, [Cruce, VAL-Segmentos])),
        envía_msg(diagnosis,  temporal, T, [Cruce, VAL-Segmentos])),
        ( VAL == 0 ->   %% Si el grado de congestion es nulo se elimina del grafo
            ( del_vertices(G, [NodoVacio], Gnueva) -> true ; true ) ;
            ( member(NodoVacio-ArcSal, G) -> %% VAL = 1|10 y Nodo pert. a G
                  ( Info = _-[Ta,Tb],
                    delete(G, NodoVacio-ArcSal, Gaux),
                    nuevos_arcos(ArcSal, VAL, NArcSal),
                    ( Tb = Tini ->  %% Se detecta su inter. temp. de congestion
                          Nodo =.. [Cruce,VAL-[Ta, T],Segmentos,SSal] ;
                          Nodo=..[Cruce,VAL-[Tini,T],Segmentos,SSal]  ),
                     Gaux2 = [Nodo-NArcSal|Gaux],
                     modif_nodos(Gaux2, NodoVacio, Nodo, [], Gnueva)  )
                          ;                    %% VAL = 1|10 y Nodo no pert. a G
```

```
                                    ( Nodo =.. [Cruce, VAL-[Tini, T], Segmentos, SSal],
                                      arco(Cruce, Destinos),
                                      cons_pesos(G, Destinos, ArcSal),
                                      nuevos_arcos(ArcSal, VAL, NArcSal),
                                      Gaux = [Nodo-NArcSal|G],
                                      sus_entradas(Gaux, Cruce, Nodo, [], Gnueva) )
                 ) ) )      ),
                analizar_problemas1y2(Ps, Gnueva, Gnew).
```

El predicado *analizar_problemas1y2/3* actualiza el grafo de dependencias según sean los nuevos problemas elementales y temporales detectados en la simulación actual o futura.

Los parámetros de entrada a este predicado son: `P = [pevent(Cruce, T, Tipo) | Rprob]` que es la lista de nuevos problemas detectados y `G` que representa el grafo de dependencias temporales y espaciales actual. Como salida se obtiene un nuevo grafo de dependencias `Gnew` que es una actualización del grafo `G`.

El predicado *busca_ccis/3* se encarga de localizar el tiempo de ciclo del CCI al que pertenece el cruce con problemas temporales. El resto de predicados no elementales se encargan de realizar tareas de reconstrucción sobre el grafo: *nuevos_arcos/3* se encarga de asignar el valor `VAL` como nuevo peso de cada arco, *modif_nodos/5* modifica el grafo de manera que cada nodo que tenía una conexión de salida a un nodo que se ha modificado la tenga ahora al nodo ya actualizado, `arco/2` es un hecho de la base de datos que informa de todos los posibles destinos de cada cruce, `cons_pesos/3` es una herramienta auxiliar para determinar de todos los destinos posibles de un nodo sólo los que están presentes en el grafo, y `sus_entradas/5` construye el grafo para que se introduzcan las nuevas dependencias espaciales que el nodo nuevo recibe a su entrada con respecto a los nodos actuales del grafo.

El predicado *analizar_problemas3/1* busca en el grafo actualizado los patrones de problemas asociados a las propiedades *camino_cerrado* y *bucle_cerrado*.

```
analizar_problemas3(Gnew) :-
    vertices(Gnew, Vertices),
    busca_caminos(Vertices, [], Gnew, CLazo, CNoLazo),
    comprueba_lazos(CLazo, CNoLazoAux),
    elimina_nodos_aislados(CNoLazo, CnoLazoAux, CNoLazoNew),
    comprueba_nolazos(CNoLazoNew), !.
```

El predicado *analizar_problemas3/1* recibe como parámetro de entrada un grafo de dependencias espacio-temporales, y la tarea que realiza consiste en identificar todos sus caminos abiertos de longitud máxima, así como todos sus bucles, atendiendo tanto a sus características espaciales como temporales.

El predicado *vertices/3* se encarga de asociar los vértices del grafo Gnew a la lista Vertices. El predicado *busca_caminos/5* realiza un proceso de búsqueda por el grafo Gnew para determinar los caminos que forman bucle (CLazo) y los caminos máximos que no forman bucle (CnoLazo) atendiendo únicamente a sus características espaciales.

Las características temporales de cada camino que forma un bucle se consideran en el predicado *comprueba_lazos/2*. Cada uno de los bucles en los que su intervalo temporal de congestión no sea el vacío se envía al agente interfaz y al agente de diagnosis. Si en un bucle espacial se detecta que no hay intervalo temporal de congestión común entonces este bucle se convierte en secuencia de caminos abiertos que han de ser considerados en el predicado *comprueba_nolazos/1*. Por otra parte, el predicado distingue entre rotondas e intersecciones en el sentido en que si en una intersección existe un segmento de entrada y otro de salida en la misma dirección, entonces el camino que incluye a ese segmento de entrada no puede incluir al segmento de salida. Por ejemplo, en la figura 3-10 se muestra una situación de este tipo: incluso en el caso que $[T_{i1}, T_{f1}] \cap [T_{i2}, T_{f2}] \neq \varnothing$ en el grafo se identifican dos situaciones de *camino_cerrado* (y no una situación de *bucle_cerrado*).

Como consecuencia de la ejecución de estos predicados pueden hacer sido identificados caminos de longitud cero en el grafo, por lo que lo primero que se hace es eliminar estos caminos en el predicado *elimina_nodos_aislados/3*.

La última tarea, realizada por el predicado *comprueba_nolazos/1*, consiste en identificar, del conjunto de caminos detectados que no forman bucle, todos aquellos que no tengan la longitud máxima posible atendiendo simultáneamente a sus características temporales y espaciales. Estos caminos se eliminan y el resto se envían a los agentes de interfaz y diagnosis.

3.6.3 Esquema de Funcionamiento

Este agente no precisa acceder a los datos dinámicos de cada objeto del sistema almacenados en los niveles del panel de la aplicación debido a que una gran parte de su trabajo (la identificación de las situaciones problemáticas elementales y temporales) ya ha sido realizado por el agente de preproceso y por el agente de predicción. Por lo tanto, todos los datos dinámicos que precise le son accesibles desde los datos suministrados por la petición de ejecución construida por el agente de control.

En el caso de que se hayan ejecutado acciones de control externas y éstas hubiesen finalizado su implementación, el agente de detección elimina del panel de la aplicación en el nivel futuro el grafo de dependencias temporales futuro, y actualiza el grafo de dependencias actual para que no contenga valores superiores a T_{nof}, siendo este valor el instante de comienzo de la acción de control externa que antes se haya ejecutado.

Si las acciones externas todavía no han concluido de ejecutarse entonces el agente de detección envía un mensaje al agente de control para que desestime crearle peticiones de ejecución hasta que estas acciones externas acaben de ejecutarse.

En el comienzo de cada ciclo de integración de datos de los sensores el sistema se inicializa, y por lo tanto elimina todos los datos asociados a problemas futuros detectados en predicción que se encuentren representados en el grafo de dependencias futuro. Los datos correspondientes al grafo de dependencias actual se transfieren al grafo de dependencias pasado y se asocia un grafo vacío con el nuevo grafo de dependencias actual.

La única tarea que le resta por realizar al agente de detección consiste en determinar si los problemas detectados pertenecen al grafo de dependencias actual o al de futuro. Una vez determinado, entonces actualiza el grafo asociado enviando los mensajes adecuados al agente de interfaz y al de diagnosis comunicándoles los problemas de la tercera clase detectados.

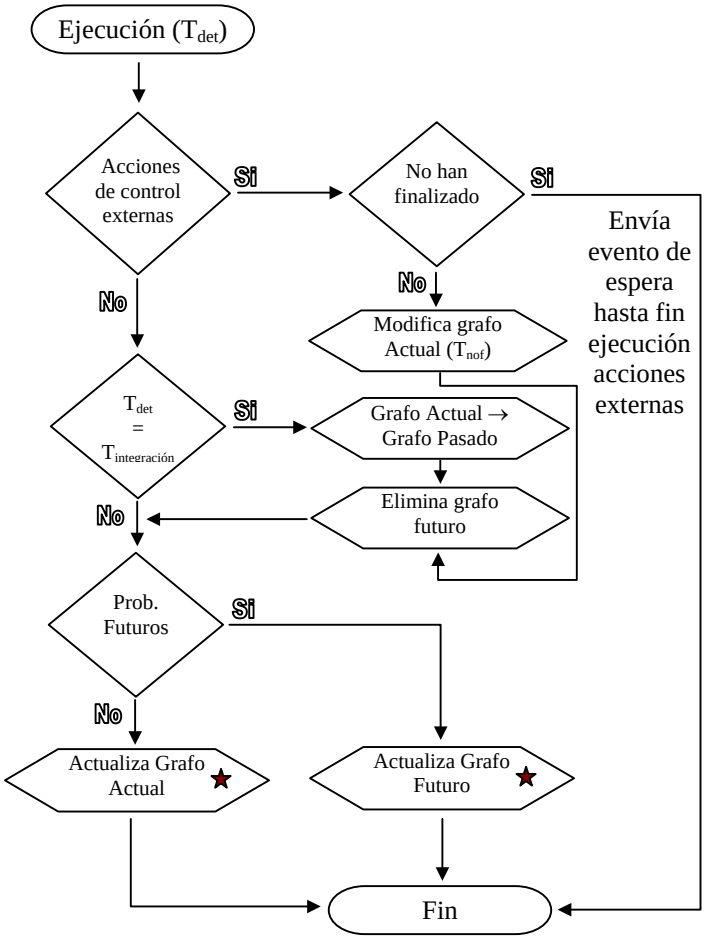


Fig 3-12 *Diagrama de flujo de la ejecución del agente de detección. Los elementos marcados con una estrella producen mensajes dirigidos a los agente de interfaz y de diagnosis.*

3.7 El Agente de Diagnosis

El agente de diagnosis determina si las situaciones problemáticas detectadas en la red de tráfico urbano son debidas a malas regulaciones en los semáforos o son debidas a la presencia de incidentes (como pueden ser las situaciones de accidentes de tráfico, vehículos mal estacionados, obstáculos que interrumpen el tráfico en los segmentos, etc.).

La presencia de incidentes en el sistema provocan situaciones de congestión inesperadas que, según transcurre el tiempo, se van expandiendo a segmentos y cruces adyacentes. Por este motivo resulta fundamental que sean detectados de forma rápida, para así poder promover las acciones de tráfico adecuadas que intenten minimizar o eliminar sus efectos.

Esta tarea se realiza durante el primer subciclo de ejecución del ciclo de integración de datos. Los datos iniciales que precisa este agente para su ejecución los recibe de mensajes enviados por los agentes de detección de problemas y de preproceso.

El agente de detección de problemas envía las situaciones problemáticas detectadas en la red urbana durante el último periodo de integración de datos al agente de diagnosis. Cada una de estas situaciones problemáticas (elemental, temporal o espacio-temporal) está asociada con el conjunto de datos cualitativos de cada uno de los objetos del sistema en el intervalo temporal en el que se ha producido cada situación.

El agente de preproceso lee los datos de los sensores y vuelve a analizar el estado del tráfico desde el instante inicial del anterior periodo de integración hasta el instante de comienzo del nuevo periodo de integración. Cada una de las situaciones problemáticas que detecta se la comunica al agente de diagnosis para que la compare con las situaciones problemáticas procedentes del agente de detección de problemas obtenidas por simulación cualitativa durante el periodo de integración anterior.

En el sistema de control de tráfico urbano propuesto el momento en que estos incidentes pueden ser detectados varía entre un mínimo de unos pocos segundos (el incidente comenzó pocos segundos antes de que se enviasen los datos de los sensores al sistema) a un máximo de $t_{integración}$ segundos (en el caso de que el incidente hubiese comenzado justo después del envío de datos de los sensores al sistema). Estos márgenes se obtienen considerando que el agente de diagnosis sea capaz de detectar a la primera las situaciones debidas a incidentes (que es una situación ideal que no resulta fácil de conseguir) [García, 00b].

Esta característica de retardo en la detección de incidentes permite al sistema establecer un primer filtro entre incidentes de tráfico persistentes y puntuales. Los incidentes de tráfico puntuales no precisan de ninguna acción de intervención puesto que tienden a desaparecer por el propio comportamiento del tráfico. Los incidentes de tráfico persistentes identifican situaciones peligrosas para la regulación del tráfico y una vez detectadas deben ser comunicadas al responsable de la sala de control de tráfico para que tome las medidas oportunas.

Por otra parte, también resulta especialmente importante para la gestión de tráfico que estas medidas a adoptar ante la presencia de incidentes persistentes se tomen lo antes posible para intentar evitar la expansión del área de congestión del incidente a áreas vecinas, por lo que resulta también fundamental para el buen funcionamiento del sistema que los incidentes de tráfico persistentes sean detectados lo antes posible.

3.7.1 Breve Descripción del Método de Ejecución

El modelo utilizado en este sistema para realizar esta comparación entre situaciones problemáticas procedentes del agente de detección y el análisis del estado del tráfico urbano calculado por el agente de preproceso con los datos procedentes de los sensores se basa en las definiciones de tipos de incidentes expuestos en el marco del proyecto EQUATOR [EQUATOR, 94].

Este método consiste en buscar segmentos que formen parte de las situaciones de *camino_cerrado* o de *bucle_cerrado* en los que se haya producido una pérdida de capacidad en su tráfico. Esta pérdida puede haber sucedido de dos formas distintas. La primera de ellas es la presencia de un *patrón de atasco inesperado* y la segunda es la presencia de un *patrón de ausencia de tráfico inesperado* [García, 00b].

Ambas propiedades se calculan reduciendo el estado cualitativo del tráfico en cada segmento perteneciente a cada una de estas situaciones problemáticas durante su intervalo temporal de duración a un único valor. Para realizar esta tarea el agente de diagnosis accede al nivel pasado del panel de la aplicación del TKB para recoger esos datos asociados a ese intervalo temporal.

Las fórmulas utilizadas para condensar la evolución del flujo de tráfico en un segmento durante un cierto intervalo temporal en un único valor son las siguientes:

$$FlujoAsociado\left(\text{segmento}_k, [t_s, t_f]\right) = \frac{\sum_{\forall q \in EQ} \left(v_q\left(\text{segmento}_k, [t_s, t_f]\right) \cdot flujo_q \right)}{\sum_{\forall q \in EQ} v_q\left(\text{segmento}_k, [t_s, t_f]\right)}$$

$$\forall q \in EQ;\ \ v_q\left(\text{segmento}_k, [t_s, t_f]\right) = \sum_{i=t_s}^{t_f} \sum_{j=l_s}^{l_f} (t_i - t_{i-1}) \cdot (l_j - l_{j-1}) \cdot \Phi(k, q, i, j)$$

$$\Phi(k, q, i, j) = \begin{cases} 1 & \text{si el subsegmento } [l_{j-1}, l_j] \text{ del segmento}_k \text{ tiene el valor cualitativo q en el intervalo } [t_{i-1}, t_i] \\ 0 & \text{en otro caso} \end{cases}$$

Fórmulas 1 *Conjunto de Fórmulas a aplicar a cada segmento que forme parte de un camino o bucle cerrado durante un cierto intervalo temporal.*

El resultado de la aplicación de esta fórmula es un valor entre *0* y *1*. El valor *0* indica una situación de flujo nulo en el tráfico, mientras que un valor *1* indica una situación de flujo máximo en el tráfico [García, 00b].

El predicado *resume_estado/4* calcula estos valores de la forma siguiente:

```
resume_estado([uevent(Segmento, T, C) | Rval], Ti, Long, Lval, Medida) :-
    localize(C, state, [_|L]),
    recorre_segmento(L, T-Ti, Lval, Lval_aux), !,
    resume_estado(Rval, T, Long, Lval_aux, Medida).

resume_estado([], _, Long, [V1, V2, v3, V4, V5, V6, V7, V8], Medida) :-
    af(densidad_tráfico2, d1, [_, _, F1]),
    af(densidad_tráfico2, d2, [_, _, F2]),
    af(densidad_tráfico2, d3, [_, _, F3]),
    af(densidad_tráfico2, d4, [_, _, F4]),
    af(densidad_tráfico2, d5, [_, _, F5]),
    af(densidad_tráfico2, d6, [_, _, F6]),
    af(densidad_tráfico2, d7, [_, _, F7]),
    af(densidad_tráfico2, d8, [_, _, F8]),
    Medida is (V1 * F1 + V2 * F2 + V3 * F3 + V4 * F4 +
                  V5 * F5 + V6 * F6 + V7 * F7 + V8 * F8) /
                 (V1 + V2 + V3 + V4 + V5 + V6 + V7 + V8).

recorre_segmento([], _, Lval, Lval).
recorre_segmento([[Li, Lf, EQ] | Rsegmento], T, Lval, Lval_aux) :-
    Lval = [V1, V2, V3, V4, V5, V6, V7, V8],
    ( EQ = d1, Lval2 = [V1 + T * (Lf – Li), V2, V3, V4, V5, V6, V7, V8] ;
      EQ = d2, Lval2 = [V1, V2 + T * (Lf – Li), V3, V4, V5, V6, V7, V8] ;
      EQ = d3, Lval2 = [V1, V2, V3 + T * (Lf – Li), V4, V5, V6, V7, V8] ;
      EQ = d4, Lval2 = [V1, V2, V3, V4 + T * (Lf – Li), V5, V6, V7, V8] ;
      EQ = d5, Lval2 = [V1, V2, V3, V4, V5 + T * (Lf – Li), V6, V7, V8] ;
      EQ = d6, Lval2 = [V1, V2, V3, V4, V5, V6 + T * (Lf – Li), V7, V8] ;
      EQ = d7, Lval2 = [V1, V2, V3, V4, V5, V6, V7 + T * (Lf – Li), V8] ;
      EQ = d8, Lval2 = [V1, V2, V3, V4, V5, V6, V7, V8 + T * (Lf – Li)] ),
    recorre_segmento(Rsegmento, T, Lval2, Lval_aux).
```

Los parámetros `[uevent(Segmento, T, C) | Rval]`, `Ti`, `Long`, `Lval` y `Medida` identifican los siguientes datos:

- `[uevent(Segmento, T, C) | Rval]`: Lista ordenada de todos los valores cualitativos para el segmento `Segmento` en el intervalo de detección del problema.
- `Ti`: Instante inicial de la presencia del problema.
- `Long`: Longitud del segmento multiplicado por la duración del intervalo temporal de presencia de la situación problemática asociada.

- Lval: Lista cuyo valor inicial es [0, 0, 0, 0, 0, 0, 0, 0].
- Medida: Resultado de la ejecución del predicado.

Mediante la ejecución de este predicado se consigue una medida de la congestión del tráfico en ese segmento para la situación detectada por el agente de detección, y otra medida de la congestión del tráfico en ese segmento para la situación detectada por el agente de preproceso con los datos de los sensores.

Si la diferencia entre la medida asociada al problema detectado por el agente de preproceso respecto de la medida asociada al problema detectado por el agente de detección es superior a una cierta cantidad umbral, cuyo valor se determina de forma experimental, entonces el segmento presenta un *patrón de atasco inesperado*. Si esa diferencia es inferior a esa misma cantidad umbral entonces el segmento presenta un *patrón de ausencia de tráfico inesperado*. En cualquier otro caso se asume que la desviación entre ambas medidas se corresponde con una mala regulación de las luces de los semáforos.

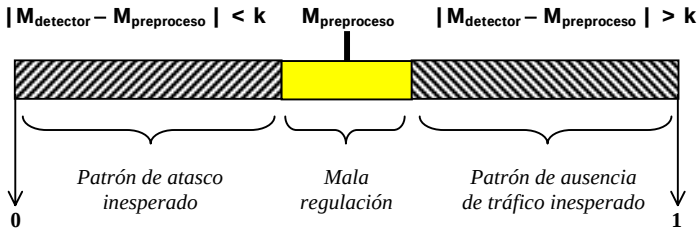


Fig 3-13 *Detección de patrones de atascos en un segmento en función de la diferencia entre la medida de la evolución cualitativa del segmento proporcionada por el agente de preproceso y la proporcionada por el agente de detección. El valor* ***k*** *identifica la cantidad umbral a partir de la cual diferenciar entre mala regulación e incidentes del tráfico.*

El sistema deduce que existe un ***incidente*** cuando todos los segmentos asociados a la propiedad *camino_cerrado* o *bucle_cerrado* presentan un *patrón de atasco inesperado* o un *patrón de ausencia de tráfico inesperado*. En el caso que alguno de sus segmentos no presente uno de estos patrones entonces el problema se asume que es debido a una ***mala regulación*** de las luces de los semáforos.

3.7.2 Esquema de Funcionamiento

La tarea del agente de diagnosis es una tarea de la segunda fase. Por lo tanto, cuando a este agente se le conceda derecho de ejecución, ya tendrá disponibles los datos preprocesados de los sensores del sistema respecto a la evolución cualitativa del sistema en el periodo de integración anterior. También estarán disponibles los problemas espacio-temporales detectados por el agente detector asociados a ese mismo intervalo temporal, puesto que esos datos fueron calculados durante el anterior periodo de integración.

Para cada uno de los problemas espacio-temporales detectados este agente busca y recoge los datos cualitativos del pasado para los intervalos temporales asociados a cada uno de ellos. Si al ejecutar el método de detección de incidentes se determina que alguna de estas situaciones se corresponde con un incidente en el sistema entonces esta información se le envía un mensaje al agente de interfaz para que el responsable de la sala de control de tráfico tome las medidas oportunas.

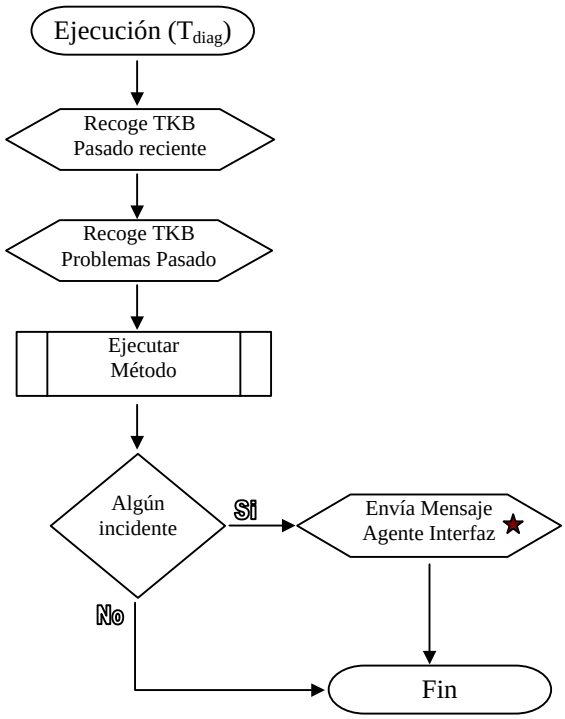


Fig 3-14 *Diagrama de flujo de la ejecución del agente de diagnosis. En el caso de que se detecte algún incidente, éste es transmitido al agente de interfaz para que lo comunique al responsable de tráfico en la sala de control.*

3.8 El Agente de Solución

La tarea del agente de solución consiste en determinar un vector de reparto a cada cruce durante todo el intervalo de integración actual que sea capaz de reducir el número de situaciones problemáticas generadas como consecuencia de la evolución del tráfico en toda la red urbana.

Resulta importante destacar que esta tarea se debe de realizar mediante un **análisis global** de cada uno de los cambios propuestos en la asignación de vectores de reparto a cruces del sistema, puesto que en caso contrario, si se buscan soluciones locales el resultado de su aplicación sobre el red urbana puede ser la traslación de los problemas locales detectados a otra zona de la red de tráfico.

3.8.1 Breve Descripción del Método de Ejecución

Para realizar esta tarea el agente de solución debe de estimar las ventajas e inconvenientes de la asignación de cada vector de reparto a cada cruce durante el periodo de evaluación. Como cada cruce tiene *8* posibles vectores de reparto distintos, el número de situaciones potenciales a evaluar durante el intervalo de simulación es 8^N siendo *N* el número de cruces de la red urbana, y estableciendo que durante este intervalo cada cruce sólo puede cambiar de vector de reparto una vez.

Cada una de estas situaciones expresa el vector de reparto a aplicar en cada momento para cada cruce durante el intervalo de simulación y recibe el nombre de *plan de señales de tráfico*.

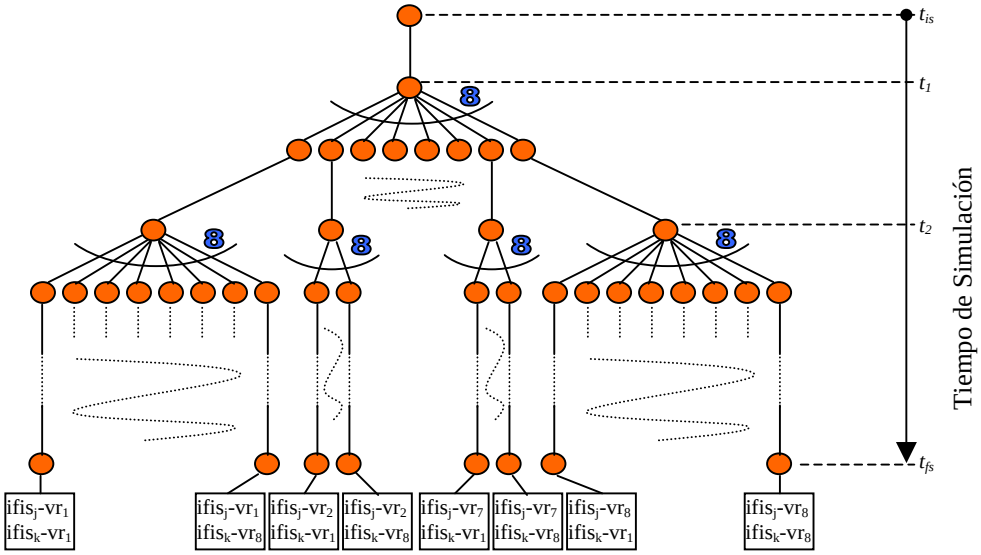


Fig 3-15 *Árbol de generación de cada posible plan de señales para un cci compuesto de sólo 2 cruces. La altura del árbol es función del número de cruces: altura = 2 + Ncruces. El cruce ifis$_j$ puede cambiar de reparto en el instante t_1, mientras que el cruce ifis$_k$ lo hace en el instante t_2. Este árbol tiene 64 hojas (8^2), cada una de ellas representa un plan de señales posible a aplicar en el intervalo de simulación $[t_{is},t_{fs}]$.*

Resulta imposible realizar una evaluación exhaustiva de cada uno de estos planes de tráfico ya que este proceso precisa de un tiempo de ejecución exponencial con respecto al número de cruces de la red de tráfico urbano. Por lo tanto, se necesita el establecimiento de unas heurísticas de trabajo que permitan eliminar grandes conjuntos de estas situaciones potenciales.

Una primera medida a considerar para reducir el tamaño del espacio de búsqueda consiste en la utilización del orden el que se declaran los distintos vectores de reparto

para cada cruce. Estos vectores se ordenan según el tiempo que dediquen al estado verde para la dirección principal del cruce; de máximo tiempo posible a mínimo tiempo posible.

Esta característica junto con la propiedad de que una vez establecido el estado del tráfico en un cruce, su tiempo de rojo (es decir, el tiempo del ciclo que este cruce no está en verde) determina la longitud de la cola de vehículos que se puede formar en ese cruce (a mayor tiempo de rojo mayor será la longitud de la cola de vehículos en esa entrada, figura 3-16) se puede utilizar para delimitar el conjunto de vectores de reparto aplicables para cada cruce.

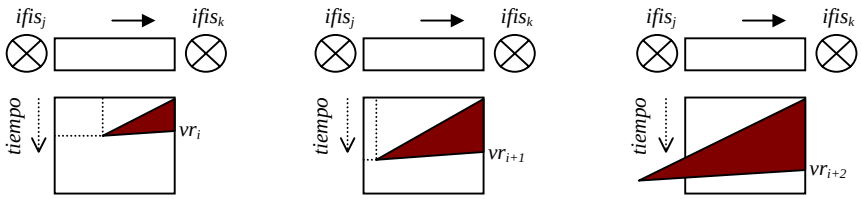


Fig 3-16 *Relación entre tiempo de rojo en una entrada (especificado según un determinado vector de reparto) y longitud de la cola de vehículos generada.*

En la figura anterior se observa que a medida que se incrementa el tiempo de rojo en la entrada a un cruce ($ifis_k$), la cola de vehículos que se genera tiende a crecer. Para vectores de reparto con índice superior a $i+1$ se observa que esta cola llega a invadir el cruce anterior ($ifis_j$) provocando la aparición de una situación problemática elemental. Por lo tanto, una primera medida para el cálculo de qué vectores de reparto son candidatos válidos para cada cruce consiste en eliminar todos aquellos vectores que produzcan situaciones problemáticas elementales.

El objetivo del agente de solución en este sistema se puede definir como la búsqueda de aquellos vectores de reparto para cada cruce que minimicen el número de situaciones problemáticas detectadas durante un cierto intervalo temporal de simulación. Es, por lo tanto, un problema de optimización que calcula el mínimo de una función. Dado que el tiempo de ejecución del que dispone el agente de solución para realizar este cálculo es un tiempo limitado, entonces el resultado devuelto se corresponde, al menos, con un mínimo local.

El dominio de los datos de entrada a este problema consiste de todos los posibles vectores de reparto para cada cruce, y su dominio de salida es el plan de tráfico calculado junto con un valor numérico, denominado el *nivel de congestión de la red de tráfico*, que proporciona una medida de las situaciones problemáticas detectadas durante el intervalo de simulación.

Con el propósito de simplificar y por lo tanto hacer más rápido, el proceso de evaluación de las situaciones problemáticas detectadas sólo se consideran aquellas

situaciones correspondientes a problemas elementales y temporales, no valorando las situaciones espacio-temporales que puedan aparecer en la red urbana.

Esta decisión, que puede parecer drástica o incluso inadecuada para los objetivos finales del agente de solución, se toma debido a las características del modelado de problemas de tráfico implementado. En esta organización de problemas, los problemas de niveles superiores se construyen mediante problemas pertenecientes a niveles inferiores y, por lo tanto, una situación de *bucle_cerrado* no aparece de forma espontanea, sino que se genera mediante la combinación de varias situaciones de *congestión_media* y de *congestión_completa* persistentes en la red de tráfico urbano. Este es el motivo por el que, de forma indirecta, la minimización del número de situaciones problemáticas temporales tiende a aminorar también el número de situaciones problemáticas espacio-temporales presentes en el sistema.

El *nivel de congestión de la red de tráfico* se calcula mediante la obtención de dos medidas íntimamente relacionadas:

- IP: Representa la evolución de los problemas en cada instante para cada cruce durante todos los ciclos que quepan en el presente intervalo temporal de simulación. Si esta medida se incrementa entonces representa una situación de incremento en la congestión del tráfico, si disminuye entonces representa una situación de decremento en la congestión.

- NP: Representa la acumulación de la evolución de las situaciones problemáticas detectadas hasta el momento ponderadas por su tiempo de permanencia, es decir, $NP = \sum_{\forall e \in Eventos} (t_{ini}(e) - t_{fin}(e)) * IP_{tfin}$.

El valor de la medida IP se calcula por medio del predicado *valproblemas/6* [Toledo, 96].

```
valproblemas([],_,I,I,SC,SC) :- !.

valproblemas([pevent(A,T,C) | P1], P, I, I2, SC, SC2) :-
  valproblemas2(A, T, C, P, II, SCC),  %% analizar cada problema de P1
  valproblemas(P1, P, I, I3, SC, SC3),
  I2 is I3 + II,
  SC2 is SC3 + SCC.

valproblemas2(A, T, C, [pevent(A,T2,C2) | P], II, SC) :- %% encontré el anterior
  !, (   memberof(blocked([]), C),     memberof(blocked([]), C2), I1 = 0  ;
         memberof(blocked([]), C), \+ memberof(blocked([]), C2), I1 = -1 ;
    \+ memberof(blocked([]), C),      memberof(blocked([]), C2), I1 = 1  ;
    \+ memberof(blocked([]), C), \+ memberof(blocked([]), C2), I1 = 0   ),
  !, (   memberof(congestion_level(I2a), C),  %% Si hay cambio a fase 1 en T
         (   memberof(congestion_level(I2b), C2),
             TT2 = T2  ;
```

```
                \+ memberof(congestion_level(_), C2),
                        valproblemas2b(A, P, TT2, I2b)   ),
                I2 is I2a  - I2b,          %% Grado de cong. actual – Grado de cong. pasado
                SC is I2 * (T - TT2)  ;  %%       ponderado por los segundos transcurridos.
        \+ memberof(congestion_level, C),
                I2 = 0,
                SC = 0     ),
    !, II is I1 + I2.

valproblemas2(A, T, C, [_ | L], II, SC) :- %% Busco el problema anterior del mismo
    !, valproblemas2(A, T, C, L, II, SC).      %%    cruce que sea más reciente

valproblemas2(_, _, C, [], II, 0) :-          %% No lo encontré
    !, (    memberof(blocked([]), C), II = 0 ;       %% No hay cruces_cerrados en T
        \+ memberof(blocked([]), C), II = 1), !.   %% Si hay  cruces_cerrados en T

valproblemas2b(_, [], T, 0) :- tini(T), !.      %% No existe cambio a fase 1 anterior
valproblemas2b(A, [pevent(A, T2, C2) | P], TT, I2b) :- %% Busca cambio a fase 1
    memberof(congestion_level(I2b), C2), TT = T2, !;
    !, valproblemas2b(A, P, TT, I2b).
valproblemas2b(A, [_ | P], TT, I2b) :-        %% Sigue buscando
    !, valproblemas2b(A, P, TT, I2b).
```

Los parámetros P1, P, I, I2 y SC2 identifican a los siguientes datos (el parámetro SC tiene 0 como valor inicial, su función es servir de acumulador para el cálculo del parámetro SC2):

- [pevent(A, T, C)| P1]: Nuevos problemas detectados C para el instante T en el cruce A.
- P: Problemas pasados ordenados de más recientes a más antiguos.
- I: Índice anterior.
- I2: Nuevo valor para el índice de problemas.
- SC2 Grado de congestión entre dos ciclos consecutivos por la duración del ciclo en segundos.

El índice I2 representa la evolución de los problemas, si $I2 > I$ es que la evolución del tráfico es hacia una situación más congestionada, si $I2 \leq I$ la evolución es hacia un tráfico más fluido.

Para tratar con todas estas características, estos métodos se implementan utilizando el paradigma de programación lógica basada en restricciones (CLP [Hentenryck, 89], [Marriot, 98]). A cada cruce se le asocia una variable con un dominio finito de valores entre 1 y 8 (representando cada uno de estos valores el índice de cada posible vector de reparto según la ordenación inicialmente establecida).

El método de minimización utilizado es *ramificación y poda* limitado a un tiempo máximo de ejecución. El proceso consiste en la simulación cualitativa del estado del tráfico en la red urbana durante el intervalo temporal de integración. Cada instante en que un cruce cambia al ejecutar su primera fase es un instante candidato para que ese cruce pueda revisar el vector de reparto que se está utilizando hasta ese momento. Para ello calcula el coste asociado a cambiar de vector respecto al coste asociado a mantener el vector actual.

El cálculo del coste asociado a un cruce y a un vector de reparto se realiza de la forma siguiente:

- Para cada segmento de entrada al cruce se recorre toda su historia pasada en este intervalo de simulación y se construyen dos vectores de ocho posiciones, uno para almacenar datos de su extremo inicial y el otro para almacenar datos de su extremo final. La posición *i* de cada vector representa el tiempo total que el extremo asociado ha permanecido en el estado cualitativo *di*. Cada uno de estos vectores se resume en un único valor que mejor represente la evolución en cada extremo del segmento. Al final de este proceso se consigue una lista con la siguiente información: $[[l_{i1}, Long_{i1}, EQ_{ini\_i1}, EQ_{fin\_i1}], ..,[l_{in}, Long_{in}, EQ_{ini\_in}, EQ_{fin\text{-}in}]]$ donde l_{ij} son los segmentos de entrada al cruce $ifis_i$, $Long_{ij}$ es la longitud del segmento l_{ij}, EQ_{ini_ij} representa el estado cualitativo del extremo inicial del segmento l_{ij}, y EQ_{fin_ij} representa el estado cualitativo del extremo final del segmento l_{ij}.

- Para cada segmento de entrada al cruce se le asocia un *coste* nulo si el segmento comienza en una entrada del sistema. Si comienza en otro cruce ($ifis_k$) entonces se plantean dos situaciones. Si el estado cualitativo de su extremo inicial es superior o igual a *d4* entonces el coste de esa entrada se define como el tiempo de rojo que este cruce permanece en cualquiera de los repartos posibles. Si el estado cualitativo inicial es inferior a *d4* entonces se plantea la posibilidad de que exista una cola intermedia en el segmento que crezca más rápidamente hacia el cruce $ifis_k$ que hacia el cruce que se está evaluando. En este caso el cruce se le define un coste constante y pequeño (por ejemplo el valor **12**), en caso de que no exista esta situación el valor del coste es nulo. Al final de este proceso se consigue una lista, Coste, que contiene la siguiente información: $[Coste_{l\_i1}, Coste_{l\_i2}, .., Coste_{l\_in}]$ donde $Coste_{i\_ji}$ puede ser *0*, *12* o T_{rojo} (variable de dominio finito asociada a cada posible tiempo de rojo para cada posible reparto).

- Para cada elemento de la lista Coste se evalúa cual es su resultado si el reparto elegido es el reparto actual y de estos valores se selecciona el que proporciona el valor mínimo. De esta forma se construye una nueva lista de valores constantes de los que se selecciona el máximo como coste del cruce al adoptar el criterio de mantener el reparto actual.

- Para cada elemento de la lista Coste se evalúa cual es su resultado si el reparto elegido es distinto del reparto actual y de estos valores se selecciona el que proporciona el valor mínimo. De esta forma se construye una nueva lista de valores constantes de los que se selecciona el máximo como coste del cruce al adoptar el criterio de cambiar de vector de reparto.

De los dos resultados el sistema implementa el que resulte menor, pero almacenando la otra opción por si la selección realizada no fuese la mejor al finalizar la simulación. Por lo tanto, en este método la evolución del tráfico en el pasado determina el vector de reparto a aplicar en cada instante. En el caso de seleccionar la opción de cambio de vector de reparto, el sistema debe determinar qué vector de reparto asignar al cruce de entre el resto de vectores de reparto válidos.

El sistema presenta tres alternativas de selección:

- Elección del vector de reparto más cercano al vector actual con la finalidad de minimizar los posibles efectos de su aplicación.

- Elección del vector de reparto más alejado al vector actual para dar más prioridad de uso a la dirección principal en el cruce.

- Elección del vector de reparto más alejado al vector actual para dar menor prioridad de uso a la dirección principal del cruce.

La alternativa a ejecutar se establece por medio del valor de un parámetro enviado a este agente desde el agente de interfaz de usuario.

Cuando el sistema alcanza la primera hoja del árbol se ha obtenido un primer plan de señales de tráfico y una medida asociada a su nivel de congestión para todo el intervalo de simulación.

Esta medida se utiliza como valor de referencia en el posterior proceso de poda. Si el sistema dispone de tiempo de cómputo entonces busca la última asignación de vector de reparto distinto del vector de reparto inicial realizada durante el proceso de generación de la primera hoja del árbol. En este nodo rechaza el vector de reparto seleccionado y con la misma alternativa de selección escoge otro vector de reparto distinto.

Esta nueva selección se evalúa de forma que en cada instante en el que algún objeto del sistema produce un evento indicador de un cambio cualitativo en su evolución, el sistema evalúa el nivel de congestión actual. El mecanismo de poda en el método corta

la rama actual del árbol si el producto del nivel de congestión actual por el tiempo de simulación restante no es inferior al nivel de congestión de referencia (primera hoja del árbol). Para realizar esta poda se asume que el nivel de congestión en ese instante se mantiene constante para el resto de tiempo de simulación. Esta suposición tiene el inconveniente de que puede hacer que se poden ramas que presentan asignaciones inicialmente malas pero que con el tiempo evolucionan a buenas regulaciones de tráfico.

En el caso de que se llegue a obtener otra hoja en el árbol, entonces su valor de congestión asociado pasa a ser el nuevo valor de referencia para el proceso de poda, y su plan de señales de tráfico pasa a ser el mejor plan calculado hasta ese momento.

Este proceso continua hasta que se agote el tiempo asignado a la ejecución del agente de solución. El mejor plan de señales de tráfico calculado en este proceso (asociado a la última hoja que se ha generado en el árbol) se envía al agente de refinamiento de soluciones para que intente mejorarlo en algunos de sus aspectos laterales y al agente de interfaz para que se lo comunique al usuario a efectos informativos.

En resumen, este método de ejecución es muy voraz respecto del tiempo, en el sentido que cuanto más tiempo disponga para su ejecución, mejor será el resultado calculado.

3.8.2 Esquema de Funcionamiento

La tarea del agente de solución es una tarea perteneciente a la tercera clase, por lo tanto, dispone de un tiempo máximo de ejecución que coincide con el fin del ciclo de ejecución de T_{ciclo} segundos. Esta situación se repite periódicamente para todos los ciclos del tiempo de integración.

Por otra parte, el método de ejecución de este agente es muy voraz respecto al tiempo de ejecución que puede llegar a consumir, por lo tanto, cuanto más tiempo se le asigne mejor será la solución calculada.

Por lo tanto, la primera forma de integración de este método en este modo de funcionamiento consiste en ejecutar este método para todo el intervalo de simulación correspondiente al periodo de integración de datos. De esta forma se consigue una visión global de la red urbana y se comprueban el máximo de posibles planes de señales ejecutables.

No obstante, esta situación presenta graves inconvenientes sobre cómo interrumpir el proceso de ramificación y poda cuando se ha alcanzado el máximo tiempo disponible para este ciclo, para reiniciarlo posteriormente en esa misma rama del árbol y con la misma información histórica almacenada cuando el agente de control le vuelva a conceder derecho de ejecución.

Esta tarea no resulta sencilla de implementar, puesto que habría que almacenar la situación global de cada uno de los objetos del sistema, las decisiones tomadas en cada nodo y su evolución pasada desde el inicio de este proceso. Todas estas acciones

precisan de un tiempo de cómputo que sería de más utilidad, para los objetivos del sistema de tráfico, emplearlo en el cálculo de mejores planes de señales.

Para evitar toda esta sobrecarga de trabajo se plantea un escenario distinto de trabajo, en el que este método se aplique a intervalos de T_{ciclo} segundos. En vez de ejecutar este agente en intervalos de $T_{integración}$ segundos la idea del método consiste en dividir esa duración en intervalos de T_{ciclo} segundos. De esta forma en cada ciclo el agente calcula cuál es la mejor asignación posible de vectores de reparto en cada cruce para cada ciclo. Este plan se almacena en el nivel actual del panel de la aplicación junto con los estados cualitativos finales de cada uno de los objetos del sistema.

Cada nueva ejecución del agente de solución recoge del nivel actual el último plan de señales calculado, que será el actual en ejecución cuando ésta sea la primera ejecución del agente de solución para el primer ciclo del nuevo periodo de integración, o bien será un plan ya calculado por ejecuciones anteriores del agente de solución.

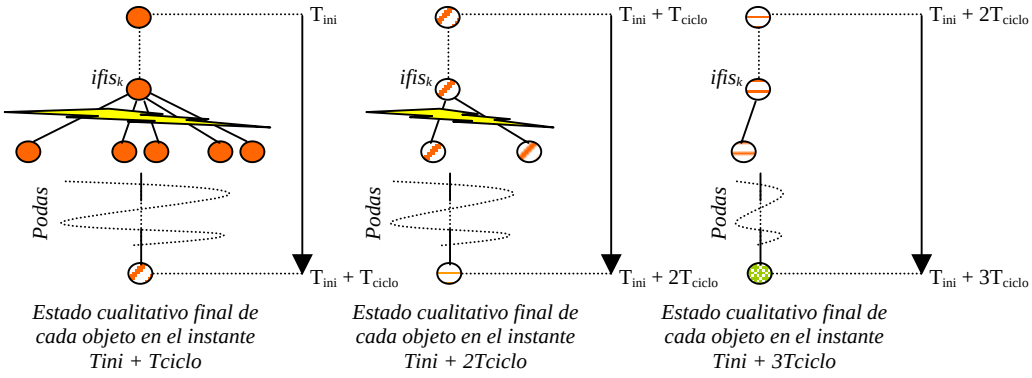


Fig 3-17 *Esquema de ejecución del método de solución dividida en intervalos de T_{ciclo} segundos. Durante el primer intervalo de ejecución, $[T_{ini}, T_{ini}+T_{ciclo}]$, el método realiza pocas podas, alcanzando un estado cualitativo final que sirve como nueva situación inicial para la siguiente ejecución. La última ejecución realiza pocas podas puesto que la mayoría de cruces han estabilizado el valor de su vector de reparto.*

Una vez seleccionado el plan de señales a aplicar, el agente de solución simula el nuevo periodo de T_{ciclo} segundos, realizando los cambios de vector de reparto en aquellos cruces que no estén *estabilizados*. Un cruce se considera *estabilizado* si en el proceso de generación de este árbol la heurística de poda que se aplicó por última vez se correspondía con la opción de reafirmar el valor del vector de reparto en ese instante, y por lo tanto, no lo quedan otros valores de vectores de reparto para evaluar.

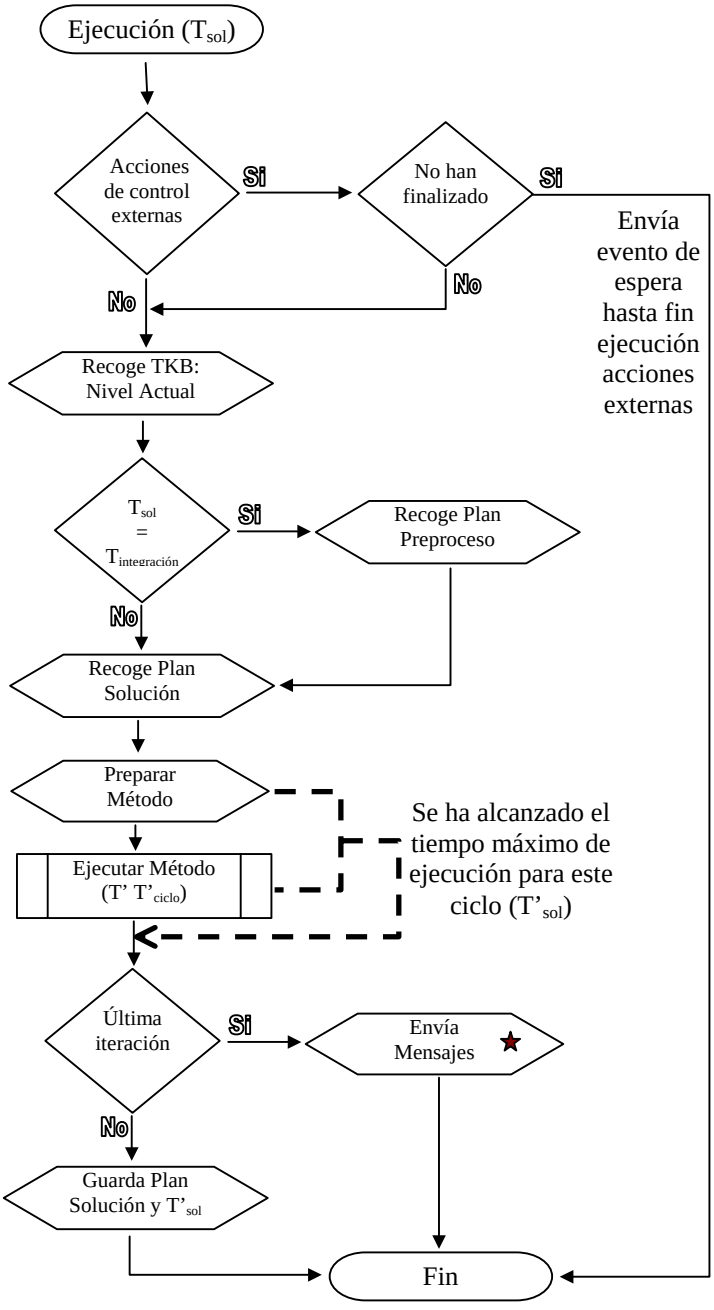


Fig 3-18 *Diagrama de flujo de la ejecución del agente de solución. El elemento marcado con una estrella envía mensajes al agente de refinamiento de soluciones y al agente de interfaz.*

Por otro lado, si la última heurística aplicada para un cruce fue la de selección de otro reparto distinto del reparto inicial entonces el proceso de simulación puede probar cual va a ser la evolución del sistema al aplicar los restantes valores de vectores de reparto válidos. Estos valores posibles, asociados a cada cruce, serán datos de entrada para la siguiente ejecución del agente de solución.

En la figura 3-17 se representa la evolución de la aplicación de este método para un periodo de integración múltiplo de tres veces el tiempo de ciclo. En el árbol de la izquierda se representa los vectores de reparto disponibles para el cruce $ifis_k$ al final de la primera ejecución. El árbol del centro de la figura representa las opciones restantes para ese mismo cruce al final de la segunda ejecución. Por último, la tercera ejecución (árbol de la derecha), al basarse en los resultados y asignaciones realizadas en las anteriores ejecuciones, realiza pocas podas puesto que gran parte de los cruces en el sistema ya tendrán su valor del vector de reparto estabilizado.

Todo este proceso está acotado por un tiempo máximo de ejecución que se corresponde con el cumplimiento del periodo de T_{ciclo} segundos. En el instante en que se cumpla este tiempo, el agente de solución deja de ejecutar el método y almacena en el nivel actual la información necesaria para su próxima ejecución.

En el caso que este sea el último ciclo de simulación antes de la integración de datos de los sensores del sistema, el agente comunica al agente de interfaz y al agente de refinamiento de las soluciones el plan de señales calculado.

Este método, aunque ejecutable dentro de la arquitectura propuesta, presenta algunos desventajas respecto al método inicialmente propuesto (tal y como se expone en el apartado 3.11 de este capítulo).

3.9 El Agente de Refinamiento de las Soluciones

La tarea del agente de refinamiento de las soluciones consiste en recoger la solución calculada por el agente de solución y proponer ligeros cambios en algunas de las asignaciones de vectores de reparto en algunos cruces con el fin de mejorar algunos de los aspectos de la solución.

Este agente se ejecuta en el último ciclo de T_{ciclo} segundos del periodo de integración puesto que precisa de la solución aportada por el agente de solución como dato inicial, y este dato sólo está disponible al final del periodo de integración.

3.9.1 Breve Descripción del Método de Ejecución

Este agente aplica conocimiento heurístico sobre las características de la red de tráfico urbana que se está controlando para mejorar la selección de vectores de reparto realizada por el agente de solución. Este conocimiento heurístico expresa el conocimiento del responsable de la sala de control de tráfico respecto a las características de cada objeto del sistema en cada instante de ejecución.

El método de ejecución consiste en la búsqueda de la regla heurística a aplicar para la solución inicial según el tiempo que resta hasta alcanzar el periodo de integración y según el momento del día en que esta solución se va a ejecutar. Para realizar esta tarea se establecen unos niveles de granularidad espacial respecto del refinamiento a utilizar relativos al periodo del día asociado a esa solución.

Se definen tres niveles de granularidad espacial en las acciones de refinamiento:

- *Cruces prioritarios* en la red de tráfico. La congestión de alguno de estos cruces provoca la rápida extensión de esta congestión a cruces adyacentes provocando un colapso circulatorio. Por lo tanto, conviene que estos cruces mantengan el tiempo de verde de su dirección principal lo más alto posible según la solución inicial.

- *Rutas prioritarias* en la red de tráfico. La extensión del concepto de cruce prioritario a una secuencia de cruces prioritarios en la dirección del tráfico. Si esta secuencia está formada únicamente por cruces prioritarios entonces se denomina *ruta prioritaria principal*, en el caso de que en la secuencia existan cruces prioritarios y no prioritarios la secuencia se denomina *ruta prioritaria secundaria*. Los cruces que forman estas rutas convienen que tengan su tiempo de verde para cada dirección principal lo más alto posible según la solución inicial. Para las rutas prioritarias principales este valor será mayor que para las rutas prioritarias secundarias.

- *Planes prioritarios*. Un plan prioritario es un plan modelo asociado a un conjunto determinado de situaciones de congestión más o menos habituales en rutas prioritarias. Cada cruce de cada ruta del plan se ha de intentar que tenga el máximo tiempo de verde posible respecto de la solución inicial.

La determinación de los objetos del sistema que se pueden declarar como *cruce prioritario*, *ruta prioritaria* y *plan prioritario* la realiza el responsable de la sala de control de tráfico según las características de la red urbana y de la franja horaria de su aplicación. Estas declaraciones son estáticas para el agente de refinamiento en el sentido que una vez declaradas no son modificables externamente.

La selección de que tipo de refinamiento ejecutar depende del tiempo que disponga el agente de refinamiento para su ejecución (que será el tiempo que quede hasta alcanzar el final del actual periodo de integración).

Si el agente de refinamiento recibe el derecho de ejecución cuando resta muy poco tiempo para alcanzar el final del periodo de integración entonces selecciona el método de refinamiento por cruces prioritarios. En el caso de que ese tiempo le permita refinar por rutas prioritarias entonces intenta aplicar este método, pero si según se ejecuta el método estima que no le queda el tiempo suficiente para finalizar este proceso, entonces cambia a utilizar el método de refinamiento por cruces prioritarios. En esta situación el agente utiliza la información de refinamiento de rutas que hubiese calculado hasta ese momento.

Por último, si se dispone de tiempo suficiente el agente de refinamiento ejecuta el método de refinamiento por planes. Si según se va ejecutando este proceso se estima que no podrá finalizarlo, entonces cambia a ejecutar el método de refinamiento por rutas, y si también estima que no dispone de tiempo suficiente para finalizarlo entonces cambia a ejecutar el método de refinamiento por cruces.

Las estimaciones iniciales respecto de la duración de cada uno de los procesos de refinamiento se definen de forma experimental de acuerdo a la experiencia acumulada. Cada uno de los planes y de las rutas prioritarias tiene asociado un tiempo de cómputo proporcional al número de cruces que contiene. Las estimaciones parciales internas en cada proceso de refinamiento se realizan de forma ponderada al número de cruces que quedan por analizar.

En resumen, este método presenta un comportamiento afín a las denominadas técnicas de *razonamiento aproximado* [Lesser, 88] en los que se dispone de múltiples métodos para solucionar un problema y es cuestión del tiempo disponible tanto la selección del método más adecuado [Decker, 94] como la secuencia de ejecuciones a realizar. La decisión de que método utilizar en cada instante recibe el nombre de *planificación sujeta al tiempo* [Garvey, 93] y se asume que cuando se decide cambiar de método porque el tiempo disponible no es suficiente para finalizar la actual ejecución, el siguiente método a ejecutar utiliza como datos iniciales los resultados intermedios de la ejecución del método anterior.

3.9.2 Esquema de Funcionamiento

La tarea del agente de refinamiento de soluciones es una tarea de la cuarta clase. Por lo tanto, su ejecución está determinada por la disponibilidad de los resultados de la ejecución del agente de solución (tercera clase).

Además este agente sólo se ejecuta en el último ciclo de ejecución próximo a la finalización del actual periodo de integración, por lo tanto, debe de dedicar el tiempo que se le asigne a realizar la modificación de la solución aportada por el agente de solución de la forma más eficiente posible.

El agente realiza una primera tarea de estimación del tiempo que le queda disponible, y en función de ese tiempo y de otra serie de características como son la hora y día en el que esta solución se va a aplicar deduce cual es el mejor método a aplicar. Cada uno de los métodos tiene una unidad de refinamiento distinta.

En el caso del refinamiento por planes esta unidad es cada una de las rutas (prioritarias o secundarias) que contiene. En el caso del refinamiento por rutas la unidad de refinamiento es por cruces contenidos en la ruta. El método de refinamiento por cruces se ejecuta en una sola unidad.

Al finalizar la ejecución de cada unidad de refinamiento, el agente vuelve a predecir el tiempo que le queda para acabar de ejecutar este método de refinamiento respecto del tiempo que le queda de ejecución real. Si esta diferencia es negativa entonces el agente cambia de método de refinamiento a uno más rápido pero también con menos mejoras globales asociadas a toda la red urbana bajo control.

En el momento en que se alcance el final del tiempo disponible para la ejecución del agente de refinamiento, éste devuelve la solución refinada que haya calculado hasta ese momento.

La solución refinada se envía al resto de agentes en el caso que el sistema funcione en el modo automático de asignación de vectores de repartos a cruces. En caso contrario este plan de tráfico se envía únicamente al agente de interfaz para que lo muestre al usuario y sea éste quien decida si lo ejecuta o no, en su totalidad o de forma parcial, mediante la ejecución de las acciones externas de control adecuadas.

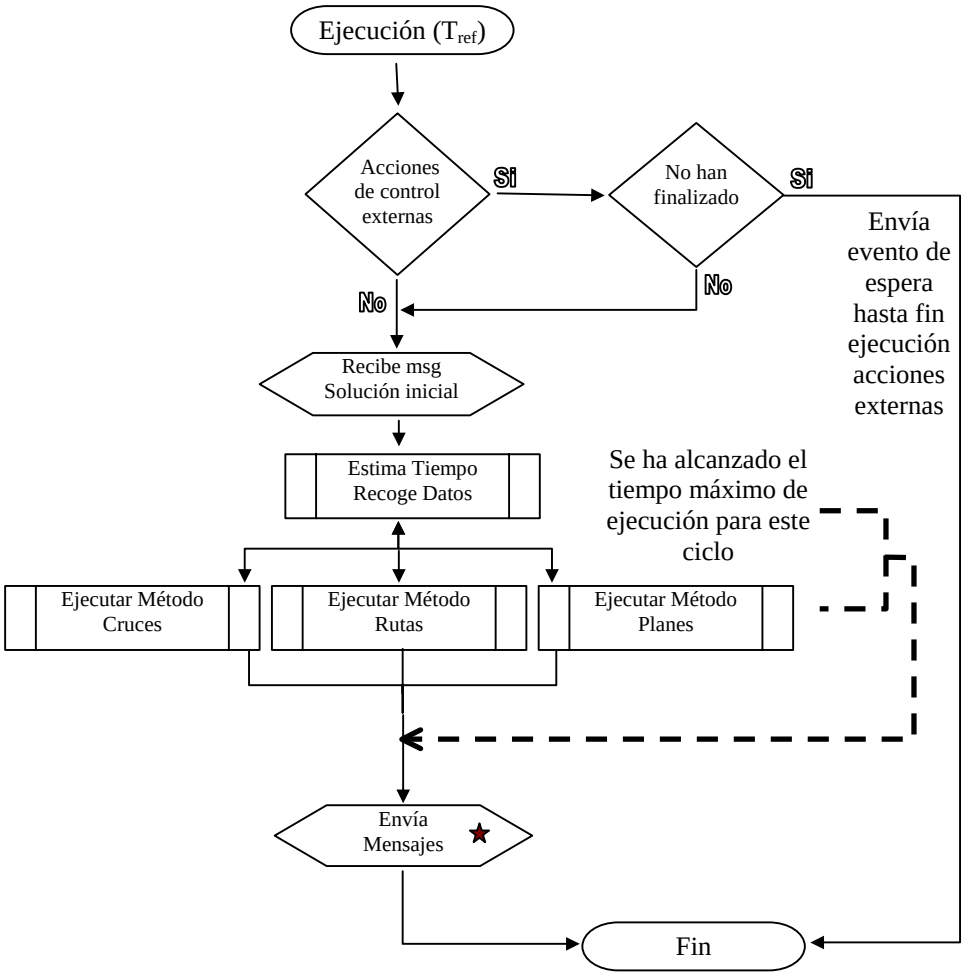


Fig 3-19 *Diagrama de flujo de la ejecución del agente de refinamiento de soluciones. El elemento marcado con una estrella envía mensajes a todos los agentes del sistema si el modo de ejecución actual es automático respecto a la asignación de vectores de reparto a cruces. En caso contrario, este agente sólo envía el mensaje al agente de interfaz.*

3.10 El Agente de Interfaz

El agente de interfaz es el encargado de presentar la evolución del sistema y de interaccionar con el usuario para procesar sus peticiones de información o de ejecución de acciones externas.

De forma general el agente de interfaz debe cumplir el número máximo de los siguientes objetivos [García, 99]:

- Visualización de la evolución cualitativa del sistema acorde al tiempo real en la red de tráfico urbano. Esta visualización incluye también la predicción de estado futuro en el supuesto de que no se presenten incidentes en el sistema. Esta visualización debe de cumplir dos requisitos: ser sencilla de interpretar y poseer un alto grado de información representativa.

- Resolver las peticiones de una información más extensa, que la especificada en el punto anterior, acerca del estado cualitativo de cada objeto del sistema realizadas por el usuario.

- Atender a las peticiones de ejecución de cualquiera de las acciones externas que se puedan ejecutar proporcionando al usuario del sistema el máximo de información posible para que pueda ajustar los valores cuantitativos de esas acciones de una forma sencilla.

- Permitir la alteración del modo de funcionamiento global del sistema (repartos automáticos/supervisados) o de las opciones configurables de forma individual para cada agente de la aplicación en el sistema.

- Destacar las situaciones “anormales” que sucedan en el sistema mediante la comunicación de un mensaje de alerta ha ser visualizado de forma inmediata (como sucede, por ejemplo, cuando se ha realizado una petición de acción de control externa sobre un objeto del sistema y el agente de preproceso no la ejecuta porque existe una acción externa todavía en ejecución sobre ese mismo objeto).

Un grave problema asociado a la consecución de estos objetivos es que el conseguirlos implica la dedicación de grandes recursos de cómputo (tiempo de procesador), por lo que se está consumiendo un tiempo muy valioso que podría ser muy útil para la ejecución de otros agentes cuyos resultados son fundamentales para el sistema. Por otra parte una interfaz con menos opciones que las señaladas se convierte en una herramienta con la que resultará difícil que el usuario del sistema trabaje de forma cómoda.

La búsqueda de un equilibrio entre estas dos formas de funcionamiento para el sistema global no resulta sencillo, y de hecho, será otra de las razones que provoquen el cambio de arquitectura de sistema como se verá en el siguiente apartado.

Después de valorar los pros y contras de ambas posturas se ha optado por incorporar el total de los objetivos anteriores en el desarrollo del agente de interfaz. Estos objetivos se han cumplido de la forma siguiente:

- Utilización de una ventana para representar la evolución del presente y otra para el futuro predecible. En cada ventana se representa la red urbana formada

por rectángulos, divididos en secciones, y por círculos. Cada rectángulo representa un segmento de tráfico y cada una de sus secciones se rellena con un color identificativo de su valor cualitativo. Cada círculo representa un cruce y en su interior se representa un número que identifica cual es la fase en ejecución en cada momento. Cada uno estos elementos (rectángulos y círculos) son objetos sensibles al paso del puntero del ratón por su superficie. En ese instante se activan y al presionar una tecla vuelcan en una nueva ventana toda la información disponible para ese objeto. En el caso de un segmento se representa su longitud, número de carriles, estado cualitativo en cada sección en ese momento, su nombre, código y elementos que definen su extremo inicial y final. Para el caso de un cruce esta información incluye un gráfico de la fase actual en ejecución, su código, segmentos de entrada, salida, vector de reparto en ejecución, próximo reparto previsto (en el caso que se hubiese lanzado una petición de ejecución de una acción externa sobre ese objeto) y valor actual de coordinación.

- Una ventana para mostrar los resultados de la ejecución del agente de detección de una forma ordenada y compacta, indicando los problemas elementales, temporales y espacio-temporales detectados hasta ese momento en el instante presente y también en la predicción de futuro.

- Una ventana para visualizar los resultados del agente de solución y de refinamiento de soluciones.

- Un menú de selección de acciones externas de control que se pueden ejecutar sobre cruces (modificación del vector de reparto a aplicar o del valor de coordinación respecto al inicio del ciclo del CCI al que pertenece) y sobre CCIS (modificación discreta de la duración del ciclo).

- Una ventana de seguimiento del cumplimiento de cada ejecución de acción de control externa ejecutada en el sistema. Si la acción es sobre un cruce se muestra un mensaje a la finalización de su ejecución. Si la acción es sobre un CCI se muestra una ventana en la que se representa de forma dinámica los cruces del CCI que van alcanzando el nuevo valor final de duración del ciclo.

- Una ventana de visualización de mensajes de alerta del sistema. Estos mensajes pueden producirse por varios motivos: la recepción de la identificación de un incidente en el sistema, el comienzo, desarrollo o el fin de la ejecución de alguna acción de control externa y el rechazo de alguna petición de acción de control externa.

- Un menú de configuración de los modos de funcionamiento del sistema y de las opciones ajustables de cada uno de los agentes de la aplicación.

El agente de interfaz está implementado como un proceso aparte del resto de agentes del sistema de forma que el tiempo disponible por el procesador se divide entre este proceso y el proceso de ejecución del resto de agentes del sistema.

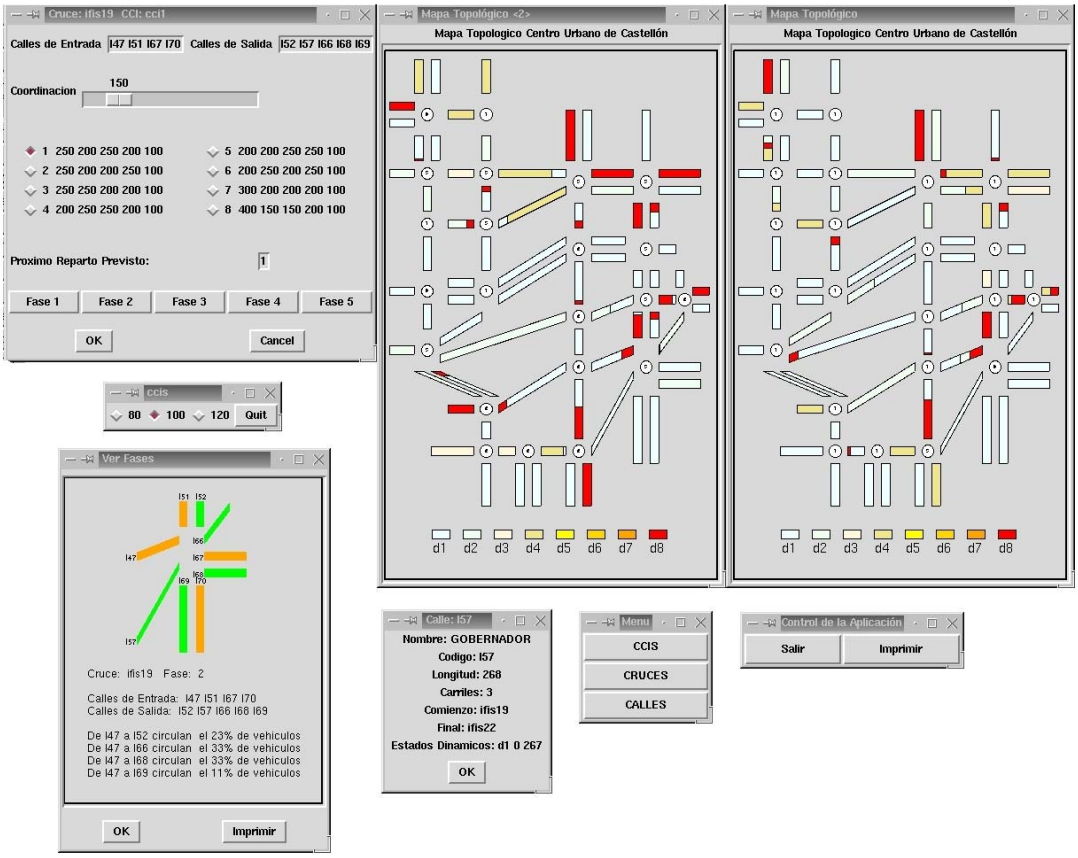


Fig 3-20 *Algunas de las ventanas utilizadas por el agente de interfaz. Las dos ventanas superiores de la derecha representan la evolución cualitativa actual y de futuro (predicción). La ventana superior izquierda representa las acciones de control externas que se pueden ejecutar sobre el cruce ifis19. La ventana que está justo debajo permite modificar de forma discreta la duración del cci del sistema. Las ventanas inferiores de la izquierda muestran la información cualitativa actual del cruce ifis19 y del segmento l57, mientras que las dos ventanas inferiores de la derecha representan dos de los menús de configuración en el sistema.*

Al ser un proceso independiente, este agente no tiene acceso al contenido dinámico de la pizarra por lo que cada cambio cualitativo o el resultado de cualquier acción externa ejecutada en el sistema debe de ser comunicado al agente de interfaz para que exista un alto grado de coherencia respecto de las informaciones almacenadas en ambos procesos.

La comunicación entre ambos procesos se realiza mediante el envío de mensajes de actualización de la información cuando esta se produce. El agente de interfaz envía al agente de preproceso las peticiones de ejecución de acciones externas propuestas por el usuario. El agente de preproceso envía al agente de interfaz cada cambio cualitativo que simula en el sistema así como el desarrollo de cualquier acción de control externa que se

esté ejecutando. Por lo tanto, como el agente de preproceso se ejecuta mucho más rápido que la evolución real del tráfico, el agente de interfaz debe de retener los datos cualitativos que recibe de este agente hasta que se cumpla el instante real en que estos cambios suceden en la red urbana. En ese instante es cuando los visualiza en la ventana asociada al estado actual del tráfico.

El resto de agentes de la aplicación en el sistema tienen una comunicación menos fluida con el agente de interfaz que la del agente de preproceso. El agente de predicción envía al agente de interfaz el estado cualitativo de cada objeto del sistema al final de cada uno de sus periodos de predicción. El agente de detección comunica al agente de interfaz cada uno de las situaciones problemáticas que detecta y el agente de diagnosis la información referente a cada incidente detectado. Por último, los agentes de solución y de refinamiento de soluciones envían sus resultados al agente de interfaz para que los muestre al usuario.

La información dinámica que precisa el agente de interfaz la obtiene, por lo tanto, de los mensajes que recibe. La información estática la recibe de los ficheros topológicos de la red suministrados a este agente en el momento de arranque de la aplicación.

3.11 Experiencia Aprendida en el Desarrollo del Sistema Secuencial

El sistema de control de tráfico expuesto en los apartados anteriores se empezó a implementar, en su versión secuencial, utilizando como dominio de aplicación el centro urbano de Castellón de la Plana.

La primera tarea para realizar esta implementación consistió en identificar, seleccionar y recoger datos de cada objeto incluido en la red urbana bajo estudio. Esta información se obtuvo de las conclusiones del proyecto "*Programa Director para los estudios de tráfico y transporte en Castellón*" financiado por el Excmo. Ayuntamiento de Castellón de la Plana y desarrollado por la Universidad Politécnica de Cataluña y la Universidad Jaume I de Castellón durante el año 1997.

Algunos de estos datos, los referentes a características topológicas de los objetos, fueron proporcionados de antemano, mientras que los datos indicadores de los porcentajes de giro en cada cruce fueron medidos dentro del proyecto.

El dominio de aplicación del sistema incluye 72 segmentos dirigidos, 22 cruces, 11 entradas, 17 salidas y 1 cci. Estos elementos forman el subconjunto más relevante del centro urbano en estudio. Cada uno de estos elementos fue seleccionado siguiendo los criterios establecidos en el citado proyecto. Los tipos de datos proporcionados y medidos para cada uno de esos objetos se muestran en la tabla 3.4.

Para uniformizar al máximo cualquier intersección entre dos o más segmentos de la red urbana se ha impuesto que estas situaciones siempre se asocien a un cruce regulado por semáforos en el prototipo. Si esta intersección en la red no se corresponde con un cruce regulado por semáforos entonces se asume que este cruce en el prototipo siempre

tiene los semáforos en verde para todas las direcciones posibles, y que es el comportamiento de los conductores el que regula su funcionamiento.

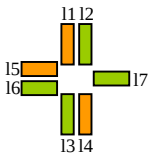
Tipo	Datos Estáticos	Datos Medidos
sensor	Código Id. segm. que lo contiene Distancia a la línea de parada Tipo sensor (sensor de lazo) Activo / Inactivo Última fecha envío de datos	
segmento	Código Nombre Longitud Número de carriles Elemento conectado al extremo inicial Elemento conectado al extremo final	
entrada	Código Id segm. al que está conectado	
salida	Código Id segm. al que está conectado	
cruce	Código Código CCI al que pertenece Códigos segm. de entrada Códigos segm. de salida Vectores de reparto Fases	% vehículos que circulan de una entrada a una salida  **Código:** *ifis2* **Seg. entrada**: *l1*,*l4*,*l5* **Seg. salida:** *l2*,*l3*,*l6*,*l7* **Fase** *1* **Desde** *l1* **hasta** *l6*: *30%* **Desde** *l1* **hasta** *l3*: *70%* **Desde** *l4* **hasta** *l2*: *80%* **Desde** *l4* **hasta** *l7*: *20%*
cci	Código Conjunto de cruces Valores posibles	

Tabla 3-4 *Tabla resumen de los datos utilizados por cada uno de los objetos pertenecientes al dominio seleccionado para su experimentación.*

Uno de los problemas que presentaba el conjunto de datos disponibles para la zona urbana en estudio era la ausencia de datos procedentes de los sensores. Esta falta de datos se tradujo en la necesidad de construir unos escenarios iniciales de tráfico con los

que alimentar al prototipo, y el suponer en las pruebas de laboratorio realizadas que no se producían incidentes durante la evolución del sistema.

Estos escenarios de tráfico permitieron experimentar con el sistema variando algunas de las siguientes características:

- Valores cualitativos iniciales para cada uno de los objetos del dominio.
- Valores de coordinación para los cruces, lo que permite experimentar con la evolución del tráfico en el prototipo al poder establecer distintas ondas verdes.
- Definición de uno o dos ccis y de sus cruces asociados.

Los primeros agentes desarrollados fueron los agentes de control, preproceso, predicción e interfaz. A cada uno de estos agentes se les proporcionó un método para medir el tiempo de ejecución empleado por cada uno de ellos durante cada ciclo básico de ejecución del ciclo de control en el sistema.

El análisis de estos resultados indicaron dos características básicas respecto del comportamiento del sistema:

- El agente de interfaz consume gran parte del tiempo de ejecución del sistema, y, además, no responde con la suficiente suavidad a las diferentes peticiones de información o de ejecución de acciones externas realizadas por el usuario.
- El intercambio de datos entre los agentes de preproceso y de predicción permite acelerar la ejecución del agente de preproceso en los ciclos siguientes al inicial en el caso que no se produzcan acciones externas de control.

La primera característica, como ya se describió en el apartado anterior, resulta difícil de evitar si se busca una relación de equilibrio entre el objetivo del dominio de la aplicación y la facilidad de interacción con el usuario.

La segunda característica orienta hacia la búsqueda de un estudio sobre cual seria la secuencia óptima de ejecuciones en el sistema para poder ejecutar todas las peticiones pendientes durante un intervalo de integración en el supuesto de ausencia tanto de incidentes en la red como de ejecuciones de acciones externas de control. Como resultado adicional de este estudio también se analiza cual debe de ser el tamaño de la ventana temporal de futuro, es decir, el máximo tiempo de predicción de futuro establecido en el sistema para que funcione de la forma prevista.

Esta secuencia depende en gran medida del estado de la red y de su tamaño (su número asociado de objetos dinámicos) puesto que estos parámetros determinan el tiempo de ejecución de cada agente.

Una óptima secuencia de ejecuciones de agentes seria aquella que dedica el máximo tiempo de ejecución posible a las tareas más costosas. En la figura 3-21 se muestra una

de tales secuencias asociada a un periodo de integración igual a tres veces el tiempo de ciclo del sistema. El primer ciclo se dedica a generar el presente y a predecir y evaluar el futuro. El agente de preproceso no tiene que simular el presente durante el segundo ciclo puesto que esta tarea ya la ha realizado el agente de predicción en el ciclo anterior. Por lo tanto, el agente de preproceso y el de detección recogen la información del nivel futuro del TKB y lo trasladan al nivel actual. La mayor parte del tiempo de este ciclo se dedica a la ejecución del agente de Solución. La ejecución del tercer ciclo sigue las mismas pautas que la del segundo excepto que la ejecución del agente de solución finaliza ante de alcanzar el $T_{integración}$ para poder ejecutar, en los instantes finales, el agente de refinamiento con la solución calculada hasta ese momento.

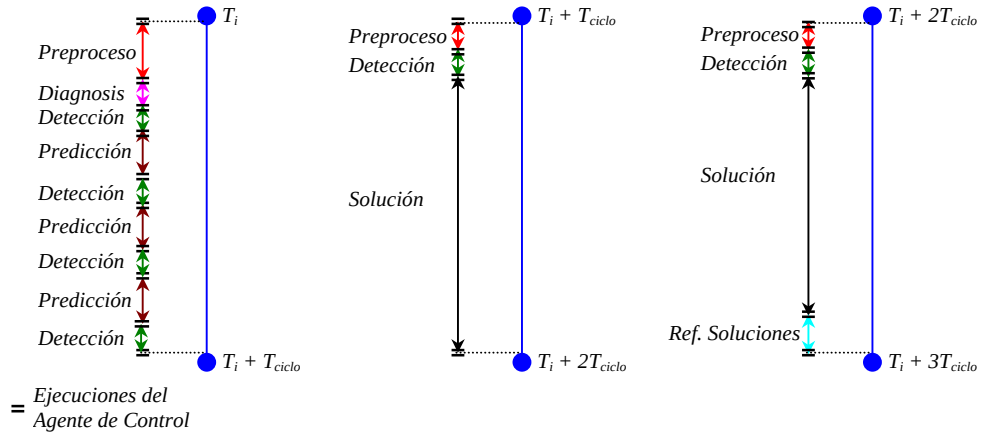


Fig 3-21 *Estimación de cual debe de ser la secuencia óptima de ejecuciones para un periodo de integración igual a tres veces el tiempo de ciclo del sistema.*

Como resultado adicional, y dado que el agente de predicción se ejecuta un total de tres veces, se considera que en esta secuencia el tamaño de la ventana temporal varía entre $3T_{ciclo}$ (en el primer ciclo de ejecución) y T_{ciclo} (en el tercer ciclo de ejecución) segundos.

Esta secuencia se considera óptima en el sentido que permite que el agente de solución disponga de un intervalo de tiempo para poder alcanzar un resultado aceptable. No obstante, también presenta alguna consecuencia no deseable. Una de ellas está relacionada a cómo se ejecuta el agente de predicción. Este agente recoge datos del nivel de futuro, calcula su evolución para el siguiente intervalo y los deposita de nuevo en el nivel de futuro. Como este agente se ejecuta varias veces en un ciclo, entonces está empleando un tiempo no despreciable en dejar datos en el TKB que vuelve a recoger casi de forma instantánea. Este problema también ocurre con el agente de detección aunque de forma más suave. Una manera de evitar esta situación consistiría en modificar el comportamiento del agente de control para que agrupe todas estas ejecuciones individuales en una sola (mediante un empleo más extenso por parte del agente de control de la relación *combina*).

Estas situaciones no son extrañas en el sistema puesto que en el se integran tareas con costes muy distintos, y todas se fuerzan a ser ejecutadas por intervalos temporales. Para algunos agentes, la situación normal es que tengan tiempo suficiente para ejecutarse de una sola vez, como es el caso de los agentes de preproceso, detección o diagnosis. Otros agentes ven interrumpida su ejecución múltiples veces, degradando de esta forma la calidad de sus resultados, como es el caso del agente de solución o del agente de predicción. En este último caso el agente de predicción se deduce que degrada sus resultados por el tiempo que emplea en dejar datos en el TKB para volver a seleccionarlos casi de forma instantánea, sumado al tiempo empleado por el agente de control en decidir volver a asignarle el derecho de ejecución.

Además, algunos agentes emplean una parte no despreciable de su tiempo de ejecución en tareas de inicialización. Esta característica empeora el rendimiento del sistema cuando este agente interrumpe su ejecución, para volver a ejecutarla más tarde, en el mismo ciclo o en el siguiente.

Esta situación es especialmente delicada en el caso de las ejecuciones del agente de solución. El método inicial desarrollado para este agente consistía en realizar un proceso de ramificación y poda en el proceso de simulación durante el intervalo de carencia de datos de sensores en el sistema, $t_{integración}$ segundos. Cada vez que un cruce cambia a su primera fase se realiza un proceso de selección del reparto más adecuado en ese instante. Cuando se alcanza la primera solución el sistema realiza un proceso de *retroceso* (backtracking) para revisar el resto de opciones en cada cruce, podando aquellas que, previsiblemente, no conduzcan a mejores soluciones que la mejor calculada hasta el momento.

Este método extiende su intervalo temporal de estudio a todo el periodo de integración, buscando soluciones que mejoren el estado global de la red de tráfico. No obstante, al dividir este proceso en bloques de T_{ciclo} segundos, y al disponer de una cantidad de tiempo de ejecución variable para cada uno de estos bloques, resulta difícil de garantizar que se encuentre una solución que reduzca considerablemente la cantidad de situaciones problemáticas detectadas. Esto es así, porque en la ejecución de cada bloque se parte de la situación final alcanzada en el bloque anterior, y los instantes de selección de repartos de ese bloque anterior resultan inalcanzables desde este nuevo bloque de ejecución.

Esta situación tiene su origen en que el método del que se parte está diseñado para ser aplicado de una sola vez a todo el periodo de integración en el sistema. La solución adaptada para ser aplicada en esta arquitectura rompe con este diseño, y por eso, pierde eficiencia en los resultados que obtiene.

Todo este análisis se ha realizado sin tener en cuenta las consecuencias de la ejecución de acciones externas de control en el sistema. Estas acciones no son frecuentes, pero el sistema es muy sensible al momento en que se ejecutan. No es lo mismo ejecutarlas durante el primer ciclo del periodo de integración, en cuyo caso el siguiente ciclo se podrá ejecutar sin interrupciones (aunque tardará un poco más en disponer de los datos del presente), que ejecutarlas durante el segundo o posteriores

ciclos, puesto que en estas situaciones se invalidan los datos de simulación actual y todos los procesos que habían obtenido resultados a partir de estos datos como datos iniciales. Además, queda menos tiempo para alcanzar el fin del periodo de integración y, por lo tanto, para que estos métodos alcancen unos resultados de una calidad aceptable.

De este análisis se deduce que las tareas que han de realizar cada uno de los agentes que se integren en este sistema deben de tener una característica básica: deben de ser métodos con un coste computacional bajo o bien métodos que permita ser ejecutados por subintervalos del periodo de integración sin que se reduzca la calidad de las soluciones aportadas. Por lo tanto, esta arquitectura no está diseñada para integrar cualquier método de ejecución.

Esta falta de generalidad se podría subsanar al considerar que esta arquitectura se diseñó con el objetivo final de implementarla en un sistema multiprocesador con memoria compartida. En esta situación, se podría asignar un proceso independiente a cada una de las tareas que no se adapten fácilmente a este requisito, e integrarlas en el sistema por medio de un mecanismo de envío de mensajes. De esta forma, en la arquitectura sólo se integrarían agentes cuyas tareas fuesen adecuadas para su particionado en intervalos de ejecución de T_{ciclo} segundos.

Otra característica importante que destacar de la arquitectura propuesta es que se trata de un sistema de control fuertemente centralizado, es decir, el agente de control gobierna la ejecución del resto de agentes del sistema. Este comportamiento contrasta con la tendencia actual en Inteligencia Artificial que consiste en el desarrollo de arquitecturas distribuidas con un pequeño o nulo proceso de control centralizado.

El objetivo inicial de esta tesis era el estudio y evaluación de arquitecturas multiprocesador con memoria compartida aplicadas al control del tráfico urbano. Este objetivo general se desarrollaría mediante el uso de programación lógica basada en restricciones como herramienta básica con la que conseguir construir de forma rápida un prototipo.

Por lo tanto, la programación del sistema requeriría el uso de un sistema de programación lógica paralela basada en restricciones. Parecía, en el momento de realizar la propuesta de tesis, que estas herramientas iban a estar rápidamente disponibles como productos comerciales, o al menos, como productos de uso continuo de experimentación. La evolución de los hechos fue otra muy distinta.

La primera de las herramientas utilizadas consistió en el uso del sistema[1] ECL^{i}PS^e en su versión paralela, *peclipse*. Esta herramienta sólo estaba disponible para un modelo concreto de computador. Este sistema fue desechado puesto que no se disponía de ese modelo de computador.

[1] ECL^{i}PS^e CLP System. IC-Parc, Imperial College, London. http://www.icparc.ic.ac.uk/eclipse/

La segunda de las herramientas utilizadas consistió en el uso del sistema MUSE (componente del sistema SICSTUS Prolog[2]) y de otras herramientas de ayuda para su programación y análisis (como es el caso de la herramienta VISANDOR[3]). Aunque estos sistemas eran ejecutables en bastantes más modelos diferentes de computador que el sistema peclipse, (entre otros los multiprocesadores de la familia Sun O.S. 5.1) también presentaban un grave inconveniente: no eran, de forma individual, productos comerciales, y por lo tanto, no tenían mantenimiento oficial. A partir de la versión 6.0 de Sicstus Prolog la herramienta MUSE dejó de ser soportada, y por lo tanto, dejó de ser viable para nuestro interés.

Todos estos sistemas compartían un mismo modo de ejecutar el paralelismo, el paralelismo OR. En este tipo de funcionamiento, el propio sistema decidía qué ramas del árbol de ejecución del prolog ejecutar en paralelo de forma independiente y cuándo hacerlo. Este modo de funcionamiento no era el modo de trabajo que interesaba para el desarrollo de este sistema de control de tráfico, puesto que el único proceso que podría interesar que fuese más rápido era la tarea del agente de solución, y esta tarea, no era sencilla de modelar con programación paralela OR. No obstante, estos sistemas también permitían especificar que tareas se deseaban ejecutar en paralelo mediante el uso de etiquetas identificativas en ciertas secciones de los programas, pero este modo de funcionamiento no era el modo "natural" de ejecutar el paralelismo OR.

Un problema añadido al de la disponibilidad de software de programación adecuado para tales sistemas es que podía haber más agentes en el sistema que procesadores físicos. Esta situación podría llegar a resultar en una degradación de las prestaciones del sistema si las tareas a desarrollar por estos agentes tenían un coste elevado o se introducían otras tareas a realizar por nuevos agentes.

Además, la tendencia inicial de diseño, desarrollo y comercialización de máquinas multiprocesador con memoria compartida ha ido poco a poco reduciéndose debido fundamentalmente a que son sistemas difíciles de ampliar, tanto en memoria como en procesadores; es decir, al aumentar el tamaño del dominio de la aplicación el sistema puede quedar pequeño, y su ampliación resulta cara.

La tendencia actual consiste en utilizar sistemas distribuidos construidos mediante clusters de PCs conectados mediante una red de interconexión de altas prestaciones. Estos sistemas implementan el modelo de ejecución de sistemas distribuidos con memoria compartida virtual mediante mecanismos de paso de mensajes tipo PVM o MPI. Mediante este mecanismo se consigue un funcionamiento parecido al de las máquinas multiprocesadores con memoria compartida consiguiendo unas buenas prestaciones a un bajo coste.

Toda esta evolución ha promovido el desarrollo de una nueva arquitectura para la implementación de sistemas de control de tráfico urbano en la que se han unido tanto las tendencias actuales en el desarrollo de sistemas multiprocesadores como la búsqueda de

[2] SICSTUS Prolog, http://www.sics.se/sicstus.html
[3] A visualizer of Parallel Executions. http://www.clip.dia.fi.upm.es/Software/index.html

una mejor afinación de las tareas de cada agente, y de cómo estas se ejecutan, en el sistema.

Este sistema hereda gran parte de las experiencias y métodos alcanzados con el desarrollo del prototipo MASHGREEN SM, y propone modificaciones sobre este modelo con las que evitar, o reducir, los defectos señalados.

CAPÍTULO 4
EL PROTOTIPO MASHGREEN DM 97-99

Índice del capítulo 4

4.1 Características del Sistema MASHGREEN DM

El prototipo ***MASHGREEN DM*** (a ***M***ulti***A***gent ***S***ystem for ***H***elpin***G*** u***R***ban traffic d***E***cision tak***EN*** with ***D***istributed ***M***emory) surge como una evolución para el control de tráfico urbano que intenta reducir los defectos identificados en el prototipo MASHGREEN SM .

Las principales limitaciones identificados se pueden resumir en los siguientes ítems:

- Exigencia de que todas las tareas de los agentes se ajusten a tiempos de ejecución menores o iguales a t_{ciclo} segundos.

- Múltiples procesos de iniciación y de recogida de datos para un mismo agente dentro del mismo ciclo de ejecución.

- Dificultad de integrar tareas cuyos costes computacionales son muy diferentes entre sí.

- Orientación hacia un sistema multiprocesador con memoria compartida y las dificultades en el empleo de herramientas de programación lógica basada en restricciones para este tipo de sistemas.

Todos estos defectos han orientado hacia la definición de una nueva arquitectura funcional que integre algunas de las características del prototipo anterior y que trate de minimizar algunos de sus defectos.

Las siguientes propuestas reducen, y en algunos casos, eliminan las deficiencias anteriores. La primera deficiencia identificada se puede eliminar si se permite la coexistencia de tareas que tengan distintos tiempos de ejecución. No obstante, todas las ejecuciones deben de acomodarse a un tiempo máximo de ejecución igual al periodo de integración de los datos de los sensores en el sistema. El agente de coordinación es el encargado de coordinar los tiempos de ejecución para todos los agentes en el sistema. La segunda deficiencia resulta también eliminada como efecto lateral de la propuesta anterior, puesto que un agente no finaliza su ejecución hasta que no acaba su tarea por lo que, en ausencia de acciones de control externas, sólo tiene que realizar un proceso de inicialización por ciclo y no interrumpe su ejecución hasta que ésta finaliza.

Para solventar las dos últimas deficiencias se propone utilizar una arquitectura de computadores que sea distribuida y heterogénea. Esta arquitectura se implementa como un cluster de PCs conectados mediante una red de interconexión de altas prestaciones. Al disponer de varios computadores PCs se puede dedicar un computador a que realice las tareas de un único agente e incluso, al tratarse de una red heterogénea, también permite dedicar los computadores más potentes a los agentes cuyas tareas presenten un mayor coste computacional. Por último, el empleo de mecanismos de sincronización linda [Carriero, 90] y su habitual integración en las herramientas de programación

lógica basada en restricciones (como es el caso de Sicstus[1] prolog o BinProlog[2]) permite implementar mecanismos de coordinación y de transferencia de datos entre los diferentes agentes del sistema, y por lo tanto, entre los distintos computadores de la arquitectura.

Esta nueva arquitectura rompe con el concepto de arquitectura paralela cerrada, en la que tanto el número de agentes como el de procesadores disponibles del sistema eran parámetros fijos que difícilmente se podían modificar, para plantear una arquitectura distribuida abierta en la que se pueden modificar estos parámetros de una forma más sencilla.

Para realizar todos estos cambios se tienen que modificar las aptitudes de los agentes del sistema, otorgándoles una mayor independencia respecto de la evolución del sistema. Como consecuencias de esta nueva definición surgen dos cuestiones respecto al modelo de ejecución en el nuevo sistema: ¿cómo se organiza el acceso a los datos del TKB? y ¿cómo integrar tareas con distintos ciclos de ejecución?

4.1.1 Organización de los datos

Los datos de la aplicación almacenados en el TKB poseen dos características básicas: la descripción estática (que no varía durante la ejecución del sistema) y la descripción dinámica (sujeta a cambios según avanza el tiempo).

Las alternativas posibles para organizar los datos en el nuevo sistema son dos: 1) disponer todos los datos de la aplicación en un mismo almacén (el TKB), es decir, mantener la organización del sistema anterior; y 2) distribuir los datos de la aplicación entre los distintos agentes de forma que cada uno de ellos se haga responsable de la validez y corrección de los datos que almacena.

La primera opción tiene la ventaja de que, al estar todos los datos posibles en ese almacén, los agentes pueden acceder a los datos que le son necesarios en cualquier instante. No obstante esta ventaja es también parte de su inconveniente, puesto que al ser fácilmente accesibles esos datos por cualquiera de los agentes se necesita un mecanismo que asegure su integridad. Por otra parte, este defecto se subsana mediante la asignación a cada agente del conjunto de datos del TKB que utiliza de una forma más intensa. Esta otra alternativa presenta también su inconveniente, y es que cada vez que un agente precisa de datos que no posee debe de enviar un mensaje de petición de los datos al agente adecuado y realizar un proceso de espera hasta que los recibe. Además, cada agente de be de conocer donde reside cada posible dato que pueda necesitar, por lo que la distribución de datos del TKB debe de ser pública y estática.

Para determinar que alternativa escoger se plantea un estudio previo en el sistema MASHGREEN SM del tipo de datos que utiliza cada agente, así como del instante temporal en que los necesita.

[1] http://www.sics.se/sicstus.html
[2] http://www.binnetcorp.com/BinProlog/

En el supuesto de que no existan acciones de control externas, los datos cualitativos iniciales de cada objeto del sistema son generados por el agente de preproceso en el instante de comienzo del actual periodo de integración. Estos datos, una vez calculados, pueden ser utilizados por el agente de solución para empezar a analizar las posibles consecuencias de la modificación de los vectores de reparto en cada cruce. Según el agente de preproceso simule el primer subciclo, aquellas situaciones problemáticas detectadas son enviadas al agente de detección para que éste las analice de forma conjunta. Como consecuencia de este análisis se puede detectar situaciones problemáticas espacio-temporales que son comunicadas al agente de diagnosis.

Al acabar de calcular los datos correspondientes al primer subciclo, el agente de preproceso comunica al agente de predicción los últimos datos cualitativos de cada objeto del sistema para que este continúe calculando su evolución desde este instante hasta que llegue a generar toda su ventana temporal de predicción de futuro.

En el comienzo de ejecución del subciclo siguiente el agente de preproceso pregunta al agente de predicción si dispone de todos los datos del subciclo actual, si este es el caso, entonces el agente de predicción se los suministra al agente de preproceso que se encarga de almacenarlos en el nivel actual del panel de aplicación en el TKB. En el caso que el agente de predicción no disponga de esos datos el agente de preproceso se encarga de generarlos y deja al agente de predicción que continúe su tarea. Tanto el agente de predicción como el de preproceso, durante el proceso de simulación cualitativa, detectan las situaciones elementales y temporales que van apareciendo, y como consecuencia, las comunican al agente de detección.

Este proceso continua hasta que finaliza el actual periodo de integración. En el instante de comienzo del siguiente periodo el agente de preproceso analiza los datos proporcionados por los sensores de tráfico y reconstruye la evolución del periodo de integración anterior comunicando al agente de diagnosis todas aquellas situaciones problemáticas temporales y elementales que va detectando. Estos nuevos datos son almacenados en el nivel pasado del panel de la aplicación del TKB.

El agente de diagnosis analiza estas situaciones y las compara con las recibidas desde el agente de detección en el anterior periodo de integración. Para poder realizar su tarea el agente de diagnosis necesita también los datos de simulación del agente de preproceso durante el periodo de integración anterior. Estos datos se los solicita al agente de preproceso y, como resultado lateral este agente los elimina del nivel actual del nivel de aplicación del TKB. Con todos los datos necesarios el agente de diagnosis ya puede identificar las situaciones correspondientes a incidentes mediante la comparación de los datos cualitativos de cada objeto relacionado con una situación problemática durante su intervalo de detección en simulación con los datos cualitativos de cada situación problemática detectada según los datos de los sensores.

En el supuesto que se ejecuten acciones externas de control, el acceso a los datos de la aplicación sufre ligeras modificaciones. El agente de preproceso y el de detección deben realizar un análisis temporal de la acción externa ejecutada para determinar el instante a partir del cual los resultados calculados no son coherentes con la evolución

real del sistema. Además, el agente de predicción y el de solución deben de abortar sus ejecuciones puesto que esta acción externa ha modificado sustancialmente los valores iniciales de los elementos del sistema con los que estos agentes comenzaron a calcular sus resultados. Cuando esta acción externa de control finalice totalmente su ejecución, estos agentes solicitarán al agente de preproceso los datos necesarios para intentar realizar sus tareas hasta el momento de integración de datos de sensores.

Como se observa de todo este análisis, hay muy pocos datos del TKB que sean accedidos por varios agentes simultáneamente. Además, cuando esto sucede los accesos suelen ser de lectura de datos. No obstante, el instante temporal en que son solicitados varía entre las distintas peticiones y, por lo tanto, podría darse el caso de un agente esté utilizando datos iniciales que están siendo modificados por otro agente. Un ejemplo de esta situación podría darse entre el agente de preproceso y el de diagnosis cuando la ejecución de una acción de control externa implicará una retrosimulación justo antes del comienzo de un nuevo periodo de integración de datos.

Para evitar estas situaciones y dado que cada agente tiene un dominio de aplicación perfectamente identificado, se opta por dividir el TKB entre los agentes del sistema. Esta decisión permite que dados los datos iniciales que precisa un agente para su ejecución, la tarea a desarrollar pueda realizarse de una forma integra y sin tener que competir por datos del TKB con otros agentes, lo que también proporciona unas mejores prestaciones en el sistema.

El efecto negativo de esta decisión es el tiempo que tiene que esperar cada agente desde que solicita unos datos hasta que los recibe. En algunos casos este tiempo de espera ya estaba contemplado de forma implícita mediante la definición de las clases de agentes en el sistema MASHGREEN SM (en el que las peticiones de ejecución de agentes de la clase $i + 1$ no se podían realizar hasta que todas las peticiones de ejecución de agentes de la clase i no estaban completamente resueltas). Esta clasificación evidenciaba las dependencias de datos entre los resultados de cada agente y los datos iniciales necesarios para otros agentes.

La distribución de datos del TKB propuesta es la siguiente:

- El agente de preproceso contiene los datos del pasado y del presente que se está generando en cada instante del actual periodo de integración.

- El agente de predicción contiene los datos del futuro hasta la duración máxima de la ventana de predicción de futuro. Si parte de estos datos pertenecen al actual periodo de integración, entonces estos datos son suministrados al agente de preproceso en el caso de que estén disponibles en el momento en que se solicitan. En caso contrario estos datos son eliminados.

- El agente de detección contiene los datos enviados por el agente de preproceso y de predicción mediante mensajes.

- El agente de solución contiene los datos iniciales calculados por el agente de preproceso. Con estos datos construye los comportamientos del sistema durante el actual periodo de integración.

- El agente de diagnosis contiene los datos del último periodo de integración finalizado tanto en datos de simulación como en datos resultado del análisis de los datos procedentes de los sensores. Estos dos conjuntos de datos le son suministrados por el agente de preproceso mediante el envío de los mensajes adecuados.

Todos los agentes necesitan acceder a los componentes estáticos de cada uno de los elementos de la red urbana para poder realizar sus tareas. Por este motivo la descripción estática de cada uno de ellos se replica en cada agente. Por lo tanto, cada agente posee la descripción estática de todos los elementos del sistema y las descripciones dinámicas de aquellos elementos que le resulten más útiles según la distribución anterior.

Como consecuencia de este modo de funcionamiento y de la forma en que los agentes se distribuyen en el sistema el agente de refinamiento de las soluciones pierde parte de su razón de existencia como agente independiente. Este se debe a que ahora se le puede dedicar todo el tiempo necesario al agente de solución sin tener que ser interrumpido cada t_{ciclo} segundos, y por lo tanto, se pueden dedicar algunos segundos antes del cumplimiento de su tiempo de integración a realizar las tareas de refinamiento de las soluciones. Por lo tanto, el agente de refinamiento se integra dentro del agente de solución.

Otra característica de este nuevo sistema es que durante su periodo de desarrollo no se dispuso de datos reales de tráfico referentes a su dominio de aplicación. Por este motivo, y dado que el propósito principal de esta tesis no era desarrollar nuevas teorías y algoritmos para tratar cada una de las cuestiones relevantes en la gestión de control del tráfico urbano (aunque en algunos aspectos si que se han desarrollado nuevas técnicas y se han estudiado los beneficios de la aplicación intensiva y novedosa de otras técnicas) sino el diseñar un escenario en las que estas tareas se pudieran integrar y colaborar, se ha optado por no implementar el agente de diagnosis y que el agente de preproceso no realice la gestión de los datos de los sensores de tráfico del sistema.

Para tratar de realizar un sistema más completo y debido a la ausencia de estas tareas, se ha incorporado un nuevo agente que trata de determinar cual es la mejor **duración del ciclo** para cada uno de los ccis presentes en el dominio de aplicación dadas las características del tráfico durante un periodo de integración. Al igual que sucedía con los repartos calculados por el agente de solución, los resultados del agente de solución especializado en la duración de cada cci del sistema pueden ser implementados de forma automática o bien supervisado por el operador del sistema. Para implementar estas nuevas funciones se han modificado levemente algunos de los agentes del sistema. La descripción del agente de solución duración cci se expone en el apartado 4.8 de este capítulo.

4.1.2 Elección del modelo de ejecución

El sistema MASHGREEN DM se implementa en una arquitectura distribuida de PCs. Por lo tanto, una de las primeras tareas para su diseño es decidir el tipo de mecanismo de control que ejecutar. Las tendencias actuales en la programación y diseño de los sistemas distribuidos en Inteligencia Artificial se orientan cada vez más hacia un control distribuido en el que cada agente del sistema decide qué hacer y cómo hacerlo en cada instante.

No obstante, la elección de un modelo de control centralizado para el instante actual de desarrollo del prototipo aporta algunas ventajas de consideración:

- Las tareas de todos los agentes se han de sincronizar, al menos, una vez cada $t_{integración}$ segundos. Cada una de estas tareas puede ejecutarse varias veces en cada periodo de integración, por lo que resulta más sencillo dedicar a un agente del sistema la organización de estas secuencias de ejecución.

- Uno de los requisitos fundamentales en esta arquitectura es que los relojes internos de los agentes estén sincronizados entre sí. Resulta fundamental para el sistema evitar que coexistan dos o más relojes distintos. Esta coherencia respecto al tiempo resulta más sencilla de implementar si se dispone de un agente de coordinación que suministra la referencia temporal para el resto de agentes del sistema.

- El prototipo MASHGREEN DM está en fase de diseño e implementación por lo que resulta fundamental disponer de medidas de monitorización en el sistema para poder estudiar el comportamiento de cada agente del sistema. Estas tareas de monitorización resultan más sencillas de considerar si se centraliza la información en un único agente.

- La experiencia obtenida en el desarrollo del prototipo MASHGREEN SM permitió identificar unas dependencias de datos entre los agentes que podían verse afectadas al cambiar el modo de ejecución de estos agentes en el sistema. Por lo tanto, un mecanismo de control débilmente centralizado permite seguir trabajando con las mismas dependencias de datos e incluso identificar otras nuevas como consecuencia de la nueva arquitectura.

- El agente de coordinación sirve como agente de referencia sobre el que efectuar las tareas de sincronización del resto de agentes cuando así lo decida o se lo soliciten.

- El agente de coordinación sirve como almacén de datos de configuración del sistema por lo que accediendo a esa información se puede alterar esta configuración. Esta característica permite la incorporación de una característica muy útil en el sistema como es la posibilidad de introducir o de eliminar agentes en el sistema sin tener que parar la ejecución del sistema.

Más aún, mediante el uso de este mecanismo se podría desarrollar funciones de tolerancia a fallos en el sistema.

- Por último, pero no menos importante, no hay que olvidar que este sistema es un prototipo, no un producto comercial, y que por lo tanto, resulta más sencillo su desarrollo y puesta a punto si se ejecuta con un control centralizado.

Todas estas características de comportamiento se podrían implementar en un sistema distribuido, pero requiere un conocimiento más completo sobre la evolución del sistema del que se dispone a priori. Por lo tanto, la opción seleccionada consiste en desarrollar un primer prototipo, evaluarlo, y como resultado mejorar aquellos aspectos de su comportamiento que no sean completamente satisfactorios.

4.1.3 El problema de la referencia temporal común

Uno de los problemas principales asociados a sistemas distribuidos en dominios de aplicación temporal es la coexistencia de diferentes relojes físicos independientes entre sí. Este problema no es sencillo de resolver en su caso general [Tel, 94]. En el dominio de aplicación del tráfico urbano se ha optado por utilizar un mecanismo simple de sincronización de los relojes temporales de cada uno de los agentes del sistema. Este método se basa en las siguientes características:

- La unidad básica de medida temporal para cada objeto del sistema es el segundo. Esta medida es muy grande comparada con el tiempo de ejecución de cualquier acción atómica en un agente. Por lo tanto, no importa el instante concreto en que se decida ejecutar una acción, puesto que esta no podrá ser efectiva hasta el próximo tic del reloj. Por ejemplo, no importa que una acción se decida ejecutarla en el instante 1001, 1460 o 1999 milisegundos, puesto que no podrá ser efectiva hasta el instante 2000 milisegundos.

- El tiempo de transmisión de un mensaje por la red de interconexión entre los agentes del sistema, independientemente del tamaño del mensaje, es mucho menor que la unidad básica de medida temporal.

- Existe un agente en el sistema (el agente de coordinación) que posee el valor correcto para todos lo agentes del sistema. El resto de agentes deben de sincronizar sus relojes al valor del reloj del agente de coordinación.

- La existencia en el sistema de primitivas que permitan implementar mecanismos de sincronización por barrera y del envío de mensajes por multidifusión.

El proceso para realizar esta sincronización es el indicado en la figura 4.1

Este proceso de sincronización temporal se realiza obligatoriamente al inicio de la ejecución del sistema y cuando el sistema sufra fuertes alteraciones, como por ejemplo cuando se producen altas o bajas de agentes.

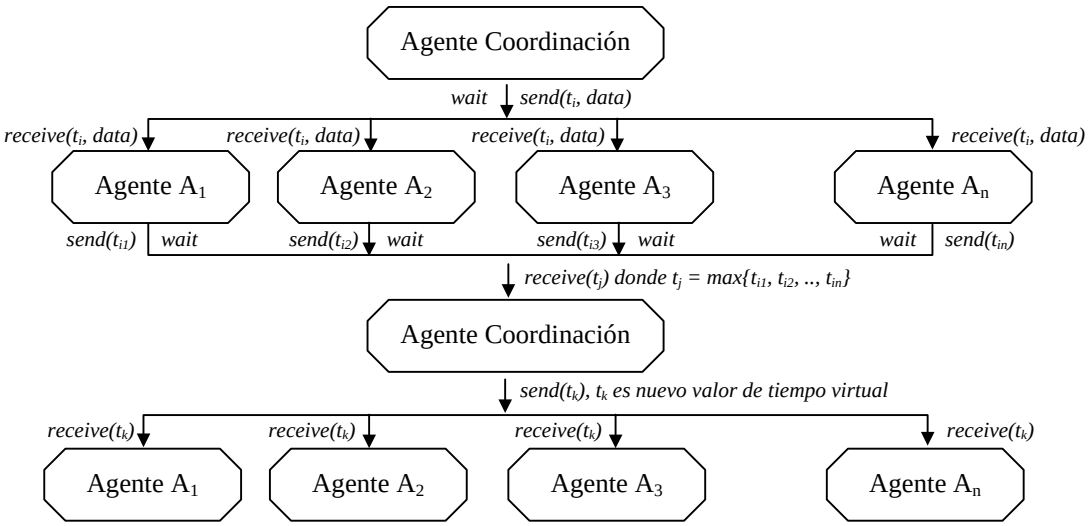


Fig 4-1 *Esquema de ejecución del proceso de sincronización del tiempo por barrera. En el instante t_i el agente de coordinación lanza la propuesta de sincronización y envía los datos asociados al motivo de esta sincronización. El resto de agentes reciben y procesan esa petición. Como consecuencia, cada uno de ellos envía al agente de coordinación la notificación de su recepción. El agente de coordinación, una vez recibidas todas las notificaciones, t_j, envía el nuevo valor de tiempo virtual, t_k, para el resto de agentes.*

No obstante, este método presenta un inconveniente. Aunque el valor final de los relojes de cada uno de los agentes del sistema es el mismo cuando finaliza el proceso (t_k en la figura 4-1)[3], no es el mismo el tiempo empleado por cada uno de los agentes para llevarlo a cabo.

Además, como se observa en la figura 4-1 algunos agentes devuelven la notificación de la petición antes que otros, y se quedan en proceso de espera hasta que el agente de coordinación les envíe el nuevo valor correcto para su reloj interno. Durante el proceso de espera, este agente podría estar recibiendo mensajes de otros agentes que todavía no han alcanzado su barrera y, por lo tanto, al no ser contestados, podría producirse una situación de bloqueo a tres. Por ejemplo, el agente de coordinación espera recibir la notificación de los agentes A_1 y A_2. El agente A_1 ha alcanzado la barrera y se pone en proceso de espera. El agente A_2 todavía no ha alcanzado la barrera y solicita datos al

[3] Debido a las características expuestas sobre el dominio de la aplicación y del sistema físico que lo implementa si el agente de coordinación tiene un tiempo t_k, el resto de agentes tienen o un tiempo t_k (en la mayoría de las situaciones en que se ejecute este proceso de sincronización temporal) o un tiempo $t_k + 1$ (debido a que el instante de sincronización se realizó pocos milisegundos antes del siguiente tic del reloj del agente de coordinación).

agente A_1. Como el agente A_1 está en proceso de espera, no atiende esta petición, y por lo tanto el agente A_2, si no posee un mecanismo de *timeout* para la petición, podría quedarse esperando indefinidamente, al igual que A_1 (que espera por el tiempo que debería de ser enviado por el agente de coordinación) y al igual que el agente de coordinación (que espera que el agente A_2 le envíe la notificación de recepción de la petición de sincronización por barrera).

Para evitar las situaciones de bloqueo mutuo cualquier agente que esté en estado de espera debe de garantizar respuesta a cualquier mensaje de petición de datos que se le envíe, notificando al agente que solicita los datos que no puede suministrárselos por encontrarse en estado de espera. El agente solicitante recibe esta respuesta y continúa su ejecución hasta que alcance la barrera.

Esta solución evita las situaciones de bloqueo pero no elimina el desequilibrio existente entre los distintos tiempos de espera de los agentes del sistema hasta que todos hayan alcanzado la barrera. Por este motivo, y dado que todos los relojes internos de los agentes del sistema avanzan con la misma velocidad (al menos con una velocidad que se puede considerar constante para las características de este dominio de aplicación), sólo se realizan sincronizaciones en las situaciones señaladas anteriormente. De esta forma se evita ejecutarlas de forma periódica lo que produciría un empleo de tiempo de proceso para alcanzar sincronizaciones temporales similares a las que ya se poseían antes de la sincronización.

4.2 Estructura y Organización de los Agentes

Los agentes del prototipo MASHGREEN DM poseen un mayor grado de autonomía que los definidos en el prototipo MASHGREEN SM. Cada uno de estos nuevos agentes incorpora tanto los datos que precisa para su ejecución como un mecanismo que le permite establecer comunicaciones con el resto de agentes del sistema. Cada agente en este sistema conoce la existencia del resto de agentes y de la parte del TKB del cual cada uno es responsable.

La posibilidad de incorporar o dar de baja a agentes dentro del sistema sin necesidad de parar su ejecución permite realizar tareas tales como dar de baja a agentes cuyos resultados sean considerados anómalos y activar otros agentes que produzcan resultados correctos mediante el uso de otros métodos [Corkill, 97]

Para poder integrar esta funcionalidad todos los agentes del sistema tienen que ser definidos utilizando unos mismos criterios. La definición de un *agente* no es una cuestión sencilla, puesto que cada sistema desarrollado suele aplicar una definición distinta. No obstante, de forma genérica suele estar bastante aceptada la definición de agente debida a Russell [Russell, 95]:

$$agente = arquitectura + programa.$$

4.2.1 Componente programa

El componente *programa* de la definición anterior debe presentar el mayor número posible de una serie de características que, en el dominio de aplicación del sistema MASHGREEN DM, son las siguientes:

- Autonomía: resulta limitada en el sistema debido a la decisión de ejercer un control débilmente centralizado. No obstante, son factibles dos tipos de iniciativas en cada uno de los agentes: las debidas al cumplimiento de ciertos instantes temporales en la ejecución de acciones (la evolución del tiempo), y las relacionadas con el envío y la recepción de mensajes con otros agentes del sistema.

- Colaboración: algunos agentes colaboran entre sí para intercambiar datos referentes al estado del dominio del problema. Como cada agente es responsable de un determinado conjunto de datos, cualquier agente que precise de datos que no pertenezcan a su espacio de datos debe de solicitarlos al agente adecuado. Si esta colaboración no finaliza de forma correcta, entonces esos datos debe de conseguirlos por otras vías (bien sea mediante cómputos locales o por inferencia a partir de los últimos datos recibidos). En un sentido más general, todos los agentes del sistema MASHGREEN DM colaboran para solucionar problemas del estado de tráfico urbano.

- Flexibilidad y Versatilidad. La decisión de la acción a ejecutar en cada instante está determinada por el estado del entorno. Este estado se construye con la información suministrada por otros agentes mediante mensajes y por la propia información histórica almacenada en cada agente. Por ejemplo, cada agente tiene una tarea específica a realizar que se vea alterada cuando se ejecutan acciones de control externas en el sistema.

- Representación del conocimiento. Los datos en cada agente se organizan según sea la tarea fundamental que realice dentro del sistema, es decir, su especialización. El mecanismo de representación subyacente a todos ellos es el modelo cualitativo de tráfico urbano expuesto en el capítulo 2, y estos datos se organizan en estructuras de datos más complejas tales como grafos de dependencias temporales/espaciales o en sistemas de pizarras divididas en múltiples niveles.

- Comunicación. En este sistema los datos que precisa cada agente para su ejecución, y que no están contenidos en su espacio de datos, se solicitan al agente que los contiene y se realiza un proceso de espera hasta su recepción. De esta forma mediante el envío y la recepción de mensajes se realiza la interpretación del estado del sistema de una forma continua y distribuida.

En resumen, cada agente en este sistema representa un conjunto de técnicas y de experiencias para la resolución de problemas. El objetivo fundamental del sistema global es coordinar habilidades, conocimientos, planes y experiencias adquiridas entre

distintos agentes para monitorizar, proponer y ejecutar acciones de control que mejoren el estado del tráfico urbano.

4.2.2 Componente arquitectura

La componente *arquitectura* se define como una metodología particular para la construcción de agentes, es decir, la especificación del mecanismo por el que un agente se divide en módulos y la forma en que estos módulos interaccionan entre sí.

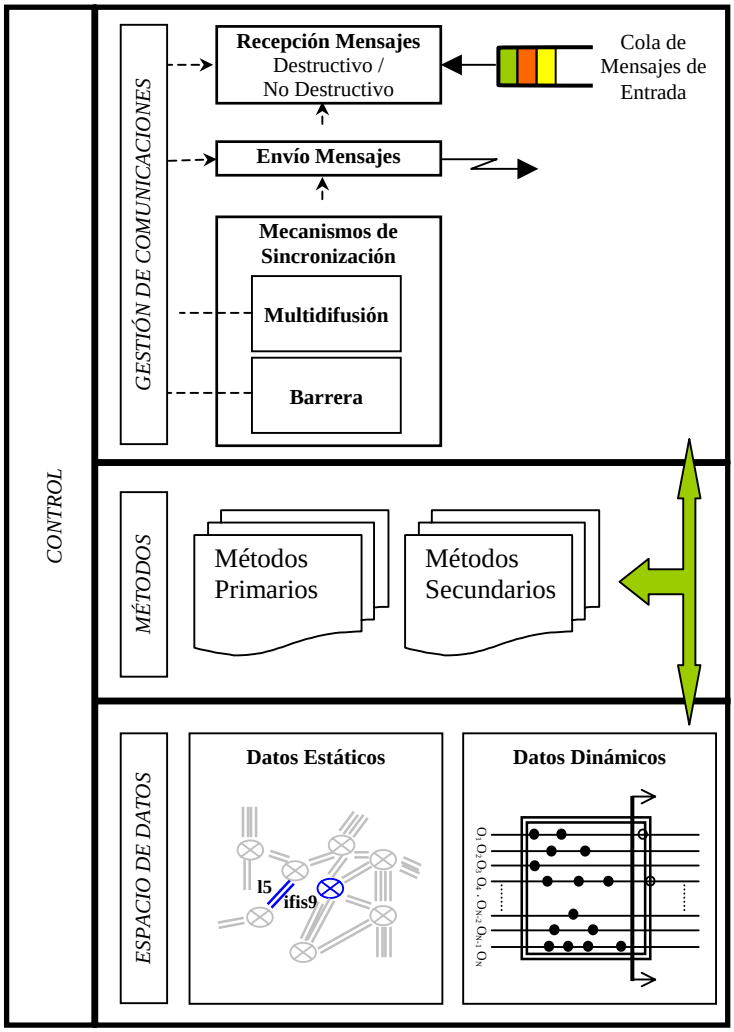


Fig 4-2 *Esquema de bloques de un agente del sistema. Se organiza en cuatro bloques: control, gestión de comunicaciones, métodos y espacio de datos Todos estos bloques pueden intercambiar información entre sí.*

El esquema de bloques de cualquier agente en el sistema MASHGREEN DM es el representado en la figura 4-2

Como ya fue descrito en apartados anteriores, cada agente posee su propio espacio de trabajo en el que almacena tanto los datos estáticos de cada objeto del sistema (e insertados durante el proceso de inicialización del mismo) como los datos dinámicos resultado de la ejecución de sus tareas.

El componente métodos clasifica las tareas que puede realizar el agente en dos categorías:

- Métodos primarios: Representan las tareas fundamentales a ejecutar por el agente y determina, por lo tanto, cuál es su especialización.

- Métodos secundarios: Representa a todas aquellas tareas administrativas del agente, bien sean internas (mantenimiento de la coherencia en su espacio de datos) o externas (coherencia con los datos que representan las creencias de otros agentes sobre el estado del sistema respecto a alguna de sus coordenadas, tareas de sincronización entre agentes o análisis de alertas)

Para poder garantizar que un agente pertenece (o puede pertenecer) a un agente de este tipo se debe de poder definir formalmente de acuerdo a la siguiente notación.

Definición

Un agente *A* del sistema *MASHGREEN DM* es una tupla de ocho componentes

$$A = (Task, s_0,, T, MsgTyp, K, IdAg, Agents, MsgSel)$$

donde

Task es el conjunto finito de tareas ejecutables por el agente. Se ha de cumplir que $Task = Task_p \cup Task_s$ donde $Task_p$ es el conjunto de tareas primarias y $Task_s$ es el conjunto de tareas secundarias. Además, algunas de las tareas secundarias pueden ejecutarse como herramientas auxiliares en la ejecución de tareas primarias.

s_0 es la tarea a ejecutar en la fase de arranque del agente *A*, por lo tanto $s_0 \in Task_s$, representa el proceso de inicialización del agente *A*.

T es una función que devuelve el tiempo virtual en el sistema (común a todos los agentes del sistema), $T: Z \rightarrow Z$, de forma que $T(t_i) = t_{vi}$, es decir, el instante t_i local a este agente, es el instante t_{vi} común para todos los agentes.

IdAg es el conjunto (infinito) de posibles identificadores de agentes.

MsgTyp es el conjunto finito de tipos de mensajes que el agente *A* puede recibir o enviar.

K es el conjunto finito de primitivas de comunicación en el sistema. Este conjunto debe de ser común para todos los agentes del sistema. En el sistema *MASHGREEN DM* este conjunto de primitivas es el formado por la primitiva *asyn_send* (envío asíncrono) y por *asyn_receive* (recepción asíncrona). Estas primitivas se combinan entre sí para conseguir otras formas de comunicación más avanzadas tales como *sync_send* (envío con sncronización), *sync_receive* (recepción con sincronización), *sync_barrier* (sincronización por barrera) o *broadcasting* (multidifusión de un mensaje al resto de agentes).

Agents es una función que devuelve el conjunto de agentes actuales en el sistema, $Agents: T \rightarrow 2^{IdAg}$, de forma que $Agents(T(t_i)) = Agents(t_{iv})$ es el conjunto de agentes activos en el instante t_{iv} común a todo el sistema.

MsgSel es la función de selección de qué mensaje analizar en cada instante, $MsgSel: 2^{Agents \times Agents \times MsgTyp \times T \times Data} \rightarrow Agents \times Agents \times MsgTyp \times T \times Data$, de forma que de todos los mensajes recibidos por este agente y todavía no tratados, esta función selecciona cual de ellos analizar. Esta función debe especificarse tanto para aquellos instantes en los que se ha de decidir que tarea ejecutar como durante el proceso de ejecución de alguna de ellas. Su definición debe de cumplir con dos propiedades para que la ejecución del agente resulte beneficiosa para el sistema. La primera es que todos los mensajes recibidos han de ser analizados o bien eliminados (se denomina la propiedad de *viveza* o de *no inanición*). La segunda propiedad es que no se pueden producir situaciones de *bloqueos mutuo* entre dos o más agentes del sistema. Esta situación se produce, por ejemplo, cuando un agente A_1 espera por datos de otro agente A_2, y este agente A_2 está esperando por datos del agente A_1.

En la descripción de la función *MsgSel* queda expuesto cual es el formato de todos los mensajes que se producen en el sistema: *Agents* x *Agents* x *MsgTyp* x *T* x *Data*. De forma que $(A_1, A_2, ejecución, t_0, Orden)$ es un mensaje si $A_1, A_2 \in Agents$, A_1 es el agente que envía el mensaje, A_2 es el agente que va a recibir el mensaje, *ejecución* identifica a un tipo de mensaje válido, $ejecución \in MsgTyp$, t_0 es el instante virtual temporal en que este mensaje fue enviado o va a ser enviado, y *Orden* es el comando que se desea ejecutar en el agente A_2.

Como ya se ha especificado previamente, esta arquitectura permite la definición dinámica de agentes, es decir, el programar tareas específicas para permitir alterar el conjunto de agentes activos en cualquier instante.

Una de las formas en que se podría implementar esta características sería por medio del siguiente proceso. El agente interfaz proporciona el mecanismo por el que el usuario da de alta o de baja a agentes en el sistema. En el caso que se trate de una alta, el

usuario también ha de proporcionar la máquina sobre la que se va a ejecutar. Esta información llega al agente de coordinación que actúa de la siguiente manera:

- Si se trata de una alta entonces el mensaje recibido es *<interfaz, control, new_agent, T_i, [agent_name, machine_name, execution_type]>*. En esta situación el agente de coordinación ejecuta en la máquina *machine_name* el agente *agent_name*. Además anota que este nuevo agente se ha de ejecutar en el comienzo de cada ciclo (*cycle*), en el comienzo de cada nuevo periodo de integración (*integration*) o de manera independiente con respecto a esas medidas temporales (*continuous*). Este nuevo agente ejecuta sus tareas de inicialización (es decir, s_0) y se espera hasta se que cumple el actual tiempo de integración en el sistema. En ese instante el agente de coordinación sincroniza todos los agentes del sistema para suministrarles la información asociada al nuevo agente, y de paso para suministrarles su valor de tiempo que sirva como referencia temporal para el resto de agentes.

- Si se trata de una baja entonces el mensaje recibido es *<interfaz, control, del_agent, T_i, [agent_name]>*. En esta situación el agente de coordinación envía un mensaje al agente *agent_name* para que escriba sus datos de monitorización que no hayan sido enviados hasta el momento y se autodestruye. En el momento en que se cumpla el actual tiempo de integración en el sistema el agente de coordinación envía un mensaje por barrera a todos los agentes activos del sistema comunicándoles el agente que ha causado baja y de paso vuelve a suministrarles su valor de tiempo que sirva como nueva referencia temporal.

Los resultados de la ejecución de cada nuevo agente en el sistema son enviados al agente interfaz. Si este agente no dispone de ventanas asignadas para la visualización de sus resultados, entonces el agente interfaz le proporciona una ventana de texto en la que volcar sus resultados.

El sistema *MASHGREEN DM* es un conjunto de agentes que cumplen con las especificaciones anteriores.

4.3 El Agente de Coordinación

El modo de ejecución del agente de coordinación se simplifica bastante con respecto al funcionamiento del agente de control implementado en el sistema MASHGREEN SM. Se mantienen las definiciones de los parámetros temporales t_{ciclo} y $t_{integración}$ para todos los agentes del sistema, pero las definiciones de las variables t_{actual} y $t_{simulación}$ se modifican levemente. El valor de la variable $t_{simulación}$ se define interno a cada agente y, por lo tanto, su valor puede diferir de unos agentes a otros. El valor de la variable t_{actual}es el valor del agente de coordinación para todos los agentes del sistema. Por lo tanto, el contenido de esta variable es común a todos ellos.

Cada uno de los componentes del agente de coordinación se define de la forma siguiente:

Tareas Primarias

Sólo posee una. La denominada *ciclo_largo*. Esta tarea especifica la secuencia de ejecución de agentes a realizar en el sistema durante el tiempo real de la integración de datos, es decir, durante $t_{integración}$ segundos.

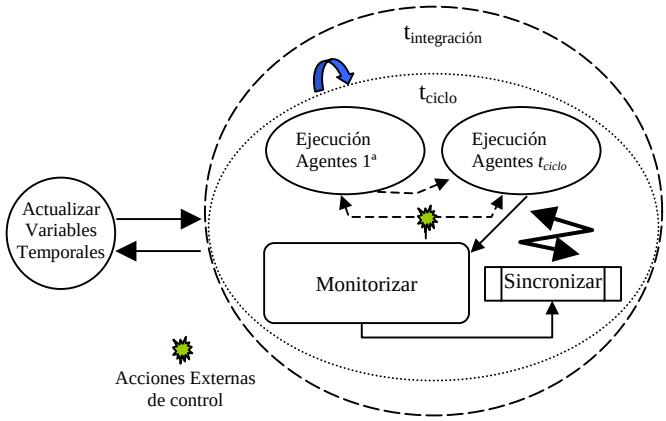


Fig 4-3 *Esquema de ejecución del agente de coordinación. El círculo externo indica el tiempo máximo dedicado a ejecuciones de los agentes que precisan de activaciones periódicas de algunos de sus métodos. Este tiempo se divide en intervalos de t_{ciclo} segundos. En cada uno de ellos se envían las peticiones de ejecución a los agentes que les corresponde ejecutarse y se dedica a consumir el resto del tiempo de este ciclo en recoger datos de monitorización del sistema y en acciones de sincronización.*

El agente de coordinación elimina las definiciones de clases de agentes establecidas en el prototipo MASHGREEN SM, el constructor de tareas de la agenda y el gestor de eventos. En su lugar implementa un ciclo de ejecución más sencillo en el que envía a cada agente un mensaje indicándole el momento en que debe de ejecutar alguno de sus métodos. Los agentes cuyo ciclo de ejecución sea menor o igual a t_{ciclo} y que se han de ejecutar en cada ciclo del sistema (como es el caso del agente de preproceso) reciben un mensaje de petición de ejecución al comienzo de cada ciclo, mientras que el resto de agentes cuya ejecución se tenga que realizar de forma periódica sólo reciben el mensaje de petición de ejecución al comienzo de cada periodo de integración.

Este mecanismo de envío de mensajes se puede alterar al ejecutar acciones de control externas sobre el sistema. En estas situaciones todos los agentes reciben un mensaje de notificación de la acción, y para aquellos que tengan que abortar los procesos que tenían en marcha, el agente de coordinación envía un nuevo mensaje de petición de ejecución en el comienzo del próximo ciclo.

Después de enviar las peticiones de ejecución a cada uno de los agentes que han de ejecutarse en el ciclo, el agente de coordinación dedica el resto de los t_{ciclo} segundos del ciclo actual en recoger los datos de monitorización que le envían el resto de agentes de la aplicación y en realizar acciones de sincronización cuándo sea oportuno.

Durante la ejecución de esta tarea el agente de coordinación puede recibir los siguientes mensajes:

Tipo Mensaje	Origen	Propósito
fin_sistema	INTERFAZ	acabar ejecución sistema
ciclo	PREPROCESO	cambio de valor de la duración del ciclo básico
fin_ejecución	PREPROCESO	el agente de preproceso ha finalizado la ejecución de su tarea primaria
no_futuro	PREPROCESO	cambios debidos a acciones externas de control

Como consecuencia de su ejecución el agente de coordinación puede enviar los siguientes mensajes:

Tipo Mensaje	Destino	Propósito
ejecución	DETECTOR	ejecución tarea primaria
ejecución	PREDICCIÓN	ejecución tarea primaria
ejecución	SOLUCIÓN	ejecución tarea primaria
ejecución	SOLUCIÓNCCI	ejecución tarea primaria
ejecución	PREPROCESO	ejecución tarea primaria

Como ya se ha indicado previamente el mensaje a PREPROCESO se envía cada t_{ciclo} segundos, y el resto de mensajes a otros agentes cada $t_{integración}$ segundos.

Esta secuencia se puede alterar en el caso de que se produzca la ejecución de acciones de control externas (que como ya se ha visto coincide con la recepción de un mensaje de tipo *no_futuro*).

Tareas Secundarias

Hay tres tareas secundarias: *iniciar_agentes*, *finalizar_sistema* y *monitorización*.

La tarea *iniciar_agentes* (se corresponde con s_0 como elemento de la tupla que define un agente) se ejecuta en el instante de activación del agente de coordinación en el sistema y consiste en la ejecución de tareas administrativas del propio agente (el establecimiento de sus datos estáticos y dinámicos en el espacio de datos) así como de la identificación y puesta en marcha de los agentes iniciales del sistema y del establecimiento de una sincronía por barrera para comunicar el tiempo de su reloj interno para que sirva de referencia para el resto de agentes del sistema.

Este agente envía el mensaje *inicio* al resto de agentes del sistema, y espera hasta recibir la confirmación de que el resto de los agentes han finalizado su proceso de inicialización mediante la recepción del mensaje *fin_inicio*.

La tarea *finalizar_sistema* se ejecuta como consecuencia de recibir un mensaje del tipo *fin_sistema* desde el agente de interfaz. Este mensaje puede recibirse tanto en estado de espera para el comienzo del siguiente ciclo de integración como durante la ejecución de su tarea primaria. Su ejecución implica el establecimiento de una sincronización por barrera para recibir los últimos datos de monitorización del resto de agentes, proceder a guardarlos en disco y enviar la orden de finalización de ejecución para todos ellos.

La tarea *monitorización* se ejecuta al recibir un mensaje de tipo *med*. Su ejecución implica la recogida de los datos de monitorización incluidos en el mensaje y su inserción en las estructuras de datos adecuadas.

Selección de mensaje a ejecutar

Al finalizar de ejecutar una tarea (primaria o secundaria) se analizan en el siguiente orden los mensajes que se hayan recibido (en el caso de estar presentes):

1. *fin_sistema*, al informar que se desea finalizar la ejecución del sistema este mensaje es de la máxima prioridad.

2. *ciclo*, informe de cambio en la duración del valor del parámetro t_{ciclo}.

3. *no_futuro*, informe de la ejecución de acciones de control externas en el sistema.

4. *med*, recepción de datos de monitorización.

El orden establecido anterior es fijo y se fundamenta en la relevancia de los mensajes recibidos respecto a la fidelidad con que el sistema representa al tráfico urbano real. Una característica común a todos los mensajes que puede recibir este agente es que al no existir dependencias de datos entre ellos no se pueden presentar situaciones de bloqueo mutuo.

Además, debido a las características del agente de control se garantiza la viveza de cada mensaje recibido que tenga relevancia para la ejecución del sistema, puesto que la única categoría de mensajes que puede presentar varios elementos es la de monitorización, y estos mensajes, en el caso de que no hubiese tiempo suficiente para su procesamiento completo (situación poco habitual puesto que el agente de control sólo realiza tareas que implican una ligera carga computacional, por lo que la mayor parte de su tiempo disponible de ejecución se encuentra en estado ocioso), no son fundamentales para los objetivos principales del sistema.

Durante el proceso de ejecución de la tarea primaria el agente de coordinación lanza (en el momento adecuado) y consume en el orden indicado los siguientes mensajes en el sistema:

1. ejecución (DETECTOR), ejecución (PREDICCIÓN), ejecución(SOLUCIÓN), ejecución (SOLUCIÓNCCI), ejecución (PREPROCESO).

2. consume los mensajes de monitorización (med).

3. consume los mensajes de multidifusión para los que no se haya especificado ninguna tarea asociada.

Las primeras acciones son inmediatas (no son síncronas, es decir, no se espera respuesta de recepción de ninguno de los agentes). Por lo tanto, se respeta la propiedad de viveza (al menos se garantiza para las peticiones relevantes para el objetivo principal del sistema) y se garantiza la no presencia de situaciones de bloqueo mutuo entre el agente de coordinación y cualquier otro agente del sistema.

4.4 El Agente de Preproceso

El agente de preproceso del sistema MASHGREEN DM cumple la misma función que el agente de preproceso del sistema MASHGREEN SM: conseguir que en todo momento la representación de la evolución del tráfico en el sistema sea coherente con su evolución real. Cada uno de los componentes del agente de preproceso se define de la forma siguiente:

Tareas Primarias

Sólo posee una. La denominada *preprocesar_datos*. Su esquema de funcionamiento se corresponde con el del agente de preproceso del sistema MASHGREEN SM (figura 3-7) [García, 97].

Esta tarea se ejecuta como consecuencia de recibir un mensaje del tipo *ejecución* desde el agente de coordinación. Este mensaje se recibe siempre al inicio de cada ciclo incluido en el periodo de integración de datos de los sensores en el sistema.

Durante la ejecución de esta tarea el agente de preproceso puede recibir los siguientes mensajes:

Tipo Mensaje	Origen	Propósito
cambio_coord	INTERFAZ	acción de control externa cambio en la coordinación de algún cruce
cambio_cci	INTERFAZ	acción de control externa cambio en la duración de algún cci
cambio_rep	INTERFAZ	acción de control externa cambio del vector de reparto en algún cruce

Como consecuencia de su ejecución el agente de preproceso puede enviar los siguientes mensajes:

Tipo Mensaje	Destino	Propósito
fin_ejecución	COORDINACIÓN	información de finalización tarea primaria
no_reparto *no_coord* *no_cci*	INTERFAZ	información al usuario que alguna acción de control externa no se ha realizado
tdb	INTERFAZ	visualización datos cualitativos tráfico actuales
tdbr	INTERFAZ	visualización datos cualitativos tráfico en retrosimulación
dame_datos_fut	PREDICCIÓN	petición datos futuro residentes en agente predicción

probprep	DETECTOR	problema detectado en simulación
ciclo	COORDINACIÓN	modificación del valor de ciclo en el sistema

Además de este envío de mensajes dirigidos a un único agente, esta tarea también puede realizar envío de mensajes por multidifusión:

Tipo Mensaje	**Propósito**
cambio_reparto *cambio_coord* *cambio_cci*	se ha podido realizar alguna acción de control externa
no_futuro *no_simulación*	como consecuencia se informa del instante en que se ha ejecutado

Tareas Secundarias

Hay siete tareas secundarias: *iniciar_agente*, *organiza_final*, *dame_datos*, *cambio_func*, *solución_repartos*, *solucion_cci* y *monitorización*.

La tarea *iniciar_agente* (componente s_0 de la tupla de definición de un agente en este sistema) se ejecuta como consecuencia de recibir un mensaje de tipo *inicio* desde el agente de coordinación. Este mensaje sólo se recibe una vez y se corresponde con el momento de arranque del sistema. Las funciones que realiza esta tarea son de establecimiento de los datos estáticos y dinámicos de su espacio de datos y la identificación de los agentes iniciales en el sistema.

En el instante en que este agente acaba de realizar todas estas tareas envía un mensaje de *fin_inicio* al agente de coordinación y se queda en estado de espera de la recepción del tiempo virtual del sistema (valor del reloj interno del agente de coordinación).

La tarea *organiza_final* se ejecuta como consecuencia de la recepción de un mensaje *fin_sistema* desde el agente de coordinación. Su ejecución implica el envío de sus últimos datos de monitorización al agente de coordinación y el permanecer en estado de espera hasta que se reciba el mensaje de conformidad desde el agente de coordinación para que el agente de preproceso se autodestruya.

La tarea *dame_datos* se ejecuta como consecuencia del procesamiento de un mensaje de tipo *dame_datos_act*. Este mensaje indica el instante temporal del cual se precisa conocer los datos y el agente solicitante. Como consecuencia, el agente de preproceso busca entre su espacio de datos dinámico si posee, o puede calcular, cual es el estado de cada uno de los objetos del dominio de la aplicación para ese instante. En caso afirmativo, el agente de preproceso recoge los datos y se los suministra al agente solicitante por medio de un mensaje de tipo *toma_datos_act*.

Otra tarea secundaria que el agente de preproceso puede ejecutar es la denominada *cambio_func*. Esta tarea se ejecuta cuando se recibe un mensaje de tipo *chgmode* procedente del agente de interfaz e indica que el operador del sistema ha decidido cambiar el modo de funcionamiento del sistema MASHGREEN DM. Se definen tres tipos de funcionamiento:

1. Valores de reparto y de cci supervisados: las soluciones calculadas por los agentes solución repartos y solución duración de las ccis se muestran al operador del sistema en una ventana y éste decide que hacer con ellas.

2. Valores de repartos automáticos y de cci supervisado: las soluciones de nuevos valores de reparto se ejecutan sin intervención del operador del sistema, mientras que las soluciones respecto a la duración de cada cci del sistema se muestran en la pantalla.

3. Valores de repartos supervisados y de duración del cci automáticos: las soluciones de nuevos valores de reparto se muestran al operador mientras que las soluciones calculadas para la duración de cada cci se implementan de forma automática.

El agente de preproceso modifica el modo de funcionamiento actual en el sistema de acuerdo al valor indicado en este mensaje.

La tarea *solución_repartos* se ejecuta cuando el modo de funcionamiento del sistema es el de repartos automáticos (modo 2) y se ha recibido un mensaje del agente de solución indicando cuales son los nuevos valores de reparto para los cruces. Esta información es almacenada en el espacio de datos del agente de preproceso para que cuando cualquiera de esos cruces comience la primera fase se ejecute el nuevo vector de reparto a aplicar.

La tarea *solución_cci* se ejecuta cuando el modo de funcionamiento del sistema es el de duración de cci automático (modo 3) y se ha recibido un mensaje del agente de solución cci indicando cuales son los nuevos valores de la duración de cada cci del sistema. Esta información es almacenada en el espacio de datos del agente de preproceso indicando el nuevo valor del cci para cada uno de sus cruces. A medida que cada cruce alcance su primera fase entonces pasa a comportarse con los valores adecuados al nuevo valor de duración del cci al que pertenece. Un cci no estará estable hasta que todos sus cruces hayan actualizado los valores de las duraciones de cada una de sus fases para el reparto actual al nuevo valor de la duración.

Por último, la tarea *monitorización* se ejecuta cuando se ha alcanzado algún instante de ejecución preestablecido en el agente de preproceso. En ese instante se poseen datos de monitorización considerados relevantes que son enviados al agente de coordinación para su posterior procesamiento.

Selección de mensaje a ejecutar

Al finalizar de ejecutar una tarea (primaria o secundaria) se analizan en el siguiente orden los mensajes que se hayan recibido:

1. *fin_sistema*, al informar que se desea finalizar la ejecución del sistema este mensaje es de la máxima prioridad.

2. *chgmode*, ejecuta el más reciente en el caso de la presencia de varios mensajes de este tipo.

3. *ejecución*, sólo se recibe un mensaje por ciclo de ejecución del sistema. El agente se le puede considerar egoísta en el sentido que prioriza sus intereses (obtener cuanto antes los datos cualitativos del ciclo actual) respecto de los intereses de otros agentes.

4. *dame_datos_act*, en el caso de que se presenten varios mensajes de este tipo el agente de preproceso selecciona el más antiguo.

El orden establecido ha de garantizar el cumplimiento de las propiedades de viveza y de no bloqueo mutuo.

Para conseguir que se cumpla la propiedad de viveza se ha de garantizar que en t_{ciclo} segundos como máximo el agente de preproceso pueda preprocesar los t_{ciclo} segundos de evolución real en el sistema y, como mínimo, gestionar *N-3* peticiones de datos (es decir, de procesar *N-3* mensajes del tipo *dame_datos_act*) siendo *N* la cantidad de agentes iniciales en el sistema al que se le restan los agentes que no realizan estas peticiones: el agente de coordinación, el agente de preproceso y el agente de interfaz.

En cualquier caso, como ya se expuso en el capítulo cinco la ejecución del proceso de simulación cualitativa está acotado superiormente por el número de segmentos, de cruces y de sus correspondientes características estáticas. Por lo tanto, se puede establecer una cota superior respecto a su coste computacional que, junto con la estimación del coste asociada a la recogida de datos de simulación proporciona una medida de prestaciones que debe de ser posible alcanzarla con el computador destinado a ejecutar el agente de preproceso.

La propiedad de la no presencia de situaciones de bloqueo mutuo es sencilla de observar puesto que no existen dependencias de datos con otros agentes. El agente de preproceso genera datos que son suministrados a otros agentes bajo petición previa y no realiza ningún proceso de espera respecto a la conformidad con los datos enviados.

Durante el proceso de ejecución de la tarea primaria el agente de preproceso lanza y consume los siguientes mensajes en el sistema:

1. *chgmode*, recibe la petición de cambio de modo de funcionamiento del sistema MASHGREEN DM. Se puede recibir en cualquier instante de ejecución.

2. *cambio_reparto*, recibe la petición de cambio del vector de reparto a aplicar en un cruce y envía por multidifusión, en caso de ser efectivamente ejecutada, la notificación al resto de agentes. Se puede recibir en cualquier instante de ejecución.

3. *cambio_coord*, recibe la petición de cambio del valor de coordinación de un cruce y envía por multidifusión, en caso de ser efectivamente ejecutada, la notificación al resto de agentes. Se puede recibir en cualquier instante de ejecución.

4. *cambio_cci*, recibe la petición de cambio de valor de la duración de un cci y envía por multidifusión, en caso de ser efectivamente ejecutada, la notificación al resto de agentes. Se puede recibir en cualquier instante de ejecución.

5. *solrep*, recibe la notificación de los nuevos valores de vectores de reparto a aplicar en cada cruce en el instante en el que pasen a ejecutar su primera fase. Este mensaje se recibe, si el modo de funcionamiento del sistema es automático para los repartos, antes de comenzar la primera simulación con los datos obtenidos de la combinación de los datos preprocesados de los sensores junto con el proceso de simulación del periodo de integración anterior. Es decir, justo antes de empezar a realizar la simulación correspondiente al primer ciclo del nuevo periodo de integración.

6. *solcci*, recibe la notificación de los nuevos valores de duración de cada cci del sistema a aplicar en cada cruce perteneciente a cada cci en el instante en el que pasen a ejecutar su primera fase. Este mensaje se recibe, si el modo de funcionamiento del sistema es automático para los valores de los ccis, justo antes de empezar a simular en el primer ciclo del nuevo periodo de integración (en la misma situación que sucedía en el caso del mensaje *solrep*).

7. *dame_datos_fut*, envía la petición de datos al agente de predicción al comienzo de cualquier ciclo de ejecución del sistema que no sea el primero del actual periodo de integración y siempre que no se hayan ejecutado acciones externas de control sobre el sistema.

8. *probprep*, envía un mensaje de este tipo en el instante que se detecte un problema espacial o temporal (siguiendo las definiciones establecidas en el capítulo 2) al agente de detección para que realice un análisis más completo del estado del tráfico.

9. *tdb*, envía al agente de interfaz los datos cualitativos correspondientes a un nuevo instante de simulación en el presente ciclo.

Todos los mensajes enviados al agente de preproceso son consumidos en el instante en que pueden ser efectivamente ejecutados. Los mensajes de tipo *chgmode*, *solrep* y *solcci* se consumen al inicio del primer ciclo del periodo de integración.

Los mensajes de tipo *cambio_reparto*, *cambio_coord* y *cambio_cci* se consumen justo antes del comienzo de cualquiera de los ciclos incluidos en el actual periodo de integración puesto que la información que suministran puede alterar sustancialmente los valores cualitativos de los objetos del sistema. Por lo tanto, se garantiza la viveza de todos los mensajes recibidos por el agente de preproceso.

El comportamiento del agente de preproceso ante la llegada y envío de mensajes en cada ciclo de ejecución se puede representar mediante un diagrama de transición de estados como el de la figura 4-4.

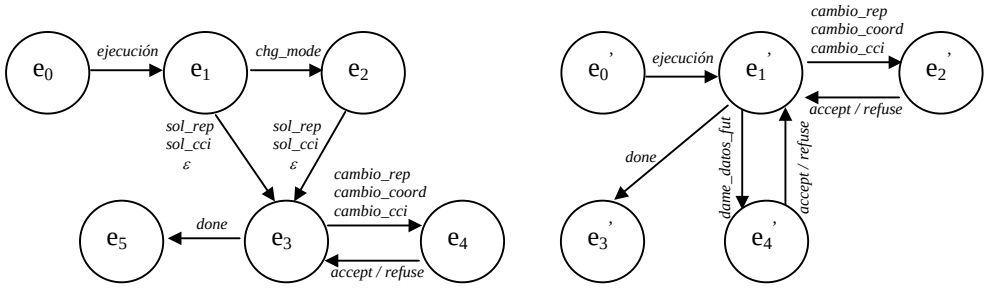


Fig 4-4. *Diagramas de transición de estados de la ejecución de la tarea principal del agente de preproceso. La figura de la izquierda representa los estados alcanzables para la primera ejecución del ciclo del periodo de integración. La figura de la derecha representa los estados alcanzables para el resto de ciclos del actual periodo de integración.*

Propiedad 4.1

La ejecución de la tarea principal del agente de preproceso finaliza en un tiempo finito.

Demostración

La única situación en la que puede presentarse una situación de bloqueo mutuo es en el estado e_4'. El resto de estados identifican situaciones que, o bien identifican situaciones de lectura de mensajes recibidos que no precisan de peticiones de datos adicionales, o bien son situaciones internas al agente. Sea *t* el instante de comienzo de la ejecución del ciclo actual. Si el agente alcanza el estado e_4' entonces comprueba si

existe alguna petición pendiente del agente de predicción que solicite datos del instante $t\text{-}t_{ciclo}$ (esta petición se realizó en el ciclo anterior al actual). En caso afirmativo, no se solicitan los datos evitando el bloqueo mutuo entre los agentes de preproceso y de predicción. Adicionalmente, en esta situación, se elimina la petición del agente de predicción y se le informa de la caducidad de su anterior petición. En caso contrario se solicitan los datos al agente de predicción puesto que no hay situación de bloqueo mutuo. En cualquier caso, los datos necesarios para la ejecución del agente de preproceso son o bien generados por el propio agente de preproceso utilizando datos pasados o bien recibidos desde el agente de predicción.

4.5 El Agente de Predicción

El agente de predicción tiene como función principal el adelantarse a los estados cualitativos de cada objeto antes de que estos cambios ocurran en el sistema de tráfico real. Además, como cálculo complementario, este agente realiza un primer análisis de los problemas de tráfico que pueden presentarse ante la ausencia de acciones de control externas y de la presencia de incidentes.

Cada uno de los componentes del agente de predicción se define de la forma siguiente:

Tareas Primarias

Sólo posee una. La denominada *predice*. Esta tarea se encarga de calcular la evolución del sistema desde el instante siguiente al valor $T_{coord} + t_{ciclo}$ hasta el valor $T_{coord} + T_{pred}$ donde T_{pred} es el tiempo máximo de predicción definido en el sistema (que se corresponde con la longitud máxima de la ventana temporal de futuro). El valor T_{coord} es el valor inicial del actual periodo de integración en el sistema que se corresponde con el contenido del reloj interno del agente de coordinación en ese instante.

Esta tarea se ejecuta cuando el agente recibe un mensaje del tipo *ejecución* procedente del agente de coordinación.

Una de las características de este agente es que no recibe directamente las soluciones calculadas por los agentes de solución o de solución para el cálculo de la duración de cada cci del sistema. Esto es así, puesto que en los datos iniciales de este agente ya se incluye toda la información referente a cual es la evolución de cada objeto del sistema, ya que estos datos son suministrados por el agente de preproceso que ya ha recibido, en caso de que el sistema se esté ejecutando en el modo de funcionamiento adecuado, las soluciones desde estos agentes.

Durante la ejecución de esta tarea el agente de predicción puede recibir los siguientes mensajes:

Tipo Mensaje	Origen	Propósito
fin_sistema	COORDINACIÓN	acabar ejecución sistema
no_futuro	PREPROCESO	eliminar toda la predicción de futuro debido a la ejecución de alguna acción de control externa
toma_datos_act	PREPROCESO	datos iniciales del instante $T_{coord} + t_{ciclo}$
dame_datos_fut	PREPROCESO	datos cualitativos de un instante múltiplo de t_{ciclo}

Como consecuencia de su ejecución el agente de preproceso puede enviar los siguientes mensajes:

Tipo Mensaje	Destino	Propósito
dame_datos_act	PREPROCESO	petición datos actuales residentes en agente preproceso
tdbf	INTERFAZ	visualización datos cualitativos tráfico futuro
toma_datos_fut	PREPROCESO	envío datos futuro residentes en agente predicción
probpred	DETECTOR	problema detectado en futuro

Para no sobrecargar la tarea del agente de interfaz, los datos cualitativos correspondientes a la predicción del tráfico futuro que se envían sólo hacen referencia al valor de cada objeto en el instante asociado a la finalización del cálculo en cada ciclo de predicción.

Tareas Secundarias

Hay cinco tareas secundarias: *iniciar_agente*, *organiza_final*, *dame_datos*, *acción_externa* y *monitorización*.

La tarea *iniciar_agente* (componente s_0 de la tupla) se ejecuta como consecuencia de recibir un mensaje de tipo *inicio* desde el agente de coordinación. Este mensaje sólo se envía una vez y se corresponde con el momento de arranque del sistema. Las funciones que realiza esta tarea son de establecimiento de los datos estáticos y dinámicos de su espacio de datos, la identificación de los agentes iniciales en el sistema y el establecimiento del tiempo virtual en todo el sistema (mediante el mismo mecanismo que el utilizado por el agente de preproceso).

La tarea *organiza_final* se ejecuta como consecuencia de la recepción de un mensaje *fin_sistema* desde el agente de coordinación. Su ejecución implica el envío de sus últimos datos de monitorización al agente de coordinación y el permanecer en estado de espera hasta que se reciba el mensaje de conformidad del agente de coordinación para que el agente de predicción se auotdestruya.

La tarea *dame_datos* se ejecuta como consecuencia del procesamiento de un mensaje de tipo *dame_datos_fut*. Este mensaje contiene el instante temporal del cual se precisa conocer los datos así como el agente que los solicita. Como consecuencia, el agente de predicción busca entre su espacio de datos dinámico si posee el estado de cada uno de los objetos del dominio de la aplicación para ese instante. En caso afirmativo, el agente de predicción recoge los datos y se los suministra al agente solicitante por medio de un mensaje de tipo *toma_datos_fut*.

La tarea *acción_externa* se ejecuta como consecuencia de la recepción de un mensaje de tipo *no_futuro*. Este mensaje indica que se ha ejecutado alguna acción de control externa en el sistema, y por lo tanto, la predicción realizada no es correcta. Como consecuencia la elimina y recoge los nuevos valores asociados a esa acción de control mediante el procesamiento de los tipos de mensajes adecuados: *cambio_rep*, *cambio_coord* y/o *cambio_cci*.

Por último, la tarea *monitorización* se ejecuta cuando se ha alcanzado algún instante de ejecución preestablecido en el agente de predicción. En ese instante se poseen datos de monitorización considerados relevantes que son enviados al agente de coordinación para su posterior procesamiento.

Selección de mensaje a ejecutar

Al finalizar de ejecutar una tarea (primaria o secundaria) se analizan en el siguiente orden los mensajes que se hayan recibido:

1. *fin_sistema*, al informar que se desea finalizar la ejecución del sistema este mensaje es de la máxima prioridad.

2. *no_futuro*, informe de la ejecución de acciones de control externas en el sistema.

3. *ejecución*, sólo se recibe un mensaje por ciclo de integración de datos en el sistema.

4. *dame_datos_fut*, en el caso de que existan varios mensajes de este tipo, el agente de predicción selecciona el más antiguo.

El orden establecido ha de garantizar el cumplimiento de las propiedades de viveza y de no bloqueo mutuo. El tiempo de ejecución de este agente depende en gran medida del tamaño de la ventana temporal de futuro que tenga que predecir. A mayor tamaño más tiempo tardará en procesar las peticiones de datos de futuro. Para evitar que las peticiones de datos que se le realicen tarden mucho tiempo en ser procesadas el agente de predicción también procesa estas peticiones durante la ejecución del proceso de predicción.

La propiedad de la no presencia de situaciones de bloqueo mutuo es sencilla de observar puesto que no existen dependencias de datos con otros agentes. El agente de predicción genera datos que son suministrados a otros agentes bajo petición previa y no realiza ningún proceso de espera respecto a la conformidad con los datos enviados.

Durante el proceso de ejecución de la tarea primaria el agente de predicción lanza y consume los siguientes mensajes en el sistema:

1. *no_futuro*, se puede recibir en cualquier instante de ejecución y provoca la finalización de la ejecución de esta tarea primaria y la lectura de aquellos mensajes que estén relacionados con la acción externa de control que ha provocado la emisión de este mensaje: *cambio_rep*, *cambio_coord* y *cambio_cci*.

2. *dame_datos_act*, envía la petición de datos al agente de preproceso al comienzo del primer ciclo de ejecución del sistema del periodo de integración y siempre que no se hayan ejecutado acciones externas de control sobre el sistema. Si se producen acciones de control externas entonces esta petición puede realizarla en el instante de comienzo de ejecución del ciclo en que se han ejecutado (o se van a ejecutar) las acciones externas de control.

3. *toma_datos_act*, recibe los datos que ya habían sido solicitados por este agente mediante el envío de un mensaje de tipo *dame_datos_act*.

4. *dame_datos_fut*, recibe la petición de datos de su espacio de datos dinámico.

5. *toma_datos_fut*, envía los datos solicitados por un mensaje recibido previamente del tipo *dame_datos_fut*.

6. *probpred*, envía un mensaje de este tipo en el instante que se detecte un problema espacial o temporal (siguiendo las definiciones establecidas en el

capítulo 2) al agente de detección para que realice un análisis más completo del estado del tráfico.

7. *tdbf*, envía al agente de interfaz los datos cualitativos correspondientes a un nuevo instante de predicción futuro.

El mensaje *no_futuro* es de prioridad máxima, puesto que si se recibe se ha de anular todo el proceso y los datos generados por el agente de predicción a partir del instante en que se haya ejecutado la acción de control externa que lo ha motivado. En este instante el agente de predicción analiza el tipo de acción externa ejecutada y modifica el valor de los objetos con los que está relacionada mediante la lectura de sus mensajes asociados: *cambio_rep*, *cambio_coord* y/o *cambio_cci*. Por lo tanto se garantiza la viveza de los mensajes recibidos no asociados a intercambio de datos.

El comportamiento del agente de predicción ante la llegada y envío de mensajes se puede representar mediante un diagrama de transición de estados como el de la siguiente figura.

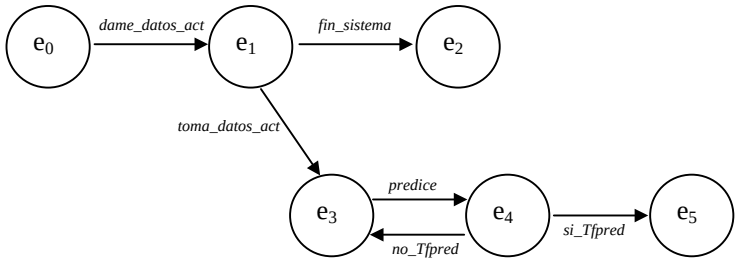


Fig 4-5. *Diagrama de transición de estados de la ejecución de la tarea principal del agente de predicción. El estado e_1 es un estado de espera en el que permanece el agente hasta que se reciben los datos solicitados o bien se acaba la ejecución del sistema. El estado e_4 calcula la evolución del sistema para los siguientes t_{ciclo} segundos de predicción de futuro hasta que se alcance el tiempo de ejecución máximo o se alcance el instante de integración de datos en el sistema.*

Propiedad 4.2

La ejecución de la tarea principal del agente de predicción finaliza en un tiempo finito.

Demostración

La única situación en la que puede presentarse una situación de bloqueo mutuo es en el estado e_1. En este estado el agente de predicción comprueba si tiene alguna petición de datos del agente de preproceso sin procesar antes de enviar su propia petición. En caso afirmativo, se proporcionan los datos solicitados por el agente de preproceso evitando de esta forma el bloqueo mutuo entre los agentes de preproceso y de predicción. En caso contrario se solicitan los datos al agente de preproceso puesto que

no hay situación de bloqueo mutuo. En cualquier caso, los datos necesarios para su ejecución son recibidos pudiendo, por lo tanto, realizar la transición al siguiente estado del diagrama.

4.6 El Agente de Detección

El agente de detección analiza los problemas espaciales y temporales detectados por los agentes de predicción y de preproceso para tratar de determinar los problemas espacio-temporales en la evolución del tráfico.

Cada uno de los componentes del agente de predicción se define de la forma siguiente:

Tareas Primarias

Sólo posee una. La denominada *detección*. Esta tarea se encarga de construir y de mantener los grafos de dependencias espacio-temporales que representan las situaciones problemáticas de tráfico identificadas en el sistema [García, 00a].

Esta tarea se ejecuta cuando el agente recibe un mensaje del tipo *ejecución* procedente del agente de coordinación.

El diagrama de transición de estados de esta tarea es el siguiente:

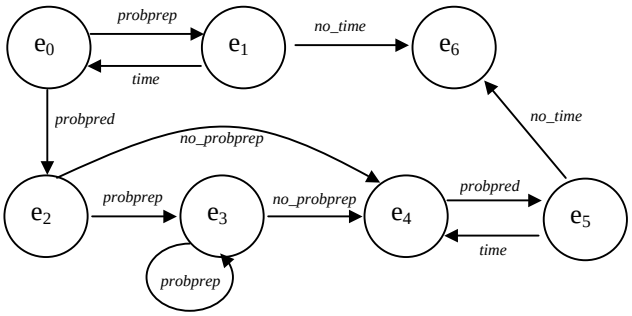


Fig 4-6. *Diagrama de transición de estados de la ejecución de la tarea principal del agente de detección. El estado e_0 es un estado de espera de recepción de problemas del ciclo actual. Si se recibe un problema asociado a datos de predicción es debido a que todos los problemas actuales ya han sido detectados, luego se han de procesar (estado e_3). El estado e_4 procesa los problemas detectados por el agente de predicción.*

Una de las características de este agente es que el orden en el que se reciben los mensajes indicando situaciones problemáticas no tiene porque ser secuencial. Por lo tanto, en este sistema se ha de establecer un mecanismo de encadenamiento temporal entre mensajes para que la información que proporcionen sea convenientemente procesada en los grafos de dependencias espacio-temporales. Para conseguir este comportamiento cada agente emisor de uno de estos mensajes identifica

secuencialmente el orden en el que los está enviando. De esta forma, el agente de detección procesa el siguiente al último mensaje analizado.

Durante la ejecución de esta tarea el agente de detección puede recibir los siguientes mensajes:

Tipo Mensaje	Origen	Propósito
fin_sistema	COORDINACIÓN	acabar ejecución sistema
no_futuro	PREPROCESO	eliminar el grafo de futuro y modificar el de presente debido a la ejecución de alguna acción de control externa
probprep	PREPROCESO	modificar el grafo de presente con el nuevo problema detectado
probpred	PREDICCIÓN	modificar el grafo de futuro con el nuevo problema detectado

Como consecuencia de su ejecución el agente de detección puede enviar los siguientes mensajes:

Tipo Mensaje	Destino	Propósito
elemental	INTERFAZ DIAGNOSIS	aviso detectado problema elemental
temporal	INTERFAZ DIAGNOSIS	aviso detectado problema temporal
camino	INTERFAZ DIAGNOSIS	aviso secuencia abierta de cruces con problemas
deadlock	INTERFAZ DIAGNOSIS	aviso secuencia cerrada de cruces con problemas

Tareas Secundarias

Hay cuatro tareas secundarias: *iniciar_agente*, *organiza_final*, *acción_externa* y *monitorización*.

La tarea *iniciar_agente* (componente s_0 de la tupla) se ejecuta como consecuencia de recibir un mensaje de tipo *inicio* desde el agente de coordinación. Este mensaje sólo se recibe una vez y se corresponde con el momento de arranque del sistema. Las funciones que realiza esta tarea son de establecimiento de los datos estáticos y dinámicos de su espacio de datos, la identificación de los agentes iniciales en el sistema y el establecimiento del tiempo virtual en todo el sistema (mediante el mismo mecanismo que el utilizado por el agente de preproceso). Adicionalmente, también realiza cálculos auxiliares respecto del grafo topológico asociado al dominio de la aplicación, como por ejemplo, la identificación de los cruces vecinos a cada cruce.

La tarea *organiza_final* se ejecuta como consecuencia de la recepción de un mensaje *fin_sistema* desde el agente de coordinación. Su ejecución implica el envío de sus últimos datos de monitorización al agente de coordinación y el permanecer en estado de espera hasta que se reciba el mensaje de conformidad del agente de coordinación para que el agente de detección se autodestruya.

La tarea *acción_externa* se ejecuta como consecuencia de la recepción de un mensaje de tipo *no_futuro*. Este mensaje indica que se ha ejecutado alguna acción de control externa en el sistema, y por lo tanto, la información representada en el grafo futuro de dependencias espacio-temporales no es correcta. Como consecuencia, se elimina el grafo de futuro y se actualiza el grafo de presente al instante de ejecución de la acción de control externa. Además, el agente de detección recibe los nuevos valores asociados a esa acción de control mediante los tipos de mensajes adecuados: *cambio_rep*, *cambio_coord* y/o *cambio_cci*, y en el caso que se haya cambiado el valor de la duración de algún cci en el sistema, el agente actualiza los cruces pertenecientes a ese cci para que reflejen el cambio realizado.

Por último, la tarea *monitorización* se ejecuta cuando se ha alcanzado algún instante de ejecución preestablecido en el agente de detección. En ese instante se poseen datos de monitorización considerados relevantes que son enviados al agente de coordinación para su posterior procesamiento.

Selección de mensaje a ejecutar

La tarea de detección en este agente tiene un tiempo dedicado de ejecución que coincide con el periodo de integración de datos en el sistema. Por este motivo, cualquier mensaje que se recibe es procesado durante la ejecución de la tarea primaria.

Durante el proceso de ejecución de esta tarea primaria el agente de detección lanza y consume los siguientes mensajes en el sistema:

1. *fin_sistema*, al informar que se desea finalizar la ejecución del sistema este mensaje es de la máxima prioridad.

2. *no_futuro*, informe de la ejecución de acciones de control externas en el sistema, como consecuencia se ejecuta la tarea secundaria *acción_externa*.

3. *probprep*, se selecciona el siguiente según la etiqueta identificativa del último problema del presente procesado.

4. *probpred*, se selecciona el siguiente según la etiqueta identificativa del último problema de predicción de futuro procesado.

5. *elemental*, *temporal*, *camino* y *deadlock*. Los mensajes de estos tipos se envían a los agentes interesados en el momento en el que los problemas que representan son identificados.

El orden establecido hace que todos los mensajes relevantes para la interpretación de la evolución del sistema sean procesados. Si se produce un mensaje *fin_sistema* entonces los mensajes pendientes no son interesantes puesto que se ha decidido poner fin a la ejecución.

En el caso de recibir un mensaje de tipo *no_futuro* entonces se eliminan todos los mensajes de tipo *probpred* y aquellos mensajes de tipo *probprep* cuya etiqueta temporal sea superior al instante de ejecución de la acción de control externa puesto que son mensajes cuyo contenido puede no ser coherente con la evolución del sistema.

Los mensajes de tipo *probprep* se procesan en el orden en que fueron enviados (que puede no ser el mismo que el orden en que se hayan recibido). Para facilitar este proceso, se consumen de forma secuencial a su etiqueta numérica, que es un dato referente asignado por el agente emisor.

Los mensajes de tipo *probpred* se consumen cuando no queden mensajes de tipo *probprep* por procesar. Cuanto más avanzado en el tiempo sean los mensajes de tipo *probpred* recibidos menor será su importancia relativa para la evolución del sistema, puesto que cada $t_{integración}$ segundos se reajusta el sistema, y por lo tanto, cuando más se separe de esta frontera temporal mayor será la distancia que exista entre los problemas previstos en el futuro y los que se produzcan cuando se cumpla ese instante temporal en la evolución del tráfico.

Se puede concluir, por lo tanto, que este agente cumple con la propiedad de vivacidad para todos los mensajes que recibe.

La propiedad de la no presencia de situaciones de bloqueo mutuo es sencilla de observar puesto que no existen dependencias de datos respecto a otros agentes.

4.7 El Agente de Solución Repartos

El agente de solución repartos tiene como función principal el calcular el vector de reparto a aplicar en cada cruce durante todo el actual periodo de integración que sea capaz de reducir el número de situaciones problemáticas detectadas en la evolución del tráfico de la red urbana.

A diferencia del funcionamiento que presentaba el agente de solución en el prototipo MASHGREEN SM este nuevo agente no tiene porqué interrumpir sus cálculos cada t_{ciclo} segundos, puesto que ahora dispone de todo el tiempo de su periodo de integración para construir el nuevo plan de señales. Por este motivo la calidad de la solución aportada mejora puesto que se realiza una visión más global y continua de la evolución del sistema y ninguna asignación de vector de reparto a un cruce se considera estabilizada hasta que no se han comprobado todas las posibles opciones.

Resulta conveniente destacar que, debido al proceso de selección que conduce a una hoja del potencial árbol de evolución del sistema según el vector de reparto a aplicar en cada instante, una vez conseguida la primera solución el proceso de vuelta atrás realiza un gran número de podas eliminando grandes conjuntos de planes de señales razonablemente peores que el obtenido en el cálculo de la primera hoja.

Cada uno de los componentes del agente de solución repartos se define de la forma siguiente:

Tareas Primarias

Sólo posee una. La denominada *solución_rep*. Esta tarea se encarga de determinar la mejor asignación de vectores de reparto a cada uno de los cruces del sistema según el árbol de evolución del tráfico asociado al periodo de integración actual. Esta tarea utiliza de forma intensa las situaciones problemáticas espaciales y temporales identificadas en el proceso de simulación en cada instante.

El agente de solución repartos ejecuta esta tarea cuando recibe un mensaje del tipo *ejecución* procedente del agente de coordinación.

Esta orden normalmente se recibe al comienzo de cada periodo de integración, pero en el caso de que se haya producido la ejecución de alguna acción externa de control entonces este agente también recibe esa orden en el comienzo del ciclo siguiente al ciclo en el que se ejecutó la acción externa.

Durante la ejecución de esta tarea el agente de solución repartos puede recibir los siguientes mensajes:

Tipo Mensaje	Origen	Propósito
fin_sistema	COORDINACIÓN	acabar ejecución sistema
no_futuro	PREPROCESO	eliminar el actual proceso de generación del árbol
solcci	SOLUCIÓNCCI	recoger la solución calculada para las duraciones de los ccis
toma_datos_act	PREPROCESO	datos iniciales del instante de comienzo del cálculo de los mejores repartos
chgmode	INTERFAZ	notificación del modo de funcionamiento del sistema

Como consecuencia de su ejecución el agente de solución repartos puede enviar los siguientes mensajes:

Tipo Mensaje	Destino	Propósito
dame_datos_act	PREPROCESO	petición datos actuales residentes en agente preproceso
solrep	INTERFAZ PREPROCESO SOLUCIONCCI	plan de señales generado durante el actual periodo de integración

Tareas Secundarias

Hay cinco tareas secundarias: *iniciar_agente*, *organiza_final*, *acción_externa*, *cambio_func* y *monitorización*.

La tarea *iniciar_agente* (componente s_0 de la tupla) se ejecuta como consecuencia de recibir un mensaje de tipo *inicio* desde el agente de coordinación. Este mensaje sólo se recibe una vez y se corresponde con el momento de arranque del sistema. Las funciones que realiza esta tarea son de establecimiento de los datos estáticos y dinámicos de su espacio de datos, la identificación de los agentes iniciales en el sistema y el establecimiento del tiempo virtual en todo el sistema (mediante el mismo mecanismo que el utilizado por el agente de preproceso). Además realiza cálculos auxiliares

respecto de la duración de cada una de las fases de cada cruce según el vector de reparto a aplicar. De esta forma se consigue que sea más rápido el proceso de generación de nodos en el árbol de evolución del sistema.

La tarea *organiza_final* se ejecuta como consecuencia de la recepción de un mensaje *fin_sistema* desde el agente de coordinación. Su ejecución implica el envío de sus últimos datos de monitorización al agente de coordinación y el permanecer en estado de espera hasta que se reciba el mensaje de conformidad del agente de coordinación para que el agente de solución repartos se autodestruya.

La tarea *acción_externa* se ejecuta como consecuencia de la recepción de un mensaje de tipo *no_futuro*. Este mensaje indica que se ha ejecutado alguna acción de control externa en el sistema. Si esa acción ha sido de cambio del valor del vector de reparto a aplicar entonces no tiene consecuencias en el proceso de generación del árbol. En el caso que la acción de control haya sido de modificación de la duración del cci o del valor de coordinación en algún cruce entonces el árbol de generación de planes desarrollado hasta el momento no es válido, y por lo tanto, se elimina. Como consecuencia, el agente de solución repartos acaba prematuramente la ejecución de su tarea principal. En cualquier caso, el agente de solución repartos recibe los nuevos valores asociados a la acción de control mediante el procesamiento de los tipos de mensajes adecuados: *cambio_rep*, *cambio_coord* y/o *cambio_cci*.

Otra tarea secundaria que el agente de solución repartos puede ejecutar es la denominada *cambio_func*. Esta tarea se ejecuta cuando se recibe un mensaje de tipo *chgmode* procedente del agente de interfaz e indica que el operador del sistema ha decidido alterar el modo de funcionamiento. Los modos posibles son los ya definidos en el apartado 4.4.

Por último, la tarea *monitorización* se ejecuta cuando se ha alcanzado algún instante de ejecución preestablecido en el agente de solución repartos. En ese instante se poseen datos de monitorización considerados relevantes que son enviados al agente de coordinación para su posterior procesamiento.

Selección de mensaje a ejecutar

Al finalizar de ejecutar una tarea (primaria o secundaria) se analizan en el siguiente orden los mensajes que se hayan recibido:

1. *fin_sistema*, al informar que se desea finalizar la ejecución del sistema este mensaje es de la máxima prioridad.

2. *no_futuro*, informe de la ejecución de acciones de control externas en el sistema.

3. *chgmode*, informe de la modificación del modo de funcionamiento del sistema.

4. *ejecución*, orden de ejecutar la tarea principal del agente. Esta orden normalmente será recibida una sóla vez en cada periodo de integración a no ser que se produzca la ejecución de alguna acción de control externa.

El orden establecido hace que todos los mensajes relevantes para la interpretación de la evolución del sistema sean procesados. Si se produce un mensaje *fin_sistema* entonces los mensajes pendientes no son interesantes puesto que se ha decidido poner fin a la ejecución.

En el caso de recibir un mensaje de tipo *no_futuro* entonces se reciben los datos asociados a la acción de control ejecutada.

El mensaje asociado a la notificación de cambio de modo de funcionamiento tiene una gran importancia para la construcción del árbol de repartos por lo que debe de procesarse antes de que este árbol se empiece a generar.

El procesamiento de todos los mensajes anteriores implica la ejecución de tareas de coste computacional muy ligero por lo que la mayor parte del tiempo disponible, hasta que se cumpla el final del actual periodo de integración, se dedica a ejecutar el mensaje correspondiente a la petición de ejecución de la tarea primaria del agente de solución repartos.

Se puede concluir, por lo tanto, que este agente cumple con la propiedad de vivacidad para todos los mensajes que reciba.

La propiedad de la no presencia de situaciones de bloqueo mutuo es sencilla de observar puesto que no existen dependencias de datos con otros agentes.

Durante el proceso de ejecución de la tarea primaria, y debido, a que esta tarea consume todo el tiempo que le sea posible hasta alcanzar el final del actual periodo de integración, el agente de solución repartos lanza y consume los siguientes mensajes en el sistema:

1. *fin_sistema*, recibe el mensaje que le informa que se desea finalizar la ejecución del sistema.

2. *no_futuro*, recibe la notificación de que ha sucedido alguna acción de control externa en el sistema.

3. *chgmode*, recibe la notificación de que se ha modificado el modo de funcionamiento del sistema, y que en el caso de que este modo implique la asignación de repartos automática entonces la solución que se calcule debe de enviarse al agente de preproceso además de al agente de interfaz.

4. *dame_datos_act*, envía la petición de datos al agente de preproceso para que le suministre los datos iniciales sobre los cuales construir el mejor plan de señales que le de tiempo a calcular.

Los dos primeros tipos de mensajes enviados al agente de solución repartos se consumen tan pronto sean recibidos, mientras que el mensaje de tipo *chgmode* se consume inmediatamente antes o después de la generación del plan de señales solución.

Por último, el mensaje *dame_datos_act* se envía justo antes de empezar la construcción del árbol de generación de planes de señales, y se espera por esos datos antes de continuar la ejecución de la tarea principal.

Como consecuencia de este funcionamiento se deduce que se cumple la propiedad de viveza y la de no presencia de situaciones de bloqueo mutuo puesto que aunque el agente de solución repartos realiza un proceso de espera por los datos del agente de preproceso, el agente de preproceso no realiza ninguna petición al agente de solución, y por lo tanto entre estos dos agentes no puede presentarse una situación de bloqueo mutuo.

4.8 El Agente de Solución Cci

La tarea del agente de solución cci consiste en determinar cual es la mejor duración de cada cci del sistema de forma que se reduzca el número de situaciones problemáticas generadas como consecuencia de la evolución del tráfico en la red urbana.

Esta tarea es nueva con respecto a las tareas realizadas por el prototipo MASHGREEN SM. Por este motivo, en este apartado se realiza un estudio más exhaustivo de su método de ejecución y de su esquema de funcionamiento que el realizado para el resto de agentes del prototipo MASHGREEN DM.

4.8.1 Breve Descripción del Método de Ejecución

Para determinar cual es la mejor duración de cada cci del sistema el agente de solución cci utiliza de forma intensa el modelo cualitativo de la evolución del tráfico urbano expuesto en el capítulo 2 de esta memoria de tesis.

La idea básica consiste en simular cual es el comportamiento del tráfico, la categoría y el tipo de problemas que se pueden identificar cuando se modifican los valores de duración de cada uno de los ccis del sistema.

En el sistema MASHGREEN DM cada cci sólo puede tomar uno de tres posibles valores: 80, 100 ó 120 segundos. El agente de solución cci simula $t_{integración}$ segundos con cada una de las posibles situaciones de duración del cci, de forma que si en la red urbana se definen N ccis entonces el número de situaciones a evaluar es N^3. Por ejemplo, si en una red urbana se definen dos ccis, cci_1 y cci_2, entonces se han de evaluar las siguientes situaciones: *{ {cci_1-80, cci_2-80}, {cci_1-80, cci_2-100}, {cci_1-80, cci_2-120}, {cci_1-100, cci_2-80}, {cci_1-100, cci_2-100}, {cci_1-100, cci_2-120}, {cci_1-120, cci_2-80}, {cci_1-120, cci_2-100}, {cci_1-120, cci_2-120} }.*

Al finalizar cada una de estas simulaciones se obtiene una medida que indica la bondad de la simulación realizada. Esta medida es el ya definido *nivel de congestión de la red de tráfico* (capítulo 3, agente solución). A mayor valor del *nivel de congestión* peor es la evolución del tráfico que representa.

Por lo tanto, un método para realizar esta tarea consiste en realizar todas estas simulaciones y quedarse con la mejores resultados proporcione. No obstante, este método presenta un fuerte inconveniente: el tiempo del que se dispone para realizar estas simulaciones es un tiempo finito, por lo que, según sea el tamaño de la red urbana a controlar, dado por su número de cruces y de segmentos, y la cantidad de ccis definidos en el sistema, puede no ser suficiente.

Por ello, a este sistema se le dota de una serie de heurísticas que permiten establecer un orden entre los posibles valores candidatos para cada cci. Las heurísticas implementadas son las siguientes:

- *Cambios mínimos*: su ejecución implica la ordenación de las tuplas de valores en orden inverso al número de modificaciones de cada cci a simular respecto de la situación actual, es decir, se ejecutan antes las simulaciones en las que el número de modificaciones en los ccis sea menor.

- *Persistencia de valores*: su ejecución implica la preferencia de ejecución a aquellas tuplas de valores que presenten modificaciones vecinas de los valores aplicados a cada cci en el instante actual. Si un cci está funcionando con un valor de *80* entonces se da prioridad a la tupla en la que se le asigna una nueva duración de *100* segundos. Si funciona con un valor de *120* entonces se da prioridad a la tupla en la que se le asigna una nueva duración de *100* segundos. Si el valor actual es de *100* segundos entonces, en función de un parámetro interno del agente (que depende de las características de la red urbana bajo control y del instante real del tráfico en el sistema), se elige o un valor de *80* o un valor de *120* segundos.

La definición de estas heurísticas corresponde al funcionamiento habitual de variación de los valores de los ccis en los sistemas de control de tráfico convencionales. La ejecución de un cambio en la duración del cci es una operación que precisa de un tiempo bastante largo hasta que alcanza de forma estable su valor final (en los actuales sistema de control de tráfico suele llevar un tiempo de ejecución de aproximadamente cinco minutos). Por lo tanto, esta es una operación que no se realiza de forma muy frecuente, por lo que en este agente se concede prioridad a los valores actuales (o más cercanos) de cada cci sobre aquellos que representen modificaciones bruscas de sus valores.

Otro de los inconvenientes de este proceso es que, a pesar de que el proceso de simulación cualitativa es un proceso con un coste computacional ligero (en el capítulo 2 se expuso que el número máximo de valores cualitativos a calcular responde a una relación polinómica respecto del número de segmentos y de cruces en el sistema), el tiempo disponible para realizar las N^3 simulaciones puede no ser suficiente.

Por lo tanto, en aquellos dominios de aplicación con un número elevado de ccis, este proceso se podría mejorar incorporando un mecanismo de poda que pudiese eliminar aquellas simulaciones que "previsiblemente" no vayan a proporcionar niveles de congestión razonablemente más pequeños que los obtenidos hasta ese instante.

Una medida que se puede utilizar para realizar esta poda consiste en estimar cada t_{ciclo} segundos en el sistema si el nivel de congestión que se puede alcanzar va a ser superior al mejor de los obtenidos hasta el momento. La elección de t_{ciclo} segundos responde a que es una cantidad de tiempo en la que casi todos los cruces de cada cci han ejecutado todas las fases que componen su reparto.

Esta estimación se calcula siguiendo el mismo método que el empleado en el agente de solución repartos, es decir, multiplicar el nivel de congestión actual por el número de segundos que quedan de simulación hasta alcanzar el instante en que se cumple el periodo de integración. Si este valor es más alto que el mejor valor obtenido hasta el momento entonces se realiza la poda y se prueba con la siguiente configuración no simulada hasta el momento.

4.8.2 Arquitectura del Agente de Solución Cci

El agente de solución cci implementa los métodos especificados en el apartado anterior. Cada uno de los componentes de su arquitectura se define de la forma siguiente:

Tareas Primarias

Sólo posee una. La denominada *solución_cci*. Esta tarea se encarga de determinar la mejor asignación de duración de cada cci del sistema modificando la duración de cada una de las fases para cada uno de sus cruces. El método utiliza de forma intensiva el proceso de simulación cualitativa junto con la detección y evaluación de los problemas espaciales y temporales generados durante este proceso.

El agente de solución cci ejecuta esta tarea cuando recibe un mensaje del tipo *ejecución* procedente del agente de coordinación.

Esta orden normalmente se recibe al comienzo de cada periodo de integración, pero en el caso de que se haya producido la ejecución de alguna acción externa de control entonces este agente también recibe esa orden en el comienzo del ciclo siguiente al ciclo en el que se ejecutó la acción externa.

La ejecución de esta tarea finaliza cuando se presente una de estas tres situaciones:

- cuando se hayan analizado todas las posibles asignaciones de valores de duración para cada cci del sistema,

- cuando se haya alcanzado el instante de integración de los datos de los sensores en el sistema,

- cuando se reciba la notificación de que se ha ejecutado alguna acción externa de control.

Durante la ejecución de esta tarea el agente de solución cci puede recibir los siguientes mensajes:

Tipo Mensaje	Origen	Propósito
fin_sistema	COORDINACIÓN	acabar ejecución sistema
no_futuro	PREPROCESO	eliminar el actual proceso de evaluación de alternativas
solrep	SOLUCIÓN	recoger la solución calculada para los repartos a aplicar en cada cruce
toma_datos_act	PREPROCESO	datos iniciales del instante de comienzo del cálculo de las mejores duraciones
chgmode	INTERFAZ	notificación del modo de funcionamiento del sistema

Como consecuencia de su ejecución el agente de solución cci puede enviar los siguientes mensajes:

Tipo Mensaje	Destino	Propósito
dame_datos_act	PREPROCESO	petición datos actuales residentes en agente preproceso
solcci	SOLUCIÓN PREPROCESO	mejor duración de ciclo de cada cci durante el actual periodo de integración
mincci	INTERFAZ	información al usuario de las características de la solución calculada

Tareas Secundarias

Hay cinco tareas secundarias: *iniciar_agente*, *organiza_final*, *acción_externa*, *cambio_func* y *monitorización*.

La tarea *iniciar_agente* (componente s_0 de la tupla) se ejecuta como consecuencia de recibir un mensaje de tipo *inicio* desde el agente de coordinación. Este mensaje sólo se recibe una vez y se corresponde con el momento de arranque del sistema. Las funciones que realiza esta tarea son de establecimiento de los datos estáticos y dinámicos de su espacio de datos, la identificación de los agentes iniciales en el sistema y el establecimiento del tiempo virtual en todo el sistema (mediante el mismo mecanismo que el utilizado por el agente de preproceso). Adicionalmente se realizan cálculos auxiliares respecto de la duración de cada una de las fases de cada cruce para todos los posibles vectores de reparto a aplicar y para todas las posibles duraciones de cada cci. De esta forma el proceso previo a la generación de cada simulación se realiza un poco más rápidamente.

La tarea *organiza_final* se ejecuta como consecuencia de la recepción de un mensaje *fin_sistema* desde el agente de coordinación. Su ejecución implica el envío de sus últimos datos de monitorización al agente de coordinación y permanecer en estado de espera hasta que se reciba el mensaje de conformidad del agente de coordinación para que el agente de solución cci se autodestruya.

La tarea *acción_externa* se ejecuta como consecuencia de la recepción de un mensaje de tipo *no_futuro*. Este mensaje indica que se ha ejecutado alguna acción de control externa en el sistema. Si esa acción ha sido de cambio del valor de la duración de algún cci del sistema entonces no tiene consecuencias para la ejecución de la tarea primaria. En el caso que la acción de control haya sido de modificación del vector de reparto o del valor de coordinación en algún cruce entonces el proceso realizado hasta el momento no es válido, y por lo tanto, se elimina. Como consecuencia, el agente de solución cci acaba prematuramente la ejecución de su tarea principal. En cualquier caso, el agente de solución cci recibe los nuevos valores asociados a esa acción de control mediante el procesamiento de los tipos de mensajes adecuados: *cambio_rep*, *cambio_coord* y/o *cambio_cci*.

Otra tarea secundaria que el agente de solución cci puede ejecutar es la denominada *cambio_func*. Esta tarea se ejecuta cuando se recibe un mensaje de tipo *chgmode* procedente del agente de interfaz e indica que el operador del sistema ha decidido alterar el modo de funcionamiento. Los modos posibles son los ya definidos en el apartado 4.4.

Por último, la tarea *monitorización* se ejecuta cuando se ha alcanzado algún instante de ejecución preestablecido en el agente de solución cci. En ese instante se poseen datos de monitorización considerados relevantes que son enviados al agente de coordinación para su posterior procesamiento.

Selección de mensaje a ejecutar

Al finalizar de ejecutar una tarea (primaria o secundaria) se analizan en el siguiente orden los mensajes que se hayan recibido:

1. *fin_sistema*, al informar que se desea finalizar la ejecución del sistema este mensaje es de la máxima prioridad.

2. *no_futuro*, informe de la ejecución de acciones de control externas en el sistema.

3. *chgmode*, informe de la modificación del modo de funcionamiento del sistema.

4. *ejecución*, orden de ejecutar la tarea principal del agente. Esta orden normalmente será recibida una sóla vez en cada periodo de integración a no ser que se produzca la ejecución de alguna acción de control externa.

El orden establecido hace que todos los mensajes relevantes para la interpretación de la evolución del sistema sean procesados. Si se produce un mensaje *fin_sistema* entonces los mensajes pendientes no son interesantes puesto que se ha decidido poner fin a la ejecución.

En el caso de recibir un mensaje de tipo *no_futuro* entonces se reciben los datos asociados a la acción de control ejecutada.

El mensaje asociado a la notificación de cambio de modo de funcionamiento tiene una gran importancia para la simulación, puesto que proporciona los vectores de reparto a aplicar en cada cruce, por lo que debe de procesarse antes de que esta simulación se empiece a realizar.

El procesamiento de todos los mensajes anteriores implica la ejecución de tareas de coste computacional muy ligero, por lo que la mayor parte del tiempo disponible, hasta que se cumpla el final del actual periodo de integración, se dedica a ejecutar el mensaje correspondiente a la petición de ejecución de la tarea primaria del agente de solución cci.

Se puede concluir, por lo tanto, que este agente cumple con la propiedad de vivacidad para todos los mensajes que reciba.

La propiedad de la no presencia de situaciones de bloqueo mutuo es sencilla de observar puesto que no existen dependencias de datos con otros agentes.

Durante el proceso de ejecución de la tarea primaria, y debido, a que esta tarea consume todo el tiempo que le sea posible hasta alcanzar el final del actual periodo de integración, el agente de solución cci lanza y consume los siguientes mensajes en el sistema:

1. *fin_sistema*, recibe el mensaje que le informa que se desea finalizar la ejecución del sistema.

2. *no_futuro*, recibe la notificación de que ha sucedido alguna acción de control externa en el sistema.

3. *chgmode*, recibe la notificación de que se ha modificado el modo de funcionamiento del sistema, y que en el caso de que este modo implique la asignación de cci de forma automática entonces la solución que se calcule debe de enviarse a los agentes de preproceso, solución e interfaz.

4. *dame_datos_act*, envía la petición de datos al agente de preproceso para que le suministre los datos iniciales sobre los cuales evaluar la mejor asignación de duración para cada cci en el sistema que le de tiempo a calcular.

Los dos primeros tipos de mensajes enviados al agente de solución cci se consumen tan pronto sean recibidos, mientras que el mensaje de tipo *chgmode* se consume inmediatamente antes o después de la evaluación de los posibles valores de duración para cada cci.

Por último, el mensaje *dame_datos_act* se envía justo antes de empezar la simulación, y se realiza un proceso de espera por esos datos antes de continuar la ejecución de la tarea principal.

Como consecuencia de este funcionamiento se deduce que se cumple la propiedad de viveza y la de no presencia de situaciones de bloqueo mutuo puesto que, aunque el agente de solución cci realiza un proceso de espera por los datos del agente de preproceso, el agente de preproceso no realiza ninguna petición al agente de solución cci, y por lo tanto, entre estos dos agentes no puede presentarse una situación de bloqueo mutuo.

4.9 El Agente de Interfaz

El agente de interfaz es el encargado de mostrar los resultados proporcionados por el resto de agentes del sistema y de recibir las peticiones de información y de ejecución de acciones de control externas realizadas por el usuario. Su ciclo de funcionamiento es continuo en el sentido que no es afectado por las longitudes del ciclo ni del periodo de integración de datos en el sistema.

Para realizar esta tarea el agente de interfaz se programa como dos procesos que intercambian información. Uno de los procesos es el encargado de almacenar los valores actuales de cada uno de los objetos del sistema mientras que el otro proceso se encarga de gestionar el entorno gráfico. Los dos procesos actúan unas veces como clientes y otras como servidores en función de la tarea que estén ejecutando en cada instante.

Cada uno de los componentes del agente de interfaz se define de la forma siguiente:

Tarea primaria

Sólo posee una, la denominada *interfaz*. Esta tarea se encarga de recibir todos los mensajes procedentes del resto de agentes del sistema, para analizar su información y mostrarla al usuario en el formato adecuado.

Una de las características de funcionamiento de este agente es que la visualización de la información recibida en algunos de sus mensajes sólo puede realizarse en concretos instantes temporales. Por ejemplo, la información de simulación asociada a un instante de tiempo real *t* debe de mostrarse cuando el reloj virtual del agente de interfaz cumpla ese instante. De esta forma se realiza una visualización de la evolución del tráfico de forma simultanea a su evolución real.

La misma situación sucede con respecto a la exposición de situaciones de alerta en el sistema o de visualización de los resultados calculados por los agentes que proporcionan alternativas de control (el agente de solución repartos y el agente de solución cci).

En el caso que la información proporcionada por un mensaje haga referencia a situaciones pasadas, respecto del instante actual del tiempo virtual del sistema, entonces a estos mensajes se les concede prioridad de procesamiento. Esta situación sucede cuando se han ejecutado acciones de control externas en el sistema que el agente de preproceso no ejecutó en el momento en que calculó la simulación, por lo que, una vez enterado debe de retroceder en la simulación realizada durante el ciclo anterior para volver a construir de nuevo el estado del tráfico para el comienzo del ciclo actual. Esta información se ha de visualizar tan pronto como esté disponible.

Por último, algunos otros mensajes contienen información cuya visualización no está relacionada con el instante temporal virtual del reloj del agente de interfaz. Esta información se muestra en los instantes de tiempo ociosos del agente de interfaz, es decir, en aquellos instantes en los que no se reciben mensajes de los tipos anteriores.

Durante la ejecución de esta tarea el agente de interfaz puede recibir los siguientes mensajes:

Tipo Mensaje	**Origen**	**Propósito**
tdb	PREPROCESO	datos simulación instante *t*
tdbr	PREPROCESO	datos simulación pasados al instante actual virtual del agente de interfaz
tdbf	PREDICCIÓN	datos simulación futuro
solrep	SOLUCIÓN	nuevos valores de reparto

mincci	SOLUCIÓNCCI	nuevos valores de duración de cada cci del sistema
no_futuro	PREPROCESO	eliminar la ventana de futuro debido a la ejecución de alguna acción de control externa
no_simulación	PREPROCESO	comienzo proceso de retrosimulación
cambio_reparto *cambio_coord* *cambio_cci*	PREPROCESO	informe de la ejecución de alguna (o varias) acciones de control externas
no_reparto *no_coord* *no_cci*	PREPROCESO	informe de que alguna acción de control solicitada no se puede ejecutar
elemental *temporal* *camino* *deadlock*	DETECTOR	presencia de alguna de las situaciones problemáticas en la evolución

Como consecuencia de su ejecución el agente de interfaz puede enviar los siguientes mensajes:

Tipo Mensaje	**Destino**	**Propósito**
fin_sistema	COORDINACIÓN	notificación de que el usuario desea finalizar la ejecución del sistema

Tareas Secundarias

Hay siete tareas secundarias: *iniciar_agente*, *dame_info*, *cambio_reparto*, *cambio_coord*, *cambio_cci*, *cambio_func* y *organiza_final*.

La tarea *iniciar_agente* (componente s_0 de la tupla) se ejecuta como consecuencia de recibir un mensaje de tipo *inicio* desde el agente de coordinación. Este mensaje sólo se recibe una vez y se corresponde con el momento de arranque del sistema. Las funciones que realiza esta tarea son de establecimiento de los datos estáticos y dinámicos de su

espacio de datos, la identificación de los agentes iniciales en el sistema y el establecimiento del tiempo virtual en todo el sistema (mediante el mismo mecanismo que el utilizado por el agente de preproceso). Esta tarea también se encarga de lanzar el proceso dedicado a gestionar el entorno gráfico, así como de preparar las estructuras de datos adecuadas para almacenar los datos asociados a cada una de las ventanas iniciales definidas en el sistema.

La tarea *dame_info* la solicita el usuario del sistema como consecuencia de la petición de información extra sobre los datos de alguna de sus ventanas de resultados. Esta petición la recibe el proceso de gestión del entorno gráfico que se la comunica al proceso gestor de datos activos para que le envíe los datos solicitados actuales. Estos datos pueden ser de muy distinta índole, como por ejemplo, el estado cualitativo actual de algún objeto del sistema, o los posibles vectores de reparto a aplicar en un determinado cruce.

Las tareas *cambio_reparto*, *cambio_coord* y *cambio_cci* se ejecutan bajo petición expresa del usuario. Como consecuencia de esta ejecución, se envía un mensaje al agente de preproceso notificándole el tipo de acción que se desea ejecutar y el instante temporal virtual de solicitud de esa acción.

El modo en que se pueden desencadenar la ejecución de acciones de control externas se realiza mediante la ejecución de alguna de estas tres tareas secundarias. El primer paso consiste en que el usuario las solicite. Esta petición produce la emisión de un mensaje al agente de preproceso que decide si se puede ejecutar o no es posible ejecutarla porque alguna otra acción de control sobre el mismo objeto se está realizando en ese momento. En el caso que la acción se puede ejecutar entonces el agente de preproceso se lo notifica al agente de interfaz en el momento en que se ha ejecutado para que le comunique al usuario el éxito de su petición. En caso contrario el agente de preproceso envía un mensaje de rechazo al agente de interfaz para notificar al usuario que su petición no va a poder ser ejecutada. Todas las informaciones proporcionadas al usuario son visualizadas en forma de alertas del sistema.

La tarea *cambio_func* se ejecuta bajo petición expresa del usuario del sistema y como consecuencia de su ejecución se envía un mensaje de notificación del nuevo modo de funcionamiento del sistema a los agentes de preproceso, solución repartos y solución cci.

Por último, la tarea *organiza_final* se ejecuta como consecuencia de una petición del usuario mediante la activación del botón de fin de sistema. Su ejecución implica el envío de esta petición al agente de coordinación, para que posteriormente este agente le comunique su conformidad respecto a esa petición cuando el resto de agentes hayan sido informados.

Selección de mensaje a ejecutar

Las tareas secundarias en este agente son ejecutadas bajo la petición del usuario del sistema (a excepción de la tarea *iniciar_agente* que es ejecutada en su fase de arranque).

Por lo tanto, la única situación en la que se pueden presentar varios mensajes simultáneamente es durante la ejecución de su tarea primaria.

La preferencia de selección de mensaje a procesar durante la ejecución de esta tarea es la siguiente:

1. *fin_sistema*, recibe y emite el mensaje que le informa que se desea finalizar la ejecución del sistema.

2. *no_futuro*, recibe la notificación de que ha sucedido alguna acción de control externa en el sistema.

3. *no_simulación*, recibe la notificación de que el agente de preproceso va a realizar una retrosimulación como consecuencia de la ejecución de alguna acción de control externa ya ejecutada en el dominio del tráfico real pero no tenida en consideración durante el proceso de simulación.

4. *tdbr*, datos de retrosimulación.

5. *tdb*, datos de simulación siempre y cuando hagan referencia a un tiempo real que sea menor o igual que el instante de tiempo virtual establecido en el sistema.

6. *cambio_rep*, *cambio_coord*, *cambio_cci*, *no_reparto*, *no_coord*, *no_cci* informe de que las peticiones de ejecución de acciones externas se han o no se han ejecutado satisfactoriamente.

7. *tdbf*, datos de simulación en predicción de futuro.

8. *solrep*, *mincci*, recibe la notificación de las soluciones calculadas por los agentes de solución repartos y solución cci.

9. *elemental*, *temporal*, *camino* y *deadlock*, recibe los problemas detectados en simulación actual y de futuro que han sido identificados por el agente de detección.

Este orden garantiza que todos los mensajes enviados al agente de interfaz son analizados cuando así sea conveniente para los intereses del sistema. Cuando se recibe un mensaje de tipo *no_futuro* entonces se eliminan todos los mensajes pendientes de proceso del tipo *tdbf* y aquellos mensajes de tipo *tdb* con datos asociados a momentos posteriores al instante en que se ejecutó la acción de control.

Además, la lectura de mensajes de tipo *tdb* y *tdbr* ha de realizarse de forma ordenada, según el instante en que estos mensajes fueron enviados. Por lo tanto, el método establecido ha de garantizar que no se queden mensajes por procesar respecto a situaciones pasadas. Este método consiste en enganchar un mensaje de un tipo con el siguiente mensaje del mismo tipo mediante el empleo de un dato “eslabón”. En este método este “eslabón” consiste en el tiempo final asociado a los datos del primer

mensaje. Este tiempo final debe de corresponderse con el tiempo inicial del siguiente mensaje del mismo tipo a procesar. Hasta que este mensaje no se reciba no se consume ningún otro mensaje del mismo tipo.

Por lo tanto, y dado que la única función de esta tarea es procesar mensajes, se garantiza la viveza de todos los mensajes enviados al agente de interfaz.

La propiedad de no presencia de situaciones de bloqueo mutuo se cumple puesto que el agente de interfaz no realiza ningún proceso de espera de datos procedente de otro agente (a no ser por las situaciones de lecturas encadenadas de mensajes de los tipos *tdb* y *tdbr* para los que ya se ha apuntado la corrección del método utilizado).

4.10 Conclusiones

En este nuevo prototipo, a diferencia de la especificación de los agentes del prototipo MASHGREEN SM, cada uno de los agentes se define mediante el establecimiento de una misma arquitectura funcional. Esta característica surge como consecuencia tanto en tratar de paliar las deficiencias detectadas en el prototipo MASHGREEN SM como del nuevo escenario de ejecución, un entorno distribuido, en el que los agentes deben de colaborar de una forma más independiente para realizar una mejor gestión del tráfico urbano. Para que esta colaboración se realice de una manera efectiva y eficiente resulta necesario ajustar más las capacidades de cada agente así como el modo en que son implementadas. Este ajuste implica la especificación de una forma más concreta de los datos, métodos y mecanismos de colaboración que puede ejecutar cada uno de los agentes en el nuevo prototipo.

Esta división explícita entre datos, métodos y control fundamenta la definición de la arquitectura de cada agente en el prototipo MASHGREEN DM. Esta característica permite establecer un modelo con el que escribir agentes con una mayor autonomía en su ejecución respecto de la ejecución del resto de agentes. Como efecto lateral de este incremento respecto de las habilidades en cada agente, también se precisa de un mecanismo extra para detectar la posible presencia de situaciones anómalas como son la ausencia de una misma referencia temporal común, la presencia de situaciones de bloqueo mutuo y de inanición.

El problema de la referencia temporal común afecta a todos los agentes del prototipo MASHGREEN DM y se soluciona estableciendo un valor de reloj común para todos los agentes. Este valor se establece durante el proceso de inicialización del prototipo mediante el uso de un mecanismo de sincronización por barrera.

El resto de las situaciones anómalas están asociadas a tareas de colaboración directa entre algunos de los agentes. No obstante, debido a las características del dominio de la aplicación del prototipo y que el prototipo trabaja a ciclos de $T_{integración}$ segundos, estas situaciones son resueltas mediante el análisis individual de cada agente respecto al resto agentes del prototipo. La garantía para que estas situaciones no se produzcan depende de cada agente de forma individual. Por ello, como parte de la definición de cada agente

se establece una función que debe garantizar la ausencia de las situaciones de inanición y de bloqueo mutuo.

CAPÍTULO 5

IMPLEMENTACIÓN DEL PROTOTIPO MASHGREEN DM

Índice del capítulo 5

5.1 Entorno de Aplicación

El sistema MASHGREEN DM se ha implementado en una red de computadores de tipo *Beowulf*[1].

5.1.1 Sistemas Tipo Beowulf

En la taxonomía estándar de computadores paralelos, los cluster *Beowulf* pertenecen a la intersección de las categorías de Procesadores Masivamente Paralelos (como el nCube, CM5, Convex SPP, Cray T3D, etc) y las Estaciones de Trabajo. Los sistemas *Beowulf* se benefician de los resultados obtenidos en el desarrollo de estas dos categorías. Los Procesadores Masivamente Paralelos gestionan un mayor número de procesadores que un sistema *Beowulf*, además de utilizar una red de interconexión con un menor tiempo de latencia. Los programadores de este tipo de sistemas tienen que preocuparse de cuestiones como la localidad, el equilibrado de carga, la granularidad y la sobrecarga de las comunicaciones para afinar sus programas y obtener las mejores prestaciones posibles. Aquellos programas que no precisan ni de una computación ni de unas comunicaciones de grano fino pueden ser trasladadas y ejecutadas de una forma efectiva en un sistema *Beowulf*. La tarea de programación en una red de estaciones de trabajo implica la ejecución de algoritmos que sean extremadamente tolerantes a problemas de equilibrado de carga y a altos valores de latencia en las comunicaciones. Cualquier programa que se esté ejecutando en una red de estaciones de trabajo debería de poder ejecutarse sin apenas cambios también en un sistema *Beowulf*.

Las redes de estaciones de trabajo y las redes de computadores de la clase *Beowulf* se distinguen por varias sutiles, pero significativas, características. Primero, los nodos que pertenecen al cluster sólo ejecutan tareas del cluster. Esta característica facilita la tarea del equilibrado de carga de trabajo debido a que las prestaciones de cada uno de los nodos no están sujetas a factores externos. Además, como la red de interconexión entre los nodos del sistema *Beowulf* está aislada de la red de comunicaciones externa, la carga de las comunicaciones está determinada únicamente por la aplicación que se está ejecutando en el cluster. Esta característica elimina los problemas asociados a la dificultad de predecir los valores de la latencia en redes de estaciones de trabajo. Todos los nodos que componen el cluster están bajo la jurisdicción administrativa del cluster. Por lo tanto, la única autentificación necesaria entre los procesadores de un sistema *Beowulf* se realiza para asegurar la integridad del sistema. Esto es así debido a que la red de interconexión del cluster no es accesible desde fuera de él. En una red de estaciones de trabajo se debe de asegurar la seguridad de la red. Otro ejemplo es que en un sistema *Beowulf* cada proceso se identifica con una identificador de proceso global. Esta característica permite implementar un mecanismo por el que un proceso de un nodo puede enviar señales a otro proceso residente en otro nodo del cluster sin tener que

[1] La dirección http://beowulf.gsfc.nasa.gov proporciona una buena información inicial para el desarrollo de este tipo de sistemas.

especificar el dominio del usuario. Esta característica no se puede ejecutar en una red de estaciones de trabajo. Por último, en un sistema *Beowulf* los parámetros del sistema operativo pueden ajustarse para mejorar las prestaciones del sistema. Por ejemplo, una estación de trabajo puede afinarse para que proporcione la mejor respuesta posible en interactivo, mientras que en un cluster los nodos se pueden afinar para que proporcionen la mejor productividad para tareas de grano gordo debido a que estos nodos no interaccionan directamente con los usuarios.

Esta arquitectura, generalmente está constituida por diversos componentes hardware, en los que se ejecuta un sistema operativo de soporte tipo linux (dominio público), conectados mediante una red privada de comunicaciones de altas prestaciones. Todos los nodos que componen un sistema de este tipo están conectados entre sí y cualquier conexión que se quiera realizar con algún otro computador que no pertenezca al cluster se ha de realizar por medio de un único nodo.

De esta forma resulta factible el disponer de una herramienta en la que realizar computación de altas prestaciones a un coste entre un tercio y un décimo del coste de un supercomputador tradicional.

5.1.2 Características Sistema Beowulf Utilizado

El grupo de investigación *"Computación de Altas Prestaciones"*[2] de la Universitat Jaume I de Castellón ha montado un sistema *Beowulf* cuyas principales características se exponen a continuación.

El sistema consta de un nodo servidor con el que establecer las comunicaciones del sistema al mundo exterior. Este nodo servidor es un computador con dos procesadores Pentium II a 400 MHz con 8 Gb de disco duro y 128 Mb de memoria RAM. El sistema operativo que se ejecuta tanto en este nodo servidor como en el resto de nodos del cluster es el sistema operativo Linux (la misma versión para todos ellos). El resto de nodos, nodos trabajadores del sistema, son computadores basados en un procesador Pentium II a 300 MHz con 3 Gb de disco duro y 128 Mb de memoria RAM. Estos nodos trabajadores, como ya se ha indicado como característica particular de los sistema Beowulf, no pueden realizar conexiones con el exterior del cluster de forma directa, sino que únicamente a través del nodo servidor.

Las comunicaciones en el sistema *Beowulf* se pueden realizar utilizando dos redes diferentes. La red de menores prestaciones consiste de una *Fast Ethernet* que tiene una velocidad de transmisión de mensajes de 100 Mbits por segundo. La red de mayores prestaciones consiste de una red *Myrinet* con una velocidad de transmisión de mensajes de hasta 1'2 Gbits por segundo. Cada una de estas redes utiliza dos switch especializados, y cada uno de estos switch gestiona hasta 16 nodos trabajadores. Por lo tanto, con la configuración actual el número de nodos trabajadores es de 32, y cada uno de ellos puede realizar las comunicaciones por cualquiera de las dos redes disponibles.

[2] Página Web http://anna.act.uji.es

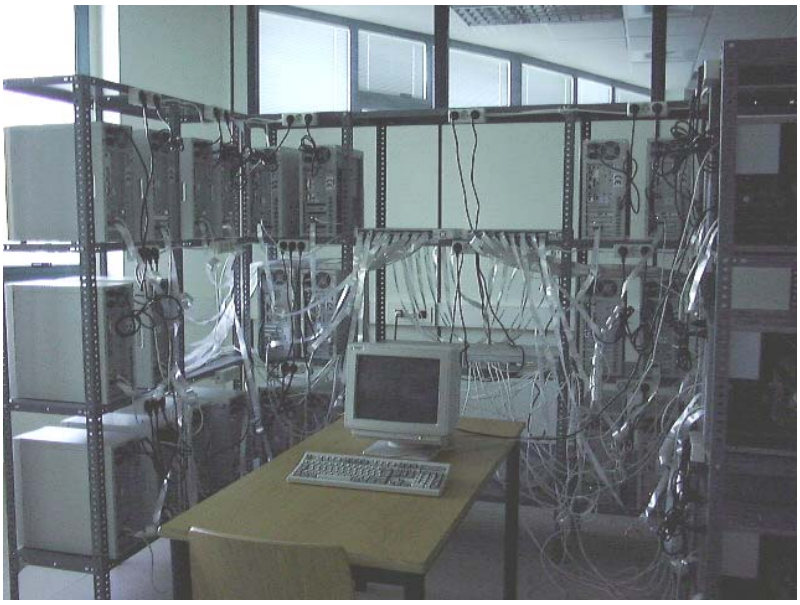


Fig 5-1. Fotografías del sistema *Beowulf* utilizado para implementar el sistema de gestión de tráfico *MASHGREEN DM.*

Las prestaciones de las comunicaciones se definen en términos de *latencia* y *ancho de banda*. El término *latencia* se define como el periodo de llegada del primer byte de un mensaje desde el nodo emisor al nodo receptor. El término *ancho de banda* se define como la cantidad de información que se puede transmitir después de haber superado la latencia del mensaje. La red *Fast Ethernet* presenta una latencia de 1 ms y un ancho de banda de 7 Mbytes por segundo, mientras que la red *Myrinet* presenta una latencia de 15

μs y un ancho de banda de 30 a 35 Mbytes por segundo. Estas medidas no son estancas puesto que tanto la latencia como el ancho de banda dependen del tamaño del mensaje.

Otra consideración a destacar del sistema utilizado es su coste económico. Cada uno de los switch necesarios para realizar las comunicaciones por al red *Myrinet* tiene un relativo alto coste, alrededor de 2 millones de pesetas. Este coste, resulta alto al compararlo con el coste de cada uno de los switch necesarios para realizar las comunicaciones por la red *Fast Ethernet* que ronda las 300.000 pesetas (1/6 del coste del switch para una *Myrinet*).

El coste global de este sistema *Beowulf*, considerando todo el hardware necesario, fue de 9,5 M pesetas en el año 1999. En el caso que no fuera precisa la red *Myrinet*, puesto que la aplicación a ejecutar en el sistema no sea fuertemente dependiente de la velocidad de las comunicaciones, este coste se vería reducido aproximadamente a la mitad del coste inicial. Y más aún si se considera la constante bajada de precios de los computadores Pentium II utilizados como nodos trabajadores.

5.1.3 Distribución Agentes Prototipo MASHGREEN DM en el Sistema Beowulf

Una de las primeras cuestiones referentes a como implementar el prototipo MASHGREEN DM consiste en plantearse cómo realizar las comunicaciones entre los distintos agentes del sistema. La solución adoptada consiste en utilizar el modelo de programación concurrente *linda* [Carreiro, 90]. El esquema genérico de un sistema *linda* se muestra en la figura 5.2.

Esta elección se debe a varios factores. El primero a tener en cuenta es su disponibilidad. Casi todos los lenguajes de programación, incluidos explícitamente los lenguajes de programación lógica basada en restricciones, cuentan con bibliotecas especializadas para la definición de las primitivas de comunicaciones *linda*. Además, estas primitivas proporcionan un alto grado de libertad con el que poder definir el formato y el modo en el que se transmite la información.

Mediante el uso de esta metodología de programación concurrente se ha destinado un nodo trabajador en exclusiva para cada agente, excepto en el caso del agente de interfaz, que al tener que exportar los gráficos que representan el estado del sistema en cada instant,e debe residir en el nodo servidor del sistema *Beowulf*. De esta forma se utilizan seis nodos trabajadores para realizar las tareas de preproceso, predicción, detección, coordinación, solución y soluciónncci. Adicionalmente, se utiliza un nodo trabajador para ejecutar el servidor del proceso linda encargado de la gestión del espacio de datos, así como el nodo servidor del sistema *Beowulf* para ejecutar el agente de interfaz de usuario. De esta forma se utilizan un total de siete nodos trabajadores junto con el nodo servidor. Todos estos nodos trabajadores pertenecen al mismo switch, por lo que el resto de nodos del cluster quedan libres para realizar otro tipo de tareas, como por ejemplo, la ampliación del sistema a un sistema en el que la ejecución de cada agente esté duplicada, permitiendo por lo tanto, la implementación de un sistema tolerante a fallos. Otro opción de expansión de las características del sistema consiste en la

implementación de agentes dinámicos. Tal expansión permitiría, mediante el uso de las mecanismos de sincronización del sistema adecuados, dar de alta a nuevos agentes y dar de baja a aquellos agentes que no estén proporcionando los datos adecuados.

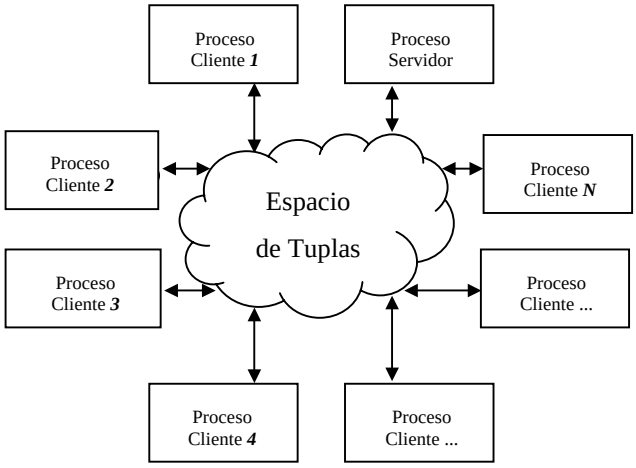


Fig 5-2. *Esquema de un sistema* linda. *Un proceso servidor se encarga de gestionar un espacio de datos común, el espacio de tuplas, a un conjunto de procesos clientes. La dirección asociada al espacio de datos debe ser pública con el propósito de que cualquier potencial cliente pueda conectarse al espacio de datos. En la dirección se incluye la dirección* Ethernet *del proceso servidor, por lo que se permite que cualquier cliente, sea o no residente en el mismo computador que el proceso servidor, pueda conectarse al espacio de datos mediante el uso de las primitivas de comunicaciones estándar* tcp/ip.

Las primitivas de comunicación utilizadas emplean exclusivamente la red *Fast Ethernet*. Esto es debido a que la programación de las primitivas de *linda* se realiza mediante el uso de *sockets*. Por lo tanto, con esta red se consigue un empleo instantáneo de estas primitivas, sin tener que realizar modificaciones en su implementación para que utilicen la red *Myrinet* de más altas prestaciones.

La elección de la red más lenta para programar la gestión de las comunicaciones no produce ningún descenso significativo en las prestaciones. Esto es debido tanto a las características del problema que se está controlando como a que ni el número de mensajes intercambiado en cualquier instante de la ejecución del prototipo *MASHGREEN DM* ni el tiempo empleado para transmitirlos son tareas cuyas prestaciones sean críticas para la consecución de sus objetivos operacionales. Las prestaciones de la red utilizada permite, por lo tanto, la correcta ejecución del sistema. Como consecuencia de este uso se consigue una mayor independencia de las herramientas de programación utilizadas para el desarrollo del sistema. Esto es, en el caso de haber precisado de la utilización de la red *Myrinet*, se tendría que haber hecho un uso intensivo del software de comunicaciones específicamente desarrollado para la programación de esta red, es decir, del uso de primitivas de comunicaciones según el

modelo *MPI* o *PVM*. En esta situación, se tendría que haber rediseñado todo la gestión de comunicaciones al cambiar de paradigma de programación distribuida.

5.2 Resultados de la Implementación

La evaluación del prototipo MASHGREEN DM se realizó mediante ejecuciones off-line en laboratorio. Los resultados e incidencias más relevantes acaecidas durante el periodo de evaluación y desarrollo se exponen a continuación.

5.2.1 Especificación Conjuntos de Datos Aplicados

La falta de datos reales procedentes de los sensores de tráfico de la zona urbana seleccionada determinó la necesidad de crear dos escenarios iniciales de tráfico, uno en el que se representa una situación congestionada del flujo y otro con un flujo ligero de vehículos. Cada uno de estos escenarios se combinó con otros dos posibles conjuntos de valores de coordinación para cada cruce del sistema. Por lo tanto, se dispuso de cuatro conjuntos de datos iniciales con los que ejecutar el prototipo.

Todos estos conjuntos comparten el mismo número de segmentos, cruces, entradas y salidas. También es común a todos los cruces el número de sus fases y la especificación del tráfico en cada una de ellas. Todos estos datos fueron obtenidos como resultado del proyecto "*Programa Director para los estudios de tráfico y transporte en Castellón*" financiado por el Excmo. Ayuntamiento de Castellón de la Plana y desarrollado por la Universidad Politécnica de Cataluña y la Universidad Jaume I de Castellón durante el año 1997[3] [Castellón, 97].

Los valores de coordinación establecidos para cada conjunto de datos cualitativos iniciales tienen el objetivo de intentar promocionar el tráfico por algunas de las vías principales del escenario seleccionado. Es decir, intentan establecer unos valores adecuados de *ondas verdes*. Estas preferencias de tráfico no son determinantes para la evolución del sistema, debido a que en el funcionamiento de los agentes de solución y de solución cci estas preferencias de tráfico no son valoradas de forma individual sino que ambos agentes buscan la mejor asignación de valores a los cruces que mejoren el estado de tráfico *global* en todo el escenario seleccionado.

En el apéndice A de esta memoria de tesis se muestran los valores seleccionados para cada uno de estos cuatro conjuntos de datos.

5.2.2 Incidencias Durante el Proceso de Implementación del Prototipo

Los agentes del prototipo MASHGREEN DM fueron implementados en el lenguaje de programación *prolog*[4], utilizando de forma intensa algunas de las bibliotecas de rutinas que permiten la utilización de primitivas de comunicaciones *linda*, un

[3] Este conjunto de datos se corresponde con los especificados en el apartado 3.11 de esta tesis.
[4] Concretamente *Sicstus prolog versión 3.7.1*, http://www.sics.se/sicstus.html

mecanismo de interpretación de comandos *tcl/tk* con el que desarrollar el interfaz gráfico y la programación lógica basada en restricciones de dominios finitos, *clpfd*.

La elección de este lenguaje de programación respondía a varios factores. El primero de ellos es que se trataba de desarrollar un *primer* prototipo de sistema de control de tráfico urbano, y la utilización de las características de este lenguaje permitía un tiempo de implementación previsiblemente bajo. A este factor ayudó de forma decisiva que el dominio de la aplicación no precisaba de un tratamiento de las tareas del sistema en tiempo real "*fuerte*", sino que bastará con que las tareas finalizasen o se interrumpiesen en algún instante cercano al cumplimiento de alguna situación excepcional durante la ejecución del sistema.

El segundo de los factores se debe a que uno de los objetivos iniciales de la tesis consistía en evaluar la utilización de herramientas de programación lógica paralela con restricciones para la implementación de prototipos. Este objetivo, como ya se indicó en apartado 3.11, no pudo realizarse de forma adecuada debido a la falta de mantenimiento de las escasas herramientas de programación lógica paralelas y su paulatino abandono como productos comerciales. No obstante, el empleo de las primitivas de comunicaciones *linda* y la búsqueda de un sistema de computadores adecuados, el sistema *Beowulf*, consiguieron reconvertir los objetivos iniciales de buscar acomodo para el prototipo MASHGREEN SM en un sistema multiprocesador con memoria compartida a otros objetivos más ambiciosos y bastante más acordes con respecto a las tendencias tanto en la programación de sistemas paralelos y distribuidos como en la disponibilidad de sistemas con los que realizar computación de altas prestaciones. Como resultado de este cambio de estrategia se desarrolló el prototipo MASHGREEN DM.

Una vez decididas las herramientas de programación a utilizar y la composición del sistema a desarrollar, tanto respecto a sus capacidades funcionales como a la definición de los algoritmos para llevarlas a cabo, se pasó a la fase de implementación del prototipo. Durante esta fase se definieron las estructuras de datos con las que tratar tanto los datos estáticos de la aplicación como sus datos dinámicos.

Con el objetivo de buscar aquellas estructuras de datos más eficientes se realizaron distintas pruebas comparando el uso de predicados dinámicos (assert/1, retract/1) respecto de los predicados dinámicos de la base de datos interna de prolog (recorded/3, erase/1, etc.). Éste último tipo de predicado dinámico fue introducido en las primeras versiones de prolog como medio de proporcionarle una forma eficiente de realizar operaciones sobre grandes cantidades de datos. Sin embargo, la introducción de algoritmos de indexación para los predicados dinámicos tradicionales convirtió en obsoletos a los predicados tipo recorded/3. No obstante, todos los sistemas prolog desarrollados siguieron contando con ambos tipos de predicados dinámicos. Resulta importante destacar que todos los manuales y obras de referencia de programación en prolog consultadas, promueven con firmeza el uso de los predicados dinámicos tradicionales respecto de los de la base interna.

Las primeras pruebas desarrolladas para determinar las estructuras de datos a utilizar consistieron en la programación de un sistema de pizarra muy sencillo utilizando ambos tipos de predicados dinámicos. Los resultados de su evaluación eran muy favorables a los no recomendados predicados dinámicos de la base interna respeto de los predicados dinámicos tradicionales (aproximadamente eran más eficientes en una razón de 2 a 1). El análisis de estos resultados conllevó a la utilización de los *obsoletos* predicados dinámicos para implementar algunas de las estructuras dinámicas con las que representar el comportamiento dinámico de los objetos del sistema (aún siendo conscientes que los expertos en programación de sistemas prolog desaconsejaban su uso y también *suponiendo* que tal indicación correspondía a un intento de primar la utilización de predicados estándar respecto de los no estándar). Como se analizará y explicará al final de este apartado esta elección fue bastante desafortunada.

El desarrollo del prototipo secuencial, el prototipo MASHGREEN SM, se realizó implementando únicamente los agentes de preproceso, predicción, detección, coordinación y control. El comportamiento del prototipo era aceptable, en el sentido que todas las acciones de los agentes de preproceso, predicción y detección se realizaban dentro de los intervalos temporales adecuados. Como ya se describió en el apartado 3.11 el principal problema de este sistema era su baja capacidad de interacción con el usuario, aún cuando el agente de interfaz se programase como un agente independiente del resto del sistema. Además, según se realizaban ejecuciones más largas, los tiempos necesarios para que los agentes de preproceso y de predicción finalizasen su ejecución se hacían más extensos, aunque siempre dentro de los límites temporales expuestos por el periodo de integración y del ciclo.

Con la experiencia aprendida en el desarrollo del prototipo MASHGREEN SM se empezó a desarrollar el prototipo MASHGREEN DM. Para este nuevo prototipo se siguieron utilizando las mismas estructuras de datos y algoritmos que los utilizados en el prototipo MASHGREEN SM. Una vez desarrollado se realizaron distintas pruebas para validar su comportamiento. Pero en este caso, los tiempos de ejecución para cada uno de los agentes del sistema para ejecutar sus tareas principales se hacían más extensos a medida que el prototipo MASHGREEN DM se ejecutaba durante mayores intervalos de tiempo. Como consecuencia, los agentes de preproceso y de predicción llegaban a abortar las ejecuciones de sus tareas principales antes de que éstas finalizasen.

Otro problema que se presentaba en el sistema es que las prestaciones que se obtenían eran peores cuanto más avanzado en el día se empezase a realizar la ejecución. Es decir, las prestaciones eran peores si el sistema arrancaba a las 16.00 que a las 8:00, y consecuentemente mucho peor si el sistema arrancaba a las 20:00.

Para analizar las causas de este comportamiento se utilizaron diferentes herramientas internas y externas al prototipo MASHGREEN DM. El análisis de los tiempos dedicados por cada agente durante la ejecución de su tarea primaria se realizó mediante los datos suministrados por su tarea secundaria de monitorización. Como complemento, y debido a que cada agente se ejecutaba en un computador distinto, se utilizaron

herramientas *linux* para analizar el estado de la memoria de cada computador de forma simultanea a la ejecución de cada agente.

En cada nueva ejecución de la tarea primaria en el agente de preproceso la solicitud de más memoria para poder realizar su ejecución era más exigente. Un análisis más minucioso de las subtareas que se ejecutaban durante la ejecución de la tarea primaria mostraba que este incremento en la necesidad de más memoria se realizaba cuando se accedía a estructuras de datos en las que utilizaban predicados dinámicos de la base de datos interna de prolog. Esta situación provocó que se volviese a escribir todas aquellas secciones de la programación de los agentes en las que se utilizasen estos predicados para emplear los predicados tradicionales.

El análisis de la memoria, después de realizados estos cambios, mostraba un comportamiento estable sin importar la duración del periodo de evaluación. Este problema, por lo tanto, quedaba solucionado. No obstante, el sistema seguía decayendo en sus prestaciones temporales. Ahora, al no ser abortadas las ejecuciones de los agentes de la aplicación, el sistema permitía realizar pruebas durante un mayor tiempo. En estas nuevas situaciones se presentaban nuevos problemas asociados a la pérdida de la referencia temporal e incluso a aparentes situaciones de bloqueo entre los agentes.

El análisis de estas nuevas situaciones se realizó mediante el estudio de los datos suministrados por las tareas secundarias de monitorización en los agentes de la aplicación. Como consecuencia de este estudio, se realizó de nuevo modificaciones en las estructuras de datos utilizadas. En este caso los cambios estaban enfocados a utilizar bases de datos externas, y las estructuras de datos adecuadas con los que, al trasladar datos pasados de la memoria principal a estas bases de datos, la búsqueda de los datos actuales se realizaría más rápido.

Efectivamente, con estas modificaciones las tareas de recogida de datos actuales con los que iniciar la ejecución de las tareas principales requería de un tiempo constante independientemente de la duración del periodo de prueba del prototipo.

No obstante, el prototipo seguía ralentizando su ejecución sin aparentes motivos. Se realizaron más cambios en las estructuras de datos intentando encontrar alguna que mejorase sustancialmente las prestaciones, pero esta fue una tarea en la que no se obtuvo éxito.

Una vez agotadas las posibilidades respecto a la utilización de distintas estructuras de datos, se pasó a analizar otros factores, como pudieran ser las posibles alteraciones o errores en las mediciones de los relojes internos de cada computador del sistema *Beowulf*, la ejecución de alguna tarea del sistema operativo *linux* que ralentizase "*de alguna manera*" las ejecuciones de los agentes (bien sea por fallos de funcionamiento en las comunicaciones, afectos laterales de otros computadores, privilegios de ejecución, o la ejecución de tareas periódicas de algún usuario en el mismo computador o en otro distinto). Pero todos estos análisis resultaron negativos.

La solución a este problema se encontró al analizar el comportamiento de la biblioteca *linda* proporcionado por el sistema *Sicstus prolog 3.7.1*. En la biblioteca *linda* se dispone básicamente de tres primitivas síncronas (asíncronas): la primitiva *rd/2* (*rd_noblock/2*) provoca la lectura de una tupla del espacio de tuplas, la primitiva *out/2* realiza la escritura de la tupla indicada en el espacio de tuplas, y la primitiva *in/2* (*in_noblock/2*) que provoca la lectura destructiva de una tupla en el espacio de tuplas.

El problema residía en que la primitiva *in/2* (*in_noblock/2*) no optimizaba el espacio de tuplas restante al eliminar una de las tuplas del espacio. Por lo tanto, al enviar muchos mensajes pequeños al espacio de tuplas, y al ser éstos recogidos mediante el uso de esta primitiva, los huecos dejados en el espacio de tuplas eran cada vez mayores, y como consecuencia, cuando algún agente realizaba un análisis del espacio de tuplas para buscar mensajes que le pudieran interesar precisaba de un mayor tiempo para recorrer las tuplas y los huecos del espacio de tuplas. Evidentemente, a medida que la duración del periodo de ejecución se hacía mayor, se realizaban más accesos de primitivas *in/2* (*in_noblock/2*) al espacio de tuplas, y como consecuencia, el tiempo necesario para llevarlo a cabo se iba incrementando de forma constante.

La solución a este problema consistió en agrupar en un único mensaje aquellos mensajes que algún agente enviaba casi simultáneamente, de forma que su análisis conjunto no produjese ninguna anomalía en la ejecución de los agentes de la aplicación. Esta reforma se realizó con los mensajes del agente de detección de problemas.

Esta deficiencia en la ejecución de las primitivas de comunicaciones *linda* era la causante de la gran degradación en las prestaciones que se producía en el prototipo MASHGREEN DM desde sus inicios. Por último, destacar que estas deficiencias no se deben al modelo de programación *linda* sino en la particular e ineficiente versión proporcionada por la herramienta de desarrollo.

Al modificar los algoritmos de los agentes de preproceso, predicción y de detección para que asumiesen estos cambios, se consiguió una ejecución absolutamente estable e independiente de la duración de la ejecución del prototipo MASHGREEN DM para todos sus agentes.

Además, el sistema dejó de ser sensible al instante inicial de su ejecución. Es decir, las prestaciones que se obtenían no dependían de la hora del día en el que se empezaba a ejecutar el prototipo. El motivo de este cambio también reside en la ineficiencia de la primitiva *in/2* (*in_noblock/2*). Todos los mensajes enviados al espacio de tuplas contienen al menos una, y normalmente varias, referencias temporales. Estas referencias temporales ocupan un mayor espacio de almacenamiento según representen horas del día más avanzadas. Al minimizar el uso de estos mensajes por el agente de detección también se eliminó esta anomalía en el sistema.

5.2.3 Comportamiento Individual y Colectivo de cada Agente en el Prototipo

El prototipo MASHGREEN DM se ha ejecutado durante periodos de una hora con los conjuntos de datos especificados en el apartado 5.2.1. Como resultados de estas

ejecuciones se han obtenido una serie de ficheros de datos que representan los tiempos empleados por cada uno de los agentes en la ejecución de algunas de sus tareas primarias y secundarias. En el caso de las tareas primarias este estudio se extiende al cálculo del tiempo empleado por cada una de sus subtareas más relevantes.

Todos estos datos se corresponden con los datos suministrados por cada tarea secundaria de monitorización en cada agente. Las unidades de medida temporales se corresponden con milisegundos de tiempo real transcurrido entre que se empieza a ejecutar una tarea hasta que finaliza. Esta medida proporciona una cota superior del tiempo virtual empleado para su ejecución, entendiendo por tiempo virtual el tiempo efectivo en el que el procesador estuvo ejecutando instrucciones pertenecientes a la tarea que se está monitorizando.

El análisis de la ejecución del agente de preproceso resulta fundamental en el sistema, puesto que el resto de agentes utilizan sus resultados como datos de entrada a partir de los cuales ejecutar sus tareas primarias. Por este motivo la tarea primaria del agente de preproceso se ha analizado evaluando sus secciones más importantes.

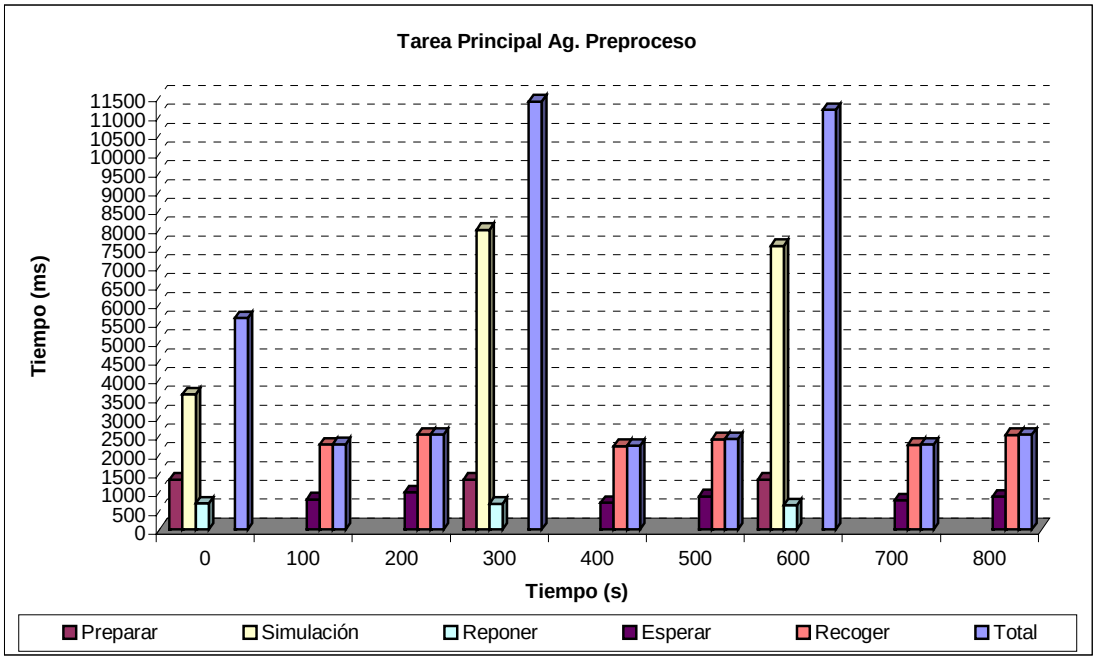


Fig 5-3. *Gráfica del comportamiento del agente de preproceso durante la ejecución de su tarea primaria.*

La tarea primaria del agente de preproceso puede ejecutarse de dos formas diferentes. Una de las formas implica la ejecución de un proceso de simulación cualitativa utilizando los datos que pertenecen a su espacio de datos. Esta modalidad se utiliza cuando se inicia la ejecución de un nuevo periodo de integración o después de haber detectado la presencia de acciones de control externas en el sistema. En cualquier otra situación la tarea primaria del agente de preproceso envía un mensaje de solicitud de los datos de simulación al agente de predicción. En el caso que estos datos ya hubiesen sido

generados por el agente de predicción, éste agente recoge esos datos y se los envía al agente de preproceso, quien los almacena en las estructuras de datos dinámicas asociadas a los datos actuales y los envía al agente de interfaz para que los muestre en la pantalla. En el caso que el agente de predicción no hubiese terminado de calcular esos datos el agente de preproceso le envía un mensaje indicándole esa circunstancia, por lo que al agente de preproceso no le queda otra alternativa que realizar la simulación con sus propios datos.

En la figura 5-3 se observa que la tarea primaria del agente de preproceso se ejecuta de la primera forma comentada en los inicios de cada periodo de integración (es decir, cada 300 segundos, en los instantes 0, 300, 600, etc.). Esta tarea se compone de tres partes: la preparación de la simulación (búsqueda de los datos iniciales para cada objeto del sistema y cálculo de su posterior evolución), el proceso de simulación cualitativa y la reposición de los datos obtenidos en las estructuras de datos dinámicas asociadas al presente periodo de integración. Como se observa en la figura, estas ejecuciones se comportan de forma constante durante el periodo de estudio. La componente que precisa de un mayor tiempo de ejecución es la que ejecuta el proceso de simulación cualitativa, mientras que las otras dos tienen un coste mucho menor.

El resto de ejecuciones de la tarea primaria del agente de preproceso reciben del agente de predicción los datos que necesitan (se corresponden con los instantes 100, 200, 400, 500, 700, 800, etc.). Por ello, estas tareas se componen de dos partes. La primera es una tarea de petición de los datos y de espera hasta su recepción. La segunda consiste en la reposición de los datos recibidos en su posición final. Ambas partes tienen un tiempo de ejecución aproximadamente constante durante todos los periodos de integración evaluados.

Como se observa en la figura 5-3, la tarea que implica la simulación de los datos tiene un coste de aproximadamente 9 a 1 con respecto al tiempo total que tiene que simular, es decir, los 100 segundos de simulación del sistema los ejecuta en unos 11 segundos. Este resultado es coherente con las expectativas obtenidas al calcular el coste teórico del proceso de simulación cualitativa (apartado 2.4.1.8).

En aquellas situaciones en los que no se precisa ejecutar el proceso de simulación cualitativa se obtienen mejores prestaciones, del orden de 30 a 1. Esto es debido al tipo de comunicación definida entre los agentes de preproceso y de predicción y a la adecuación de las estructuras de datos utilizadas para almacenar los datos dinámicos de cada objeto del sistema. En la gráfica se observa también que el tiempo de espera por los datos es bastante reducido (alrededor de segundo o segundo y medio).

Al ejecutar este agente con los cuatro conjuntos de datos iniciales disponibles se observa que este comportamiento temporal se mantiene constante independientemente de la duración del periodo de ejecución. La figura 5-4 muestra este comportamiento en forma de sierra, de forma que los picos se corresponden con ejecuciones del proceso de simulación en los instantes de comienzo de cada periodo de integración y los valles representan las ejecuciones asociadas a recepciones de datos ya calculados por el agente de predicción.

Las pequeñas diferencias entre unos picos y otros son asociables a cuestiones temporales de la carga del computador en el que se está ejecutando el agente de preproceso. Como se observa en la figura, según sea el conjunto de datos iniciales aplicados el comportamiento temporal puede incrementarse de forma ligera pero uniforme con respecto a otros conjuntos de datos iniciales para los que la simulación es más rápida.

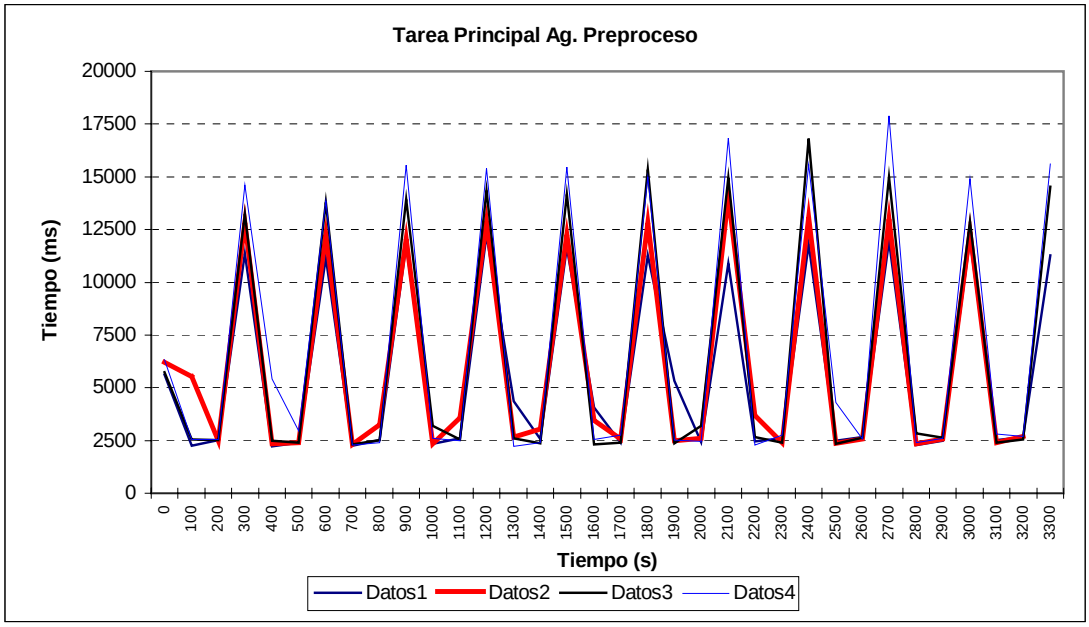


Fig 5-4. *Ejecución de la tarea primaria del agente de preproceso durante 3300 segundos con cuatro conjuntos iniciales diferentes de datos.*

Otra de las tareas monitorizadas del agente de preproceso es la tarea secundaria *DameDatosPreproceso*. Esta tarea está asociada a responder a las peticiones de datos realizadas por cualquiera de los agentes en el sistema. Esta tarea se supervisó durante periodos de ejecución de 3300 segundos con los cuatro conjuntos iniciales diferentes de datos disponibles. El resultado se muestra en la figura 5-5.

En esta figura se observa que sólo tres agentes solicitan datos al agente de preproceso. Estos son los agentes de predicción, solución repartos y solución cci. Estas peticiones se realizan al comienzo de cada nuevo periodo de integración de datos en el sistema. Todas las peticiones se satisfacen una vez concluido el proceso de determinación de los datos actuales de simulación para el primer periodo de simulación del nuevo periodo de integración de datos.

De todas las peticiones realizadas, las asociadas al agente de predicción tienen un coste ligeramente superior al del resto de agentes. Esto es debido a la forma en que se organizan los datos dinámicos en el agente de preproceso. No obstante, todas las peticiones son satisfechas con bastante celeridad (menos de 0.2 segundos).

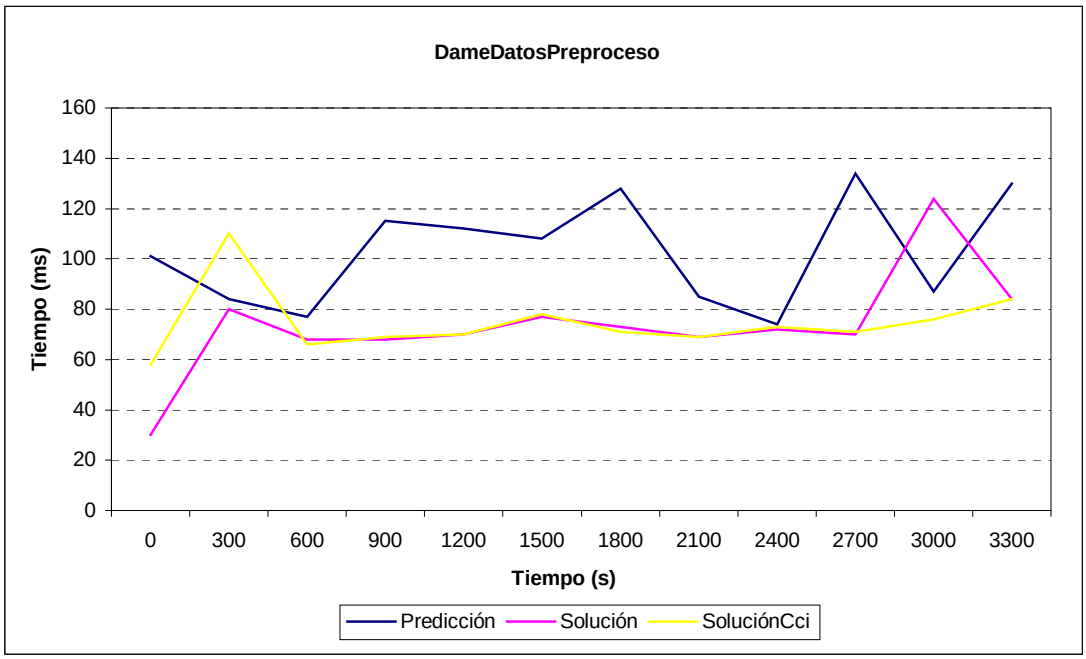


Fig 5-5. *Ejecución de la tarea secundaria DameDatosPreproceso como respuesta a la recepción de un mensaje de petición de datos procedente de alguno de los agentes del sistema.*

Todas las pruebas realizadas se han ejecutado utilizando el modo de funcionamiento automático para los repartos. La tarea asociada a aplicar las soluciones proporcionadas por el agente de solución repartos se ejecuta como una componente más del proceso de simulación cualitativa. El análisis del coste asociado a su ejecución se muestra en la figura 5-6.

Como se observa en esa figura la evaluación de su comportamiento respecto a los cuatro conjuntos iniciales de datos disponibles aporta unos resultados que siempre están acotados de forma superior por el valor 1 segundo, lo que supone un coste casi despreciable.

Otra de las tareas que se monitorizó en el prototipo MASHGREEN DM fue la tarea secundaria *Iniciar_agentes*. Esta tarea está compuesta de dos componentes que se ejecutan de forma secuencial. La primera componente especifica la secuencia de acciones a realizar para componer la situación inicial del agente. Dentro de esta componente se ejecutan tareas asociadas a la organización de los datos estáticos y dinámicos del espacio de datos.

La segunda componente se encarga de realizar la sincronización con el resto de agentes del sistema mediante un mecanismo de sincronización por barrera. Como resultado, el agente obtiene la referencia temporal del agente de coordinación del sistema. La figura 5-7 muestra el comportamiento de esta tarea para todos los agentes del sistema respecto de los cuatro conjuntos iniciales de datos.

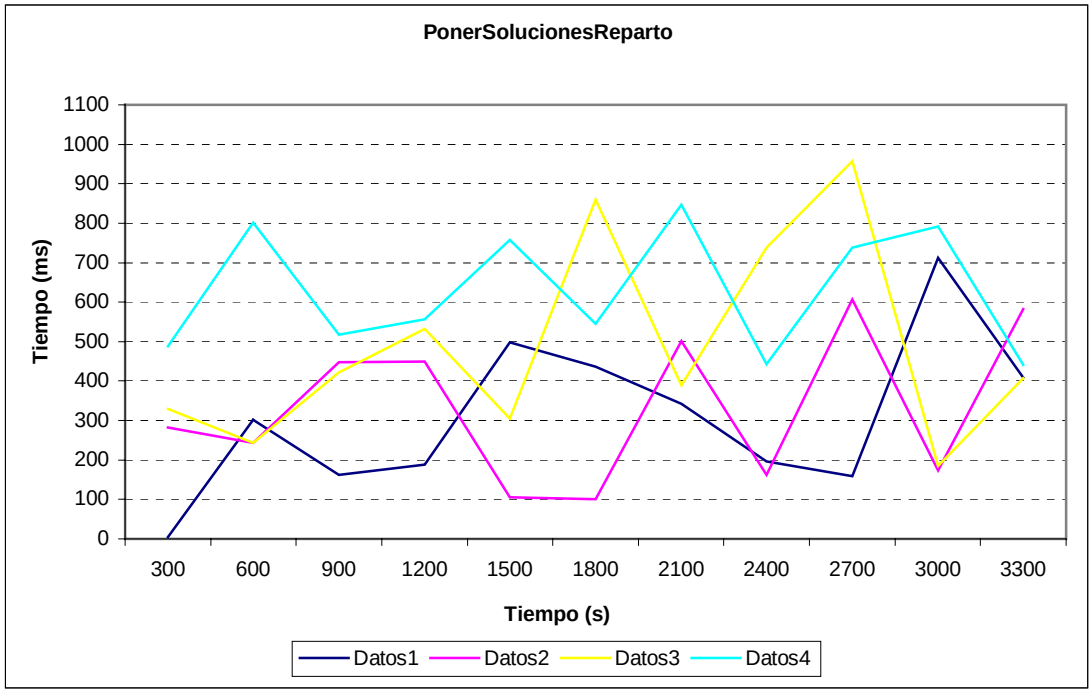


Fig 5-6. *Resultados de la ejecución de la componente PonerSolucionesReparto con cuatro conjuntos diferentes de datos iniciales.*

Para cada uno de los agentes se muestra el tiempo empleado para ejecutar de forma completa la tarea *iniciar_agentes* junto con el tiempo empleado para ejecutar la sincronización por barrera. Las diferencias apreciables entre los agentes responden a la forma en que éstos son inicialmente ejecutados. El programa *lanzador* del prototipo MASHGREEN DM ejecuta secuencialmente cada uno de los agentes en el siguiente orden: coordinación, interfaz, detector, solución, soluciónccci, predicción y preproceso. Esta secuencia hace que los primeros agentes que se ejecuten tengan que realizar un mayor tiempo de espera para realizar la sincronización por barrera.

Otro factor que incide en este comportamiento es que la ejecución de estos agentes se realiza de forma interpretada, no compilada, por lo que también se han de considerar los accesos a los ficheros de configuración del sistema, a la compilación de las bibliotecas de rutinas utilizadas por cada agente y a la ejecución del interprete de prolog.

En la gráfica se observa que, independientemente del conjunto de datos iniciales a aplicar, el comportamiento de los agentes con respecto a la sincronización por barrera es estable. La diferencia de alturas de cada barra que ocupa una posición impar respecto de la siguiente barra representa el tiempo empleado para realizar la iniciación del espacio de datos en cada agente.

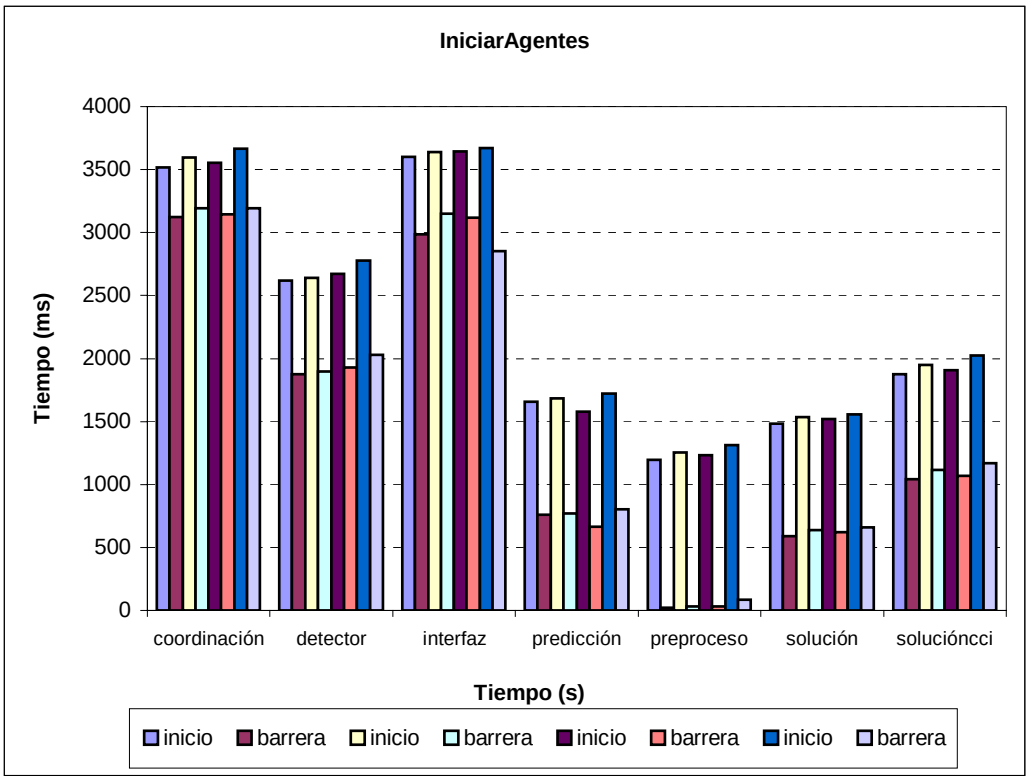


Fig 5-7. *Resultados de la ejecución de la tarea IniciarAgentes con los cuatro conjuntos de datos disponibles para agente del prototipo.*

Esta tarea, como ya se ha indicado, sólo se ejecuta una vez durante toda la ejecución del prototipo. En el caso de ampliar el sistema con el mecanismo propuesto de permitir las altas y bajas dinámicas de agentes de la aplicación en el sistema, el método de la sincronización por barrera sería la herramienta clave para realizar su implementación.

El comportamiento del agente de predicción también resulta estable tanto para la ejecución de su tarea primaria como para cualquiera de sus tareas secundarias. La figura 5-8 muestra el desglose de la ejecución de su tarea primaria.

El tiempo asignado como cota temporal superior para el sistema fue de 800 segundos a contar desde el momento de integración de datos en el sistema. Los primeros 100 segundos son calculados por el agente de preproceso. El agente de predicción, por lo tanto, realiza un total de 7 procesos de simulación, cada uno de ellos de 100 segundos. Esta tarea de simulación, tal como pasaba en el agente de preproceso, se compone de tres subtareas: la preparación de todos los datos dinámicos y de control necesarios para realizar la simulación, el propio proceso de simulación y el almacenamiento de los resultados de la simulación en las estructuras de datos adecuadas.

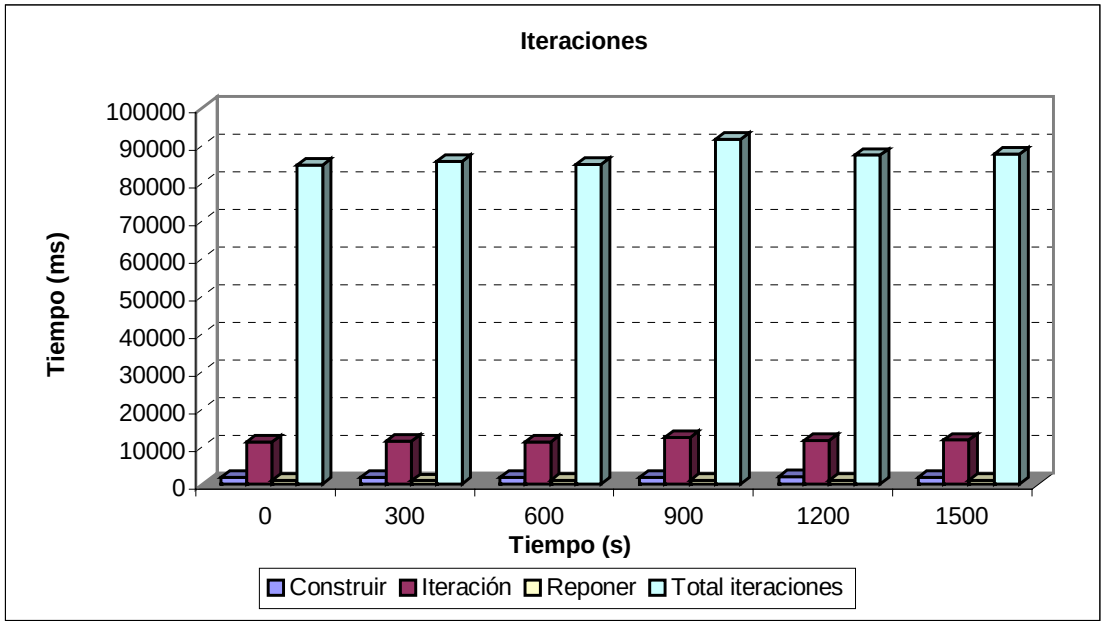


Fig 5-8. *Gráfica del comportamiento de la tarea primaria del agente de predicción.*

En la figura 5-8 se observa que el comportamiento temporal de la ejecución de la tarea primaria es razonablemente estable para cada una de sus componentes. La barra correspondiente al total de iteraciones es aproximadamente la suma de las barras de construcción, iteración y reponer multiplicado por siete veces que se repite ese proceso. Las barras asociadas a las componentes de construcción y de reponer datos mantienen una misma altura durante todo el periodo de evaluación, lo que indica que las estructuras de datos utilizadas para almacenar los datos dinámicos y de control tienen unas prestaciones uniformes.

El análisis del comportamiento del agente de predicción con respecto del tiempo disponible para su ejecución (que es el periodo de integración de datos de sensores en el sistema, definido como 300 segundos) muestra que, en ausencia de la presencia de acciones de control externas, el agente de predicción permanece en estado ocioso aproximadamente dos tercios del tiempo total disponible. Este dato proporciona una idea del tamaño de la ventana temporal de futuro que este agente sería capaz de calcular. La figura 5-9 muestra esta relación.

En la suma total del tiempo empleado por el agente de predicción para realizar su tarea se incluyen, además de los tiempos dedicados al cálculo de cada simulación de 100 segundos, los tiempos de espera entre que el agente de predicción solicita los datos iniciales al agente de preproceso hasta que éstos son recibidos, junto con el tiempo empleado en desempaquetar estos mensajes y disponer su contenido en las estructuras de datos adecuadas.

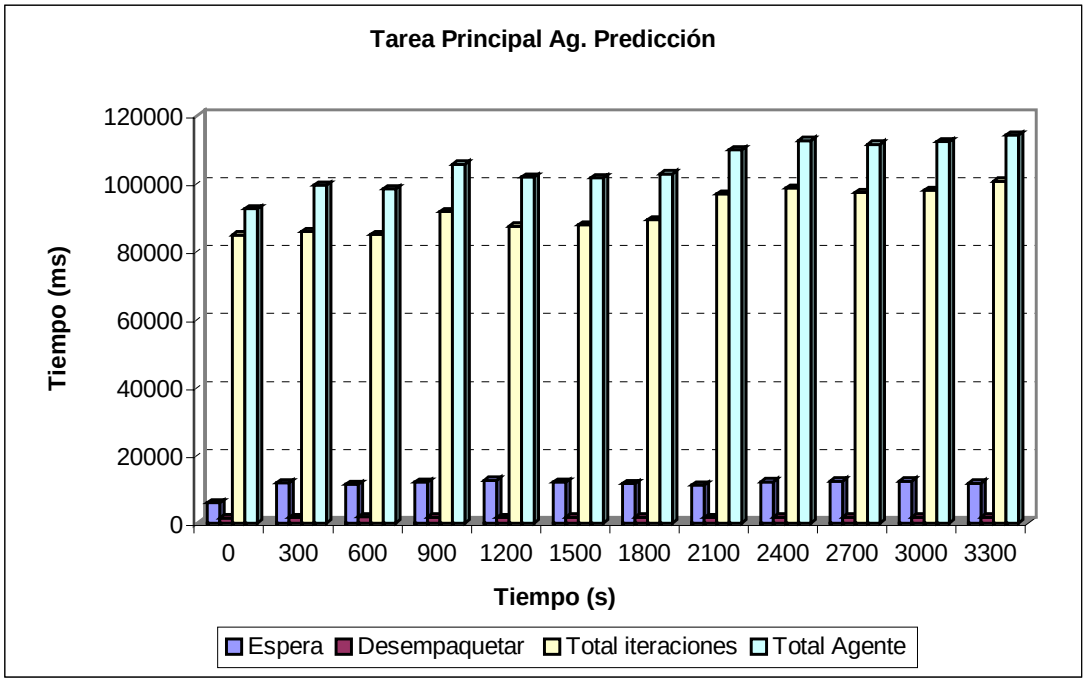


Fig 5-9. *Análisis del tiempo utilizado por el agente de predicción para realizar su tarea primaria respecto del tiempo total disponible.*

Como ya se ha indicado, hay una relación entre los datos calculados por el agente de preproceso y los del agente de predicción. En la ausencia de acciones de control externas, algunos de los datos calculados por el agente de predicción asociados al actual periodo de integración estarán calculados antes de que éstos sean necesarios para el agente de preproceso. Por lo tanto, antes de empezar a calcularlos, el agente de preproceso envía un mensaje al agente de predicción solicitándole esos datos. En el caso de que esos datos estuvieran ya disponibles, el agente de predicción se los envía. Si los datos no estuviesen todos disponibles entonces el agente de predicción le comunica esta circunstancia y el agente de preproceso comienza a ejecutar su proceso de simulación.

La tarea secundaria del agente de predicción que realiza esta gestión se denomina *DarDatosPred*. Para conseguir que el tiempo de espera por respuesta del agente de preproceso sea la menor posible, el agente de predicción comprueba al finalizar cada uno de los ciclos que componen el proceso de predicción cualitativa si se ha recibido algún mensaje de petición de datos. En caso afirmativo, el agente de predicción abandona momentáneamente la ejecución del proceso de predicción cualitativa para atender a esta petición de datos. Una vez satisfecha, el agente de predicción retorna a la ejecución del proceso de predicción cualitativa.

Esta tarea se ha monitorizado durante sucesivas pruebas con los cuatro conjuntos de datos iniciales y los resultados obtenidos se muestran en la gráfico 5-10.

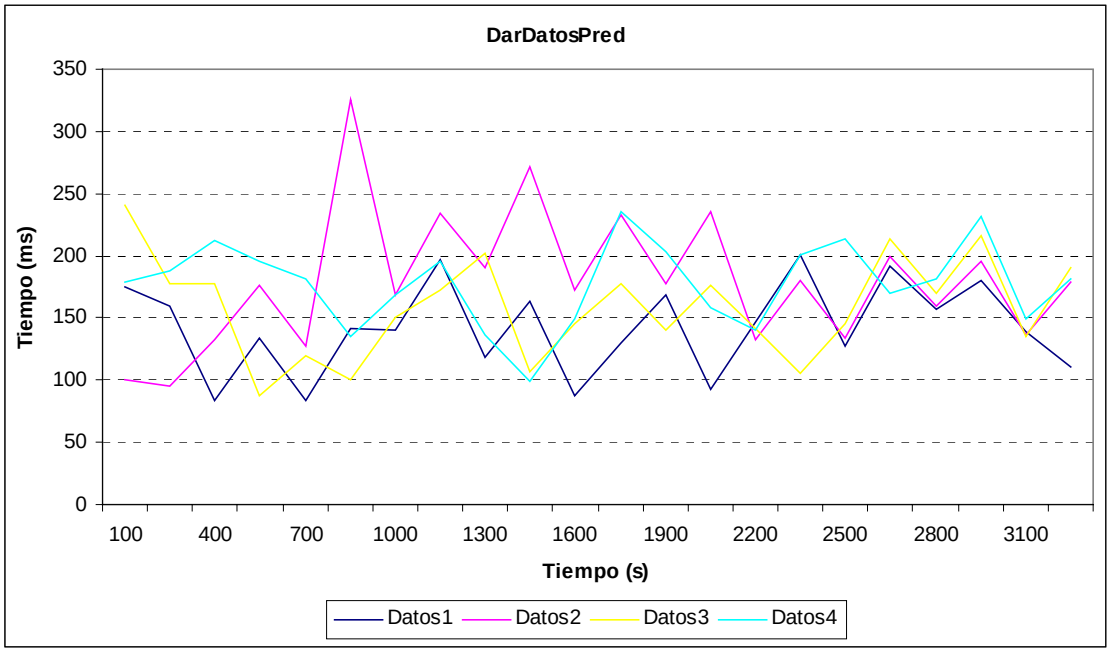


Fig 5-10. *Gráfica del comportamiento de la tarea secundaria* DarDatosPred *durante la ejecución del agente de predicción.*

Como se observa en la figura 5-10 los tiempos dedicados a la búsqueda y envío de los datos solicitados por el agente de preproceso se mantienen en unos intervalos temporales razonablemente estables (menos de medio segundo) durante todo el periodo de prueba con los cuatro conjuntos de datos.

El análisis del comportamiento del agente de coordinación en el sistema se muestra en la figura 5-11. Como ya se ha indicado previamente, la principal función del agente de coordinación consiste en realizar el proceso de sincronización inicial con el que determinar el tiempo virtual para todos los agentes del sistema. Otras de las tareas que realiza son la recogida y ordenación de los datos de monitorización enviados por el resto de agentes del sistema, y el envío de las órdenes de ejecución para las tareas primarias de los agentes de la aplicación según sean sus características operativas.

Todas estas funciones podrían ser realizadas por cada agente del sistema. Por ejemplo, la tarea de sincronización por barrera podría encargarse al agente de interfaz y, al ser este agente el medio de comunicación con el usuario, también podría realizar el proceso dinámico de dar de alta o de baja a determinados agentes. Las tareas de monitorización se podrían realizar de forma individual en cada agente. Por último, como todos los agentes comparten la misma referencia temporal común, entonces no sería necesario el envío de ningún mensaje de ejecución de las tareas primarias para cada agente. Los propios agentes decidirían en función del tiempo virtual común cuando deben ejecutar sus tareas primarias.

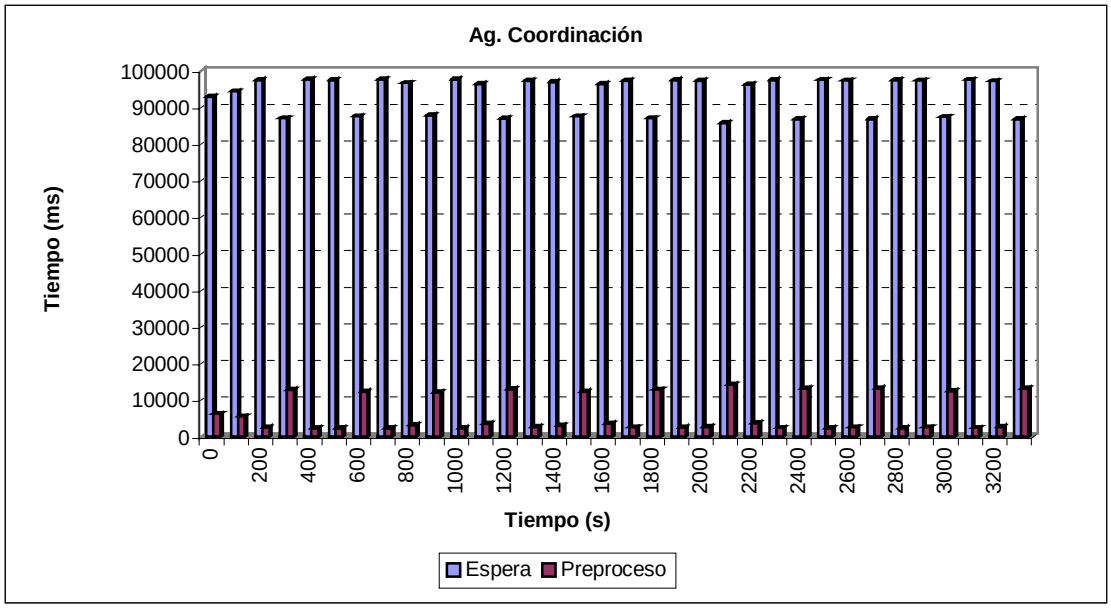


Fig 5-11. *Gráfica del tiempo empleado por el agente de coordinación cada periodo de 100 segundos.*

El agente de coordinación, aún no siendo estrictamente necesario, se utiliza debido a que el prototipo MASHGREEN DM es un primer prototipo. Como tal, resulta más sencillo apoyarse en el desarrollo de un sistema débilmente centralizado que en uno distribuido, puesto que el análisis de su comportamiento se puede realizar de una forma más sencilla.

Como se observa en la figura 5-11 el agente de coordinación pasa la mayor parte de su tiempo en estado ocioso. Durante este tiempo su única tarea consiste en recoger datos de monitorización. El resto del tiempo lo emplea en esperar que el agente de preproceso finalice la ejecución de su tarea principal.

Este comportamiento es una herencia del proceso de construcción del prototipo desde los primeros agentes desarrollados en el prototipo MASHGREEN SM. En este sistema las tareas a ejecutar se clasificaban y ejecutaban de forma jerárquica, de forma que una vez finalizadas todas las ejecuciones pendientes de una categoría se pasaban a ejecutar las peticiones de la categoría inmediatamente inferior. La tarea de preproceso tenía la máxima prioridad entre todas. En el prototipo MASHGREEN DM se mantuvo este único periodo de espera por la finalización del agente de preproceso en las primeras etapas de su desarrollo. Su permanencia permitía analizar las posibles incidencias en las prestaciones del sistema según se iban añadiendo nuevos agentes (apartado 5.2.2)

El agente de detección también se ha monitorizado para comprobar sus prestaciones. En este agente se han medido los tiempos empleados para analizar los mensajes asociados a problemas del actual periodo de simulación y a los problemas asociados a simulaciones realizadas por el agente de predicción. El resultado de este análisis se muestra en la figura 5-12.

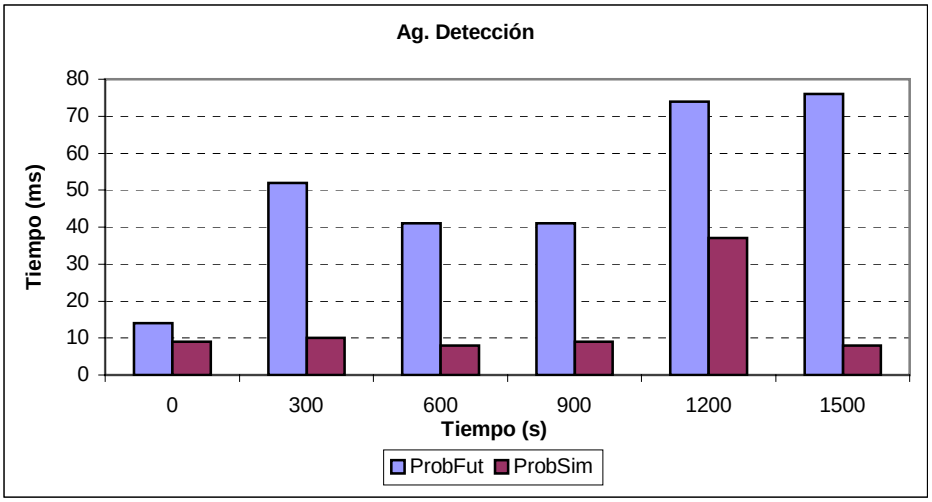


Fig 5-12. *Gráfica del tiempo empleado por el agente de detección de problemas para analizar los mensajes de problemas enviados por los agentes de preproceso (ProbSim) y de predicción (ProbFut).*

Todos los problemas elementales y temporales determinados por los agentes de preproceso y de predicción durante un periodo de 100 segundos se agrupan en un único mensaje que es enviado al agente de detección. A partir de los datos almacenados en estos mensajes el agente de detección puede deducir las situaciones problemáticas espaciales/temporales presentes en el dominio del sistema. Los mensajes recibidos durante cada periodo de integración se han agrupado en una única barra que indica su agente emisor. Como se observa del gráfico, estas tareas de análisis precisan de tiempos de ejecución muy cortos (menos de 0.1 segundos)

El agente de solución repartos es, junto con el agente de preproceso, un agente cuya tarea es determinante en el prototipo MASHGREEN DM. Por este motivo, además de analizar su comportamiento junto con el resto de agentes del sistema, también se han analizado sus prestaciones cuando se ejecuta de forma aislada. Los resultados de este estudio se clasifican según el tiempo dedicado a la búsqueda de soluciones, el conjunto de datos iniciales y la heurística de selección de cruce utilizada cuando se decide cambiar de vector de reparto para alguno de los cruces.

El tiempo dedicado para buscar soluciones en cada una de las pruebas coincide con el tamaño del intervalo temporal de simulación en el que determinar la mejor asignación de vectores de reparto a cruces. De esta forma, para determinar los mejores repartos para un intervalo de simulación de 100 segundos se han dedicado 100 segundos de tiempo real de ejecución.

Los conjuntos de datos iniciales utilizados se clasifican según los valores de las coordinaciones para cada cruce y los valores cualitativos iniciales para cada segmento del sistema. Los conjuntos *D1* y *D2* comparten los mismos valores de coordinación, al igual que sucede con los conjuntos *D3* y *D4*. Los conjuntos *D1* y *D3* comparten los mismos valores cualitativos iniciales que se corresponde con una situación global del

tráfico ligero. Los conjuntos *D2* y *D4* comparten también los mismos valores cualitativos iniciales, pero en este caso la situación global del tráfico que representan se corresponde con un tráfico ligeramente congestionado.

El número total de hojas, para cualquiera de las pruebas, es de 8^{22}. Por lo tanto, el proceso de búsqueda implica la poda de un gran número de caminos previsiblemente peores que los recorridos durante el proceso de la mejor solución obtenida hasta el momento.

La primera de las heurísticas aplicadas se corresponde con la elección del vector de reparto más cercano al vector actual con la finalidad de minimizar los posibles efectos producidos como consecuencia de su aplicación.

	D1T100	**D2T100**	**D3T100**	**D4T100**
Hojas	1	213	1	1
Nodos Podados	3823	0	2459	3904
Nodos Pasados	545	242	344	619
Tiempo 1ª	6	6	6	6
Tiempo Resto	94	94	94	94
1ª Solucion	162-2	1040-13	140-2	941-9
Mejor Solucion	162-2	1040-13	140-2	941-9
Numero Cruces	5	9	5	6

	D1T200	**D2T200**	**D3T200**	**D4T200**
Hojas	35	1	1	2
Nodos Podados	0	274	43	63
Nodos Pasados	806	313	834	614
Tiempo 1ª	11	12	11	12
Tiempo Resto	189	188	189	188
1ª Solucion	850-8	2376-17	823-7	2274-13
Mejor Solucion	848-8	2376-17	823-7	2254-13
Numero Cruces	4	9	4	6

	D1T300	**D2T300**	**D3T300**	**D4T300**
Hojas	6	19	1	5
Nodos Podados	21	85	40	25
Nodos Pasados	1060	910	1195	1083
Tiempo 1ª	17	18	16	18
Tiempo Resto	283	282	284	282
1ª Solucion	1840-9	4150-20	1653-8	4110-15
Mejor Solucion	1788-9	3831-18	1653-8	4083-15
Numero Cruces	4	8	4	5

La segunda de las heurísticas utilizadas se corresponde con la elección del vector de reparto más alejado del valor actual para dar más prioridad de uso a la dirección principal del cruce.

	D1T100	D2T100	D3T100	D4T100
Hojas	1	220	1	1
Nodos Podados	3823	0	2451	3928
Nodos Pasados	545	250	346	622
Tiempo 1ª	6	6	5	6
Tiempo Resto	94	94	95	94
1ª Solucion	162-2	1040-13	140-2	941-9
Mejor Solucion	162-2	1040-13	140-2	941-9
Numero Cruces	5	9	5	6

	D1T200	D2T200	D3T200	D4T200
Hojas	35	1	1	1
Nodos Podados	0	293	43	64
Nodos Pasados	809	334	831	603
Tiempo 1ª	11	12	11	12
Tiempo Resto	189	188	189	188
1ª Solucion	850-8	2227-14	825-6	2274-12
Mejor Solucion	848-8	2227-14	825-6	2274-12
Numero Cruces	4	9	4	6

	D1T300	D2T300	D3T300	D4T300
Hojas	6	1	1	6
Nodos Podados	21	422	42	21
Nodos Pasados	1057	481	1205	1067
Tiempo 1ª	18	18	16	17
Tiempo Resto	282	282	284	283
1ª Solucion	1840-9	3777-17	1620-8	4077-16
Mejor Solucion	1776-9	3777-17	1620-8	4051-16
Numero Cruces	4	9	4	5

En estas tablas el valor de *hojas* indica el número de hojas (soluciones) alcanzadas. El valor *nodos podados* indica el número de podas realizadas durante la ejecución del agente. El valor *nodos pasados* referencia a aquellos nodos a los que al aplicar el criterio de selección de que reparto aplicar deciden mantener el vector de reparto actual. También se indica en la tabla el tiempo que le cuesta al método alcanzar la primera solución (*Tiempo 1ª*) y el tiempo dedicado a calcular el resto de soluciones (*Tiempo Resto*). Las soluciones calculadas por el método se componen de dos indicadores *NP-IP*. El indicador *NP* representa el valor de la acumulación de la evolución de los situaciones problemáticas elementales y temporales detectadas durante el intervalo de simulación ponderadas por el tiempo en que éstas han estado presentes. El indicador *IP*

representa la tendencia a la congestión durante todo el intervalo. El valor *1ª Solución* representa la primera solución alcanzada (es decir, la primera hoja) mientras que el valor de *Mejor Solución* representa la solución obtenida con menor valor del indicador *NP*. Por último, el valor de *Número cruces* representa la cantidad de cruces que cambian de vector de reparto a aplicar como consecuencia de seleccionar la *Mejor Solución* obtenida como resultado del proceso.

La tercera heurística aplicada se corresponde con la elección del vector de reparto más alejado del vector actual con la finalidad de dar menor prioridad de uso a la dirección principal del cruce.

	D1T100	**D2T100**	**D3T100**	**D4T100**
Hojas	1	206	1	1
Nodos Podados	3807	0	2475	3648
Nodos Pasados	543	234	347	579
Tiempo 1ª	6	6	6	6
Tiempo Resto	94	94	94	94
1ª Solucion	162-2	1040-13	140-1	941-9
Mejor Solucion	162-2	1040-13	140-1	941-9
Numero Cruces	5	8	6	6

	D1T200	**D2T200**	**D3T200**	**D4T200**
Hojas	35	1	1	1
Nodos Podados	0	257	43	64
Nodos Pasados	790	294	831	607
Tiempo 1ª	11	11	10	11
Tiempo Resto	189	189	190	189
1ª Solucion	850-8	2254-12	816-6	2303-11
Mejor Solucion	850-8	2254-12	816-6	2303-11
Numero Cruces	4	8	4	6

	D1T300	**D2T300**	**D3T300**	**D4T300**
Hojas	2	1	1	2
Nodos Podados	25	370	41	36
Nodos Pasados	1044	422	1201	1126
Tiempo 1ª	18	18	16	17
Tiempo Resto	282	282	284	283
1ª Solucion	1840-9	3723-15	1640-8	4125-14
Mejor Solucion	1796-10	3723-15	1640-8	3975-13
Numero Cruces	4	9	4	5

Al analizar estas tablas de forma conjunta se puede observar una serie de características bastante acordes con los comportamientos que cabría esperar del algoritmo utilizado.

El tiempo que se tarda en alcanzar la primera hoja es constante con respecto al valor de la longitud del intervalo de simulación propuesto. Además, este valor es independiente de la heurística utilizada. En aquellos conjuntos de datos con menor duración del intervalo de simulación la mejor solución encontrada coincide con la primera solución calculada (casos de ejecuciones para 100 y 200 segundos).

Estos resultados son coherentes puesto que las decisiones sobre cambios en los valores de los cruces se realizan durante el primer ciclo de simulación para todos los cruces del sistema y para los siguientes ciclos estos valores no se pueden modificar. Para longitudes pequeñas del intervalo de simulación muy pocos cruces alcanzan a tener más de un ciclo completo de ejecución y, por lo tanto, el efecto de las modificaciones realizadas en cada cruce no llegan a evaluarse durante más de un ciclo completo. Las soluciones obtenidas difieren muy poco de la realizada durante la generación de la primera de las soluciones. En el caso de las soluciones calculadas durante intervalos de duración 300 segundos la diferencia entre el valor de la primera solución y el de la mejor solución aumenta.

Otra característica a destacar es la gran incidencia que tienen los conjuntos de datos cualitativos iniciales para el sistema. Si estos datos representan situaciones de congestión ligera (conjuntos *D2* y *D4*) entonces las soluciones proporcionadas son de una mayor magnitud que las proporcionadas cuando al sistema se le inicia con datos cualitativos que representan situaciones de tráfico ligero (conjuntos *D1* y *D3*).

Más aún, la mejor solución aportada mejora en mayor medida el valor de la primera solución para aquellos conjuntos de datos que representan una situación inicial de tráfico ligeramente congestionado. Esto es razonable, puesto que las modificaciones propuestas tienen más situaciones problemáticas a las que ser sometidas que en el caso de que el número de estas situaciones fuese más reducido.

No obstante, con estos resultados no resulta sencillo el poder determinar cual de las heurísticas proporciona los mejores resultados. Para intentar responder a esta cuestión se ha vuelto a ejecutar el agente de solución durante un mayor intervalo de simulación. Los resultados obtenidos para cada conjunto de datos y cada heurística se muestran en la siguiente tabla.

Heurística 1	**D1T600**	**D2T600**	**D3T600**	**D4T600**
Hojas	2	2	1	1
Nodos Podados	13	31	29	48
Nodos Pasados	1268	1408	1541	1524
Tiempo 1ª	58	57	55	56
Tiempo Resto	542	543	545	544
1ª Solución	5593-14	9286-20	4559-10	8579-16
Mejor Solución	5378-11	8443-18	4559-10	8579-16
Número Cruces	4	8	4	5

Heurística 2	**D1T600**	**D2T600**	**D3T600**	**D4T600**
Hojas	4	2	1	2
Nodos Podados	16	521	72	55
Nodos Pasados	1632	1232	2185	1931
Tiempo 1ª	45	41	42	43
Tiempo Resto	555	559	558	557
1ª Solución	5593-14	8092-18	4360-10	8658-15
Mejor Solución	4907-10	7323-14	4360-10	7608-12
Número Cruces	4	9	5	4

Heurística 3	**D1T600**	**D2T600**	**D3T600**	**D4T600**
Hojas	3	1	1	1
Nodos Podados	16	674	44	87
Nodos Pasados	1621	770	2040	2000
Tiempo 1ª	44	41	40	43
Tiempo Resto	556	559	560	557
1ª Solución	5593-14	7154-13	4480-9	7878-13
Mejor Solución	5357-11	7154-13	4480-9	7878-13
Número Cruces	4	10	4	6

El análisis de estos datos parece indicar que para situaciones de tráfico ligero resulta más interesante el empleo de la heurística 2 mientras que para las situaciones de tráfico ligeramente congestionado la heurística 3 o la 2 proporcionan los mejores resultados. No obstante, las diferencias obtenidas no son determinantes y, por lo tanto, podría darse la situación de que con otro conjunto de datos inicial diferente las mejoras observadas se orientasen hacia el empleo de la heurística 1.

Como análisis final de las prestaciones que se pueden obtener de la ejecución de la tarea primaria del agente de solución se puede considerar a ésta como muy dependiente de los datos iniciales para la simulación. Por lo tanto, para decidir que heurística aplicar en cada instante en un sistema real se debe de evaluar esta tarea con distintos conjuntos de datos iniciales asociados a diferentes situaciones reales de congestión del tráfico en la red.

El comportamiento del agente de solución repartos cuando se ejecuta dentro del prototipo MASHGREEN DM es el mostrado en la gráfica 5-13. Los tiempos dedicados a esperar los datos iniciales con los que comenzar la búsqueda de los mejores repartos es dependiente del instante en que el agente de preproceso finalice su tarea. En ese instante el agente de preproceso recibe las peticiones de datos procedentes de los agentes de predicción, solución repartos y solución cci.

El agente de solución recibe los datos del agente de preproceso y los almacena en las estructuras de datos adecuadas para realizar el proceso de búsqueda. La tarea de preparar los datos ocupa un tiempo muy pequeño en comparación con el de espera de recepción de los datos. Por último, el agente de solución ejecuta el proceso de búsqueda que finalizará cuando se cumpla el tiempo de integración de datos de los sensores en el

sistema o bien cuando se produzca la ejecución de alguna acción de control externa por parte del operador del sistema. En este caso, el agente de solución anula el proceso de búsqueda y vuelve a solicitar los datos iniciales para esta nueva situación, una vez que los cambios asociados a esa acción externa han sido realizados, al agente de preproceso. Con estos datos intenta de nuevo buscar la mejor asignación de vectores de reparto para cada cruce durante el tiempo que le reste hasta el cumplimiento del instante de integración de datos.

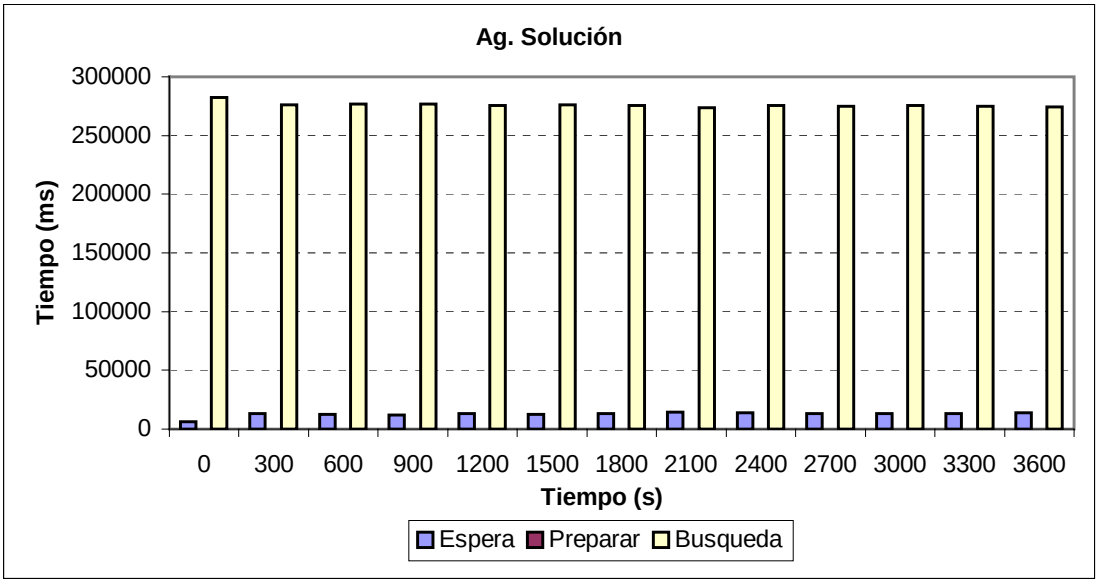


Fig 5-13. *Gráfica del comportamiento del agente de solución repartos durante cada uno de los intervalos de integración de datos en el sistema MASHGREEN DM. Durante el proceso de búsqueda no se ejecutaron acciones de control externas al sistema.*

El agente de solución para el cálculo de los mejores valores para la duración de cada ciclo definido en el sistema también se ha evaluado de dos formas diferentes: aislado del resto de agentes del sistema y actuando conjuntamente con ellos.

El análisis individual se ha realizado utilizando los mismos conjuntos iniciales de datos que los empleados en la evaluación del agente de solución repartos. Todos los cruces pertenecientes a cada uno de los escenarios iniciales pertenecen a un mismo cci. El proceso de simulación consistió en determinar el comportamiento de la simulación cuando el cci funciona con cada uno de los valores de longitud permitidos para el ciclo. Durante el proceso de simulación los valores de reparto iniciales para cada cruce se mantienen constantes.

Los resultados se analizaron de acuerdo a tres parámetros: tiempo empleado en preparar el proceso de simulación, coste obtenido al ejecutar la simulación y tiempo empleado en ejecutar el proceso de simulación.

Duración	**80/100/120**	**80/100/120**	**80/100/120**	**80/100/120**
	D1T100	**D2T100**	**D3T100**	**D4T100**
Preparar	17/17/17	18/50/20	17/17/17	17/48/19
Coste	130-2 162-2 143-2	920 1040-13 996-10	121-1 140-2 152-1	1003-10 941-9 862-8
Tiempo Utilizado	4542 4148 3737	4939 5201 4562	3933 3705 3583	5010 4251 4004
	D1T200	**D2T200**	**D3T200**	**D4T200**
Preparar	21/21/21	21/24/26	20/21/20	21/24/24
Coste	643-7 850-8 396-2	2403-14 2391-17 2533-10	388-2 829-7 457-4	2311-12 2267-12 2778-15
Tiempo Utilizado	8738 7696 6934	8838 8463 7768	7995 7060 6685	9196 7727 7446
	D1T300	**D2T300**	**D3T300**	**D4T300**
Preparar	26/22/24	21/24/26	20/21/20	21/24/24
Coste	1667-7 1840-9 1507-7	3769-11 4155-19 4353-17	1164-6 1764-9 1718-9	4008-14 3926-14 4577-18
Tiempo Utilizado	13969 12203 10905	13614 12576 11551	12955 11216 10478	13984 12085 11690

Los resultados obtenidos muestran que para los conjuntos de datos que representan situaciones de tráfico ligeramente congestionadas (conjuntos *D2* y *D4*) el coste de la simulación realizada es mayor que si la situación de tráfico inicial se corresponde con un flujo ligero (conjuntos *D1* y *D3*). El coste asociado a la tarea de preparar los datos para la simulación permanece razonablemente estable para cada intervalo común de simulación.

Otro factor que se desprende de este estudio es que cuanto mayor es el valor de la duración del ciclo, menor es el tiempo que se precisa para realizar la simulación. Esto es debido a que a mayor valor de la longitud del ciclo menos cambios de fase se realizan en los cruces durante el periodo de evaluación, y por lo tanto, se producen menos eventos en el sistema.

El comportamiento del agente de solución cci en el prototipo MASHGREEN DM se ha analizado mediante la monitorización de algunas de las componentes principales de su tarea primaria. La primera acción que el agente de solución realiza cuando recibe la orden de ejecutar su tarea primaria es el envío de un mensaje al agente de preproceso para que le suministre los datos iniciales correspondientes al nuevo periodo de análisis. Una vez recibidos, el agente de solución cci prepara los datos para poder realizar todas las simulaciones que le sean posibles hasta que se cumpla el instante de integración de

datos en el sistema. La componente encargada de preparar los datos para las simulaciones tiene un comportamiento temporal despreciable con respecto al tiempo empleado por el resto de componentes de su tarea primaria.

En el caso de que se ejecuten acciones de control externas en el sistema, entonces el agente de solución cci anula el proceso de búsqueda de los mejores valores para los ccis, y se espera hasta que finalice la ejecución de esa acción de control. En ese instante solicita los nuevos datos actualizados al agente de preproceso e inicia su tarea primaria hasta que se agote el tiempo disponible.

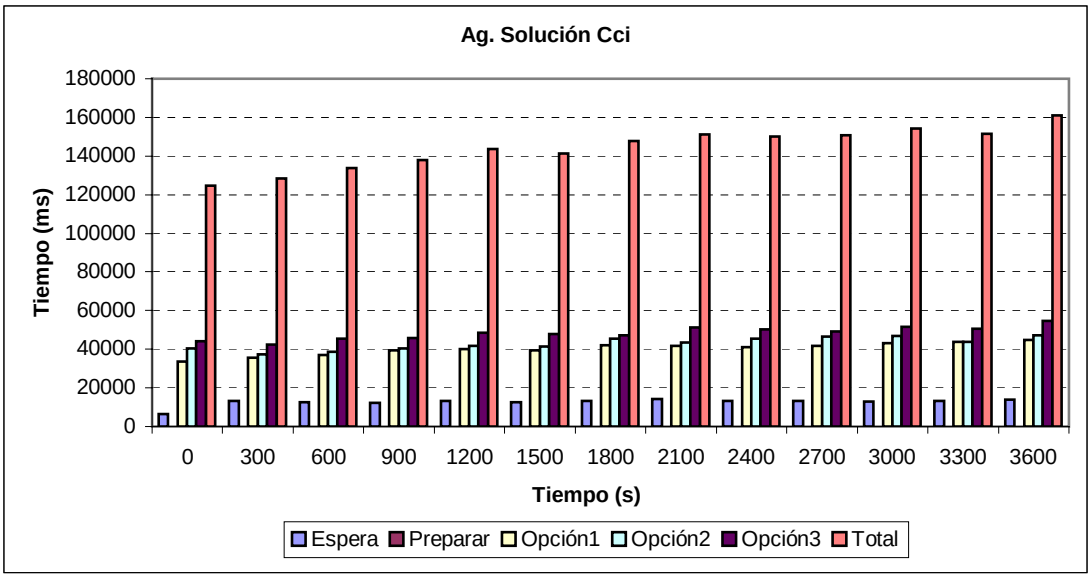


Fig 5-14. *Gráfica del comportamiento de la tarea primaria del agente de solución cci durante cada intervalo de integración de datos y ante la ausencia de la ejecución de acciones de control externas en el sistema.*

En la figura 5-14 se observa que, al tener que calcular únicamente tres opciones de valores para cada cci y disponer de un único cci, la tarea primaria del agente de solución cci finaliza antes de que se alcance el instante de integración de datos en el sistema.

Una característica que se destaca de este estudio es que, al contrario de lo que sucedía con las ejecuciones aisladas e individuales del agente de solución cci, cada nuevo opción a calcular dentro de un mismo periodo de integración precisa de una cantidad de tiempo levemente superior a la utilizada para la opción previa. Este comportamiento se explica en función de la sobrecarga asociada al mantenimiento de los datos cualitativos dinámicos de cada objeto del sistema durante cada periodo de simulación.

La última gráfica de este estudio se corresponde con la relación existente entre el tiempo disponible para cada agente respecto del valor del periodo de integración de datos en el sistema.

Este análisis se muestra en la figura 5-15.

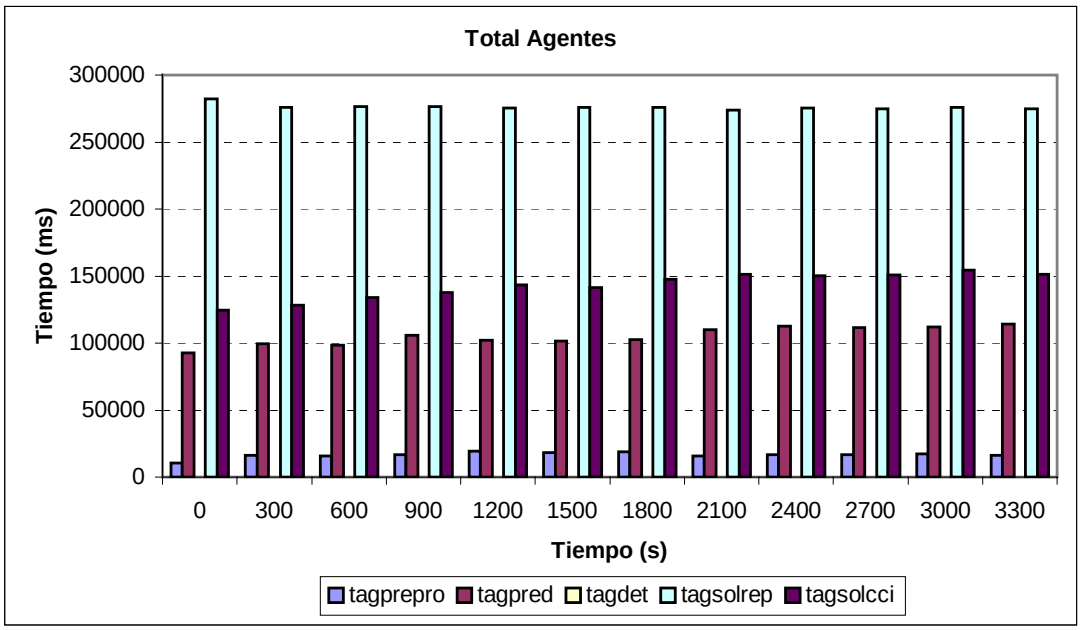


Fig 5-15. *Gráfica de la relación existente entre el tiempo dedicado por cada agente del sistema en ejecutar sus tareas primarias respecto del tiempo total disponible para su ejecución (el periodo de integración).*

Como se observa en la figura, los agentes de predicción, solución repartos y solución cci utilizan gran parte del tiempo disponible para ejecutar sus tareas primarias.

El agente de preproceso es un caso particular, puesto que en las pruebas se ha ejecutado sin utilizar datos reales de los sensores y, por lo tanto, la componente de su tarea primaria dedicada a evaluar cual ha sido la evolución del sistema durante el último periodo de integración no se ejecuta. Otro factor a considerar respecto del tiempo dedicado por el agente de preproceso a ejecutar su tarea primaria es que se ejecuta en tres instantes diferentes durante cada periodo de integración, al contrario que el resto de agentes que sólo se ejecutan una vez al inicio de cada periodo de integración. Como ya se ha indicado este comportamiento se altera en el caso que se ejecuten acciones de control externas en el sistema.

Un caso aparte es el comportamiento del agente de detección de problemas. Tal como está planteado, la duración de la ejecución de su tarea primaria es muy corta, por lo que en el caso de que no se dispusiera de suficientes nodos trabajadores del sistema *Beowulf* para cada uno de los agentes del prototipo *MASHGREEN DM*, entonces el agente de detección podría compartir nodo con otro de los agentes con menor carga, como pueden ser el nodo en el que se ejecuta el agente de preproceso o en el nodo en el que se ejecuta el agente de predicción.

Otro factor que hay que considerar respecto al análisis de los datos representados en la figura 5-15 es que los algoritmos utilizados para desarrollar cada una de las tareas

primarias de los agentes del prototipo, no son definitivos, sino unas primeras aproximaciones de cómo la información extraída del modelo cualitativo desarrollado puede ayudar en la resolución de algunos de los problemas asociados al control de tráfico urbano.

CONCLUSIONES Y TRABAJO FUTURO

Conclusiones y Trabajo Futuro.

Los aspectos generales que han intentado cubrirse durante el desarrollo del prototipo MASHGREEN DM se pueden agrupar en tres apartados:

1. Beneficio en el empleo de simulación cualitativa.

2. Definición de una arquitectura funcional abierta.

3. Integración de la arquitectura en un sistema real.

Las tendencias actuales en la programación de simuladores de tráfico urbano se orientan hacia un mayor esfuerzo en el desarrollo de simuladores microscópicos. Este tipo de simuladores proporciona una gran variedad de medidas muy cercanas a la realidad. No obstante, su empleo tiene que ser constantemente equilibrado entre grado de detalle de la simulación y tiempo de ejecución. Éste es el motivo por el que gran parte de los microsimuladores de tráfico urbano desarrollados están siendo reprogramados en entornos de ejecución paralelos[1]. El enfoque seleccionado para el desarrollo del prototipo MASHGREEN DM utiliza otro enfoque distinto. El método consiste en utilizar un simulador macroscópico centrada en la determinación de la densidad de tráfico como herramienta fundamental con el que desarrollar métodos aplicables a la gestión de tráfico urbano.

El simulador macroscópico seleccionado presenta una serie de características que lo hacen adecuado para este propósito. El coste computacional del simulador es muy bajo (lineal respecto al número de segmentos, cruces y número máximo de fases, capítulo 2) lo que permite incrementar el tamaño de la red urbana a modelar con lo que se tiene una visión más global del estado de la red. Esta característica, por lo tanto, hace que el análisis y la toma de decisiones en el sistema se realice con un gran conocimiento del estado general del tráfico en la red. Otra de las ventajas del simulador seleccionado es que presenta un bajo coste espacial. Efectivamente, al ser un simulador orientado a eventos discretos, el simulador sólo almacena aquellas situaciones que presenten un nivel cognitivo significativo en la representación de la evolución del tráfico.

El inconveniente principal de este simulador es que sólo proporciona la densidad de vehículos en cada segmento, por lo que si, por ejemplo, se desea estudiar la evolución de su composición de vehículos este simulador no es válido. Por lo tanto, se trata de desarrollar tareas que, utilizando estos datos cualitativos como datos iniciales, sean capaces de resolver cuestiones generales de gestión de tráfico.

Con estas premisas el prototipo MASHGREEN DM se compone de una colección de agentes cuyas habilidades se fundamentan en el uso intenso de este modelo cualitativo. La tarea de preproceso utiliza datos cualitativos para determinar la evolución del tráfico urbano ante la ausencia de datos procedentes de los sensores. Los datos de los sensores,

[1] En SMARTEST, http://www.its.leeds.ac.uk/smartest , se analizan 17 microsimuladores de tráfico junto con algunas de sus versiones paralelas.

en el momento en que están disponibles, se utilizan junto con el modelo cualitativo para determinar el estado del tráfico actual. Como consecuencia de este proceso puede aparecer diferencias entre el escenario de tráfico obtenido ante la ausencia de datos de los sensores respecto del obtenido con los datos de los sensores. El análisis de esta diferencia puede ayudar a detectar posibles incidentes en el sistema (tarea realizada por el agente de diagnosis) y a la tarea de identificar los sensores de tráfico con un funcionamiento anómalo (como el método apuntado en [Moreno, 96]). Además, debido a la gran variedad de sensores de tráfico urbano disponibles (apartado 1.3), un trabajo a desarrollar consiste en cómo construir datos cualitativos utilizando las medidas proporcionadas por los tipos de sensores disponibles. Algunos de estos métodos ya están desarrollados, como por ejemplo, los que permiten la conversión de datos procedentes de los sensores de espira [Moreno, 96].

Otra tarea a mejorar consiste en proporcionar algoritmos que saquen beneficio de los datos de los sensores tan pronto como estos estén disponibles. Si algunos sensores pudiesen proporcionan datos antes del vencimiento del periodo de integración entonces estos datos se podrían utilizar para ajustar el comportamiento del proceso simulador de una forma más continua, evitando la situación actual que consiste en realizar ajustes bruscos del estado del tráfico cada $T_{integración}$ segundos.

El bajo coste computacional del proceso de simulación cualitativa permite al agente de predicción estimar cual puede ser la situación del tráfico a medio y largo plazo. De esta forma, conjuntamente con los resultados proporcionados por el agente de detección de problemas, el usuario puede determinar las acciones de control externas adecuadas para evitar potenciales problemas de tráfico. Para realizar este proceso de análisis del estado del tráfico en la red, el agente de detección utiliza una clasificación jerárquica y constructiva de las distintas situaciones problemáticas que se pueden presentar en el tráfico urbano. Esta clasificación parte del análisis de los datos cualitativos de cada segmento en las inmediaciones de un cruce. A partir de estos datos se identifican situaciones problemáticas elementales, que si persisten en el tiempo se convierten en temporales, y que si involucran a cruces conectados entre si representan situaciones problemáticas espacio-temporales.

Esta definición constructiva y jerárquica resulta muy adecuada para la tarea del control del tráfico urbano puesto que esta tarea pertenece a la categoría de sistemas no controlables[2]. Por lo tanto, resulta fundamental para el sistema de control de tráfico el poder adelantarse a las situaciones de congestión antes de que éstas se produzcan. Éste es el principio en el que se basan las tareas realizadas por el agente de solución repartos y solución cci. Ambos agentes toman como datos iniciales los datos obtenidos mediante el análisis conjunto de los datos de los sensores y los proporcionados por el proceso de simulación cualitativa. Estos datos son utilizados de forma intensa para intentar conseguir minimizar el número y categoría de los problemas detectados en la red urbana. De esta forma, "suavizando" el comportamiento del tráfico en el presente se intenta mejorar el estado del tráfico futuro.

[2] Es decir, aquellos sistemas para los cuales no resulta posible ejecutar una secuencia de acciones que desde cualquier estado intermedio pueda alcanzar el estado inicial en un tiempo finito.

Este mecanismo de búsqueda de soluciones para los problemas de tráfico actuales permite que el sistema de control de tráfico pueda aplicar de forma automática las soluciones calculadas sin necesidad de la intervención continua del operador de la sala. De esta forma se suaviza su carga de trabajo asociada a la tarea de supervisión y actuación sobre el tráfico. En el prototipo MASHGREEN DM también se aporta un agente que calcula cuál puede ser la mejor asignación para la duración del ciclo de cada cci del sistema. No obstante, la aplicación automática y continua de tales resultados, aunque el sistema lo permita, no es aconsejable, puesto que el cambio del valor de la duración del cci es una acción de control que no se realiza frecuentemente pues suele provocar efectos adversos en el tráfico. Las soluciones aportadas por este agente pueden ser utilizadas para realizar un análisis off-line del comportamiento del sistema, o bien para su aplicación automática cuando estas soluciones se repitan de forma frecuente.

El prototipo MASHGREEN DM depende en gran medida de la fiabilidad del modelo cualitativo de tráfico seleccionado. Por este motivo es muy necesario algún mecanismo de reajuste del modelo para evitar conflictos cuando el tráfico no se comporta exactamente según el modelo. El método de reajuste utilizado en este momento consiste en el contraste de los datos cualitativos obtenidos con los datos de los sensores respecto de los obtenidos por sólo simulación. Este método sólo es válido para obtener los datos iniciales para cada periodo de integración. Por lo tanto, una de las tareas a desarrollar es el desarrollo de técnicas que permitan la adaptación continua del modelo cualitativo a las características del tráfico.

Una de las formas en las que realizar esta adaptación podría basarse en el ajuste dinámico de los porcentajes de giro en cada intersección. En el modelo cualitativo actual cada intersección se comporta de forma constante con respecto al porcentaje de vehículos que giran en cada dirección permitida para cada una de sus fases. Este comportamiento podría alterarse mediante la inclusión de un proceso de aprendizaje que se fuese ajustando a los porcentajes de giro medidos durante cada periodo de integración. Esta solución requeriría que los sensores de tráfico utilizados proporcionasen, además del flujo de tráfico medido, los giros realizados durante el periodo de integración. Resultaría tarea del agente de preproceso el análisis de estos datos y el posterior ajuste de los porcentajes de giro.

Otra de las cuestiones a mejorar es el comportamiento del modelo cualitativo en su aplicación por parte de los agentes de predicción, solución repartos y solución cci. En todos estos agentes se considera que no se producen nuevas entradas de vehículos durante cada periodo de integración. Para estos agentes sería interesante disponer de un modelo de llegada de vehículos para cada entrada. Uno de los métodos más sencillos sería el suponer que estas entradas persisten respecto de sus valores iniciales durante un determinado tiempo (menor que la duración del ciclo del cci al que pertenece el primer cruce en su dirección) o bien permitir su ajuste paulatino según los valores proporcionados por algunos selectos sensores del sistema.

Además, de todas estas características y propuestas de mejora, no hay que olvidar que estos agentes son experimentales, en el sentido que han sido evaluados en pruebas

de laboratorio y, por lo tanto, precisan de un proceso de afinación cuando sus ejecuciones sean realizadas en un sistema real.

El segundo aspecto a evaluar del prototipo MASHGREEN DM es la arquitectura funcional desarrollada. El prototipo MASHGREEN DM requiere de la definición de una arquitectura funcional adecuada para aplicaciones en las que sea preciso ejecutar mecanismos de comportamiento en tiempo real *suave*[3] en la que se pudiesen integrar todos los métodos y tareas desarrolladas. La primera de las arquitecturas propuestas utiliza el modelo de pizarra con algunas modificaciones respecto a la comunicación entre agentes del sistema (capítulo 3). El desarrollo de este primer prototipo en su versión secuencial permitió identificar problemas de diseño y de medios materiales con los que programarla. Estas cuestiones (expuestas en el apartado 3.11) junto con la comercialización de nuevos sistemas de computadores más versátiles determinaron el cambio de diseño de la arquitectura funcional a un sistema distribuido multiagente.

La primera tarea desarrollada consistió en definir cual iba a ser la arquitectura común para cada agente del nuevo prototipo. El análisis de los requerimientos de cada uno de los agentes de la versión secuencial en el primer prototipo desarrollado motivaron el establecimiento de tres niveles para cada agente del sistema.

Uno de los niveles está dedicado a la definición e implementación de los mecanismos de comunicación comunes a todos los agentes del sistema. Los agentes del sistema comparten unos mismos mecanismos de sincronización, la sincronización por barrera, que permite realizar operaciones de modificación de la información común a todos los agentes del sistema. Este mecanismo se emplea en el prototipo desarrollado para el establecimiento del valor del reloj virtual para todos los agentes del sistema, aunque también podría emplearse para incorporar o dar de baja a alguno de sus agentes. De esta forma se puede conseguir una arquitectura en la que el número de agentes que la constituyen, así como las funciones que realizan, puede ser modificado dinámicamente sin necesidad de detener la ejecución del prototipo para realizar los cambios.

Otra de las características de este nivel de comunicaciones es que establece un conjunto de tipos de mensajes que todos los agentes del sistema deben ser capaces de procesar. Estos mensajes son de cuatro tipos: mensaje de inicio, monitorización del prototipo, sincronismo y petición de datos. Alternativamente a estas categorías de mensajes, el prototipo no impone ninguna restricción respecto al resto de mensajes que puedan circular por el sistema salvo que los agentes participantes en cada comunicación entiendan su significado.

Cada agente del prototipo contiene otro nivel en el que reside el conjunto de datos que precisa para alcanzar sus objetivos operacionales. Este nivel se denomina el espacio de datos, y su definición surge como consecuencia del análisis del uso de los datos dinámicos por el primero de los prototipos desarrollados. En este análisis se determinó que una de las formas de garantizar la integridad de los datos era que cada agente se

[3] Entendiendo por tal a aquel sistema que impone restricciones temporales no estrictas para la ejecución de alguna tarea

hiciese responsable de una parte de los mismos. Además, como se comprobó en el primer prototipo, se podía establecer una distribución de datos que no decrementase de forma significativa las prestaciones del sistema. La información de dónde reside cada conjunto de datos es estática y debe ser conocida por todos los agentes del sistema. Los agentes que precisan acceder a datos que no pertenecen a su espacio de datos realizan peticiones a los agentes adecuados. Además, estas peticiones se realizan poco frecuentemente, generalmente una vez por ciclo excepto en el caso que se ejecuten acciones de control externas en el sistema en cuyo caso se solicitan otra vez más.

Para asegurar la correcta interpretación de los datos que se comunican en el sistema todos los agentes comparten el mismo conocimiento de las estructuras de datos utilizadas para representar cada uno de los objetos del sistema. Es decir, todos los agentes comparten el conocimiento de cómo se representan los valores de fases y sus tiempos de comienzo y fin para cualquiera de los cruces del sistema y la representación de la evolución dinámica en cualquiera de los segmentos o cruces del dominio.

Otra de las ventajas adicionales que proporciona el distribuir los datos de la aplicación entre los agentes es que se aumenta la fiabilidad del sistema ante agentes que dejen de funcionar. Resulta más sencillo conseguir este comportamiento en el segundo prototipo (en el que los agentes son independientes entre sí) que en el primero de los prototipos desarrollados.

El tercer nivel en la estructura de cualquiera de los agentes del nuevo prototipo especifica la colección de métodos (aptitudes) que es capaz de realizar, es decir, su especialización. Uno de esos métodos se define como primario pues representa su principal objetivo operacional. El resto de métodos se definen como secundarios pues representan objetivos relacionados con la integración de este agente en la arquitectura propuesta. Los métodos secundarios se ocupan de la realización de las tareas asociadas con la recepción o envío de mensajes. De esta forma, cada agente debe poseer al menos cuatro métodos secundarios básicos, uno para tratar cada categoría de mensaje común a todos los agentes del sistema. Cuando el método primario de un agente precise establecer la comunicación con algún otro agente del sistema entonces se debe solicitar la ejecución del método secundario adecuado.

La definición establecida para cada agente del prototipo es deliberadamente sencilla para permitir la fácil integración y cooperación entre todos los agentes del sistema. Los mecanismos complejos que determinan sus actuaciones se establecen de forma interna al agente, en función de la implementación de estos tres niveles y de las interacciones que se establezcan entre éstos.

Más aún, las técnicas que se puedan utilizar para el desarrollo de cada agente del prototipo son independientes entre sí, lo que permite, en principio, el desarrollo de agentes programados en diferentes lenguajes de programación o que ejecuten distintos métodos de inferencia para resolver sus tareas.

El análisis conjunto de los agentes en la arquitectura propuesta presenta una serie de características a destacar. El prototipo desarrollado se orienta hacia la especialización de

los agentes por tareas y no por áreas de la red urbana. A diferencia de otros prototipos (como por ejemplo el prototipo *KITS* comentado en el apartado 2.4.3.4) la arquitectura propuesta en el prototipo MASHGREEN DM incorpora un especialista para conseguir cada uno de los objetivos operacionales. Cada especialista tiene un dominio de aplicación que comprende a toda la red urbana bajo análisis. Este comportamiento resulta factible al considerar la red urbana desde un punto de vista cualitativo. Como ya se ha indicado previamente, este análisis se puede realizar de forma muy rápida lo que permite incrementar el tamaño de la red urbana bajo estudio.

No obstante, esta orientación hacia las tareas no es la única admitida. Al tratarse de una arquitectura abierta se permite la incorporación de otros agentes especializados en zonas urbanas vecinas con las pertenecientes al dominio original en el sistema. Así, por ejemplo, seria posible que el dominio original analice una malla urbana, y que posteriormente se incorporen agentes que ayuden a la gestión de tráfico de una arteria principal próxima a la malla original. La conexión entre ambos dominios, en el caso de que exista, vendría establecida por los objetos comunes a ambas zonas (y esta posibilidad representa otro argumento a favor del establecimiento de categorías de mensajes y de estructuras de datos comunes para todos los agentes integrados en el sistema).

Otra de las características funcionales de esta arquitectura es el bajo número de mensajes que se comunican durante cada periodo de integración. Efectivamente, un análisis pormenorizado del comportamiento de cada agente en el sistema durante cada periodo de integración permite identificar una cota superior para el número de mensajes emitidos, así como analizar su tamaño. En este análisis no se consideran los mensajes de monitorización, pues su finalidad en el prototipo no es ayudar a la resolución de tareas de tráfico, sino ayudar en el desarrollo de la implementación del prototipo.

El agente de preproceso, en el prototipo actual, es el agente que mayor número de mensajes emite. Como resultado del proceso de simulación, este agente envía al agente de interfaz un máximo de un mensaje por segundo. Cada uno de estos mensajes contiene los cambios de estado cualitativos producidos en algunos de los S segmentos y C cruces que componen el dominio de aplicación. Por lo tanto, el tamaño máximo de cada uno de estos mensajes está acotado superiormente por $S + C$. En el caso que no tenga que ejecutar el proceso de simulación, el agente de preproceso solicita los datos al agente de predicción. Esta petición se realiza dos veces en cada periodo de integración. En cada ciclo de simulación aquellas situaciones problemáticas elementales y temporales detectadas son agrupadas y enviadas como un solo mensaje al agente de detección. Este mensaje tiene un tamaño acotado superiormente por el número de cruces, C, para cada instante del ciclo. Por último, el agente de preproceso envía los datos iniciales al agente de predicción, solución repartos y solución cci una vez por cada periodo de integración. Cada uno de estos mensajes tiene un tamaño acotado por los valores cualitativos de cada objeto del dominio junto con las características de funcionamiento de cada cruce (2 por cada cruce). El tamaño de cada uno de estos mensajes es $S + 3C$.

El agente de predicción realiza una petición de datos al agente de preproceso durante el primer ciclo de simulación del periodo de integración. Para cada uno del resto de ciclos, el agente de predicción envía un mensaje al agente de preproceso en cuyo contenido se incluyen los datos cualitativos de cada objeto del dominio durante todo el ciclo. Este mensaje tiene un tamaño máximo acotado por el valor *(S + C) * Número de segundos de un ciclo*. Si se denota por *NP* al número de ciclos de predicción que ejecuta este agente, entonces se envían *NP* mensajes al agente de interfaz (cada uno de un tamaño máximo *S + C*) y otros *NP* mensajes al agente detector (cada uno de un tamaño máximo *C* para cada instante del ciclo).

Agente	**Número Mensajes Emitidos**		**Tamaño Mensaje**
Preproceso	300 2 1 1 1	Interfaz Predicción Detector SoluciónRep SoluciónCci	S + C 1 C * 100 S + 3C S + 3C
Predicción	1 6 2 6	Preproceso Detector Preproceso Interfaz	1 C * 100 (C + S) * 100 S + C
Detector	300	Interfaz	1
Coordinación	3 1 1 1 1	Preproceso Predicción SoluciónRep SoluciónCci Detector	1 1 1 1 1
SoluciónRep	1 1 1	Preproceso Preproceso SoluciónCci	1 1 1
SoluciónCci	1 1 1	Preproceso Preproceso SoluciónRep	1 1 1
Total	**633 Mensajes/Integración**		

Tabla 1. *Mensajes emitidos durante cada periodo de integración de 300 segundos dividido en ciclos de 100 segundos. El agente de predicción realiza 6 ciclos de predicción (NP = 6).*

El agente de detección de problemas emite un máximo de un mensaje al agente de interfaz por cada instante de duración del periodo de integración, mientras que los agentes de solución repartos y solución cci emiten un mensaje de comunicación de las

soluciones calculadas a cada agente que lo solicite (en el prototipo son tres agentes). Por último, el agente de coordinación emite un total de siete mensajes de petición de ejecución de la tarea primaria durante todo el periodo de integración. Todos los mensajes emitidos por los agentes de coordinación, solución repartos y solución cci pueden ser considerados de tamaño 1.

Como se observa en la tabla resumen 1 el número de mensajes emitidos durante cada periodo de integración ante la ausencia de la ejecución de acciones de control externas en el sistema es muy bajo. Además, debido a las características de comportamiento en tiempo real *suave* del dominio de la aplicación, el tiempo exigido para realizar la comunicación de cualquiera de los mensajes no es crítico para la evolución del sistema.

En el caso que se ejecuten acciones de control externas el número de mensajes emitidos no cambia sustancialmente. Efectivamente, durante cada ciclo de ejecución, todas las acciones de control externas realizadas se ejecutan al comienzo del siguiente ciclo. Por lo tanto, por cada una de estas situaciones, el agente de preproceso ha de recomponer los estados cualitativos de todos los objetos del sistema desde el instante asociado a estas situaciones y proporcionar los nuevos datos iniciales para el resto de agentes de la arquitectura. La siguiente tabla resumen identifica los mensajes emitidos como consecuencia de esta situación.

Agente	**Número Mensajes Emitidos**		**Tamaño Mensaje**
Preproceso	299 1 1 1 1	Interfaz Resto Detector SoluciónRep SoluciónCci	S + C 1 C * 100 S + 3C S + 3C
Predicción	2	Preproceso	1
SoluciónRep	1	Preproceso	1
SoluciónCci	1	Preproceso	1
Total	**307 Mensajes/Integración**		

Tabla 2. *Mensajes emitidos como consecuencia de la ejecución de acciones de control externas.*

Una última característica de la arquitectura funcional propuesta es que las tareas del agente de coordinación del sistema pueden ser asumidas por el resto de agentes, y por lo tanto, esta arquitectura puede evolucionar a un sistema completamente distribuido.

El tercer aspecto a evaluar en este resumen general es la arquitectura física en la que implementar el prototipo MASHGREEN DM. El planteamiento inicial de la tesis se orientaba hacia el uso de un multiprocesador con memoria compartida. No obstante, esta elección no resultó satisfactoria por una serie de cuestiones (apartado 3.11). La elección, por lo tanto, se orientó hacia la búsqueda de una arquitectura física que combinase tres factores: coste económico razonable, buenas prestaciones computacionales y la disponibilidad de herramientas estables que permitan su programación de forma paralela/distribuida. Estas características se dan en los clusters de PCs tipo *Beowulf* (apartado 5.1).

En estos sistemas se dispone fundamentalmente de mecanismos de comunicación por paso de mensajes mediante el empleo de los modelos *PVM* y *MPI*. Además, otros estándares de comunicaciones, como es el caso del modelo de programación *linda*, también pueden ser utilizados. No obstante, las mejores prestaciones se obtienen mediante el uso de los modelos *PVM* y *MPI* puesto que utilizan la red de interconexión de altas prestaciones del sistema.

La elección del mecanismo de comunicaciones *linda* se realizó teniendo en cuenta tanto su disponibilidad como las características del prototipo desarrollado, en el que las altas prestaciones en las comunicaciones no son un objetivo fundamental. Como ya se comentó, el número de mensajes intercambiados en el sistema es muy bajo, por lo que la mejora en las prestaciones del prototipo está fundamentalmente condicionada por la forma en que se ejecutan las tareas primarias en cada agente. Este argumento, junto con la disponibilidad de utilizar las primitivas de comunicación *linda* en casi cualquier entorno de programación lógico basado en restricciones, determinó la elección del estilo de comunicaciones a utilizar en el prototipo MASHGREEN DM.

Una ventaja fundamental de los sistemas *Beowulf* es que pueden estar compuestos de distintos tipos de computadores (es decir, son clusters heterogéneos). Por lo tanto, se pueden asignar los computadores más potentes a las tareas con mayor coste computacional. Esta característica es muy importante para el prototipo MASHGREEN DM pues, como se ha apuntado anteriormente, en este sistema de pueden ejecutar tareas especificadas en diferentes lenguajes y con diferentes métodos de inferencia. Por ejemplo, uno de los nodos del cluster puede ser un sistema tetraprocesador, y a ese nodo asignarle el agente cuya tarea sea más costosa. Esta tarea podría ejecutarse haciendo uso de la programación con *threads* para intentar obtener sus resultados de una forma más rápida.

Además, los sistemas *Beowulf* se pueden ampliar sin tener que realizar grandes desembolsos económicos ni modificaciones complejas en la configuración del sistema original. Esta característica, junto con las especificaciones realizadas en la definición de cada agente del prototipo MASHGREEN DM permite el diseño de tareas secundarias para ampliar el número de agentes en ejecución en el sistema sin ocasionar pérdidas en las prestaciones del resto de agentes en ejecución (tal y como se apuntó en el capítulo 4). Por lo tanto, otro de los trabajos a desarrollar en un futuro consiste en implementar mecanismos que permitan realizar altas y bajas dinámicas de agentes en el sistema, o

incluso, implementar un mecanismo de tolerancia a fallos para aquellos agentes cuyos resultados sean fundamentales para la ejecución del resto de componentes.

Otra de las ventajas de este tipo de sistemas es que proporcionan un mecanismo limitado de acceso desde el mundo exterior al cluster. Esta característica permite realizar un mejor equilibrado de la carga, puesto que los agentes disponen de la mayor parte de los recursos en su nodo asociado para ejecutar sus métodos. Por otra parte, al asignar el agente de interfaz al nodo externo sobre el que realizar las comunicaciones con el mundo exterior al sistema *Beowulf*, también se permite comunicar datos del dominio del sistema a otras aplicaciones externas al prototipo MASHGREEN DM.

El prototipo implementado tiene un *talón de Aquiles*. La correcta ejecución del nodo servidor del proceso *linda* resulta fundamental para el sistema. Si este nodo falla, todo el sistema de comunicaciones queda inutilizado. De hecho, el comportamiento del espacio de tuplas en un sistema *linda* puede compararse con la función que realiza la pizarra en una arquitectura de pizarra clásica.

En el prototipo MASHGREEN DM este espacio de datos se utiliza exclusivamente para realizar las comunicaciones, pero también podría utilizarse para dejar datos comunes accesibles por cualquiera de los agentes del prototipo. Si el nodo que se encarga de la gestión de este espacio de tuplas queda inutilizado, los agentes no pueden comunicarse entre sí. Por lo tanto, para que el prototipo se comporte de una forma completamente distribuida, y con bastante más seguridad, no basta con eliminar el agente de coordinación, puesto que el sistema sigue teniendo un comportamiento centralizado con respecto al modo en que se efectúan las comunicaciones. Una de las tareas por desarrollar, por lo tanto, consiste que el sistema implemente una tolerancia a fallos para el proceso servidor del espacio de tuplas *linda* o bien utilice alguno de los mecanismos de paso de mensajes proporcionados por la arquitectura *Beowulf* (*PVM* o *MPI*) lo que proporcionará una mayor seguridad en el proceso de comunicaciones.

Por último, indicar que el objetivo del desarrollo y análisis de estos prototipos no es la consecución de un sistema directamente aplicable para el control de tráfico urbano, sino la identificación de una arquitectura funcional capaz de ser aplicada a este tipo de problemas. Durante el proceso de definición e implementación de los prototipos se fueron identificando componentes cuyos comportamientos precisaban de mejores métodos que los definidos durante las etapas iniciales de su desarrollo. Como resultado de todo este proceso, se obtiene un prototipo, estable respecto a su comportamiento en ejecución, que indica un camino hacia la construcción de nuevos sistemas de control de tráfico urbano que se incorporen a la tendencia actual en el diseño de sistemas multiagente, utilizando de forma intensa procesos de razonamiento cualitativo y que puedan ser ejecutadas en arquitecturas físicas razonablemente económicas y con buenos índices de prestaciones.

La principal motivación que guió el proceso de generación del prototipo MASHGREEN DM se resume en la (gran) experiencia expresada en el siguiente párrafo:

"Primero se hizo, luego se hizo bien y posteriormente se hizo rápido" La experiencia acumulada en esta sentencia indica que cuando se construyen (nuevos) sistemas software no se debería de intentar conseguir los tres resultados a la vez. Un primer prototipo puede proporcionar una orientación más ajustada de cómo realizar el trabajo correctamente y de forma clara, como consecuencia se obtiene el conocimiento necesario para garantizar la solución. Intentar conseguir estos tres resultados a la vez es garantía de fracaso [Nierstrasz, 92].

APÉNDICE A
ESPECIFICACIÓN DE LOS CONJUNTOS DE DATOS UTILIZADOS EN LA EVALUACIÓN DEL PROTOTIPO MASHGREEN

Índice del Apéndice A.

A.1 Características Topológicas y de Control

El escenario urbano seleccionado como datos de entrada para el prototipo MASHGREEN se corresponde con una selección amplia de segmentos y cruces pertenecientes al centro urbano de la ciudad de Castellón de la Plana.

Los valores asociados a las características topológicas de cada uno de los segmentos y cruces seleccionados fueron proporcionados en el desarrollo del proyecto *"Programa Director para los estudios de tráfico y transporte en Castellón"* [Castellón, 97] financiado por el Excmo. Ayuntamiento de Castellón de la Plana y desarrollado por la Universidad Politécnica de Cataluña y la Universidad Jaume I de Castellón durante el año 1997. Estos valores se representan mediante hechos *prolog* que tienen el siguiente formato:

```
top(Id, segmento, Coordenadas, Orientación, Nombre Calle,
    Longitud, Número Carriles, Obj entrada, Obj Salida).

top(Id, cruce, Coordenadas, CCI, Segmentos de Entrada,
    Segmentos de Salida).
```

Además de los segmentos y cruces, también se representan los objetos asociados a las entradas y salidas del escenario de tráfico seleccionado. Estos elementos se representan mediante hechos *prolog* con el siguiente formato:

```
top(Id, entrada, Nombre Calle, Id Segmento).

top(Id, salida,  Nombre Calle, Id Segmento).
```

Todos estos hechos se almacenan en un mismo fichero, *calles_data*, cuyo contenido, (para los segmentos, entradas y salidas seleccionadas), es el siguiente (los datos asociados a cruces se representarán junto a las características de control):

```
top(l1, segmento, data(38,19,47.5,19,47.5,57,38,57), v,
    Doctor Clara, 140, 3, 0, ifis1).

top(l2, segmento, data(57,57,66.5,57,66.5,19,57,19), v,
    Doctor Clara, 140, 3, ifis1, 0).

top(l5, segmento, data(9.5,66.5,38,66.5,38,76,9.5,76), h,
    Monte Blanco, 180, 3, 0, ifis1).

top(l6, segmento, data(38,85.5,9.5,85.5,9.5,95,38,95), h,
    Monte Blanco, 180, 3, ifis1, 0).

top(l7, segmento, data(76,76,104.5,76,104.5,85.5,76,85.5), h,
    Juan Pascual, 140, 3, ifis1, ifis2).

top(l3, segmento, data(38,104.5,47.5,104.5,47.5,133,38,133), v,
    Doctor Clara, 160, 3, ifis1, ifis3).

top(l4, segmento, data(57,133,66.5,133,66.5,104.5,57,104.5), v,
    Doctor Clara, 160, 3, ifis3, ifis1).

top(l8, segmento, data(114,57,123.5,57,123.5,19,114,19), v,
    Perez Galdos, 128, 3, ifis2, 0).
```

```
top(l9, segmento, data(114,133,123.5,133,123.5,104.5,114, 104.5), v,
    Perez Galdos, 128,3, ifis4, ifis2).

top(l10, segmento, data(38,142.5,9.5,142.5,9.5,152,38,152), h,
    República Argentina, 208, 3, ifis3, 0).

top(l11, segmento, data(104.5,142.5,76,142.5,76,152,104.5,152), h,
    República Argentina, 100, 3, ifis4, ifis3).

top(l12, segmento, data(209,142.5,133,142.5,133,152,209,152), h,
    República Argentina, 200, 4, ifis5, ifis4).

top(l13, segmento, data(209,76,218.5,76,218.5,133,209,133), v,
    Paseo Ribalta, 280, 3, 0, ifis5).

top(l14, segmento, data(228,133,237.5,133,237.5,76,228,76), v,
    Paseo Ribalta, 280, 3, ifis5, 0).

top(l15, segmento, data(285,142.5,237.5,142.5,237.5,152,285, 152), h,
    Ronda Magdalena, 260, 3, ifis6, ifis5).

top(l16, segmento, data(237.5,161.5,285,161.5,285,171,237.5, 171), h,
    Ronda Magdalena, 260, 3, ifis5, ifis6).

top(l17, segmento,data(361,142.5,313.5,142.5,313.5,152,361,152), h,
    Ronda Magdalena, 140, 3, 0, ifis6).

top(l18, segmento,data(313.5,161.5,361,161.5,361,171,313.5,171), h,
    Ronda Magdalena, 140, 3, ifis6, 0).

top(l19, segmento, data(294.5,76,304,76,304,133,294.5,133), v,
    Joaquin Costa, 320, 3, 0, ifis6).

top(l20, segmento, data(47.5,161.5,57,161.5,57,190,47.5,190), v,
    Navarra, 68, 3, ifis3, ifis7).

top(l21, segmento, data(47.5,218.5,57,218.5,57,256.5,47.5, 256.5), v,
    Navarra, 200, 3, ifis7, ifis11).

top(l22, segmento, data(114,190,123.5,190,123.5,161.5,114, 161.5), v,
    San Vicente, 84, 3, ifis8, ifis4).

top(l23, segmento, data(114,256.5,123.5,256.5,123.5,218.5,114,218.5),
    v, San Vicente, 192, 3, ifis12, ifis8).

top(l24, segmento, data(76,199.5,104.5,199.5,104.5,209,76, 209), h,
    Ronda Mijares, 80, 3, ifis7, ifis8).

top(l25, segmento, data(133,199.5,209,161.5,209,171,133,209), ds,
    Ronda Mijares, 200, 3, ifis8, ifis5).

top(l26, segmento, data(285,180.5,294.5,180.5,294.5,209,285, 209), v,
    A. Alfonso, 132, 3, ifis6, ifis10).

top(l27, segmento, data(304,209,313.5,209,313.5,180.5,304, 180.5), v,
    A. Alfonso, 132, 3, ifis10, ifis6).

top(l28, segmento, data(38,275.5,9.5,275.5,9.5,285,38,285), h,
    San Francisco, 140, 3, ifis11, 0).

top(l29, segmento, data(104.5,266,76,266,76,275.5,104.5, 275.5), h,
    Avenida Rey Don Jaime, 100, 3, ifis12, ifis11).

top(l30, segmento, data(76,285,104.5,285,104.5,294.5,76, 294.5), h,
    Avenida Rey Don Jaime, 100, 3, ifis11, ifis12).

top(l31, segmento, data(209,218.5,133,266,133,275.5,209,228), ds,
    Avenida Rey Don Jaime, 208, 3, ifis9, ifis12).

top(l32, segmento, data(133,285,209,237.5,209,247,133,294.5), ds,
    Avenida Rey Don Jaime, 208, 3, ifis12, ifis9).
```

```
top(l33, segmento, data(275.5,218.5,237.5,218.5,237.5,228,275.5,228),
    h, Avenida Rey Don Jaime, 240, 3, ifis10, ifis9).

top(l34, segmento, data(237.5,237.5,275.5,237.5,275.5,247,237.5,247),
    h, Avenida Rey Don Jaime, 240, 3, ifis9, ifis10).

top(l35, segmento, data(218.5,180.5,228,180.5,228,209,218.5, 209), v,
    Zaragoza, 184, 3, ifis5, ifis9).

top(l36, segmento, data(218.5,247,228,247,228,294.5,218.5, 294.5), v,
    Zaragoza, 120, 3, ifis9, ifis14).

top(l37, segmento, data(47.5,294.5,57,294.5,57,323,47.5,323), v,
    Real, 180, 3, ifis11, ifis13).

top(l72, segmento, data(66.5,332.5,114,304,114,313.5,66.5, 342), ds,
    Ruiz Zorrila, 128, 2, ifis13, ifis12).

top(l38, segmento, data(38,342,9.5,342,9.5,351.5,38,351.5), h,
    Trinidad, 172, 3, ifis13, 0).

top(l39, segmento, data(76,408.5,104.5,408.5,104.5,418,76, 418), h,
    Herrero, 108, 3, 0, ifis17).

top(l40, segmento, data(104.5,456,57,456,57,465.5,104.5, 465.5), h,
    Plaza Fadrell, 100, 3, ifis20, 0).

top(l41, segmento, data(209,304,66.5,351.5,66.5,361,209, 313.5), ds,
    Enmedio, 240, 3, ifis14, ifis13).

top(l42, segmento, data(285,285,237.5,304,237.5,313.5,285, 294.5),ds,
    Enmedio, 240, 3, ifis15, ifis14).

top(l43, segmento, data(38,370.5,47.5,370.5,114,399,104.5, 399), dp,
    Falco, 92, 3, ifis13, ifis17).

top(l71, segmento, data(123.5,399,133,399,66.5,370.5,57, 370.5), dp,
    Gasset, 108, 2, ifis17, ifis13).

top(l44, segmento, data(114,427.5,123.5,427.5,123.5,446.5,114,446.5),
    v, Asensi, 200, 3, ifis17, ifis20).

top(l45, segmento, data(114,475,123.5,475,123.5,522.5,114, 522.5), v,
    Maestro Ripolles, 212, 3, ifis20, 0).

top(l46, segmento, data(209,361,133,408.5,133,418,209,370.5), ds,
    Mayor, 260, 3, ifis17, ifis18).

top(l47, segmento, data(237.5,361,285,342,285,351.5,237.5,370.5), ds,
    Mayor, 252, 3, ifis18, ifis19).

top(l48, segmento, data(218.5,323,228,323,228,351.5,218.5, 351.5), v,
    Colon, 184, 3, ifis14, ifis18).

top(l49, segmento, data(285,256.5,294.5,256.5,294.5,275.5,285,275.5),
    v, San Luis, 120, 3, ifis10, ifis15).

top(l50, segmento, data(304,275.5,313.5,275.5,313.5,256.5,304,256.5),
    v, San Luis, 120, 3, ifis15, ifis10).

top(l51, segmento, data(285,304,294.5,304,294.5,332.5,285, 332.5), v,
    San Luis, 200, 3, ifis15, ifis19).

top(l52, segmento, data(304,332.5,313.5,332.5,313.5,304,304, 304), v,
    Pestagua, 180, 3, ifis19, ifis15).

top(l53, segmento, data(313.5,228,332.5,228,332.5,237.5,313.5,237.5),
    h, Santo Tomas, 140, 3, ifis10, 0).

top(l54, segmento, data(218.5,380,228,380,228,446.5,218.5, 446.5), v,
    Vives, 128, 3, ifis18, ifis22).
```

```
top(l55, segmento, data(152,456,133,456,133,465.5,152,465.5), h,
    Tarrega, 180, 3, ifis21, ifis20).

top(l56, segmento, data(209,456,180.5,456,180.5,465.5,209, 465.5), h,
    Gobernador, 180, 3, ifis22, ifis21).

top(l57, segmento, data(285,370.5,237.5,456,237.5,465.5,285, 380),ds,
    Gobernador, 268, 3, ifis19, ifis22).

top(l58, segmento, data(152,475,161.5,475,161.5,522.5,152, 522.5), v,
    Hermanos Bou, 240, 3, ifis21, 0).

top(l59, segmento, data(171,522.5,180.5,522.5,180.5,475,171, 475), v,
    Hermanos Bou, 240, 3, 0, ifis21).

top(l60, segmento, data(209,475,218.5,475,218.5,522.5,209, 522.5), v,
    Avenida del Mar, 148, 3, ifis22, 0).

top(l61, segmento, data(228,522.5,237.5,522.5,237.5,475,228, 475), v,
    Avenida del Mar, 148, 3, 0, ifis22).

top(l62, segmento, data(332.5,285,313.5,285,313.5,294.5,332.5,294.5),
    h, Felix, 312, 3, ifis16, ifis15).

top(l63, segmento, data(332.5,275.5,342,275.5,342,256.5,332.5,256.5),
    v, Arquitecto Maristan, 80, 3, ifis16, 0).

top(l64, segmento, data(370.5,275.5,351.5,275.5,351.5,285,370.5,285),
    h, San Roque, 248, 3, 0, ifis16).

top(l65, segmento, data(351.5,294.5,370.5,294.5,370.5,304,351.5,304),
    h, San Roque, 248, 3, ifis16, 0).

top(l66, segmento, data(313.5,342,342,304,342,313.5,313.5, 351.5),ds,
    Sanahuja, 348, 3, ifis19, ifis16).

top(l67, segmento, data(361,361,313.5,361,313.5,370.5,361, 370.5), h,
    Avenida de los Capuchinos, 248, 3, 0, ifis19).

top(l68, segmento, data(313.5,380,361,380,361,389.5,313.5, 389.5), h,
    Avenida de los Capuchinos, 248, 3, ifis19, 0).

top(l69, segmento, data(285,399,294.5,399,294.5,475,285,475), v,
    Avenida Virgen del Lidon, 960, 3, ifis19, 0).

top(l70, segmento, data(304,475,313.5,475,313.5,399,304,399), v,
    Avenida Virgen del Lidon, 960, 3, 0, ifis19).


top(e1, entrada,Doctor Clara, l1).

top(e5, entrada, Monte Blanco, l5).

top(e13, entrada, Ribalta, l13).

top(e17, entrada, Ronda Magdalena, l17).

top(e19, entrada, Joaquin Costa, l19).

top(e39, entrada, Herrero, l39).

top(e59, entrada, Hermanos Bou, l59).

top(e61, entrada, Avenida del Mar, l61).

top(e64, entrada, San Roque, l64).

top(e67, entrada, Avenida de los Capuchinos, l67).

top(e70, entrada, Avenida Virgen del Lidon, l70).
```

```
top(s2, salida, Doctor Clara, l2).
top(s6, salida, Monte Blanco, l6).
top(s8, salida, Perez Galdos, l8).
top(s10, salida, Republica Argentina, l10).
top(s14, salida, Ribalta, l14).
top(s18, salida, Ronda Magdalena, l18).
top(s28, salida, San Francisco, l28).
top(s38, salida, Trinidad, l38).
top(s40, salida, Plaza Fadrell, l40).
top(s45, salida, Maestro Ripolles, l45).
top(s53, salida, Santo Tomas, l53).
top(s58, salida, Hermanos Bou, l58).
top(s60, salida, Avenida del Mar, l60).
top(s63, salida, Arquitecto Maristan, l63).
top(s65, salida, San Roque, l65).
top(s68, salida, Avenida de los Capuchinos, l68).
top(s69, salida, Avenida Virgen del Lidon, l69).
```

Las características topológicas de los cruces junto con sus características de control fueron determinadas en el proyecto [Castellón, 97]. Los datos asociados al porcentaje de giro en cada intersección, junto con sus fases y vectores de reparto se representan en las siguientes figuras:

Código Cruce: *ifis1*

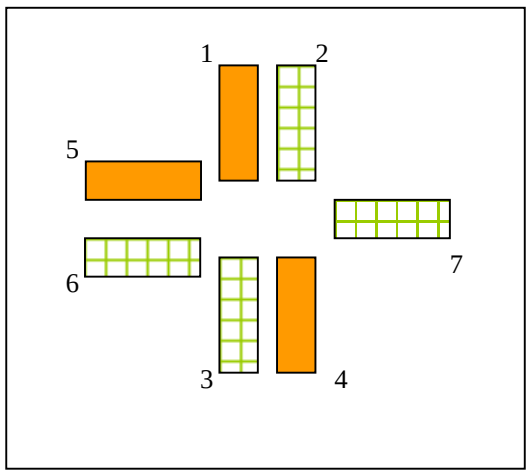
Cruce Nº 12

Entradas: *l1,l4,l5*
Salidas: *l2,l3,l6,l7*

FASES:
1. (l1,l6,30), (l1,l3,70), (l4,l2,80), (l4,l7,20)
2. (l1,l7,100)
3. (l5, l2,40), (l5,l3,40), (l5,l7,20)
4. ()

REPARTOS:
1. (500, 200, 200, 100)
2. (450, 250, 200, 100)
3. (400, 250, 250, 100)
4. (350, 300, 250, 100)
5. (300, 300, 300, 100)
6. (250, 350, 350, 100)
7. (200, 350, 350, 100)
8. (150, 400, 350, 100)

CROQUIS



Código Cruce: *ifis2*

Cruce Nº X_1

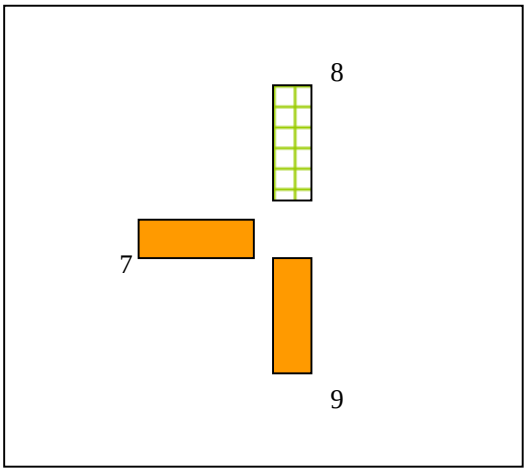
Entradas: *l7,l9*

Salidas: *l8*

FASES:
1. (l7,l8,100), (l9,l8,100)

REPARTOS:
1. (1000)
2. (1000)
3. (1000)
4. (1000)
5. (1000)
6. (1000)
7. (1000)
8. (1000)

CROQUIS



Código Cruce: *ifis3*

Cruce Nº X_2

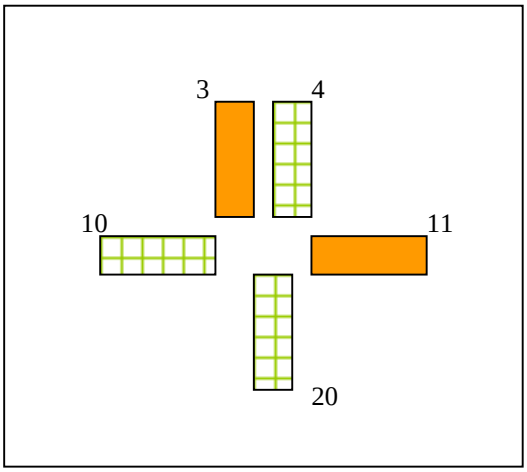
Entradas: *l3,l11*

Salidas: *l4,l10,l20*

FASES:
1. (l3,l10,55), (l3,l20,45), (l11,l4,100)
2. ()
3. (l11,l4,15), (l11,l10,45), (l11,l20,40)
4. ()

REPARTOS:
1. (600, 100, 200, 100)
2. (550, 100, 250, 100)
3. (500, 100, 300, 100)
4. (450, 100, 350, 100)
5. (400, 100, 400, 100)
6. (350, 100, 450, 100)
7. (300, 100, 500, 100)
8. (250, 100, 550, 100)

CROQUIS



Código Cruce: *ifis4*

Cruce Nº 11

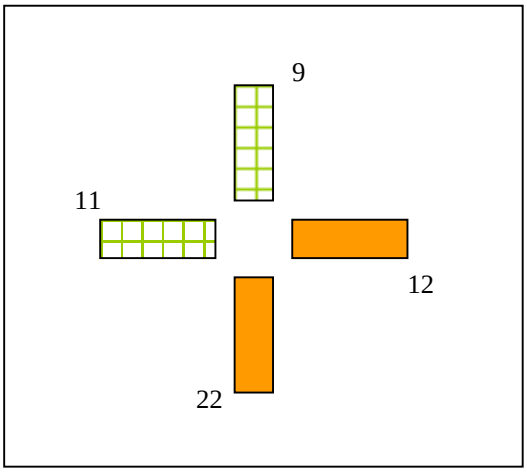
Entradas: *l12,l22*

Salidas: *l9,l11*

FASES:
1. (l22,l9,60), (l22,l11,40)
2. ()
3. (l12,l9,30), (l12,l11,70)
4. ()

REPARTOS:
1. (600, 100, 200, 100)
2. (550, 100, 250, 100)
3. (500, 100, 300, 100)
4. (450, 100, 350, 100)
5. (400, 100, 400, 100)
6. (350, 100, 450, 100)
7. (300, 100, 500, 100)
8. (250, 100, 550, 100)

CROQUIS



Código Cruce: *ifis5*

Cruce Nº 9

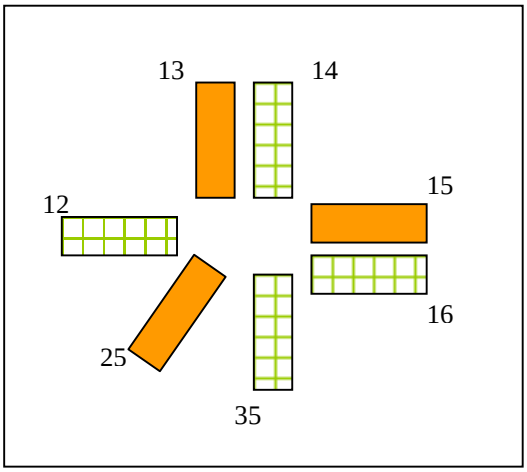
Entradas:*l13,l15,l25*

Salidas: *l12,l14,l16,l35*

FASES:
1. (l13,l12,32), (l13,l35,42), (l13,l16,26)
2. ()
3. (l25,l35,16), (l25,l16,56), (l25,l14,22), (l25,l12,6)
4. ()
5. (l15,l14,30), (l15,l12,58), (l15,l35,12)

REPARTOS:
1. (500, 100, 200, 100, 100)
2. (450, 100, 200, 100, 150)
3. (400, 100, 200, 100, 200)
4. (350, 100, 250, 100, 200)
5. (300, 100, 250, 100, 250)
6. (250, 100, 300, 100, 250)
7. (200, 100, 300, 100, 300)
8. (150, 100, 350, 100, 300)

CROQUIS



Código Cruce: *ifis6*

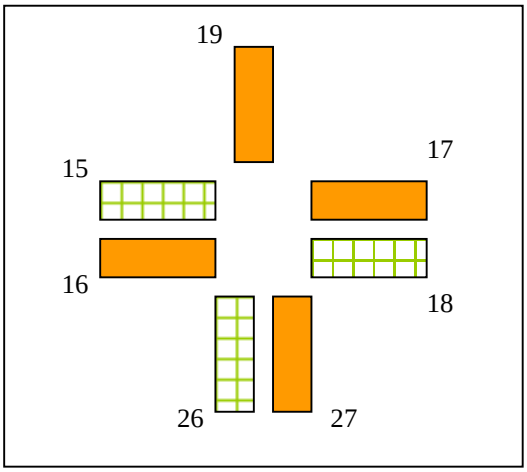
Cruce Nº 8

Entradas: *l16,l17,l19,l27*
Salidas: *l15,l18,l26*

FASES:
1. (l16,l18,77), (l16,l26,23), (l17,l15,100)
2. (l19,l15,10), (l19,l26,60), (l19,l18,30)
3. (l27,l15,100)
4. (l17,l26,100), (l27,l18,100)
5. ()

REPARTOS:
1. (500, 150, 150, 100, 100)
2. (450, 150, 150, 150, 100)
3. (400, 200, 150, 150, 100)
4. (350, 200, 200, 150, 100)
5. (300, 200, 200, 200, 100)
6. (250, 250, 200, 200, 100)
7. (200, 250, 250, 200, 100)
8. (150, 250, 250, 250, 100)

CROQUIS



Código Cruce: *ifis7*

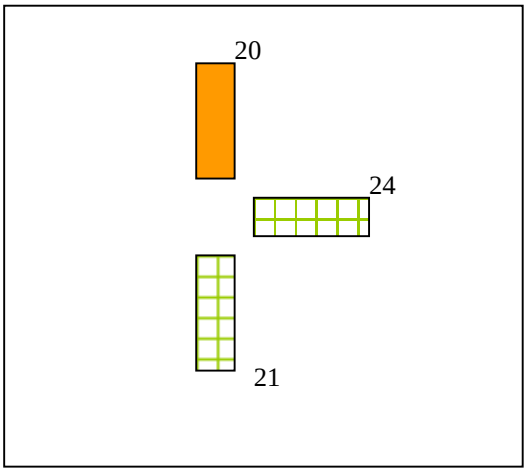
Cruce Nº X_3

Entradas: *l20*
Salidas: *l21,l24*

FASES:
1. (l20,l21,45), (l20, l24,55)
2. ()

REPARTOS:
1. (900, 100)
2. (900, 100)
3. (900, 100)
4. (900, 100)
5. (900, 100)
6. (900, 100)
7. (900, 100)
8. (900, 100)

CROQUIS



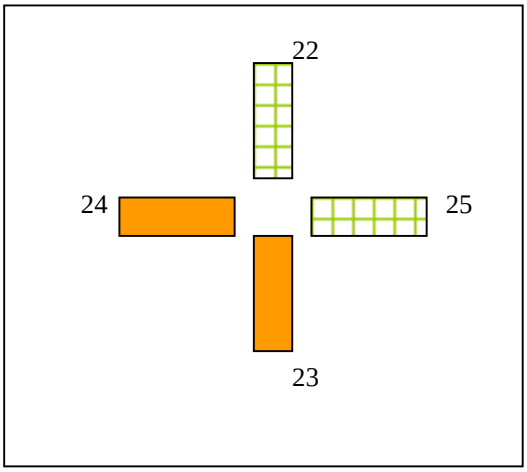
Código Cruce: *ifis8*

Cruce Nº 13

Entradas: *l23,l24*
Salidas: *l22,l25*

CROQUIS

FASES:
1. (l24,l25,88), (l24,l22,12)
2. ()
3. (l23,l25,41), (l23,l22,59)
4. ()



REPARTOS:
1. (600, 100, 200, 100)
2. (550, 100, 250, 100)
3. (500, 100, 300, 100)
4. (450, 100, 350, 100)
5. (400, 100, 400, 100)
6. (350, 100, 450, 100)
7. (300, 100, 500, 100)
8. (250, 100, 550, 100)

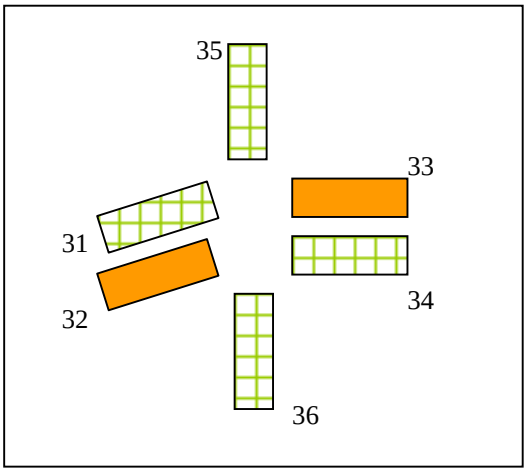
Código Cruce: *ifis9*

Cruce Nº 10

Entradas: *l32,l22,l25*
Salidas: *l31,l34,l36*

CROQUIS

FASES:
1. (l35,l31,21), (l35,l36,62), (l35,l34,17)
2. (l32,l36,60), (l32,l34,40), (l33,l31,100)
3. (l33,l36,100), (l35,l31,100)
4. ()



REPARTOS:
1. (500, 200, 200, 100)
2. (450, 250, 200, 100)
3. (400, 250, 250, 100)
4. (350, 300, 250, 100)
5. (300, 300, 300, 100)
6. (250, 350, 350, 100)
7. (200, 350, 350, 100)
8. (150, 400, 350, 100)

Código Cruce: *ifis10*

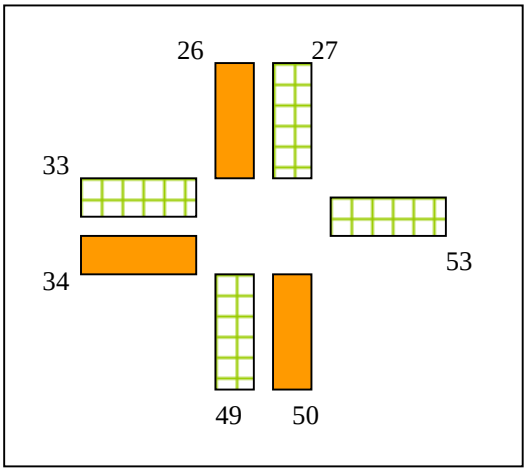
Cruce Nº 18

Entradas: *l26,l34,l50*
Salidas: *l27,l33,l49,l53*

FASES:
1. (l26,l33,22), (l26,l49,78), (l50,l53,6), (l50,l27,94)
2. ()
3. (l34,l27,56), (l34,l49,34), (l34,l53,10)
4. ()

REPARTOS:
1. (600, 100, 200, 100)
2. (550, 100, 250, 100)
3. (500, 100, 300, 100)
4. (450, 100, 350, 100)
5. (400, 100, 400, 100)
6. (350, 100, 450, 100)
7. (300, 100, 500, 100)
8. (250, 100, 550, 100)

CROQUIS



Código Cruce: *ifis11*

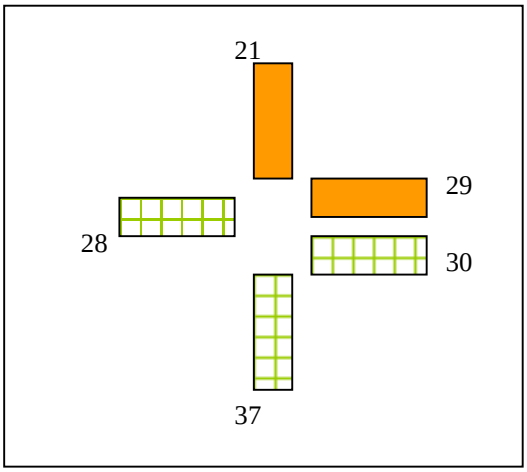
Cruce Nº 14

Entradas: *l21,l29*
Salidas: *l28,l30,l37*

FASES:
1. (l21,l28,60), (l21,l37,40)
2. (l21,l30,100)
3. ()
4. (l29,l28,44), (l29,l37,56)
5. ()

REPARTOS:
1. (500, 200, 100, 100, 100)
2. (450, 200, 100, 100, 150)
3. (400, 200, 100, 100, 200)
4. (350, 250, 100, 100, 200)
5. (300, 250, 100, 100, 250)
6. (250, 300, 100, 100, 250)
7. (200, 300, 100, 100, 300)
8. (150, 350, 100, 100, 300)

CROQUIS



Código Cruce: *ifis12*

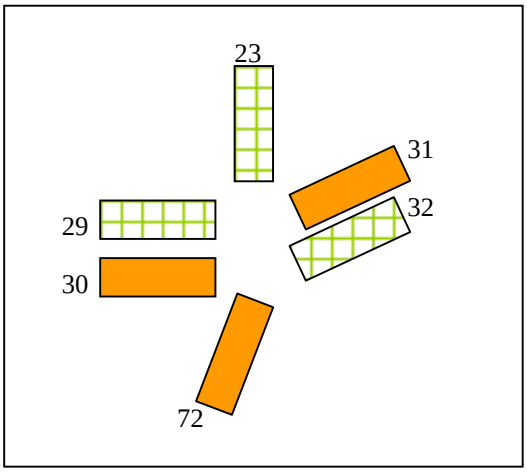
Cruce Nº X_4

Entradas: *l30,l31,l72*
Salidas: *l23,l29,l32*

FASES:
1. (l30,l32,60), (l30,l23,40)
2. (l31,l23,75), (l31,l29,25)
3. (l72,l32,65), (l72,l23,30), (l72,l29,5)
4. ()

REPARTOS:
1. (500, 200, 200, 100)
2. (450, 250, 200, 100)
3. (400, 250, 250, 100)
4. (350, 300, 250, 100)
5. (300, 300, 300, 100)
6. (250, 350, 350, 100)
7. (200, 350, 350, 100)
8. (150, 400, 350, 100)

CROQUIS



Código Cruce: *ifis13*

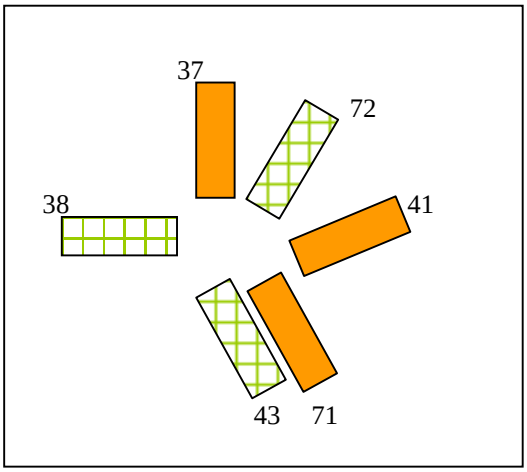
Cruce Nº 26

Entradas: *l37,l41,l71*
Salidas: *l38,l43,l72*

FASES:
1. (l71,l43,8), (l71,l72,92), (l37,l38,100)
2. (l37,l38,12), (l37,l43,88)
3. (l41,l43,25), (l41,l38,60), (l41,l72,15)
4. ()

REPARTOS:
1. (500, 200, 200,100)
2. (450, 250, 200, 100)
3. (400, 250, 250, 100)
4. (350, 300, 250, 100)
5. (300, 300, 300, 100)
6. (250, 350, 350, 100)
7. (200, 350, 350, 100)
8. (150, 400, 350, 100)

CROQUIS



Código Cruce: *ifis14*

Cruce Nº 21

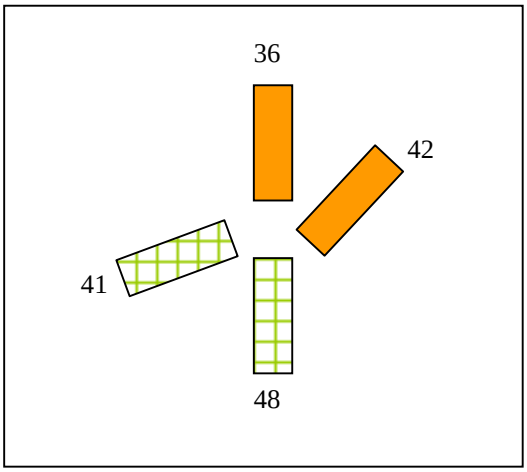
Entradas: *l36,l42*

Salidas: *l41,l48*

FASES:
1. (l36,l41,20), (l36,l48,80)
2. (l42,l41,76), (l42,l48,24)
3. ()

REPARTOS:
1. (650, 250, 100)
2. (600, 300, 100)
3. (550, 350, 100)
4. (500, 400, 100)
5. (450, 450, 100)
6. (400, 500, 100)
7. (350, 550, 100)
8. (300, 600, 100)

CROQUIS



Código Cruce: *ifis15*

Cruce Nº 19

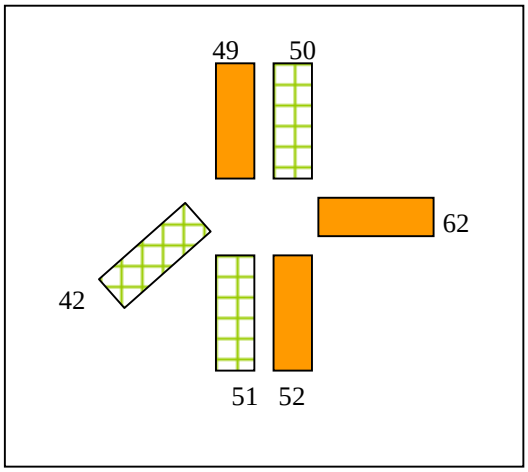
Entradas: *l49,l52,l62*

Salidas: *l42,l50,l51*

FASES:
1. (l62,l50,14), (l62,l42,47), (l62,l51,39)
2. (l49,l42,27), (l49,l51,73)
3. (l52,l50,70), (l52,l42,24), (l52,l51,6)
4. ()

REPARTOS:
1. (500, 200, 200, 100)
2. (450, 250, 200, 100)
3. (400, 250, 250, 100)
4. (350, 300, 250, 100)
5. (300, 300, 300, 100)
6. (250, 350, 350, 100)
7. (200, 350, 350, 100)
8. (150, 400, 350, 100)

CROQUIS



Código Cruce: *ifis16*

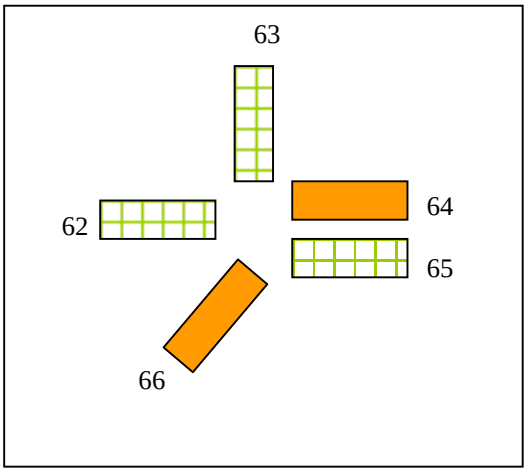
Cruce Nº 17

Entradas: *l64,l66*
Salidas: *l62,l63,l65*

FASES:
1. (l64,l63,2), (l64,l62,98)
2. (l66,l65,70), (l66,l63,18), (l66,l62,12)
3. ()

REPARTOS:
1. (650, 250, 100)
2. (600, 300, 100)
3. (550, 350, 100)
4. (500, 400, 100)
5. (450, 450, 100)
6. (400, 500, 100)
7. (350, 550, 100)
8. (300, 600, 100)

CROQUIS



Código Cruce: *ifis17*

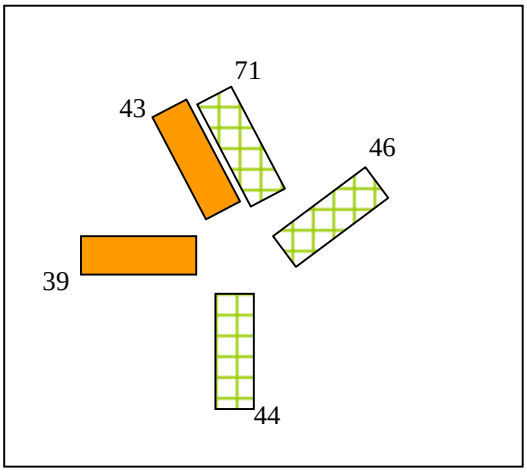
Cruce Nº 28

Entradas: *l39,l43*
Salidas: *l44,l46,l71*

FASES:
1. (l39,l71,25), (l39,l46,60), (l39,l44,15)
2. (l43,l44,47), (l43,l46,38), (l43,l71,15)
3. ()

REPARTOS:
1. (650, 250, 100)
2. (600, 300, 100)
3. (550, 350, 100)
4. (500, 400, 100)
5. (450, 450, 100)
6. (400, 500, 100)
7. (350, 550, 100)
8. (300, 600, 100)

CROQUIS



Código Cruce: *ifis18*

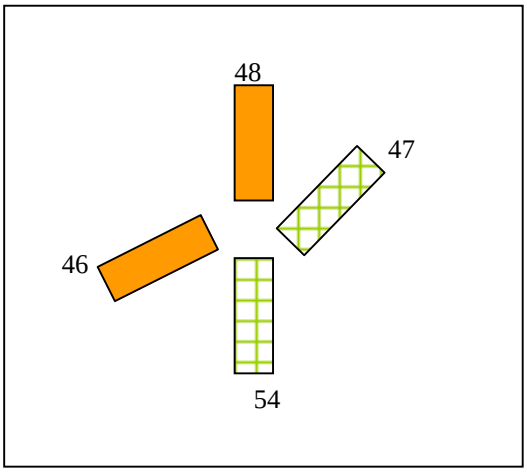
Cruce Nº 22

Entradas: *l46,l48*
Salidas: *l47,l54*

FASES:
1. (l46,l47,66), (l46,l54,34)
2. (l48,l54,77), (l48,l47,23)
3. ()

REPARTOS:
1. (650, 250, 100)
2. (600, 300, 100)
3. (550, 350, 100)
4. (500, 400, 100)
5. (450, 450, 100)
6. (400, 500, 100)
7. (350, 550, 100)
8. (300, 600, 100)

CROQUIS



Código Cruce: *ifis19*

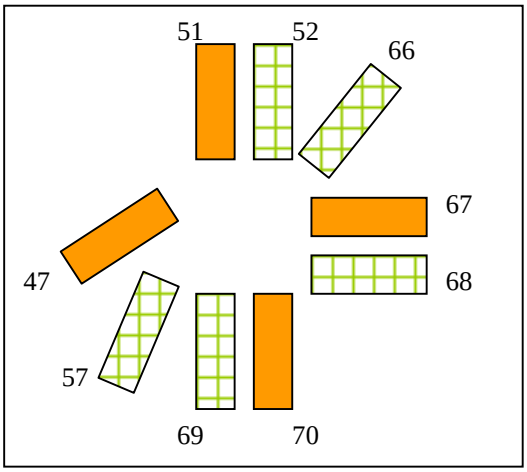
Cruce Nº 20

Entradas: *l47,l51,l67,l70*
Salidas: *l52,l57,l66,l68,l69*

FASES:
1. (l51,l57,12), (l51,l69,12), (l51,l68,56), (l51,l66,20)
2. (l47,l52,23), (l47,l66,33), (l47,l68,33), (l47,l69,11)
3. (l70,l68,22), (l70,l66,20), (l70,l52,18), (l70,l57,40)
4. (l67,l52,40), (l67,l57,10), (l67,l69,13), (l67,l66,37)
5. ()

REPARTOS:
1. (500, 150, 150, 100, 100)
2. (450, 150, 150, 150, 100)
3. (400, 200, 150, 150, 100)
4. (350, 200, 200, 150, 100)
5. (300, 200, 200, 200, 100)
6. (250, 250, 200, 200, 100)
7. (200, 250, 250, 200, 100)
8. (150, 250, 250, 250, 100)

CROQUIS



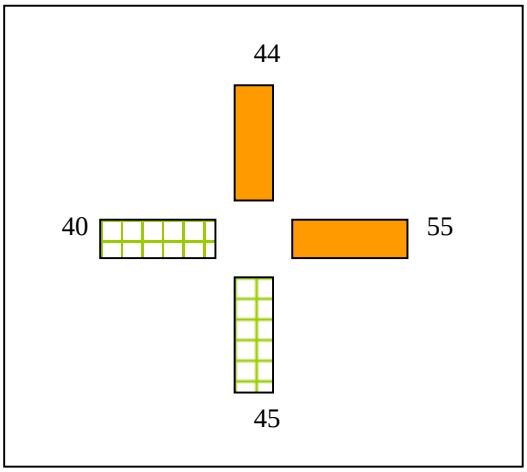
Código Cruce: *ifis20*

Cruce Nº 30

Entradas: *l44,l55*

Salidas: *l40,l45*

CROQUIS



FASES:
1. (l44, l45, 66), (l44,l40,33)
2. (l55, l40, 90), (l55, l45, 10)
3. ()

REPARTOS:
1. (650, 250, 100)
2. (600, 300, 100)
3. (550, 350, 100)
4. (500, 400, 100)
5. (450, 450, 100)
6. (400, 500, 100)
7. (350, 550, 100)
8. (300, 600, 100)

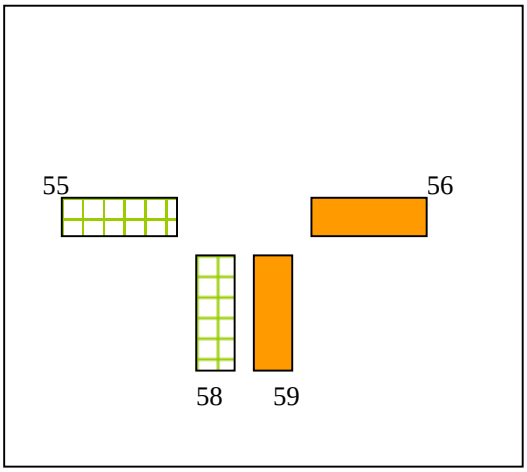
Código Cruce: *ifis21*

Cruce Nº 27

Entradas: *l56,l59*

Salidas: *l55,l58*

CROQUIS



FASES:
1. (l59, l55, 100)
2. (l56, l58, 11), (l56, l55, 89)
3. ()

REPARTOS:
1. (650, 250, 100)
2. (600, 300, 100)
3. (550, 350, 100)
4. (500, 400, 100)
5. (450, 450, 100)
6. (400, 500, 100)
7. (350, 550, 100)
8. (300, 600, 100)

Código Cruce: *ifis22*

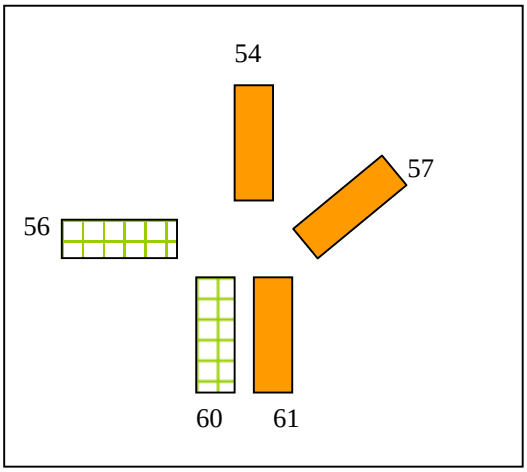
Cruce Nº 23

Entradas: *l54,l57,l61*

Salidas: *l56,l60*

CROQUIS

FASES:
1. (l54, l56, 22), (l54, l60, 78)
2. (l57, l56, 19), (l57, l60, 81)
3. (l61, l56, 100)
4. ()



REPARTOS:
1. (500, 200, 200, 100)
2. (450, 250, 200, 100)
3. (400, 250, 250, 100)
4. (350, 300, 250, 100)
5. (300, 300, 300, 100)
6. (250, 350, 350, 100)
7. (200, 350, 350, 100)
8. (150, 400, 350, 100)

Para cada uno de los cruces se especifica un esquema topológico, su identificador en el prototipo MASHGREEN (*Código Cruce*), el número de cruce del proyecto de medición de datos manuales (*Cruce Nº xx*), los identificadores de sus segmentos de entrada y de salida, las fases que ejecuta y los valores posibles para el vector de reparto a aplicar en cualquier instante.

Cada una de las fases especifica cual es el porcentaje de giro estimado para cada dirección de tráfico permitida. Estos valores se han calculado según las estimaciones medidas para cada cruce para las diferentes categorías de vehículos. El ingeniero de tráfico del ayuntamiento de Castellón de la Plana identificó los cruces que era preciso evaluar. En el proceso de medición se realizaron cuentas distinguiendo entre tres categorías de vehículos (vehículos ligeros, coches y camiones) durante un periodo de evaluación de 15 minutos. Todos estos datos fueron especificados en una tabla como, por ejemplo, la mostrada en la tabla A-1 (cruce *ifis8*, que se corresponde con el cruce *Nº 13* del proyecto de medición).

El método empleado para sacar los porcentajes de giro según cada una de estas tablas consiste en considerar el total de vehículos que circulan sin distinguir entre las categorías expuestas. Se realiza un redondeo al entero más cercano respecto al tanto por cien indicado.

INTERSECCIÓN NÚMERO 13

Fecha: Miércoles, 05 de Febrero de 1997	**Hora: 10:35**

MOVIMIENTO	V.LIGEROS		COCHES		CAMIONES		TOTAL	
	Veh/h	%	Veh/h	%	Veh/h	%	Veh/h	%
l24-l22	8	13.3%	80	10.5%	26	17.8%	114	11.8%
l24-l25	52	86.7%	684	89.5%	120	82.2%	856	88.2%
l23-l22	16	57.1%	206	58.5%	24	60.0%	246	58.6%
l23-l25	12	42.9%	146	41.5%	16	40.0%	174	41.4%

Tabla A-1. *Resultados de la evaluación de la intersección Nº 13.*

A.2 Características Dinámicas Iniciales

Los datos cualitativos iniciales para cada uno de los segmentos del dominio de la aplicación se especifican en hechos prolog *seg_sim* almacenados en ficheros externos. El formato de estos hechos es el siguiente:

```
IdSeg(Tiempo, Lista Estados Cualitativos Iniciales).
```

Hay dos ficheros de datos cualitativos iniciales. El primero representa una situación de flujo de tráfico ligeramente congestionada y sus valores son los siguientes:

```
l1(0, [[0, [[state, [d2,0,100], [d8,100,110], [d4,110,140]]]]]).
l2(0, [[0, [[state, [d4,0,45], [d3,45,90], [d1,90,140]]]]]).
l5(0, [[0, [[state, [d5,0,34], [d3,34,90], [d1,90,140], [d8,140,180]]]]]).
l6(0, [[0, [[state, [d4,0,50], [d3,50,100], [d2,100,180]]]]]).
l7(0, [[0, [[state, [d4,0,20], [d1,20,99], [d2,99,110], [d8,110,140]]]]]).
l3(0, [[0, [[state, [d4,0,90], [d1,90,140], [d8,140,160]]]]]).
l4(0, [[0, [[state, [d4,0,35], [d2,35,100], [d8,100,110], [d4,110,160]]]]]).
l8(0, [[0, [[state, [d4,0,60], [d1,60,100], [d7,100,128]]]]]).
l9(0, [[0, [[state, [d4,0,55], [d1,55,95], [d7,95,128]]]]]).
l10(0, [[0, [[state, [d4,0,55], [d1,55,150], [d6,150,208]]]]]).
l11(0, [[0, [[state, [d4,0,25], [d1,25,75], [d8,75,100]]]]]).
l12(0, [[0, [[state, [d4,0,20], [d1,20,150], [d8,150,200]]]]]).
l13(0, [[0, [[state, [d4,0,34], [d2,34,120], [d1,120,180], [d8,180,210], [d4,210,280]]]]]).
l14(0, [[0, [[state, [d3,0,60], [d1,60,280]]]]]).
l15(0, [[0, [[state, [d4,0,100], [d1,100,200], [d8,200,260]]]]]).
l16(0, [[0, [[state, [d4,0,25], [d1,25,100], [d6,100,140], [d8,140,170], [d4,170,260]]]]]).
```

```
l17(0, [[0, [[state, [d2,0,45], [d8,45,70], [d4,70,140]]]]]).
l18(0, [[0, [[state, [d4,0,56], [d2,56,90], [d3,90,140]]]]]).
l19(0, [[0, [[state, [d2,0,100], [d3,100,150], [d1,150,270], [d8,270,320]]]]]).
l20(0, [[0, [[state, [d4,0,44], [d8,44,68]]]]]).
l21(0, [[0, [[state, [d4,0,35], [d1,35,140], [d8,140,160], [d4,160,200]]]]]).
l22(0, [[0, [[state, [d4,0,20], [d1,20,80], [d4,80,84]]]]]).
l23(0, [[0, [[state, [d4,0,45], [d2,45,70], [d3,70,100], [d1,100,150], [d8,150,192]]]]]).
l24(0, [[0, [[state, [d4,0,40], [d2,40,60], [d1,60,80]]]]]).
l25(0, [[0, [[state, [d4,0,50], [d1,50,170], [d8,170,200]]]]]).
l26(0, [[0, [[state, [d4,0,34], [d1,34,100], [d8,100,132]]]]]).
l27(0, [[0, [[state, [d4,0,50], [d1,50,100], [d8,100,132]]]]]).
l28(0, [[0, [[state, [d4,0,36], [d3,36,78], [d2,78,140]]]]]).
l29(0, [[0, [[state, [d4,0,32], [d2,32,75], [d8,75,100]]]]]).
l30(0, [[0, [[state, [d2,0,40], [d1,40,60], [d8,60,70], [d4,70,100]]]]]).
l31(0, [[0, [[state, [d4,0,30], [d1,30,90], [d2,90,140], [d8,140,170], [d4,170,208]]]]]).
l32(0, [[0, [[state, [d4,0,45], [d2,45,80], [d4,80,90], [d5,90,160], [d6,160,190], [d8,190,208]]]]]).
l33(0, [[0, [[state, [d4,0,66], [d2,66,190], [d8,190,240]]]]]).
l34(0, [[0, [[state, [d4,0,56], [d1,56,210], [d8,210,240]]]]]).
l35(0, [[0, [[state, [d4,0,40], [d1,40,120], [d3,120,150], [d8,150,160], [d4,160,184]]]]]).
l36(0, [[0, [[state, [d4,0,60], [d1,60,100], [d8,100,110], [d4,110,120]]]]]).
l37(0, [[0, [[state, [d4,0,44], [d1,44,100], [d5,100,120], [d8,120,150], [d4,150,180]]]]]).
l38(0, [[0, [[state, [d4,0,15], [d6,15,50], [d1,50,100], [d4,100,172]]]]]).
l39(0, [[0, [[state, [d3,0,70], [d4,70,108]]]]]).
l40(0, [[0, [[state, [d4,0,45], [d2,45,100]]]]]).
l41(0, [[0, [[state, [d4,0,25], [d1,25,65], [d5,65,190], [d8,190,240]]]]]).
l42(0, [[0, [[state, [d4,0,60], [d1,60,120], [d2,120,180], [d6,180,215], [d8,215,240]]]]]).
l43(0, [[0, [[state, [d4,0,10], [d1,10,60], [d8,60,92]]]]]).
l44(0, [[0, [[state, [d4,0,28], [d1,28,130], [d8,130,160], [d4,160,200]]]]]).
l45(0, [[0, [[state, [d4,0,28], [d1,28,160], [d3,160,212]]]]]).
l46(0, [[0, [[state, [d4,0,50], [d1,50,150], [d8,150,200], [d4,200,260]]]]]).
l47(0, [[0, [[state, [d4,0,60], [d1,60,100], [d3,100,175], [d8,175,252]]]]]).
l48(0, [[0, [[state, [d4,0,43], [d1,43,135], [d7,135,150], [d8,150,184]]]]]).
l49(0, [[0, [[state, [d4,0,55], [d2,55,100], [d8,100,120]]]]]).
l50(0, [[0, [[state, [d4,0,23], [d1,23,90], [d8,90,100], [d4,100,120]]]]]).
l51(0, [[0, [[state, [d4,0,45], [d2,45,120], [d4,120,150], [d8,150,160], [d4,160,200]]]]]).
l52(0, [[0, [[state, [d1,0,80], [d3,80,135], [d8,135,180]]]]]).
l53(0, [[0, [[state, [d4,0,15], [d1,15,70], [d4,70,130], [d8,130,140]]]]]).
l54(0, [[0, [[state, [d4,0,34], [d1,34,90], [d8,90,110], [d4,110,128]]]]]).
l55(0, [[0, [[state, [d4,0,44], [d1,44,110], [d3,110,160], [d8,160,180]]]]]).
```

```
l56(0, [[0, [[state, [d4,0,45], [d3,45,90], [d1,90,160], [d8,160,180]]]]).
l57(0, [[0, [[state, [d4,0,30], [d1,30,130], [d3,130,190], [d7,190,220], [d8,220,268]]]]).
l58(0, [[0, [[state, [d1,0,35], [d3,35,150], [d8,150,200], [d4,200,240]]]]).
l59(0, [[0, [[state, [d3,0,50], [d1,50,190], [d8,190,230], [d4,230,240]]]]).
l60(0, [[0, [[state, [d4,0,55], [d1,55,100], [d4,100,148]]]]).
l61(0, [[0, [[state, [d3,0,100], [d1,100,110], [d8,110,148]]]]).
l62(0, [[0, [[state, [d4,0,51], [d1,51,240], [d3,240,312]]]]).
l63(0, [[0, [[state, [d4,0,10], [d1,10,40], [d4,40,80]]]]).
l64(0, [[0, [[state, [d2,0,55], [d3,55,150], [d1,150,190], [d8,190,210], [d4,210,248]]]]).
l65(0, [[0, [[state, [d2,0,50], [d4,50,160], [d5,160,200], [d1,200,248]]]]).
l66(0, [[0, [[state, [d4,0,40], [d1,40,90], [d2,90,285], [d8,285,348]]]]).
l67(0, [[0, [[state, [d1,0,90], [d4,90,220], [d8,220,248]]]]).
l68(0, [[0, [[state, [d4,0,60], [d1,60,190], [d4,190,248]]]]).
l69(0, [[0, [[state, [d4,0,60], [d2,60,700], [d3,700,960]]]]).
l70(0, [[0, [[state, [d2,0,230], [d3,230,500], [d1,500,800], [d8,800,960]]]]).
l71(0, [[0, [[state, [d4,0,34], [d1,34,80], [d8,80,90], [d4,90,108]]]]).
l72(0, [[0, [[state, [d4,0,30], [d1,30,60], [d8,60,75], [d4,75,128]]]]).
```

El segundo representa una situación de flujo de tráfico ligero y sus valores son los siguientes:

```
l1(0, [[0, [[state, [d3,0,30], [d2,30,140]]]]).
l2(0, [[0, [[state, [d2,0,30], [d3,30,140]]]]).
l5(0, [[0, [[state, [d2,0,25], [d4,25,50], [d1,50,180]]]]).
l6(0, [[0, [[state, [d4,0,20], [d2,20,180]]]]).
l7(0, [[0, [[state, [d2,0,140]]]]).
l3(0, [[0, [[state, [d2,0,160]]]]).
l4(0, [[0, [[state, [d3,0,160]]]]).
l8(0, [[0, [[state, [d1,0,128]]]]).
l9(0, [[0, [[state, [d1,0,128]]]]).
l10(0, [[0, [[state, [d3,0,208]]]]).
l11(0, [[0, [[state, [d1,0,12], [d6,12,25], [d1,25,100]]]]).
l12(0, [[0, [[state, [d1,0,30], [d4,30,55], [d1,55,200]]]]).
l13(0, [[0, [[state, [d4,0,28], [d1,28,280]]]]).
l14(0, [[0, [[state, [d1,0,280]]]]).
l15(0, [[0, [[state, [d1,0,30], [d4,30,50], [d1,50,260]]]]).
l16(0, [[0, [[state, [d1,0,10], [d4,10,50], [d1,50,260]]]]).
l17(0, [[0, [[state, [d5,0,14], [d1,14,140]]]]).
l18(0, [[0, [[state, [d3,0,140]]]]).
l19(0, [[0, [[state, [d1,0,35], [d2,35,320]]]]).
```

```
l20(0, [[0, [[state, [d3,0,68]]]]]).
l21(0, [[0, [[state, [d2,0,20], [d3,20,50], [d2,50,200]]]]]).
l22(0, [[0, [[state, [d3,0,50], [d2,50,84]]]]]).
l23(0, [[0, [[state, [d4,0,100], [d1,100,192]]]]]).
l24(0, [[0, [[state, [d1,0,80]]]]]).
l25(0, [[0, [[state, [d5,0,40], [d6,40,50], [d1,50,200]]]]]).
l26(0, [[0, [[state, [d2,0,132]]]]]).
l27(0, [[0, [[state, [d2,0,132]]]]]).
l28(0, [[0, [[state, [d1,0,140]]]]]).
l29(0, [[0, [[state, [d1,0,100]]]]]).
l30(0, [[0, [[state, [d4,0,50], [d1,50,100]]]]]).
l31(0, [[0, [[state, [d3,0,45], [d2,45,208]]]]]).
l32(0, [[0, [[state, [d4,0,20], [d3,20,55], [d1,55,208]]]]]).
l33(0, [[0, [[state, [d2,0,100], [d1,100,240]]]]]).
l34(0, [[0, [[state, [d3,0,30], [d4,30,100], [d1,100,240]]]]]).
l35(0, [[0, [[state, [d5,0,50], [d1,50,184]]]]]).
l36(0, [[0, [[state, [d4,0,50], [d1,50,120]]]]]).
l37(0, [[0, [[state, [d2,0,180]]]]]).
l38(0, [[0, [[state, [d3,0,172]]]]]).
l39(0, [[0, [[state, [d6,0,15], [d1,15,108]]]]]).
l40(0, [[0, [[state, [d1,0,100]]]]]).
l41(0, [[0, [[state, [d1,0,20], [d3,20,50], [d5,50,100], [d1,100,240]]]]]).
l42(0, [[0, [[state, [d7,0,40], [d1,40,240]]]]]).
l43(0, [[0, [[state, [d1,0,60], [d6,60,92]]]]]).
l44(0, [[0, [[state, [d5,0,75], [d1,75,200]]]]]).
l45(0, [[0, [[state, [d1,0,212]]]]]).
l46(0, [[0, [[state, [d2,0,260]]]]]).
l47(0, [[0, [[state, [d1,0,25], [d2,25,252]]]]]).
l48(0, [[0, [[state, [d2,0,184]]]]]).
l49(0, [[0, [[state, [d2,0,12], [d1,12,120]]]]]).
l50(0, [[0, [[state, [d2,0,120]]]]]).
l51(0, [[0, [[state, [d1,0,15], [d7,15,50], [d2,50,200]]]]]).
l52(0, [[0, [[state, [d3,0,180]]]]]).
l53(0, [[0, [[state, [d1,0,140]]]]]).
l54(0, [[0, [[state, [d4,0,128]]]]]).
l55(0, [[0, [[state, [d5,0,20], [d6,20,50], [d1,50,180]]]]]).
l56(0, [[0, [[state, [d3,0,180]]]]]).
l57(0, [[0, [[state, [d4,0,45], [d5,45,120], [d1,120,268]]]]]).
l58(0, [[0, [[state, [d3,0,240]]]]]).
```

```
l59(0, [[0, [[state, [d1,0,240]]]]]).
l60(0, [[0, [[state, [d2,0,148]]]]]).
l61(0, [[0, [[state, [d3,0,148]]]]]).
l62(0, [[0, [[state, [d2,0,312]]]]]).
l63(0, [[0, [[state, [d1,0,80]]]]]).
l64(0, [[0, [[state, [d2,0,248]]]]]).
l65(0, [[0, [[state, [d3,0,248]]]]]).
l66(0, [[0, [[state, [d4,0,30], [d5,30,90], [d1,90,348]]]]]).
l67(0, [[0, [[state, [d1,0,248]]]]]).
l68(0, [[0, [[state, [d1,0,248]]]]]).
l69(0, [[0, [[state, [d2,0,15], [d4,15,100], [d1,100,960]]]]]).
l70(0, [[0, [[state, [d1,0,960]]]]]).
l71(0, [[0, [[state, [d6,0,10], [d7,10,40], [d2,40,108]]]]]).
l72(0, [[0, [[state, [d6,0,15], [d7,15,50], [d2,50,128]]]]]).
```

Además de estos valores cualitativos iniciales estos ficheros se pueden combinar con otros dos ficheros en los que se representan diferentes valores de coordinación para los cruces. Estos valores se muestran en la siguiente tabla:

CRUCE	VALOR COORDINACIÓN 1	VALOR COORDINACIÓN 2
ifis1	100	100
ifis2	100	100
ifis3	600	100
ifis4	600	100
ifis5	550	100
ifis6	100	0
ifis7	100	100
ifis8	100	100
ifis9	100	110
ifis10	100	140
ifis11	100	110
ifis12	100	80
ifis13	0	120
ifis14	900	140
ifis15	400	130
ifis16	400	140
ifis17	100	130
ifis18	100	160
ifis19	300	150
ifis20	400	210
ifis21	350	190
ifis22	300	170

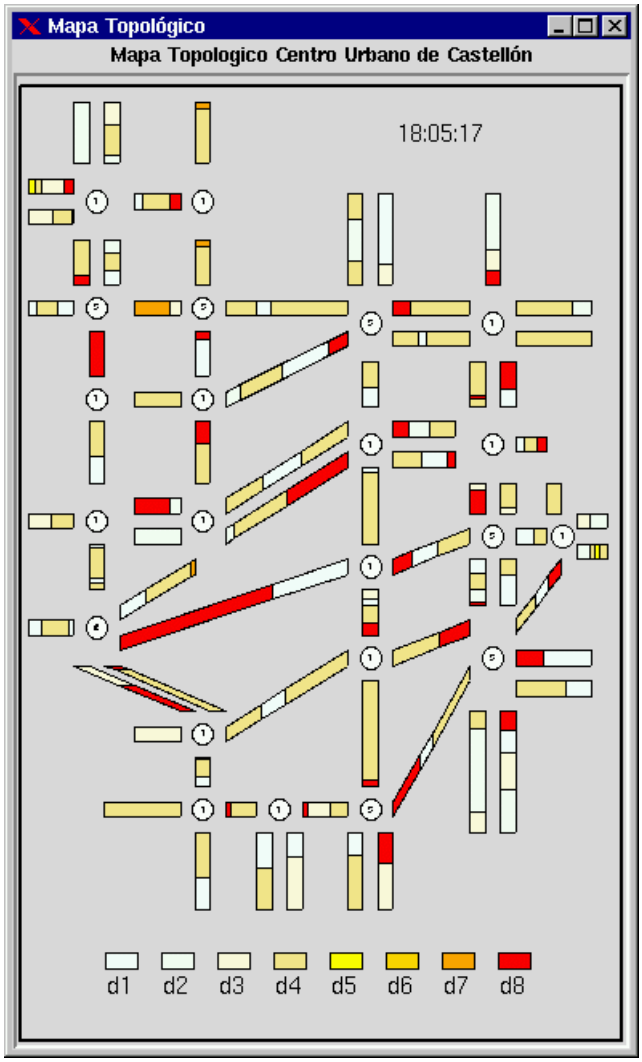
APÉNDICE B
INTERFAZ DE USUARIO DEL PROTOTIPO MASHGREEN DM

Índice del Apéndice B

B.1 Visualización de los Resultados de los Agentes

Durante la ejecución del prototipo MASHGREEN DM aquellos resultados que puedan ser interesantes respecto de la evolución del tráfico se envían al agente de interfaz. Este agente los comunica al operador del prototipo mediante el empleo de las ventanas de la aplicación adecuadas. La programación gráfica del interfaz está desarrollado utilizando las herramientas software público *tcl/tk*[1] y su interfaz con el sistema de desarrollo *sicstus prolog 3.7.1*.

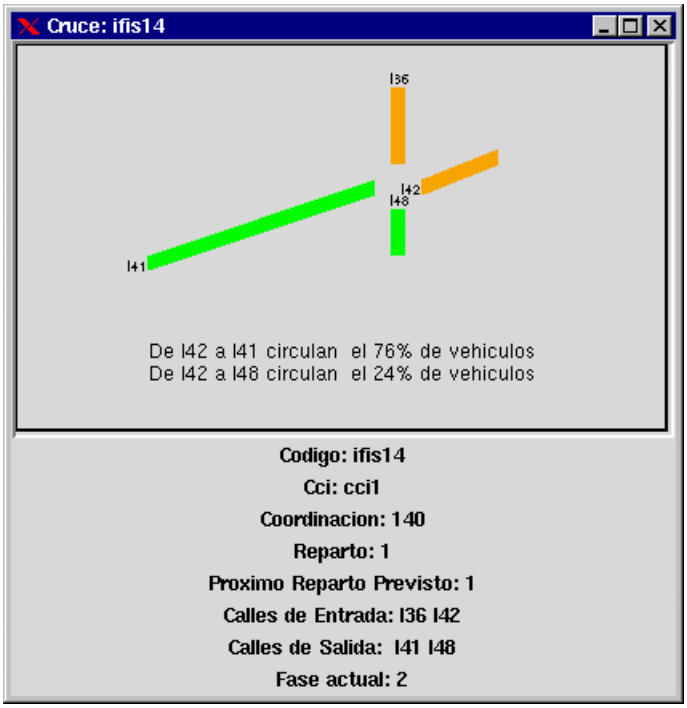
El estado cualitativo del tráfico en cualquier instante, calculado por el agente de preproceso, se visualiza mediante la siguiente ventana:



[1] http://www.scriptis.com

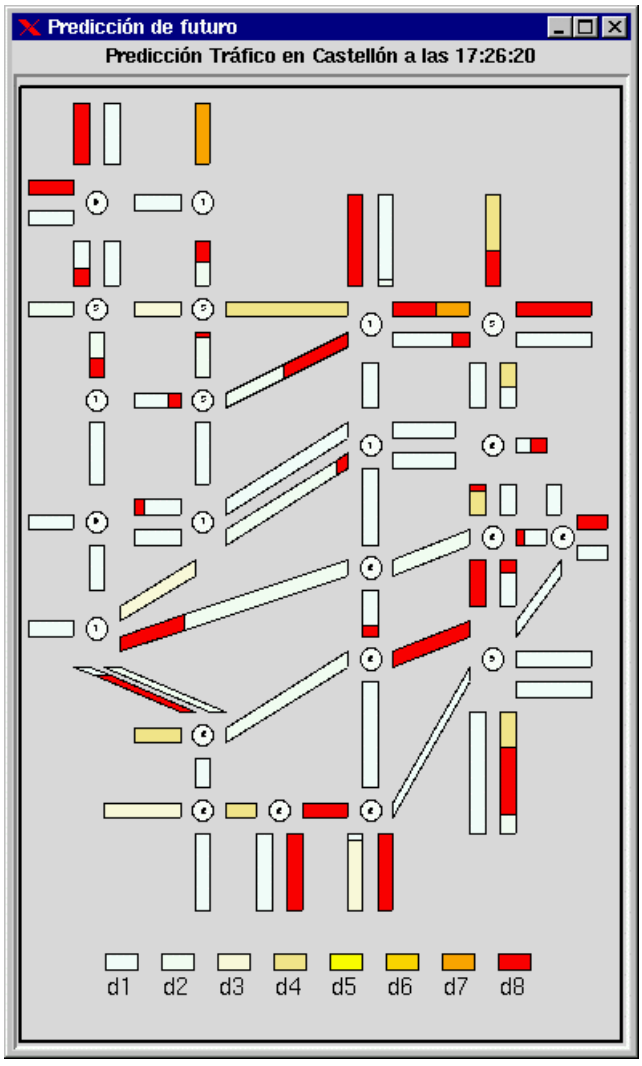
Cada rectángulo identifica a un segmento y cada una de las franjas de colores que lo constituye representa su estado cualitativo (desde *d1*, densidad nula de tráfico, hasta *d8*, congestión absoluta). La última línea de la ventana sirve como leyenda para interpretar cada una de las densidades cualitativas. El contenido de esta ventana evoluciona según se produzcan nuevos eventos en el sistema. El instante temporal exacto en el que se producen se muestra en la parte superior derecha de la ventana. Los círculos representan cruces y el número especificado en cada uno de ellos identifica la fase actual que se está ejecutando. Cada uno de los objetos del dominio representados puede ser identificado instantáneamente al seleccionarlo mediante el ratón. De esta forma si se selecciona el segmento *l51* se muestra la siguiente información:

El segmento seleccionado, con una longitud de 200 metros y 3 carriles, pertenece a la calle *San Luis*. Este segmento comienza en el cruce *ifis15* y finaliza en el cruce *ifis19*. Los estados cualitativos actuales se muestran en la última línea de la ventana. Cuando se deja de seleccionar el segmento esta ventana desaparece. Si se selecciona un cruce entonces sus características actuales se muestran en una ventana como la siguiente:



El cruce seleccionado es el *ifis14* que pertenece al cci *cci1*. El valor actual de coordinación es *140* y el vector de reparto que se está aplicando es el *1*. En este cruce son entradas los segmentos *l36* y *l42*. Los segmentos de salida son *l41* y *l48*. La fase actual que se está ejecutando es la *fase 2* en la que el *76%* de los vehículos circulan desde *l42* a *l41*, mientras que el resto, el *24%* circula desde *l42* a *l48*. En el instante en que se deje de seleccionar este cruce, su ventana asociada desaparece.

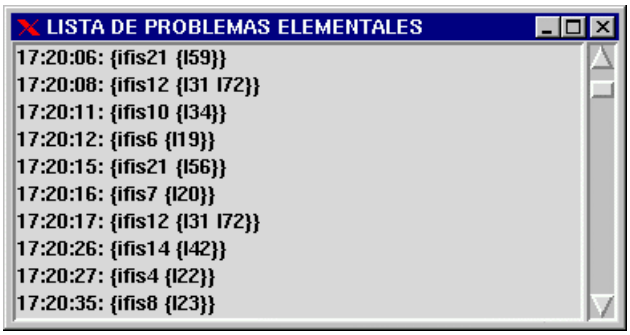
Cada una de las situaciones de tráfico previstas por el agente de predicción se muestran al usuario mediante el uso de un tipo de ventana parecida a la anterior:



Esta ventana tiene las mismas propiedades funcionales que la ventana asociada a los resultados del agente de preproceso. Todos los objetos representados en esta ventana pueden ser seleccionados para acceder a sus valores concretos.

Los resultados proporcionados por el agente de detección se muestran agrupados en 4 categorías, una por cada situación problemática definida. El acceso a cada una de estas categorías se realiza mediante la selección del botón adecuado de la siguiente ventana:

Si se selecciona el botón *Elementales* el interfaz muestra una ventana de texto en la que se especifican las situaciones problemáticas elementales detectadas:



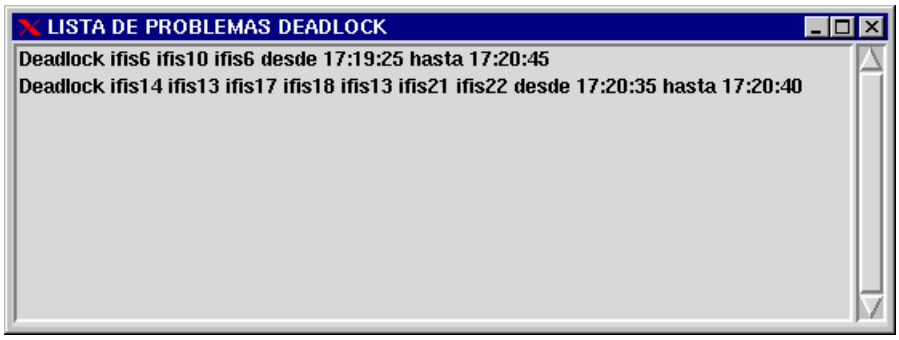
Cada una de sus líneas identifica el instante, el cruce y el segmento para el que se ha detectado la situación problemática elemental. La selección del botón *Temporales* hace que se muestre la siguiente ventana:

Cada una de sus líneas identifica el instante, cruces y valoración de su importancia de las situaciones problemáticas temporales detectadas mediante, al menos uno, ciclos sucesivos.

La selección del botón *Rutas Cong.* activa la visualización de la siguiente ventana:

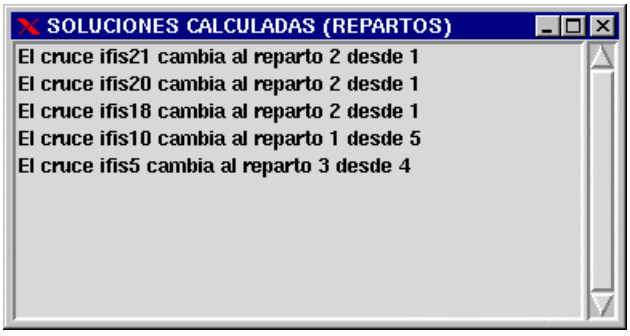
En ella se identifican cada una de las rutas congestionadas en la red durante un cierto intervalo temporal.

Por último, si se selecciona el botón *Deadlocks* se muestra la siguiente ventana:



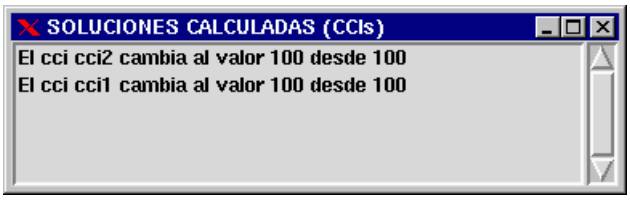
En esta ventana se identifican aquellas situaciones problemáticas detectadas en la red que forman un camino cerrado y que comparten un cierto intervalo temporal común de presencia.

Los resultados del agente de solución repartos se muestra en una ventana de la siguiente apariencia:



En ella sólo se muestran aquellos cruces para los que se propone un reparto que mejore las prestaciones actuales de tráfico medidas.

El agente de solución cci proporciona también sus resultados al agente de interfaz que los visualiza mediante una ventana del siguiente tipo:



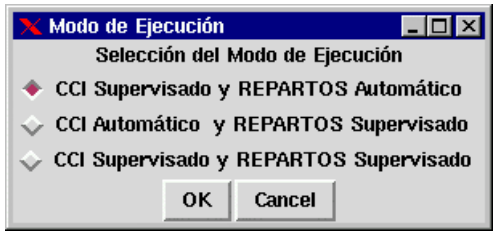
El agente interfaz proporciona el modo de finalizar la ejecución del sistema por medio de la siguiente ventana:

El botón *Salir* envía un mensaje a todos los agentes del sistema para que finalicen sus ejecuciones. Esta acción establece un mecanismo de sincronización por barrera para que todo el trabajo pendiente en cada agente se resuelva satisfactoriamente. El botón *Imprimir* sirve para obtener un fichero postcript de la ventana de simulación.

B.2 Ejecución de Acciones de Control Externas.

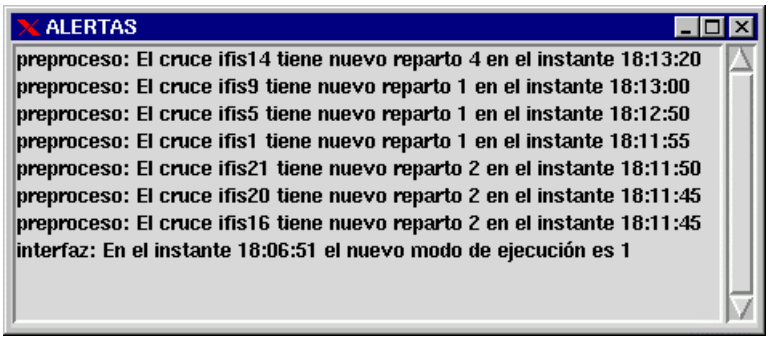
Cualquier cambio de funcionamiento que se quiera establecer sobre algún objeto del dominio se puede realizar por medio de la siguiente ventana:

El botón *Modo* permite cambiar el modo en el que se ejecutan las soluciones proporcionadas por los agentes de solución repartos y de solución cci. En el caso que se seleccione este botón, el interfaz muestra la siguiente ventana:



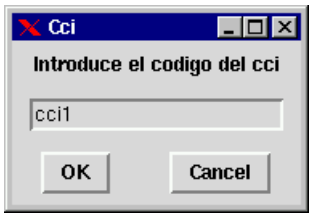
Sólo se puede elegir una de las tres opciones. En el caso que se seleccione otra opción diferente de la ejecutada hasta el momento, el agente de interfaz comunica esta nueva situación al resto de agentes. En el instante en que esta acción se pone en

funcionamiento, el agente de interfaz recibe la notificación que es mostrada al usuario mediante una ventana de alertas como la siguiente:



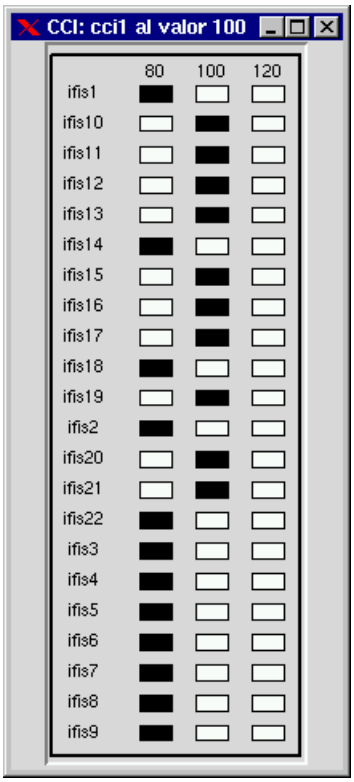
En esta ventana se observa que ha sido modificado el modo de funcionamiento del prototipo de forma que los resultados proporcionados por el agente de solución son ejecutados de forma automática en el instante adecuado.

El botón *CCIS* de la ventana *Menú* permite acceder a la información asociada a cada uno de los ccis. Su selección implica la identificación del cci al que se quiere acceder por medio de una ventana como la siguiente:

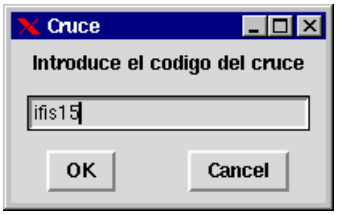


En el caso que el identificador escrito sea correcto, se muestra otra ventana en la que se puede modificar el valor de su duración:

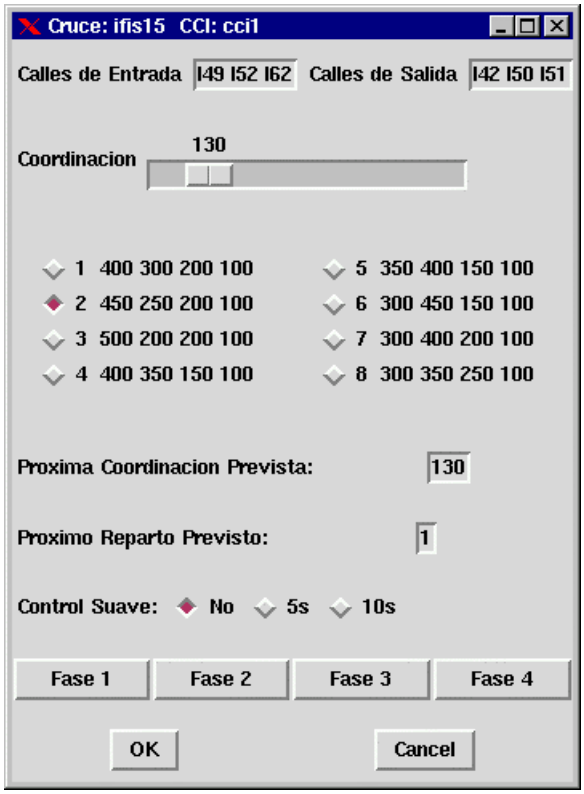
Si se selecciona un nuevo valor de duración, entonces el interfaz muestra una ventana en la que se comprueba el instante en que cada uno de los cruces que pertenecen a ese cci cambio de valor de duración:



La selección del botón *Cruces* de la ventana *Menú* implica la identificación del cruce al que se quiere acceder por medio de la siguiente ventana:



Si el identificador introducido es correcto, el agente de interfaz muestra una ventana cuyo contenido permite variar las características de funcionamiento del cruce seleccionado:

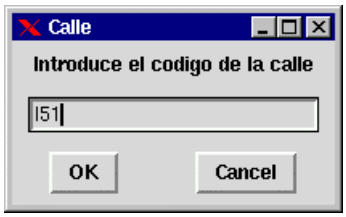


En esta ventana se identifican cuales son las calles de entrada de este cruce (*l49*, *l52* y *l62*) así como sus calles de salida (*l42*, *l50* y *l51*). El valor de coordinación actual es de *130* unidades, pero puede ser modificado moviendo la barra de deslizamiento con el ratón hacia la derecha o hacia la izquierda. El reparto actual que se está aplicando es el número *2*, cuyos valores se especifican en la segunda línea de los vectores de reparto. Sólo puede estar seleccionado uno de estos conjuntos en cualquier instante. Si se desea cambiar de reparto, basta con seleccionar otro de los vectores disponibles.

Cualquiera de las acciones de control que se realicen sobre un cruce puede ser ejecutada de diferentes formas. Esta característica se expone en la opción *Control Suave*. Si se selecciona la opción *No*, entonces la acción se ejecuta de una sola vez. La opción *5s* implica la ejecución de la acción en bloques de 5 segundos hasta que se ésta se complete mientras que la opción *10s* implica esta ejecución en bloques de 10 segundos.

Como parte de la información al usuario también se incluyen unos botones que permiten visualizar el tráfico permitido en cada una de las fases. Esta información se muestra en ventanas cuyo contenido es similar que el que se obtiene al seleccionar con el ratón un segmento de la ventana de resultados del agente de preproceso o de predicción.

Por último, la selección del botón *Calles* de la ventana *Menú* implica la identificación del segmento al que se desea acceder por medio de una ventana como la siguiente:



Si el identificador introducido es correcto, el agente de interfaz muestra una ventana de información sobre el segmento seleccionado (no se pueden realizar cambios de control sobre los segmentos):

El contenido de esta ventana es similar que el que se obtiene al seleccionar con el ratón un segmento de la ventana de resultados del agente de preproceso o de predicción.

Referencias Bibliográficas

[Aiello, 86] N. Aiello, *User-directed control of parallelism: The CAGE system*, KSL 86-31, Stanford University, Stanford, CA, 1986.

[Ambrosino, 91a] Ambrosino G., et. al., *Expert Systems Approach to Road Traffic Control*, In Concise Encyclopedia of Traffic and Transportation Systems, M. Papageorgiou, Pergamon Press, Oxford, 1991.

[Ambrosino, 91b] Ambrosino G., Bielli M., Boero M., Mastreta M. *A Blackboard Model for Traffic Control Operations*, In Advanced Telematics in Road Transport, Proc. of the DRIVE conference, Elsevier Pub., pp. 596-614, Brussels, 1991.

[Ambrosino, 97] Ambrosino G., Bielli M., Boero M., *Application of Artificial Intelligence and Expert Systems to Traffic Control*, In Advance Vehicles and Infrastructure Systems, John Wiley & Sons Ed., 1997.

[ATMS, 99] *Proceedings and Recomendations*. In Workshop on Models in Support of Advanced Traffic Management Systems. Palm Coast, Florida, May 16-19, 1999.

[BANCAIXA, 94] Plan de Promoción de la investigación, Fundació Caixa-Castelló 1994, *Diseño e implementación de una Arquitectura Paralela con Memoria Compartida para la Construcción de un Sistema Basado en el Conocimiento aplicado al Control de Tráfico Urbano*. Proyecto P14A94-21, Duración: 1994-1997.

[BarberF, 94] F. Barber, V. Botti, E. Onaindía, A. Crespo, *Temporal Reasoning in REAKT (An environment for Real-Time Knowledge-Based Systems),* AI Communications, Vol 7, Num 3, Sep, pp 175-202.

[Barber, 94] Barber F., Dondossola G., Berlanga R., Toledo F., Martín G. *A CLP framework for time-related reasoning in industrial applications*. Proc. 2nd Conf. Intelligent Systems Engineering, IEE-BCS.

[Barceló, 91] Barceló, J. *Traffic Management Systems* In Concise Encyclopedia of Traffic and Transportation Systems, M. Papageorgiou, Pergamon Press, Oxford, 1991

[Barceló, 94] Barceló J., Ferrer J.L., Grau R., *AIMSUN2 and the GETRAM Simulation Environment*, Internal Report, Departamento de

Estadística e Investigación Operativa, Universitat Politècnica de Catalunya, 1994.

[Bell, 90] Bell, M.C., Martin, P.T. *Use of Detector Data for traffic control strategies* Third international Conference on Road and Traffic control. May 1990 Conference publication Nr. 320 IEE pp86-90

[Bell, 91] Bell M. C., Scename G., Ibbetson L.J. *CLAIRE: An Expert System for Congestion Management*. In Advanced Telematics in Road Transport, Proc. of the DRIVE congerence, Elsevier Pub., pp. 596-614, Brussels, 1991.

[Biasini, 89] Biasini R., Forin A. *Paralellization of Blackboard architectures and the Agora System*. In V. Jagannathan, R. Dodhiawala and L. baum eds. Blackboard Architectures and Applications, Academic Press, 1989.

[Bretherton, 82] Bretherton R.D. and Raj G.I., *The use of SCOOT in low flow conditions*. In Traffic Engineering & Control, pp 574-576, 1982.

[Carriero, 90] Carriero Nicholas, Gelernter David. *How to write parallel programs. A first course.* Cambridge, Massachussets The MIT Press pub. 1990.

[Carver, 94] N. Carver and V. Lesser, *The Evolution of the Blackboard Control Architectures*, in: Expert Systems with Applications. Special Issue on The Blackboard Paradigm and It´s Applications, Vol 7, no 1, 1994.

[Castellón, 97] Memoria del proyecto "*Programa Director para el estudio del Tráfico y del Transporte en Castellón de la Plana*". Universidad Politécnica de Cataluña y Universitat Jaume I de Castellón.

[Corkill, 91] Corkill Daniel D. Blackboard Technology Group Inc. *Blackboard Systems* in: AI Expert 6(9):40-47, September 1991.

[Corkill, 97] Corkill Daniel D., *Countdown to Success: Dynamic Objects, GBB and RADARSAT-1*. Communications of the ACM, Pg 49-58, Vol. 40, No 5, May 1997.

[Craig, 89] I.D. Craig, *The CASSANDRA Architecture: Distributed Control in a Blackboard System*, Ellis Horwood Ltd., Chichester, UK, 1989.

[Craig, 93] I.D. Craig, *A New Interpretation of the Blackboard Metaphor*, Deparment of Computer Science, University of Warwick, 1993.

[Cuena, 88] Cuena, J. *The Qualitative Modelling of Axis-Based Flow Systems: Methodology and Examples*. In Proc. of European Conference onArtificial Intelligence, 1988, Munchen.

[Cuena, 89] Cuena, J. *El Sistema Experto AURA para control de Tráfico en accesos urbanos*. In Universidad Politécnica de Madrid, 1989.

[De Kleer, 84] De Kleer J., Brown J.S. *Qualitative Reasoning about physical systems*. Artificial Intelligence, 1984.

[Decker, 94] Decker K., Garcey A., Humphrey M., Lesser V. *A Real-Time Control Architecture for an Approximate Processing Blackboard System*. Tech. Report Department of Computer Science, University of Massachussets April, 1994.

[DT, 95] *Detection Technology for IVHS –Final Report Addedum*, Federal Highway Administration Contract DTFH61-91-C-00076, U.S. Department of Transportation, Washington D.C., 1995.

[Duncan, 96] Duncan G., *Paramics Traffic Simulation Ltd*, Dirección de internet: http://www.paramics.com

[Durfee, 94] Durfee Edmun H., Rosenschein Jeffrey S. *Distributed Problem Solving and Multi-Agent Systems: Comparations and Examples*, EECS Department, University of Michigan, May, 1994.

[Durkin, 96] Durkin John, *Expert Systems. Design and Development*, Prentice Hall, Englewood Cliffs, New Jersey, 2nd edition, 1996.

[EQUATOR, 90] Moreno S., Toledo F., Martín G. *A Qualitative Simulator for Event Abstraction. The (λ, t) Formalism*, ESPRIT project 2049. July, 90.

[EQUATOR, 91] Equator Team, *Formal Specification of the GRF and CRL*, Deliverable D123-1 ESPRIT project 2049.

[EQUATOR, 94] *EQUATOR Final Report*, ESPRIT p2409.

[Erman, 80] L.D. Erman and F. Hayes-Roth, V.R. Lesser, and D.R. Reddy, *The Hearsay-II speech-understanding system: integrating knowledge to resolve uncertainty*, ACM Computing Surveys, 12(2):213-253, 1980.

[ETRA, 90] ETRA, SYCECA *Illustration of an Urban Traffic Application in CRL*. Informe interno del proyecto EQUATOR, Julio, 1990.

[Foraste, 86] Foraste B., Scemama G., *Surveillance and congested traffic control in Paris by Expert System*, 2nd IEE Conference on Road Traffic Control, London, 1986.

[Forbus, 84] Forbus, K.D. *Qualitative Process Theory*. Artificial Intelligence, 25, pag 85-108, 1984.

[Forbus, 88] Forbus K.D. *Qualitative Physics: past, present and future*. In Exploring Artificial Intelligence, Morgan Kauffan Eds, 1988.

[Forgy, 82] Forgy Charles, *Rete: A Fast Algorithm for the Many Pattern/Many Objects Pattern Match Problem*. Artificial Intelligence, 19, pp 17-37, 1982.

[Gabard, 91] *Car Following Models*, Concise Encyclopedia of Traffic and Transportation Systems, M. Papageorgiou, Pergamon Press, Oxford, 1991.

[García, 96a] García L.A., *Arquitecturas de Pizarra. Definiciones, Antecedentes, Evolución e Implementaciones*. Trabajo de 3 créditos e Informe Técnico DI 04-07/96, Departamento de Informática, Universitat Jaume I, Castellón, 1996.

[García, 96] García L.A., Toledo F., Moreno S., Bonet E. *An Expert System with Deep Knowledge for Urban Traffic Control based on a Parallel Architecture*. Proc. of EXPERSYS-96, Expert Systems Applications & Artificial Intelligence, J. Zarka, E. Mercier-Laurent, D.L. Crabtree, Narasipuram M. editors, I.I.T.T. International, París, 1996.

[García, 97] García L.A., Toledo F. *Cooperative Solving of Urban Traffic Problems*, Proc of Industrial And Engineering Applications of Artificial Intelligence and Expert Systems IEA/AIE 97, Atlanta, Georgia, USA, June 1997, pp. 277-283.

[García, 98] García L.A., Toledo F. *A Centralised Hierarchical Task Scheduler for an Urban Traffic Control System Based on a Multiagent Architecture*, Proc of Industrial And Engineering Applications of Artificial Intelligence and Expert Systems IEA/AIE 98, Benicassim, Spain, June 1998, pp. 173-182.

[García, 99] García L.A., Toledo F. *A Deep Knowledge Expert System for Urban Traffic Control*. In Intelligent Techniques in Traffic Control, Jain Lakhim eds, (admitted for publication, 1999).

[García, 00a] García L.A., Toledo F. *Space/Temporal Urban Traffic Bad Regulation Problem Identification Using Qualitative Data and Temporal Intervals*, Proc. of 2000 International Conference on Artificial Intelligence, IC-AI'2000 June 26-29, Las Vegas, Nevada, USA.

[García, 00b] García L.A., Toledo F. *Urban Traffic Incident Detection Using Qualitative Data and Temporal Intervals*, Proc. of 2000 International Conference on Artificial Intelligence, IC-AI'2000 June 26-29, Las Vegas, Nevada, USA.

[Garvey, 93] A. Garvey, M. Humphrey and V. Lesser, *Task Interdependencies in Design-to-time Real-time Scheduling*, in Proceedings of the Eleventh National Conference on Artificial Intelligence, 1993.

[Garvey, 93] Garvey A., Lesser V. *Design-to-time real-time scheduling*. IEEE Transactions on Systems, Man and Cybernetics, 1993.

[Garvey, 94] A. Garvey and V. Lesser. *Design-to-time Real-Time Scheduling*, in IEEE Transactions on Systems, Man and Cybernetics. Special Issue on Planning, Scheduling and Control, Vol 23:6, 1994.

[Gerhard, 99] Gerhard Weiss (ed), *Multiagent Systems. A Modern Approach to Distributed Artificial Intelligence*, MIT Press, 1999.

[Giarratano, 98] Joseph Giarratano, Riley Gary, *Expert Systems. Principles and Programming, Third Edition*. PWS Publishing Company, 1998.

[Gipps, 86] *A Model for the Structure of Lane-Changing Decisions*, Transportation Research, 20 B, páginas 403-414, 1986.

[Green, 98] Greennough , John C. and Kelman, W. Les. *Metro Toronto Scoot Traffic Adaptative Control Operation*, ITE Rewiew, August 1998.

[Hayes-Roth, 84] B. Hayes-Roth, *BB1: an architecture for blackboard systems that control, explain, and learn about their own behaviour*, STAN-CS-84-1034, Stanford University, Stanford, CA, 1984.

[Hayes-Roth, 85] Hayes-Roth Frederick, *Rule Based Systems*, John Wiley and Sons Inc., 1985.

[HCM, 85] *Highway Capacity Manual*, Special Report No. 209, National Research Council, Washington D.C., 1985.

[Hentenryck, 89] Van Hentenryck, Pascal, *Constraint satisfaction in logic programming*, Cambridge, Massachussets. MIT Press cop. 1989

[Irgens, 97] Irgens M., Krogh C., Traettebers H., *Model Based Collaboration Architectures for Traffic Control*, In Advance Vehicles and Infrastructure Systems, John Wiley & Sons Ed., 1997.

[Jackson, 90] Jackson Peter, *Introduction to Expert Systems. Second Edition*, Addison-Wesley Publishing Company, 1990.

[KITS, 96] DRIVE II R&D Programme, *DRIVE Project V2039, Deliverable No. 13, Final Assessment of Applications: Tested Methodologies and Models*, 1996.

[Kowalski, 86] Kowalski R., Sergot M. *A logic-based calculus of events*. New Generation Computers, 4, 1986.

[Kuipers, 86] Kuipers B. *Qualitative Simulation.* In Artificial Intelligence 29, 1986.

[Kuipers, 94] Kuipers B. *Qualitative Reasoning. Modeling and Simulation with Incomplete Knowledge*. The MIT Press, Massachusetts Institute of Technology. Cambridge, Massachusetts 02142, 1994.

[Lear, 87] Lear D.W., Bell M.C. *An envestigation of the differences between British and German traffic control policies* NVTRG Research report 1, 1987.

[Lesser, 88] Lesser V., Pavlin J. Durfee E. *Approximate processing in real time problem solving* AI Magazine, 9, Pp 49-61, Spring 1988.

[Lieberman, 97] Lieberman Edward, Rathi Ajay K., *Traffic Simulation*, Chapter 10 of *Traffic Flow Theory. A state of the art report*, Transportation Research Board, Federal Highway Administration, Washington D.C., 1997.

[Long, 95] Q. Long and V. Lesser, *A Heuristic Real-Time Parallel Scheduler Based on Task Structures*, Technical Report 95-92, Department of Computer Science, University of Massachussets, 1995.

[Maes, 94] Maes P., *Agents that Reduce Work and Information Overload*, Communications of the ACM, pp 31-40, 1994.

[Marriot, 98] Marriot Kim, Stuckey P.J., *Programming with constraints. An introduction.* Cambridge, Massachussets. MIT Press cop. 1998.

[Martín, 88] Martín G., Toledo F., Sanchez-Barcaiztegui V., de la Rosa J.C., *Vanesa: Una aplicación de los sistemas basados en el*

conocimiento a la gestión del tráfico urbano de Valencia, Tecnotrafic 88, Madrid, 1988.

[McDonald, 91] *Road Traffic Control: TRANSYT and SCOOT*, MCDonald M. and Hounsell N.B. Concise Encyclopedia of Traffic and Transportation Systems, M. Papageorgiou, Pergamon Press, Oxford, 1991.

[Michalopoulos,91] Michalopoulos P.G., Ping Yi, D.E. Beskos and A.S.Lyrintzis, *Continuum Modelling of Traffic dynamics*, Proc. of the 2nd International Conference on Application of Advanced Technologies in Transportation Engineering, August, 1991, Minneapolis, Minnesota, pp 36-40.

[Moreno, 90] Moreno S., Toledo F., Rosisch F., Martín G. *VANESA una aplicación de los sistemas basados en el conocimiento a la gestión de tráfico urbano en Valencia*. Proc II Congreso Iberoamericano e IA pp. 549-568, LIMUSA S.A.

[Moreno, 93] Moreno S., Toledo F., Rosich F., y Martín G. *Qualitative Simulation for Temporal Reasoning in Urban Traffic Control*. Qualitative Reasoning and Decision Technologies, N. Piera Carreté and M.G. Singh Editors, CIMNE, Barcelona, 1993.

[Moreno, 93] Moreno S. *Formalización del Razonamiento Cualitativo y Temporal para el Tratamiento de Problemas Asociados al Control de Sistemas Físicos con Representación Multidimensional*. PhD Thesis, 1993.

[Moreno, 96] Moreno S., Toledo F., Bonet E., Martín G. *Qualitative Modelling for Building Sensor Independent Knowledge Bases in Control Applications*. Proc of EXPERSYS-96 Expert Systems Applications & Artificial Intelligence, París, 1996.

[Müller, 97] Müller Jörg P., Woodbredige Michael J., Jennings Nicholas R. Eds. *Intelligent Agents III. Agent Theories, Architectures and Languages*. Springer-Verlang, 1997.

[MUTC, 96] *Manual on Uniform Traffic Control Devices for Streets and Highways*, Federal Highway Administration, Washington D.C. 1996.

[Newell, 62] A. Newell, *Some problems of the basic organization in problem-solving programs*. M.C. Yovits, G.T. Jacobi, and G.D. Goldstein, eds. Proceedings Second Conference on Self-Organizing Systems, pp. 393-423. Spartan Books, 1962.

[Nierstrasz, 92] Nierstrasz O. referee in *Model Based Architectures for Traffic Control*. M. Irgens, C. Krogh, H. Traettebers In Advance Vehicles and Infrastructure Systems, Jhon Wiley & Sons Ed., 1997

[Nii, 86] H. P. Nii, *Blackboard Systems: the blackboard model of problem solving and the evolution of blackboard architectures*, AI Magazine, 7(2):38-53, 1986.

[Nii, 88] H.P. Nii, E. A. Feigenbaum, J.J. Anton, and A.J. Rockmore, *Signal to symbol transformation: HASP/SIAP case study*, in: R.S. Engelmore and A.J. Morgan, eds., Blackboard Systems, chapter 6, pp. 135-157, Addison-Wesley, Reading, MA, 1988.

[Papacostas, 93] Papacostas C.S. & Prevedouros P.D.. *Transportation Engineering and Planning*. Prentice Hall, Second Edition, New Jersey 1993.

[Petty, 95] Petty, Karl, Noeimi Hisham et all, *The Freeway Service Patrol Evaluation Project: Database, Support, Programs and Accesibility*, California PATH Program, University of California, Berkeley, 1995.

[RTUA, 87] *Roads and Traffic in Urban Areas*, The institution of Highways and Transportation & The Department of Transport, 1987.

[Rice86] J.P. Rice, *POLIGON, a system for parallel problem solving*, KSL 86-19, Stanford University, Stanford, CA, 1986.

[Robertson, 87] Robertson G.D., *Handling congestion with SCOOT*. In Traffic Engineering & Control, pp 228-232, 1987.

[RT-TRACS, 98] PB Farradyne Inc., *RT-TRACS Adaptative Control Algorithms, VFC-OPAC*. TRB Commitee, Adaptative Traffic Control Workshop, Pacific Grove, CA, July, 1998.

[Russell, 95] Russell S., Norving P., *Artificial Intelligence –A Modern Approach*, Prentice Hall, New Jersey, USA, 1995.

[SCATS, 98] Transcore Corp., *SCATS Adaptative Traffic System*, TRB Commitee, Adaptative Traffic Control Workshop, Pacific Grove, CA, July, 1998.

[SCOOT, 98] Siemens, *SCOOT version 3.1*. TRB Committee, Adaptative Traffic Control Workshop, Pacific Grove, CA, July 1998.

[Shanahan, 90] Shanahan Murray. *Representing Continuous Change in The Event Calculus*. In Proceedings ECAI 90, pp 598-603.

[Shapiro, 92] S. C. Shapiro, *Encyclopedia of Artificial Intelligence*. Ed. John Wiley & Sons, cop 1992. New York.

[Singh, 98] Singh *Intelligent Agents VI. Agent Theories, Architectures and Languages*. Springer-Verlang, 1997.

[SMARTEST, 99] *Simulation Modelling Applied to Road Transport European Scheme Tests, Rewiew of Micro-simulation Models*. SMARTEST Deliverable 3, Institute for Transport Studies, University of Leeds, April 1999.

[Sybille, 87] Sybille, E. *System Experts et Temp Reel* Genie Logiciel, nº 8, 1987, pp 62-70.

[TCSH, 96] *Traffic Control Systems Handbook*, Publication No. FHWA-SA-95-032, Federal Highway Administration, February 1996.

[Tel, 94] Gerard Tel, *Introduction to Distributed Algorithms*. Cambridge University Press, USA, 1994.

[Toledo, 90] Toledo F., Moreno S., Rosich F, Barrachina P. y Martin G. *Análisis de dos modelos basados en confluencias para el razonamiento cualitativo en una red de tráfico urbano*.

[Toledo, 91] Toledo F., et. al.. *Qualitative Simulation in Urban Traffic Control: Implementation of Temporal Features*. In IMACS International Wotkshop on Decision Support Systems and Qualitative Reasoning, pp 196-199, 1991.

[Toledo, 93] Toledo F., Moreno S., Martín G. *An Application of (λ, t) formalism for Handling Systems with Continuous Change: The Many Tanks Problem Case-Study*. Qualitative Reasoning and Decision Technologies, N. Piera Carreté and M.G. Singh Editors, CIMNE, Barcelona, 1993.

[Toledo, 96] Toledo F., Moreno S., Bonet E., Martín G. *Using Constraint Technology for Predictive Control of Urban Traffic Based on Qualitative and Temporal Reasoning*. Proc. of the Ninth International Conference IEA/AIE-96, Fukuoka, Japan, June, 1996.

[TRANSYT, 87] *TRANSYT-7F: Users Manual*, Federal Highway Administration, U.S. Department of Transportation, Washington D.C., July, 1987.

[Valdés, 88] Valdés, Antonio. *Ingeniería de Tráfico*. Editorial Bellisco, Tercera Edición, Madrid 1988.

[Velthuijsen, 92] H. Velthuijsen, *The Nature and Applicability of the Blackboard Architecture*. PTT Research, 1992.

RESUMEN

La presente memoria de tesis expone los resultados obtenidos en la consecución de los siguientes tres objetivos: 1) La identificación e implementación de aquellas tareas asociadas al control de tráfico urbano que puedan utilizar mecanismos de razonamiento profundo como herramientas fundamentales para su resolución. El modelo seleccionado para explicar el comportamiento del tráfico urbano es un modelo cualitativo entre cuyas características destaca su bajo coste computacional temporal y espacial. Este modelo se aplica a las tareas de preproceso de datos, detección y diagnosis de problemas, predicción de tráfico y generación de soluciones. 2) La definición de una arquitectura funcional con restricciones de ejecución en tiempo real suave en la que integrar el conjunto de tareas desarrolladas. Cada una de estas tareas es ejecutada por un componente especializado, denominado *agente*, compuesto por tres capas funcionales: protocolos de comunicación, métodos (especialización del agente) y espacio de trabajo. En esta arquitectura se cumple que las prestaciones en la ejecución de cada agente no se encuentran comprometidas por la incorporación en el sistema de otros agentes, o por las interacciones entre los agentes activos en el sistema. 3) La implementación del prototipo en una arquitectura computacional de altas prestaciones computacionales, un sistema *Beowulf*.