Universitat de Girona

DYNAMIC TASK ALLOCATION AND COORDINATION IN COOPERATIVE MULTI-AGENT ENVIRONMENTS

Silvia Andrea SUÁREZ BARÓN

ISBN: 978-84-694-2593-0
Dipòsit legal: GI-373-2011
http://hdl.handle.net/10803/7754

UNIVERSITAT DE GIRONA

Dynamic Task Allocation and Coordination in Cooperative Multi-Agent Environments

by

Silvia Andrea Suárez Barón

Advisors

Dr. Josep Lluís de la Rosa and **Dr. Christian G. Quintero M.**

Doctoral Programme in Technology

Doctoral Thesis

A thesis submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy (major subject: computer science) at the University of Girona

2010

Dynamic Task Allocation and Coordination in Cooperative Multi-Agent Environments

A thesis submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy (major subject: computer science) at the University of Girona

Author:

Silvia Andrea Suárez Barón

Advisor:

Dr. Josep Lluís de la Rosa

Advisor:

Dr. Christian G. Quintero M.

Programa de Doctorat en Tecnologia

Departament d'Enginyeria Elèctrica, Electrònica i Automàtica

To my family

Abstract

Distributed task allocation and coordination have been the focus of recent research in last years and these topics are the heart of multi-agent systems. Agents in these systems need to cooperate and consider the other agents in their actions and decisions. Moreover, agents may have to coordinate themselves to accomplish complex tasks that need more than one agent to be accomplished. These tasks may be so complicated that the agents may not know the location of them or the time they have before the tasks become obsolete. Agents may need to use communication in order to know the tasks in the environment, otherwise, it may take a long time to find the tasks into the scenario. Similarly, the distributed decision-making process may be even more complex if the environment is dynamic, uncertain and real-time.

In this dissertation, we consider constrained cooperative multi-agent environments (dynamic, uncertain and real-time). In this regard, we propose two approaches that enable the agents to coordinate themselves. The first one is a semi-centralized mechanism based on combinatorial auction techniques and the main idea is minimizing the cost of assigned tasks from the central agent to the agent teams. This algorithm takes into account the tasks' preferences of the agents. These preferences are included into the bid sent by the agent. The second one is a completely decentralized scheduling approach. It permits agents schedule their tasks taking into account temporal tasks' preferences of the agents. In this case, the system's performance depends not only on the maximization or the optimization criterion, but also on the agents' capacity to adapt their schedule efficiently.

Furthermore, in a dynamic environment, execution errors may happen to any plan due to uncertainty and failure of individual actions. Therefore, an indispensable part of a planning system is the capability of replanning. This dissertation is also providing a replanning approach in order to allow agents recoordinate his plans when the environmental problems avoid fulfil them.

All these approaches have been carried out to enable the agents to efficiently allocate and coordinate all their complex tasks in a cooperative, dynamic and uncertain multi-agent scenario. All these approaches have demonstrated their effectiveness in experiments performed in the RoboCup Rescue simulation environment.

Resum

La coordinació i assignació de tasques en entorns distribuïts ha estat un punt important de la recerca en els últims anys i aquests temes són el cor dels sistemes multi-agent. Els agents en aquests sistemes necessiten cooperar i considerar els altres agents en les seves accions i decisions. A més a més, els agents han de coordinar-se ells mateixos per complir tasques complexes que necessiten més d'un agent per ser complerta. Aquestes tasques poden ser tan complexes que els agents poden no saber la ubicació de les tasques o el temps que resta abans de que les tasques quedin obsoletes. Els agents poden necessitar utilitzar la comunicació amb l'objectiu de conèixer la tasca en l'entorn, en cas contrari, poden perdre molt de temps per trobar la tasca dins de l'escenari. De forma similar, el procés de presa de decisions distribuït pot ser encara més complexa si l'entorn és dinàmic, amb incertesa i en temps real.

En aquesta dissertació, considerem entorns amb sistemes multi-agent amb restriccions i cooperatius (dinàmics, amb incertesa i en temps real). En aquest sentit es proposen dues aproximacions que permeten la coordinació dels agents. La primera és un mecanisme semi-centralitzat basat en tècniques de subhastes combinatòries i la idea principal es minimitzar el cost de les tasques assignades des de l'agent central cap als equips d'agents. Aquest algoritme té en compte les preferències dels agents sobre les tasques. Aquestes preferències estan incloses en el bid enviat per l'agent. La segona és un aproximació d'scheduling totalment descentralitzat. Això permet als agents assignar les seves tasques tenint en compte les preferències temporals sobre les tasques dels agents. En aquest cas, el rendiment del sistema no només depèn de la maximització o del criteri d'optimització, sinó que també depèn de la capacitat dels agents per adaptar les seves assignacions eficientment.

Adicionalment, en un entorn dinàmic, els errors d'execució poden succeir a qualsevol pla degut a la incertesa i error de accions individuals. A més, una part indispensable d'un sistema de planificació és la capacitat de re-planificar. Aquesta dissertació també proveeix una aproximació amb replanificació amb l'objectiu de permetre als agent re-coordinar els seus plans quan els problemes en l'entorn no permeti la execució del pla.

Totes aquestes aproximacions s'han portat a terme per permetre als agents assignar i coordinar de forma eficient totes les tasques complexes en un entorn multi-agent cooperatiu, dinàmic i amb incertesa. Totes aquestes aproximacions han demostrat la seva eficiència en experiments duts a terme en l'entorn de simulació RoboCup Rescue.

Acknowledgements

First of all, all my gratitude to my advisors Dr. Josep Lluís de La Rosa and Dr. Christian Quintero for their advice during the development of this thesis. I also thank Dra. Beatriz López and thanks to professor John Collins for their valuable help and suggestion about my work during my research stays at the University of Minnesota (U.S.A).

Thanks to the people from ARLab who share some time with me while I was doing this PhD.

Special thanks to all my friends both abroad and in Girona who supported me during these years. My very special gratitude goes to Cheli, Jorge, Annie, Edwin, Yudit, Liliana, Maite, Ana María, Sandra Pilar and Sandra Milena, I greatly appreciate their advice and especially, their patience.

Finally, special thanks to my family and my boyfriend Javi. They have always believed in me, a priceless motivation which I truly appreciate.

Contents

List of Figures

List of Tables

CHAPTER 1
Introduction

This chapter provides an introduction to the work presented in this thesis. Particularly, the motivation, the objectives and the main contributions are briefly described. Finally, the chapter concludes with an overview of the structure and contents of the thesis.

1.1 OVERVIEW

The multi-agent systems are dynamic environments that are composed of intelligent entities called agents. Multi-agent systems let us to study many aspects of dynamic environments and are very similar to real life systems. Using multi-agent systems, we can study how entities (humans, robots, software agents, artifacts, network nodes) interact with each other. The agents' interactions can be either cooperative or selfish. That is, the agents can share a common goal, or they can pursue their own interests. In addition, for a single agent, it is a hard work to achieve the objectives of the system. For this reason, agents are grouped to form multi-agent systems and thus be able to take full advantage of teamwork.

Distributed task allocation and coordination have been the focus of recent research in last years and these topics are the heart of multi-agent systems. Agents have to perceive their environment, reason on these perceptions and choose some actions in order to achieve their goals. These actions' choices are complicated by the fact that agents are not alone into the environment. They need to cooperate and consider the other agents in their actions and decisions. Moreover, agents may have to coordinate themselves to accomplish complex tasks that need more than one agent to be accomplished. These tasks may be so complicated that the agents may not know the location of them or the time they have before the tasks become obsolete. Agents may need to use communication in order to know the task in the environment, otherwise, it may take a long time to find the task into the scenario. Similarly, the distributed task allocation process may be even more complex if the environment is dynamic, uncertain and real-time. Dynamic and uncertain means that the environment is in constant evolution and that an agent cannot know with certainty how the world will evolve or how its actions will impact the world. And real-time means that the agent has to respect some time constraints when making its decisions.

In this sense, it is very important to decide which coordination mechanism to use in order to synchronize the actions of agents. This thesis addresses the challenge of designing these mechanisms to enable agents are interacting in dynamic environment to choose the

best actions or tasks to perform within the environment. In this context, it proposes some approaches to enable the agents to make suitable decisions and to coordinate themselves in order to accomplish their tasks as efficiently as be possible. Briefly, this thesis proposes:

- A semi-centralized task allocation algorithm called Combinatorial Auctions for Task Allocation (CATA). This algorithm uses combinatorial auctions to allocate tasks among agents and has some features such as: it takes into account the task and agent capabilities to the bid configuration and it has a re-adjustment phase to allow all the agents be allocated in each auction round;
- A completely distributed task selection algorithm called Sequencing Algorithm for Task Allocation (SATA). Using this algorithm, agents choose the tasks to be performed in a decentralized manner. To do this, agents take into account the priorities and time constraints of tasks;
- A mechanism to allow agents make replanning of tasks when they face problems which make them unable to follow the initial plan.

1.2 Motivation

Disaster management has become an important issue in the last few years due to the large number of disasters occurring such as the 2010 Haiti Earthquake, the Chilean earthquake and other recent catastrophes. Disaster management involves coordinating a large number of emergency responders to rescue either people or infrastructure in possibly hazardous environments where uncertainty about events is predominant [80]. These catastrophes result in the death of many people and the entire destruction of cities and towns.

A disaster environment is a dynamic environment with unpredictable situations. The kinds of rescue activities that take place depend on the kind of disaster that has occurred and can range from rescuing victims, to extinguishing forest fires, re-establishing urban services, cleaning beaches, etc. Rescue resources should be assigned in such a way as to accomplish the various tasks required for optimal recovery from the disaster.

These unfortunate events inspired the research in this socially significant domain and a group of research developed a computer program for simulating earthquakes. The prototype version was called the 'RoboCup Rescue Simulator System' [44]. This simulator permits to analyze ways in which rescue operations are carried out in the seconds after that an earthquake has happened. Some intelligent agents were introduced in this environment to play an active role in the simulation and to have them operate either autonomously or as members of a team to deal with the effects of the disaster.

This test-bed environment has motivated us and it has been used to test our approaches. The RoboCup Rescue environment consists of a simulation of an earthquake happening in a city. The goal of the agents (representing firefighters, policemen and ambulance teams) consists in minimizing the damages caused by a big earthquake, such as civilians buried,

buildings on fire and blocked roads. The RoboCup Rescue simulation environment has all the complex characteristics mentioned previously and it is thus a complex test-bed for cooperative multi-agent systems.

Artificial intelligence and new information technologies should be a support for such important issues. In this sense, this thesis discusses the use of new technologies based on scheduling and task allocation to decrement damage caused by natural catastrophes.

1.3 Objectives of the research

The main objective is to develop algorithms and mechanisms to efficient task allocation and coordination in dynamic and uncertainty environments. Particularly, we focus on constrained environments such as a rescue scenario. In this kind of scenarios it is important to take into account temporal constraints because every spent minute difficult the rescue operation. The proposed algorithms must take advantage of the decentralized and centralized features of this kind of environments and to be real-time in order to allow agents face with changes in the scenario. To accomplish this objective, this thesis deals with the following specific objectives:

1. To apply market based techniques, particularly auctions to improve the task allocation between rescue agents. This mechanism is dealing with centralized task allocation using direct supervision.

2. To apply scheduling techniques such as sequencing techniques to improve task selection among rescue agents. This algorithm takes into account temporal constraints and priorities, for instance, the due-date of the task. In addition, it is dealing with decentralized task coordination using mutual adjustment.

3. To design a rescheduling and replanning algorithm in order to allow agents face with problems within the environment which are beyond of their control (for instance, the common problem of blocked paths and roads).

1.4 Thesis outline

Following is a general description of the contents of this dissertation. This doctoral thesis is organized in the next 6 chapters:

Chapter 1 presented the introduction, motivation and objectives of this dissertation.

Chapter 2 gives a overview of the background information regarding distributed and multi-agent systems, agent technology, task oriented domains, and resource allocation topics which is required to carry out the approach presented in chapter 4 and 5. In addition, the RoboCup Rescue test bed is presented in this chapter.

Chapter 3 presents a survey of the most relevant work related to the approaches tackled in this thesis.

Chapter 4 describes the formalization of the tasks allocation and coordination approach. The new algorithms and mechanisms are described in this chapter.

Chapter 5 presents the implementation, experimentation and results of the approach on the test bed of the RoboCup Rescue which has been proposed in chapter 4.

Chapter 6 discusses and analyzes the results, provides the conclusions and contribution of this thesis and outlines the most promising directions for the future work.

CHAPTER 2
Background Information

This chapter introduces and reviews general concepts of agents, multi-agent systems, coordination and cooperation, task-oriented domain and resource allocation.

2.1 DISTRIBUTE ARTIFICIAL INTELLIGENCE AND MULTI-AGENT SYSTEMS

According to [115], Distributed Artificial Intelligence (DAI) is the study, construction, and application of multi-agent systems, that is, systems in which several interacting, intelligent agents pursue some set of goals or perform some set of tasks. In others words, research in DAI addresses the problem of designing automated intelligent systems which interact effectively.

DAI involves studying a broad range of issues related to the distribution and coordination of knowledge and actions in environments involving multiple entities. According to [16], there are several reasons to choose a distributed AI approach. The first one being the necessity to treat distributed knowledge in applications that are geographically dispersed, such as sensor networks, air-traffic control, or cooperation between robots. In addition, DAI can be used in large complex applications. The second reason is to attempt to extend man-machine cooperation, and the third is that DAI represents a new perspective in knowledge representation and problem solving, because it provides richer and realistic scientific applications.

The DAI field is divided into two research lines: Distributed problem solving (DPS) and research into multi-agent systems. In DPS work, the emphasis is on the problem and how to get multiple intelligent entities (programmed computers) to work together to solve it in an efficient manner. In this sense, many computer applications are open distributed systems in which the (very many) constituent components are spread throughout a network, in a decentralised control regime, and which are subject to constant change throughout the system's lifetime [79].

In multi-agent systems, components are called agents and have the same properties as in real communities where the agents need to cooperate in order to achieve their goals and the goals of the communities involved [51]. Agents are autonomous and may be homogeneous or heterogeneous. Autonomy is defined as the agent's ability to make its own

decisions about what activities to do, when to do them, what type of information should be communicated and to whom, and how to assimilate the information received. Inversely to studies on DPS, in multi-agent sytems, agents must reason out the coordination problem among themselves.

Multi-agent systems approaches include different application domains as for example: industrial procurement [34, 119], manufacturing systems [63, 49], public transport [37], logistics [19], the grid [42], network routing [121], airport traffic management [111], earth observation satellites [23], planetary rover path planning [96], and crisis management [102, 105]. In all of these cases, however, there is a need to have autonomous components (agents) that act and interact in flexible ways in order to achieve their design objectives in uncertain and dynamic environments [79].

2.2 What is an intelligent agent?

In recent years, agents are one of the most prominent and attractive technologies in computer science at the world. As is to be expected from a fairly young area of research, there is not yet a universal consensus on the definition of an agent [71]. However, the Wooldridge and Jennings definition (See below) is increasingly adopted:

"An agent is a computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to meet its delegated objectives" [116].

Wooldridge distinguishes between an agent and an intelligent agent, which is further required to be reactive, proactive and social.

Another definition, according to Luck states that:

"An agent is a computer system that is capable of flexible autonomous action in dynamic, unpredictable, generally multi-agent domains" [56].

In addition, the agent definition is stated by Weiss in his book as it follows:

"Agents are autonomous, computational entities, which perceive their environment through sensors and act upon their environment through effectors" [115].

According to [115, 116] an intelligent agent must meet the following three requirements:

- Reactivity: Intelligent agents are able to perceive their environment, and respond in a timely fashion to changes that occur in it in order to satisfy their design objectives;
- Pro-activeness: Intelligent agents are able to exhibit goal-directed behaviour by taking the initiative in order to satisfy their design objectives;
- Social ability: intelligent agents are capable of interacting with other agents (and possibly humans) in order to satisfy their design objectives.

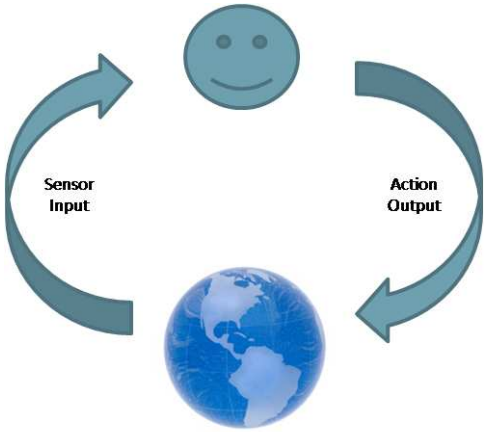

Figure 2.1: An agent in its environment. The agent takes input from the environment and generates actions that modify this environment. This is a cyclic behaviour

More recently the situatedness property of agents is stated [71]. ”Situatedness” refers that agents are situated in an environment. Agents tend to be used where the environment is challenging, more specifically, typical agent environments are dynamic, unpredictable and unreliable. These environments are dynamic in that they change rapidly; it means agents cannot assume that the environment will remain static while they are trying to achieve a goal. These environments are unpredictable in that it is not possible to predict the future states of the environment; often this is because it is not possible for an agent to have perfect and complete information about the environment, and because the environment is being modified in ways beyond the agent's knowledge and influence. Finally, these environments are unreliable in that the action that an agent can perform may fail for reasons that are beyond an agent's control. For instance, a robot attempting to rescue a victim in a disaster scenario may fail for a wide range of reasons including unexpected worsening of health condition of the victim or worsening of environment conditions. In other words, agents need to be reactive to face to these situations slightly trustworthy, without forgetting their design aims (pro-activeness), and the implication inside its team (social abilities).

Figure 2.1 gives a top level view of agent interaction in its environment. In this diagram, we can see the action output generated by the agent in order to affect its environment. In most domains of reasonable complexity, an agent will not have complete control over its environment. It will have at best partial control, in that it can influence it. From the point of view of the agent, this means that the same action performed twice in apparently identical circumstances might appear to have entirely different effects, and in particular it may fail to have the desired effect. Thus agents in all but the most trivial of environments must be prepared for the possibility of failure [116].

Intelligent agents are not independent of one another. In most cases they interact with other agents in order to reach their design objectives. This grouping of agents constitutes a multi-agent systems. Agents that interact in cooperative multi-agent system, need to coordinate tasks in order to fulfill its goals. Task allocation is an essential requirement for multi-agent

systems operating in cooperative environments. It allows agents to know their individual goals in order to improve the overall system performance.

2.3 AGENT ENVIRONMENTS

Environments in which agents operate can be defined in different ways. It is helpful to view the following definitions as referring to the way the environment appears from the point of view of the agent itself [83].

- Observable vs. partially observable

  In order for an agent to be considered an agent, some part of the environment - relevant to the action being considered - must be observable. In some cases (particularly in software) all of the environment will be observable by the agent. This, while useful to the agent, will generally only be true for relatively simple environments.

- Deterministic vs. stochastic

  An environment that is fully deterministic is one in which the subsequent state of the environment is wholly dependent on the preceding state and the actions of the agent. If an element of interference or uncertainty occurs then the environment is stochastic. Note that a deterministic yet partially observable environment will appear to be stochastic to the agent.

  An environment state wholly determined by the preceding state and the actions of multiple agents is called strategic.

- Episodic vs. sequential

  This refers to the task environment of the agent. A task environment is episodic if each task that the agent must perform does not rely upon past performance and will not affect future performance. Otherwise it is sequential.

- Static vs. dynamic

  A static environment, as the name suggests, is one that does not change from one state to the next while the agent is considering its course of action. In other words, the only changes to the environment are those caused by the agent itself. A dynamic environment can change, and if an agent does not respond in a timely manner, this counts as a choice to do nothing.

- Discrete vs. continuous

  This distinction refers to whether or not the environment is composed of a finite or infinite number of possible states. A discrete environment will have a finite number of possible states, however, if this number is extremely high, then it becomes virtually continuous from the agents perspective.

- Homogeneous vs. heterogeneous

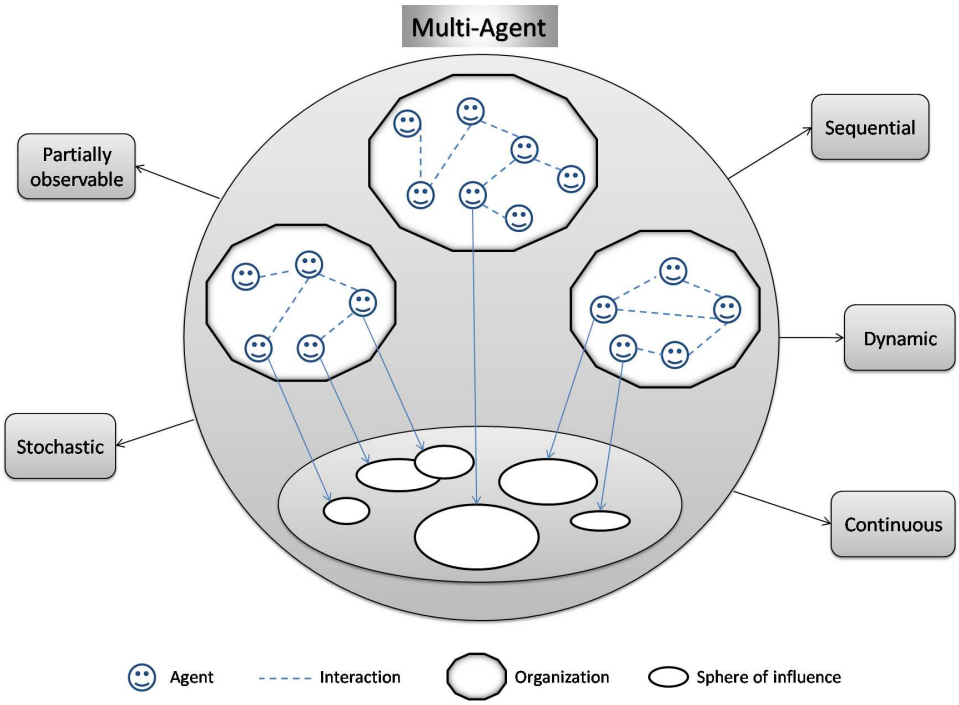

Figure 2.2: Typical structure of a multi-agent system

Homogeneous teams are composed of team members that have exactly the same hardware and control software, while in heterogeneous teams the robots differ either in the hardware devices or in the software control procedures. This distinction is used also for multi-agent systems, but in that case the differences are obviously only in the software implementation of the agents' behaviours [28].

- Single-agent vs. multiple agent

  An environment is only considered multiple agent if the agent under consideration must act cooperatively or competitively with another agent to realise some tasks or achieve goal. Otherwise, another agent is simply viewed as a stochastically behaving part of the environment.

This dissertation is addressed to agent environments which are partially observable, stochastic, sequential, dynamic and continuous. These kinds of features are related to real-world multi-agent environments.

Figure 2.2 illustrates the typical structure of a multi-agent system. The system contains a number of agents, which communicate with one another. The agents are able to act in an environment; different agents have different "spheres of influence", in the sense that they will have control over - or at least be able to influence - different parts of the environment. These spheres of influence may coincide in some cases. The fact that these spheres of influence may coincide may give rise to dependencies between the agents.

2.4 GOAL-ORIENTED AGENTS

Goal-orientation or goal-driven/directed is a property of systems that are able to reason or infer using symbols. Intelligent agents tend to achieve a goal and demonstrate it in subsequent actions. A great variety of goal-directed models of agency, focused at various level on representational, deliberative and action selection mechanisms, have been developed over the last two decades to design adaptive, autonomous and socially interactive agents. In this context, it is dealing with the goal-directed model of agency, where agents are intended as autonomous, resource bounded entities that attempt to arbitrate between several goals interacting in dynamic, partially observable environments [76].

Basically, a goal-oriented model involves temporal deadlines and temporal constraints. Goal-oriented research is addressed to design of systematic techniques to support the process of refining goals, identifying agents, and exploring alternative responsibility assignments. In this regard, the underlying principles are to refine goals until they are assignable to single agents and to assign a goal to an agent only if the agent can realize the goal [52].

2.5 MULTI-AGENT COOPERATION

Cooperative multi-agent systems in which agents must interact together to achieve their goals is a very active field of research. The term of ”cooperation” is frequently used in the concurrent systems literature to describe systems that must interact with one another in order to carry out their assigned tasks. Multi-agent cooperation refers to how agents can be designed so that they can work together effectively. In particular, cooperation involves the sharing both of tasks and of information, and the dynamic coordination of multi-agent activities [116].

2.5.1 TASK SHARING

How does a group of agent work together to solve problems?. Task sharing takes places when a problem is decomposed to smaller subproblems or subtasks and allocated to different agents. The key problem to be solved in a task-sharing system is that of how tasks are to be allocated to individual agents. In cases where the agents are really autonomous (and can hence decline to carry out tasks), then task allocation will involve agents reaching agreements with others, perhaps by using the auction or negotiation techniques.

2.5.2 INFORMATION SHARING

It involves agents sharing information relevant to their subproblems. This information may be shared proactively (one agent sends another agent some information because it believes that the other will be interested in it), or reactively (an agent sends another information in

response to a request that was previously sent).

2.5.3 HANDLING INCONSISTENCY

One of the major problems that arise in cooperative activity is that of inconsistencies between different agents in the system. Agents may have inconsistencies with respect to both their beliefs (the information they hold about the world) and their goals/intentions (the things that they want to achieve). Inconsistencies between the beliefs that agents have can arise from several sources. Firts, the viewpoint that agents have will tipically be limited -no agent will ever be able to obtain a complete picture of their environment. Also, the sensors that agents have may be faulty, or the information sources that the agent has access to may in turn be faulty [116].

2.6 MULTI-AGENT COORDINATION

From the organizational point of view, coordination means integrating or linking together different parts of an organization to accomplish a collective set of tasks [10]. In this sense, there are three fundamental processes to solve the coordination problem [61]: mutual adjustment, direct supervision and standardization. Mutual adjustment means that each agent is trying to adapt its behavior to improve the coordination. Direct supervision, means that there is one agent that can send orders to other agents. Finally, standardization means that there are some social laws enforcing the coordination among the agents.

Simply defined, agent coordination involves the selection, ordering, and communication of the results of agent activities so that an agent works effectively in a group setting [50].

From the multi-agent systems point of view, there exist four main approaches that have been developed for dynamically coordinating activities [116]:

1. **Coordinating through partial global planning:** The main principle of partial global planning is that cooperating agents exchange information in order to reach common conclusions about the problem-solving process. Planning is partial because the system does not (indeed cannot) generate a plan for the entire problem. It is global because agents form non-local plans by exchanging local plans and cooperating to achieve a non-local view of problem solving. Partial global planning involves three iterated stages.

    - Each agent decides what its own goals are and generates short-term plans in order to achieve them.
    - Agents exchange information to determine where plans and goals interact.
    - Agents alter local plans in order to better coordinate their own activities.

2. **Coordination through joint intentions:** This approach to coordination refers to the use of human teamwork models and it is focused on the concept of practical reasoning, and how central intentions are in this practical reasoning process. In this sense, intentions also play a critical role in coordination: They provide both the stability and the predictability that are necessary for social interaction, and the flexibility and reactivity that are necessary to cope with a changing environment. This approach distinguishes a coordination action that is not cooperative from a coordinated cooperative. In this sense, commitments and conventions are playing an important role in the agent interaction. In summary, this approach is related with the classical commitments, believes and intentions theory of rational agents [81].

3. **Coordination by mutual modelling:** This approach is closely related to the model of coordination through joint intentions discussed above. The idea of this coordination approach is that agents in a cooperative group put themselves in the place of the other to build a model of other agents- their beliefs, intentions, and the like- and to coordinate their activities around the predictions that this model makes.

4. **Coordination by norms and social laws:** This approach is based on the use of norms and social laws to coordinate agent's behaviours. In this sense, conventions play a key role in the social process. They provide agents with a template upon which to structure their action repertoire and also simplify an agent's decision-making process, by dictating courses of action to be followed in certain situations. There are two main approaches to decide the most effective method by which conventions and social laws can come to exist within an agent society: (a) Offline design in which social laws are designed offline, and hardwired into agents, and (b) Emergence from within the system which is dealing with a number of techniques by which a convention can ”emerge” from within a group of agents.

2.7 TASK ORIENTED DOMAIN

Task-Oriented Domain (TOD) is simply that agents who have tasks to carry out may be able to benefit by reorganizing the distribution of tasks among themselves; but this raises the issue of how to reach agreement on who will do which tasks [82]. For example, in a computational environment two agents might each be given separate sets of database queries to evaluate from a common database. Each operation on the database (such as a join, or a projection) is an indivisible task, with a well-defined cost. The agents would not have to worry about interference between their database accesses, and can possibly benefit (but not be harmed) by the other's jobs. Other example of a TOD is a multi-robot scenario where agents have a set of tasks to be performed; in this scenario robots need some protocol or tool to distribute tasks among themselves and by sharing tasks, robots may benefit by their team mate's jobs.

The idea of Task Oriented Domains (TODs) can serve as a useful approach to many kinds of activity. The TOD approach models agent activity in a straightforward way by restricting

in to a well-defined sequence of atomic actions.

Rosenschein and Slotking defined the notion of a task oriented domain (TOD). A task-oriented domain is a triple

$$\langle T, AT, c \rangle \tag{2.1}$$

Where:

- T is the set of all possible tasks;
- $AT = AT_1, AT_2, ..., AT_n$ is an ordered list of agents;
- c is a monotonic function $c : [2^T] \rightarrow R^+.[2^T]$ stands for all the finite subsets of T. For each finite set of tasks $X \subseteq T, c(X)$ is the cost of executing all tasks in X by any single agent. c is monotonic, i.e., for any two finite subsets: $X \subseteq Y \subseteq T, c(X) \leq c(Y)$.
- $c(\emptyset) = 0$.

A TOD specifies the set of all possible tasks that can be executed, a group of agents, and a cost function. Any agent is capable of carrying out any task or combination of tasks. Each finite set of tasks has a cost, which is independent or not of the agent or agents that carry it out (it depends of the particular domain or application). By adding more tasks, the cost will stay the same or increase (generally, will increase). The cost of empty set of tasks is defined to be 0.

2.8 Resource allocation

Resource allocation topic is related to the problem of how agents can allocate or distribute scarce resources among them [84]. We refer to the items that are being distributed as resources, while agents are the entities receiving them. We should stress that this terminology is not universally shared. In the context of applications of resource allocation in manufacturing or task-oriented domains, for instance, we usually speak of tasks that are being allocated to resources. That is, in this context, the term ”resource” (i.e. the resources available to the manufacturer for production) refers to what we would call an ”agent” here [18].

In a resource allocation problem, the resource in question is scarce, and is typically desired by more than one agent. In particular, auctions are effective at allocating resources efficiently; in the sense of allocating resources to those that value them the most [116]. In particular, combinatorial markets can be used to reach economically efficient allocations of items, services, tasks, resources, etc., in multi-agent systems [86].

2.9 AUCTIONS

The research community is now focusing on new marked-based paradigms such as auctions. An auction is a market institution with an explicit set of rules determining resource allocation and prices on the basis of bids from the market participants. Auctions provide principled ways to allocate them to agents. In particular, auctions are effective at allocating resources efficiently, in the sense of allocating resources to those that value them the most [116].

An auction consists of an auctioneer (seller) and potential bidders (buyers). Auctions are often used in situations where the auctioneer wants to sell an item and get the highest possible payment for it while the bidders want to acquire the item at the lowest possible price.

2.9.1 CLASSIFYING AUCTIONS

There are several factors that can affect both the auction protocol and the strategy that agents use [116]:

1. **Items for auction have a private, public/common or correlated value:** *common* value if agents value the item in the same way and a *private* value if agents value it differently. For instance, consider an auction for a one-dollar bill. It should be worth exactly $1; then the common value will be that ($1) for all the bidders in the auction. However, suppose you were a big fan of the Beatles, and the dollar bill happened to be the last dollar bill that John Lennon spent. Then it may be that, for sentimental reasons, this dollar bill was worth considerably more to you - you might be willing to pay $100 for it. The third type of valuation (*correlated* value) is when an agent's valuation of the item depends partly on private factors, and partly on other agent's valuations of it. An example might be where an agent was bidding for a painting that it liked, but wanted to keep open the option of later selling the painting. In this case, the amount you would be willing to pay would depend partly on how much you liked it, but also partly on how much you believed other agents might be willing to pay for it if you put it up for auction later.
2. **The winner determination:** Another dimension along which auction protocols may vary is the *winner determination*. It means, who gets the item that the bidders are bidding for, and what they pay. In this sense, there exist two protocols: the *first price* auctions and the *second-price* auctions. In the first-price the agent that bids the most is allocated the item, and pays the amount of the bid. In the second-price auction the item is allocated to the agent that bid the highest, but this agent pays only the amount of the second highest bid.
3. **The bids made by the agents are or not known to each other:** Auctions protocols may vary depending on whether or not the bids are *common knowledge*. Firstly, if

every agent can see that what every other agent is bidding, then the auction is said to be *open cry*. Second, if the agents are not able to determine the bids made by other agents, then the auction is said to be a *sealed-bid* auction.

4. **The mechanism by which bidding proceeds:** there exist three possibilities. Firstly, the *one shot* auction, which have a single round of bidding, after which the auctioneer allocates the good to the winner. Second, the *ascending* auction in which the price starts low and successive bids are for increasingly large amounts and third *descending* auction in which the auctioneer to start off with a high value, and to decrease the price in successive rounds.

Given the classification above, a wide range of different possible auction types can be identified. In this sense, mainly two types can be found: *Auction for Single Items* and *Combinatorial Auctions*.

2.9.2 AUCTION FOR SINGLE ITEMS

Auctions for Single Items are the simplest type of auction, which concerns the allocation of just a single item. Among these kinds of auctions we can find:

1. **English auctions**: These are the most commonly known type of auction. These auctions are *first-price, open cry, ascending auctions*. In an English auction the auctioneer starts off by suggesting a reservation price for the good, bids are then invited from agents, who must bid more than the current highest bid, when no agent is willing to raise the bid, then the good is allocated to the agent that has made the current highest bid, and the price they pay for the good is the amount of this bid.

2. **Dutch auctions:** These auctions are *open-cry, descending auctions*. In these auctions the auctioneer starts out offering the good at some artificially high value, the auctioneer then continually lowers the offer price of the good by some small value, until some agent makes a bid for the good which is equal to the current offer price. The good is then allocated to the agent that made the offer.

3. **First-price sealed-bid auctions:** These auctions are examples of *one-shot* auctions. In such an auction, there is a single round, in which bidders submit to the auctioneer a bid for the good; there are no subsequent rounds, and the good is awarded to the agent that made the highest bid. The winner pays the price of the highest bid. There are hence no opportunities for agents to offer larger amounts for the good.

4. **Vickrey auctions:** This auction is the most unusual of all the auction types. Vickrey auctions are *second-price, sealed-bid* auctions. In these auctions there is a single bidding round, during which each bidder submits a single bid; bidders do not get to see the bids made by other agents. The good is awarded to the agent that made the

highest bid; however, the price this agent pays is not the price of the highest bid, but the price of the second-highest bid.

Taking into account other aspects such as for instance: the number of units of items, the order in which the items are auctioned or the roles in buyers and sellers, the auctions may be classified such as follows:

1. **Number of units of items:** This issue is related to the number of items to be auctioned. There are two main kinds of auction: firstly, a *single unit auction* in which only one item is available for sale. And second, *multiunit auctions* [26] in which several (more than one) items are available for sale.

2. **Order in which the items are auctioned:** There are two main orders in which items can be auctioned. Firstly, *sequential auctions* in which the items are auctioned one at time; and second, *simultaneous or parallel auctions* [15] in where the items are open for auction simultaneously.

3. **Number of buyers:** regarding the number of buyers involved, it is possible to distinguish two main types of auctions; firstly, *Reverse auctions* [98] (n sellers - 1 buyer) that is a type of auction in which the roles of buyers and sellers are reversed. In an ordinary auction (also known as a forward auction), buyers compete to obtain a good or service, and the price typically increases over time. In a reverse auction, sellers compete to obtain business, and prices typically decrease over time. In business, the term most commonly refers for a specific type of auction process (also called procurement auction, e-auction, sourcing event or e-sourcing) used in industrial business-to-business procurement. Second, *Direct (forward) auctions* (1 seller - n buyers) which is the normal auction type that involves a single seller and many buyers.

4. **Roles of the participants:** regarding the roles of the participants, that is, becoming an auctioneer (seller) or a bidder (buyer) two classes of auctions are found. Firstly, *Double Auction* [54] in which, buyers and sellers are treated symmetrically with buyers submitting bids and sellers submitting asks. Both buyers and sellers submit bids. A single agent may even submit both, offering to buy or sell depending on the price. Second, *Asymmetric Auctions* in which the buyers and sellers cannot change their roles.

5. **Trading phases:** regarding the different phases involved in the auction process, two main auction design are defined: *Continuous double auction* (CDA) [113, 69] which run in continuous mode i.e., there are no ”trading phases” like other auctions may have. In a CDA, after a transaction occurs, the outstanding bid and ask are removed, and a new round of CDA starts, in which the auction process is repeated. *Discrete double auctions* in which there are ”trading phases”, i.e. they run in discrete mode.

2.9.3 Combinatorial auctions

In combinatorial auctions, bidders can bid on combinations of items. For example, if A, B, and C are three different items, a bidder can place separate bids on seven possible combinations, namely, {A}, {B}, {C}, {A,B}, {B,C}, {C,A}, and {A,B,C}. In the case of Combinatorial Auctions, the value of an item a bidder wins depends on other items that he wins. The notions of complementarity and substitutability are very important[67].

- Complementarity - Suppose an auctioneer is selling different goods. A bidder might be willing to pay more for the whole than the sum of what he is willing to pay for the parts. This property is called complementarity.
- Substitutability - A bidder may be willing to pay for the whole only less than the sum of what he is willing to pay for parts. This is called substitutability. This is the case if the bidder has a limited budget or the goods are similar or interchangeable.

Due to the expressiveness that combinatorial auctions offer to the bidders, such auctions tend to yield more economically efficient allocations of the items because bidders do not get stuck with partial bundles that are of low value to them [91]. The winner determination problem means choosing the subset of bids that maximizes the seller's revenue, subject to two constraints; that each item can be allocated only once, and that the seller wants to sell all the items.

In the next section the test bed of the RoboCup Rescue simulator is presented. All the experiments in this dissertation has been done using this test bed.

2.10 The test bed - RoboCup Rescue

This section presents the RoboCup Rescue simulation environment which has been used as a test-bed for all of the algorithms presented in this thesis. The most important characteristics of this environment are presented, followed by a discussion on some of its challenges that make it a really interesting and hard test-bed for multi-agent research.

The simulation project of the RoboCup Rescue is one of the activities of the RoboCup Federation [2], which is an international organization (originally called as Robot World Cup Initiative) is an international research and education initiative. It is an attempt to foster artificial intelligence and intelligent robotics research by providing a standard problem where a wide range of technologies can be integrated and examined, as well as being used for integrated project-oriented education.

RoboCup is registered in Switzerland, to organize international effort to promote science and technology using soccer games and rescue operations by robots and software agents. RoboCup chose to use soccer game as a central topic of research, aiming at innovations to be applied for socially significant problems and industries. The ultimate goal of the

RoboCup project is by 2050, develop a team of fully autonomous humanoid robots that can win against the human world champion team in soccer. The first international RoboCup Soccer competition took place in 1997 in Nagoya, Japan. Since then, the activities of the RoboCup Federation have been diversified. In 2001, the RoboCup Federation initiated the RoboCup Rescue project in order to specifically promote research in socially significant issues. Currently, the RoboCup Federation has 12 leagues regrouped in three major domains:

- RoboCup Soccer
  - Simulation League
  - Small Size Robot League
  - Middle Size Robot League
  - Four-Legged Robot League
  - Humanoid League
  - Standard Platform
- RoboCup Rescue
  - Rescue Simulation League
  - Rescue Robot League
- RoboCup Junior
  - Soccer Challenge
  - Dance Challenge
  - Rescue Challenge

As mentioned before, our work has been done in the RoboCup Rescue Simulation League. This simulation scenario consists of a simulation of an earthquake happening in a city [108]. The main goal of the agents (firefighters, policemen and ambulance teams) is to minimize the damages caused by an earthquake, such as civilians buried and damaged, buildings on fire and road blocked. As in real situations, agent teams have a limited scope. For instance, agents can see visual information within a radius of 10 meters. In addition, also, there are important constraints on the communications, for instance, one agent is capable to say or listen a fixed maximum of messages each simulation cycle.

RoboCup Rescue is chosen because it is a well-known domain used for benchmarking multi-agent coordination algorithms. It is a complex problem in which teams of agents have to allocate and to perform tasks using incomplete information in a stochastic multi-agent environment, in real time.

RoboCup Rescue scenario is based on real-world scenarios, with detailed simulators modeling different parts of the system and it is particularly pertinent to exploring coordination at

different levels of granularity, and coordination processes which interact with each other [4]. In addition, a great spectrum of multi-agent issues and techniques applied for coordination, task allocation, planning, replanning can be tested in this real-world scenario which implements a lot of complex inter-agent and agent-environment interactions. Thus, it represents a good platform for evaluating the efficacy of our approach.

In the next sections, we first describe the RoboCup Rescue simulator, then we present the different agents that have to be implemented and finally we explain why this environment is an interesting test-bed environment for our multi-agent algorithms.

2.10.1 RoboCup Rescue simulator

In this section, the RoboCup Rescue simulator is presented. First, we present all the modules constituting the simulator. Then we present how the time is managed and how the different modules interact between each other during the simulation.

Simulator modules

The RoboCup Rescue simulation is composed of a number of modules that communicate between each other using the TCP protocol. These modules consist of: the kernel, the rescue agents (FireBrigade, PoliceForce, etc.), the civilian agents, the simulators (FireSimulator, TrafficSimulator, etc.), the GIS (Geographical Information System) and the viewers. All these modules can be distributed on different computers. Figure 2.3 presents the relations between these modules. More precisely, these modules can be described as follows:

- **Kernel:** The kernel is at the heart of the RoboCup Rescue simulator. It controls the simulation process and it manages the communications between the modules. For example, all messages between the agents have to go trough the kernel. There are no direct communications between the agents. When the kernel receives the messages, it verifies them to make sure that they respect some predefined rules. Then it sends the valid messages to their intended recipients. The kernel is also responsible for the integration of all the simulation results returned by the simulators. Moreover, the kernel controls the time steps of the simulation and it manages the synchronization between the modules.
- **GIS:** The GIS module provides the initial configuration of the world at the beginning of the simulation, i.e. locations of roads, buildings and agents. During the simulation, this module is responsible for providing all the geographical information to the simulators and the viewers. It also records the simulation logs, so that the simulation can be analyzed afterwards.
- **Simulators:** The simulator modules are responsible for the dynamic aspects of the

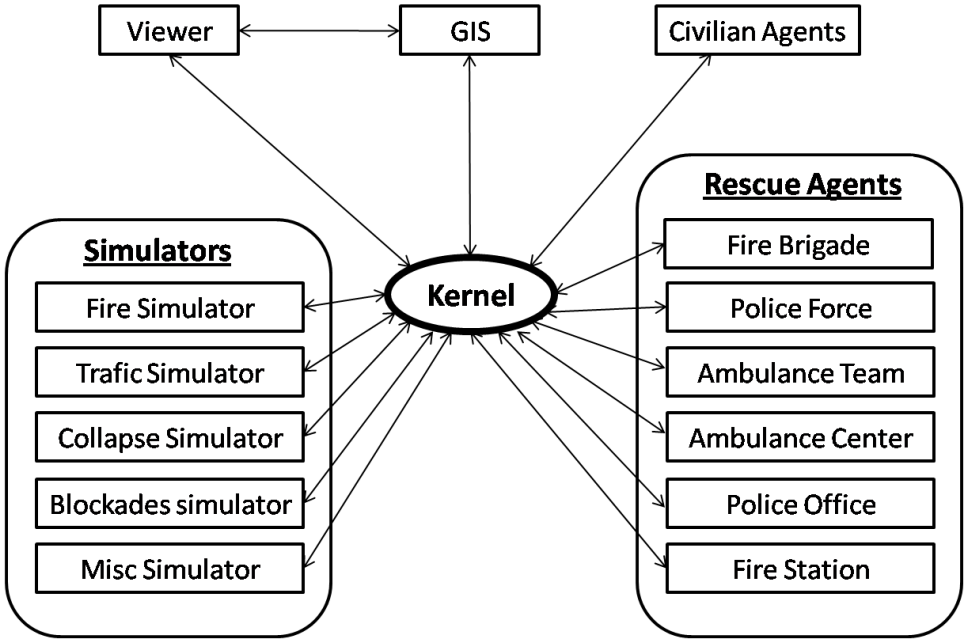

Figure 2.3: RoboCup Rescue simulator architecture.

world. They manage all dynamic parts of the world, including the effects of the agents actions. To achieve that, they manipulate the environment information provided by the GIS module. There are five simulators:

- Fire simulator: The fire simulator controls the fire propagation, which depends on the buildings compositions, the wind, the amount of water thrown by the FireBrigade agents and the distance between the buildings.
- Traffic simulator: The traffic simulator is in charge of simulating the agents movements in the city. It has to deal with blockades and traffic jam.
- Collapse simulator: The collapse simulator simulates the impact of the earthquake on the buildings. It controls how badly a building is damaged by the earthquake and how deeply the agents are trapped in the buildings.
- Blockades simulator: The blockade simulator simulates the impact of the earthquake on the roads. It generates all the blockades on the roads.
- Misc simulator: The misc simulator simulates the agents injuries and the agents actions: load, unload, rescue and clear.

- **Civilian agents:** This module controls the civilian agents, which have to be rescued by the rescue agents. These agents have really simple behaviors. They scream for help if they are trapped in buildings or they simply move around in the city, trying to reach a refuge.

- **Viewer:** The viewer module graphically presents the information about the world provided by the GIS module. It is possible to have more than one viewer connected to the kernel at the same time. There are 2D and 3D viewers available.

- **Rescue agents:** These modules control the rescue agents. For the RoboCup Rescue competition, participants have to develop these modules. There are six different modules or type of agents that have to be developed:
  - FireBrigade: There are between 0 to 15 agents of this type that have to extinguish fires.
  - PoliceForce: There are between 0 to 15 agents of this type that have to clear the roads.
  - AmbulanceTeam: There are between 0 to 10 agents of this type that have to rescue agents or civilians that are trapped in collapse buildings.
  - FireStation: This is the FireBrigades control station which is responsible for the communications between the FireBrigade agents and the other type of agents.
  - PoliceOffice: This is the PoliceForces control station which is responsible for the communications between the PoliceForce agents and the other type of agents.
  - AmbulanceCenter: This is the AmbulanceTeams control station which is responsible for the communications between the AmbulanceTeam agents and the other type of agents.

Time management

The RoboCup Rescue system simulates five hours after the earthquake has happened. It is a discrete simulation in which each time step corresponds to one minute. Therefore, the simulation is executed in 300 time steps. It is the kernel that is responsible for managing the time of the simulation. To achieve that, it should impose a real-time constraint for all the modules; the kernel does not wait for the modules responses. If the kernel receives an information from a module too late, this information is discarded. This is an important constraint to keep in mind, because it limits the time allowed for an agent to reason about its next action. Therefore, all the algorithms developed for the agents have to be fast and efficient. In the current settings of the environment, the complete loop of the kernel takes two seconds, which leaves approximately one second for all the modules to compute their actions.

Simulations progress

The first step of a simulation consists for all the modules to connect to the kernel. Figure 2.4 illustrates the initialization process. At the beginning of the connection process, the GIS module sends the initial configuration of the world to the kernel. The kernel then forwards this information to the simulator modules and it sends to the rescue agents only their perceptual information. At the same time, the viewer asks the GIS for the graphical information about the world. Afterwards, the simulation starts. As mentioned before, the

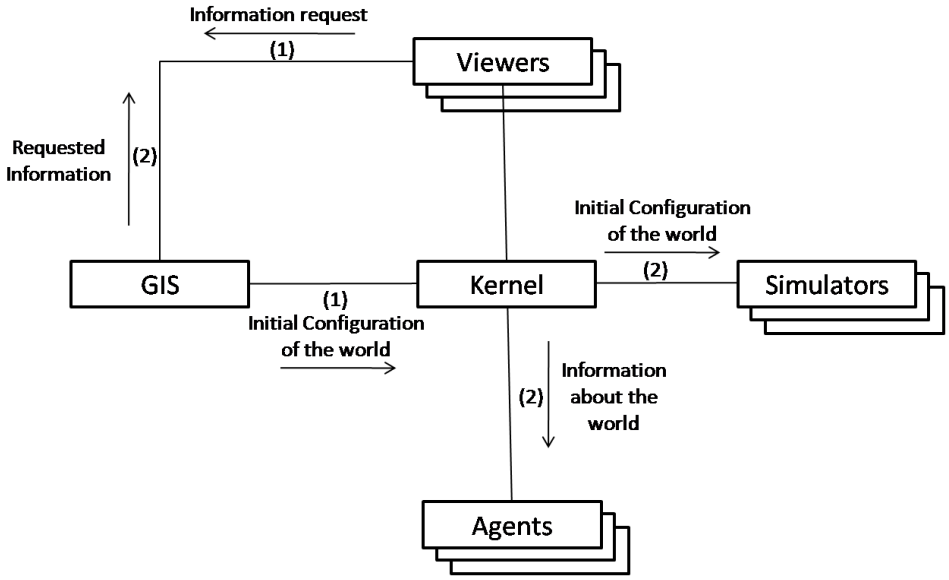

Figure 2.4: Communication between the simulator modules during the initialization phase.

simulation is composed of 300 cycles and each one of them is composed of the following steps (see Figure 2.5):

1. At the beginning of every cycle, the kernel sends to the rescue agents all their sensors information (visual and auditive). The visual information of an agent contains all the objects that are in a 10 meters radius around the agent. The auditive information contains all voice messages and radio messages. The communication between the agents is explained later.

2. Each agent module then uses the sensors information received to decide which actions it should do. Notice that the actions include the physical actions and the communication actions.

3. The kernel gathers all the actions received from the agents and it sends them to the simulator modules. The actions received are sometimes filtered by the kernel. For example, if the kernel receives an action from a dead agent, the action would not be considered. Also, since the simulation proceeds in real-time, the kernel ignores all the actions that do not arrive in time. Only accepted actions are sent to the simulator modules

4. The simulator modules individually compute how the world will change based upon its internal state and the actions received from the kernel. Then, each simulator module sends its simulation results to the kernel.

5. The kernel integrates the results received from the simulator modules and it sends them to the GIS module and to the simulator modules. The kernel only integrates the results that are received on time.

6. The kernel increases the simulation clock and it notifies the viewers about the update.

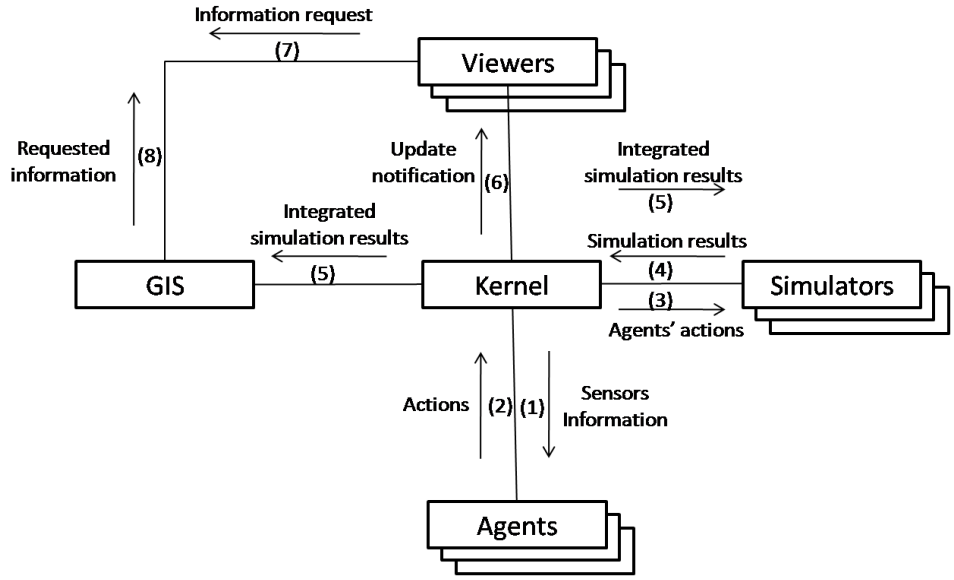

Figure 2.5: Communication between the simulator modules during one cycle of the simulation.

7. The viewers request the GIS to send the updated information of the world.

8. The GIS keeps track of the simulation results and it sends to the viewers the information they requested. Finally, the viewers visually display the information received from the GIS.

Evaluation function

In the RoboCup Rescue simulation, the performance of the rescue agents is evaluated by considering the number of agents that are still alive, the healthiness of the survivors and the unburned area. As we can see in the following equation, the most important aspect is the number of survivors. Therefore, agents should try to prioritize the task of rescuing civilians.

$$Score = \left(nA + \frac{H}{H_{ini}}\right)\sqrt{\frac{B}{B_{ini}}} \tag{2.2}$$

where nA is the number of living agents, H is the remaining number of health points *(hp)* of all agents, H_{ini} is the total number of *hp* of all agents at the beginning, B_{ini} is total buildings area at the beginning and B is the undestroyed area which is calculated using the fieriness value of all buildings. The fieriness attribute indicates the intensity of the fire and how badly the building has been damaged by the fire. This attribute can take values from 0 to 8, as presented in Table 2.1 Using these fieriness values, Table 2.2 presents the rules used to evaluate the unburned area of each building.

Table 2.1: Meaning of the buildings fieriness attribute values

fieriness	Meaning
0	Intact building.
1	Small fire.
2	Medium fire.
3	Huge fire.
4	Not on fire, but damage by the water.
5	Extinguished, but slightly damage.
6	Extinguished, but moderately damage.
7	Extinguished, but severely damage.
8	Completely burned down.

Table 2.2: Score rules used to evaluate the area burned based on the buildings fieriness attribute.

fieriness	Score rules
0	No penalty.
1 or 5	$\frac{1}{3}$ of the buildings area is considered destroyed.
4	Water damage, also $\frac{1}{3}$ of the buildings area is considered destroyed.
2 or 6	$\frac{2}{3}$ of the buildings area is considered destroyed.
3, 7 or 8	The whole building is considered destroyed.

GRAPHICAL REPRESENTATION OF THE SIMULATION

In order to see the evolving simulation, the RoboCup Rescue simulator has a viewer module responsible for the graphical representation of the simulation. Figure 2.6 presents an example of a RoboCup Rescue situation. Buildings are represented as polygons. Gray polygons means that the buildings are not on fire. If the building is on fire, then it is yellow, or orange. If the building was on fire and then extinguished, it is blue. Green buildings represent refuges where the injured agents have to be sent. White buildings represent the three center agents (FireStation, PoliceOffice and AmbulanceCenter). The darker a building is, the more damage it is.

Agents are represented as circles: FireBrigade (red), PoliceForce (yellow), AmbulanceTeam (white) and Civilians (green). Again, the darker an agent is, the more injured it is. The blue lines represent the water thrown by the FireBrigade agents on the fires. A little x on a road means that this road is blocked. When a PoliceForce agent clears a road, the x disappears.

2.10.2 RESCUE AGENTS

The objective of the RoboCup Rescue simulation project is to study rescue strategies, and also collaboration and coordination strategies between rescue teams [109]. Participants in

Figure 2.6: Viewer of the Kobe Map from the RoboCup Rescue scenario.

the RoboCup Rescue championship have to develop software agents representing teams of firefighters, polices and paramedics, in order to manage the disaster the best way they can. These agents have to:

- determine where are the emergencies with the highest priorities,
- choose which roads to clear so that strategic places can be reached,
- choose where to dig in order to rescue the most civilians,
- carry injured civilians to the refuges,
- choose which fires to extinguish in priority,
- etc.

For the current test-bed, there are approximately 100 agents representing groups of people (civilian families, firefighter teams, police forces, ambulance teams). This grouping has been done to simplify the simulation. However, the objective of the RoboCup Rescue committee is to have more than 10 000 agents in the simulation to make it more realistic [108]. The number of agents will be increased when the computer hardware will support that many deliberative agents in one simulation. In the simulation, agents can accomplish different actions that can be divided in two classes: actions shared by all agents and actions specialized and available to only some types of agents.

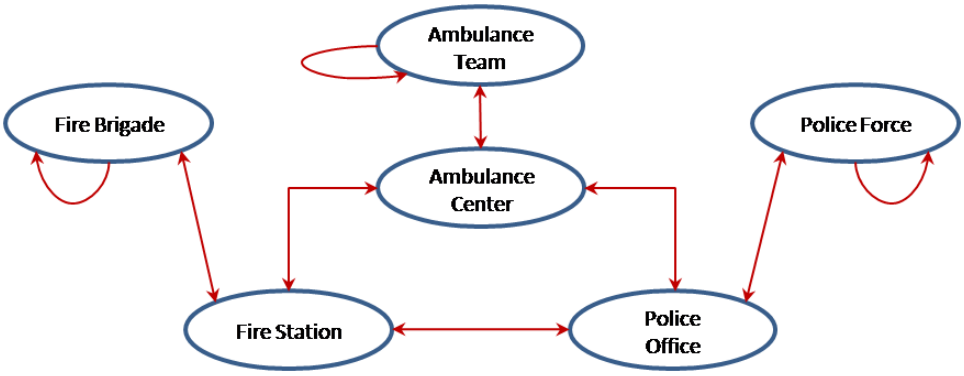

Figure 2.7: Communication organization. Links between different types of agents indicate that a message can be sent by radio between these two types of agents.

- *Shared actions:* These modules control the rescue agents. For the RoboCup Rescue competition, participants have to develop these modules. There are six different modules or type of agents that have to be developed:
  - Move (except for building agents);
  - Speak to near agents;
  - Communicate by radio with all the agents of the same type and their center agent;
  - Do nothing.
- *Specialized actions:* These modules control the rescue agents. For the RoboCup Rescue competition, participants have to develop these modules. There are six different modules or type of agents that have to be developed:
  - FireBrigade agents can extinguish fires;
  - PoliceForce agents can clear roads;
  - AmbulanceTeam agents can dig to rescue civilians and they can transport other agents (civilians or rescue agents);
  - Center agents (FireStation, PoliceOffice and AmbulanceCenter) can communicate with the other center agents.

The coordination and the collaboration between the agents are really important, because the agents efficiency can be improved if the agents collaborate with each other. The firefighter agents, the police agents and the paramedic agents work faster if they work in teams. For example, if there are many FireBrigade agents that cooperate to extinguish the same fire, then the fire will be extinguished much faster than if only one agent tries to extinguish it.

There are two different communication actions: Say and Tell. With the Say action, an agent can speak to all agents in a 30 meters radius around it. With the Tell action, agents communicate by radio. All radio messages are broadcasted to the other agents following the communication organization presented on Figure 2.7. For example, if a FireBrigade agent

sends a message by radio, it will be received at the next time step by all the other FireBrigade agents and by the FireStation agent. One should note that this communication organization limits the ability to communication between heterogeneous agents. For example, a FireBrigade agent cannot directly send a message to a PoliceForce agent. The message has to go from the FireBrigade agent to the FireStation agent, to the PoliceOffice agent and finally to the PoliceForce agent. As we can see, it needs at least three time steps for a message to go from a FireBrigade agent to a PoliceForce agent. This communication constraint is only one of the many constraints imposed by the RoboCup Rescue simulation environment. In the next section, we present why it is such a complex problem.

2.10.3 Environment complexity

The RoboCup Rescue simulation is a complex environment that imposes many constraints like:

- A real-time constraint on the agents response time. All agents have to return their action in less than a second after they received their perceptions.
- The agents perceptions are limited to a 10 meters radius.
- The length and the number of messages that an agent can send or received are limited.
- The FireBrigade agents have a limited amount of water available.
- The civilians die if they are not saved on time.
- The time left before a civilian dies is unknown.
- The fires are spreading fast if they are not controlled rapidly.
- Rescue agents can easily create traffic jam.

One of the most important problems in the RoboCup Rescue simulation is the partial observability of the environment. In the simulation, agents have only a local perception of their surroundings. Agents only perceive the objects that are in a 10 meters radius around them. Consequently, there is no agent that has a complete view of the environment state. Even more than that, the RoboCup Rescue simulation is a collectively partially observable environment [66]. This means that even if all the agents perceptions are regrouped, these agents would not have a perfect vision of the situation.

This uncertainty complicates the problem greatly. Agents have to explore the environment, they cannot just work on the visible problems. Therefore, one major problem for the agents is to acquire useful information in a reasonable time [44]. Agents also have to communicate with each other to improve their local perceptions, even though they will never have a

Table 2.3: Maximal number of messages per time step that an agent can send or receive. n is the number of mobile agents of the same type as the center agent.

Agents type	Receive	Send
Mobile agents	4	4
Center agents	2n	2n

perfect knowledge about the environment. Communications are quite restricted, but they are still really important, because the coordination between the agents depends a lot on the efficiency of the communications between them.

As mentioned before, agents have to communicate to compensate for their restrictive local perceptions. However, agents have to be really careful about the messages they send, because it is really easy to loose a message due to the limitations on the number of messages that can be sent or received and because of the communication organization presented in Figure 2.7. The maximum number of messages that can be sent or received during one time step of the simulation are presented in Table 2.3. As we can see, center agents have better communication capabilities because they can receive and send more messages than the mobile agents. Each center agent can receive and send $2n$ messages per time step where n is the number of mobile agents of the same type as the center agent. For example, if there are 10 FireBrigade agents, then the FireStation agent can send and receive 20 messages per time step. Since center agents can receive more messages, they normally have a better knowledge of the global situation. Therefore, center agents are the best agents to serve as the center of coordination for the mobile agents of the same type. With such communication constraints, there is a good chance that a message gets lost and that it does not reach its intended recipient. For example, consider the case where 10 FireBrigade agents each sends one message during one time step. This is really under the limitation of the agents, because they could each send 4 messages in one time step. However, even with only one message sent per agent, each agent will receive 9 messages, which is more than twice the number of messages an agent can receive in one time step. Consequently, each agent will lost 5 messages. The situation can be worst if the agents have more than one message to send or if there are messages coming from other types of agents trough the center agent. It then becomes really important for the agents to have a good strategy to choose which messages they should send or listen to.

Moreover, the communications in the RoboCup Rescue simulation are situated communications [70], which means that the information contained in a message depends a lot on the position of this information on the map. For example, an information about a fire is useless if the agent does not transmit the position of the fire. For the communication between the agents, the complexity happens when agents have to choose which are the most important messages to listen to. For example, a message coming from a near agent has more chance to be useful than a message coming from a far agent, because normally we need more coordination messages for agents working on the same problem. Consequently, to efficiently choose which messages to listen to, each agent has to estimate the position of the other

agents in the city. This could be quite hard since agents are always moving. Another difficulty of the RoboCup Rescue environment is that agents are heterogeneous. They have different capabilities and there is no agent that can do everything by itself. Consequently, agents have to collaborate with each other if they want to accomplish their tasks efficiently [73]. Agents have to coordinate their actions in order to profit from each others capabilities.

In the simulation, it is also really important to efficiently manage the resources, because there is a lot of work to do with few resources. Logistic and more particularly distributed logistic become then a complex problem. There are a lot of problematic situations in the simulated city and the agents have to be assigned to the problems that will maximize their actions results.

2.10.4 MULTI-AGENT TESTBED

The RoboCup Rescue simulation environment is a good testbed for multi-agent algorithms, because it has some really interesting characteristics for research in this domain. Here are some of its advantages as a testbed environment:

- The environment is complex enough to be realistic.
- The testbed is easily accessible.
- The testbed covers a lot of different multi-agent problems.
- The testbed enables to compare the approaches developed with the other participants at the competition.

The RoboCup Rescue simulation environment offers a complex testbed allowing many multi-agent research opportunities or more generally many artificial intelligence research opportunities [43]. These opportunities are present in domains like: *Multi-agent planning.* There are many heterogeneous agents that have to plan and act in an hostile and dynamic environment. Anytime and real-time planning. Agents have to plan while following some real-time constraints.

Robust planning. Planning has to be done with incomplete information. The planning system has to be able to efficiently replan if some information changes.

Resources management. Resources are really limited in the simulation, thus it becomes important to manage them efficiently. Learning. Tasks are also quite complicated, thus agents have to learn how to assign the resources to the different tasks. They also have to learn some dynamic aspects of the environment in order to estimate its evolution.

Information gathering. Agents have to explicitly plan for information gathering actions in order to improve the agents global vision of the environment.

Coordination. Agents have to coordinate their actions because more than one agent is usually needed to accomplish the tasks.

Decision-making in large scale systems. Agents have to analyze many possibilities and choose an action to accomplish in a really huge partially observable state space.

Scheduling. There are many civilians that have to be rescued and each of them has a different estimated death time and a different rescue time. These dynamic tasks have to be schedule in order to maximize the number of civilians rescued.

In the next chapter, we review selected publications related to the topics covered in this thesis. The word related to the RoboCup Rescue environment, crisis management and the task allocation research. Particularly, we will focus on market based techniques such as auctions and combinatorial auctions.

CHAPTER 3
Related Work

This chapter presents an overview of the main works focused on the topics addressed in this dissertation.

3.1 COORDINATION IN CRISIS MANAGEMENT DOMAIN

Agent technology can be used to support many processes throughout the phases of the disaster management cycle, agent-based simulation systems are generally used to model human and systems behaviour during or after disaster events. By including intelligent agents it is possible to analyze complex disaster scenarios [30].

Disaster management has become an important issue in the last few years due to the large number of disasters occurring such as the Haiti Earthquake in 2010 and other recent catastrophes. Disaster management involves coordinating a large number of emergency responders to rescue either people or infrastructure in possibly hazardous environments where uncertainty about events is predominant [80].

The research in crisis management begins about the 80s. Quarantelli [77] stands that there are three problematical areas in disaster management field: (1) The communication process, (2) the exercise of authority, and (3) the development of coordination. He emphasizes that the problem related to the communication process generally involves what is communicated rather than how communication occurs. It means that communication problems do not necessarily arise from equipment scarcity or damage facilities. In the majority of cases, problems arise from the exchanged communication flow due to the number of staff using the communication system which increases greatly. This issue results in ”overload,” the net result of which causes either system failure or results in the loss or delay of information to, from, and among staff member.

The other problems described by Quarantelli refers to coordination. He stands that organizations experience a large number of coordination problems during a community disaster. Basically, the problem focuses on the absence of an explicit understanding of what coordination means in operational terms. This lack of agreement will increase the organizational coordination problems. Quarantelli stands also that coordination (i.e., mutually agreed linking of activities of two or more groups) between organizations working on common but new tasks is also difficult. Even local agencies that are accustomed to working together, such as

police and fire departments, may encounter difficulties when they suddenly try to integrate their activities to accomplish a novel disaster task, such as the handling of mass casualties. In addition, Quarantelli stands out for the magnitude and increased frequency of new tasks to be performed coupled with the need to integrate too many established, emergent groups and organizations minimize the effectiveness of organizational coordination during disaster situations.

In recent years, interest in disaster management has ballooned. Several approaches to coordination in this socially significant domain have been carried out. Addressed to an explicit understanding of what coordination means in management crisis, we have focused on the fundamental processes to solve the coordination problem [61] previously described in the background chapter in section 2.5: direct supervision and mutual adjustment. Direct supervision, means that there is one agent that can send orders to other agents and mutual adjustment means that each agent is trying to adapt its behavior to improve the coordination. In this sense the coordination approaches to crisis management may be divided as: centralized approaches (direct supervision) and decentralized approaches (mutual adjustment).

In the centralized process, practically all decisions are made by a supervisor agent. Therefore, the decision process is centralized in one agent. [73] presents a centralized approach applied to the RoboCup Rescue scenario [1], in this approach the central agent (fire station) is informed of the situation, i.e. where the fires are, by messages sent by all agents in the simulation. At each turn, the central agent uses a function, to sort all fires according to their importance. Afterwards, the central agent sends a message to each agent (FireBrigade) containing the two best task (fire) it has identified. Those two fires are seen as orders by the FireBrigades, so they obey and try to extinguish the first fire on the list. When the first one is extinguished, they try to extinguish the second one. Other approach using a centralized approach is presented on [103]. This paper describes a RoboCup Rescue approach which uses multi-criteria decision making technique to distribute the victims to be rescued among the agents. In [36] the centralized agents send all their sensed information to their centers and wait for the center to tell them what to do in the next cycle. So all the actual thinking is done in centers. Moreover, several teams participating in the RoboCup Rescue competition uses centralized approaches. For instance, Caspian team [95] uses a centralized coordination approach and regards the partitioning strategy as a social law, applying it to all the rescue agents. MRL team [97] describes a centralized algorithm in which the central agent sorts victims based on their time to death, then checks the ability to rescue the victim before death and allots agents to the victim. SBCe_saviour [68] uses centralized coordination strategy and sets up a virtual center that coordinates all the decisions made by central agents. Black Sheep [99] assigns a central point to each police force agent; each police force agent is only responsible for the district surrounding its central point.

In the decentralized process, each agent chooses the task it wants to carry out by itself, so all decisions concerning to rescue operation is taken in a distributed manner, locally by each agent. For instance, SOS [58] describes a decentralized mechanism in which

the basic strategy is to queue those civilians who may die during simulation and sorting them by the number of ambulances that each civilian needs to stay alive. In Poseidon [62] approach, firstly each agent knows its position and situation correctly, then communication is used for agent coordination; each agent acts separately and informs the others from its situation and decision.

Despite these approaches are described separately, in crisis management and RoboCup Rescue research the centralized and decentralized approaches are combined to obtain all the benefits from the advantages of both of them.

Among many coordination paradigms proposed, market-based coordination is a promising technique which is well suited to the requirements of the multi-agent systems [24].

Among the free market-based mechanisms, auctions have been widely exploited for task coordination in multi-agent systems. Approaches have been used with centralized allocation, both for combinatorial auctions [20, 87, 117, 12] and single task auctions [107, 59, 48].

3.2 Auctions applied to RoboCup Rescue

In this section we review the research done so far using Auctions techniques within the Robot Rescue domain. For each author we briefly detail the purpose of the approach, if any, the league in which it is applied to and a general description of the approach.

3.2.1 Ahmed et al.

In this work [5], they proposed the application of a dynamic auctioning scheme in the context of a UAV (Unmanned Aerial Vehicle) search and rescue mission and presented early experimentations using physical agents. Ahmed et al. addressed the combinatorial problem by extending a forward/reverse auction algorithm [13]. This algorithm alternates between rounds of forward and reverse auctions. In the forward stage, agents bid for tasks, while in the reverse stage tasks (conceptually) bid for agents by reducing their prices. The auction starts once a new target is detected inside the mission area. The nearest UAV is considered as its auctioneer agent and it announces a new auction. After the selection, all UAV agents start competing for the new target and the auction runs in rounds, each of a predetermined time. At the end of each round, the auctioneer agent evaluates its current state: if it collects the required number of agents for the target, then it announces the result to winner agents which form a winner group to service the target, and this information becomes common knowledge in the team. Otherwise, the auctioneer agent reduces the price of the current target (reverse step) and propagates the new price to the agent team members. The above procedure is repeated until the target is assigned.

3.2.2 Akin and Ozkucur.

Reference [7] presented an auction algorithm to coordinate tasks of the rescue team agents in the RoboCup Rescue simulation League. In this approach, the auctions are used where the tasks are bid according to a plan. The agent can bid if it is not assigned to any task of that kind. At each time step, each agent sends its status to notify its center. The center collects these messages and issues a task list message. Upon receiving the task list, the agent creates a plan by internally simulating the auction phase. The agent selects the best target for it and auctions the task among its teammates and develops a plan for specified size. If the plan size is small all the agents will participate in the task, on the other hand if the plan size is large most of the task will be assigned to agent with larger plans and some of the agents cannot get a job. Therefore, the size of the task plans controls the number of agents participating in the task and the total completion time. Here, Platoon agents and center agents both exchange the same auctioneer role to control the task assignment. However, they state that his preference is using the center as an auctioneer.

3.2.3 Sedaghat et al.

Sedaghat et al. [94] presented a study to allocate blocked roads to police forces in the RoboCup Rescue simulation League. In this approach, the police office takes on the role of an auctioneer, and the police forces take on the roles of the bidders. The items that are bid for are the tasks or, in this environment, the blocked roads. In the task allocation process, firstly, fire brigades and ambulances send a request for clearing the obstacles in a blocked road to the fire station or the ambulance center respectively. Then, fire station and ambulance center collect the received requests and send a request to the police office. Next, the police office notifies police forces about the received requests. The real auction starts here. Police forces receive the requests and make bids on them, and send their bids to the police office. The police office collects all the bids and determines the winner. Once the winner determination is solved, all the police forces will be notified about the winner by the police office. After accomplishing the task, the police force who won the auction will notify the center that he is free, so that he can participate in coming auctions. In addition, another partitioning-based approach is designed to allocate tasks between the police force agents. Sedaghat et al. states that both methods are combined to obtain the common benefits. The auction mechanism is used to urgent request and the partitioning is used when police agents do not receive urgent request. However, the remaining features about urgency and task priority were not taken into account yet and are described to be a future work.

3.2.4 Adams et al.

The work presented by [3] describes an auction approach carried out into the context of the ALADDIN Project. ALADDIN is a multi-disciplinary project that is developing techniques, architectures, and mechanisms for multi-agent systems in uncertain and dynamic environments. In the disaster management domain the different stakeholders correspond to the police, fire and ambulance services, rescuers, and civilians. The resources include the service vehicles and equipment, routes, sensors, bandwidth, etc. Research in ALADDIN is focusing on the design of auction strategies (i.e., bidding and payment strategies) and mechanisms to achieve fair, rational, and efficient system goals and coordination. It has examined dynamic auctions which are modeled as multiple auctions closing simultaneously. The extended RoboCup Rescue system is being used to test the research since it is a dynamic environment with multiple stakeholders.

3.3 Combinatorial auctions applied to RoboCup Rescue

In this section we review the research done so far using Combinatorial Auctions techniques within the Robot Rescue domain. For each author we briefly detail the purpose of the approach, if any, the league in which it is applied to, and a general description of the approach.

3.3.1 Nair et al.

Maybe, the closest to our study, Nair et al. [65] presented an initial approach for including a Combinatorial Auction technique in the task allocation of the team of fire brigade agents in the RoboCup Rescue simulation League. In this auction mechanism, the centres take on the role of auctioneers, and the ambulances, fire brigades and police forces take on the roles of bidders. The items being bid for are the tasks. At the beginning of each cycle, each free agent makes several bids - each bid consists of a different combination of tasks and an estimate of the cost of performing sequentially the tasks. The auctioneer receives all the bids and determines the winning bids. The algorithm used for winner determination is based on [89]. It is a depth-first branch-and-bound tree search that branches on bidders. The branching factor of the search tree is O(b), where b is the maximum number of bids that a single bidder makes. Nair et al states some shortcomings of using combinatorial auctions for task allocation: (1) Exponential number of possible bids. (2) Difficult to make cost estimate. (3) Domain-imposed communication constraints. Since the work was only a first introduction of the model, no experiments were reported.

3.3.2 Habibi et al.

In the work presented by Habibi et al. [35] agents are allowed to bid directly for bundles of resources. Suggesting a set of resources, the agents propose a value for them and use combinatorial auction techniques to solve the coordination problem in the RoboCup Rescue simulation League. Here, the auctioneer is assumed to allocate resources to bidders so that the overall benefit of the system is maximized. In this system, the buildings are assumed to play the role of bidders and agents are regarded as the extinguishing resources. So it can be assumed that buildings suggest values for a bundle of fire brigades. To evaluate each bid the followings are important: (1) The importance of the building which is an indicator of its strategic and intrinsic importance. (2) The amount of time it takes the proposed coalition to put out the building. This measure can be calculated by means of statistical and computational analysis. Regarding winner determination the approximate algorithms developed by E. Zurel have been exploited. This algorithm uses some heuristic functions to prune the search space very fast. Then it uses some greedy methods and by means of hill climbing it finds an approximate of the optimal solution [125].

3.4 Other approaches for task coordination applied to RoboCup Rescue

Besides market-based techniques, other approaches have been studied to solve the task coordination problem. In this section, we review those fields where most researchers have focused their efforts on, such as: Division of Labor in swarms, Distributed Constraint Optimization, Partially observable Markov processes, Evolutionary algorithms, Markov games formulation, Reinforcement learning, Fuzzy theory and Neural networks used within the RoboCup Rescue domain.

3.4.1 Division of labor in swarms (Ferreira et al.)

Division of labor is fundamental to the organization of insect societies and is thought to be one of the principal factors in their ecological success. Division of labor in insect colonies is characterized by two features: (a) different activities are performed simultaneously by (b) groups of specialized individuals, which is assumed to be more efficient than if tasks are performed sequentially by unspecialized individuals. The study in Ferreira et al. [29] proposes an algorithm (Swarm-GAP) for task allocation using division of labor theory in a disaster scenario. Swarm-GAP is designed with the aim to allow agent to decide individually which task to execute, assuming that the communication does not fail. This model describes task distribution among individuals using the stimulus produced by tasks that need to be performed and an individual response threshold related to each task (this threshold is a measure related to the capability of the agent to carry out each task). The intensity of this stimulus can be associated with a pheromone concentration, a number of encounters

among individuals performing the task, or any other quantitative cue sensed by individuals. An individual that perceives a task stimulus higher than its associated threshold has a higher probability to perform this task. This approach presents some results and evaluations using different stimulus values and number of agents. The Swarm-GAP method has been evaluated in an abstract scenario and in the scenario of the RoboCup Rescue. In addition, Swarm-GAP is compared with other approximated algorithm for Distributed Constraint Optimization Problem (DCOP) and a centralized greedy algorithm.

3.4.2 Distributed constraint optimization (Scerri et al.)

Of particular interest is the Distributed Constraint Optimization Problem (DCOP) model, in which a group of agents must choose values in a distributed fashion for a set of variables such that the cost of a set of constraints over the variables is either minimized or maximized [120]. The work of Scerri et al. [92] proposes an algorithm for task allocation (LA-DCOP) for the DCOP framework, in where each agent is provided with a variable to which it must assign values which correspond to tasks the agent will perform. In this approach, a token is created for each value. The holder of a token has the exclusive right to assign the corresponding value to its variable, and must either do so or pass the token to a teammate. In this way, the authors state that conflicts cannot occur and communication is reduced. Given the token-based access to values, the decision for the agent becomes whether to assign to its variable values represented by tokens it currently has or to pass the tokens on. LA-DCOP uses a threshold on the minimum capability an agent must have in order to assign the value. This threshold is attached to the token. If the agent computes that its own capability is less than the minimum threshold, it passes it randomly to a teammate. (To avoid agents passing tokens back and forth, each token maintains the list of agents it has visited; if all agents have been visited, the token can revisit agents, but only after a small delay). This threshold would guide the tokens towards agents with higher capabilities to perform them. To do that, the authors present a model which allows calculation of the threshold. This type of calculation can be done by a team member to determine the best threshold for a newly-created token. This calculation is based on the expected utility to the team of using that threshold. Moreover, this approach presents a mechanism to manage interdependencies between tasks. Basically, the idea consists on allowing values to be represented by potential tokens. By accepting a potential token, an agent confirms that it will perform the task once the interdependencies have been worked out. In the meantime, the agent can perform other tasks. For example, in an AND constraint, the team only receives reward for each task if all the constrained tasks are simultaneously executed. An AND constrained set of tasks can be used to represent a task that requires multiple agents to successfully perform (such as extinguishing a large fire). The RoboCup Rescue simulator and an abstract simulator have been used for experimentation.

3.4.3 Partially observable markov decision processes (Paquet et al.)

When faced with partially observable environments, a general model for sequential decision problems is to use the (POMDPs). Several problems can be modelled with POMDPs, but very few can be solved because of their computational complexity (POMDPs are PSPACE-complete [72]). The main problem with POMDPs is that their complexity makes them applicable only on small environments. Several approximation algorithms have been developed recently to solve the problem offline and online. The work of Paquet et al. [75] presented an online algorithm called Real-Time Belief Space Search (RTBSS) based on a look-ahead search in the belief state space to find the best action to execute at each cycle in a dynamic environment. RTBSS uses a factored POMDP representation and a branch and bound strategy. Basically, a factored POMDP means to reduce the states space by consider only possible states that are reachable by the agent. The authors state that it allows representing the states much more compactly and respecting the real-time constraint. To speed up the search, the algorithm uses a ”Branch and Bound” strategy to cut some sub-trees. This approach has been applied to the policeman agents in the RoboCup Rescue scenario. Their task is to clear the most important roads as fast as possible. In the RoboCup Rescue POMDP representation, the online search in the belief state space represents a search in the possible paths that an agent can take. In the tree, the probability to go from one belief state to another depends on the probability that the road used is blocked. One specificity of this problem is that a path has to be returned to the simulator, thus the RTBSS algorithm is used to return the best branch of the tree instead of only the first action.

3.4.4 Evolutionary algorithms

Evolutionary computation is based on the mechanics of natural selection and the process of evolution [38]. Chromosomes encode the potential solutions of the problem to solve. During the search, chromosomes are combined and mutated in order to find the best solution (although it is not guaranteed to find the optimal one). Kleiner et al. [45] proposed an evolutionary method of Genetic algorithms (GA) for the optimization of victims' sequences in the RoboCup Rescue Simulation League.

Studies of GA are implemented in a computer simulation in which a population of chromosomes of individuals to an optimization problem evolves toward better solutions. The evolution usually starts from a population of randomly generated individuals and happens in generations. In each generation, the fitness of every individual in the population is evaluated, multiple individuals are stochastically selected from the current population (based on their fitness), and modified (recombined and possibly randomly mutated) to form a new population. The study of Kleiner et al. takes into account the time needed for rescuing civilians and the life time of civilians to estimate the overall number of survivors after executing a rescue sequence by a simulation. The time for rescuing civilians is approximated

by a linear regression based on the buriedness of a civilian and the number of ambulance teams that are dispatched to the rescue. Travel costs between two targets are estimated by averaging over costs sampled during previous simulation runs. For each rescue sequence $S = \{t_1, t_2, ..., t_n\}$ of n rescue targets, an utility U(S) is calculated that is equal to the number of civilians that are expected to survive. The GA is initialized with heuristic solutions, for example, solutions that greedily prefer targets that can be rescued within a short time or urgent targets that have a short lifetime. The fitness function of solutions is set equal to the previously described utility U(S). In order to guarantee that solutions in the genetic pool are at least as good as the heuristic solutions, an elitism mechanism, which forces the permanent existence of the best found solution in the pool, has been used. Furthermore, a simple one-point-crossover strategy, a uniform mutation probability of $p \approx 1/n$, and a population size of 10 have been utilized. Within each cycle, 500 populations of solutions are calculated by the ambulance station from which the best sequence is broadcasted to the ambulance teams that synchronously start to rescue the first civilian in the sequence. Moreover, the work in [78] proposed a GA approach to determine a priority for each road to be cleared in the RoboCup Rescue simulator scenario.

3.4.5 Markov game formulation (Chapman et al.)

Chapman et al. [17] reports a decentralized game-theoretic approach to allocate victims among ambulance teams in the RoboCup Rescue simulation environment. In this work, each agent has to perform a sequence of tasks over time and often tasks may require more than one agent for their successful completion. Markov games are an extension of standard non cooperative games, for repeated interactions in which the game played by the agents at each time-step, t, varies probabilistically as a function of the state and the choice of strategies in the previous round, or simply as some environmental process evolves. In the Chapman et al. Markov game formulation, agents compute a global utility u into a particular time window $t + w$. This global utility maximizes the local utility of each agent in the environment. This global utility u for instance, increases with the number of civilians rescued, and for each civilian, increases with lower completion times. Then, agents select these victims which maximizes its utility u for a particular $t + w$ time. In the cooperative allocation algorithm, an agent has some probability p of activation, known as the degree of parallel executions. Given an opportunity to update its action, an agent selects the action with the greatest increase in payoff - its best response - or if no change improves the payoff, the agent does not change its strategy (for instance, the agent follows with its own initial plan).

3.4.6 Reinforcement learning, fuzzy and neural networks

The work of Aghazadeh et al. [4] presented a reinforcement learning method for police force (PF) agent's decision making in RoboCup Rescue Simulation. Reinforcement learning

algorithms attempt to find a policy that maps states of the world to the actions the agent ought to take in those states. The main drawback of this technique is the large state space that most problems present. As a consequence, a large amount of learning steps are required to find the policy that matches states and actions. Nevertheless, researchers have tried to overcome these drawbacks as for instance, Aghazadeh et al. who focused their work on learning to select areas (paths) to be cleared in the virtual map for the PFs in the RoboCup Rescue simulation. It proposed an Action Evaluation Function (AEF) which is a linear combination of some parameters which affect the importance of an action. In fact, the aim of the learning algorithm is discovering the importance of each parameter comparing with other ones. These parameters to select paths are for instance: distances of the path to the police force, the number of buried humans including civilians and agent teams buried in the buildings connected to the path and the reported blockades of the path. Reported blockade parameter of a path increases when an agent requests PF agents to open it. Rewards are only positive and are computed according to the number of agents encountered the target, the number of agents that have requested opening of the target and some other factors. In this way, the PF agent can estimate the expected return of state-action pairs that it has never experienced before (for instance to select the higher priority roads in a particular ”state” or map area).

The work of [32] proposed a fuzzy - neural networks method in which both of these techniques are used for predicting the future of fire points and selecting fire points for the fire brigade teams in the RoboCup Rescue environment. In this approach, the properties of fiery buildings and their neighbors are used as inputs to fuzzy inference parts. The fuzzy inference rules are of the type: if area of burning buildings is low and distance from fire point to nearest refuges is high then output is low (it means that the fire point has low importance by the agent). Afterwards, agents will select the building with maximum criteria as the next fire point to be extinguished.

3.5 Challenges in market-based multi-agent/robot coordination

Auctions and combinatorial auction techniques tackled in this dissertation are the most important mechanisms among the market-based mechanisms. For this reason, we presents in this section the challenges found in this field for multi-agent coordination.

In the last years, a growing popularity of maket-based multi-agent/robot coordination approaches has been extended within the AI research community. Market-based approaches effectively meet the practical demands of agent teams while producing efficient solutions by capturing the respective strengths of both distributed and centralized approaches. First, they can distribute much of the planning and execution over the team and thereby retain the benefits of distributed approaches, including robustness, flexibility, and speed. They also have elements of centralized systems to produce better solutions. Auctions concisely gather information about the team and distribute resources in a team-aware context. In this

sense, among the challenges currently facing the field are [25]:

3.5.1 Challenges in replanning

Replanning is a major remaining challenge and relates closely to the continuing challenges for teams operating in dynamic environments. Existing market mechanisms are not yet fully capable of replanning task distributions, redecomposing tasks, rescheduling commitments, and replanning coordination during execution.

3.5.2 Challenges in dinamicity

The principle challenges relevant to coordinating a team are ensuring graceful degradation of solution quality with failures, enabling team functionality despite imperfect and uncertain information, maintaining high response speed to dynamic events, and effectively accommodating evolving task and team composition.

3.5.3 Challenges in task allocation and sequencing

One of the greatest strengths of a market is its ability to utilize the local information and preferences of its participants to arrive at an efficient solution given limited resources. The fundamental optimization problem encountered in market-based multi-agent/robot systems is the task allocation problem. While some theoretical guarantees for simple cases of auction algorithms are now known, implemented systems are generally more complex and can include online, multi-task, peer-to-peer, simultaneous, and overlapping auctions as well as task and scheduling constraints. Additionally, solution quality depends on cost and utility estimates, which are sometimes difficult to accurately specify.

3.5.4 Challenges in tight coordination

Significant challenges remain in using market-based techniques for tight coordination. Among other requirements the field needs some formalization of solution quality, market mechanisms that can handle persistent coordination, and bidding techniques that concisely encapsulate complex constraints. Additionally, even in the general multi-agent/robot literature, tight coordination in highly dynamic environments a largely remains unsolved problem.

The study of auctions and market-based mechanisms in dynamic environments and with uncertainty is one of the challenges presented by Dias et al. In this sense, well-designed auctions mechanisms must be addressed to face this dynamicity and to implement more

robust solutions. In the same way, such environments need the design of replanning techniques or methods to allow agents to recover or adjust its plans when the changes in the environment so require it. This issue is addressed to the challenge tackled by Dias et al. it says ”Replanning is a major remaining challenge and relates closely to the continuing challenges for teams operating in dynamic environments”. On the other hand, task allocation is the key issue of this dissertation, and it is very important to design and to implement task allocation mechanisms to allow agents know which tasks are the most important and which report the most benefits to them. Otherwise, agents would not know where to start and their work would be unsatisfactory.

3.6 FINAL REMARKS

Given a team of agents, an amount of resources, and team tasks, researchers must develop a method of distributing the resources among the team so the task is accomplished well, even as teammates' interactions, the environment, and the mission change. RoboCup Rescue domain is a complex environment with all these features of dynamicity and uncertainty. It is a strongly challenging domain because of the complexity involved in each inter-agent and agent-environment interaction.

Such as it has been presented in this related work, a vast majority of coordination approaches have been carried out that use different techniques and some of them have elements that are centralized and distributed and thus reside in the middle of the spectrum. Market-based and auction approaches fall into this hybrid category, and regards coordination, they present some advantages as for instance, Market-based concerns the correct use of the communication because of the simplicity of the protocols required to implement such mechanisms. Another reason to utilize market-based techniques is the recent push in the design of powerful algorithms for combinatorial auctions that perform reasonably well in practice [18].

In summary, we think that market based techniques such as auction and combinatorial auctions are a suitable solution to implement effective coordination strategies and mechanisms in the domain of our interest. In addition, such as it was remarked at the beginning of this chapter, RoboCup Rescue approaches require of hybrid studies in order to utilize the advantages of its centralized architecture and at the same time to provide to agents with alternatives when the central agent fails or it is not available. In this sense, Market-based and auction approaches fall into the hybrid category of coordination approaches, and, if designed well, they can opportunistically adapt to dynamic conditions to produce more centralized or more distributed solutions [25].

Regards combinatorial auctions for task allocation in the rescue scenario, some issues remain unresolved:

The existing approaches are preliminary works and hence, more exhaustive studies and experimentation about the efficacy of combinatorial auctions for task allocation should be

carried out. Reference [65] proposed the distance criteria to create the bids sent by the agents in the rescue scenario. In this regard, other approaches taking into account the tasks and agents' preferences and capabilities are needed. To do that, agents may include into their bids issues such important as the urgency and priority of the tasks (for instance, the death-time of the victim and the urgency of the fires).

In addition, it is clear that approaches related to the number of bids sent by the agents are necessary to deal with the problem of an exponential number of possible bids received by the auctioneer. In this sense, it is necessary the creation of a new method to decide on what tasks a bidder should bid on.

Moreover, the current works are not dealing with replanning of tasks which is very relevant in this kind of dynamic scenarios.

CHAPTER 4
Task Allocation and Coordination Approach

This chapter presents the approach for crisis management proposed in this dissertation. This approach is aimed at improving the operation and tasks execution of agents interacting in a dynamic multi-agent environment. The problem description, formalization aspects and algorithms for multi-agent coordination and replanning are introduced in this chapter.

4.1 TASK COORDINATION PROBLEM

The main problem addressed in this dissertation is concerned to the lack of appropriate coordination and task allocation mechanisms and algorithms addressed to highly dynamical and uncertainty scenarios such as a rescue environment. Explicitly, these environments having bad coordination among agents result in a lower performance. Agents without coordination get lost in their context and they are not able to determine their goals. In addition, the lack of appropriate coordination increase the entropy in the agent organization, getting many lost messages and a lot of communication problems among agents. On the other hand, cooperation -not merely coordination- may improve the performance of the individual agents or the overall behaviour of the system they form. It is due research in multi-agent systems addresses the problem of designing agents which interact effectively.

The cause of the mentioned lack of coordination among agents in multi-agent systems is an inefficient allocation of tasks among agents. Task allocation guarantees agents an efficient determination of goals and a successful execution of tasks which permits agents to reach their aims in a cooperative way. In this sense, it is necessary the creation and design of new task allocation and coordination mechanisms to provide agents with an efficient tool to make decisions in these complex scenarios.

In the next section the particular requirements found in the RoboCup Rescue scenario are presented.

4.2 System requirements

Taking into account the analysis presented in the related work and the challenges exposed in the RoboCup Rescue scenario, we have detected some important requirements for a good performance of our agents. The requirements will be explained taking into account different aspects such as: The kind of rescue agent, the task allocation algorithms, the robustness of the system, the cooperation between heterogeneous agents and the replanning of tasks.

The RoboCup Rescue environment is conformed by *heterogeneous agents* with different capabilities and capacities. For this reason, we think different task allocation and coordination algorithms and mechanisms must be created according to the features of design of each one of them. In this sense, it is necessary the implementation of ad-hoc algorithms taking into account these features.

Regards fire brigade agents, for instance, as previously stated in Chapter 2, the fire station agent has a better global view of the situation. Therefore, it can suggest good fire areas to fire brigade agents. On the other hand, the fire brigade agents have a more accurate local view, consequently, they can choose more efficiently which building on fire to extinguish. In this sense, the central information about fires within the environment is crucial for the fire brigade operation and at the same way, to take advantage of the local vision of the fire brigades should be a remarkable issue.

Regards ambulance team agents operation, for instance, they have to rescue the victims, but the number of victims that can be rescued depends a lot on the order in which they are rescued. In the RoboCup Rescue scenario, the victims are not known at the beginning of the simulation. For this reason, agents have to explore the environment to find the victims and then incorporate them in their schedule.

In this sense, the new designed algorithms should take into account these intrinsics features of the heterogeneous rescue agents. Moreover, it should be also taken into account that agents have to be able to adapt the plans frequently to take the dynamic changes of the environment into consideration. In addition, other remarkable issue is that RoboCup Rescue is an uncertain environment, it means, the tasks' parameters could change between two time steps. At this light, new real-time algorithms should be implemented.

In the RoboCup Rescue scenario, another important feature is the *robustness* of the algorithms and mechanisms designed. With robustness, we mean, the capacity of the system to maintain good performance when faced with hard communication constraints. In this regard, the new algorithms should be robust.

During the RoboCup Rescue simulation, agents can face some hard environmental problems. For instance, agents can stay on blocked roads. In this light, the new algorithms should take into account the *replanning* of tasks from the initial scheduling.

In the RoboCup Rescue domain, the *cooperation* between heterogeneous rescue agents is very important. For instance, ambulance teams need information from the fire brigades and police forces to find the victims; at the same way, the ambulance teams and fire brigades need cleared roads for the police forces. For this reason, a strategy of sharing information and cooperation among heterogeneous agents is necessary; this strategy should take into account the communication constraints to avoid overload of the system.

The rescue agents need *task allocation algorithms* that take into account their *preferences*, for instance, the priority of the victims with regard the ambulances teams operation, and the priority or urgency of fires in the fire brigade operation. In this sense, the new algorithms should include this information into the tasks assignation problem.

This thesis tackles the tasks allocation and coordination problem and it is looking for efficient mechanisms and algorithms to solve the previous mentioned problems. These algorithms are presented in the next sections of this chapter.

4.3 Multi-agent task coordination

Multi-agent task coordination approaches have received significant attention and gained considerable popularity within the agent research community in the last years. These have been successfully implemented in a variety of domains ranging from resource and task allocation to robot rescue. Agent simulation permits to test mechanisms and techniques into simulated environments where agents have to interact in order to reach their goals. That way, these new techniques can be then implemented on different robotic domains.

In order to reach their goals, agents need to coordinate their tasks. Many of these tasks can be better achieved by a team of agents than by a single agent. By working together, agents can complete tasks faster, increase systems robustness, improve solution quality, and achieve impossible tasks for a single agent. Nevertheless, coordinating such a team requires overcoming many formidable research challenges .

Given a team of agents, a limited amount of resources, and a set of tasks, Multi-agent task coordination field focuses on developing a method of distributing the resources among the team so the tasks are accomplished well, even as teammates' interactions, the environment, and the mission change. These changing features give to the research a dynamical perspective which increases the complexity.

Some open questions remain in the areas of multi-agent coordination and task assignment. How should a group or agents distribute task among its members? And How should agents cooperate in order to accomplish a specific task? In this sense, both issues deal with the efficiency in the tasks execution into the multi-agent environment.

Problems of coordination are not unique to software agents, but exist at multiple levels of activity in a wide range of real-life applications. For instance, people pursue their own goals through communication and cooperation with other people or machines and each living

organism (animals or particles) interacts with other, and form communities. Research in multi-agent coordination is being widely extended to different real-life applications; however it still has a long way to go and there are many research challenges.

4.4 TASK ALLOCATION

Task allocation refers to the way that tasks are chosen, assigned, and coordinated. Let's suppose a sports team goes out to play a game, before of the beginning, a large amount of time is spent deciding which player will play a particular position. The position that a player is placed in will often be dependent on their level of skill. In the same way than sports teams, in multi-agent teams tasks are assigned to agents according to their skills and capabilities. We consider the multi-agent coordination problem to be a task allocation problem where the tasks arrive dynamically and can change in intensity. In our approach, task allocation refers to task planning for a group of agent teams. These plans must be executed by them in dynamic scenarios and in this sense, planning of tasks allows agent teams coordinate among themselves in order to reach their goals.

Agents in a multi-agent system, even when identical in their design and implementation, may be heterogeneous in the sense that they each have different sets of capabilities. They may further differ over the tasks they need to perform. Such heterogeneous agents may need to perform tasks for which they do not have the required capabilities [11]. In this sense, one agent with some capabilities may be unable to perform a task due to the intrinsic constraints of the task (for instance, its due-date). Cooperation with other agents, possibly in the form of task sharing to the latter, can resolve this problem.

Taking into account the context previously mentioned, for each agent, the reasoning tasks which concern coordination and planning may be organized as follows:

1. Goal Determination: Generate the goal to achieve. An agent can determine a goal in three manners: Firstly, autonomously to detect goals with respect to the current situation or location of the agent in the environment. It is mainly reached by exploration. Secondly, to be asked to achieve a goal of other agents. And thirdly, to get an assigned goal by a supervisor or central agent.
2. Execution: determine which action or actions to begin for the fulfillment of the current goal. Normally, these actions are events or commands which are triggered by an agent within the multi-agent environment (For instance, move until position 23, load victim in position 23, unload in position 23, clear obstructed path id 12).

4.4.1 EXAMPLE OF REASONING TASKS WITHIN A RESCUE ENVIRONMENT

Rescue scenario: in a rescue scenario, agent teams have to explore the environment to look for victims, once an agent team has found some victim, it has to reach it, try to rescue

it, load it on the ambulance car and finally, take it until closest refuge. Let us suppose a more particular example, the rescue agent team *A* finds a victim *V* in position 23. The reasoning activities of *A* are as follows:

Reasoning task 1. Reach the victim.

1. Goal determination: According the rescue example, the goal would be: reach *V* on position 23. This goal refers that one agent team who locates a buried or injured victim has to go until the victim's position.
2. Execution: Trigger the events. At this time, the agent team has to generate the proper commands in order to perform the current goal: for instance, *A* would trigger the next two commands: Search path (until position 23), then Move (through the returned path).

Reasoning task 2. Rescue the victim.

1. Goal determination: Following the example, this goal refers to: rescue *V* on position 23. For instance, in the RoboCup Rescue context, one victim has to be rescued if the victim cannot move because it is buried under a pile of objects. In this sense, rescue a victim is the process where an agent team must dig up and extract the victim of the rubbles. This process permits the victim to be mobilized until a safer place (which is the final goal in the reasoning process of the agent).
2. Execution: Here, the agent team should trigger the command: "Rescue *V*" (on position 23). This command is triggered every time step until the victim is completely removed from the rubbles.

Reasoning task 3. Load the victim.

1. Goal determination: Following the example, this goal consists on: "load *V* on position 23" and it means that an agent team which has removed a victim from the rubbles has to load it on the ambulance in order to be transported until the hospital or shelter.
2. Execution: The command triggered in this goal would be: Load *V* (on position 23).

Reasoning task 4. Reach the closest refuge.

1. Goal determination: In this part of the agent team's reasoning activities; the agent team has *V* under its control. Then, the next goal consists on: "reach the closest refuge". The agent has to look for the closest shelter into its world map and then reach it.

2. Execution: Trigger the events: Search path (until refuge's position), then Move (to refuge's position).

Reasoning task 5. Unload *V* into the refuge.

1. Goal determination: In the final activity of the reasoning process, the agent team which has reached the refuge and it is transporting a victim *V* has to unload it into the refuge to be treated.
2. Execution: Trigger the events: Unload *V* into refuge.

The previous example shows the reasoning activities that a rescue agent has to perform in order to accomplish rescue tasks in a general disaster scenario. This agent team' reasoning implies an individual agent. In a multi-agent system, there are other phases which may precede this agent's reasoning. For instance, how a group of agents must reason in order to accomplish their tasks in the environment, taking into account that these are not alone and these are part of a group. In this sense, the interaction in a group of the agents implies coordination. For instance, Figure 4.1 shows the agent teams' reasoning who are interacting within the environment.

In this sense, in this dissertation the coordination approach is presented since two issues: firstly, task allocation is studied as the way that agent teams can generate goals efficiently into a multi-agent environment. In this light, we treat each goal as a particular task to perform in any moment during the inter-agent interaction. Secondly, we are dealing with task allocation in a multi-agent scenario in which a supervisor agent may exist and also the case when the supervisor agent is not available. These two approaches permit agent teams adapt their behaviours according to the system characteristics at anytime. In addition, this thesis also tackles replanning of tasks. Reallocation or replanning is a part of any task allocation process [25]. Replanning permits agent teams restore from fails occurred during the tasks performing or from allocation problems, such as allocation of unreachable tasks which avoids the timely task execution of an agent team. In the next section, we present the formalization aspects of our multi-agent approach and the implemented algorithms.

4.5 Formalization aspects

Let us suppose that an agent team AT is belonging to a cooperative group G into a scenario S. A cooperative group must generally involve more than agent team for the execution of a task (see Figure 4.2). That is:

$$\forall AT_i, AT_j \in G_k | i \neq j \quad and \quad G_k \subseteq GA \quad \text{where} \quad GA = \{AT_1, AT_2, ..., AT_n\} \tag{4.1}$$

Where GA is the total group of n agent teams into the scenario. In addition, one AT cannot belong to more than one group G in a specific time step t.

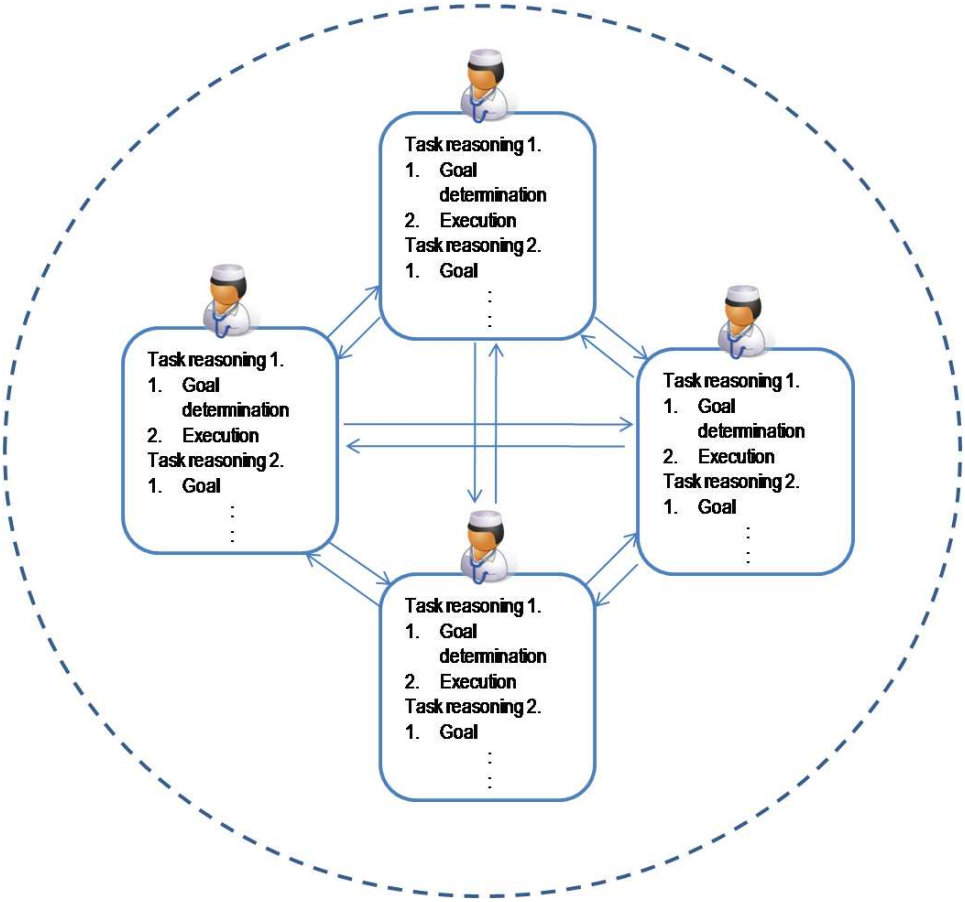

Figure 4.1: Agent teams' reasoning who are interacting within their environment.

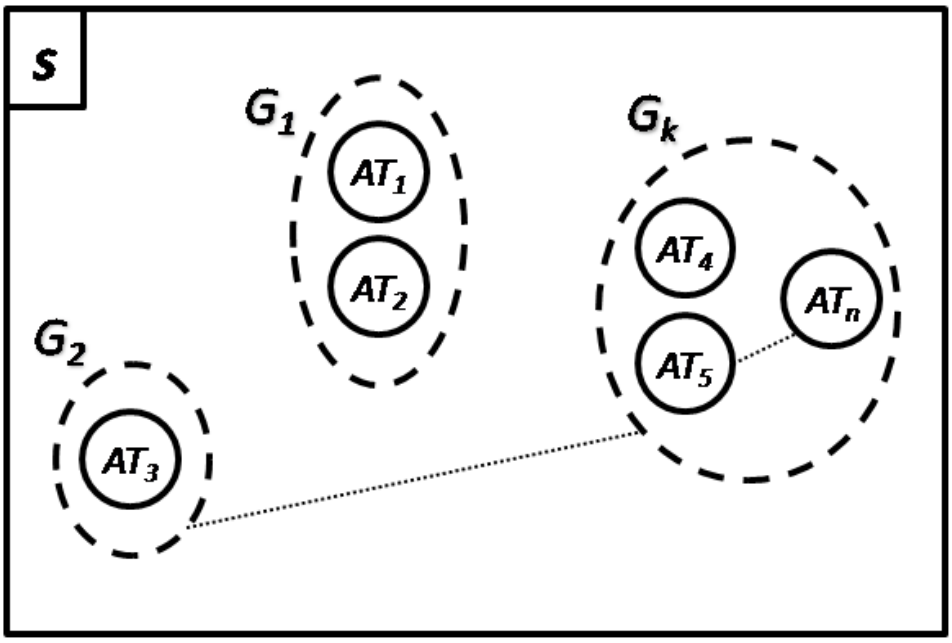

Figure 4.2: One scheme of agent teams within the scenario S.

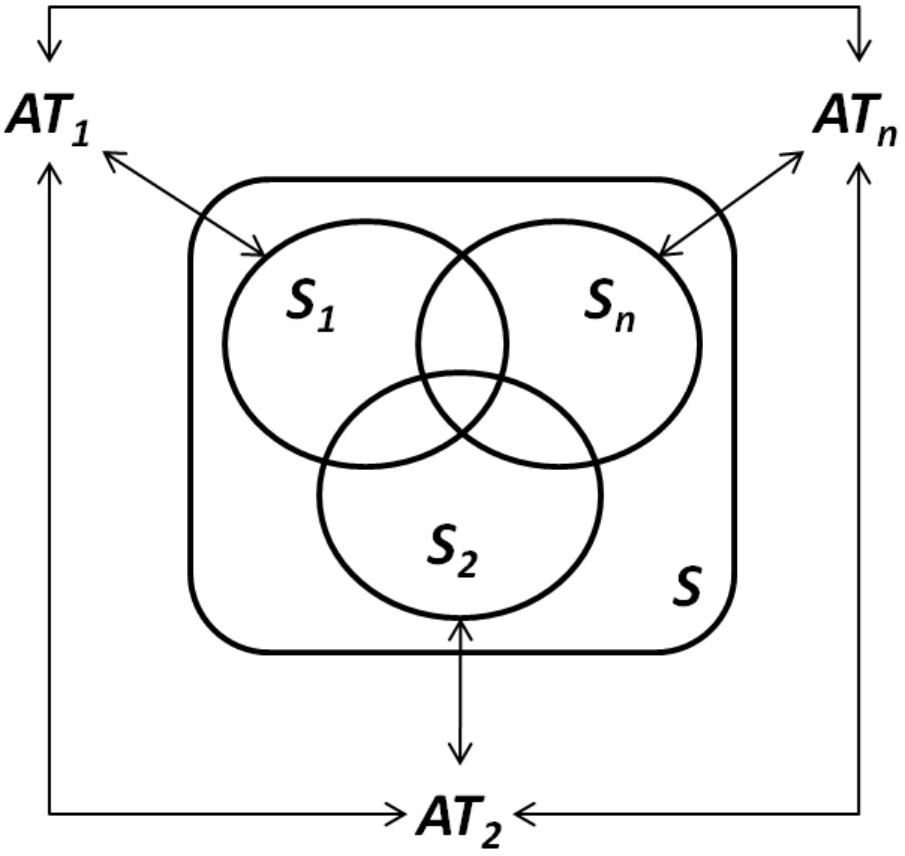

Figure 4.3: Environment interaction. Each agent AT_i only knows a fraction S_i of the actual environment or scenario S.

In a typical multi-agent environment, generally, some local information constraints are presented. For instance, the fact that an agent typically has only local information about the environmental state, and this information may differ from the one another agent has [115]; this situation is illustrated by figure 4.3. Each agent team AT only knows a fraction S_i of the actual scenario S. The agent may know different aspects of the actual state of the scenario and there may be environmental aspects that none of the agents knows. In addition, the agent interacts with each other as well as with the environment. That is,

$$\exists S_i, S_j \subseteq S | i \neq j \quad and \quad \sum_{i=1}^{n} S_i \simeq S \tag{4.2}$$

Where S_i refers to the part of S that is known to the agent team AT_i.

4.5.1 Multi-agent coordination

As mentioned in Chapter 2, there are three fundamental processes to solve the coordination problem [110]: mutual adjustment, direct supervision and standardization. Mutual

adjustment means that each agent is trying to adapt its behavior to improve the coordination. Direct supervision, means that there is one agent that can send orders to other agents. Finally, standardization means that there are some social laws enforcing the coordination among the agents. In this light, we think that multi-agent coordination problem is related to direct supervision and mutual adjustment processes. The latter, standardization, is not considered to be a proper dynamic coordination process because laws enforcing the coordination are fixed.

MUTUAL ADJUSTMENT COORDINATION PROCESS

Let us begin with agent teams within a mutual adjustment process. Suppose a group of agent teams within the cooperative scenario S. In this sense, agents with shared set of capabilities C_m and goals go_m should belong to the same cooperative group G_m (see figure 4.4). That is,

$$\exists c_i, c_j \in C_m | i \neq j \quad and \quad C_m \subseteq CP \tag{4.3}$$

and

$$\exists go_i, go_j \in go_m | i \neq j \quad and \quad go_m \subseteq GO \tag{4.4}$$

and

$$\exists c_m, go_m \in G_m \tag{4.5}$$

Where CP is the total of capabilities of the set of agent teams GA and GO is the total of goals to reach within the cooperative scenario S.

In this sense, a cooperative process of mutual adjustment allows agent teams with shared capabilities and goals join efforts in order to perform their tasks. In other words, agent teams make decisions about tasks assignment based on their capabilities and the constraints of the tasks to be performed. The tasks matching these constraints are scheduled for the agent team as their targets or goals.

DIRECT SUPERVISION COORDINATION PROCESS

Now, let us suppose that a supervisor or central agent CA_m is part of a cooperative group G_m of agent teams within the cooperative scenario S. In this sense, suppose that agents with shared set of capabilities C_m and goals go_m must be supervised by the same CA (see figure 4.5). That is,

$$\exists CA_i, CA_j \in G_m | i \neq j \quad and \quad G_m \subseteq GA \quad \text{where} \quad CA = \{CA_1, CA_2, ...CA_m\} \tag{4.6}$$

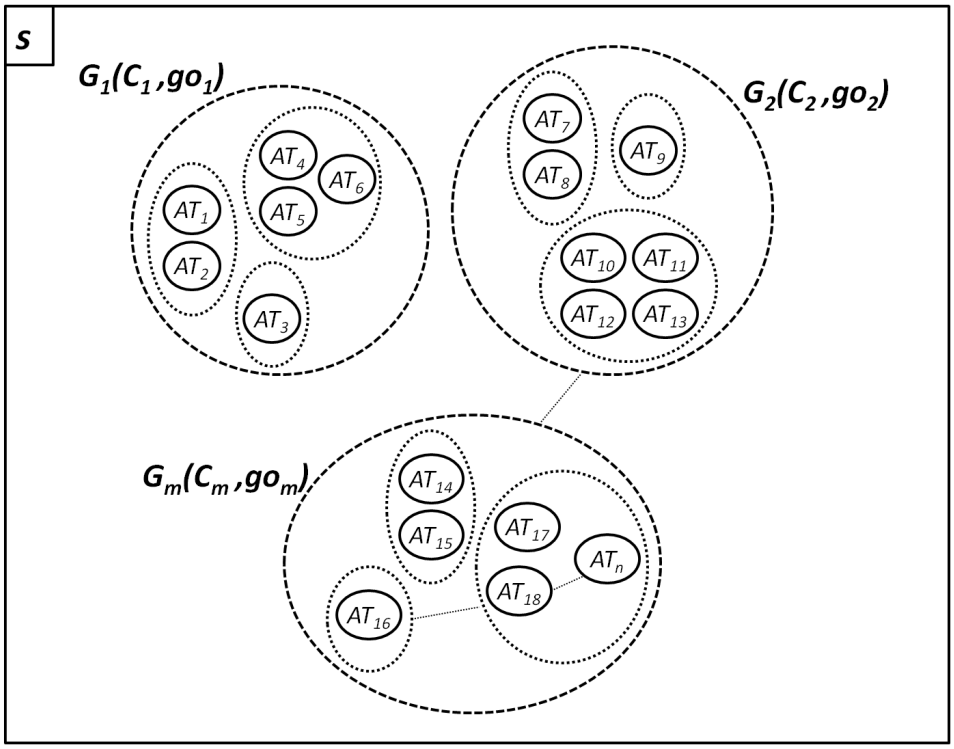

Figure 4.4: One general scheme of a group of agent teams in a mutual adjustment process.

and

$$\forall AT_i, AT_j \in G_m | i \neq j \, \exists \, CA_m \quad \text{and} \quad CA_m \subseteq CA \tag{4.7}$$

Suppose an agent team AT_i working to reach a determined goal go_m. Each goal go_m is composed by a set of tasks T_l to be performed and the set of needed capabilities C in order to reach the goal (see figure 4.6). That is:

$$\exists t_i, t_j \in T_l | i \neq j \quad and \quad T_l \subseteq TA \tag{4.8}$$

and

$$\exists t_i, t_j \in T_l | i \neq j \in go_m \quad and \quad go_m \subseteq GO \tag{4.9}$$

and

$$\exists c_i, c_j \in C_l | i \neq j \in go_m \quad and \quad go_m \subseteq GO \tag{4.10}$$

Where TA refers the number of total tasks to perform within the scenario S.

4.5.2 Agent team and task capabilities

Let us formalize the group of agent teams and tasks capabilities. The agents or tasks capabilities c can be defined by using a capability matrix [53]. Suppose there are together

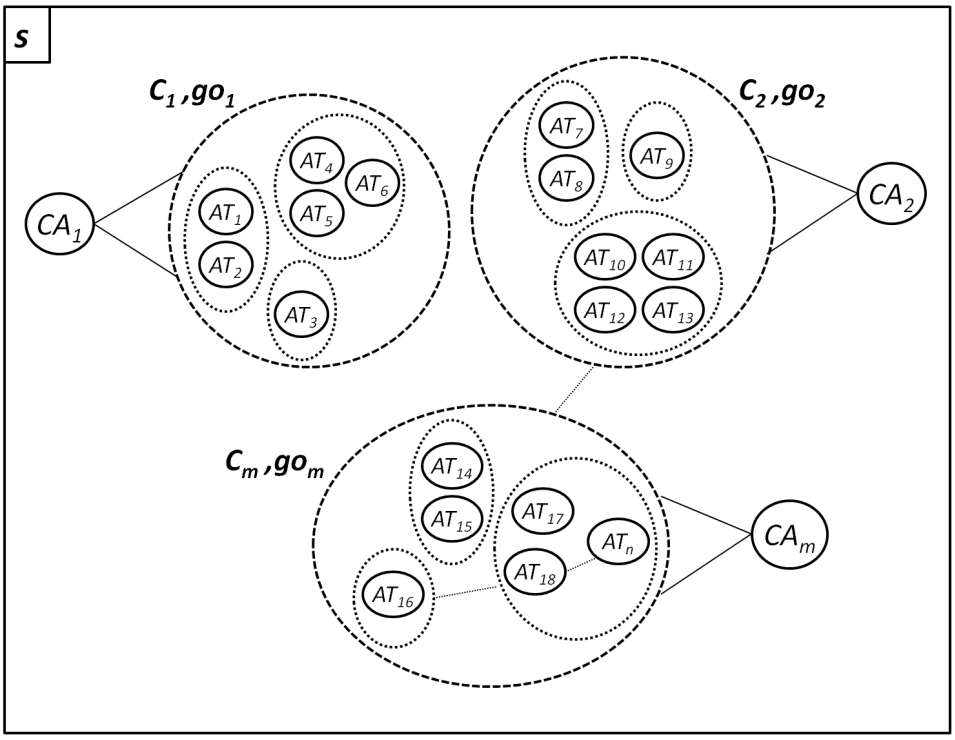

Figure 4.5: One scheme of supervisor agents within the scenario.

m capabilities and each one can be denoted as c_j, then the group of capabilities can be denoted as a set:

$$C = \{c_j\}, 1 \leq j \leq m \tag{4.11}$$

And, let us suppose there are n heterogeneous agent teams: AT_i, $1 \leq i \leq n$

Then for some agent team AT_i, $(1 \leq i \leq n)$, We can define its capability matrix C_i^{AT} such as that:

$$C_i^{AT} = \begin{bmatrix} \delta_{i1} & 0 & \dots & 0 \\ 0 & \delta_{i2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \delta_{im} \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_m \end{bmatrix}$$
$$\cong AT_i \cdot C$$

Where δ_{ij} corresponds to capability c_j of agent AT_i and $\delta_{ij} \geq 0$. If agent AT_i does not have capability c_j, then $\delta_{ij}\, equals\, 0$.

Let us define the task capability matrix. Suppose there are l independent tasks: t_k, $1 \leq k \leq l$.

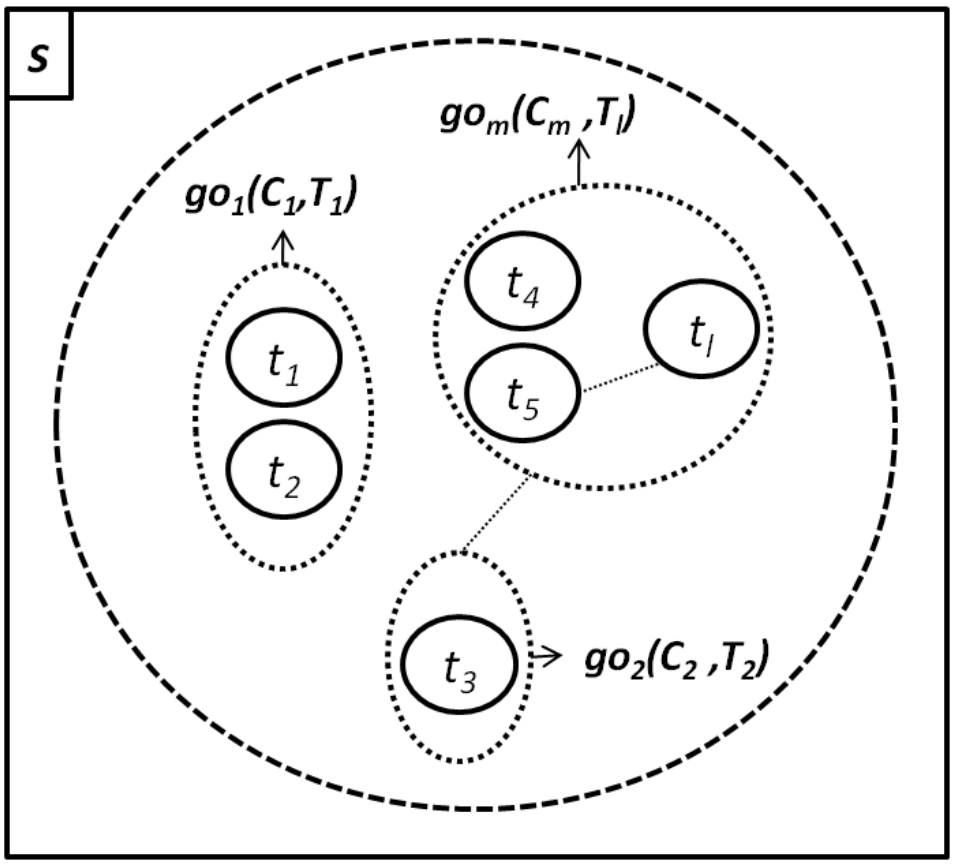

Figure 4.6: One scheme of the tasks and the capabilities needed to perform a determined goal within the scenario S.

Then for task t_k we define its corresponding capability matrix C_k^t such as follow:

$$C_k^t = \begin{bmatrix} \varepsilon_{k1} & 0 & \dots & 0 \\ 0 & \varepsilon_{k2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \varepsilon_{km} \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_m \end{bmatrix}$$

$$\cong E_k \cdot C$$

Where ε_{kj} corresponds to the needed capability c_j of task t_k and $\varepsilon_{kj} \geq 0$. If task t_k need not capability j, then ε_{kj} equals 0.

According to the capability matrix formulation, an AT in its environment must have some capability c_j to the accomplishment of a particular task t_k. Then, some AT_i, can accomplish some task t_k alone if: $\delta_{ij} \geq \varepsilon_{kj}$ and in the general formulation the condition is denoted as: $C_i^{AT} \geq C_k^t$.

On the contrary, some agent AT_i is unable to accomplish task t_k alone if: $\delta_{ij} < \varepsilon_{kj}$. In this situation it is claimed that agent AT_i can not accomplish task t_k alone.

4.6 Algorithms for distributed task allocation and scheduling

Task allocation is a very important issue among agents in a multi-agent collaborative environment. Some advantages of task allocation are that decision making process becomes more precisely and it's very similar to the process happens in the real world. Distributed task allocation algorithms require a set of agents to intelligently allocate their resources to a set of tasks. The problem is often complicated by the fact that resources may be limited, the set of tasks may not be exactly known, and may change over time [93]. In this sense, the complexity of the task allocation problem increases in an uncertain environment where the task can arrive dynamically and can change in intensity. The algorithms implemented in this thesis are trying to solve the coordination problem in open and dynamic multi-agent environments. To do that, we tackle both coordination processes: direct supervision and mutual adjustment [110].

The direct supervision process is based on a free market task allocation technique which uses combinatorial auctions for the assignment of tasks among agents. The mutual adjustment process is implemented using a due-date sequencing technique used in job shop scheduling.

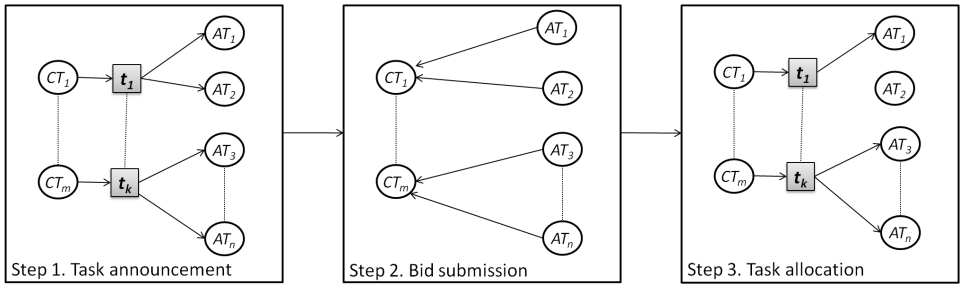

Figure 4.7: Typical combinatorial auction process. It can be divided into three main steps: task announcement, bid submission and task allocation.

4.6.1 Algorithm for task allocation using direct supervision

Several algorithms for task assignment based on direct supervision in multi-agent systems have been implemented [33, 60, 104]. In some previous free market based multi-agent coordination works, auction mechanisms is the main idea [39]. The idea of auctions is that some candidate agent reaches assigned tasks by bidding for these. In this light, some research brings forward the so called combinatorial auction method in the multi-agent domain [88, 46, 25, 55]. In combinatorial auction, some candidate agent team may gain more benefit by bidding on sets or bundles of items [112]. A typical combinatorial auction process is demonstrated in Figure 4.7. It contains three main steps: task announcement, bid submission and task allocation. In the first step, central or supervisor agents who know about tasks announce them to their surroundings. Then, the second step, each candidate agent team bids for its/their expected task/s (these may be a set of tasks). At last, when the supervisor agent has received all the bids, determines the winners and allocates the corresponding tasks.

In the optimal allocation process using combinatorial auction, some agents may remain unallocated (see Figure 4.7 and step 3). In this case, these agents should wait for next rounds of auctions with the aim that some tasks are assigned to them.

We have modified the typical combinatorial auction coordination process for adding and taking into account the agents and tasks capabilities in the bidding process, and to allow all the agents be allocated in each auction round. The algorithm steps are presented in figure 4.8. The *CA* is the supervisor or central agent and *AT* represents each agent team within the scenario. The algorithm steps are as follows:

1. Task recognition: Every *AT* that discovers new tasks sends to the *CA* the information of these tasks. The sent message includes the task attribute (for instance task location and capability matrix). The *CA* receives the tasks from the *AT*'s and builds a list of all the informed tasks (including the information of each task).

2. Task announcement: The *CA* commences an auction by announcing the tasks list and requesting bids from the *AT*'s.

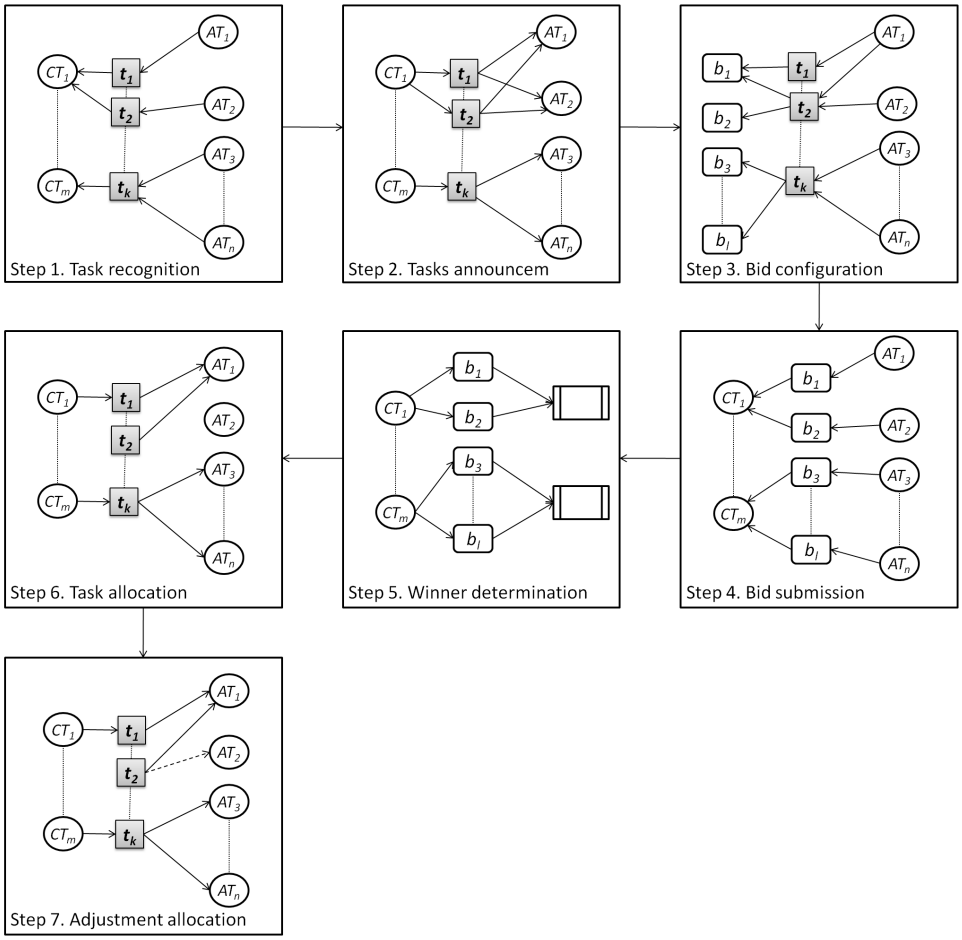

Figure 4.8: Proposed combinatorial auction coordination process. It can be divided into seven main steps: task recognition, task announcement, bid's configuration, bid submission, winner determination, task allocation and adjustment allocation.

3. Bid's configuration: An *AT* that receives information about the tasks being auctioned can respond with a bid if and only if has at least one existing task in its task list. A task can pertain to the bid if and only if the agent team's capabilities are enough to accomplish the particular task's capabilities. For instance, if the agent team is able to finish the task before of its due date time. In addition, if the agent has too much tasks to be performed, the tasks within the bid will be those with the higher priority. The value of the bid is given by the sum of distances from the *AT* current location to the tasks' location into the bid. Each *AT* sequences the tasks into the bid, in non-decreasing distance order.
4. Bids submission: Once the agent team has configured its bid, it sends it to the *CA*. If the configuration of the bid is not successful (for instance, there is no task that matches its capabilities), it follows looking for new tasks and waiting for the auction finalization. This agent will be allowed to perform the adjustment allocation step (see step 7 of this algorithm).
5. Winner determination: The *CA* continues to receive bids till all the bids from the *AT*'s have been received. Then, by some optimal algorithm for winner determination, it will select which team agents will accomplish which tasks. If the *CA* does not receive any bids it restarts the auction.
6. Task allocation: The *CA* informs the winning *AT*'s that they were selected to perform the tasks, while the *AT*'s that lost the auction are informed that they were not selected to perform the tasks.
7. Adjustment allocation: Agent teams which have not been allocated start an adjustment allocation process. This process consists in that the corresponding *CA* allocates tasks to unallocated team agents using some mutual adjustment mechanism (for instance, *CA* permits an unassigned *AT* to schedule their own tasks according to tasks priority).
8. Task execution: Each selected *AT* goes to the location of the task to perform it.

To adequate our algorithm to dynamic environments such as this of our interest, we had to tackle some issues such as: (1) After first auction agents send the new tasks found each time step, (2) Once some agent has finished their first tasks allocated by means of the algorithm in its first iteration, this agent starts a new auction cycle.

Our decision to use auctions as a coordination mechanism stems from the existence of several desirable properties of these approaches [124]:

Efficiency. Auctions are able to produce efficient solutions with respect to a variety of team objective functions. Given an initial solution, auctions can be used as a local search heuristic to improve the solution over time. Sandholm [84] has proven that the optimal solution can be reached in a finite number of auction steps with a sufficiently expressive set of bids, and Andersson and Sandholm [9] have demonstrated empirically that low-cost solutions can be reached given time limitations and a more restricted bids set.

Robustness. Market-based approaches can be made robust to several types of malfunctions, including complete or partial failures of robots and limitations of communications infrastructure. Additionally, these systems do not require a central coordinator agent that might create a single point of failure. However, taking into account the characteristics of the RoboCup Rescue scenario, central agents can be exploited, both, to coordinate their agent teams via auctions and to find tasks into the scenario. (central agents can view tasks that are outside the scope of the agent teams).

Scalability. The computational and communication requirements of market-based approaches are usually manageable, and do not prohibit these systems from providing efficient solutions as the size of the team or input increases. In the case of combinatorial auctions, although there is an exponential number of task subsets to consider, heuristic approaches to bundle reduction perform well in practice and such auctions can be cleared quickly.

Online input. Auction-based approaches are able to seamlessly incorporate the introduction of new tasks or the deletion of tasks, as well as the addition or removal of agents.

Uncertainty. Even with little or no prior information, market-based systems are able to operate in unknown and dynamic environments by allowing individuals to adapt cost estimates over time, and reallocate tasks when appropriate.

The auctions mechanism is natural for any protocol where agents have to allocate tasks and estimate costs via some established mechanism. Auctions can be seen from a centralized or distributed point of view and can offers solutions by both kind of approaches. Since we would like to compare agents' performance into distributed and centralized settings, an auction mechanism seems appropriate for our problem.

4.6.2 Algorithm for task allocation using mutual adjustment

Mutual adjustment permits agents organizing their own tasks. In this coordination model there is no outside control on decisions and agent teams coordinate their work themselves. Many times, in a multi-agent environment, a central agent or supervisor does not exist. This case may be a consequence of several aspects, for instance that the supervisor agent fails or be injured which avoids its proper operation.

Now, let us suppose a cooperative group G_m of agent teams in a dynamic environment S, for instance a rescue team in a disaster scenario, each agent team *AT* which is exploring the surrounding may recognize a number k of independent tasks t to be performed. In that situation, the agent teams need to schedule these tasks in order to accomplish them as soon as be possible. The scenario is such as is shown in Figure 4.9.

At first sight, it seems that every agent team is performing tasks in an independent way. However, that is not true at all because there is a group of intersection of tasks which may

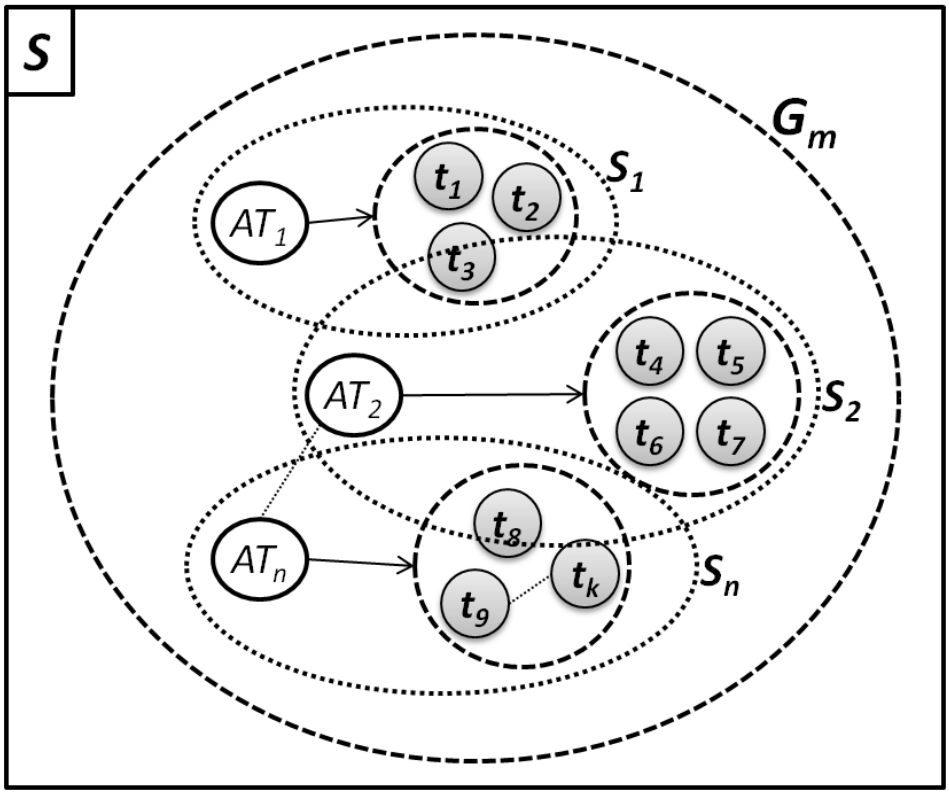

Figure 4.9: One scheme of a cooperative group within the scenario S. Each agent team has a scenario S_i which can perceive a certain set of tasks of the total scenario S.

be selected by the agent that are in more than one S_i scenario. That is,

$$\exists t_i, t_j \in S_n \mid i \neq j \quad and \quad S_n \subseteq S \quad \text{where} \quad S_i \cap S_j \neq \{\} \tag{4.12}$$

The current algorithm is based on the premise that tasks within the scenario S_i are the closest tasks to AT_i and for this reason, the cost of arriving until this particular set of tasks may be considerably lower for AT_i.

Then, based on this discussion, the key issue here deals with how every agent should schedule their corresponding tasks within its S_i scenario. The proposed algorithm is addressed to solve this scheduling problem. It is based on the special case of finite sequencing for a single machine and, particularly, with the sequencing theory according to due-dates.

4.6.3 SEQUENCING TASKS ACCORDING DUE-DATE

A problem of multi-agent tasks allocation can be modeled as a task scheduling problem in which each task consists of a single operation. Of the various measures of performance that have been considered in research on sequencing, certainly the measure that arouses the most interest in those who actually face practical problems of sequencing is the satisfaction of pre-assigned task due-dates [21]. Due-dates are used to sequence the tasks so that:

$$d_{[1]} \leq d_{[2]} \leq ... \leq d_{[n]} \tag{4.13}$$

According to the scheduling theorem by [40], when scheduling an $n/1//Lmax$ scheduling problem, the maximum task lateness and maximum task tardiness are minimized by sequencing the tasks in non-decreasing due_date order. In 4.13, $d_{[i]}$ corresponds to the due-date of each task; and the $n/1//Lmax$ notation corresponds to allocating n number of tasks one machine (agent), so as to minimize the maximum lateness.

TASK COMPLETION DESCRIPTION

As we take into account the tasks completion details to task allocation, we should discuss in more detail about the task capabilities involved in the agent's operation. In section 4.4.2, we gave a formal description of task and agent team capabilities. In this description capability matrix is defined as a metric of different capability aspects. We have instanced these formal aspects of capabilities to multi-agent environments. In our approach the task capabilities refers the aspects which are constraining agents during the task completion. Aspects such as for instance, the due date for the task, task completion time, task duration, and priority of the tasks are determining on the decision making of an agent about task scheduling and

planning. In addition, we take into account other aspects of agent capabilities, such as the possibility that an agent team can accomplish a particular task. Next, these main aspects of task and agent capabilities are presented:

1. Due date for the task *"due_date"*: It is the time by which the processing of the task should be completed.
2. Start time of the task t_s: It is the time at which a task may begin in the schedule. This time is domain-dependent. For instance, in a robot allocation problem, this time would be the instant when the robot is aware of the tasks within the scenario [31]. Other example is the start time of a task related to rescue a victims. In this case, the start time would be the detection time of the victim through exploration.
3. Task completion time *"tct"*: It is used to indicate actual or estimated completion time for one task. Times can be entered in any of the following time units: Seconds, minutes, hours, days, weeks.
4. Real task duration *"rtd"*: It is the real duration time in which one agent team (or group of agent teams) can perform one task. In real-world environments, it is normal that agent teams cooperate to accomplish a particular task. For instance, suppose an agent team in a disaster scenario that detects a victim. Such as any dynamic environment, this victim may be being rescued for another agent teams at the same time. Then, in this situation the agent has to make decisions about including or not this victim into its scheduling and that way to cooperate with their team mates. It means that an agent team has to take into account the total number of agents that are performing the task at the current scheduling time (it is the m parameter on equation 4.14). The β parameter is an average value to express a time period additional and domain-dependent, for instance the average arrival time to the victim, the period to load and unload, or the arrival time to the refuge (for a disaster scenario). That is,

$$rtd(t_i) = \frac{tct(t_i)}{m} + \beta \tag{4.14}$$

5. Earliest completion time *"ect"*: The earliest time for completion of one task or group of tasks into the current scheduling. That is,

$$ect(t_j) = rtd(t_0) + \sum_{i=1}^{i=j} rtd(t_i) \tag{4.15}$$

6. Priority of the task *"pr"*: Prioritization is a method of giving specific tasks preference in being dispatched. That is,

$$pr(t_j) = \frac{ect(t_j)}{due\_date(t_j)} > \theta \tag{4.16}$$

7. Possibility of task completion *"ptc"*: It is a way to measure if one agent is able to accomplish a particular task (See equation 4.17). In this sense, one agent is able to perform a task if the possibility of task completion is < 1. That is,

$$ptc(t_j) = \frac{ect(t_j)}{due\_date(t_j)} < 1 \tag{4.17}$$

Now, that we have explained the main parameters and equations of our approach, the steps of the task scheduling algorithm are as follows:

1. The *AT* that discovers a task *t* commences the tasks sequencing by computing its due date and earliest completion time by using equation 4.15.
2. Using the computed times in step 1, the *AT* computes the priority of the task and the possibility of task completion by using equations 4.16 and 4.17.
3. The *AT* that has discovered the task *t* can include it into its targets' list if only if: *ptc(t)* <1 and *pr(t)*$> \theta$. The θ value is fixed depending on the simulation results.
4. If the targets list of the *AT* is not empty, it goes to the first task location in its list and while it also informs to the *CA* about the tasks' location and due date which have not been included into its targets' list.
5. The *CA* schedules the discarded tasks received from the *AT*'s according to a non-decreasing *due_date* order.
6. If *CA* has scheduled tasks, send them to the idle *AT*'s, at this way idle *AT*'s can know the task location and go to perform it.

In this algorithm the *CA* has a role of a data base of tasks to be completed. That way, lost or free agents may know about higher priority or urgent tasks within the environment.

4.6.4 Replanning of tasks

Dynamic task allocation involves both the initial allocation of tasks within a team and reallocation in response to anticipated problems in task execution [64]. Many real life problems require planning in which a number of participants (for instance agents) perform tasks collaboratively in order to achieve a goal [114].

In a dynamic environment, execution errors may happen to any plan due to uncertainty and failure of individual actions. Therefore, an indispensable part of a planning system is the capability of replanning. The importance of replanning in executing a plan has been discussed in [22]. Whilst a number of approaches have been proposed for distributed planning and replanning, most of them assume that the planning and replanning are carried out by a set of agents who have complete knowledge of the environment [122]. This assumption does not hold for dynamic and open multi-agent environments where agent teams have to interact and collaborate in order to accomplish their tasks.

In this dissertation we address this limitation of existing approaches by proposing an algorithm for distributed replanning in dynamic environments where agents have incomplete information of the surrounding.

In general, replanning process fixes a plan with two basic operations, unrefinement and refinement [47]. The unrefinement operation removes some actions that may obstacle the reach ability to goal. The refinement operation adds actions that improve the reach ability to goal.

We use an unrefinement strategy to allow agents replanning their scheduled tasks in any moment that their plans fail. In our approach the aim of replanning is to solve task execution failures, which may happen in any action of the plan execution. Then, our agent teams start replanning from any action that fails during the operation. For instance, the inability of agent teams to move and reach the goal.

The proposed re-allocation or replanning algorithm is thought to be applied in the case when task allocation planning fails due to extern constraints to the agents. In the most of cases, these constraints are related to the environment. The algorithm is motivated to multi-agent environments which present obstruction problems in the motion of the agents in order to reach the tasks (For instance, an agent that moves towards an aim but finds an object on the way that incapacitates its movement).

Let us suppose a multi-agent environment where agents have some planned tasks by means of any allocation mechanism. Sometimes agent teams may find problems in order to reach the tasks due to different factors of the environment, for instance an agent or robot is blocked in a road due to environmental conditions such as rubbles on the way or not accessible paths. This situation avoids agent to accomplish their goals due to several motion problems. In such issue, the agent team needs to replanning the scheduled tasks in order to accomplish its main goal.

Our task replanning algorithm is shown in Figure 4.10. In this every agent *AT* temporally ”forget” the unreachable tasks *t* keeping them in a list of delayed tasks. Then, *AT* continues with the development of the next *t* in its schedule. In successive cycles, once *t* is reachable (for example the obstructed road is cleared) *AT* is informed about that by the central agent *CA*. Then, it re-schedules *t*, introducing it in its list of pending tasks. Previously, the agent verifies that *t* is realizable, for example, if the conditions have changed so much that currently is not possible to perform *t*, in this case it is not included in the tasks list again and it is forgotten forever. To see more details of the implementation of this algorithm, see section Implementation in Chapter 5.

The proposed algorithms in this chapter are motivated by the needs of task allocation and replanning in the concrete domain of the RoboCup Rescue. In next chapter, the implementation details of the approach are presented.

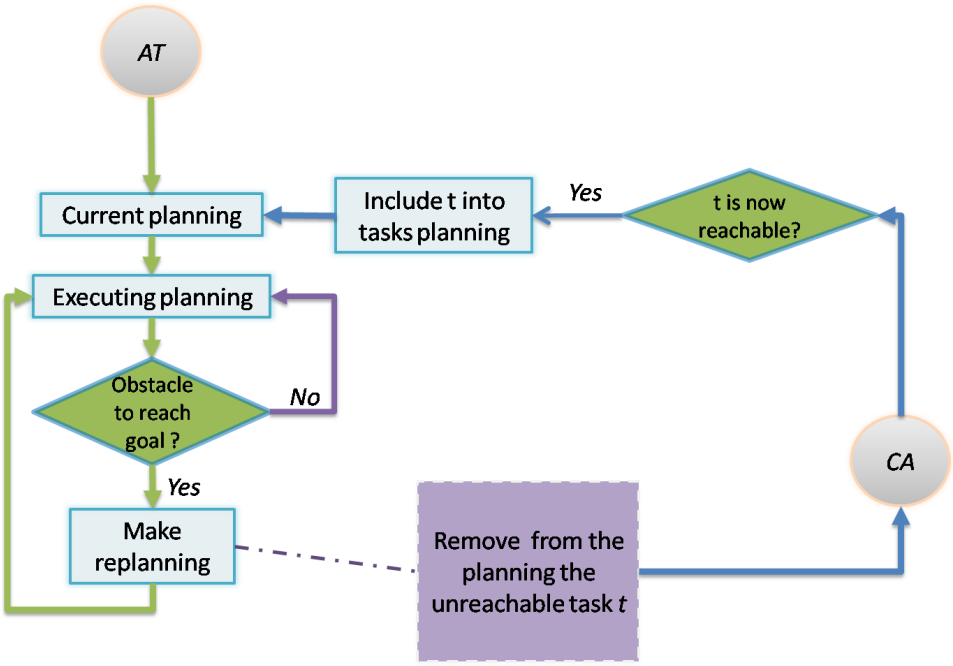

Figure 4.10: Replanning algorithm.

CHAPTER 5
Implementation and results

This chapter presents the application of the proposed approach on the RoboCup Rescue scenario. It describes the implementation of the approach presented in Chapter 4. Firstly, The SATA implementation is presented. The sequencing technique, the death time estimation and implementation details of this algorithm are presented. Then, we follow with the CATA algorithm implementation. The combinatorial auction formulation, the RoboCup Rescue Formulation, the bid configuration and the winner determination algorithms are described in this chapter. In addition, we present the implementation of task replanning. Finally, the results of this research are presented at the end of this chapter.

In this dissertation, we focus on the decision process regards to task selection and allocation that must be implemented by a rescue agent. The ultimate goal of this work is that our agents exhibit rational behaviour. In other words, the agent would always act to minimize the cost to perform their tasks and maximize the utility of its operation. In this chapter, we will show the process from the agent team viewpoint and observe the various decisions that such an agent must make. In addition, we will focus on the problem of task scheduling and allocation and the implementation details of this work in the RoboCup Rescue simulator.

We have used the RoboCup Rescue scenario presented in section 2.10 as a testbed. Such as it was previously mentioned, the RoboCup Rescue scenario is composed by six kind of agents: ambulance team, fire brigades and police forces and the three stations (ambulance center, fire station and police office) managing each one of these teams. The decision process starts when the agent finds one task or group of task to be performed. For instance, imagine an ambulance team finds a group of five victims to be rescued from the debris in the location (building, road or node) id 234567. Attributes of the victims, for instance, might include the cost (incurred by rescuing a particular victim) and the victim's death time.

In the previous example, the victims have been found through exploration. In other cases, the victims may be known by means of communication (from the central agent which has a more global vision or other agents into the scenario). Figure 5.1 shows the victims sequences found by the ambulance in a particular time. Our example deals with rescue victims because it is the most important issue in the crisis management, much more than to extinguish fires, clear roads or other logistical issues. So far, our example is not situated in time and we have not discussed the incurred cost for completing this rescue plan. In

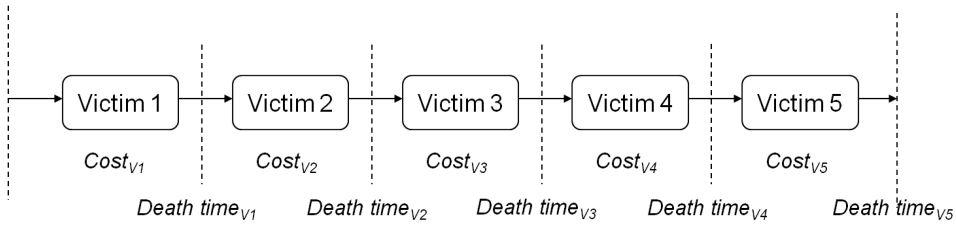

Figure 5.1: Plan for the example of victims' rescue.

a rescue operation, the quality of the scheduling of victims is very important. The victims should be rescued as quickly as be possible in order to give them all the possibilities to be rescued alive. We are treating the victim rescue as atomic and the recovery of each victim must be completed before his death time is over.

In the RoboCup Rescue simulator, there are civilian agents that are controlled by the simulator and represent people to be rescued. Civilians can be trapped under rubble, which affects the time needed to rescue them. The simulation begins with a simulated disaster (generally an earthquake). The disaster simulators begin simulating collapsed buildings, fires, and blocked roads. The civilian simulator tracks the health of each civilian, which deteriorates over time if the civilian is trapped or on fire. The goal of the simulation is to minimize both civilian casualties and property damage (due to fires, which spread quickly).

These civilians are wounded when they are buried in collapsed buildings and they can die if they are not saved fast enough. The health of injured civilians can worsen with time. Therefore, the ambulance team agents have to dig in the detritus of the collapsed buildings to save the civilians that are trapped. Afterwards, they have to transport them to refuges where they can be treated. The ambulance team agents are helped in their work by the ambulance center agent with which they are in contact by radio.

The RoboCup Rescue scenario is a system where the set of tasks is not initially known. Thus, the agents have to explore the environment to find the tasks to accomplish. In the RoboCup Rescue, this means that the ambulance team agents have to explore the collapsed buildings to find the injured civilians. These civilians are seen as tasks and since the health state of a civilian is uncertain, the parameters of the tasks could change in time. For example, if one civilian catches on fire, its expected death time would drop rapidly. In other words, the deadline of the task would be reduced. Notice that, there are also important constraints on the communications in the RoboCup Rescue: agents are limited in the number of messages they can send or receive and the messages' length is also limited. With these limitations, it becomes primordial to manage the communications efficiently (See Figure 2.6) in Chapter 2.

5.1 SCHEDULING OF TASKS

In complex multi-agent systems such as the RoboCup Rescue scenario, the agents could be faced with many tasks to accomplish. In addition, when the tasks have different due dates, the order in which the tasks are accomplished becomes quite important. In such settings, agents have to decide how many agents to assign to each task and in which order they should accomplish the tasks. To achieve that, agents need some scheduling algorithm that can maximize the number of tasks accomplished before their due date.

Into the Artificial Intelligence community, scheduling is a special case of planning in which the actions are already chosen, leaving only the problem of determining a feasible order [100]. Other definition less trivial states that scheduling is the problem of assigning limited resources to tasks over time to optimize one or more objectives [57]. Our work deals with scheduling problems in which all tasks have the same value and where the set of tasks may be not accomplishable in the time allowed. According to multi-agent and scheduling systems paradigms, we consider agents to be resources that can accomplish tasks. We talk about an agent accomplishing a task and about the allocation of an agent to a task. In this sense, this kind of scheduling problem tries to solve the agents' problem which has to schedule the tasks in order to maximize the number of tasks accomplished.

Furthermore, in multi-agent systems, the scheduling can be done in a centralized [27] or decentralized [106] way. Likewise, the execution can also be done in a centralized or decentralized way. Centralized scheduling means that there is one agent responsible for scheduling the tasks of all the agents. On the other hand, in decentralized scheduling each agent is responsible for the scheduling of its tasks.

As it was mentioned in Chapter 4, task coordination involves mainly two models, mutual adjustment and direct supervision. In the next sections, we present our decentralized implementation (mutual adjustment) and our centralized implementation (direct supervision) to allow rescue agents to schedule and allocate tasks among themselves. The first one is implemented by our *Scheduling Algorithm for Task Allocation* (SATA) and the second one, is implemented by our *Combinatorial Auctions for Task Allocation* Algorithm (CATA). The RoboCup Rescue has been used for this implementation.

5.2 SCHEDULING ALGORITHM FOR TASK ALLOCATION (SATA)

The rescue allocation problem can be modeled as a task scheduling problem in which each task consists of a single operation [74]. Our approach is concerned with the special case of finite sequencing for a single machine. Particularly, with the sequencing theory according to due-date which is explained in section 4.5.3 of chapter 4.

5.2.1 SEQUENCING ACCORDING TO DUE-DATE

Our algorithm is mainly based on the Earliest Due Date algorithm (EDD) [40]. To schedule a set of tasks, the EDD algorithm sorts all the tasks in the non-decreasing order of their due dates. Consequently, the first task to execute is the task with the earliest due date. This algorithm has complexity $O(n \log_n)$ [14] (where n is the number of tasks). This algorithm is optimal if there is no overload, for example, if it is possible to accomplish all the tasks in the given time. Although some overload could happen in RoboCup Rescue environment, the performance of this algorithm is satisfactory, as presented in the experimentation section.

Since EDD is a greedy algorithm, it is possible to just find the first task to accomplish without having to schedule all the tasks. This first task is simply the task with the earliest feasible due date . This property of the EDD algorithm is interesting in dynamic environments where the agents have to react to changes in the environment. With this algorithm, the scheduling can be done really fast, because the scheduler agent only has to do one iteration over the tasks to find the next task to accomplish. This type of greedy algorithm is well adapted to a problem of decentralized decision making because it is never necessary to reconsider a decision previously made. This enables agents to find the next task to accomplish in time $O(n)$, where n is the number of tasks [14].

5.2.2 SCHEDULING PROBLEM WHEN RESCUING VICTIMS

The current scheduling algorithm has been implemented in the ambulance team agents because of they need to sequence the victims taking into account the priority of the victims, mainly the death time of each one. In our rescue scheduling problem, we have n victims considered as n tasks and each ambulance agent is considered as one resource (machine). At the beginning of the simulation, victims are distributed throughout the scenario and ambulance agents are exploring the surrounding area. Once ambulance agents know about a victim or group of victims, they have to make the important decision about how to rescue them. In this sense, our victims' scheduling algorithm, which is shown in Figure 5.2, has three relevant issues. First, victims' scheduling that refers to organizing the set of victims to be rescued. Victims are scheduled according to death time using the EDD algorithm. Secondly, the algorithm selects highest priority victims: it is a key issue in this scenario to first rescue civilians who will die without our help meanwhile lowest priority victims are reported to the ambulance center. And thirdly, only civilians that could be rescued are returned and those that cannot be rescued before their deadline are discarded. Our scheduling algorithm is shown in Figure 5.2. In the algorithm, the $V = \{V_1, V_2, \ldots, V_n\}$ set are the victims. The m parameter is the number of ambulance teams rescuing the civilian. The δ value is fixed depending on the simulation results and id corresponds to the identification number of the victim. $VictToRescue$ is the list of victims allocated for rescue. The $deathTime(V)$ parameter is a computation of the remaining time before the victim dies. The death time estimation method is presented in section 5.2.3.

```
Scheduling Algorithm
returns VictToRescue, an optimal list of scheduled victims
    respecting their death time.
Input: V, a set of victims to schedule.

V ← EDD(V)
For V1,.... Vn
    deathTime(Vi)= deathTime_estimation();
    m=Number of Ambulances;
    RescueTime(Vi)=Buriedness(Vi)/m + 6;
    earlyComplete=earlyComplete+RescueTime(Vi);
    if (deathTime(Vi)>earlyComplete) &&
    (earlyComplete/deathTime(Vi)>δ)
            VictToRescue.put(deathTime(Vi),Vi)
    else
            Tell(id(Vi),deathTime(Vi)) To Central
End For

If (VictToRescue isNotEmpty)
     Rescue(VictToRescue)
else
    Continue exploring the surrounding.
```

Figure 5.2: Scheduling algorithm in ambulance team agents.

The rescue time is the time used for an agent to rescue a particular victim in the rescue scenario. The rescue time for a victim $RescueTime(v)$ is computed by using 5.1

$$RescueTime(v_i) = \left\{ \begin{array}{ll} m = 1 & buriedness(v_i) + \beta \\ m > 1 & \frac{buriedness(v_i)}{m} + \beta \end{array} \right\} \tag{5.1}$$

Regarding the m parameter (m=number of ambulance teams), a greater number of ambulances can rescue a victim in less time. The $buriedness(v)$ parameter shows how much the civilian v is buried in the collapse of buildings: a value of more than one means that it cannot move by him/herself. Each ambulance can decrease the $buriedness$ of the buried victim by a value of 1 each time and can dig up and carry only a single victim. The ambulance

Scheduling Algorithm in Centrals
returns *VictToRescue_Rec, an optimal list of scheduled victims respecting their death time.*
Input: *V, a set of victims to schedule.*

```
For each Vi
Hear (deathTime(Vi)) from Ambulance Agent;
        deathTime_Rec(Vi)=deathTime(Vi);
        VictToRescue_Rec.put(deathTime_Rec(Vi),Vi)

VictToRescue_Rec ← EDD(VictToRescue_Rec)

If (VictToRescue_Rec isNotEmpty)
        Send(First victim in VictToRescue_Rec) to Ambulance
```

Figure 5.3: Scheduling algorithm in ambulance central agent.

team set an average value of β=3 (time steps or seconds of simulation).

An ambulance center's scheduling algorithm is presented in Figure 5.3. This algorithm is in charge of the lowest priority victims discarded by the previous algorithm (Figure 5.2). The center's role is to communicate these victims to free ambulance agents. The algorithm in Figure 5.3 also schedules victims according to a non-decreasing death time order.

Before introducing the victim into the scheduling, agents have to decide whether the victim can be rescued alive and also if he or she has high or low rescue priority.

$$PossibilityOfRsc(v_j) = \Big\{ \frac{earlyComplete(v_j)}{deathTime(v_j)} < 1 \tag{5.2}$$

$$earlyComplete(v_j) = RscTime(v_0) + \sum_{i=1}^{n} RscTime(v_i) \tag{5.3}$$

In Equation 5.2 if *PossibilityOfRsc(v)<1*, then the victim can be rescued alive. The *early-Complete(v)* parameter is the earliest time at which the agent can efficiently rescue the set of victims inside the scheduling and in this case n is the number of victims. The v parameter represents one victim or a group of victims, depending on the number of detected victims on the rescue site. In addition, agents calculate the victims' emergency or priority using equation 5.4.

$$priority(v_j) = \Big\{ \frac{earlyComplete(v_j)}{deathTime(v_j)} > \delta \tag{5.4}$$

The ambulance team set the values of δ parameter. Then, if $PossibilityOfRsc(v) < 1$ and $priority(v) > \delta$, victim v is introduced into the ambulance agent's scheduling.

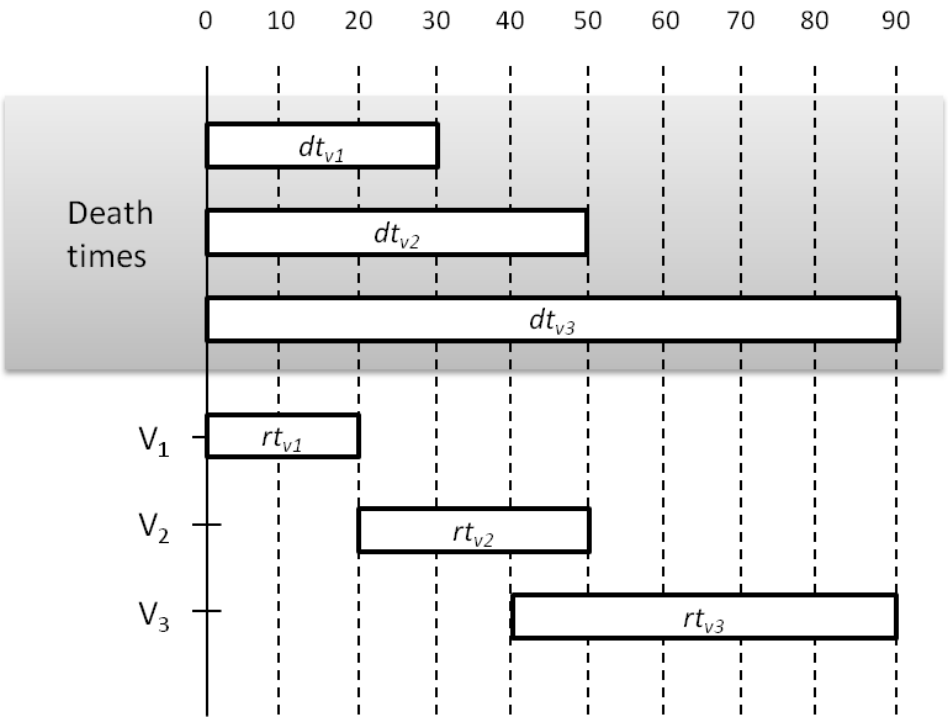

Figure 5.4: Sample bid configuration showing v_2 and v_3 infeasibilities.

Figure 5.4 shows a small example with 3 victims v_1, v_2, v_3, each victim with a death time of 30, 50 and 90 respectively and each one has a rescue time of 20, 30 and 50 respectively. The infeasibility between victim v_2 and v_3 to belong of the agent's schedule is captured by the constraint:

$$x_3(40) \geq x_2(20 + 30) \tag{5.5}$$

In this sense, agents take into account the local flexibility such as it is generalized in the next definition:

Definition: Local feasibility. Each task v_j must be able to start after the earliest possible completion time of each of its predecessors $v_{j'}$ and it must be able finish before due date time. In other words, here we are looking at the start times of each individual task v_j and its immediate predecessors P_j. It means each agent must take into account the feasibility of the task completion. That is,

$$\begin{aligned} \text{For each} \quad & v_j \in V_r, i : v_j \in V_i, i' : v_{j'} \in (V_{i'} \cap P_j), \\ & x_i t_s(v_j) \geq x_{i'}(t_s(v_{j'}) + rtd(v_{j'}))) \end{aligned} \tag{5.6}$$

5.2.3 VICTIM'S DEATH TIME ESTIMATION

An outstanding factor in successful task scheduling is being able to estimate an accurate death time for each civilian. In the RoboCup Rescue simulator, ambulance team agents have to predict the civilians death times in order to identify the civilians with the highest priorities. In this sense, the death time estimation is based on victim's *hp* and *damage* values. The *hp* value represents the life quantity of the victim. The *damage* is the number of health points (*hp*) that an injured civilian looses per time step $(hp_t = hp_{t-1} - damage)$. If the number of health points reaches zero, then the civilian is considered as dead. A civilian (*hp*) is initially set to 10000. An injured civilians *damage* is always ascending, it means, that it increases at each time step, until the civilian dies or the civilian reaches a refuge, in which case it is set to 0. Figure 5.11 shows the progression of the *damage* attribute for many civilians. As we can see in the figure, there are some tendencies in the damage progression of the civilians.

In the RoboCup Rescue scenario, the position and the health status of the civilians are unknown at the beginning of the simulation. All rescuing agents have to search in the collapsed buildings to find the buried civilians. When a civilian is found, the rescuing agents can see the current state of health of the civilian and they can try to predict its death time. Some methods have been developed to achieve an acceptable prediction of civilian death-time in the RoboCup Rescue domain [8, 97, 118, 41] most of them are using a formula based on the victim's (*hp*). For example [8] use a function $hp(t) = A * \exp^{dt}$ which is used to estimate a victim's *hp*. In this function A denotes a victim's original hp, d denotes damage and t is current simulation cycle.

In order to compute accurate enough death-time values, we have used a case based approach to model behavior by logging its output for various inputs in the miscellaneous sub-simulator of the RoboCup Rescue simulator. The agents status is modeled by this miscellaneous simulator. When an agent is inside a burning building or trapped under debris, its health is affected and starts decreasing. This is the module responsible for controlling the agents properties in these situations. In this sense, we retrieved previous experiences in our simulations, in order to create a complete case base of civilians' death times in each time. for that purpose, the outputs of three different maps of the simulator have been used. These ones are shown in figures 5.5, 5.6 and 5.7. Based on these outputs, we have created a case base of (hp/damage $\rightarrow$ death time) peers which is used for victims' death time estimation in future simulations. First, the mechanism looks for the group (GA) of victims inside of the case base, whose hp values correspond with the hp value of the current victim. Then, the mechanism looks for the subgroup (GB) of victims inside of the group (GA) whose damage values correspond with the damage value of the current victim. The output of the algorithm is a group (G_{dt}) of (hp/damage $\rightarrow$ death time) peers. Three criteria to find the death time value are used: median, mean and minimal inside of G_{dt}.

Figure 5.11 depicts the *damage-time* evolution for each victim in the three disaster maps during 300 cycles. The time-hp and time-damage curves are similar on each map. Victim's

Figure 5.5: Foligno (Italy) map of the RoboCup Rescue simulation scenario.

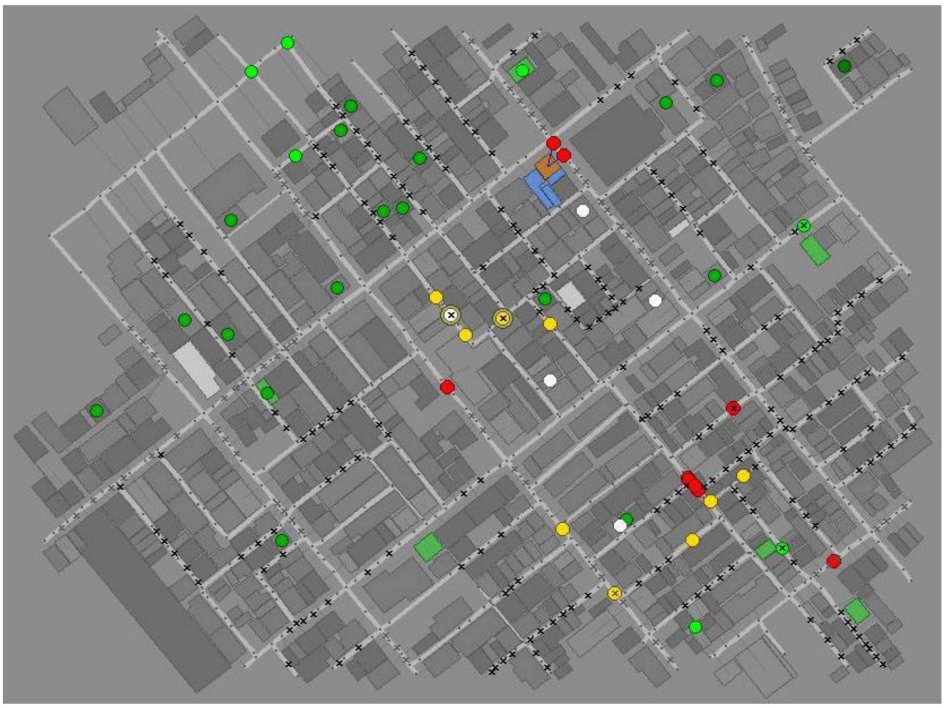

Figure 5.6: Kobe (Japan) map of the RoboCup Rescue simulation scenario.

Figure 5.7: Random map of the RoboCup Rescue simulation scenario.

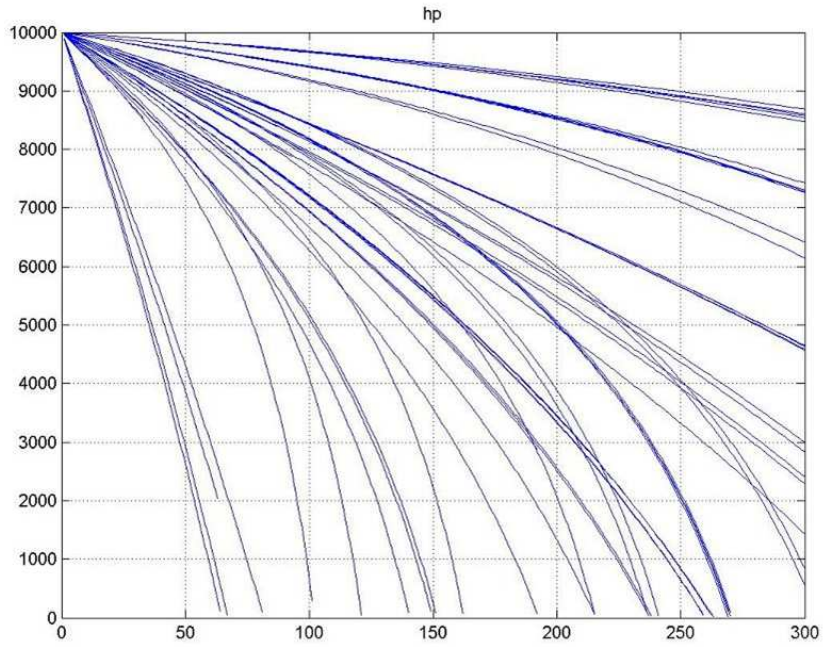

Figure 5.8: Health points (hp) of victims on the Foligno Map for 300 cycles.

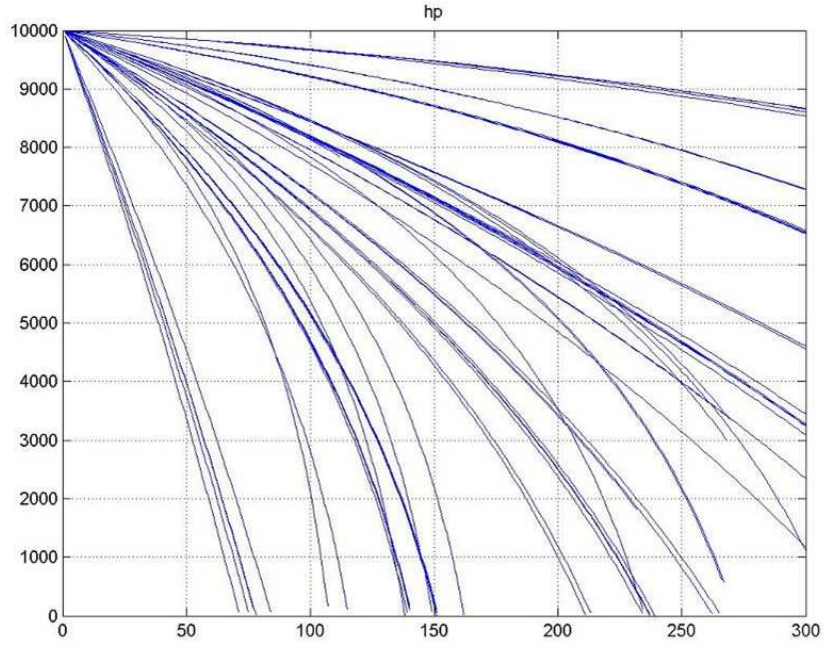

Figure 5.9: hp of victims on the Kobe Map.

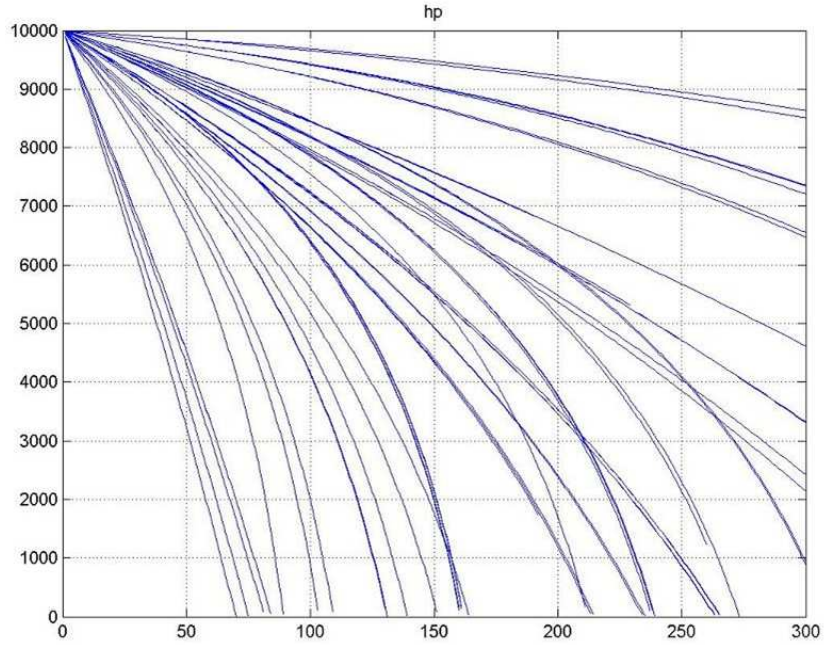

Figure 5.10: hp of victims on the Random Map.

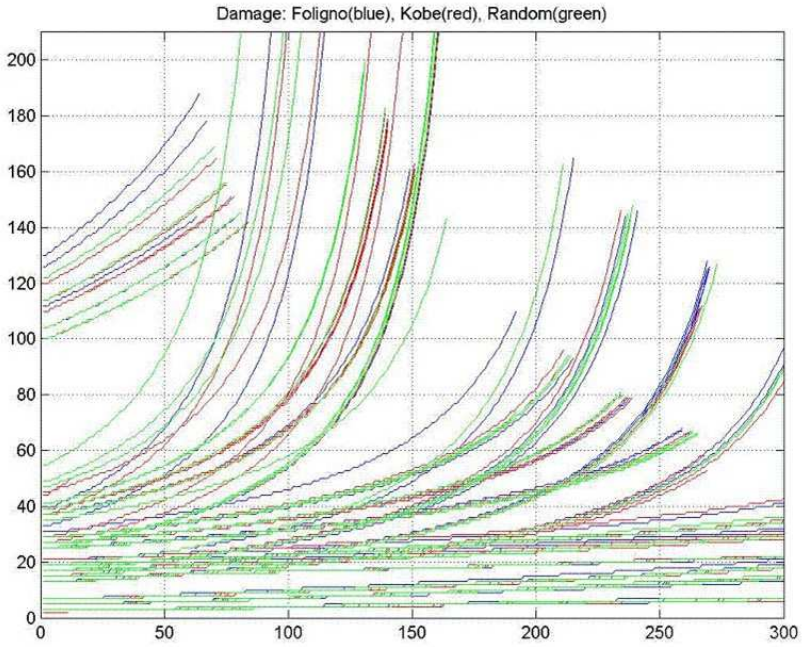

Figure 5.11: Damage to victims on all three Maps.

damage curve is an exponential function with a reasonable error. It is described in equation 5.7.

$$damage(t) = damage(0) * e^{\frac{t}{\tau}} \tag{5.7}$$

Figures 5.8, 5.9 and 5.10 show the hp-time curve in the different scenarios during 300 cycles. hp is described in equation 5.8.

$$hp(t) = hp(0) - \int_0^t damage(t)dt \tag{5.8}$$

Replacing 5.7 in 5.8:

$$hp(t) = hp(0) - \int_0^t damage(0) * e^{\frac{t}{\tau}} \tag{5.9}$$

Then, the hp curve is described by equation 5.10.

$$hp(t) = hp(0) - damage(0) * \tau * (e^{\frac{t}{\tau}-1}) \tag{5.10}$$

Assuming that if hp(t) = 0 the victim has died, death time may be estimated as is shown in 5.13.

$$hp(t) = 0 \tag{5.11}$$

Replacing 5.11 in 5.10:

$$\frac{hp(0)}{damage(0) * \tau} + 1 = e^{\frac{t}{\tau}} \tag{5.12}$$

$$deathtime = \tau * \ln\left(\frac{hp(0)}{damage(0) * \tau} + 1\right) \tag{5.13}$$

In the next section, the combinatorial auction algorithm for task allocation is presented. This algorithm has been implemented to help fire brigades to coordinate on the buildings on fire to extinguish. In the RoboCup Rescue scenario, a fire brigade can discover a fire within a distance D roughly proportional to its fieriness, where $D = K*(cycles\ from\ the\ start\ of\ burnup)$ The fire station agent has a better global view of the situation and therefore it can suggest fires to fire brigade agents. The fire brigade agents have however a more accurate local view, for this reason, they can choose by which particular building on fire to extinguish they want to bid. In this sense, by using auctions, we can take advantage of the better global view of the fire station agent and the better local view of the fire brigade agent.

5.3 Combinatorial Auctions for Task Allocation (CATA)

In the task allocation problem a team of agents is required to accomplish a set of tasks according to a given criteria. This criterion can be either minimizing the time to accomplish all tasks and maximizing the utility (or minimizing the cost) of the accomplished tasks in a given time frame. In our task allocation problem agents are trying to minimize the cost and to perform the tasks as quickly as be possible.

In this seccion, we will show the implementation of the direct supervision algorithm presented in section 4.5.1. This algorithm have been implemented by using combinatorial auction techniques. In e-commerce, a combinatorial auction is a kind of auction where there exists a set of sellers willing to sell a set of items to a set of buyers. The buyers send bids that are combination of these items with one cost, and the objective is to find the bids combination that provides the biggest benefit for the seller or sellers. How it was described in the background chapter, there exist several classes of auctions. Particularly, we use reverse combinatorial auctions which are explained in the next section.

5.3.1 Reverse combinatorial auction formulation

On the contrary to the general combinatorial auction formulation, in the reverse case, the roles of auctioneers and bidders are exchanged. According to [90] reverse combinatorial

auctions are widely used in procurement in where there is a buyer who wants to obtain some goods at the lowest possible cost, and a set of sellers who can provide the goods. The buyer can hold a reverse auction to try to obtain the goods. If there is complementarity or substitutability between the goods, a combinatorial reverse auction can be beneficial. Each seller submits ”asks” that say how much the seller asks for each bundle of goods it can provide.

More formally, a traditional combinatorial reverse auction is an auction in which the buyer wants to obtain a set of items, $M = \{1, 2, ...m\}$, and the sellers submit a set of asks, $B = B_1, B_2, , B_n$. An ask is a tuple $B_j = (S_j, c_j)$, where $S_j \subseteq M$ is a set of items and c_j, $c_j \geq 0$ is an asking cost. How to determine the combination of winner bids (i.e. the ones that minimize the cost) is known as the winner determination problem. The integer programming formulation for the winner determination problem consists on labeling the asks as winning or losing so as to minimize the buyer's cost under the constraint that the buyer obtains each item:

$$Minimize \sum_{i=1}^{n} c_i x_i \quad \text{subject to the } m \text{ constraints} \\ \sum_{i:s_j \in S_i} x_i \leq 1 \quad for\, each\, s_j \in S_r \tag{5.14}$$

where c_i is the bid price for bid b_i, S_i is the set of items specified in b_i ,and the constraints ensure that at most one bid is accepted for each item. The notation $i : s_j \in S_i$ denotes the set of all i such that the task s_j is in the set of tasks S_i specified in bid b_i. This formulation will produce maximum revenue in a standard (forward) auction, but it may leave items unallocated. This is known as the *free disposal assumption*, which is valid if none of the items has residual value to the seller. If this is not the case, then it is necessary for the seller to submit its own bids to represent its residual values. These bids are known as the dummy bids.

The RoboCup Rescue Winner Determination process differs from the classic problem described here in 2 major ways:

1. The RoboCup Rescue auction is a reverse auction, and we are interested in minimizing cost rather than maximizing revenue.
2. In our approach the free disposal assumption applies. The main goal is to find the optimal solution and we do not care if all tasks are covered by the bids sent by the agents. To deal with tasks unallocated, we use a adjustment phase in our combinatorial auction algorithm (see section 4.5.1).

In the next sections, we will present the RoboCup Rescue formulation and why combinatorial auctions works better than single auctions.

5.3.2 RoboCup Rescue combinatorial auction formulation

We consider a extension of the basic formulation described above, as follows:

$$Minimize \sum_{i=1}^{n} c_i x_i \tag{5.15}$$

Subject to:

- Bid selection - each bid is either selected or not selected.

$$x_i \in \{0,1\},\ i = \{1...n\} \tag{5.16}$$

- Coverage - each task s_j may be or not included into the bids.

$$\sum_{i:s_j \in S_i} x_i \leq 1, \text{ for each} s_j \in S_r \tag{5.17}$$

5.3.3 Single versus combinatorial auctions

Combinatorial auctions attempt to remedy the disadvantages of single-item auctions by allowing bidders to bid on bundles of items. If a bidder wins a bundle, he wins all the items in that bundle and hence is able to incorporate his synergies into his bids. According to Sandholm [90], if there is complementarity or substitutability between the goods, a combinatorial reverse auction can be beneficial. In this sense, the agent' cost to carry tasks out separately may be different regards to carried them out in a sequential fashion.

For example, if the tasks are auctioned off in a single-item auction, then agent teams AT1 and AT2 first move to targets T3 and T4, respectively, and then they move to targets T1 and T2, respectively, for a total travel distance of 29 units (see Figure 5.12).

In contrast, if the tasks are auctioned off in a combinatorial auction, then agent team AT2 wins bundle T1, T2 and thus first moves to target T2 and then to target T1, and agent team AT1 wins bundle T3, T4 and thus first moves to target T3 and then to target T4, for a total travel distance of 13 units (see Figure 5.13).

Such as it is demonstrated in the previous example, by using combinatorial auctions, agents can exploit synergies to allocate tasks in a more effective way by diminishing traveling costs. In the rescue domain, the problem of assigning rescue agents to rescue tasks can be formulated as a reverse combinatorial auction problem. The RoboCup Rescue implementation is shown in the next section.

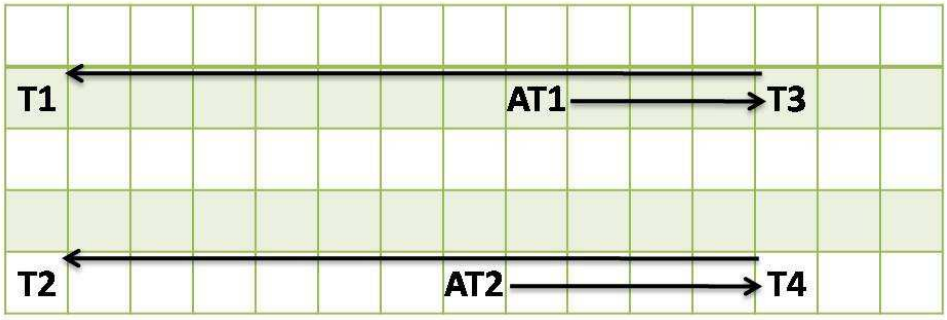

Figure 5.12: Trajectories with Single-Item Auctions.

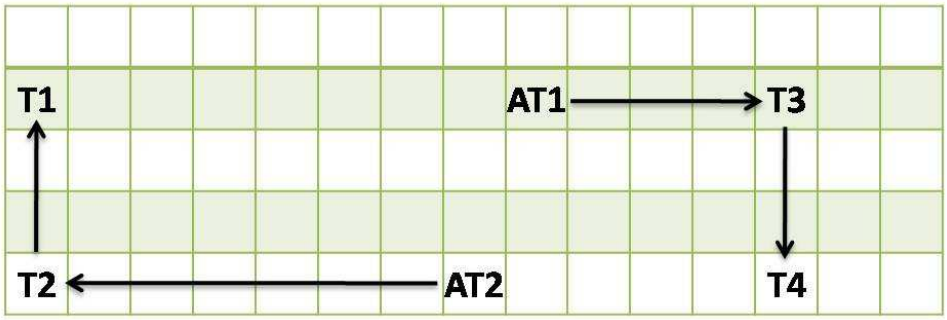

Figure 5.13: Trajectories with Combinatorial Auctions.

5.3.4 The RoboCup Rescue combinatorial auction process

In our RoboCup Rescue auction implementation, the fire station (central agent) takes on the role of auctioneer and the fire brigades (agent teams) take on the role of bidders. The task allocation process takes four cycles of simulation as follows (See Figure 5.14 is a timeline of the process):

1. First cycle: The agent teams send the local tasks that they can to detect in his surrounding.
2. Second cycle: The central agents receive the tasks from their agent teams and comeback the total list of the task to each agent team.
3. Third cycle: With the information about all the tasks, the agent teams configure and send bids corresponding to combinations or packages of tasks to the central agents. These bids express preferences of the agents by task packages.
4. Fourth cycle: central agents determine the winners, using the winner determination algorithm and send the result to the rescue agents.

Finally, the agent teams with winner bids perform their corresponding tasks. Then, the buyers are the central agents, the sellers are the rescue agents, who are submitting asks for rescue tasks.

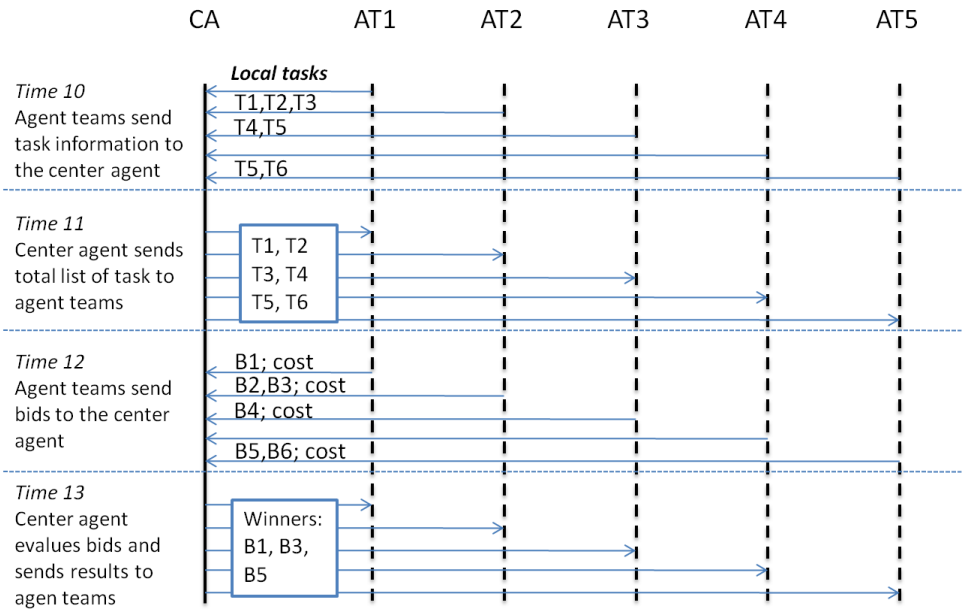

Figure 5.14: Timeline of the task allocation process for fire brigades and fire station agents.

The implementation of the combinatorial auction algorithm is presented in Figure 5.15 and 5.16. These both algorithms are executed each time step the center agent or the rescue agent receives one message. Regarding the part of the center agent (see Figure 5.15), it receives ”hear” two kinds of messages: Firstly, a ”MSG_TYPE_TASKS” which contains information about the tasks which have been informed by the rescue agents; and secondly, a ”MSG_TYPE_BID” which contains information about the bids submitted by the agents. In the implementation example on the Figures, the sent tasks are related to fires to extinguish. The bids are composed by a set of fires and the estimated cost to extinguish these fires' sets by the agent. Once a message of the type ”MSG_TYPE_TASKS” is received for the center agent, it gathers the fires informed and when it has received all the messages from the agents, the center *tells* the total list of fires to all the fire brigades. On the other hand, each time a message of the type ”MSG_TYPE_BID” is received by the center, it confirm all the agents have sent their bids and then it runs the winner determination algorithm. Then, the center *tells* the winners from of the auction result to the rescue agents.

In the part of the rescue agent (see Figure 5.16), the agent receives as well two kinds of messages: Firstly, a ”MSG_TYPE_WINNERS” which contains the information about the winners of the auction process. If the agent is a winner, it adds the fires allocated to the current targets list. Secondly, the rescue agent can receive a ”MSG_TYPE_TASKS” which contains the total lists sent from the center agent in the algorithm on Figure 5.15. With this information the agent configures the bids and then, *tells* them to the center agent.

Next, we will present the details of the implementation of this process. We will tackle issues such as: (1) The communication flow needed. (2) the bidding strategy used by the rescue agents. And (3) The solution of the winner determination problem in combinatorial auctions.

In Central Agent Part:

```
Switch (messages.getType(msg)) {

    Case Messages.MSG_TYPE_TASKS
        Read Message
        Create array firesToExtinguish<fires>
        If(all the agents have sent the fires' information)
            Tell (MSG_TYPE_TASKS, total list of firesToExtinghish).

    Case Messages.MSG_TYPE_BID
        Read Message
        Get bid
        Get Cost
        If all the bids from the agents have been received
          {
            Run the WINNER DETERMINATION ALGORITHM
            Tell (MSG_TYPE_WINNERS, winner agents.)
          }

}
```

Figure 5.15: Combinatorial auction algorithm in the center agent part. Communication and messages exchanged for the task allocation process.

In Rescue Agent part:

```
Switch (messages.getType(msg))
 {
      Case  messages.MSG_TYPE_WINNERS
            If I'm the winner
               Targets.add(allocated fires)

      Case messages.MSG_TYPE_TASKS
            Read Message
            Get Array of firesToExtinguish
            Create Bids
            Tell(MSG_TYPE_BID, bids)
 }
```

Figure 5.16: Combinatorial auction algorithm in the rescue agent part. Communication and messages exchanged for the task allocation process.

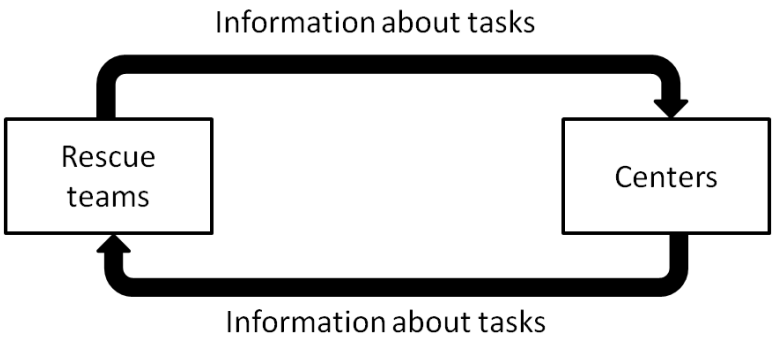

Figure 5.17: Messages flow about tasks among rescue teams and central agents.

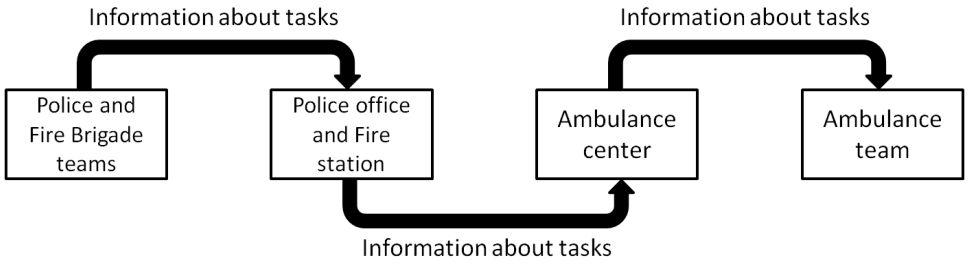

Figure 5.18: Messages flow about unblocking road tasks from police forces and fire brigades to ambulance teams.

5.3.5 The RoboCup Rescue communication flow

In order to implement the CATA algorithm we need to use a communication flow for that agents can know the most complete set of tasks into the environment and they can make the bid configuration. The communication flow presented in Figure 5.17 supports the communication and messages transference between rescue agents and central agents. Every simulation time the central agents are gathering the tasks from the rescue agents. That way, central agents have knowledge about the entire list of tasks inside the scenario, related to their correspondently rescue team.

On the other hand, in the development of their tasks, agents can find new tasks. For example, fire brigades and ambulance team agents find roads that they need be cleared in order to get to the place at which to accomplish their allocated tasks; at the same way, fire brigades and police forces find new injured victims, then, new tasks of agents are also provided by the other agents by means of the centrals correspondently (see Figure 5.18).

5.3.6 Bidding strategy

Other issue is how the agent teams compute their bids, i.e., on which bundles they bid and how they set the bid prices for these bundles. We do not limit the number of bids

that can be submitted by an agent; however, some issues must be taken into account. For instance, if the rescue teams bid on each and every possible combination of tasks then it is guaranteed that the solution obtained by winner determination is the optimal solution to the task allocation problem. However, this would lead to an exponential number of bids received by the central agent (such as it is exposed at first time in [85] in its studies on combinatorial auction algorithms). Hence it was necessary to decide on what tasks a bidder should bid on. Our implementation of combinatorial auctions have been done for the three kind of rescue agents (fire brigades, police forces and ambulance teams). In this sense, some bidding strategies have been implemented. Firstly, the fire brigade and police force share the same strategy which is based on distance among agent and tasks. Secondly, other bidding strategy has been designed for the ambulance team agent which is based on the death time of the victims. Finally, fire brigades use properties of fires to create their bids. In this sense, the bidding strategy of our agents consists on four mainly aspects:

1. Task combination size into the bundle (all the kind of agents).
2. Distance among agents and tasks (police force and fire brigade agents).
3. Death time of the victims (ambulance team agents).
4. Urgency of fires (fire brigade agents).

TASK COMBINATION SIZE

The first one issue "*Task combination size*" is used for all three kind of agents. In this sense, we have used a heuristic based approach where the bidders bid on combinations of $size < q$. Therefore, the number of bids received by the central agent is at most:

$$Number\,of\,bids = \left(\sum_{i=1}^{i=q} round(m\,Mod\,i) \right) * n \tag{5.18}$$

where m is the number of victims or targets and n the number of rescue teams or bidders. For example, for a set of four tasks $T = t_1, t_2, t_3, t_4$ and value q=2, an agent may set bids: $B_1 = t_1, t_2$ and $B_2 = t_3, t_4$. In next sections, we will explain the bidding strategies in more detail.

DISTANCE AMONG AGENTS AND TASKS

Our police force and fire brigades generate bids corresponding to combinations of tasks to be performed in sequential order. Bids are composed by different combination of tasks and the estimated cost that the agent has to assume by performing sequentially the tasks in this combination. Formally,

$$b_i = [L_i, c_i] \tag{5.19}$$

In order to select tasks for bids, agents take into account that each task should be located in a distance *d* inside a given horizon measure. Only the tasks which are in this horizon are accepted to conform the bids. In our implementation, each agent team calculates the distance from himself to each target and selects only these that are not more than a *d= 100* meters from his localization.

Then, the cost of the bid is related to the distance from the agent a_i to the itinerary that the selected tasks establish. Given, the selected tasks $L_i = [t_1^i, \ldots, t_n^i]$; the following distances are computed $d_i = [d_1^i, \ldots, d_n^i]$, where d_j^i is the distance between the place of task t_{j-1}^i and t_j^i. The first distance d_1^i corresponds to the distance from the agent to the first task t_1^i. Finally, the cost is computed as the sum of distances.

$$\sum_{j=1}^{n} d_j^i \tag{5.20}$$

All bids received by the central agent are processed using the winner determination algorithms which is explained latter. In this sense, the cost of carry out the rescue tasks is being minimized as result of solving the combinatorial auction. To illustrate with an example the bid generation, let's assume that at time 11, fire brigade ID 268415620 has knowledge about four tasks to develop, for instance, buildings on fire which have to be extinguished. The list of IDs of these buildings is as follows: [167682284, 249558638, 214940734, 260987697].

Then, the fire brigade calculates the distance from him self to each building and selects only these that are not more than 100 meters from his localization. So, the selected tasks are: building ID 214940734 with a distance of 40.566 from the agent and building ID 167682284 with a distance of 83.890 meters from the agent.

Now, we calculate the cost of the bid as distance from the agent to task 214940734 plus distance from task 214940734 to task 167682284 which is: 40.566 + 60.433 = 100.999. Then, a bid submitted by the agent is the following:

BidAgent ID 268415620 = [(214940734, 167682284), 100.999].

Figure 5.19 shows fire brigades making bids to the fire station on different combination of fires that they want to extinguish based on their proximity to these fires. Thus, as it was stated before, an agent team needs to be able to determine distances quickly, for example, from its current location to a fire or blocked road or from one fire to the next, even though it has only incomplete information about the environment. Our agent teams use an optimistic distance estimate, namely the shortest distance. They compute these distances with a Breadth First Search BFS [123]. That way, the agent teams compute the cost of the bid

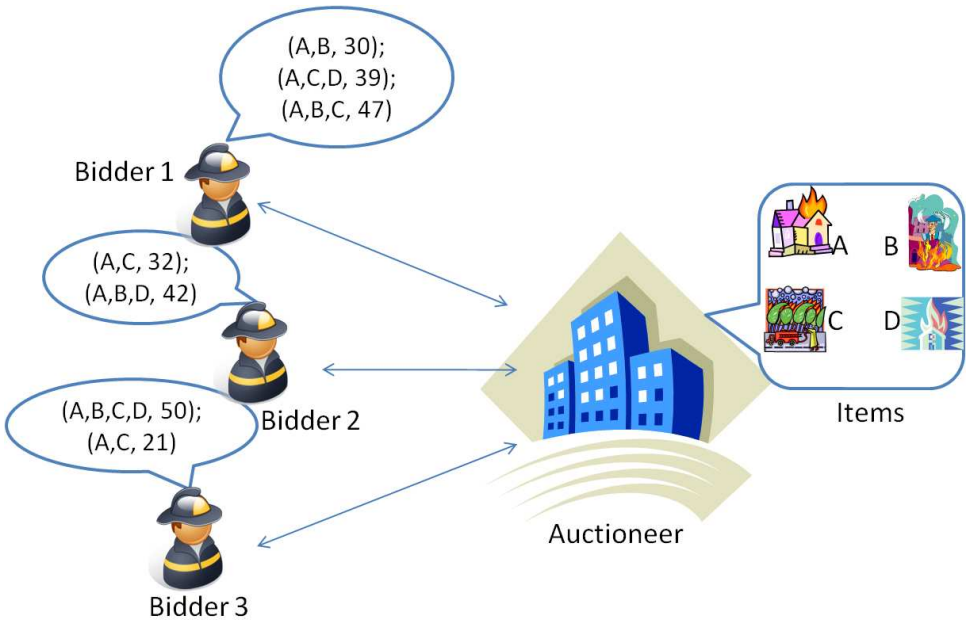

Figure 5.19: Auctioning of fires by fire brigades.

as the optimistic travel cost for visiting all targets contained in the bundle (combination of tasks) from its current location.

DEATH TIME OF THE VICTIMS

Ambulance teams configures bids taking into account the task completion feasibility to the task set. In our implementation, agents configure bids according to the scheduling study presented in Section 5.2.2. In this sense, agents sort all the tasks in the non-decreasing order of their death times and include into the bid the victims which accomplish with: *pct(t)<1* and pr(t)$>\ \theta$ according to Equations 5.2 and 5.4. In this sense, agent teams take into account the agent's capability and victims' priority for the bid configuration.

For example, in Figure 5.20 from the example in Section 5.2.2 there are three victims v_1, v_2, v_3, each victim with a death time of 30, 50 and 90 respectively and each one has a rescue time of 20, 30 and 50 respectively. There is an infeasibility between victim v_2 and v_3 to belong to the agent's bid for this reason, the bid B of the agent a will be conformed by victims v_1 and v_2. That is: $B_a = \{v_1, v_2\}$.

One example of the bid configuration of a set of five victims sorted by death times $V = \{v_1, v_2, v_3, v_4, v_5\}$ into the rescue scenario for 5 agents and values of q=$\{1 \ldots 5\}$is presented in Table 5.1. In this example, the victims are first sorted according to death times and then, the agents configure bids according to the size *q*, such as it was explained before in the "Task combination size" subsection. In this particular example, the number of bids sent by the agents is at most 65 bids.

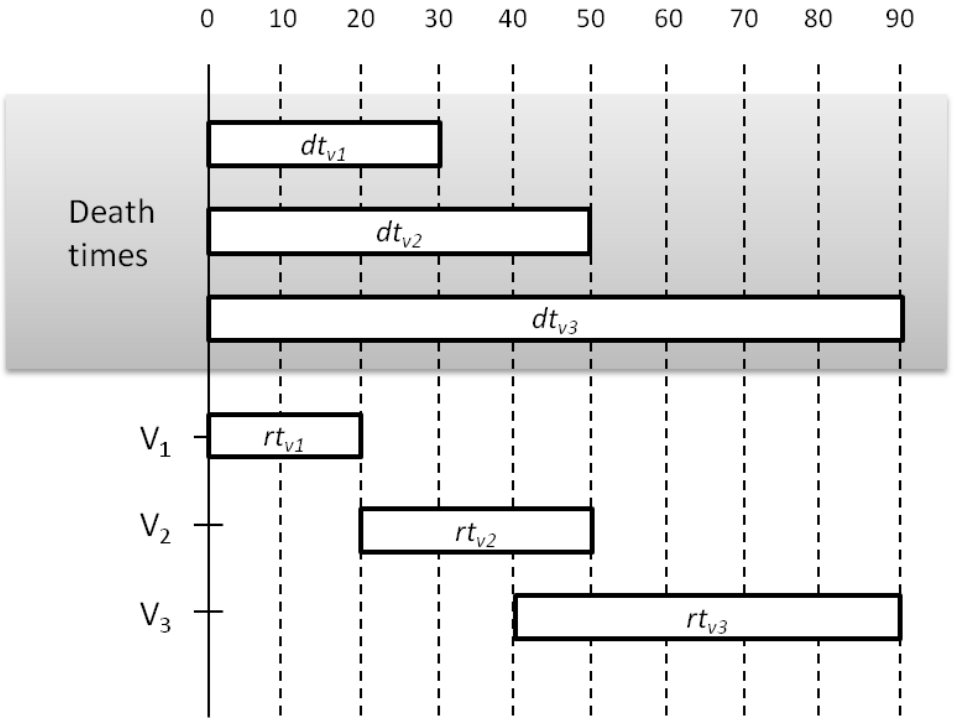

Figure 5.20: Bid configuration infeasibility in ambulance teams.

Table 5.1: Bid configuration for five victims into the rescue scenario

Number of victims	Number of agents	q	Bids from agents
5	5	1	$B_1\{v_1\}B_2\{v_2\}B_3\{v_3\}B_4\{v_4\}B_5\{v_5\}$
		2	$B_6\{v_1, v_2\}B_7\{v_3, v_4\}B_8\{v_5\}$
		3	$B_9\{v_1, v_2, v_3\}B_{10}\{v_4, v_5\}$
		4	$B_{11}\{v_1, v_2, v_3, v_4\}B_{12}\{v_5\}$
		5	$B_{13}\{v_1, v_2, v_3, v_4, v_5\}$

Table 5.2: Urgency values of fires according to their fieriness

$Fieriness(f_i)$	U_f
1	10
2	5
3	2
0,4,5,6,7,8	0

URGENCY OF THE FIRES

To bid for buildings to extinguish, a fire brigade agent builds a list of all the buildings on fire in the agent's area. This list is sorted according to a utility function that gives an idea about the usefulness of extinguishing a fire. The utility function $U(f_i)$ gives a value to a fire f_i based on the buildings and the civilians in danger if f_i propagates to buildings close by. The utility function considers all buildings in danger by the given fire f_i. Buildings in danger are near buildings which are not on fire, but that may catch fire if fire f_i is not extinguished. Based in this utility funcion, the fire brigade sort the fires into their bids. Next, the utility function is explained.

UTILITY FUNCTION $U(f_i)$

The parameters used in order to compute utility of fires are:

1. Fieriness's value of fiery buildings: it is how much the building is burned.
2. Neighbours of the fiery buildings: number of neighbours buildings of the fiery building.

1. *Urgency by fieriness* U_f

   In the RoboCup Rescue simulator, the effort needed to extinguish a fiery building depends on a number of parameters and there is not exact formula for an agent to determine it. However, there are some heuristic rules that may help to find a system to approximate it. For instance, the more the value of "fieriness" for a building, the more effort needed to extinguish it. For example, a building with fieriness=1 can be extinguished with a little effort while a building with fieriness= 3 is very hard to extinguish. A building with fieriness=1 which was put on fire 1 or 2 cycles ago will be extinguished a lot easier than a building with fieriness=1 which was put on fire 6 or 7 cycles ago. The bigger the building, the more effort needed to extinguish it and so on. Then, regards to fires, it should be extinguished in early stages, otherwise it becomes much harder. The urgency values based of the fieriness are shown on table 5.2.

2. *Urgency by neighbourhood* U_n

   The fires which have the highest number of unburned neighbour buildings, it means neighbour buildings with fieriness equal to 0 (zero), are selected first.

Then, the two urgency funcions are described by both parameters:

$$U_f(fi) = urgencyByFieriness(f_i) \tag{5.21}$$

$$U_n(fi) = urburnedNeighbours(f_i) \tag{5.22}$$

More formally, the utility function $U(f_i)$ is calculated using the following equation:

$$U(fi) = U_f(fi) * K_1 + U_n(fi) * K_2 \tag{5.23}$$

Other factors have been studied as for instance, the number of victims inside buildings. However, in the simulator, civilians will die quite fast once they are burnt, even for a few cycles, so by the time fieriness=2 they will be dead. For this reason, it is clear that buildings with lowest fieriness value should have highest urgency value to be extinguished as is shown on Table 5.2.

In the next section, the solution of the RoboCup Rescue winner determination problem is presented.

5.3.7 Solving the RoboCup Rescue Winner Determination Problem (RRWDP)

We now focus on the RoboCup Rescue winner-determination problem, originally introduced in Section 5.3.1. Such we have stated before, in the combinatorial auction and optimization research several algorithms have been developed to solve the winner determination problem in combinatorial auctions. We have applied two algorithms to solve this problem. Firstly, a depth first search DFS algorithm has been used. DFS starts at the root and explores as far as possible along each branch of the tree before backtracking. A remarkable issue is the free disposal feature that we have incorporated into the implementation of this algorithm. And the second algorithm used is an optimal tree search algorithm from Collins [20]. The former (DFS) is a classical algorithm (see, for example, [83]). For this reason, we will describe more deeply the latter Collins' algorithm in next section.

5.3.8 Optimal tree search formulation

Collins presented one algorithm to solving the standard combinatorial auction winner determination problem [20]. The Collins' Algorithm is an iterative-deepening A* formulation based on Sandholm's algorithm [86]. Although part of the implementation of this algorithm, such as the precedence constraints issues, cannot be easily applied to the RoboCup

Rescue problem, we have used the Collins' algorithm to solve the winner determination problem. This algorithm presents some advantages regards to the original sandholm's algorithm such as it has improved upon it by specifying a means to minimize the mean branching factor in the generated search tree.

In general, tree search methods are useful when the problem at hand can be characterized by a solution path in a tree that starts at an initial node (root) and progresses through a series of expansions to a final node that meets the solution criteria. Each expansion generates successors (children) of some existing node, expansions continuing until a final node is found. The questions of which node is chosen for expansion, and how the search tree is represented, lead to many different search methods. In the A* method, the node chosen for expansion is the one with the "best" evaluation (lowest for a minimization problem, highest for a maximization problem), and the search tree is typically kept in memory in the form of a sorted queue. A* uses an evaluation function:

$$f(N) = g(N) + h(N) \tag{5.24}$$

for a node N, where $g(N)$ is the cost of the path from initial node N_0 to node N, and $h(N)$ is an estimate of the remaining cost to a solution node. If $h(N)$ is a strict lower bound on the remaining cost (upper bound for a maximization problem), we call it an admissible heuristic and A* is complete and optimal; that is, it is guaranteed to find a solution with the lowest evaluation, if any solutions exist, and it is guaranteed to terminate eventually if no solutions exist [20].

BIDTREE FRAMEWORK

For a basic introduction to the A* algorithm, see [83], or another textbook on Artificial Intelligence. The Collins' algorithm depends on the bidtree structure which must be prepared before the search can run. The bidtree which was introduced by Sandholm is a binary tree that allows lookup of bids based on item content. The bidtree is used to determine the order in which bids are considered during the search, and to ensure that each bid combination is tested at most once.

A bidtree is a binary tree whose principal purpose is to support based-content look up of bids. The depth tree is $m + 1$, where m is the number of tasks of the problem. The bids appear in the leaf nodes. Each level of the bidtree represents one task and the in and out branches show if the task belong or not to the bid.

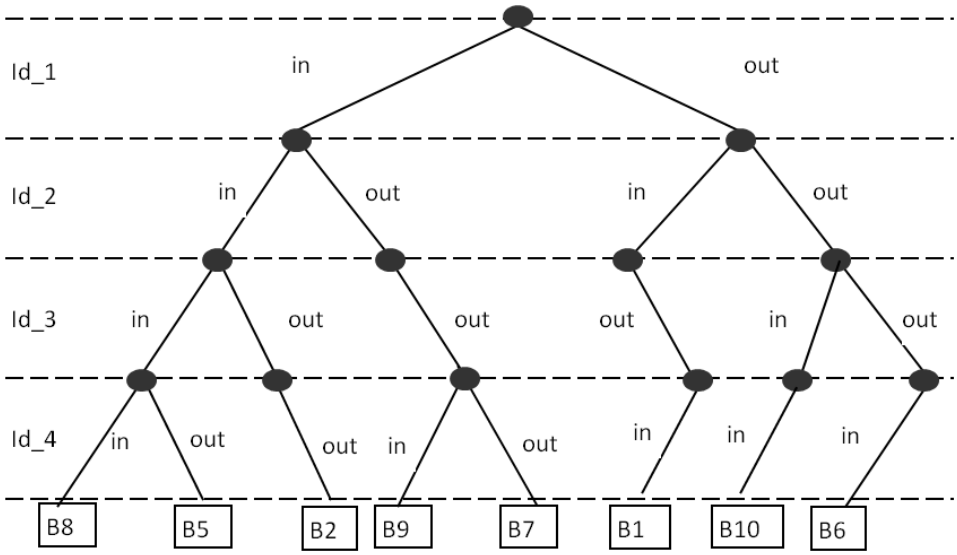

Figure 5.21: Bidtree for the example of victims rescue tasks.

BIDTREE SEARCH EXAMPLE FOR ROBOCUP RESCUE DOMAIN

We can illustrate the concept by means of an example in the rescue domain. Suppose, we have some victims in some simulation time with the following identifications: Id_1, Id_2, Id_3, Id_4 and assume that our central of ambulances has received the following bids: B1(s1, 30), B2(s2, 27), B3(s3, 17), B4(s4, 32), B5(s5, 39), B6(s6, 14), B7(s7, 12), B8(s8, 65), B9(s9, 24), B10(s10, 33) with the correspondent set of tasks as follows:

s1= (s1=[Id_2, Id_4], s2= [Id_1, Id_2], s3=[Id_1], s4=[Id_2, Id_4], s5=[Id_1, Id_2, Id_3], s6=[Id_4], s7=[Id_1], s8=[Id_1, Id_2, Id_3, Id_4], s9=[id_1, id_4], s10=[id_3, id_4]).

Taking into account that bid B3 has the same tasks set as bid B7 and cost of B3 is highest than cost of B7, B3 can be erased from the bids set as this never will generate an optimal solution. The same is the case of bid B4 regarding bid B1. So, the bidtree for this problem is shown in Figure 5.21.

BIDTREE ORDERING

Taking into account how many times the tasks appear in the bids (bid count), the tasks can be ordered in both: with increasing bid count and with decreasing bid count. For instance, regarding the rescue example in the previous section, the bid count for each task is the following: task id_1 appear in 5 bids so the bid count for this task is 5, task id_2 bid count is 4, task id_3 bid count is 3, and task id_4 bid count is 5. Then, according to the increasing bid count, the order of tasks for the bidtree is: id_1, id_4, id_2, id_3. The bidtree with this sort is shown in Figure 5.22.

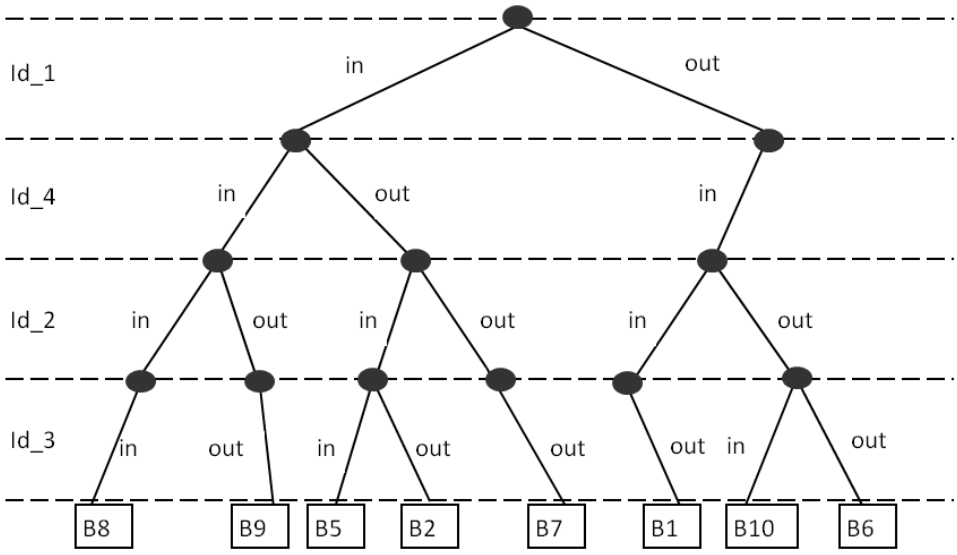

Figure 5.22: Bidtree with increasing bid count.

Conversely, taking into account a decreasing bid count the order that we get is: id_3, id_2, id_1, id_4. The bidtree for this case is presented in Figure 5.23.

GENERATING THE SEARCH TREE

In this section the solution of the RoboCup Rescue task allocation example previously described is presented. To see a more complete explanation of the collins' algorithm refers to [20].

The bidtree is used to generate the search tree, using the A* algorithm for solving the winner determination problem. Let $B = \{B_1, B_2, \ldots B_n\}$ be the set of bids submitted to the winner determination process. Let $Bc = \{Bc_1, Bc_2, \ldots Bc_m\}$ be the subset of bids of B which are still available to be appended to the search path, that is bids that do not include any tasks that have already been allocated. Let $Bl = \{Bl_1, Bl_2, \ldots Bl_n\}$ be the subset of bids in the leave nodes of the left subtree of the bidtree (in branch). For instance, these are the bids in shaded square in Figure 5.23. Let $Br = \{Br_1, Br_2, \ldots Br_n\}$ be the subset of bids in the leave nodes of the right subtree of the bidtree (out branch). For example, these are the bids in not shaded square in Figure 5.23. Let $alloc$ be a subset of B which is the best allocation found so far (this with the minimum cost), and $C(alloc_{best})$ be the total cost of $alloc$. Let $Bo = Bo_1, Bo_2, \ldots Bo_n$ be the set resulting of $Bc \cap Br$. Analogously, let $Bq(Bi) = Bq_1, Bq_2, \ldots Bq_n$ be a set which stores the bids which conflict with bid B_i. In addition, for one partial allocation $alloc = B_1 \ldots B_n$, it is possible to define:

$$Bq(alloc) = \bigcup B_i \tag{5.25}$$

The steps of the algorithm are the following:

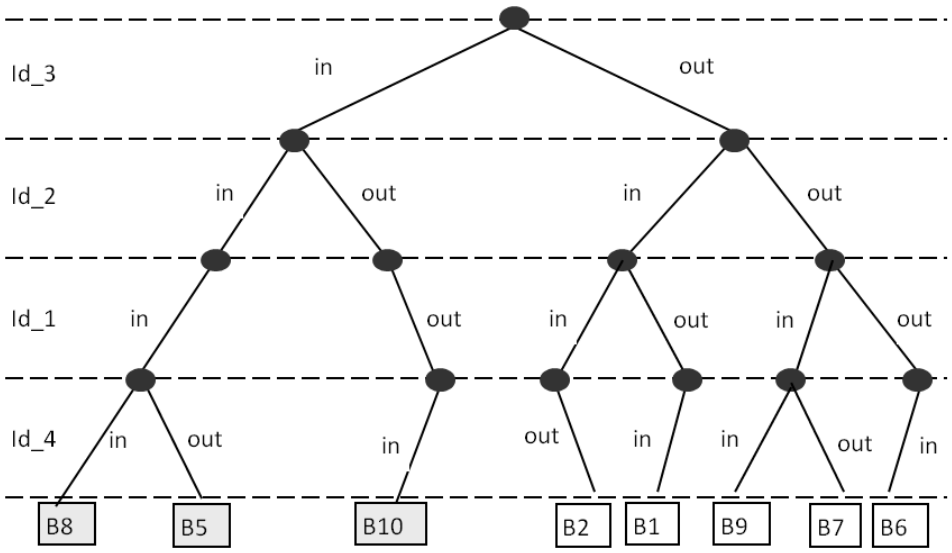

Figure 5.23: Bidtree with decreasing bid count.

Set C($alloc_{best}$) = {}

For each b from Bl

1. b=one bid from set Bl.
2. *alloc*={}
3. Bc(*alloc*)= B – Bq(*alloc*) – (*alloc*)
4. Bo=Bc(*alloc*)∩Br
5. if Bo != {} then
   - b_j= one bid from Bo which have the less cost, if any.
   - If b_j then
     - *alloc*=*alloc* ∪ bj
6. if (|tasks(*alloc*)| = total) then
   - if C(*alloc*) < C($alloc_{best}$) then
   - C($alloc_{best}$) = C(*alloc*)
   - Return to point 1, next b.
   - else
   - Backtracking
7. Return to point 3.

Figure 5.24: Algorithm to winner determination in combinatorial auctions.

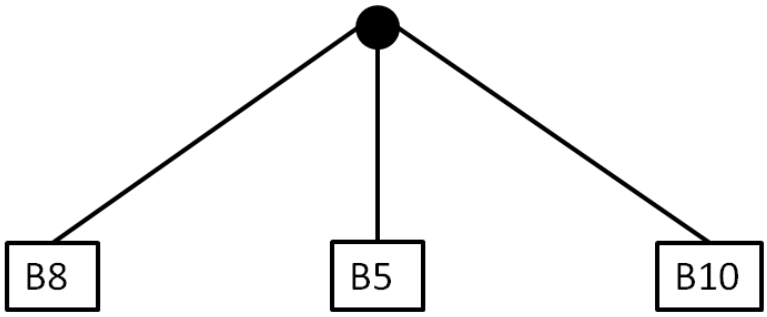

Figure 5.25: First level in the search tree.

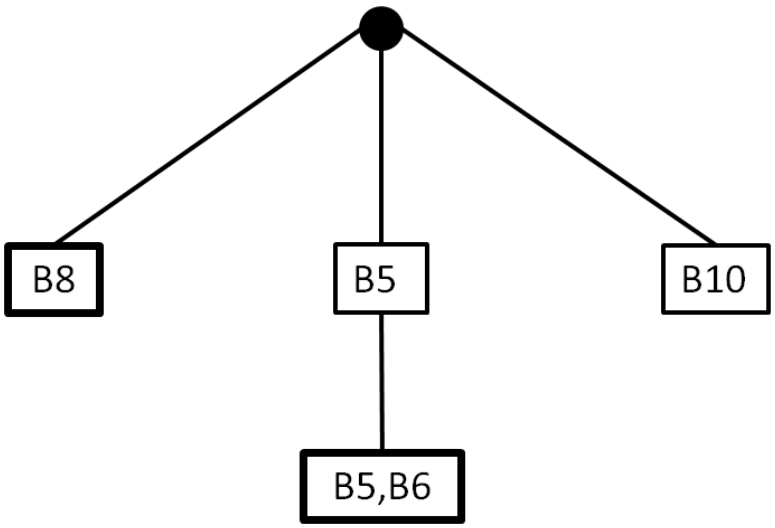

Figure 5.26: Two first solutions provided for the algorithm.

Regarding Figure 5.23 and our example regarding victims rescue tasks, the Bl set is $\{B_8, B_5, B_{10}\}$, the Br set is $\{B_2, B_1, B_9, B_7, B_6\}$. The first level in the search tree is shown Figure 5.25:

Since $\{B_5, B_6\}$ is a combination that contains all the tasks, is another feasible solution as it is shown in Figure 5.26.

Finally, the total search space generated is presented in Figure 5.27:

The best allocation found is $\{B5, B6\}$. And the cost is 55. The search space generated using the increasing bid count bidtree (see figure 5.22) is presented in Figure 5.28:

Note that the search space using the increasing bid count bidtree (Figure 5.28) generates more nodes although the solution is the same. Regarding to our example in Section 5.3.8, the ambulance agents who have submitted B_5 and B_6 bids are the winners for developing the task sets in their correspondent bids.

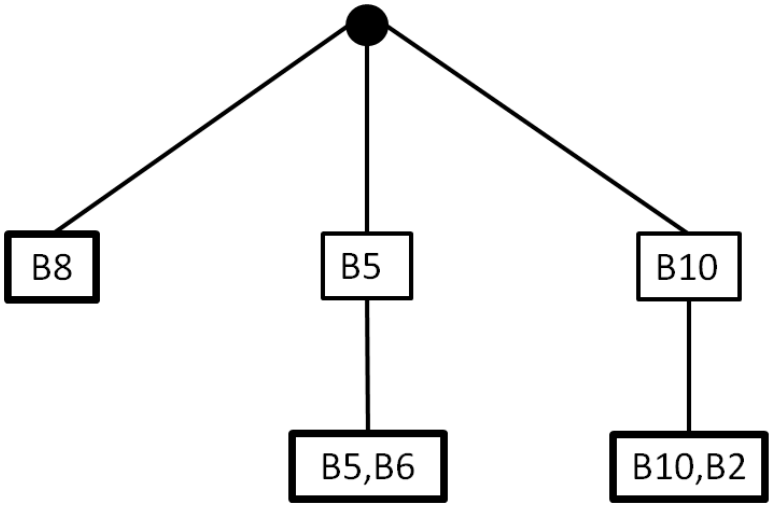

Figure 5.27: Total search space generated and solutions provided in shared square.

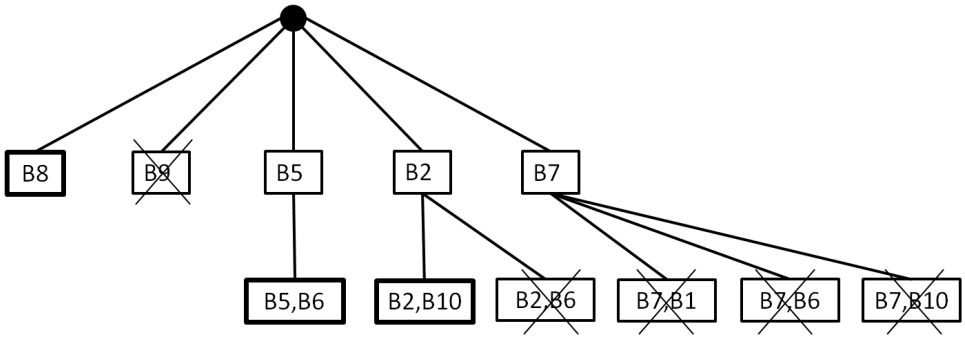

Figure 5.28: Search space generated for bidtree with increasing bid count.

5.3.9 Other implementation issues. Dummy bids

In collins' algorithm the coverage property is different to our approach. That is,

Check Coverage in collins' algorithm. Each task s_j must be included exactly once. That is,

$$\sum_{i:s_j \in S_i} x_i = 1 \quad for \quad each \quad s_j \in S_r \tag{5.26}$$

Note that under the free disposal assumption, each task would be included at most once, rather than exactly once.

In the Rescue system, the free disposal assumption applies. After the first auction when some agents are busy (because tasks have been allocated to them) the bids sent by free agents could not contain all the tasks. Therefore, the Check Coverage constraint of Collins' algorithm is not fulfilled and the algorithm does not work.

In order to solve this problem, we issue dummy bids [86] one for each task. Each dummy bid must have a much higher price than any "real" bid. Dummy bids could be part of the solution but just if some rescue agent does not send any real bid for some task. In this sense, when dummy bids are awarded, the tasks corresponding to these dummy bids remain unassigned for the next round in the allocation process.

5.3.10 Rescheduling strategy

As stated in many previous places, the RoboCup Rescue environment is uncertain and dynamic and consequently agents have to be able the reschedule when changes happen [101]. One method to do that would be to reschedule each time that something changes in the environment. However, this method could make agents to change from one task to another before completing any, if each time something changes, the scheduler modifies the first task in the schedule.

To circumvent this, we have used a strategy that follows the following principle: if a task is being executed, it will be executed until the end. If during the execution of this task, a new information is received, it is stored and it is only when the task is completed that the agents take the new information into consideration to reschedule and choose the next task to accomplish.

Other issue of task rescheduling is related to allow agents face problems which avoid them to follow with the scheduling's execution. We have tackled this issue and we have called it "Replanning". In next section, the replanning strategy implementation is presented. This replanning strategy has been introduced in Section 4.5.4 of Chapter 4.

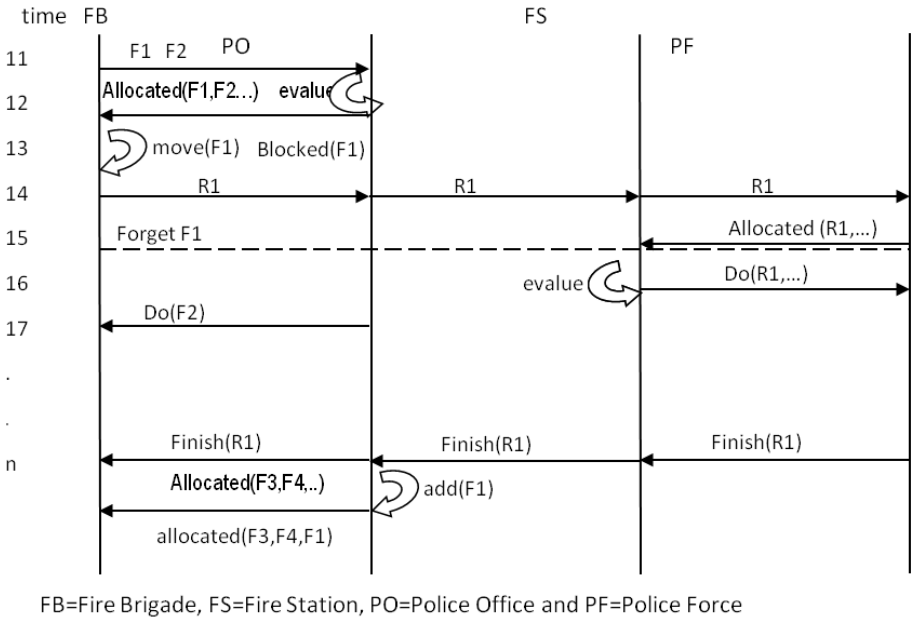

Figure 5.29: Mechanism for replanning of tasks.

5.4 Replanning implementation

Dynamic task allocation involves both the initial allocation of tasks within a team and re-allocation in response to anticipated problems in task execution [64]. RoboCup Rescue is a real life environment in which a number of agent teams perform tasks collaboratively in order to achieve a maximum number of rescued victims and to decrease the damages into the scenario.

In Rescue environments, execution problems may happen to any plan due to the uncertainty of this kind of systems. For this reason, it is indispensable to implement replanning as a part of the planning system of the agents. For instance, suppose one rescue agent which has an allocated task finds a blocked road. This problem may avoid the agent can to reach until the task location. Is in this kind of situations where we have implemented our replanning strategy which has been introduced in Section 4.5.4.

In the Robocup Rescue scenario, once tasks are allocated to agents, they try to develop them. Such as it was explained before, there exist several factors that make the agents be unable to execute their tasks, for instance, obstructed roads. At this moment, a replanning method is needed.

For example, suppose that the fire brigade agent FB have the F1 task in its scheduling of allocated tasks (see Figure 5.29). However, when the agent tries to get until F1, it finds the R1 Road blocked. In this case the agent tries to look for another way to get until its goal. When it is not possible, the agent finds it-self blocked. In our replanning implementation, in order to rescheduling the tasks, the agent temporally ”forget” F1 keeping it in a list of

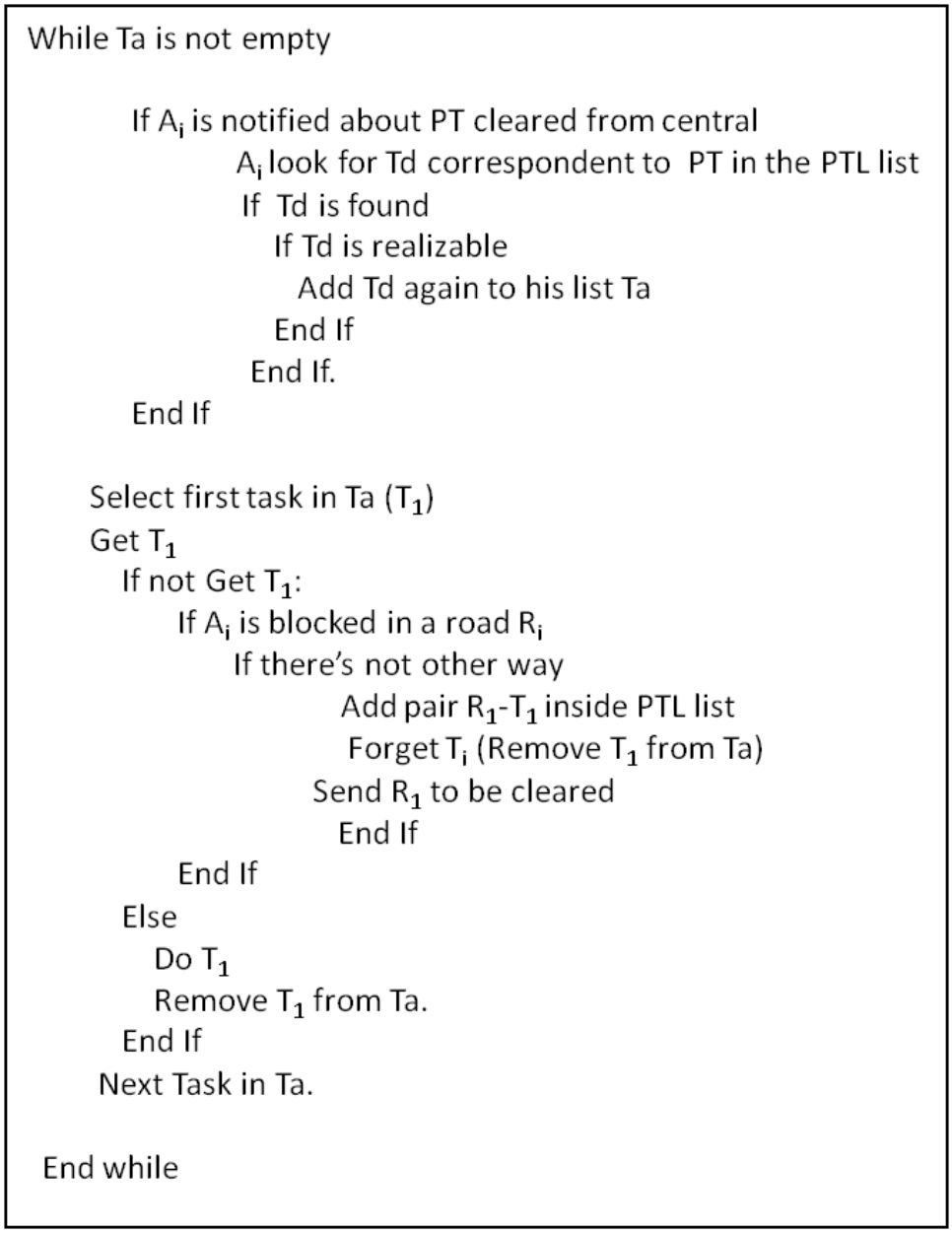

Figure 5.30: Algorithm for replanning of tasks.

delayed tasks. Next, it sends R1 to the police force as a task to be developed. Then, FB continues with the development of the next task in its schedule. In successive cycles, once R1 is cleared, the agent is informed about that, by the central agent. Then, it re-schedules F1, introducing F1 in its list of pending tasks. Previously, the agent verifies that F1 is realizable, for example, if F1 has grown so much that currently is in-extinguishable, it is no included in the tasks list again and it is forgotten for ever.

The replanning algorithm developed is presented in Figure 5.30. Agents are noted as A_i. Each agent has a list of tasks, $T_a = (T_1, T_2, T_n)$ pending to be performed. If the list is empty, the agent explores the environment looking for new tasks. In addition, each agent have a list $PTL = (PT_1 - Td_1, PT_2 - Td_2, PT_n - Td_n)$. This list maps precedence among pair of tasks $PTi - Td_i$, where PT is the task which precedes the delayed Td task. For example, PT represents the Road's Id to be cleared and Td a fire's Id which has been delayed in the process previously explained.

5.5 Experimentation and results

This section presents the experimentations that have been done to test our approach. In the first set of experiments, we present results showing the efficiency of our case-based approach to estimate the death time of a civilian. Then, we present results showing the efficiency of our SATA and CATA algorithm. Afterwards, we present results comparing the efficiencies of both centralized (CATA) and decentralized (SATA) execution of the algorithms. All tests have been made on the RoboCup Rescue simulator.

5.5.1 Death time prediction experiments

This first experiment is related with the accuracy of our death time prediction mechanism which has been explained in section 5.2.3. Figure 5.31 shows the system performance with the three different death time estimation measures (median, mean and minimum). Outputs of 15 simulations have been used for these estimations. The mean death time estimation obtains the best results as a higher percentage of victims were rescued.

Using the mean measure, we have compared the prediction accuracy of our case-based approach with an approach that only considers the current damage value of the civilian (hp/damage). With the latest approach, the estimated death time is simply calculated by dividing the health points (hp) value of the civilian by its damage value (damage). With the case-based approach, we have used past cases or experiences to predict the death time of each civilian. In this case, the estimated death time is the time step reached when the hp value reaches 0.

The results comparison of efficiency both approaches (case-based versus hp/damage) are presented in Figure 5.32. These results have been obtained during 200 simulations using three different maps from the scenario (Kobe, Foligno and Random maps). Those sce-

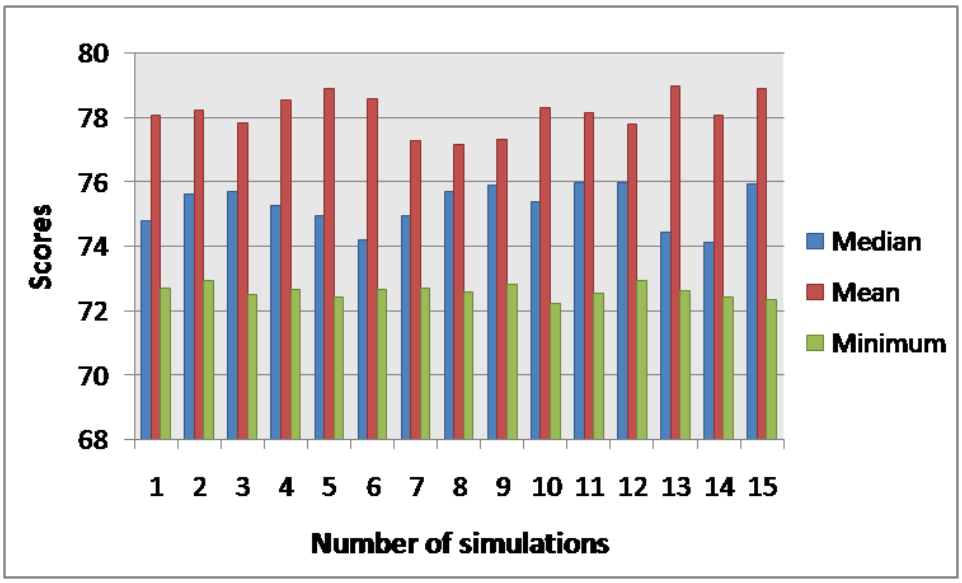

Figure 5.31: Simulation results for three different methods of victims' death time estimation: simulation results using mean, median, and minimum death time estimation.

narios were designed to gather the death-time estimations made by the ambulance teams. Therefore, we have simplified the simulations to remove everything that could interact with the ambulance team agents. To be more precise, we have removed the blocked roads as well as all the fires and the rescue actions performed by the ambulance teams.

Each time an agent team had to estimate a civilian death time, we have recorded the estimation from both mechanisms. In addition, we have compared these estimations with the real death time of the civilian. These real death times have been obtained from the misc-simulator logs of the RoboCup Rescue simulator. Then, the results are average differences between the estimated and the real value.

Each line on the graphic represents the average error of all the predictions made in a specific time step. For example, the first line represents the error average for all the predictions that have been made in time step 3. As we can see in Figure 5.32, the case-based approach makes much better predictions than the hp/damage approach. For example, at time 109, the average error in the predictions for the case-based approach is 0. This means that on average, the agents are predicting an accurate death time of a civilian. For the same predictions, at the same time, the hp/damage approach is estimating the death time of a civilian with a -241 percent of error. It means the real death time was 241 percent low than the prediction done by this method. The lines above and below the horizontal axis are represented the average error in each prediction for both mechanisms.

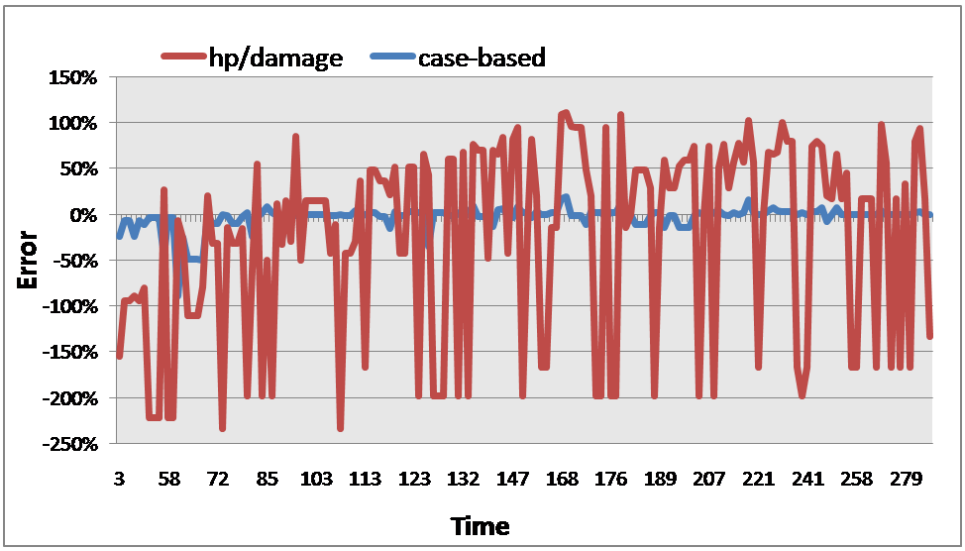

Figure 5.32: Prediction results of the Case based (CB) mechanism. This graphic presents the error made by the CB approach compared to the approach hp/damage, which consists in dividing the current hp value of a civilian by its damage value (damage). The error is the average difference between the estimation and the real value.

5.5.2 System's experiments with cooperation versus without cooperation among heterogeneous rescue agents

The objective of this experiment was to compare both, the performance of the system using cooperation among heterogeneous agents and without this kind of cooperation. This experiment has been done for the ambulance team agents operation. Such as it was stated before, due to the RoboCup Rescue scenario complexity, agents need cooperate among them. For instance, in the rescue simulation it is very difficult for the ambulances to find victims. They have to explore the surrounding in order to look for them. In this sense the cooperation among different kind of rescue agents (it is fire brigades, police forces and ambulance teams is very important). These kinds of agents are called heterogeneous because of the differences among the capabilities and design of each kind. For instance, the fire brigade can not rescue a victim or unblock a road and vice versa. In this sense, exploration of the buildings and roads on the map is very important for a successful rescue task. The civilians are injured and buried in their houses but some of them are vital whereas some are not. A good exploration should find all vital damaged civilians before they die. The only way is to explore as many buildings as possible in the short time. Since the exploration is a common task it is coordinated among all agents by visiting all the unvisited buildings each time step.

Then, to help ambulance teams to rescue victims, we have implemented a cooperation strategy among the different kind of agents. In this sense, fire brigade and police forces have to inform to ambulance teams about the victims found by them each time step. This victims' information is communicated by means of the centre agents and then it is informed

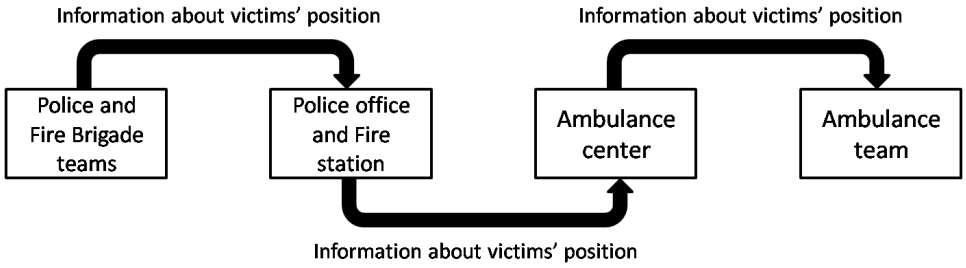

Figure 5.33: Messages flow about victims' position from police forces and fire brigades to ambulance teams.

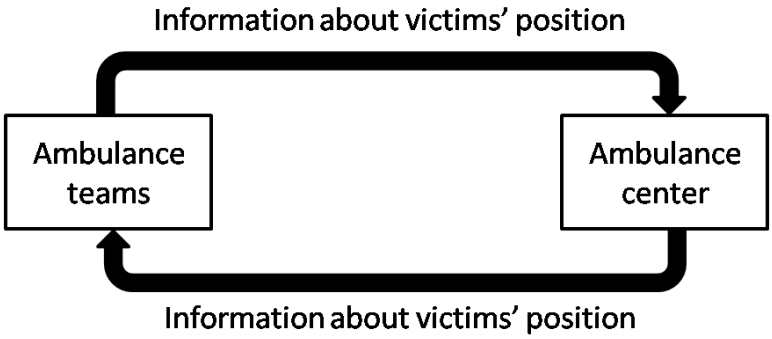

Figure 5.34: Messages flow about victims' position among ambulance teams.

to the ambulance teams, such as it is shown in the Figure 5.33. In addition, ambulance teams inform to the another ambulance teams about the victims' location (see Figure 5.34). With this strategy ambulance teams have a more complete knowledge about victims' position.

The communication flow in the figure has been explained before in Section 5.3.5. The results comparison of both system's performance (with cooperation versus without cooperation) are presented in Figure 5.35. In the simulator, the scores of each simulation are computed using the evaluation equation 2.2. shown in chapter 2. These results have been obtained during 200 simulations. This scenario was designed to prove the ambulance team performance by using this cooperation strategy. Therefore, we have simplified the simulations to remove everything that could interact with the ambulance team agents. It means we have removed the blocked roads as well as all the fires. In addition, our rescue agents try to explore only unvisited buildings each time step.

With this kind cooperation about victim's information, ambulance teams could improve its rescue operation around 21 percent. In addition, without cooperation the ambulance agents found the first victim after time step 30. On the contrary, with cooperation it was found at time step 5. In summary, the results on Figure 5.35 show the remarkable impact of the heterogeneous agents' co-operation to the rescue of victims.

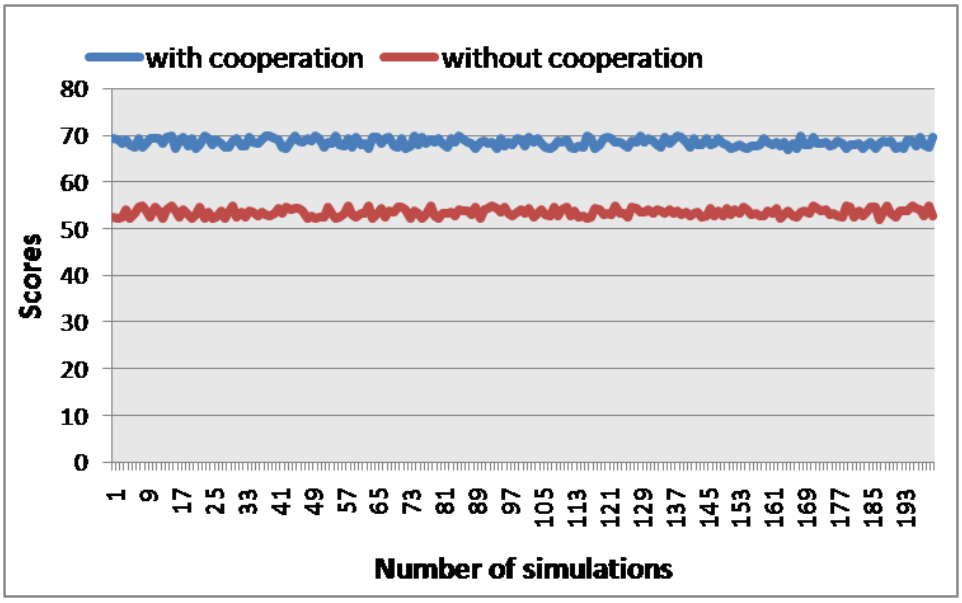

Figure 5.35: Comparison of system's performance using cooperation versus without cooperation among heterogeneous agents.

5.5.3 SATA EXPERIMENTS

In this section, we present how our SATA algorithm, previously presented, can be used in the RoboCup Rescue simulation to help the ambulance team agents to coordinate themselves on the highest priority victims to rescue. Such as it was stated before, in our SATA implementation, the tasks are the victims which are sequenced in a non-decreasing death time order. To do that, our ambulance teams search for the victims and predict their death time to establish a victim priority order in our scheduling algorithm and to coordinate themselves.

The SATA algorithm permits rescue agents scheduling the victims taking into account the priority of them and the possibility to rescue some victim. In our SATA implementation on the rescue operation, the ambulance teams can schedule the victims based on the death time and also their rescue times. Each time our ambulance agents found a victim they compute its death time in order to make a scheduling using Equation 5.4 and then, they determined if the victim could be rescued alive using Equation 5.2, before to introduce it into the scheduling.

The average results comparison of both system's performance, the performance of the rescue operation by using SATA versus the performance of rescue operation using the distance criterion, (It means, agents rescue the nearest victims first) are presented in Figure 5.36. These results have been obtained during 200 simulations. Again, the scenario was designed to prove the ambulance team performance by using the SATA algorithm. Then, the simulations have been simplified to remove everything that could interact with the ambulance team agents. In this sense, we have removed the blocked roads as well as all the

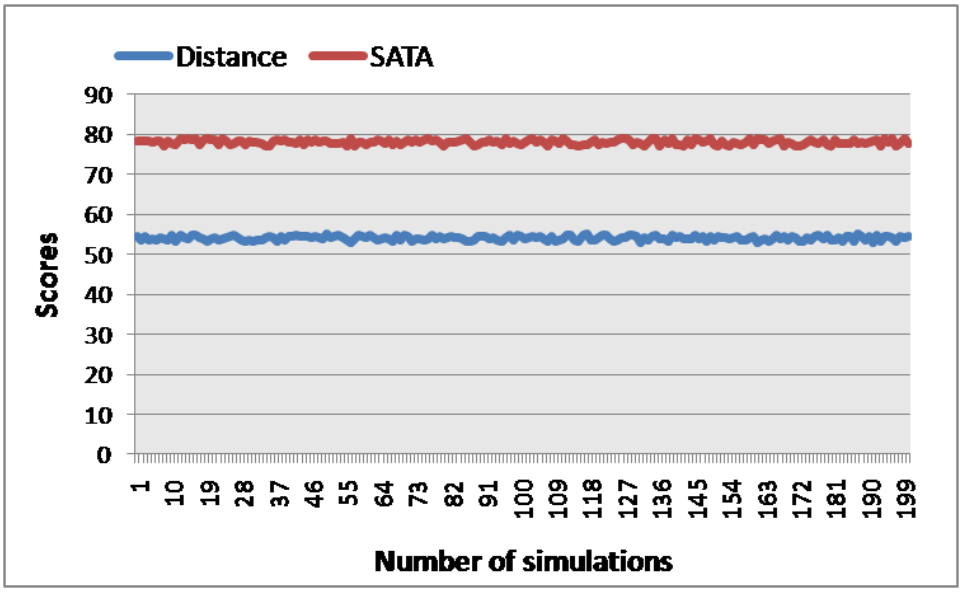

Figure 5.36: Comparison of system's performance using distance versus death time criterion for task allocation.

fires. By using the SATA algorithm the rescue agents could improve the rescue operation around 30 percent.

For our experiments, we have created four different simulation scenarios (For instance, see the configuration' files of the RoboCup Rescue simulator on the appendix at the end of this dissertation). The maps of these scenarios are shown in Figure 5.37. The scores obtained in those simulations are only dependant on the number of civilians alive, that is only on the ambulance team agents work. For our four scenarios, we have used four different maps which has the same importance, such as all scenarios have similar complexity.

Figure 5.38 presents the comparison between the performances of SATA algorithm versus distance approach. The SATA approach is better in all four scenarios. The average improvement by using SATA is around 30 - 35 percent in all the scenarios.

Notice that ambulance agents using SATA algorithm tend to accomplish the same task together. It is a consequence of the information compartition and it is a important issue because of the coordination and the collaboration between the agents are really important, because the agents efficiency can be improved if the agents collaborate with each other. The firefighter agents, the police agents and the ambulance agents work faster if they work in teams.

5.5.4 CATA WITH REPLANNING MECHANISM EXPERIMENTS

We have developed some experiments to evaluate the performance of the CATA algorithm and replanning mechanism developed which has been previously presented. Our methods

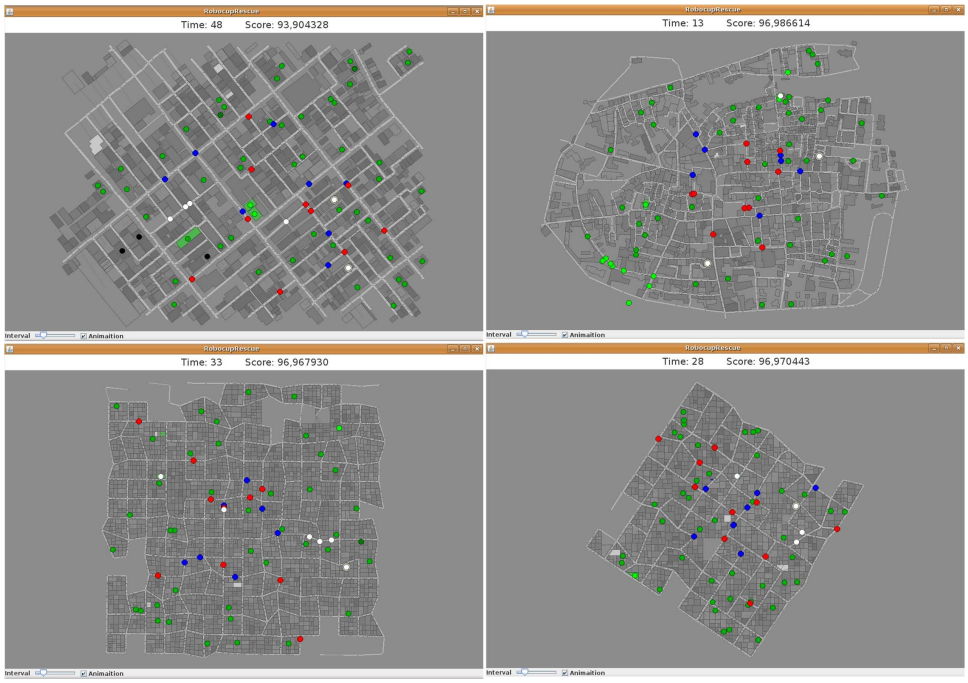

Figure 5.37: Scenarios for Kobe, Foligno, Random Large and Random Small maps.

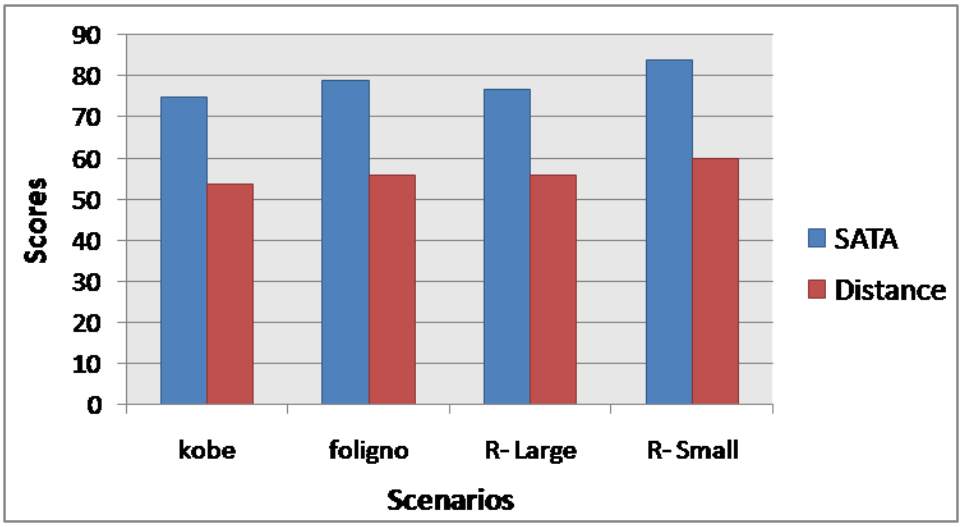

Figure 5.38: Comparison of SATA performance in four different scenarios.

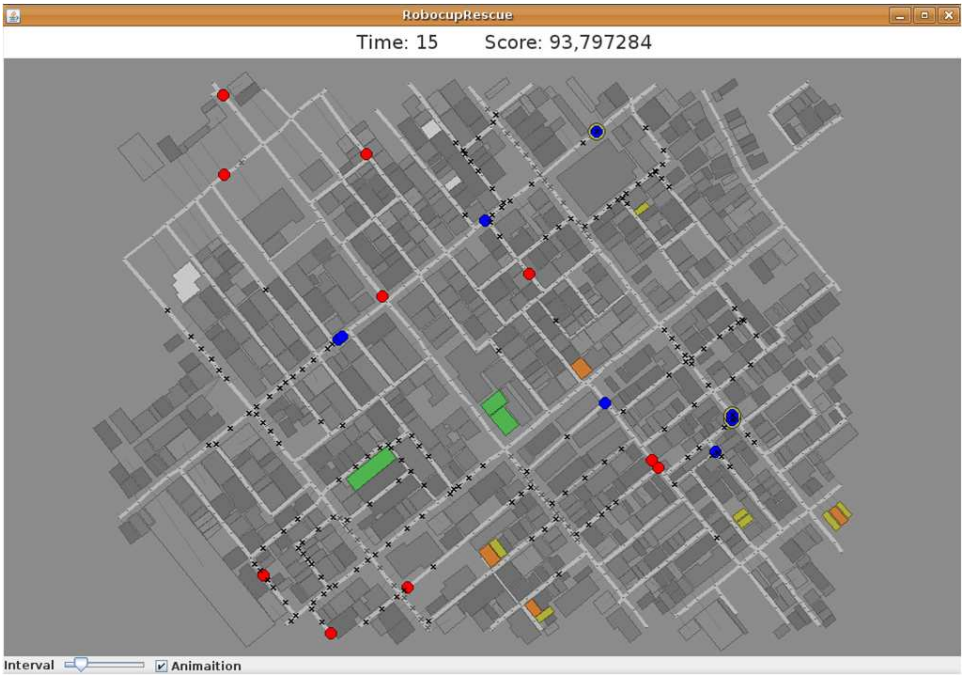

Figure 5.39: Kobe Map's initial situation.

have been applied to help the fire brigade agents to coordinate on the buildings on fire to extinguish. In order to allocate the fires, in our system, fire brigades select fires using the combinatorial auction mechanism explained before and they also use the replanning mechanisms when they found blocked roads into the scenario. It helps them to re-schedule their fire targets.

The combinatorial auction mechanism has been applied in the fire brigade operation because in the RoboCup Rescue scenario, the fire station agent has a better global view of the situation and therefore it can suggest fires to fire brigade agents. The fire brigade agents have however a more accurate local view, consequently they choose by which particular building on fire to extinguish they want to bid. By doing so, we can take advantage of the better global view of the fire station agent and the better local view of the fire brigade agent at the same time. To create the bids, agents use the urgency of fires strategy and the cost of the bid is calculated by using the distance criteria (see Section 5.3.6).

As mentioned before, experiments have been done in the RoboCup Rescue simulation environment. We have made our tests on a situation with a lot of fires, and with roads blocked. The simulations started with six fires, and the agents began to extinguish fires only after 15 simulation steps (to allow fires to propagate). Figure 5.39 shows a view of the city at time 15, just before the fire brigade agents begin to work. This gave us a hard situation to handle for the fire brigade agents. In this experiment, we are taking into account just the fire brigades agents and police force agents work, in order to provide the results of the fire extintion operation without other operations such as the ambulance agents operation when rescue victims.

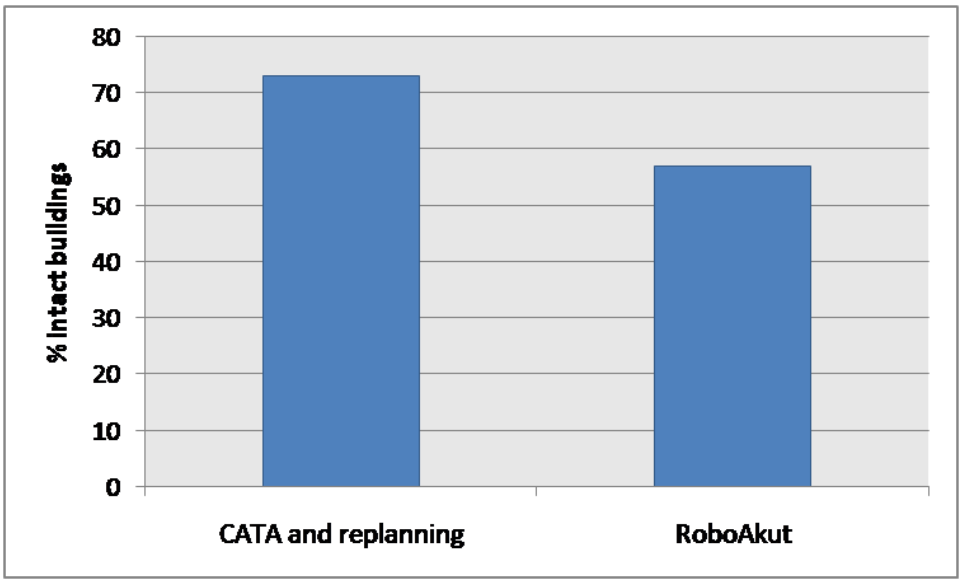

Figure 5.40: Comparison with RoboAkut strategy.

As explained before, the police forces (blue circles in the figure 5.39) are charged to clear the obstructed roads. However there are a lot of blocked roads due to the intensity of the seism. In this sense, the replanning algorithm previously explained is very useful helping fire brigades agents who remain blocked on the roads, waiting for the police force to clear them. In the implemented algorithm, agents can continue to reach the next fire in the list and they do not need to wait for the police forces to clear the road. It allows to fire brigades to save time and extinguish fires quickly.

For our experimentation, we have used the Kobe map scenario. We performed 200 simulations and we have compared the results obtained by our agents with other strategy. This strategy is from the team RoboAkut [6] which participed at the 2009 RoboCup Rescue simulation world competition. This team used single auction techniques to choose which fire area to extinguish. If we look at the performance of RoboAkut on our test map, they only obtained an average percentage of intact buildings of 57 percent. The comparison described in Figure 5.40 shows the advantage of our approach. By using our approach, fire brigades obtained an average percentage of intact buildings of 73 percent. This is a substantial improvement showing that the combinatorial auction approach and the replanning algorithm works well in the fire extintion operation. Notice that the substantial improvement is mainly due to the fact that agents are able to re-schedule the tasks. At the beggining of the simulation, without the replanning mechanism, there were several fire brigades stopped by the obstructed roads, and they were inactive during several time steps.

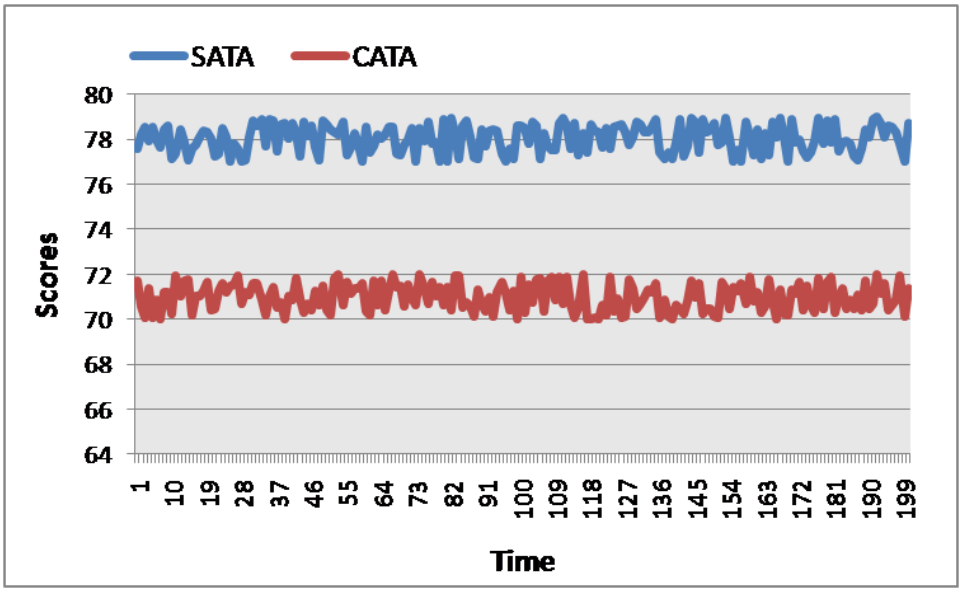

Figure 5.41: Comparison of performance among two methods for task allocation. Results using the SATA and CATA algorithms on ambulance team operation.

DISTRIBUTED (SATA) VERSUS CENTRALIZED (CATA) MECHANISM EXPERIMENTS

The goal of this experiment is to compare the performances and the communication burden of the decentralized scheduling approach compared to the centralized approach. For our experiments, again we have used four different simulation scenarios (Kobe, foligno, Random Small and Random Large). Those scenarios were designed to bring to the fore the ambulance team agents operation. We have removed everything that could interact with the ambulance team agents. It means the blocked roads as well as all the fires have been removed. The scores obtained in those simulations are consequently only dependant on the number of civilians alive, that is only on the ambulance team agents work.

The comparison results of both approaches applied to help to ambulance team to select victims is presented in Figure 5.41. The decentralized approach performs better in the case of ambulance team operation. It is slightly better in the four scenarios. Such as it is shown in the figure, it is around 10 percent better. Notice that the decentralized task allocation mechanism is in real time; so, agents avoid the waste of time due the solution wait. On the other hand, with this procedure, the number of exchanged bytes is considerably reduced as shown in Figure 5.42, where the averages results of exchanged bytes number during a single time step taken every 30 cycles are represented. In this sense, there is a 60 percent of reduction of the quantity of information sent with respect to a centralized one. This is mainly due to the fact that the agents do not have to send bids or other information they know, but only information about unattended victims to the central.

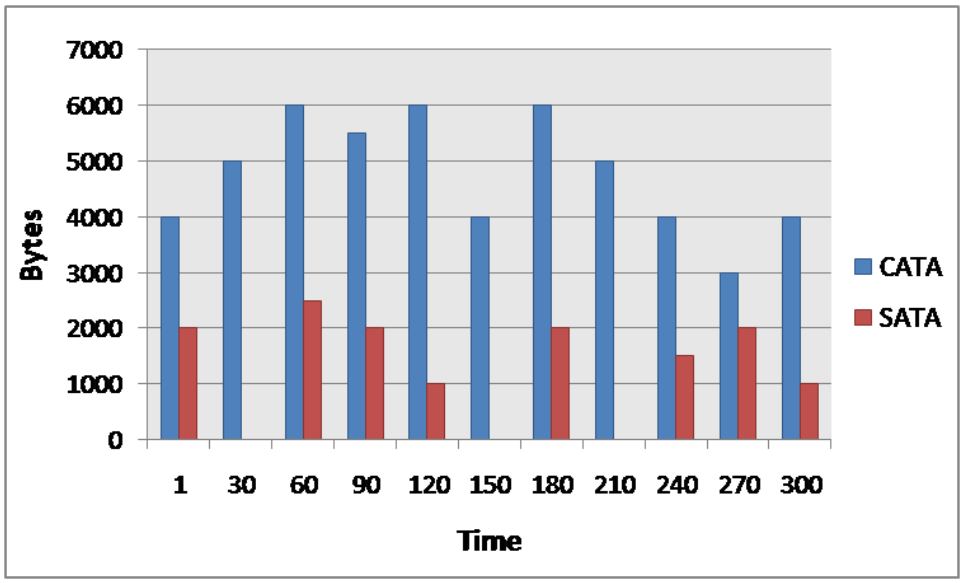

Figure 5.42: Number of bytes sent by both approaches.

5.6 Final remarks

In this chapter the implementation of SATA and CATA algorithms are presented. The SATA algorithm has been applied to help the ambulance team agents to coordinate themselves on the higher priority victims to rescue. By using this algorithm, ambulance teams tend to accomplish the same task together. It is a consequence of the shared information and it is an important issue within the rescue scenario because the agents' efficiency can be improved if the agents collaborate with each other. It means they work faster if they work in teams. The results show that the implementation of SATA algorithm improves the efficiency of the agent team when rescue victims. The main advantage of SATA is that it is a real time algorithm and it allows diminishing the communication burden in the system.

The other implemented algorithm called CATA has been applied to help the fire brigade agents to coordinate on the buildings on fire to extinguish. The combinatorial auction mechanism has been applied in the fire brigade operation because in the RoboCup Rescue scenario, the fire station agent has a better global view of the situation and therefore it can inform fires to fire brigade agents. The fire brigade agents have however a more accurate local view, consequently they choose by which particular building on fire to extinguish they want to bid. By using this algorithm, we can take advantage of the better global view of the fire station agent and the better local view of the fire brigade agent at the same time. The results show that the implementation of CATA improves the performance of the agents as they obtained a higher average percentage of intact buildings at the end to the simulation. Finally, the replanning mechanism implementation is also presented in this chapter. The results show that a substantial improvement is obtained mainly due to the fact that agents are able to replan the tasks. At the beginning of the simulation, without the replanning mechanism, there were several fire brigades stopped by the obstructed roads, and they were inactive during several time steps.

CHAPTER 6
Conclusions

This thesis is concluded by summarizing the contributions presented in the previous chapters. Moreover, some directions for possible future works in the subject of this thesis are presented.

6.1 REVISITING REQUIREMENTS

We start revisiting the requirements presented in chapter 4. As mentioned before, the RoboCup Rescue environment is conformed by heterogeneous agents with different capabilities and abilities. For this reason, we think different task allocation and coordination algorithms and mechanisms must be created according to the features of design of each one of them. In this regard, this dissertation has focused on the fire brigade and ambulance team rescue agents.

As explained in previous sections, the fire station agent has a better global view of the situation. Therefore it can suggest good fire areas to fire brigade agents. On the other hand, the fire brigade agents have a more accurate local view, consequently they can choose more efficiently which particular building on fire to extinguish. By doing so, we can take advantage of the better global view of the fire station agent and the better local view of the fire brigade agent at the same time. For this reason, we have designed a direct supervision and semi-centralized algorithm using combinatorial auctions. That way, stations report the observe fires to fire brigades, and simultaneously, fire brigades can make their own decisions about which fires they want to bid given its local view of the scenario. In addition, we have demonstrated why combinatorial performs better than single auctions in section 5.3.3. and in the results section 5.5.4.

Regards ambulance team agents operation, they have to rescue the victims, but the number of victims that can be rescued depends a lot on the order in which they are rescued. Furthermore, the agents have to be able to adapt the plans frequently to take the dynamic changes of the environment into consideration. In the RoboCup Rescue scenario, the tasks are not known at the beginning of the simulation. For this reason, agents have to explore the environment to find the tasks and then incorporate them in their schedule. In addition, RoboCup Rescue is an uncertain environment, it means, the tasks' parameters could change between two time steps. In this case, the system's performance depends not only on the maximization or the optimization criterion, but also on the agents' capac-

ity to adapt their schedule efficiently. Taking into account the previous requirements of ambulance teams, a real time scheduling algorithm using a sequencing theory has been designed. Moreover, one important parameter that the ambulance team agents have to know is the expected death time of the victims. This parameter is quite important for the ambulance team agents to make good schedules. To this end, a new case-based approach has been used to estimate the expected death time. With the case-based approach, we have used past cases or experiences to predict the death time of each victim.

In the RoboCup Rescue scenario, another important feature is the robustness of the algorithms and mechanisms designed. With robustness, we mean, the capacity of the system to maintain good performances when faced with hard communication constraints. In this dissertation the communication burden is evaluated by considering the amount of information (measured in bytes) transmitted by the agents. In our experiments, we show that a real-time scheduling system is more robust, it means, it is less sensitive to hard communication constraints.

During the RoboCup Rescue simulation, agents can face some hard environmental problems. For instance, agents are stopped on the blocked roads. Police forces must help agents in these situations and clearing the obstructed roads. However, police forces may have too many scheduled tasks and help could take a long time for arrive. At this moment, an algorithm for replanning of tasks is necessary. In this dissertation a new replanning algorithm has been designed.

In the RoboCup Rescue domain, the cooperation between heterogeneous rescue agents is very important. For instance, ambulance teams need information from the fire brigades and police forces to find the victims or the ambulance teams and fire brigades need the police forces to clear the obstructed roads. To this end, we have implemented a flow communication and a cooperation strategy among the heterogeneous agents to share the most relevant information such as the victims' information within the rescue scenario. Notice, it is very important for the ambulance team agents, because finding victims is a very hard task. In addition, our rescue agents try to explore only unvisited buildings each time step.

The agents need algorithms that take into account their preferences, for example, the priority of the victims and the urgency of fires. In this sense, the algorithms we have designed take into account such relevant matters for tasks allocation within our application domain. Firstly, our algorithm based on combinatorial auctions includes these preferences into the bid sent from the agent to the central; And secondly, our scheduling algorithm takes into account the preferences of the ambulance teams using the estimation of the victims' death time to make the scheduling.

Regards the solution of the winner determination problem in combinatorial auctions, the RoboCup Rescue winner determination process differs from the classic problem in 2 major ways: (1) The RoboCup Rescue auction is a reverse auction, and we are interested in minimizing cost rather than maximizing revenue. (2) In our approach the free disposal

assumption applies (Note that under the free disposal assumption, each task would be included at most once rather than exactly once). The main goal is to find the optimal solution and we don't care if all tasks are covered by the bids sent by the agents. To deal with the problem of allocated tasks, we use an adjustment phase in our combinatorial auction algorithm (see section 4.5.1).

6.2 Results analysis

By using the algorithms implemented in this thesis, there has been quite a significant increase in the rescue agents' performance within the RoboCup Rescue scenario. At first, our case-base approach for victims' death time estimation has improved a lot (around 241 percent) with regard the approach that only considers the current damage value of the civilian (hp/damage). Moreover, by implementing our strategy of cooperation between heterogeneous agents, the rescue operation has improved around 21 percent. In addition, by using our scheduling algorithm (SATA) our ambulance agents could improve around a 30 percent his performance. At the same way, the performance of the fire brigades has improved by using the algorithm based on combinatorial auctions (CATA). CATA and Replanning algorithms perform around 22 percent better than the single auctions strategy from team RoboAkut [6] which participed at the 2009 RoboCup Rescue simulation world competition. In this regard, by implementing CATA and replanning approaches, fire brigades obtained an average percentage of intact buildings of 73 percent against the 57 percent obtained from RoboAkut.

6.3 Contributions

This thesis has contributed to the state-of-the-art in the task allocation and coordination areas for cooperative environments in the following items:

1. **The Hybrid approach**

   A hybrid study for the coordination of tasks in a dynamic setting such as the rescue scenario is presented. It uses a decentralized and a semi-centralized mechanism. Our results show that a decentralized approach based on mutual adjustment can be more flexible and give better results than a centralized approach using direct supervision. The decentralized algorithm is based on the theory of sequencing and takes into account time constraints of the tasks to be assigned. The centralized algorithm is based on market mechanisms such as auctions. The combinatorial auctions may be viewed as semi distributed mechanisms because agents take their own decisions about what tasks they want to bid independently of the central agent or supervisor.

2. **Market based techniques**

   - *An algorithm for task allocation using combinatorial auctions is presented*. This algorithm has been designed to be used in a rescue scenario such as the RoboCup Rescue environment. The implementation and results show that such

mechanisms perform well and that it allows taking advantage of the global vision of the central agent and the local view of the agent team. This issue provides to the agents with a more accurate knowledge of the environment.

- *Three strategies to bid generation have been implemented*. The first strategy is based on the distance between the agent and the tasks. If agents bid for tasks that are closer to them, it generates a good result at the time of tasks assessment. This strategy has been implemented for the fire brigades taking into account the distances between them and each of the fires into the bid. The second strategy has been developed for ambulance team agents. This strategy takes into account the death time of the victims to make an arrangement of them within the generated bid. And finally, the third strategy is related to the urgency of fires. In this sense, the fire brigades take into account the urgency and priority of fires in the bids submitted. Moreover, a new method to able agents to decide on what tasks a bidder should bid on has been proposed. In this sense, we have used a heuristic based approach where the bidders bid on combinations of $size < q$ (see section 5.3.6.).
- *Algorithms to the winner determination in combinatorial auctions have been studied*. Two winner determination algorithms which find an optimal solution have been applied. Firstly, a classical DFS algorithm has been used. DFS starts at the root and explores as far as possible along each branch of the tree before backtracking. Secondly, the Collins' algorithm has been used. It is an optimal tree search algorithm which is based on an iterative deepening A* formulation. The Collins' algorithm presents some advantages regards to the original Sandholm's algorithm such as it has improved upon it by specifying a means to minimize the mean branching factor in the generated search tree. The Collins algorithm implementation has not been straightforward. We have required implementing dummy bids to face the free disposal feature.

3. **Application of scheduling theory in multi-agent systems**

   - An algorithm based on sequencing techniques for schedule tasks between agents in multi-agent environments is presented. The main advantage of this algorithm is that it is fully distributed, real-time. In this algorithm the time constraints of the tasks are taken into account. Another advantage of this algorithm is that it can be solved in polynomial time and helps to reduce communication in dynamic environments such as our domain.
   - We have shown the application of theories of scheduling such as sequencing to resolve the problem of allocation of tasks between agents in a distributed manner. We have also emphasized the usefulness of keeping the domains of task scheduling and multi-agent systems linked. Those two domains can help each other to find suitable solutions to common problems.
   - We have viewed that in the distributed approach, all agents are considered to be one big resource working on one task at a time and trying to maximize the number of tasks accomplished in the time allowed.

4. **Case-based approach**

   We have presented a new application of the case-based techniques to estimate the value of an uncertain parameter of a task. We have presented results showing the efficiency of the predictions in the RoboCup Rescue simulation. Three measures to find the death time value are used: median, mean and minimal. The mean death time estimation obtained the best results.

5. **Replanning of tasks**

   A replanning mechanism for multi-agent systems has been designed. The replanning helps the agents to repair his plans when external and environmental problems prevent from fulfil them. Replanning is very important in dynamic and uncertain environments where conditions are constantly changing and the agents have to cope with new events. We found the continuity of a plan and its success is linked with the design of a efficient mechanism for replanning.

6. **Multi-agent cooperation**

   - A flow of communication between agents and centers in the RoboCup Rescue scenario has been designed. This allows rescue agents and center agents to share information through existing communication channels in the simulator. The sharing of information is important in distributed and uncertainty scenarios where there are strong restrictions on communication.
   - A cooperation approach which allows agents to share information and improve their state of the world has been implemented. This mechanism implements cooperation between different types of agents. This approach has improved considerably the performance of agents in the RoboCup Rescue scenario. We can conclude that the agents efficiency can be improved if the agents collaborate with each other. They work faster if they work in teams and share important information.

7. **Reduction of communication burden**

   We have showed that a decentralized scheduling system can offer a better performance than a centralized one, while diminishing the amount of information transmitted between the agents. This was done in the objective of being more robust to constraints on the communications.

8. **Experiments**

   We have presented some tests in the RoboCup Rescue environment showing that the agents can efficiently coordinate tasks and that efficient task allocation and cooperation is really helpful to improve the agents performances in such dynamical environments. The agents obtained good results with our proposed cooperation strategy and the tasks allocation mechanisms. They had a improvement around 40 percent.

6.4 FUTURE WORK

The algorithms and mechanisms presented in this thesis are a contribution to the field of cooperative multi-agent systems. However, there is much work to do in order to developed near optimal cooperative multi-agent systems in complex environments. In this section, we present some ideas on how the approaches presented in this thesis could be extended or improved. These ideas are divided according to the main contributions of this thesis.

6.4.1 COMBINATORIAL AUCTIONS MECHANISM (CATA)

The current efforts and results of this thesis show that market based mechanisms and auction paradigms are very useful for task allocation in cooperative multi-agent environments. In addition, in this thesis some strategies have been created to help agents to create bids into the auction. However, it is still difficult to choose the necessary information to be included into the bid created by the agent. This thesis reports some results obtained by using some quite efficient bid configuration strategies. The agents could successfully to generate bids by using these strategies. In spite of this, more extensive studies on developing new methods for soliciting desirable bids for collections of tasks with more complex time constraints should be carried out to guarantee a low cost combination that covers the entire collection of tasks.

Moreover, in this thesis a size of the bid and number of bids strategies have been developed to deal with the too many bids generated problem which will cause the winner determination algorithm to take excessive time. In this sense, other strategies should be designed and compared with the strategies on this thesis with the goal to find the most optimal strategy to follow by agents.

Finally, in this dissertation, two rescue winner determination algorithms in combinatorial auctions have been applied. In this sense, other algorithms to solve the rescue winner determination problem may be applied and compared within the rescue scenario with the goal to find the most optimal performance into the solution for the particular domain.

6.4.2 SCHEDULING MECHANISM (SATA)

The scheduling mechanism has been implemented for the ambulance team operation to help the ambulance team agents to coordinate themselves on the highest priority victims to rescue. This algorithm may be implemented in other kind of agents. For instance, it may be applied for the fire brigade agents operation. Fires extinction is the task performed for fire brigades, then, agents may schedule tasks taking into account fieriness and other urgency properties of the fires into the scheduling. In spite of in this dissertation, a similar scheduling approach using fires' urgency properties for bid generation has been implemented, a future

study may include a bigger number of properties to describe the urgency of each fire within the environment.

An another improvement of our scheduling approach would be to consider the moving time between agent and tasks. In our experiments, we have used a constant time, but the scheduler agent could generate better schedules if the real moving time was considered. Using better estimations for the moving times could be a first step to improve the schedules.

6.4.3 Replanning

Usually, rescue agents in the RoboCup Rescue simulation have to deal with the problem of blocked roads during the rescue operation. In this thesis a replanning algorithm has been designed to allow agents tackle with this circumstance once they have a created plan. Multi-agent systems sometimes undergo environmental problems or changes that cause plans to become disrupted or out of date, such that the coordinated agent plans need to be repaired or replaced. In this dissertation we assume that the plans of a group of agents have been initially coordinated, but that a change occurs that forces one or more agents to revise their plans such that some coordinated tasks must be postponed. Rather than starting the planning process over again from scratch, or even just starting the coordination process over again, we want to reuse the results of the prior planning and coordination process, effectively repairing the coordinated solution to fit the revised plans. In this sense, a new research line may be open thanks to our replanning approach. We think more extensive studies on this topic should be carried out to create a complete model including an exhaustive representation of the scenario to generate efficient real-time recoordinated plans.

6.5 Publications

The results of these doctoral studies have been presented in the next publications:

6.5.1 List of publications related to this PhD

Silvia Suárez, Christian Quintero and Josep Lluís de la Rosa. Decentralized Dynamic Task Allocation: A Crisis Management Approach. *Autonomous Robots Journal* (submitted). 2010.

Silvia Suárez, Christian Quintero and Josep Lluís de la Rosa. A Real Time Approach For Task Allocation In A Disaster Scenario. *In Advances in Soft Computing: Advances in Practical Applications of Agents and Multiagent Systems.* Volume 70/2010, ISBN: 978-3-642-12383-2, Pages 157-162. Springer Berlin / Heidelberg, April of 2010.

Silvia Suárez, Christian Quintero and Josep Lluís de la Rosa. Improving Tasks Allocation And Coordination In A Rescue Scenario. *In the European Control Conference 2007 (ECC'07)*. Kos, Greece, July 2-5 of 2007.

Silvia Suárez and Beatriz López. Reverse Combinatorial Auctions for allocating Resources in Rescue Scenario. *In the International Conference About Planning and Scheduling (ICAPS' 06)*. Constraint Satisfaction Techniques for Planning and Scheduling Problems. The English Lake District, Cumbria, U.K. June 6-10 of 2006.

Silvia Suárez, John Collins and Beatriz López. Improving Rescue Operation in Disasters. Approaches about Task Allocation and Re-scheduling. *In Proceedings of The 24th Annual Workshop of the UK Planning and Scheduling Special Interest Group (PLANSIG'05)*. City University, London, UK. December 15-16 of 2005.

Silvia Suárez, Beatriz López, Josep Lluís de la Rosa. MCD Method for resource distribution in a large-scale disaster. *In "X Conferencia de la Asociación Española para la Inteligencia Artificial" (CAEPIA)*, volumen II, pages 261-264, San Sebastin, Spain, November 11-14 of 2003.

Beatriz López, Silvia Suárez and Josep Lluís de la Rosa. Task Allocation in rescue operations using combinatorial auctions. *In Frontiers in Artificial Intelligence and Applications: Artificial Intelligence Research and Development*, pages 233-243, IOS Press, ISBN 1 58603 378 6, Netherlands, October of 2003.

Silvia Suárez, Beatriz López and Josep Lluís de la Rosa. Co-operation strategies for strengthening civil agents lives in the RoboCup Rescue simulator scenario. *In Proceedings of First International Workshop on Synthetic Simulation and Robotics to Mitigate Earthquake Disaster.* Associated to RoboCup 2003. Padova, Italy. July 5-10 of 2003.

Silvia Suárez, Beatriz López and Josep Lluís de la Rosa. Girona-Eagles Rescue Team. *In Proceedings of the Internacional RoboCup Rescue Symposium. Rescue Team Description Papers*. Padova, Italy, July of 2003.

Silvia Suárez, Beatriz López, Josep Lluís de la Rosa and Esteve del Acebo. Integration Of Fuzzy Filtering, Case-Based Reasoning and Multiple Criteria Decision Techniques in Rescue Operations. *In Proceedings of Workshop of physical Agents (WAF)*. Alicante, Spain. April 3, 4 and 5 of 2003.

6.5.2 Other publications

Beatriz López, Joaquim Meléndez and Silvia Suárez. Ontology for Integrating Heterogeneous Tools for Supervision, Fault Detection and Diagnosis. *Second International Conference on Informatics in Control, Automation and Robotics. Intelligent Control Systems and Optimization*, Volume I, pages 125-132, INSTICC Press, Institute for Systems and Tech-

nologies of Information, Control and Communication, Portugal 2005.

Silvia Suárez, Beatriz López and Joaquim Meléndez. Towards holonic multiagent systems: Ontology for supervision tool boxes. *In "Workshop de Agentes Inteligentes en el tercer milenio"*. November 10 of 2003. San Sebastián, Spain.

APPENDIX A

A.1 CONFIGURATION'S FILE OF THE KOBE'S MAP IN THE ROBOCUP RESCUE SIMULATOR

[MotionLessObject]

FireStationNum=1
PoliceOfficeNum=1
AmbulanceCenterNum=1
RefugeNum=3
FireStation0=3,1,5165,22926500,3795600,-1,0,0,0
PoliceOffice0=4,1,5108,22785900,3707100,-1,0,0,0
AmbulanceCenter0=2,1,5417,22939700,3764300,-1,0,0,0
Refuge0=5,1,4755,22968500,3627800,-1,0,0,0
Refuge1=5,1,5368,22963200,3638100,-1,0,0,0
Refuge2=5,1,5207,22891400,3599400,-1,0,0,0

[MoveObject]

CivilianNum=72
AmbulanceTeamNum=6
FireBrigadeNum=10
PoliceForceNum=8
Civilian0=6,1,4951,22853700,3539300,-1,0,0,0
Civilian1=6,1,4955,22877500,3550700,-1,0,0,0
Civilian2=6,1,5445,22788200,3660800,-1,0,0,0
Civilian3=6,1,5361,22814800,3680200,-1,0,0,0
Civilian4=6,1,5176,23111200,3668100,-1,0,0,0
Civilian5=6,1,5153,22909600,3667400,-1,0,0,0
Civilian6=6,1,5285,23144600,3801900,-1,0,0,0
Civilian7=6,1,4762,23098400,3766700,-1,0,0,0
Civilian8=6,1,5208,22862600,3694500,-1,0,0,0
Civilian9=6,1,5304,23124100,3577300,-1,0,0,0
Civilian10=6,1,5430,23065300,3632800,-1,0,0,0

Civilian11=6,1,5380,23086400,3543300,-1,0,0,0
Civilian12=6,1,5354,22816100,3585600,-1,0,0,0
Civilian13=6,1,4858,22973200,3561100,-1,0,0,0
Civilian14=6,1,5263,22802100,3729000,-1,0,0,0
Civilian15=6,1,4889,23078000,3777300,-1,0,0,0
Civilian16=6,1,4845,23055900,3759900,-1,0,0,0
Civilian17=6,1,4994,22863200,3606700,-1,0,0,0
Civilian18=6,1,5380,23086400,3543300,-1,0,0,0
Civilian19=6,1,5160,22999700,3729900,-1,0,0,0
Civilian20=6,1,5024,22797500,3625000,-1,0,0,0
Civilian21=6,1,4969,22955600,3691500,-1,0,0,0
Civilian22=6,1,4802,23060300,3755800,-1,0,0,0
Civilian23=6,1,5431,22937800,3513800,-1,0,0,0
Civilian24=6,1,5022,23134200,3552900,-1,0,0,0
Civilian25=6,1,4752,23023600,3694000,-1,0,0,0
Civilian26=6,1,4871,22794200,3653900,-1,0,0,0
Civilian27=6,1,5332,23068500,3702100,-1,0,0,0
Civilian28=6,1,4756,23003700,3719300,-1,0,0,0
Civilian29=6,1,4994,22863200,3606700,-1,0,0,0
Civilian30=6,1,5227,23145900,3814800,-1,0,0,0
Civilian31=6,1,4896,23109500,3608000,-1,0,0,0
Civilian32=6,1,4839,23007300,3666500,-1,0,0,0
Civilian33=6,1,5364,22952200,3681200,-1,0,0,0
Civilian34=6,1,4769,23027000,3523300,-1,0,0,0
Civilian35=6,1,5066,22873400,3614300,-1,0,0,0
Civilian36=6,1,5028,22941200,3600400,-1,0,0,0
Civilian37=6,1,5136,23036700,3604600,-1,0,0,0
Civilian38=6,1,5092,23012200,3574600,-1,0,0,0
Civilian39=6,1,5374,22999700,3798300,-1,0,0,0
Civilian40=6,1,5209,22985800,3734500,-1,0,0,0
Civilian41=6,1,5240,23127200,3584100,-1,0,0,0
Civilian42=6,1,5376,23022200,3740200,-1,0,0,0
Civilian43=6,1,5194,23140700,3547500,-1,0,0,0
Civilian44=6,1,5045,23111900,3687200,-1,0,0,0
Civilian45=6,1,4745,22835700,3601500,-1,0,0,0
Civilian46=6,1,5410,22960900,3685200,-1,0,0,0
Civilian47=6,1,4853,23058800,3592000,-1,0,0,0
Civilian48=6,1,4977,23083500,3795000,-1,0,0,0
Civilian49=6,1,5142,22875900,3523600,-1,0,0,0
Civilian50=6,1,5047,23160200,3654500,-1,0,0,0
Civilian51=6,1,4886,22933300,3751000,-1,0,0,0
Civilian52=6,1,5012,23086000,3630100,-1,0,0,0
Civilian53=6,1,5274,22927500,3759200,-1,0,0,0

Civilian54=6,1,4924,22913900,3578800,-1,0,0,0
Civilian55=6,1,5135,23098600,3620600,-1,0,0,0
Civilian56=6,1,5039,22928900,3741800,-1,0,0,0
Civilian57=6,1,5174,23104700,3793700,-1,0,0,0
Civilian58=6,1,5211,23014000,3685100,-1,0,0,0
Civilian59=6,1,4761,23078400,3803800,-1,0,0,0
Civilian60=6,1,5375,22994200,3783700,-1,0,0,0
Civilian61=6,1,4903,23161000,3573500,-1,0,0,0
Civilian62=6,1,4877,22821400,3656300,-1,0,0,0
Civilian63=6,1,5047,23160200,3654500,-1,0,0,0
Civilian64=6,1,5204,22843600,3624600,-1,0,0,0
Civilian65=6,1,4775,23076200,3565800,-1,0,0,0
Civilian66=6,1,4767,23046500,3620500,-1,0,0,0
Civilian67=6,1,5319,22928900,3590700,-1,0,0,0
Civilian68=6,1,5393,23060200,3644000,-1,0,0,0
Civilian69=6,1,5153,22909600,3667400,-1,0,0,0
Civilian70=6,1,4841,22889900,3725600,-1,0,0,0
Civilian71=6,1,4950,23127900,3522800,-1,0,0,0
AmbulanceTeam0=6,0,6062,23070100,3630800,-1,0,0,0
AmbulanceTeam1=6,0,5490,22830700,3671000,-1,0,0,0
AmbulanceTeam2=6,2,6001,23061800,3764200,6000,23064900,3760000,3
AmbulanceTeam3=6,0,5967,23009500,3591000,-1,0,0,0
AmbulanceTeam4=6,0,6051,22917700,3780300,-1,0,0,0
AmbulanceTeam5=6,2,5616,23014000,3698000,5705,23016800,3694400,3
FireBrigade0=6,2,5819,23057600,3601100,5820,23061100,3596800,3
FireBrigade1=6,2,5724,22808400,3769000,6152,22801300,3762900,1
FireBrigade2=6,2,5689,22913500,3532100,5690,22942500,3555700,23
FireBrigade3=6,0,5782,22807400,3815100,-1,0,0,0
FireBrigade4=6,0,6094,22983300,3712300,-1,0,0,0
FireBrigade5=6,2,5796,22899100,3699600,5780,22901600,3696400,0
FireBrigade6=6,0,5988,22869300,3506100,-1,0,0,0
FireBrigade7=6,2,5686,22830700,3539100,5989,22834900,3534000,0
FireBrigade8=6,2,5708,22889600,3781200,5856,22885300,3786700,1
FireBrigade9=6,0,5822,23053700,3605500,-1,0,0,0
PoliceForce0=6,0,5719,23121300,3804500,-1,0,0,0
PoliceForce1=6,0,6184,23083400,3614600,-1,0,0,0
PoliceForce2=6,2,6002,23030800,3801100,5861,23026400,3797200,0
PoliceForce3=6,2,6242,23135300,3648500,6224,23145600,3635100,16
PoliceForce4=6,0,5709,22898000,3740100,-1,0,0,0
PoliceForce5=6,0,6038,22897500,3642400,-1,0,0,0
PoliceForce6=6,0,6203,22816100,3743100,-1,0,0,0
PoliceForce7=6,2,5651,23089000,3644700,5652,23092800,3646800,1

[FirePoint]

FirePointNum=0

A.2 Configuration's file of the Foligno's map

[MotionLessObject]

FireStationNum=1
PoliceOfficeNum=1
AmbulanceCenterNum=1
RefugeNum=4
FireStation0=3,1,2491,506990,91127,-1,0,0,0
PoliceOffice0=4,1,1708,359716,337251,-1,0,0,0
AmbulanceCenter0=2,1,1671,413784,394510,-1,0,0,0
Refuge0=5,1,1987,215478,235855,-1,0,0,0
Refuge1=5,1,1860,505729,506591,-1,0,0,0
Refuge2=5,1,2329,127886,121338,-1,0,0,0
Refuge3=5,1,1608,491672,457036,-1,0,0,0

[MoveObject]

CivilianNum=72
AmbulanceTeamNum=6
FireBrigadeNum=10
PoliceForceNum=8
Civilian0=6,1,2077,446128,249269,-1,0,0,0
Civilian1=6,1,2372,195875,132886,-1,0,0,0
Civilian2=6,1,2108,517704,245247,-1,0,0,0
Civilian3=6,1,2207,229457,407056,-1,0,0,0
Civilian4=6,1,1584,562702,335221,-1,0,0,0
Civilian5=6,1,2279,144133,347239,-1,0,0,0
Civilian6=6,1,2216,205821,423852,-1,0,0,0
Civilian7=6,1,1767,656750,402087,-1,0,0,0
Civilian8=6,1,2384,301188,144583,-1,0,0,0
Civilian9=6,1,1843,305454,486955,-1,0,0,0
Civilian10=6,1,2156,489568,166220,-1,0,0,0
Civilian11=6,1,1551,559253,417620,-1,0,0,0
Civilian12=6,1,1509,498865,430569,-1,0,0,0
Civilian13=6,1,2208,231904,399002,-1,0,0,0
Civilian14=6,1,1629,507233,324765,-1,0,0,0
Civilian15=6,1,1595,544416,325227,-1,0,0,0
Civilian16=6,1,2033,240001,208162,-1,0,0,0

Civilian17=6,1,2339,196616,142472,-1,0,0,0
Civilian18=6,1,1615,446909,434047,-1,0,0,0
Civilian19=6,1,1737,394068,433192,-1,0,0,0
Civilian20=6,1,1635,459278,319030,-1,0,0,0
Civilian21=6,1,1846,346610,494131,-1,0,0,0
Civilian22=6,1,2474,510437,153673,-1,0,0,0
Civilian23=6,1,1884,556944,542161,-1,0,0,0
Civilian24=6,1,1601,501325,445550,-1,0,0,0
Civilian25=6,1,2150,445357,162060,-1,0,0,0
Civilian26=6,1,1605,509231,455511,-1,0,0,0
Civilian27=6,1,2443,453368,30834,-1,0,0,0
Civilian28=6,1,1797,639531,325022,-1,0,0,0
Civilian29=6,1,1881,500414,531003,-1,0,0,0
Civilian30=6,1,1734,394068,407852,-1,0,0,0
Civilian31=6,1,1566,508116,422827,-1,0,0,0
Civilian32=6,1,2020,167931,229729,-1,0,0,0
Civilian33=6,1,2029,201872,196248,-1,0,0,0
Civilian34=6,1,2428,342868,115308,-1,0,0,0
Civilian35=6,1,2555,608377,89697,-1,0,0,0
Civilian36=6,1,2542,594277,143179,-1,0,0,0
Civilian37=6,1,1783,570785,333970,-1,0,0,0
Civilian38=6,1,2560,698301,254098,-1,0,0,0
Civilian39=6,1,2095,500769,257383,-1,0,0,0
Civilian40=6,1,1888,567512,523240,-1,0,0,0
Civilian41=6,1,2105,543049,259148,-1,0,0,0
Civilian42=6,1,2463,554137,159773,-1,0,0,0
Civilian43=6,1,1765,654173,365584,-1,0,0,0
Civilian44=6,1,2525,595639,134209,-1,0,0,0
Civilian45=6,1,2412,317847,73100,-1,0,0,0
Civilian46=6,1,1745,573936,428800,-1,0,0,0
Civilian47=6,1,1961,310865,213894,-1,0,0,0
Civilian48=6,1,1596,509350,359014,-1,0,0,0
Civilian49=6,1,1870,548732,549813,-1,0,0,0
Civilian50=6,1,2239,228111,444365,-1,0,0,0
Civilian51=6,1,2002,310117,230376,-1,0,0,0
Civilian52=6,1,2332,148320,173362,-1,0,0,0
Civilian53=6,1,1889,585670,521188,-1,0,0,0
Civilian54=6,1,1776,604258,401629,-1,0,0,0
Civilian55=6,1,2340,202390,141940,-1,0,0,0
Civilian56=6,1,2553,627745,130696,-1,0,0,0
Civilian57=6,1,2326,97324,170415,-1,0,0,0
Civilian58=6,1,1754,588660,449117,-1,0,0,0
Civilian59=6,1,1861,495949,512303,-1,0,0,0

Civilian60=6,1,2163,527176,207348,-1,0,0,0
Civilian61=6,1,2499,512624,32151,-1,0,0,0
Civilian62=6,1,2437,383528,84988,-1,0,0,0
Civilian63=6,1,2163,527176,207348,-1,0,0,0
Civilian64=6,1,2539,604079,143627,-1,0,0,0
Civilian65=6,1,2522,582006,120889,-1,0,0,0
Civilian66=6,1,2045,214563,173123,-1,0,0,0
Civilian67=6,1,2251,266316,474636,-1,0,0,0
Civilian68=6,1,1731,366994,383127,-1,0,0,0
Civilian69=6,1,1569,534382,410709,-1,0,0,0
Civilian70=6,1,1731,366992,383127,-1,0,0,0
Civilian71=6,1,1569,534380,410709,-1,0,0,0
AmbulanceTeam0=6,2,3403,499944,266622,3402,491212,265438,8
AmbulanceTeam1=6,2,3878,609161,136115,3879,599294,138133,4
AmbulanceTeam2=6,2,3319,228055,212077,3382,225700,202168,6
AmbulanceTeam3=6,2,3711,33860,198813,3703,66891,202849,19
AmbulanceTeam4=6,2,3019,633911,452728,3018,615654,450661,10
AmbulanceTeam5=6,0,3797,340187,123638,-1,0,0,0
FireBrigade0=6,0,3910,668641,51575,-1,0,0,0
FireBrigade1=6,2,2884,455686,319325,2883,456719,312780,0
FireBrigade2=6,0,3592,212862,413783,-1,0,0,0
FireBrigade3=6,2,3242,242919,278509,3259,241461,272679,4
FireBrigade4=6,2,3275,237649,261579,3257,236303,257991,1
FireBrigade5=6,2,2975,318608,402257,2972,322692,395909,2
FireBrigade6=6,0,3497,555074,221556,-1,0,0,0
FireBrigade7=6,2,3850,489582,84875,3846,499898,84427,2
FireBrigade8=6,0,3701,114879,238055,-1,0,0,0
FireBrigade9=6,0,2873,439816,368092,-1,0,0,0
PoliceForce0=6,0,3281,251039,259841,-1,0,0,0
PoliceForce1=6,2,3207,343590,334859,3200,347142,327311,2
PoliceForce2=6,0,3828,543512,136003,-1,0,0,0
PoliceForce3=6,0,3433,612646,232582,-1,0,0,0
PoliceForce4=6,0,3351,195360,165612,-1,0,0,0
PoliceForce5=6,0,3171,495124,543047,-1,0,0,0
PoliceForce6=6,2,2585,538601,383065,2566,538748,378341,1
PoliceForce7=6,2,2781,447169,380148,2779,448104,380211,0

[FirePoint]

FirePointNum=0

A.3 Configuration's file of the Ramdom Large's map

[MotionLessObject]

FireStationNum=1
PoliceOfficeNum=1
AmbulanceCenterNum=1
RefugeNum=3
FireStation0=3,1,14000,121989,137293,-1,0,0,0
PoliceOffice0=4,1,13874,141076,579455,-1,0,0,0
AmbulanceCenter0=2,1,13162,350675,189419,-1,0,0,0
Refuge0=5,1,13877,158452,579455,-1,0,0,0
Refuge1=5,1,13893,153364,470170,-1,0,0,0
Refuge2=5,1,11895,613432,594798,-1,0,0,0

[MoveObject]

CivilianNum=72
AmbulanceTeamNum=6
FireBrigadeNum=10
PoliceForceNum=8
Civilian0=6,1,14005,142873,137293,-1,0,0,0
Civilian1=6,1,12433,497873,677531,-1,0,0,0
Civilian2=6,1,13352,319694,440602,-1,0,0,0
Civilian3=6,1,13601,255338,228812,-1,0,0,0
Civilian4=6,1,13339,283378,430016,-1,0,0,0
Civilian5=6,1,12708,425127,247039,-1,0,0,0
Civilian6=6,1,13765,190461,175497,-1,0,0,0
Civilian7=6,1,12140,573716,312359,-1,0,0,0
Civilian8=6,1,13973,134321,186791,-1,0,0,0
Civilian9=6,1,12146,606065,320105,-1,0,0,0
Civilian10=6,1,12627,436744,425812,-1,0,0,0
Civilian11=6,1,14240,58171,398663,-1,0,0,0
Civilian12=6,1,13789,212202,123441,-1,0,0,0
Civilian13=6,1,13220,302174,554926,-1,0,0,0
Civilian14=6,1,12294,465919,334740,-1,0,0,0
Civilian15=6,1,14162,105123,70796,-1,0,0,0
Civilian16=6,1,13905,147645,453253,-1,0,0,0
Civilian17=6,1,13995,144298,97634,-1,0,0,0
Civilian18=6,1,13511,257466,639464,-1,0,0,0
Civilian19=6,1,11867,527326,575487,-1,0,0,0
Civilian20=6,1,12738,424745,67999,-1,0,0,0
Civilian21=6,1,13249,274159,494686,-1,0,0,0
Civilian22=6,1,12262,566603,528137,-1,0,0,0

Civilian23=6,1,14176,38545,651586,-1,0,0,0
Civilian24=6,1,12285,556885,401441,-1,0,0,0
Civilian25=6,1,11697,717438,402553,-1,0,0,0
Civilian26=6,1,14227,35908,382241,-1,0,0,0
Civilian27=6,1,12761,350387,40210,-1,0,0,0
Civilian28=6,1,13796,173600,93554,-1,0,0,0
Civilian29=6,1,13696,188171,330608,-1,0,0,0
Civilian30=6,1,12650,425344,461942,-1,0,0,0
Civilian31=6,1,12968,379383,183597,-1,0,0,0
Civilian32=6,1,14117,72114,370880,-1,0,0,0
Civilian33=6,1,12305,537778,333649,-1,0,0,0
Civilian34=6,1,12676,421747,391476,-1,0,0,0
Civilian35=6,1,14144,87364,126396,-1,0,0,0
Civilian36=6,1,13310,184480,404735,-1,0,0,0
Civilian37=6,1,11628,690666,599338,-1,0,0,0
Civilian38=6,1,13132,328014,130777,-1,0,0,0
Civilian39=6,1,14149,101099,122506,-1,0,0,0
Civilian40=6,1,13492,272339,37764,-1,0,0,0
Civilian41=6,1,14264,28058,281470,-1,0,0,0
Civilian42=6,1,13311,184480,413759,-1,0,0,0
Civilian43=6,1,11782,705533,291800,-1,0,0,0
Civilian44=6,1,12239,549460,657568,-1,0,0,0
Civilian45=6,1,11974,597363,235106,-1,0,0,0
Civilian46=6,1,11778,670110,301374,-1,0,0,0
Civilian47=6,1,13867,130578,567739,-1,0,0,0
Civilian48=6,1,12177,584484,280813,-1,0,0,0
Civilian49=6,1,13698,176705,330608,-1,0,0,0
Civilian50=6,1,13365,308015,382472,-1,0,0,0
Civilian51=6,1,13825,207253,44825,-1,0,0,0
Civilian52=6,1,12954,384830,277407,-1,0,0,0
Civilian53=6,1,13846,115097,554786,-1,0,0,0
Civilian54=6,1,12591,482899,37835,-1,0,0,0
Civilian55=6,1,12323,527864,295199,-1,0,0,0
Civilian56=6,1,11786,637657,115516,-1,0,0,0
Civilian57=6,1,13179,333145,101953,-1,0,0,0
Civilian58=6,1,12801,378573,688041,-1,0,0,0
Civilian59=6,1,12921,378819,382330,-1,0,0,0
Civilian60=6,1,13183,296305,612874,-1,0,0,0
Civilian61=6,1,11612,690759,614926,-1,0,0,0
Civilian62=6,1,12536,478457,425282,-1,0,0,0
Civilian63=6,1,11750,691738,250817,-1,0,0,0
Civilian64=6,1,13522,229075,530340,-1,0,0,0
Civilian65=6,1,12096,537544,437231,-1,0,0,0

Civilian66=6,1,12281,544775,384530,-1,0,0,0
Civilian67=6,1,14329,54638,34982,-1,0,0,0
Civilian68=6,1,11983,632766,234642,-1,0,0,0
Civilian69=6,1,14078,70999,492906,-1,0,0,0
Civilian70=6,1,12074,604062,487025,-1,0,0,0
Civilian71=6,1,11968,658116,387451,-1,0,0,0
AmbulanceTeam0=7,0,15323,529756,208045,-1,0,0,0
AmbulanceTeam1=7,0,16483,225963,301173,-1,0,0,0
AmbulanceTeam2=7,0,14899,664043,594435,-1,0,0,0
AmbulanceTeam3=7,0,15680,440552,168746,-1,0,0,0
AmbulanceTeam4=7,0,16433,217438,480503,-1,0,0,0
AmbulanceTeam5=7,0,14791,649331,106892,-1,0,0,0
FireBrigade0=8,0,16564,218401,580605,-1,0,0,0
FireBrigade1=8,0,14353,706702,247397,-1,0,0,0
FireBrigade2=8,0,15686,461777,83308,-1,0,0,0
FireBrigade3=8,0,16644,194050,261274,-1,0,0,0
FireBrigade4=8,0,16856,116816,190949,-1,0,0,0
FireBrigade5=8,0,14857,545420,556744,-1,0,0,0
FireBrigade6=8,0,15486,471313,427730,-1,0,0,0
FireBrigade7=8,0,16905,163225,42112,-1,0,0,0
FireBrigade8=8,0,14435,433816,677984,-1,0,0,0
FireBrigade9=8,0,14454,367244,459617,-1,0,0,0
PoliceForce0=9,0,16714,216108,50965,-1,0,0,0
PoliceForce1=9,0,17097,36789,462793,-1,0,0,0
PoliceForce2=9,0,16944,91931,571584,-1,0,0,0
PoliceForce3=9,0,15408,385384,659731,-1,0,0,0
PoliceForce4=9,0,15380,484385,702963,-1,0,0,0
PoliceForce5=9,0,16443,267543,491207,-1,0,0,0
PoliceForce6=9,0,15190,592568,107438,-1,0,0,0
PoliceForce7=9,0,15190,592566,107438,-1,0,0,0

[FirePoint]

FirePointNum=0

A.4 Configuration's file of the Ramdom Small's map

[MotionLessObject]

FireStationNum=1
PoliceOfficeNum=1
AmbulanceCenterNum=1
RefugeNum=2

FireStation0=3,1,5717,445844,201608,-1,0,0,0
PoliceOffice0=4,1,5610,319881,314616,-1,0,0,0
AmbulanceCenter0=2,1,5310,66040,237643,-1,0,0,0
Refuge0=5,1,6169,341826,410111,-1,0,0,0
Refuge1=5,1,5300,109454,186162,-1,0,0,0

[MoveObject]

CivilianNum=72
AmbulanceTeamNum=6
FireBrigadeNum=10
PoliceForceNum=8
Civilian0=6,1,5501,172249,308512,-1,0,0,0
Civilian1=6,1,5896,278137,356648,-1,0,0,0
Civilian2=6,1,5746,364907,234025,-1,0,0,0
Civilian3=6,1,5729,448368,169564,-1,0,0,0
Civilian4=6,1,5426,325671,133354,-1,0,0,0
Civilian5=6,1,5722,477872,171537,-1,0,0,0
Civilian6=6,1,6330,221363,528129,-1,0,0,0
Civilian7=6,1,5615,291092,311180,-1,0,0,0
Civilian8=6,1,6388,377274,510828,-1,0,0,0
Civilian9=6,1,5327,80452,214335,-1,0,0,0
Civilian10=6,1,6211,212206,500843,-1,0,0,0
Civilian11=6,1,5564,395252,185627,-1,0,0,0
Civilian12=6,1,5473,216940,226235,-1,0,0,0
Civilian13=6,1,5422,286078,125051,-1,0,0,0
Civilian14=6,1,5944,441896,378304,-1,0,0,0
Civilian15=6,1,5432,283034,174842,-1,0,0,0
Civilian16=6,1,5492,215074,252561,-1,0,0,0
Civilian17=6,1,5717,445844,201608,-1,0,0,0
Civilian18=6,1,6205,199478,510376,-1,0,0,0
Civilian19=6,1,5353,448642,124578,-1,0,0,0
Civilian20=6,1,5406,329897,190876,-1,0,0,0
Civilian21=6,1,6398,388645,515010,-1,0,0,0
Civilian22=6,1,5962,431592,437060,-1,0,0,0
Civilian23=6,1,6249,494208,368898,-1,0,0,0
Civilian24=6,1,5216,389646,40092,-1,0,0,0
Civilian25=6,1,5945,451283,423453,-1,0,0,0
Civilian26=6,1,5256,323855,63076,-1,0,0,0
Civilian27=6,1,5584,399451,163182,-1,0,0,0
Civilian28=6,1,6214,585330,330282,-1,0,0,0
Civilian29=6,1,5856,339703,292676,-1,0,0,0
Civilian30=6,1,5453,202749,237497,-1,0,0,0
Civilian31=6,1,6081,249660,401562,-1,0,0,0

Civilian32=6,1,5691,464300,197478,-1,0,0,0
Civilian33=6,1,5298,166327,154420,-1,0,0,0
Civilian34=6,1,5798,218916,372752,-1,0,0,0
Civilian35=6,1,6192,292392,430470,-1,0,0,0
Civilian36=6,1,5910,136637,416917,-1,0,0,0
Civilian37=6,1,5550,412701,185337,-1,0,0,0
Civilian38=6,1,5345,422201,112828,-1,0,0,0
Civilian39=6,1,6173,253041,481452,-1,0,0,0
Civilian40=6,1,6329,220185,557269,-1,0,0,0
Civilian41=6,1,5320,81969,230669,-1,0,0,0
Civilian42=6,1,5724,489482,170305,-1,0,0,0
Civilian43=6,1,6022,446387,271614,-1,0,0,0
Civilian44=6,1,5231,292274,106870,-1,0,0,0
Civilian45=6,1,6347,588010,365071,-1,0,0,0
Civilian46=6,1,5697,466353,227575,-1,0,0,0
Civilian47=6,1,5257,311223,58780,-1,0,0,0
Civilian48=6,1,5608,328124,325182,-1,0,0,0
Civilian49=6,1,6215,553200,332637,-1,0,0,0
Civilian50=6,1,5825,154064,346687,-1,0,0,0
Civilian51=6,1,6321,274316,538932,-1,0,0,0
Civilian52=6,1,5421,294670,122092,-1,0,0,0
Civilian53=6,1,6087,277817,417131,-1,0,0,0
Civilian54=6,1,5382,366181,119287,-1,0,0,0
Civilian55=6,1,6207,220880,514527,-1,0,0,0
Civilian56=6,1,5650,257209,290449,-1,0,0,0
Civilian57=6,1,6356,597255,359295,-1,0,0,0
Civilian58=6,1,6002,426524,271973,-1,0,0,0
Civilian59=6,1,6254,385890,458058,-1,0,0,0
Civilian60=6,1,6391,359441,511637,-1,0,0,0
Civilian61=6,1,5722,477872,171537,-1,0,0,0
Civilian62=6,1,5573,372878,126613,-1,0,0,0
Civilian63=6,1,5805,231076,362096,-1,0,0,0
Civilian64=6,1,6331,221363,537514,-1,0,0,0
Civilian65=6,1,6008,475442,343197,-1,0,0,0
Civilian66=6,1,6002,426520,271973,-1,0,0,0
Civilian67=6,1,6254,385894,458058,-1,0,0,0
Civilian68=6,1,6391,359443,511637,-1,0,0,0
Civilian69=6,1,5722,477874,171537,-1,0,0,0
Civilian70=6,1,5573,372876,126613,-1,0,0,0
Civilian71=6,1,5805,231078,362096,-1,0,0,0
AmbulanceTeam0=7,0,6477,277408,305185,-1,0,0,0
AmbulanceTeam1=7,0,6592,310248,42709,-1,0,0,0
AmbulanceTeam2=7,0,6938,322652,335089,-1,0,0,0

AmbulanceTeam3=7,0,7183,401687,282282,-1,0,0,0
AmbulanceTeam4=7,0,7248,465430,359959,-1,0,0,0
AmbulanceTeam5=7,0,6590,320968,95678,-1,0,0,0
FireBrigade0=8,0,6670,19994,279923,-1,0,0,0
FireBrigade1=8,0,7194,417529,244900,-1,0,0,0
FireBrigade2=8,0,7161,373867,300870,-1,0,0,0
FireBrigade3=8,0,7207,292147,327244,-1,0,0,0
FireBrigade4=8,0,6552,384582,53175,-1,0,0,0
FireBrigade5=8,0,6440,41575,222223,-1,0,0,0
FireBrigade6=8,0,7349,374083,350333,-1,0,0,0
FireBrigade7=8,0,6668,17823,276674,-1,0,0,0
FireBrigade8=8,0,6659,102414,206171,-1,0,0,0
FireBrigade9=8,0,6932,290559,176131,-1,0,0,0
PoliceForce0=9,0,7200,253700,321025,-1,0,0,0
PoliceForce1=9,0,6814,257807,185850,-1,0,0,0
PoliceForce2=9,0,7662,479031,455155,-1,0,0,0
PoliceForce3=9,0,7444,399999,395308,-1,0,0,0
PoliceForce4=9,0,7547,490818,354589,-1,0,0,0
PoliceForce5=9,0,7538,525228,303645,-1,0,0,0
PoliceForce6=9,0,6874,79607,251331,-1,0,0,0
PoliceForce7=9,0,7220,128473,404696,-1,0,0,0

[FirePoint]

FirePointNum=0

Bibliography

[1] http://www.robocuprescue.org/.

[2] Robocup official site. [online]. In *http://www.robocup.org*, 2010.

[3] N.M. Adams, M. Field, E. Gelenbe, D.J. Hand, N.R. Jennings, D.S. Leslie, D. Nicholson, S.D. Ramchurn, and A. Rogers. Intelligent agents for disaster management. In *IARP/EURON Workshop on Robotics for Risky Interventions and Environmental Surveillance (RISE)*, 2008.

[4] O. Aghazadeh, M.A. Sharbafi, and A.T. Haghighat. Implementing parametric reinforcement learning in robocup rescue simulation. In *RoboCup 2007: Robot Soccer World Cup XI*, pages 409–416. Springer Berlin / Heidelberg, 2008.

[5] A. Ahmed, A. Patel, T. Brown, M. Ham, M.W. Jang, and G. Agha. Task assignment for a physical agent team via a dynamic forward/reverse auction mechanism. In *Proceedings of The International Conference on Integration of Knowledge Intensive Multi-Agent Systems*, pages 311–317. IEEE, 2005.

[6] H.L. Akin, E. Dogrultan, T. Mericli, and E. Ozkucur. Roboakut 2009 rescue simulation league agent team description. In *RoboCup Rescue Competition, Team description papers*, 2009.

[7] H.L. Akin and E. Ozkucur. Rescue simulation league team description - roboakut team. In *Team Description Papers*. RoboCup Rescue, 2008.

[8] A. Aliakbarian, M. Hamidi, N. Maleki, M. Shahmoradi, H. Tavakoli, S. Ziaee, B. Shahgholi, and N. Movahhedinia. Team description zendehrood (iran). In *RoboCup Rescue Rescue Simulation League*, 2008.

[9] M. Andersson and T. Sandholm. Contract type sequencing for reallocative negotiation. In *Proceedings of the The 20th International Conference on Distributed Computing Systems ICDCS 2000*, page 154, 2000.

[10] H. Andrew, A.L. Delbecq, and R.Jr. Koenig. Determinants of coordination modes within organizations. *American Sociological Review*, 41(2):322–338, 1976.

[11] D. Ben-Ami and O. Shehory. Evaluation of distributed and centralized agent location mechanisms. *Lecture Notes in Computer Science*, 2446:43–72, 2002.

[12] M. Berhault, H. Huang, P. Keskinocak, S. Koenig, W. Elmaghraby, P. Griffin, and A. Kleywegt. Robot exploration with combinatorial auctions. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 1957–1962. IEEE, 2003.

[13] D.P. Bertsekas and D.A. Castanon. A forward/reverse auction algorithm for asymmetric assignment problems. Technical report, Technical Report Lids-P-2159, MIT, 1993.

[14] P. Brucker. *Scheduling Algorithms*. Springer, 2001.

[15] T. Candale and S. Sen. Multi-dimensional bid improvement algorithm for simultaneous auctions. In *Proceedings of the 20th international joint conference on Artifical intelligence*, pages 1215–1220. Morgan Kaufmann Publishers Inc. San Francisco, CA, USA, 2007.

[16] B. Chaib-draa and P. Moulin. Architecture for distributed artificial intelligent systems. *IEEE Proceedings.*, 15(7):64–69, 1987.

[17] A. Chapman, R.A. Micillo, R. Kota, and N.R. Jennings. Decentralised dynamic task allocation: A practical game-theoretic approach. In *The Eighth International Conference on Autonomous Agents and Multiagent Systems, Volume 2*, pages 915–922. Springer Berlin / Heidelberg, 2009.

[18] Y. Chevaleyre, P.E. Dunne, U. Endriss, J. Lang, M. Lemaître, N. Maudet, J. Padget, S. Phelps, J.A. Rodríguez-Aguilar, and P. Sousa. Issues in multiagent resource allocation. *Informatica*, 30:3–31, 2006.

[19] H.K.H. Chow, K.L. Choy, and W.B. Lee. A dynamic logistics process knowledge-based system - an rfid multi-agent approach. *Knowledge-Based Systems*, 20(4):357–372, 2007.

[20] J. Collins and M. Gini. Scheduling tasks using combinatorial auctions: The magnet approach. *Handbooks in Information Systems*, 3(4):263–293, 2009.

[21] R. Conway, W. Maxwell, and L. Miller. *Theory of Scheduling.* Addison-Wesley Publishing Company, 1967.

[22] W. Cushing and S. Kambhampati. Replanning: A new perspective. In *Proceedings of The International Conference on Automated Planning and Scheduling*. ICAPS, 2005.

[23] S. Damiani, G. Verfaillie, and M.C. Charmeau. An earth watching satellite constellation: how to manage a team of watching agents with limited communications. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 455–462, 2005.

[24] M. B. Dias, R. M. Zlot, N. Kalra, and A. T. Stentz. Market-based multirobot coordination: A survey and analysis. Technical report, Robotics Institute, Carnegie Mellon University, 2005.

[25] M.B. Dias, R. Zlot, N. Kalra, and A. Stentz. Marketbased multirobot coordination: a survey and analysis. In *Proceedings of the IEEE (Special Issue on Multirobot Coordination)*, pages 1257–1270. IEEE, 2006.

[26] S. Dobzinski and N. Nisan. Mechanisms for multi-unit auctions. In *Proceedings of the 8th ACM conference on Electronic commerce*, pages 346–351. ACM New York, NY, USA, 2007.

[27] E.H. Durfee. *Weiss, G., editor, Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence, chapter 3, pages 121-164*. Addison-Wesley Publishing Company, 1999.

[28] A. Farinelli, L. Iocchi, and D. Nardi. Multi robot systems: A classification based on coordination. *In IEEE Transactions on Systems, Man and Cybernetics*, 34(5):2015–2028, 2004.

[29] P.Jr. Ferreira, F.S. Boffo, and A.L. Bazzan. Using swarm-gap for distributed task allocation in complex scenarios. *Lecture Notes in Computer Science, Massively Multi-Agent Technology*, 5043:107–121, 2008.

[30] F. Fiedrich and P. Burghardt. Agent-based systems for disaster management. *Management Science journal*, 50(3):41–42, 2007.

[31] B.P. Gerkey. *On Multi-Robot Task Allocation*. PhD thesis, Computer Science Department, University of Southern California, volume CRES-03-012, 2003.

[32] B.S. Ghahfarokhi, H. Shahbazi, M. Kazemifard, and K. Zamanifar. Evolving fuzzy neural network based fire planning in rescue firebrigade agents. *Simulation Series*, 38(4):74–82, 2006.

[33] M. Ghijsen, W. Jansweijer, and B. Wielinga. The effect of task and environment factors on multi-agent coordination and reorganization. In *Proceedings of the 6th international Joint Conference on Autonomous Agents and Multiagent Systems*. ACM New York, NY, USA, 2007.

[34] A. Giovannucci, J.A. Rodríguez-Aguilar, A. Reyes, F.X. Noria, and J. Cerquides. Enacting agent-based services for automated procurement. *Engineering Applications of Artificial Intelligence*, 21(2):183–199, 2008.

[35] J. Habibi, M. Ahmadi, A. Nouri, M. Sayyadian, and M. Nevisi. Implementing heterogeneous agents in dynamic environments, a case study in robocup rescue. *Lecture Notes in Computer Science, Multiagent System Technologies*, 2831, 2004.

[36] J. Habibi, M.R. Ghodsi, H.R. Vaezi, M. Valipour, S. Aliari, and N. Hazar. Robocup rescue simulation league team description - impossibles. In *Team Description Papers*. RoboCup Rescue, 2006.

[37] Y. Hadas and A. Ceder. Multiagent approach for public transit system based on flexible routes. *Transportation Research Record: Journal of the Transportation Research Board*, 2063:89–96, 2008.

[38] J.H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.

[39] B. Horling and V. Lesser. A survey of multi-agent organizational paradigms. Technical report, Computer Science Technical Report 04–45, University of Massachusetts, 2004.

[40] J.R. Jackson. Scheduling a production line to minimize maximum tardiness. Technical report, Management science research project, Univ. of California, Los Angeles, 1955.

[41] F. Jiang, Y. Liu, X. Zhang, T. Zhu, and J.P. Minwu. Team description csuyunlu (china). In *RoboCup Rescue Rescue Simulation League*, 2008.

[42] C. Jonquet, P. Dugenie, and S.A. Cerri. Service-based integration of grid and multi-agent systems models. *Service-Oriented Computing: Agents, Semantics, and Engineering*, 5006:56–68, 2008.

[43] H. Kitano, S. Tadokor, H. Noda, I. Matsubara, T. Takhasi, A. Shinjou, and S. Shimada. Robocup rescue: Search and rescue for large scale disasters as a domain for multi-agent research. In *Proceedings of the IEEE Conference on Systems, Man, and Cybernetics*, 1999.

[44] H. Kitano and S. Tadokoro. Robocup rescue: a grand challenge for multiagent and intelligent systems. *AI Magazine*, 22(1):39–52, 2001.

[45] A. Kleiner, M. Brenner, T. Brauer, and C. Dornhege. Successful search and rescue in simulated disaster areas. In *Proceedings of the RoboCup 2005*, pages 323–334, 2006.

[46] S. Koenig, C. Tovey, X. Zheng, and I. Sungur. Sequential bundle-bid single-sale auction algorithms for decentralized control. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1359–1365. IJCAI, 2007.

[47] R.V.D Krogt and M.D. Weerdt. The two faces of plan repair. In *Proceedings of the Sixteenth Belgium-Netherlands Conference of Artificial Intelligence*, pages 147–154. BNAIC, 2004.

[48] M.G. Lagoudakis, M. Berhault, S. Koenig, P. Keskinocak, and A.J. Kleywegt. Simple auctions with performance guarantees for multi-robot task allocation. In *International Conference on Intelligent Robots and Systems*, pages 698–705. IEEE, 2005.

[49] J.H. Lee and C.O. Kim. Multi-agent systems applications in manufacturing systems and supply chain management: a review paper. *International Journal of Production Research*, 46(1):233–265, 2008.

[50] V.R. Lesser. Reflections of the nature of multi-agent coordination and its implications for an agent architecture. *Autonomous Agents and Multi-Agents Systems*, 1:89–111, 1998.

[51] V.R. Lesser. Cooperative multiagent systems: A personal view of the state of the art. *IEEE transactions on knowledge and data engineering*, 11(1):133–142, 1999.

[52] E. Letier and A.V. Lamsweerde. Agent-based tactics for goal-oriented requirements elaboration. In *Procedeedings of 24th International Conference on Software Engineering ICSE 02*, pages 83–93. ACM New York, USA, 2002.

[53] L. Lin, W. Lei, Z.Q. Zheng, and Z. Sun. A learning market based layered multi-robot architecture. In *Proceedings of the 2004 IEEE International Conference on Robotics and Automation*, pages 3417–3422. IEEE, 2004.

[54] K.M. Lochner and M.P. Wellman. *Auctions, Market Mechanisms and Their Applications*, chapter Information Feedback and Efficiency in Multiattribute Double Auctions, pages 26–39. Springer Berlin Heidelberg, 2009.

[55] B. López, S. Suárez, and J.L. De la Rosa. Task allocation in rescue operations using combinatorial auctions. *Artificial Intelligence Research and Development*, pages 233–243, 2003.

[56] M. Luck, P. McBurney, O. Shehory, and S. Willmott. *Agent Technology: Computing as Interaction (A Roadmap for Agent Based Computing)*. AgentLink, 2005.

[57] A.D. Mali and S. Kambhampati. Distributed planning. In *The Encyclopedia of Distributed Computing*. Kluwer Academic Publishers, 1999.

[58] M.G. Mehdiabad, B. Hashemi, and A. Sharifrazavian. Rescue simulation league team description. s.o.s. In *Team Description Papers*. RoboCup Rescue, 2007.

[59] I. Mezei, V. Malbasa, and I. Stojmenovic. Auction aggregation protocols for wireless robot-robot coordination. *Lecture Notes in Computer Science*, 5793:180–193, 2009.

[60] R. Micalizio and P. Torasso. Supervision and diagnosis of joint actions in multi-agent plans. In *Proceedings of the 7th international Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1375–1378. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 2008.

[61] H. Mintzberg. *The Structuring of Organizations*. Englewoods Cliffs, 1979.

[62] R. Mirhassani, S.N. Ardabili, Z. Shariati, F. Torkashvand, M. Yousefi, and M.T. Ghinani. Rescue simulation league team. poseidon (iran). In *Team Description Papers*. RoboCup Rescue, 2007.

[63] L. Monostori, J. Vncza, and S.R.T. Kumara. Agent-based systems for manufacturing. *Annals of the CIRP*, 55(2):672–720, 2006.

[64] K. Myers, P. Berry, J. Blythe, K. Conley, M. Gervasio, D. McGuinness, D. Morley, A. Pfeffer, M. Pollack, and M. Tambe. An intelligent personal assistant for task and time management. *AI Magazine*, 28(2):47–61, 2007.

[65] R. Nair, T. Ito, M. Tambe, and S. Marsella. Task allocation in the rescue simulation domain: A short note. *Lecture Notes in Computer Science*, 2377:1–22, 2002.

[66] R. Nair, M. Tambe, and S. Marsella. Team formation for reformation in multiagent domains like robocup rescue. In *Kaminka, G., Lima, P., and Roja, R., editors, Proceedings of RoboCup-2002 International Symposium, Lecture Notes in Computer Science. Springer Verlag*, 2003.

[67] Y. Narahari and P. Dayama. Combinatorial auctions for electronic business. *Sadhana (Special Issue on Electronic Commerce and Electronic Business)*, 30(2-3):179–211, 2005.

[68] E. Nazemi, M.A. Fardad, and M.M. Saboorian. Message management system in sbce_saviour team. In *Proceedings of Robocup 2004: The 8th RoboCup International Symposium*, 2004.

[69] J. Niu, K. Cai, S. Parsons, and E. Sklar. Reducing price fluctuation in continuous double auctions through pricing policy and shout improvement. In *Proceedings of the Fifth international Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1143 – 1150. ACM New York, NY, USA, 2006.

[70] I. Noda. Rescue simulation and location-based communication model. In *Proceedings of SCI-2001*, 2001.

[71] L. Padgham and W. Winikoff. *Developing Intelligent Agent Systems: A Practical Guide*. JohnWiley and sons, Ltd, 2004.

[72] C. Papadimitriou and J.N. Tsisiklis. The complexity of markov decision processes. *Mathematics of Operations Research*, 12(3):441–450, 1987.

[73] S. Paquet, N. Bernier, and B. Chaib-draa. Comparison of different coordination strategies for the robocup rescue simulation. *Innovations in Applied Artificial Intelligence*, 3029(4):987–996, 2004.

[74] S. Paquet, N. Bernier, and B. Chaib-draa. Multiagent systems viewed as distributed scheduling systems: Methodology and experiments. In *Proceedings of the Eighteenth Canadian Conference on Artificial Intelligence, Victoria, Canada*, 2005.

[75] S. Paquet, L. Tobin, and B. Chaib-draa. An online pomdp algorithm for complex multiagent environments. In *Proceedings of the Fourth international Joint Conference on Autonomous Agents and Multiagent Systems*, pages 970–977. ACM New York, NY, USA, 2005.

[76] M. Piunti, C. Castelfranchi, and R. Falcone. Expectations driven approach for situated, goal-directed agents. In *Procedeedings of 8th AI*IA/TABOO Joint Work. "From Objects to Agents": Agents and Industry*, pages 104–111. In Baldoni, M., et al., eds. Seneca Edizioni Torino, 2007.

[77] E.L. Quarantelli. Disaster crisis management: a summary of research findings. *Journal of Management Studies*, 25(4):373–385, 1988.

[78] A. Radmand, E. Nazemi, and M. Goodarzi. Integrated genetic algorithmic and fuzzy logic approach for decision making of police force agents in rescue simulation environment. In *RoboCup 2009: Robot Soccer World Cup XIII*, pages 288–295. Springer Berlin / Heidelberg, 2009.

[79] S.D. Ramchurn, D. Huynh, and N.R. Jennings. Trust in multi-agent systems. *The Knowledge Engineering Review*, 19:1(7):1–25, 2004.

[80] S.D. Ramchurn, A. Rogers, K. Macarthur, A. Farinelli, P. Vytelingum, I. Vetsikas, and N.R. Jennings. Agent-based coordination technologies in disaster management. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems: demo papers*, pages 1651–1652, 2008.

[81] A.S. Rao and M.P. Georgeff. Modelling rational agents within a bdi-architecture. In *Proceedings of Knowledge Representation and Reasoning (KRR-91) Conference*, page 473484. Morgan Kaufmann Publishers: San Mateo, CA, 1991.

[82] J.S. Rosenschein and G. Zlotkin. *Rules of Encounter: Designing Conventions for Automated Negotiation among computers*. The MIT Press: Cambridge, MA, 1994.

[83] S.J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach, second ed.* Prentice Hall, Englewood Cliffs, NJ, 2003.

[84] T. Sandholm. Contract types for satisficing task allocation: theoretical results. In *Proceedings AAAI Spring Symposium Series: Satisficing Models*, pages 68–75. Stanford University, CA, 1998.

[85] T. Sandholm. An algorithm for optimal winner determination in combinatorial auctions. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 542–547. Morgan Kaufmann Publishers Inc. San Francisco, CA, USA, 1999.

[86] T. Sandholm. Algorithm for optimal winner determination in combinatorial auctions. *Artificial Intelligence*, 135(1-2):1–54, 2002.

[87] T. Sandholm. Expressive commerce and its application to sourcing: How we conducted $35 billion of generalized combinatorial auctions. *AI Magazine*, 28(3):45–58, 2007.

[88] T. Sandholm. *Computing in mechanism design*. The new palgrave dictionary of economics, second edition, 2008.

[89] T. Sandholm and S. Suri. Improved algorithms for optimal winner determination in combinatorial auctions and generalizations. In *National Conference on Artificial Intelligence (AAAI)*, pages 90–97. AAAI Press / The MIT Press, 2000.

[90] T. Sandholm, S. Suri, A. Gilpin, and D. Levine. Winner determination in combinatorial auction generalizations. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1*, pages 69–76. ACM New York, NY, USA, 2002.

[91] T. Sandholm, S. Suri, A. Gilpin, and D. Levine. Cabob: a fast optimal algorithm for winner determination in combinatorial auctions. *Management Science journal*, 51(3):374–391, 2005.

[92] P. Scerri, A. Farinelli, S. Okamoto, and M. Tambe. Allocating tasks in extreme teams. In *Proceedings of the Fourth international Joint Conference on Autonomous Agents and Multiagent Systems*, pages 727–734. ACM New York, NY, USA, 2005.

[93] P. Scerri, P.J. Modi, W.M. Shen, and M. Tambe. Applying constraint reasoning to real-world distributed task allocation. In *Proceedings of the Workshop on Distributed Constraint Reasoning at AAMAS'02*, 2002.

[94] M.N. Sedaghat, L.P. Nejad, S. Iravanian, and E. Rafiee. Task allocation for the police force agents in robocup rescue simulation. *Lecture Notes in Computer Science*, 4020:656–664, 2006.

[95] M.N. Sedaghati, N. Gholami, and E. Rafiee. Rescue simulation team description - caspian team. In *Proceedings of Robocup 2004: The 8th RoboCup International Symposium*, 2004.

[96] B. Sellner and R. Simmons. Towards proactive replanning for multi-robot teams. In *Proceedings of 5th International Workshop on Planning and Scheduling in Space*, 2006.

[97] M.A. Sharbafi, O. AmirGhiasvand, S.A. Ramandi, O. Aghazadeh, R. Mirsharifi, and M. Shirzadi. Rescue simulation league team description - mrl team. In *Team Description Papers*. RoboCup Rescue, 2008.

[98] D.H. Shih, S.Y. Huang, and B. Lin. Linking secure reverse auction with web service. *International Journal of Services and Standards*, 2(1):15–31, 2006.

[99] C. Skinner, J. Teutenberg, and G. Cleveland. The black sheep team description. In *Proceedings of RoboCup 2004: The 8th RoboCup International Symposium*, 2004.

[100] D.E. Smith, J. Frank, and A.K. Jonsson. Coordination of multiple agents in distributed manufacturing scheduling. In *The Fifth International Conference on Artificial Intelligence Planning and Scheduling*, pages 61–94, 2000.

[101] S.F. Smith. Reactive scheduling systems. In *Intelligent Scheduling Systems*, pages 155–192, 1994.

[102] S. Suárez and B. López. Reverse combinatorial auctions for allocating resources in rescue scenario. In *Proceedings of Workshop on Constraint Satisfaction Techniques for Planning and Scheduling Problems ICAPS*, pages 62–66, 2006.

[103] S. Suárez, B. López, and J.L. De la Rosa. Co-operation strategies for strengthening civil agents lives in the robocup rescue simulator scenario. In *Proceedings of The First International Workshop on Synthetic Simulation and Robotics to Mitigate Earthquake Disaster*. RoboCup Rescue, Padova, Italy, 2003.

[104] S. Suárez, B. López, and J.L. De la Rosa. Multicriteria decision method for resource distribution in a large-scale disaster. In *X Conferencia de la Asociación Española para la Inteligencia Artificial CAEPIA, volumen II*, pages 261–264, 2003.

[105] S. Suárez, C. Quintero, and J.L. De la Rosa. Improving tasks allocation and coordination in a rescue scenario. In *Proceedings European Control Conference ECC'07*, pages 1498 – 1503, 2007.

[106] S. Suárez, C. Quintero, and J.L. De la Rosa. A real time approach for task allocation in a disaster scenario. *Advances in Practical Applications of Agents and Multiagent Systems. Advances in Soft Computing*, 70/2010:157–162, 2010.

[107] P.B. Sujit and R. Beard. Distributed sequential auctions for multiple uav task allocation. In *American Control Conference ACC'07*, pages 3955–3960. IEEE, 2007.

[108] S. Tadakoro, H. Kitano, T. Takahashi, I. Noda, H. Matsubara, A. Shinjoh, T. Koto, I. Takeuchi, H. Takahashi, F. Matsuno, M. Hatayama, J. Nobe, and S. Shimada. The robocup rescue project: A robotic approach to the disaster mitigation problem. *Proceedings of the 2000 IEEE International Conference on Robotics and Automation*, 4:4089–4094, 2000.

[109] T. Takahashi, S. Tadokoro, M. Ohta, and N. Ito. Agent based approach in disaster rescue simulation - from test-bed of multiagent system to practical application.replanning: A new perspective. In *RoboCup 2001, volume 2377 of Lecture Notes in Artificial Intelligence*, pages 102–111. Springer-Verlag, 2002.

[110] R. Tolksdorf. Models of coordination. *Lecture Notes in Computer Science. Engineering Societies in the Agents World*, 1972:78–92, 2000.

[111] K. Tumer and A. Agogino. Improving air traffic management with a learning multiagent system. *Intelligent Systems*, 24(1):18–22, 2009.

[112] S.D. Vries and R. Vohra. Combinatorial auctions: A survey. *Inform Journal on Computing*, 15(3):284–309, 2003.

[113] P. Vytelinguma, D. Cliffb, and N.R. Jennings. Strategic bidding in continuous double auctions. *Artificial Intelligence*, 172(14):1700–1729, 2008.

[114] M.D. Weerdt, A. Mors, and C. Witteveen. Multi-agent planning: An introduction to planning and coordination. In *Handouts of the European Agent Summer School*, pages 1–32, 2005.

[115] G. Weiss. *Multiagent System: A modern approach to Distributed AI*. MIT Press, 1999.

[116] M. Wooldridge. *An Introduction to MultiAgent Systems*. John Wiley and Sons Ltd., Second edition, 2009.

[117] L. Xing and Z. Lan. A grid resource allocation method based on iterative combinatorial auctions. *International Conference on Information Technology and Computer Science*, 2:322–325, 2009.

[118] R. Xiong, Y. Tao, Y. Wang, S. Zhong, W. Li, J. Zhang, B. Wang, and Y. Tian. Team description zjubase (china). In *RoboCup Rescue Rescue Simulation League*, 2008.

[119] L. Yang and M. Yasser. Multi-agent resource allocation for modeling construction processes. In *Proceedings of the 40th Conference on Winter Simulation*, pages 2361–2369, 2008.

[120] M. Yokoo, E.H. Durfee, T. Ishida, and K. Kuwabara. The distributed constraint satisfaction problem: Formalization and algorithms. *Knowledge and Data Engineering*, 10(5):673–685, 1998.

[121] H. Zafar, V. Lesser, D. Corkill, and D. Ganesan. Using organization knowledge to improve routing performance in wireless multi-agent networks. In *Proceedings of the 7th international Joint Conference on Autonomous Agents and Multiagent Systems*, volume 2, pages 821–828, 2008.

[122] J.F. Zhang, X.T. Nguyen, and R. Kowalczyk. Graph-based multiagent replanning algorithm. In *Proceedings of the 6th international Joint Conference on Autonomous Agents and Multiagent Systems*. ACM New York, NY, USA, 2007.

[123] R. Zhou and E.A. Hansen. Breadth-first heuristic search. *Artificial Intelligence Journal*, 170(4-5):385–408, 2006.

[124] R.M. Zlot. *An Auction-Based Approach to Complex Task Allocation for Multirobot Teams*. PhD thesis, The Robotics Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, 2006.

[125] E. Zurel and N. Nisan. An efficient approximate allocation algorithm for combinatorial auctions. In *Proceedings of ACM Conference on Electronic Commerce (EC-2001)*, pages 125–136. ACM New York, NY, USA, 2001.