Chapter 6

Optimization

6.1 Introduction

Up to this point, this thesis has shown a methodology that allows modeling the complexity of the constraints present in the industrial cases (chapter 4) and has also presented how to build schedules (chapter 5) either from an initial static situation or from an online environment.

The methodology for building schedules described in the previous chapter does not necessarily lead to (except some particular cases) an optimal solution as the greedy heuristics used provide sub-optimal solutions. This chapter will deal with several optimization procedures that can improve the solutions given by the procedures described in the previous chapter.

Two different approaches have been attempted: stochastic methods and mathematical programming. This chapter explains these methods and the results obtained.

6.2 Stochastic methods

In section 2.2.1 (page 11) different works concerning heuristic/stochastic methods applied to the solution of the scheduling problem were reviewed. In this thesis three different approaches have been used: simulated annealing, mixed stochastic enumerative search and genetic algorithms.

6.2.1 Simulated Annealing

6.2.1.1 Introduction

The sequencing of jobs in multi-product batch chemical plants can be seen as a blind ATSP, for which the changeover matrix is not available. This is mainly due to the same structure of operations involved in each job, which usually requires to simulate the scheduling of each pair of consecutive batches to obtain the corresponding changeover time (including clean-up times). Moreover, this situation worsens when such changeover times depend not only in the pair involved but also on other jobs in the series. This is the case when jobs do not

share all the same equipment units or they consume the same limited general utilities (steam, manpower, etc).

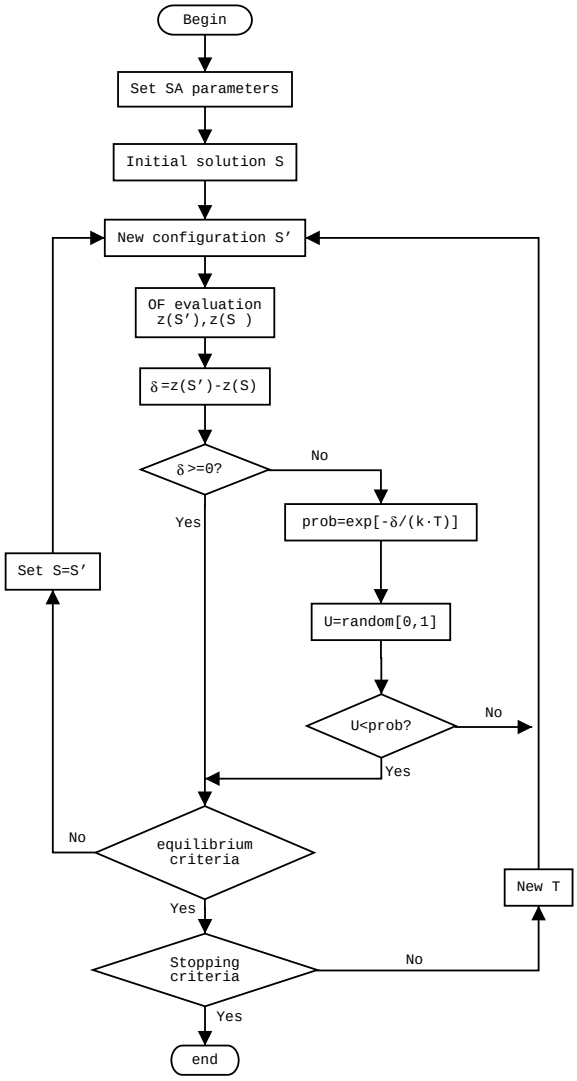

Figure 6.1: SA algorithm

Hence, Simulated Annealing has shown to be a good option for solving this kind of blind ATSP, since it is based on the simulation of each one of the sequences proposed so that they can be validated, evaluated and eventually accepted. Ku and Karimi (1991) first applied SA to multi-product problems (unconstrained job sequences). More recently, Graells et al. (1996)

used SA for solving a kind of multipurpose problems (constrained mini-job sequences). Other authors also applied SA to the scheduling problem as shown in section 2.2.1.

6.2.1.2 Methodology

The Simulated Annealing algorithm (see figure 6.1) is a probabilistic method for combinatorial optimization problems based on ideas from statistical mechanisms. It allows the improvement of an initial schedule (generated using the rule-based methods described in chapter 5) exploring a defined neighborhood and trying to improve a configurable objective function.

The Simulated Annealing method has an analogy with thermodynamics, specifically with the way metals cool and anneal. At each temperature T, the system is allowed to reach thermal equilibrium and should follow Markovian moves, where the next move is only dependent on the current position and not on the previous ones.

The state of minimum energy for the system would correspond to the optimal solution in a mathematical optimization problem. A metastable or amorphous state of higher energy reached due to a quick refrigeration would correspond, in the mathematical sense, to a suboptimal solution.

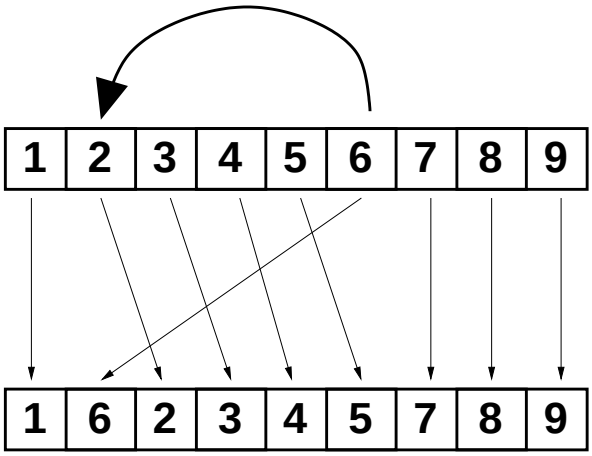

Figure 6.2: Neighborhood move used in SA

6.2.1.3 Application to the scheduling problem

In order to apply the simulated annealing to the scheduling problem within the framework presented in this thesis two aspects have to be noted:

- The neighborhood mode
- The feasibility check of the solutions.

The neighborhood move chosen in this thesis is sequence based as it is shown in figure 6.2. In this move, the batch in the position i is inserted in the position j of the sequence. Given a sequence of n batches, there are $n(n-1)$ possible moves (although leading only to $(n-1)^2$ different sequences).

The original SA algorithm is modified in order to check the feasibility of the balance and to allow assignment changes (see figure 6.3).

Every new sequence goes through a set of assignment rules like the ones described in section 5.2. The assignment rules used are:

- HPU (Highest Priority Unit): The unit with the highest priority that can process the same batch size is used.
- FU (First Unit): The first alternative unit defined in the recipe (lowest ID) that can process the same batch size is used.
- LUU (Less Used Unit): The unit which has been chosen less times in the precedent batches that can process the same batch size is used.
- MAU (More Available Unit): The unit with the most available time (the unit that end earliest in the previous batches) that can process the same batch size is chosen.
- SPTU (Shortest Processing Time Unit): the unit that is capable to perform the stage in less time for the same batch size is used.
- AUA (Already Used Assignment): If one assignment has been used with the same recipe and can process the same batch size , then it is given the highest priority.
- RU (Random Unit): One of the available units that can process the same batch size is chosen randomly.

Note that there is an important difference between the assignment rules described in section 5.2 and this set of rules: there is another constraint to take into account, the batch size. The number of possible candidates for the new assignment is reduced depending on the batch size, therefore it is insured that the batch size and the number of batches remains constant.

Additionally to the assignment, there is also a feasibility check of the sequence. The new sequence should meet all the balance constraints indicated in section 5.2.2 (page 79). If the sequence does not accomplish the balance constraints then it is infeasible and a new sequence is generated.

6.2.2 Mixed stochastic enumerative search

6.2.2.1 Introduction

Despite its many advantages, SA has some remarkable drawbacks, the main one being a technique too cautious and conservative. The key idea of SA is to allow up hill moves just in case that a local optimum is reached. However, up hill moves are still allowed even if this local optimum is not reached simply because such a situation is never detected by SA algorithms. Hence, experience shows that, when the probability of falling trapped in a local optimum is

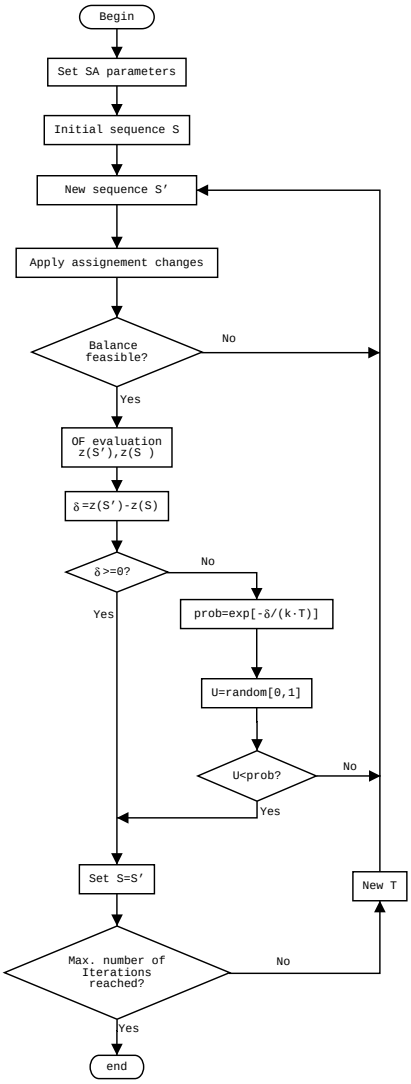

Figure 6.3: Modified SA algorithm applied to scheduling

very low, the most practical option and the one leading to the best results is often down hill random search (zero temperature). This is the case, for example, when very small computing time is available for improving a bad starting point. It is not until the situation becomes riskier (as better solutions are attained) that the need for higher temperatures allowing up hill moves seems to arise.

Several approaches have been proposed to accelerate the annealing procedure. The Non-Equilibrium Simulated Annealing (NESA)(Cardoso et al., 1994) addresses specifically the excessive caution of SA but still from a probabilistic point of view and assuming a temperature decreasing scheme. In the NESA approach "thermal" equilibrium is not necessarily attained at each contraction of the "temperature" control parameter, since the cooling schedule is enforced as soon as an improved solution is obtained.

However, the point is still why to allow backward moving when the search is progressing in the desired direction. Solution to this problem could be a combined "cooling/heating" scheme as proposed by Dowsland (1994). However, the problem is still when start heating and how to heat (heating schedule). In any case, such solution approaches give up the original simplicity of SA and better performances are attempted at the expense of increased strategy complexity.

On the other hand, this should be considered in addition to the parameter-tuning problem. The determination of a good cooling scheme (Laarhoven and Aarts, 1987), which is based on previous sampling, may be not affordable in many common situations that require practical solutions within a limited computing time for the every day changing problem (model changes such as parameters, constraints, objective function, etc.).

6.2.2.2 Local and global optima

Global optimum definition is absolute: it is the best solution among all possible solutions. The existence of local optima, however, is a consequence of the limitations of the used optimization techniques and depends on the search procedure employed and the step-size adopted. A local optimum is the best solution among all possible neighborhood solutions, attainable from a certain starting point given a search technique and a step-size. Hence, the change of the search procedure may likely result in the change of the local optima obtained for the problem considered (Fig. 6.4).

Thus, the strategy presented in this thesis for the solving of the scheduling problem is based on stochastic search, but considers the exhaustive search of neighborhood at each step in order to determine if progress is no longer possible with the current searching procedure (local optimum). When such a situation is detected, this procedure (any procedure) must obviously be replaced.

6.2.2.3 Methodology

The step-size adopted for the Traveling Salesman Problem (TSP) (analog to finding the best sequence for an specific set of batches) is usually a change in the sequence defining the solution, which is given by the permutation of two of its elements or by the reallocation of one of the elements in the sequence to another position. The latter is the one adopted in this work.

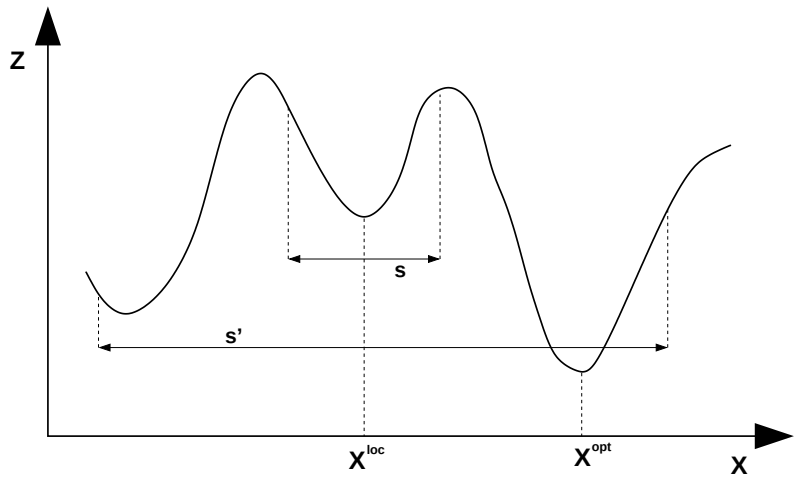

Figure 6.4: Local optima are consequence of the limitations of the search procedure and the step-size (s) employed.

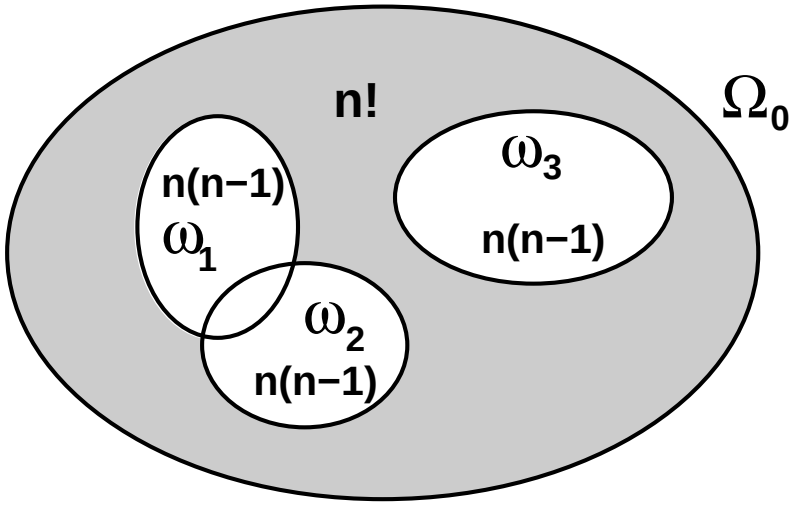

Figure 6.5: Sub-spaces explored by the MSES technique

Evolutionary search techniques as SA need short step sizes. Otherwise, the larger the step-size, the lower the chance of finding local optima, but the closer the search to behave as a pure random. For a sequence of n elements this means a solution space Ω_0 of $n!$ possible sequences, which cannot be exhaustively explored.

Given the shift of one element of the sequence as the basic step-size, the Mixed Stochastic

Enumerative Search (MSES) presented proposes the exhaustive enumeration of the surroundings of each point attained, which correspond to the subsets ω_i in figure 6.5.

The enumeration of $n(n-1)$ states in ω_1 is affordable for the search algorithm proposed (Fig. 6.6). Furthermore, the enumeration is usually not completed because as soon as a better solution is obtained it is proposed as a new candidate for local optimum. Hence, this point is taken as the center of a new sub-space (ω_2) to be exhaustively explored.

Otherwise, if the enumeration is completed the result is the detection of a local optimum. Therefore, no more single step move makes sense and a further move is necessary. This is given, for example, using two random moves as illustrated in figure 6.6.

The algorithm in figure 6.6, however, corresponds to the non-constrained case, in which all sequences are feasible. For the constrained problem the same scheme is still effective, but a checking stage must be included. In such a case, considering pre-ordering constraints (Pinto and Grossmann, 1996; Rodrigues et al., 2000) may result in increasing the efficiency and speed of the procedure, as the sub-spaces to be explored may be significantly reduced.

However, constrained situations may lead to non-convex problems, for which some sub-spaces (ω_3) may not be reachable from the currently explored sub-space (ω_2). In order to cope with this situation, as well as for increasing flexibility, the implementation of this strategy includes the option for running several times the procedure starting from different random seeds.

Put into thermal terms, the strategy proposed consists of fast "freezing" of the system considered with heating when necessary (when local optima is detected).

6.2.2.4 Case study

The following case study is an unconstrained ATSP with 20 cities whose distance or cost matrix has been randomly determined in the range 0 to 1000 (average circuit distance is 10000). Additionally, the distances for the pairs in the sequence 20-19-18...3-2-1 have been set to 1 so that the optimum solution is known (minimum distance 21).

Results are summarized in figures 6.7 and 6.8.

Several computational experiments have been carried out to compare the performance of the MSES approach with random search under different conditions. Figures 6.7 and 6.8 corresponds to the plot of the final objective function attained, OF, after $2 \cdot 10^6$ iterations versus the probability, P, for accepting a x% positive change of such objective function, $\Delta(OF)$, following Metropolis criterion:

$$P = e^{\left(\frac{-\Delta(OF)}{T}\right)} \tag{6.1}$$

Thus, each column in the plot corresponds to an "isothermal bath" performed at temperature:

$$T = \frac{-\frac{x}{100}}{\ln P} \tag{6.2}$$

When discarding a change, the procedure may return to the previous solution or to the best solution found during the search. Figures 6.7 and 6.8 correspond respectively to these situations. In both cases, six executions at each given probability are compared with other six

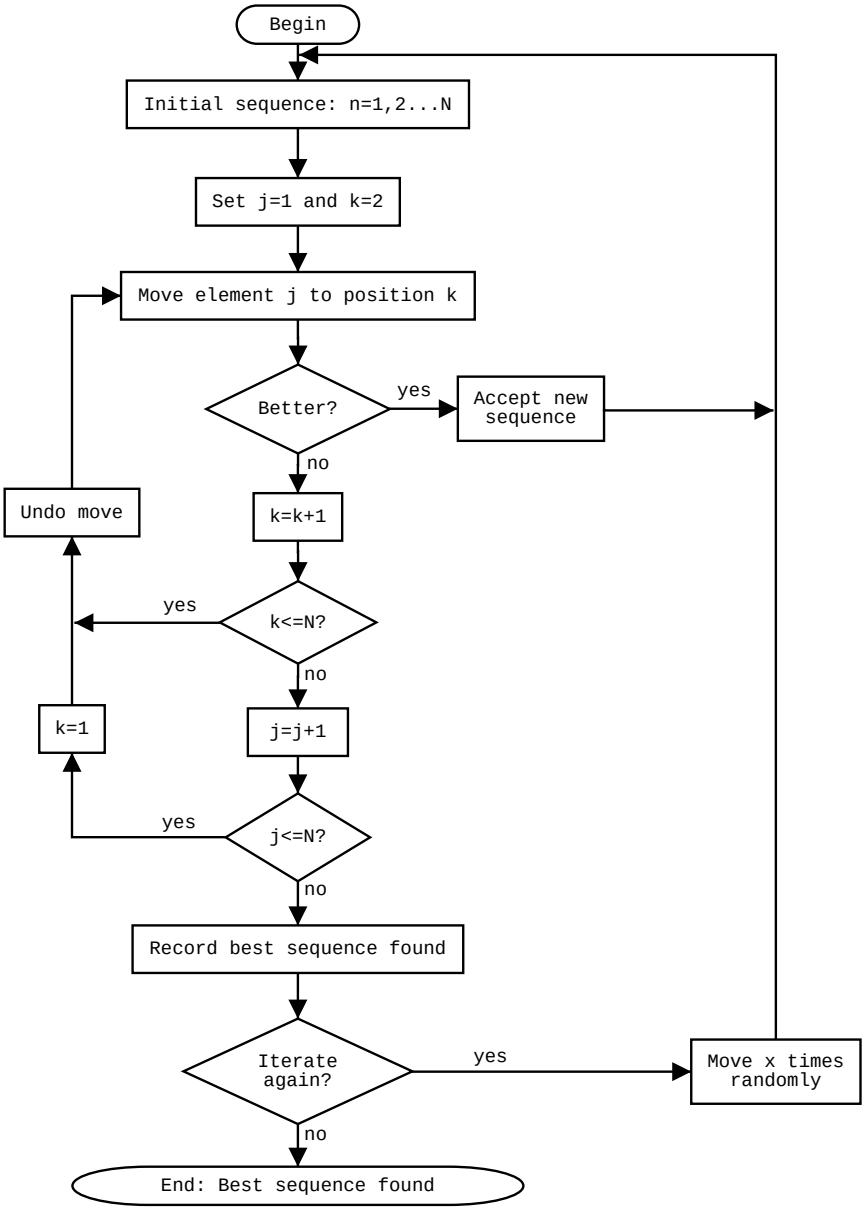

Figure 6.6: Mixed Stochastic Enumerative Search (MSES) algorithm for unconstrained sequences.

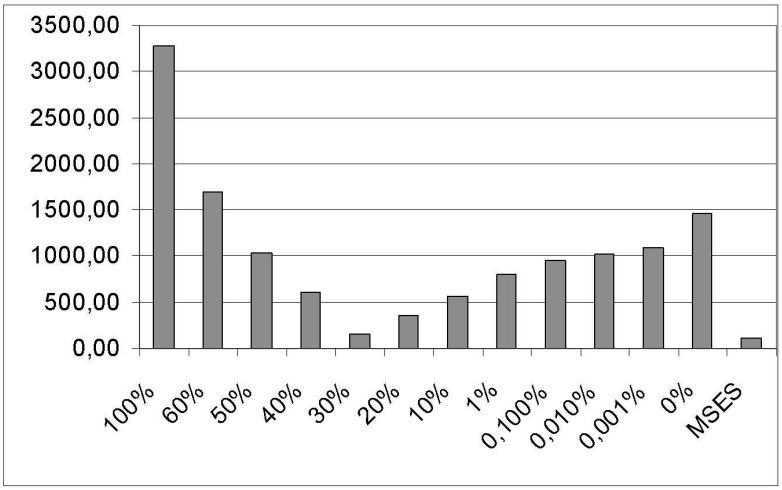

Figure 6.7: Results obtained for different random search procedures following Metropolis criterion compared with the MSES procedure. Random search returns to previous solution

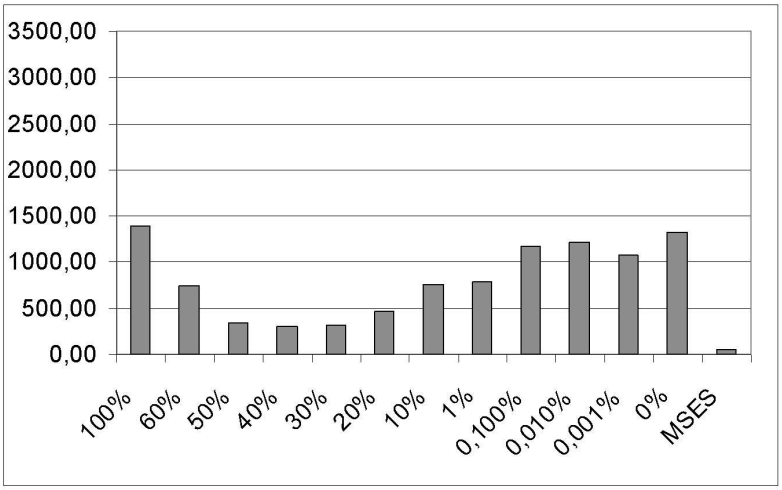

Figure 6.8: Results obtained for different random search procedures following Metropolis criterion compared with the MSES procedure. Random search returns to best solution found.

MSES executions for which the mean value is plotted. Each execution was limited to $2 \cdot 10^6$ iterations (in front of $20! = 2.4 \cdot 10^{18}$ feasible solutions). Results show how the MSES approach led readily to the known optimal solution while stochastic search following Metropolis criterion may lead to close performance only if accurately tuned.

6.2.2.5 Application to the scheduling problem

The same procedure applied in simulated annealing has been followed in MSES. The modified algorithm (see figure 6.9) includes changes in the assignment and feasibility check of the sequence. The same rules and coding used in the SA algorithm is also applied here. This aspect allows the straightforward comparison between both methods.

6.2.3 Genetic algorithms

6.2.3.1 Introduction

Genetic Algorithm (GA) is a heuristic method of local search, which allows the resolution of large combinatorial optimization problems.

The general idea of a GA is to start with randomly generated solutions and implement a "survival-of-the-fittest" strategy to evolve good solutions. An individual with a higher fitness than the general population will have a better chance to survive and become a parent of the next population. GAs rely on the collective learning process within a population, where each member represents a search point in the space of potential solutions of a given optimization problem.

Genetic algorithms have been also applied to the scheduling problem (Jung et al., 1998; Azzaro-Pantel et al., 1998; Noze et al., 1999; Wang et al., 2000) . In this work GA will be implemented within the presented framework.

6.2.3.2 Methodology

In order to apply GAs, each solution of a problem must be encoded into a gene string known as a chromosome. A symbol or bit in the string is called a gene and it represents a decision variable value. An initial population needs to be constructed as a set of the chromosomes, normally generated randomly. Then, the GA operates iteratively, generation by generation, improving those chromosomes until some termination criteria is satisfied.

The whole process of applying a GA to solve a scheduling problem is summarized in figure 6.10.

There are several utilities available that handle most of the GA common procedures. In this work GAlib, an available C++ library of genetic algorithm components developed by the Massachusetts Institute of Technology (MIT), has been used and extended. GAlib includes C++ tools for using genetic algorithms to do optimization using any representation and any genetic operators.

The library works primarily with two classes: a genome and a genetic algorithm. Each genome instance represents a single solution to the problem and contains three primary operators: initialization, mutation and crossover. The initialization operator determines how the genome is initialized and it is called when the population or the genetic algorithm is

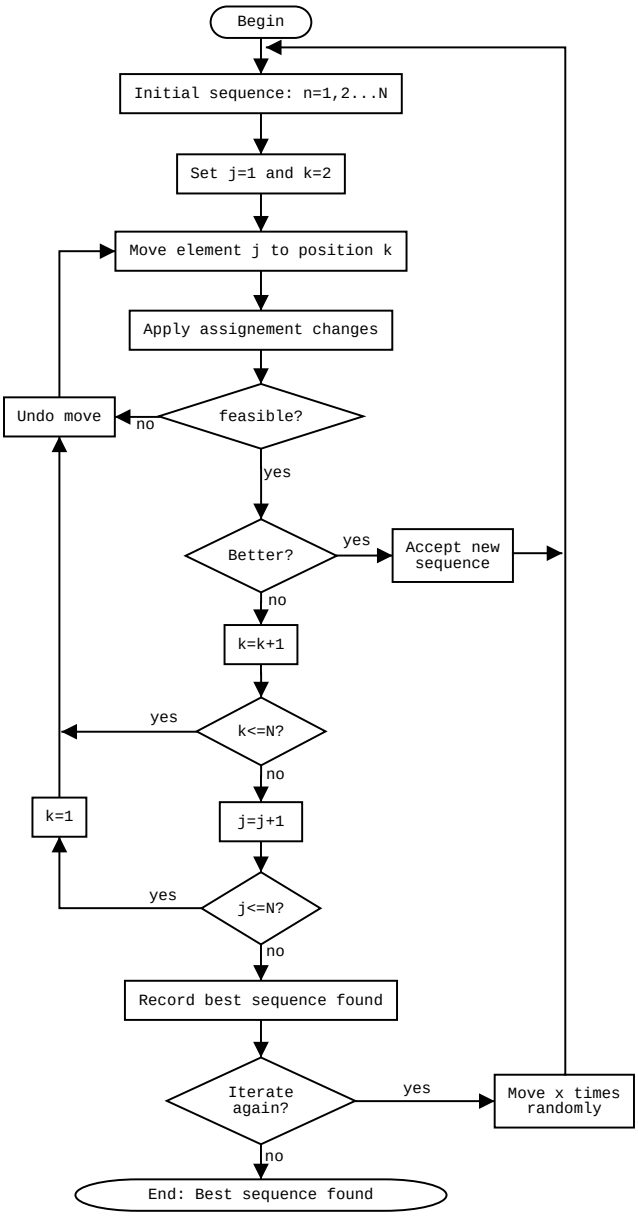

Figure 6.9: Modification of the MSES algorithm for schedule sequences

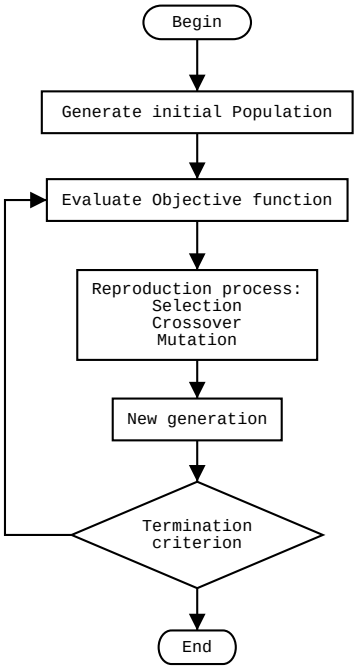

Figure 6.10: Genetic algorithm

initialized. The mutation operator defines the procedure for mutating each genome and the crossover operator specifies the procedure for generating a child from two parent genomes.

The genetic algorithm object defines how the evolution should take place and operates on the population to evolve the best solution. It uses an objective function to determine how 'fit' each genome is for survival and integrates selection and replacement strategies to generate new individuals.

6.2.3.3 Application to the scheduling problem

In order to use GAlib library for the scheduling problem several objects and algorithms have been defined:

- A new genome coding the sequence of batches
- A method for generating the initial population
- A mutation operator
- A set of crossover operators
- The evaluation of the objective function
- A feasibilization algorithm for the obtained sequences.

The first aspect to consider is the genome representation of the schedule. In this case an easy sequence coding like the one used in SA and MSES (see figure 6.2) is also utilized. Each position in the genome represents a position in the sequence.

The second step is the generation of the initial population. In this case, as an initial solution is given, the initial population is generated applying n times the mutation operator to the initial sequence.

The mutation operator used is represented in figure 6.11. As it is shown, the mutation consist in swapping two positions of the sequence. This mutation operator is different than the neighborhood move used in the SA and the MSES.

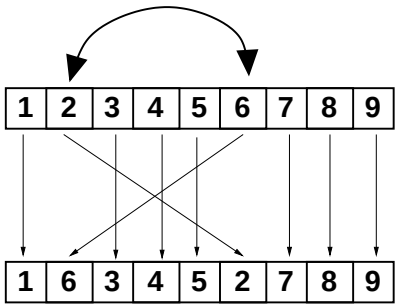

Figure 6.11: Mutation operator for the GA

Four different crossover operators adequate for the genome representation have been selected for applying the GA to the scheduling problem:

- PMX (Partially Mapped Crossover)
- OX (Order Crossover)
- CX (Cycle Crossover)
- ERX (Edge Recombination Crossover)

PMX crossover builds an offspring by taking a sub-tour from one parent and preserving the order and position of as many positions as possible from the other parent (see figure 6.12). A crossover point is randomly selected to define a matching section. The elements of this section define an interchange mapping, which is applied pointwise to the parents to get the offspring.

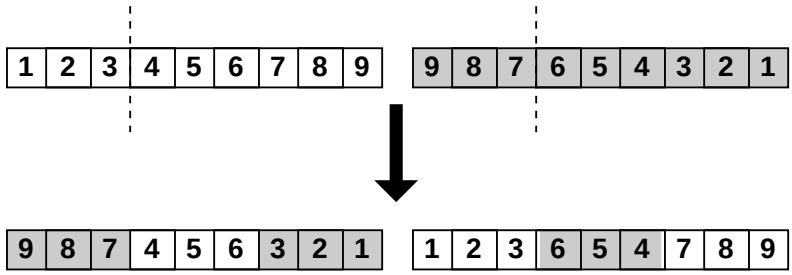

Figure 6.12: PMX crossover operator

As alternative crossover, OX takes a sub-tour from one parent and preserves the relative order of the other parent. In this case, two crossover points are picked at random and the strings between them are selected and maintained. Then, the same alleles (each one of the positions in the sequence) from the second parent are deleted and the remaining alleles are copied into the empty positions of the offspring.

The CX operator allows for the combination of two schedules to produce new ones proceeding as follows: an allele from parent 1 is selected and crossed out from parent 2. The corresponding allele from parent 1 is again selected and the process is repeated until the cycle is complete. Then, the remaining positions are copied from parent 2.

The Edge Recombination Crossover builds offspring using only the edges present in both parents. First, an edge list is build starting from the two parents. The alleles in the offspring are selected one at a time by choosing the allele with the smallest number of edges between the alleles connected with the current one (in case of alleles with the same number of edges, a random choice is made). Once an allele has been selected it is removed from the table and the alleles connected to it are considered as candidates for the next selection. The procedure is repeated until all the alleles have been selected.

Figure 6.15 and its associated table 6.1shows an example of this operator.

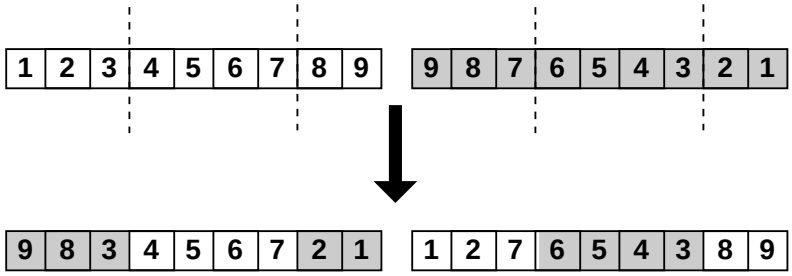

Figure 6.13: OX crossover operator

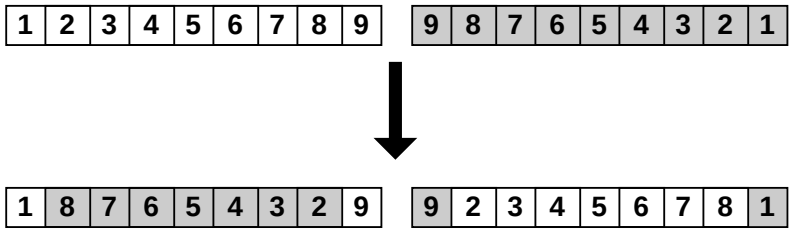

Figure 6.14: CX crossover operator

The evaluation procedure requires feasible sequences in terms of the material balance constraints (section 5.2.2, page 79) so the crossover operator lead very often to infeasible sequences. Therefore a method for obtaining feasible sequences is needed. In this work the algorithm shown in figure 6.16 is used. The resulting sequences of the crossover operator are considered as priority sequences, the resulting sequence is then feasible in terms of material balance thus allowing the evaluation of the objective function.

The same rules used in SA and MSES for assignment are also used in GA after the feasibilization algorithm. Therefore the feasibilization algorithm should also be included in the generic GA algorithm resulting in the new algorithm shown in figure 6.17.

6.3 MILP model

6.3.1 Introduction

As it has been commented in section 2.2.2 (page 13) mathematical programming has been extensively used in the field of planing and scheduling. In order to compare the stochastic methods extensively used in this thesis with the mathematical programming approach a new MILP mathematical formulation has also been developed.

The objective was to develop a detailed model that takes into account most of the aspects described in this thesis like detailed recipe description and storage constraints.

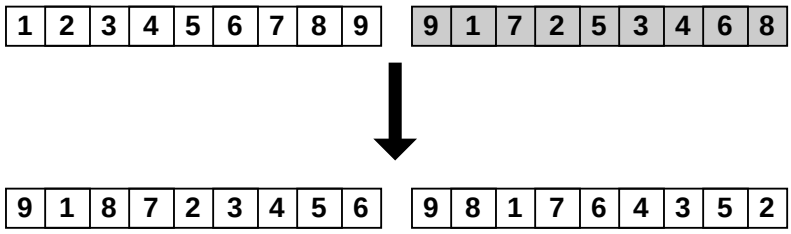

Figure 6.15: ERX crossover operator

Table 6.1: Connection table for the crossover operator shown in figure 6.15

Position	Connected to			
1	2	9	7	
2	1	3	7	5
3	2	4	5	
4	3	5	6	
5	4	6	2	3
6	5	7	4	8
7	6	8	1	2
8	7	9	6	
9	8	1		

The Batch plants scheduling problem modeled in this thesis can be described as:

Given a multipurpose batch plant consisting of a set of units U capable of producing a set of batches B each one characterized with a batch size BS_b and a set of stages S_b , and each stage capable to be performed in a set of different units U_{bs}. The goal is to produce all the batches in a minimum time.

6.3.2 Mathematical model

The following assumptions have been made to derive the proposed mathematical model.

Model parameters are all deterministic.

Processing times only depend on the operation.

The starting point for the model presented is the formulation of the EON timing model which is shown in the equations (6.3) to (6.8).

minimize:

$$Z = C^1 \cdot MS + \sum_{m=1}^{M} C_m^2 \cdot TW_m + \sum_{n=1}^{N} C_n^3 \cdot T_n \tag{6.3}$$

subject to:

$$0 \leq MS \geq T_n \qquad \forall n; n = 1 \cdots N \tag{6.4}$$

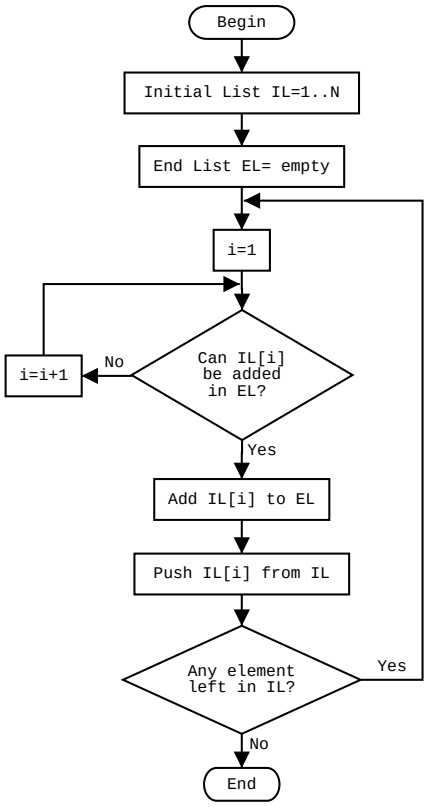

Figure 6.16: Feasibilization algorithm for GA sequences

$$0 \leq T_n \geq T_n^{min} \qquad \forall n; n = 1 \cdots N \tag{6.5}$$

$$T_{FN_m} - T_{IN_m} - TOP_m = TW_m \qquad \forall m; m = 1 \cdots M \tag{6.6}$$

$$0 \leq TW_m \leq TW_m^{max} \qquad \forall m; m = 1 \cdots M \tag{6.7}$$

$$T_{DN_k} \geq T_{ON_k} + \Delta T_k \qquad \forall k; k = 1 \cdots K \tag{6.8}$$

These equations were described in section 4.2 on page 39.

6.3.3 Allocation constraints

New allocation binary variables (Y_{bsu}) are introduced which equals to 1 if stage s of the batch b is allocated to unit u. Each stage should be assigned at least to one unit, but it can not be assigned to more than a specific number of units ($MaxUnits_{bs}$):

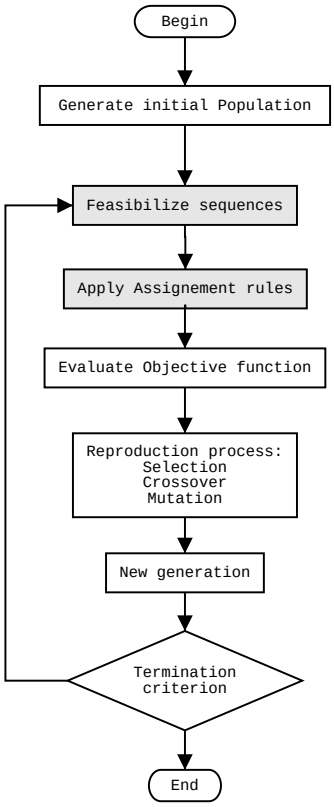

Figure 6.17: Modified GA for scheduling

$$1 \leq \sum_{u \in U_{bs}} Y_{bsu} \leq MaxUnits_{bs} \qquad \forall b \in B, s \in S_b \tag{6.9}$$

Additionally the allocated units have to be able to produce the required batch size (BS_b) given by the unit capacity (Q_u) using a size factor (SF_{bsu}) associated to the batch, the stage and the unit used:

$$\sum_{u \in U_{bs}} \left(SF_{bsu} \cdot Q_u \cdot Y_{bsu} \right) \geq BS_b \qquad \forall b \in B, s \in S_b \tag{6.10}$$

Finally, the assignment has to agree with the topology of the plant. A batch b allocated to unit $u \in U_{bs}$ at stage $s \in S_b$ can be allocated to unit u' at stage $s+1$ only if the unit u' is connected to unit u.

$$Y_{bsu} \leq \sum_{u' \in U_{bs,bs+1}} Y_{bsu'} \qquad \forall b \in B, s \in S_b \tag{6.11}$$

6.3.4 Sequence constraints

Sequence constraints complementary to equation 6.8 should also be introduced. Equation 6.8 was used in the original formulation not only for expressing the time constraints within a recipe, but also to represent the constraints between tasks in a given sequence of batches. Each stage has associated a start event (SN_{bs}) and an end event (EN_{bs}). To generate the sequence constraints the concept of predecessor, from Mendez and Cerdá (2000), is used. The predecessor concept is introduced using a new binary variable, ($P_{bb'}$) which relates a pair of batches (b and b'). If batch b is processed before batch b' then the predecessor variable $P_{bb'}$ equals to 1. At this point is important to notice than for each couple of batches only a new variable is needed as $P_{bb'} = 1 - P_{b'b}$. In fact the total amount of precedence binary variables needed to represent the sequence constraints for n batches is $\frac{n*(n-1)}{2}$. Once defined the predecessor variables, the new sequence constraints are formulated using a big M:

$$\begin{aligned} T_{SN_{b's}} \quad \geq \quad & T_{EN_{bs}} - M \cdot (1 - P_{bb'}) - M \cdot (2 - Y_{bsu} - Y_{b'su}) \\ & \forall b, b' \in B, b' > b, s \in S_{bb'} \end{aligned} \tag{6.12}$$

$$\begin{aligned} T_{SN_{bs}} \quad \geq \quad & T_{EN_{b's}} - M \cdot P_{bb'} - M \cdot (2 - Y_{bsu} - Y_{b'su}) \\ & \forall b, b' \in B, b' > b, s \in S_{bb'} \end{aligned} \tag{6.13}$$

In the simplest case, where the unit assignment is fixed, so the allocation constraints do not exist, the sequence constraints are much simpler:

$$T_{SN_{b's}} \geq T_{EN_{bs}} - M \cdot (1 - P_{bb'}) \qquad \forall b, b' \in B, b' > b, s \in S_{bb'} \tag{6.14}$$

$$T_{SN_{bs}} \geq T_{EN_{b's}} - M \cdot P_{bb'} \qquad \forall b, b' \in B, b' > b, s \in S_{bb'} \tag{6.15}$$

6.3.5 Storage constraints

As the model basis is the EON model, complex storage constraints can be easily modeled as shown in section 4.5 (page 55) . In this model only the single input-output (SIO) constraints are used. Each material (m) to be stored should be allocated to one storage. This allocation is given by an allocation binary variable X_{mbg}. If the binary variable is 1 then the flow of material m of the batch b (F_{mb}) is associated to the storage g which can be one of the storages associated to the material m (G_m). Each material flow has associated a start event (SE_{mb}) and an end event (EE_{mb}). The storage constraints can be formulated as follows:

Each material flow can only be associated to one storage:

$$\sum_{g \in G_m} X_{mbg} = 1 \qquad \forall b \in B_m, m \in M \tag{6.16}$$

In one storage it can only be one material flow at the same time:

$$\begin{aligned} T_{SE_{mb'}} \quad \geq \quad & T_{EE_{mb}} - M \cdot (1 - P_{bb'}) - M \cdot (2 - X_{mbg} - X_{mb'g}) \\ & \forall b, b' \in B_m, b' > b, g \in G_m \end{aligned} \tag{6.17}$$

$$T_{SE_{mb}} \geq T_{EE_{mb'}} - M \cdot P_{bb'} - M \cdot (2 - X_{mbg} - X_{mb'g}) \quad \forall b, b' \in B_m, b' > b, g \in G_m \tag{6.18}$$

At any time the amount of material stored in one storage unit should be between the maximum level (Max_{mg}) and the minimum level (Min_{mg})

$$Min_{mg} \leq F_{mbg} + \sum P_{b'b} F_{mb'g} + \sum (1 - P_{bb''}) F_{mb''g} \leq Max_{mg} \quad \forall b, b', b'' \in B_m, b' < b, b < b'', g \in G_m \tag{6.19}$$

where F_{mgb} is the flow associated to the material m of the batch b that is assigned to the storage g, and its value is:

$$F_{mbg} = X_{mbg} F_{mb} \qquad \forall b \in B_m, g \in G_m \tag{6.20}$$

Equations (6.19) and (6.20) should be linearized before solving. This can be easily performed again using a big M transformation as the expression

$$F_{mbb'g} = P_{bb'} F_{mb'g} \tag{6.21}$$

can be reformulated as

$$F_{mbb'g} \leq F_{mb'g} + (1 - P_{bb'}) M \tag{6.22}$$

$$F_{mbb'g} \geq F_{mb'g} - (1 - P_{bb'}) M \tag{6.23}$$

$$F_{mbb'g} \leq M \cdot P_{bb'} \tag{6.24}$$

6.4 Comparison

The industrial based case study proposed by Honkomp et al. (2000) has been chosen to compare the different optimization approaches described. The complexity of the original case study has been decreased in order to make this comparison. The following modifications have been made:

- Only recipes concerning to blending have been introduced.
- The number of storage tanks has been decreased (from 80 to 34).
- No shifts have been introduced, the availability is supposed to be the 100% the 24 hours per day.

As an illustrative example a test performing between 5 and 20 batches of different recipes is been carried out. The recipes chosen for each test are the first N recipes given in the referenced paper (i.e. the test with 7 batches calculates the production of one batch of each of the first 7 recipes provided in the case study).

Results of applying the MILP model to the test case using Gams/CPLEX 7.5 running in a 2 GHz Athlon under Windows 2000 with a resource limit of 10 hours are shown in the table 6.2 and figure 6.18.

Table 6.2: Variables and CPU time needed using the MILP model

Batches	Constraints	Variables	Binary variables	CPU (sec.)
5	811	232	190	0.56
6	1195	281	231	3.75
7	1653	331	273	52.1
8	2185	382	316	421
9	2791	434	360	1000
10	3472	491	409	5344
11	4233	549	459	6381
12	5074	608	510	14311
13	5995	668	562	10475
14	6996	729	615	22440
15	8077	791	669	36000
16	9238	854	724	36000
17	10479	918	780	36000
18	11800	983	837	36000
19	13210	1049	895	36000
20	14682	1116	954	36000

The large amount of constraints needed is due to the amount of different units capable to perform the same stage. That forces to compute equations (6.12) and (6.13) for each combination of the assignment and sequence binary variables.

A second point to show is that calculation of problems with 15 or more batches requires more than 10 hours. Additionally, it can be observed that the CPU time required to calculate 13 batches is lower that the CPU time required for 12 batches. This fact is due to a change in the recipe structure of the batch number 13.

The same experiments have been carried out using the different optimization strategies presented in this chapter and in the previous one. The same computer equipment and conditions that in the MILP experiments were used. In order to illustrate the results the following tests has been selected in order to illustrate the totality of the results:

- MILP model calculation with a CPU time limit of 5 minutes, 1 hour and 10 hours. (MILP 5' , MILP 1h, MILP 10 h).
- Generation of an schedule using all the possible combinations of methods shown in chapter 5 and selection of the best result. (Best Seq.).
- Generation of a bad initial solution using one possible combination of the sequencing, assignment and material balance rules (FP,FU,LSL) (Init. sol. A).
- Generation of a good initial solution using FP+MAU+LSL (Init. sol. B).
- MSES limited to 500 iterations starting from Init. sol. A and Init. sol. B using the assignment LUU rule (MSES A and MSES B).

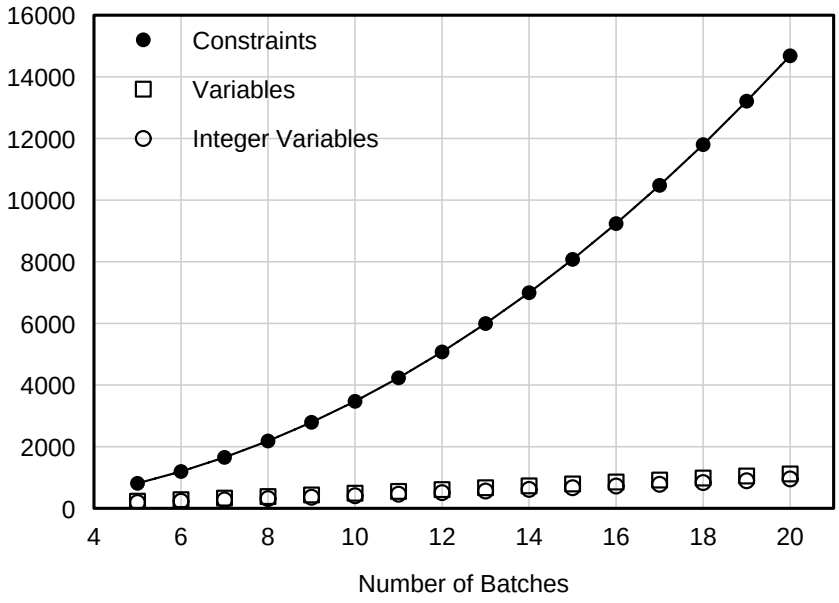

Figure 6.18: Number of constraints and variables needed using the MILP model

- SA limited to 500 iterations starting from Init. sol. A and Init. sol. B using the assignment LUU rule (SA A and SA B).
- GA of 50 generations and 10 individuals each generation using the assignment LUU rule and the PMX operator starting from Init. sol. A and Init. sol. B (GA A and GA B).

Table 6.3 shows the amount of CPU time required for each of the situations (the maximum time is shown for each of the stochastic methods). The CPU time required for each of the three stochastic methods used is of the same order of magnitude as all of them executes approximately the same number of evaluations (500). The evaluation of the 190 possible combinations of the sequencing algorithms requires a considerable amount of time, basically due to all the rules that needs the calculation of partial schedules (MAU,SPT,LPT,etc.). However, all the experiments done with stochastic methods require considerable fewer time than the time required by the MILP formulations.

Table 6.4 summarizes the results of the objective function found using the different approaches. The best solutions found are highlighted. Several aspects can be noticed:

- The MILP limited to 10 hours of calculation was able to found the best solution only in 10 of the 16 scenarios.
- The SA and MSES generated the same solution, in the best case the performance was worse that the MILP as they only find in five cases the best solution.

Table 6.3: CPU times required for the sequencing and optimization algorithms

Batches	Best. Seq. (sec)	Init. Sol (sec.)	MSES (sec.)	SA (sec.)	GA (sec.)	MILP (min.)
5	12	0.1	3	10	9	0.01
6	13	0.1	4	12	10	0.1
7	16	0.1	8	20	11	0.9
8	19	0.1	12	25	13	7.0
9	24	0.2	20	30	15	50.0
10	30	0.2	22	35	17	89.1
11	38	0.2	25	39	19	106.4
12	47	0.2	28	45	21	238.5
13	62	0.2	32	51	24	174.6
14	79	0.3	36	57	27	374.0
15	95	0.3	39	63	30	>600
16	115	0.3	43	70	33	>600
17	138	0.3	47	77	34	>600
18	163	0.4	52	85	42	>600
19	195	0.4	57	91	43	>600
20	233	0.4	62	99	50	>600

- Generating all the possible combination of assignment and sequencing rules generates in nine cases the best solution. It also generates better solutions than the MILP used when the number of batches is high.

- The Genetic Algorithm was the best performer of all the tests presented. The best solution was found in all but 1 scenario.

6.5 Conclusions

This chapter has shown all the optimization techniques used in this thesis within the the scheduling approach used. Stochastic and mathematical methods have been used and tested.

Regarding to the stochastic methods, a new optimization algorithm (MSES) has been introduced that improves the performance of the SA algorithm. The GA algorithm has also been considered and a transformation algorithm is included to convert the infeasible sequences commonly generated into feasible ones. All the stochastic methods used were adapted to be used in a batch oriented approach involving batch sequencing and rule driven unit assignment.

Regarding to the mathematical approach, the mathematical formulation presented in chapter 4 has been extended introducing sequence and assignment variables as well as storage constraints.

A motivating example has been taken as a test case. All the optimization approaches presented as well as all the sequencing algorithms presented in the previous chapter have

Table 6.4: Makespan obtained aplying the different optimization approaches

Batches	MILP 10h	MILP 1h	MILP 5 '	Best Seq.	Init. sol. A	MSES A	SA A	GA A	Init. sol. B	MSES B	SA B	GA B
5	**10.45**	**10.45**	**10.45**	**10.45**	14.52	**10.45**	**10.45**	**10.45**	**10.45**	**10.45**	**10.45**	**10.45**
6	**12.18**	**12.18**	**12.18**	**12.18**	17.27	**12.18**	**12.18**	**12.18**	**12.18**	**12.18**	**12.18**	**12.18**
7	**13.92**	**13.92**	**13.92**	**13.92**	20.02	**13.92**	**13.92**	**13.92**	**13.92**	**13.92**	**13.92**	**13.92**
8	**15.65**	**15.65**	**15.65**	**15.65**	22.77	**15.65**	**15.65**	**15.65**	**15.65**	**15.65**	**15.65**	**15.65**
9	**17.77**	**17.77**	19.80	**17.77**	26.37	18.55	18.55	**17.77**	18.55	18.55	18.55	**17.77**
10	**17.77**	**17.77**	18.78	17.83	28.67	18.55	18.55	**17.77**	18.55	18.55	18.55	**17.77**
11	**17.77**	18.47	19.48	17.83	31.18	18.55	18.55	**17.77**	18.55	18.55	18.55	**17.77**
12	**17.77**	18.78	20.50	17.83	33.70	18.55	18.55	**17.77**	18.55	18.55	18.55	**17.77**
13	**17.77**	18.78	20.82	**17.77**	36.52	18.55	18.55	**17.77**	18.55	18.55	18.55	**17.77**
14	18.47	18.55	18.55	17.83	39.33	18.55	18.55	**17.77**	18.55	18.55	18.55	**17.77**
15	**17.77**	25.70	36.28	18.55	41.85	18.55	18.55	**17.77**	18.55	18.55	18.55	**17.77**
16	21.78	25.50	22.67	18.55	44.15	18.55	18.55	**17.77**	18.55	18.55	18.55	**17.77**
17	21.47	25.87	25.84	**18.67**	46.90	19.12	19.12	**18.67**	19.12	19.12	19.12	**18.67**
18	23.70	29.83	36.08	19.95	49.72	21.40	21.40	**19.90**	19.95	19.95	19.95	**19.90**
19	27.60	29.42	43.90	**20.88**	52.23	21.40	21.40	21.07	21.07	21.07	21.07	21.07
20	25.32	28.22	28.85	**21.57**	54.53	23.02	23.02	22.08	**21.57**	**21.57**	**21.57**	**21.57**

been executed in the test case and the results compared. The GA approach have been shown to be the best approach for this case study, finding the best solution in most of the scenarios used whit short CPU times.

Nomenclature

B: Set of batches.

BS_b: Batch size of batch b.

DN_k: Destination event of link k

EE_{mb}: End event associated to F_{mb}.

ET': Finishing time of operation'

ET_n: Finishing time of operation n.

EN_{bs}: End event associated to the stage s of batch b.

F_{mgb}: Flow associated to the material m of the batch b that is assigned to the storage g.

F_{mb}: Flow of material m of the batch b .

G_m: Set of storages storages associated to the material m.

FN_m: Final event of operation m

IN_m: Initial event of operation m

IT': Starting time of operation'

IT_n: Starting time of operation n

K : Delta time

K_n :Fraction of discharge μ_n

$MaxUnits_{bs}$: Maximum number of units that can perform simultaneously the stage s of batch b.

Max_{mg}: Maximum level for material m in storage g.

Min_{mg}: Minimum level for material m in storage g.

MS : Makespan value

NI_{V_n}: Initial event of variation V_n

NF_{V_n}: End event of variation V_n

ON_k: Origin event of link k

P: Probability.

$P_{bb'}$: Binary variable which relates a pair of batches (b and b'). If batch b is processed before batch b' then the predecessor variable $P_{bb'}$ equals to 1.

Q_u: Capacity of unit u.

S_b: Set of stages of batch b .

SE_{mb}: Start event associated to F_{mb}.

SF_{bsu}: Size factor associated to the batch b, the stage s and the unit u.

SN_{bs}: Start event associated to the stage s of batch b.

TOP_n: Operation time associated to the operation n

TW_m : waiting time of operation m

TW_m^{max} : Maximum waiting time for operation m

$TWmax_n$: Maximum waiting time for the operation n

T_n: Time value associated to event n

T_n^{min}: Minimum time associated to event n

U: Set of units.

U_{bs}: Set of units that can perform the stage s of batch b.

V_i: Level variation of an storage.

X_{mbg}: Binary variable which is 1 then the flow of material m of the batch b (F_{mb}) is associated to the storage g.

Y_{bsu}: Binary variable which equals to 1 if stage s of the batch b is allocated to unit u.

Z: Objective function value

ΔT_k: Delta time associated to link k

$\Delta(OF)$: Change in the objective function.

Ω_0: Solution space of $n!$ possible sequences.

ω_i: Sub-space $\in \Omega_0$.