# Solving a Concrete Sleepers Production Scheduling by Genetic Algorithms

Pérez-Vázquez, M.E.[1]; Gento-Municio, A.M.[1] and Lourenço, H.R.[2]

([1]) Organización y Gestión de Empresas.
E.T.S. Ingenieros Industriales.
Universidad de Valladolid
Valladolid, Spain.
elena@eis.uva.es; gento@eis.uva.es

([2]) Departamento d'Economia i Empresa
Universitat Pompeu Fabra
Barcelona, Spain
helena.ramalhinho@upf.edu

**UPF Working Paper Series**

**Abstract**

PRECON S.A is a manufacturing company dedicated to produce prefabricated concrete parts to several industries as rail transportation and agricultural industries. Recently, PRECON signed a contract with RENFE, the Spanish Nnational Rail Transportation Company to manufacture pre-stressed concrete sleepers for siding of the new railways of the high speed train AVE. The scheduling problem associated with the manufacturing process of the sleepers is very complex since it involves several constraints and objectives. The constraints are related with production capacity, the quantity of available moulds, satisfying demand and other operational constraints. The two main objectives are related with maximizing the usage of the manufacturing resources and minimizing the moulds movements. We developed a deterministic crowding genetic algorithm for this multiobjective problem. The algorithm has proved to be a powerful and flexible tool to solve the large-scale instance of this complex real scheduling problem.

Keywords: production scheduling, genetic algorithms.
JEL: L61, L23, D83

## 1. Introduction

*PREfabricaciones y CONtratas, S.A.* (PRECON, S. A.) is a manufacturing company established in 1952 dedicated to produce prefabricated concrete parts to several industries as rail transportation and agricultural industries. The firm is today one of the largest industrialized construction company in Spain. It belongs to the Catalonian enterprise Cementos Molins Group, a large concrete manufacturing group found in 1928. The first activities of PRECON were developed in the field of the prefabricated pipes for irrigation, mainly for the Ministries of Agriculture and Labour, with more of 5.000 kilometers of prefabricated irrigation channel. In the 1956, the Spanish National Train Transportation Company, *REd Nacional de los Ferrocarriles Españoles* (RENFE), entrusted the production of reinforced concrete sleepers to two factories of PRECON. Later, these factories also produced sleepers to *FEVE* (narrow-gauge railway) and two underground public transportation companies, *Madrid* and *Barcelona* METROPOLITANO. In 1986, RENFE presented the project of high speed train, AVE, with an initial line between Madrid and Sevilla. PRECON signed a contract with RENFE to manufacture 117,000 meters of pre-stressed concrete sleepers for the lines and siding of the new railways. Actually, PRECON is the first Spanish company with the adequate technology and design for the manufacturing of the pre-stressed sleepers. See Figure 1 for an example of a railway siding with concrete sleepers.

The objectives of PRECON can be summarized as follows:

- Fabrication and assembly of all types of prefabricated concrete elements, preferably, products that require high technology and quality;
- Having a geographic implantation around all Spanish state;
- Reaching the best relations between quality-price and due dates of the market.

**Figure 1: Concrete sleepers in a siding railway**

The aim of this work is to solve the scheduling problem associated with the manufacturing of sleepers at PRECON. This is a complex real problem with several production constraints and multiple objectives, as maximizing the plant utilization and minimizing the changes in the production environment. The main objectives of the company are to be able to respond to the increasing demand of sleepers and to schedule efficiently its production given the strong limitations on the main production resource which are the moulds.

Scheduling refer to the assignment of limited resources over time to accomplish an organization's tasks. Some usual objectives of a scheduling system are to maximize resources utilization, to minimize completion time or the customer waiting time. Even though it is a short-term decision, it has an important strategic component, since it extends to all of the economical activities of the company. The Operations Research literature is rich in models and techniques to scheduling problems, see for example French (1982), Morton and Pentico (1993), Chrétienne *et al.* (1995), Brucker (1998) and, Pinedo and Chao (1999). However, it appears that each new real scheduling problem presents different characteristics that must be taken into account. On the other hand, many lessons can be learn from the extensive scheduling literature. Given the complexity and size of most real scheduling problems, it is clear that to be able to solve efficiently these problems, an heuristic method should be used. The scheduling planning system should be able to give a good solution within an acceptable time. The most successful methods to solve scheduling problem are based on metaheuristics, as Tabu Search, Glover and Laguna (1998), Simulated Annealing, Kirkpatrick *et al.* (1983), Van Laarhoven *et al.* (1992), Iterated Local Search, Lourenço *et al.* (2003), Genetic Algorithms, Fang (1994), Mattfeld (1995), among others. Also, the heuristic method should be easily adapted to an always changing production environment. To solve the sleepers production scheduling problem we propose a Genetic Algorithm, based on a specific technique known as deterministic crowding. Genetic Algorithms (GAs) are a flexible and powerful tool for search and optimization. The motivation to use GAs is due the good results that this technique has obtained with hard scheduling problems: Bagchi (1999), Wang and Zheng (2001), and the easy adaptation to multiobjective problems, Deb (2001).

The paper is organized as follows: first, we describe the production characteristic of the manufacturing of the concrete sleepers and the respective scheduling problem. Afterwards, we present the deterministic crowding genetic algorithm, followed by the details of our implementation. In the next section, we describe the specific characteristics of the tested instances. In section 4, we analyze the results and finally we conclude in section 5 with general remarks on this work and directions of future research.

## 2. Production system of PRECON

The main products of PRECON are the sidings for railways. Each period of two or three weeks, PRECON must plan its short-term siding production schedule. The production scheduling takes into account the siding customers demand and their respective due date to delivery. Each siding is composed by a set of concrete sleepers for the lineal extension and the radius of curvature, see Figure 1. There exist a large quantity of different types of siding, with different lineal extension and radius of curvature that allows perfectly adaptation to the landscape and circulation characteristics, in terms of velocity. Different sidings can share the same type of sleepers. The firm maintains a database that indicates the characteristics of each sleeper and the composition of each siding. In this way, the inclusion of new sidings or modifications in existing ones is realized in a easy way. The basic production unit of work is concrete sleeper, and therefore production planning only refers to sleepers. The association between sleepers and their respective siding is done a posteriori. Table 1 presents an example of a list of

15 sleepers of a specific siding. The main characteristics each of these sleepers can be defined as follows (see Table 1):

- Length of the sleeper, in millimeters.
- Configuration and manufacturing specifications.
- Mould reference (registration number of the iron box to be used in the molding process for this specific sleeper).
- Siding (type reference number) where the sleeper will be used.

The responsible for the production scheduling disposes in this way all relevant information of each sleeper needed to be manufactured to complete a specific siding.

Currently, PRECON has 75 different types of siding in its range of products, each one composed by large quantity of sleepers. The number of sleepers of a siding can vary between 70 to 600 sleepers, as for example, sidings for AVE, which need a low curvature radius to adapt its high velocity, have a large number of sleepers.

| Length | Configuration | Mould | Siding |
|---|---|---|---|
| 2617 | VAPE25 | 670F20 | DSH-P-60-318-0,11-CC-D-TC |
| 2617 | VAPE25 | 670F20 | DSH-P-60-318-0,11-CC-D-TC |
| 2617 | VAPE25 | 670F20 | DSH-P-60-318-0,11-CC-D-TC |
| 2617 | VAPE25 | 670F20 | DSH-P-60-318-0,11-CC-D-TC |
| 2667 | VAPE25 | 670G20 | DSH-P-60-318-0,11-CC-D-TC |
| 2667 | VAPE25 | 670G20 | DSH-P-60-318-0,11-CC-D-TC |
| 2800 | VAPE25 | 670A20 | DSH-P-60-318-0,11-CC-D-TC |
| 2834 | VAPE25 | 673002 | DSH-P-60-318-0,11-CC-D-TC |
| 2836 | VAPE25 | 673003 | DSH-P-60-318-0,11-CC-D-TC |
| 2843 | VAPE25 | 673005 | DSH-P-60-318-0,11-CC-D-TC |
| 2848 | VAPE25 | 673006 | DSH-P-60-318-0,11-CC-D-TC |
| 2854 | VAPE25 | 673007 | DSH-P-60-318-0,11-CC-D-TC |
| 2861 | VAPE25 | 673008 | DSH-P-60-318-0,11-CC-D-TC |
| 2869 | VAPE25 | 673009 | DSH-P-60-318-0,11-CC-D-TC |
| 2879 | VAPE25 | 673010 | DSH-P-60-318-0,11-CC-D-TC |

**Table 1: Example of a part of siding "DSH-P-60-318-0,11-CC-D-TC"**

Another important characteristic of the sleepers is the configuration with respect to the concrete and the pre-stressed steel. The attributes of the concrete are insufficient to support the flexural and axial dynamic effects that must sustain the sleepers along their life cycle. Pre-stressed steels must reinforce the internal structure with the finality of warranting the needful resistance limits. PRECON uses three types of configurations named, in this work, as 1, 2 and 3 (see Figure 2).

The PRECON plant has four manufacturing lines of 104 meters for the manufacturing of sleeper types 1 and 2. On the other hand, the configuration 3 (corresponding to AVE) is manufactured on six independent manufacturing lines of the same length.

The main phases of the manufacturing process are as follows:

1. The moulds are arranged in the bottom of the iron box (manufacturing line).
2. The steel cables are tightening to obtain the different configurations (these must be kept along all iron box).
3. The delimiters are placed between the sleepers to delimitate them.
4. The moulds are refilled up with concrete.
5. Waiting time to harden.
6. The delimiters are removed (this process will be replaced for a cutting process of the sleepers).

A Gantt chart of this manufacturing process is shown in the Figure 3. The processing times are, for some phases, estimatives, since for example phase 5 depends on the temperature and humidity and phase 1 depends how many moulds remain from the previous production plank mould. Finally, the sleepers can be extracted and transported by overhead traveling crane.

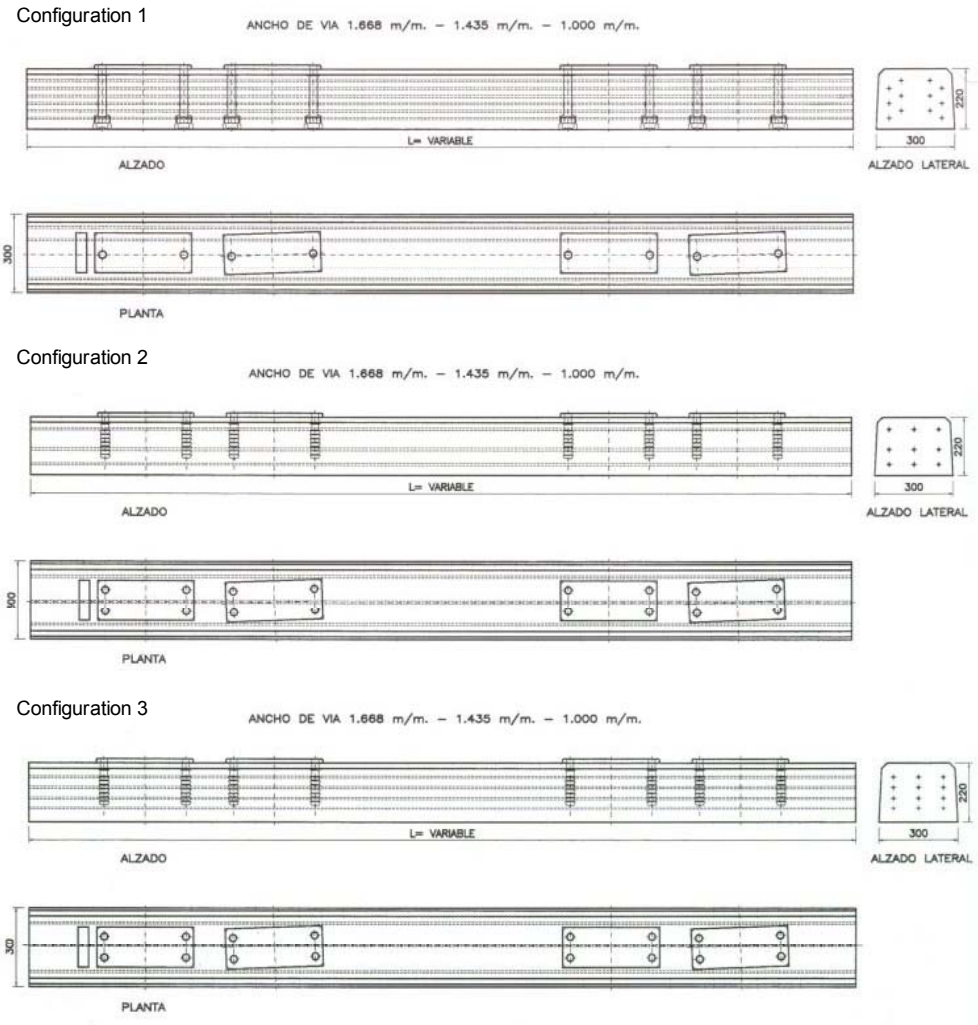


**Figure 2: Configuration types (number and distribution of pre-stressed steel)**

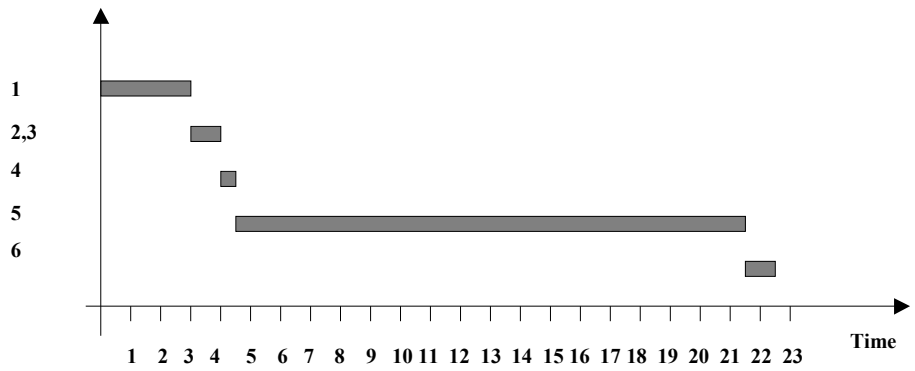


**Figure 3: Gantt of the manufacturing process**

The production of the sleepers is made by consecutive plank moulds, that use the same resources. In between plank moulds, all sleepers must be extract form the manufacturing lines and moulds must be rearranged. This lead to an important scheduling constraint of the production process, which is the limited number of available moulds. The manufacturing of each sleeper needs a type of mould. So, the number of specific sleeper that can be produce on one plank mould is limited to the number of available moulds for that sleeper. This constraint is very strong because for many types of sleepers there is only one mould. Therefore, the production scheduling must take into account this limited resource. Also, moving a mould is a complex operation that should be avoided as much as possible. The moulds are very heavy and the process of arranging them is very slow. The process time will decrease if the production scheduling avoids a large number of moves of the moulds, and so, allowing an increment of the efficiency of the manufacturing process.

The longer phase is the one related with the harden of the concrete. When this is hardened, the internal steel cables that initially are extended along all manufacturing line must be cut to separate the sleepers. This operation is especially dangerous because the stress of these cables, being one of the reasons of the convenience of filling up all (or almost all) manufacturing line. The other reason is, of course, to have the manufacturing lines filled up to obtain a high productivity.

Given the production characteristics and the siding demand for a certain period, the plant responsible must produce a feasible production scheduling that maximizes the use of the available resources and satisfies the demand. The first step of the scheduling process will be to consider the sleepers' configuration to be produced along a manufacturing line and to group them according to this. Afterwards, the scheduling is independently realized for each one of the configurations.

PRECON has two main different objectives to optimize the scheduling process:

- The first and the most important one, due the high volume of production, is to maximize the utilization of the manufacturing resources (measure by the percentage of utilization of the manufacturing line).
- The second one is to minimize the number of changes of mould between consecutive plank moulds.

Next, we will propose an heuristic method based on genetic algorithms to solve this complex real multiobjective problem. The special production environment conditions were taken into account.

## 3. An Heuristics Solution Method based on Deterministic Crowding Genetic Algorithms

In this section, we will start by present an introduction to the GA's that we will use as base for the method implemented in our study: the deterministic crowding GA.

### 3.1. Introduction to the Genetic Algorithms

To solve the concrete sleepers production scheduling problem we propose a solution method that consists in three phases, which includes optimization phases where a

crowding genetic algorithm is applied. The main reason to divide the problem in this phases is the complexity of the production process as described in section 2, as well the presence of two objective functions. The main phases are the following ones:

Phase 1: Data collection.
Phase 2: Optimization of the sleepers sequence.
Phase 3: Optimization of the production series.

The first phase consist in collect dada, i.e. obtain the unsettled orders and their respective due date. With this data, we will obtain the demand orders for concrete sleepers production within a week (or month)'s time (period of time considered by the expert user of each factory). An important aspect on this phase is to balance the sleepers production of the following weeks (or months), to avoid extra hours of work. In resume, this phase produces an assignment of production orders (sidings and sleepers to manufacture) to a period (for example, a week) such that each period is balanced.

The second phase consists in obtaining the production sequence of the sleepers, grouped by plank moulds or manufacturing lines, i.e. given the demand orders of sleepers obtained in phase 1 for the specific period, the manufacturing constraints and characteristics, optimize the sequence of these sleepers on the manufacturing lines (plank moulds) such that all orders are produced. The objective is to maximize the usage of the available production resources i.e. average percentage of manufacturing lines utilization. In this phase, we apply the crowding genetic algorithm that we will describe later in detail.

The third phase consists in deciding how to sequence the plank moulds in series for manufacturing. Note that, in phase two, it was decided the sequence of sleepers within plank moulds or manufacturing lines. Now, the plank moulds should be grouped in series, such that each serie consists at maximum of plank moulds as the number of available manufacturing lines. Note that, most of the times, there is more plank moulds to be produced than available manufacturing lines, therefore we have to sequence the plank moulds in consecutive production series. The main objective in sequencing the plank moulds is to minimize the moulds movements and changes, since as mentioned, this requires a significant amount of work and time. In this phase, we use again a crowding genetic algorithm, that we will describe later.

Before describing the details of two crowding genetic algorithms implemented, we will briefly review the genetic algorithms (GAs) and their crowding version. We will also comment on the main reasons to apply this technique.

The GAs use a search strategy based on several solutions (population) that improve according to Darwin's theories on the species evolution and the natural selection Fogel (1998). Thus, solutions with the higher fitness (the best adapted individuals to environment) have higher possibilities of surviving, reproducing and transmitting their characteristics to their descendants. In the optimizing process, the cost or the quality of the solution, between other measures, quantifies its adaptation to the environment. Solutions with lower cost than the average (or higher quality) will have more probabilities to be selected to be parents transmitting those characteristics, which make them particularity good, to other new solutions (their children). Those children are created with information of both parents. In general, the average cost of the children

will be lower than the average cost of the initial population, being this, the principle of the evolution spices. Therefore, the cost or the quality will improve along generations.

The GAs adapt to many optimization problems, and, as a population-based technique are specially fitted to solve multiobjective problems. GAs, in general, follow the flowchart presented in Figure 4, Goldberg (1989), Davis (1989) and Michalewicz (1995). The basic elements to be defined to a specific problem are: codification of the solution, selection, reproduction and replacement. These elements must be well defined and adapted to each new problem, because, if not, the performance of the algorithm can be not very good, Davis (1989).

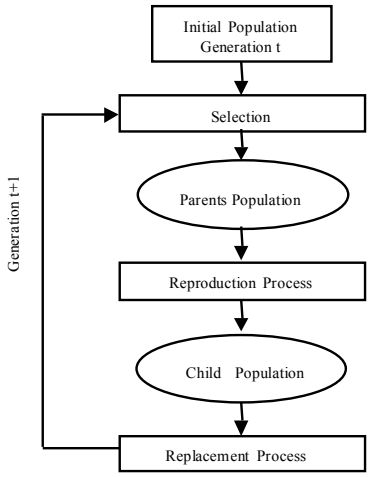


**Figure 4. Simple flowchart of the genetic algorithm**

The first basic decision that affects the GA performance is to define the codification of the solutions of the problem, that should contain all the necessary information to able the search for good solutions. Such codification depends obviously on the specific problem in study, and there appear different possibilities in the scheduling literature. Following the criterion established by Fang (1994) and taking into account the characteristics of the problem (the process belongs to permutation flow shop type), the most appropriate code is the permutation, used as well in the Travelling Salesman Problem (TSP): Greffenstette *et al.* (1985), Whitley *et al.* (1989) and, Escalada and Torres (1997).

For example, if we want to scheduling 1000 sleepers, we will need 1000 integer numbers, each one related to one sleeper: the number 1 will indicate the first sleeper, the number 2 will indicated the second sleeper, and so on.

The evolution process starts with the selection of the first solutions that will form the initial population. As suggested by many authors, Michalewicz (1995), these first solutions (initial population) could be randomly created. In this way, we give to the GA the necessary exploration to realize the best search. The power of the GAs is in the optimal balance between the exploration and the exploitation in the search.

In our case, the first solutions will be random permutations of the integer numbers that we need to code the sleepers. Let *S* be the number of sleepers to be produced, thus, the space solutions will be form by *S*! different permutations of the S integer number. And

one initial populations will be one sample of this space. The population size ($N$) will be selected by the user, with $N<<S!$.

Let us suppose that we have to schedule 9 sleepers, one possible solution is: 1 4 3 7 5 2 9 8 6. The order of the number between the string indicates the position to be scheduled. In the example, the first sleeper to be scheduled will be the codified with the number 1, then will be scheduled the sleeper codified with the number 4, etc.

Once the initial population is created the process, the process continues in an iterative way like in the natural evolution of species. The better individuals of the population will have higher possibilities to survive for the reproduction. To mimic this natural process, the parents are selected according to their fitness solution, the simplest process can be seen like a roulette wheel where each area is proportional to the fitness (quality) solution, and the dart will select the future parents (Figure 5). Some best selection methods available in the literature are the ranking Baker (1985) or tournament system Horn *et al.* (1994).

The standard parent selection is a fitness-based process. Each individual ($i$) has a chance of selection ($p_i$) that is directly proportional to its fitness ($f_i$), where $N$ is the total number of individuals on the population (the population size):

$$p_i = \frac{f_i}{\sum_{j=1}^{N} f_j} \quad \textbf{[1]}$$

Then, the roulette wheel is divided in as many sectors as individuals, and the area of each one is proportional to the chance of selection of the parent that represents (Figure 5):

| Individual | *fi* | *pi* |
|---|---|---|
| 1 | 54 | 12 |
| 2 | 67,5 | 15 |
| 3 | 45 | 10 |
| 4 | 54 | 12 |
| 5 | 63 | 14 |
| 6 | 126 | 28 |
| 7 | 40,5 | 9 |
| sum | 450 | 100 |

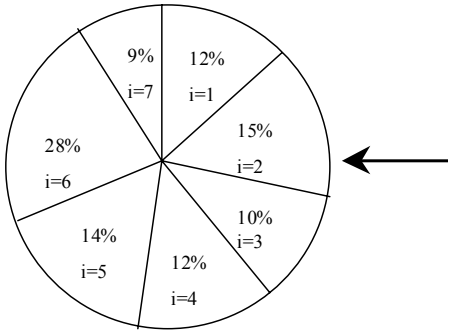


**Figure 5: roulette wheel process selection.**

We turn the roulette wheel as many times as parents we want select. Normally, the size populations are constant: initial population of each generation, parents populations, children populations, therefore we turn it $N$ times. The parents with the higher fitness than will have higher chance to be selected.

The selected individuals in the preceding process are grouped in couples, in a random way, to obtain the breeding pair. The reproduction process consists of two phases: In the first one, the codified material of two solutions is exchanged using the denominated crossover operators; In the second phase, the solution is randomly changed following a mutation process. In this way, the exploitation of good qualities (crossover) and the exploration of new areas of search space (mutation) are done. However, these two

process act in a stochastic way: the crossover process acts with a $p_c$ (crossover probability) for each couple, and the mutations process acts on each child created in the previous phase with a $p_m$ (mutation probability).

Crossover operators, as well as mutation operators, depend on the type of code used. Previous studies have shown that the Order Crossover operator (OX), Michalewicz (1995) see Figure 7, and Order Based Mutation Operator (OBM), Davis (1991) in Figure 8, obtain good results for this permutation code.

The classical crossover operator is only valid to binary code, it is presented in the figure 6. As we can see, the chromosome parents are divided in two parts and they are combined to form two news individuals (the children):

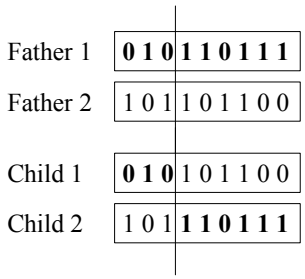


**Figure 6: Classical 1-point crossover operator**

The position of the division is randomly selected. Then, the obtained parts are crossed to form other two individuals. Each one of the children has characteristics of both parents. Thus, combining goods characteristics we hope to obtain improved individuals (with higher fitness).

Following these ideas, it was developed other crossover operators adapted to the permutation code. We must obtain children with characteristics of both parents. We will use the OX (order-based) crossover operator which maintains the order. Other available crossover operators maintain the position, Davis (1991). In our case, the refered characteristics are the orders in which the sleepers will be sequenced and the position within the string: a sleeper sited in firsts positions are sequenced before than other sleepers sited in last positions.

Let's explain in more detail the OX crossover for permutation solutions. See Figure 7 for an example. In first place, two points of cuts are randomly selected. Then, for the child 1 are created an auxiliary string with the parent 1, from the second point of cut and considering its string like circular to the preceding position of the second point of cut. For example, if the second parent is [9 2 6 | 7 3 1 | 4 5 8], the auxiliary string will be [4 5 8 9 2 6 7 3 1]. The child 1 inherits from parent 1, the numbers placed between the two points of cut in position and order (in our example, the numbers are 6 8 3). These numbers must delete of the auxiliary string (4 5 ~~8~~ 9 2 ~~6~~ 7 ~~3~~ 1). With the remaining numbers (4 5 9 2 7 1), we fill up the child 2, starting in the second point of cut and considering the string like circular to the first point of cut (see Figure 7):

| | |
|---|---|
| Auxiliary string | [4 5 9 2 7 1] |
| Child 1 | [- - - \| 6 8 3 \| - - -] |
| Phase 1 | [- - - \| 6 8 3 \| 4 5 9] |
| Phase 2 | [2 7 1 \| 6 8 3 \| 4 5 9] |

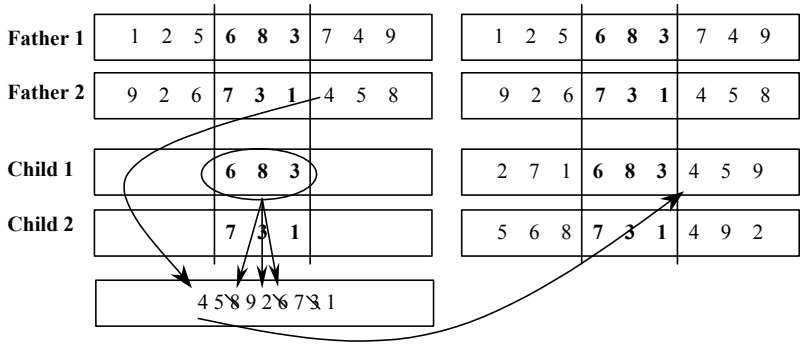


**Figure 7: OX operator**

Next, the GA applies the mutation over each child created with the crossover operation. The mutation permits to search new areas in the solution space and to recover lost characteristics in the evolutionary process. As for crossover operators, there are many mutation operators. We must select the most suitable to the code. The first developed mutation operator was the applied only to the binary codification, where each bit of the string can mutate with a probability $p_m$. Thus, if the bit is a 0, the process will change it to a 1, and if it is a 1, it will be mutated to a 0.

For the permutation code, we have selected the OBM (order based mutation). This mutation operator is based in the order of the sequence: Two positions of the string are randomly selected and the integer numbers of these positions are exchanged (see Figure 8).

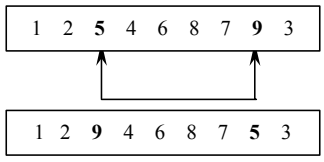


**Figure 7: OBM operator**

In this case of permutation codification, the probability of the mutation process is $p_m$ over the string, in contrast to the binary code where is also $p_m$ but for each bit of the string. The range of the values of the $p_m$ is different to each codification, for the binary one is 0.1-1%, and the permutation one is 1-20%.

Thus, we have the children population formed from (see Figure 8):

- The couples of parents that have not been crossed.
- The children created with the crossover process but not muted.
- The children created with the crossover and mutation process.

The last process of the GA algorithm selects the individuals that will form the initial population of the next generation (the final population, in Figure 8). The most used system is the elitist replacement. The best individual of initial population survives and it is directly selected to be one of the children (it replaces to the worst individuals of the children population). In this way, the evolutionary process always maintain the best solutions found to the moment.

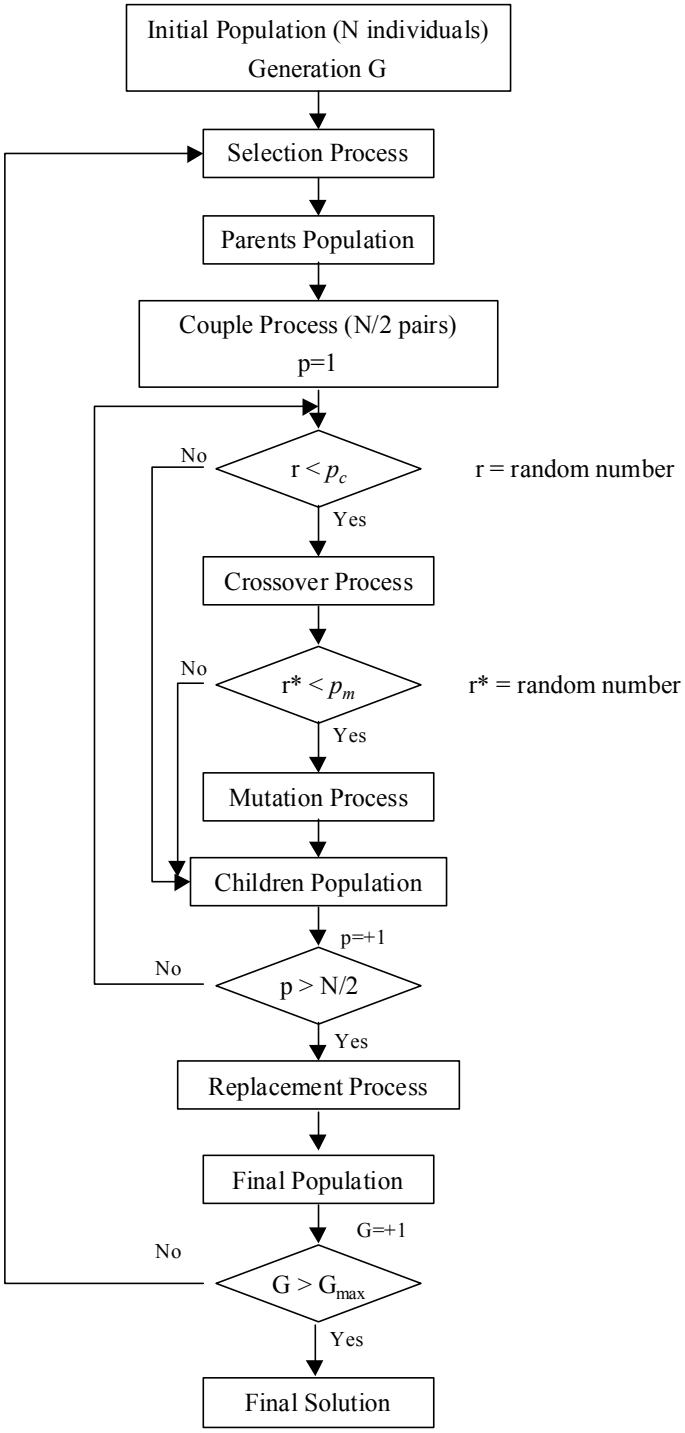


**Figure 8: Total process of a standard GA**

## 3.2. The Deterministic Crowding Genetic Algorithms

In the last years, new versions of GAs methods have been developed that lead to increasing performance in efficacy and efficiency of the GAs. They are based on the biological concept of niche. In nature, a niche is the subdivision of the environment according to the common characteristics that define it. Within a niche, individuals of the same species coexist and struggle among each other for limited resources. This division allows us to decrease the competition among individuals of different species and

maintaining the necessary diversity in the search, avoiding the premature convergence to poor solutions. Moreover, this division allows surviving different types of solutions with the same fitness (multimodal optimization). This last characteristic justifies our decision to select the deterministic crowding GA.

There are different niching GA, such as:

1. Sharing method: Goldberg and Richardson (1987); Oei *et al.*, (1991) and Pétrowski (1996).
2. Crowding method: Mahfoud (1992 and 1995).
3. Sequential method: Beasley *et al.*, (1993).

In this work, we have implemented the deterministic crowding method since it is very fast, simple and with the higher levels of efficacy, Pérez (2001).

The crowding method was proposed by Mahfoud in 1992, Mahfoud (1992). The basic idea is to form groups of couples in a totally random way from the population to apply the crossover and mutation operators, eliminating the selection process. For the replacement process each child competes in a tournament against one of his parents to obtain what individual dies and what survives. Mafhoud based his ideas on the observation done by Cavichio in 1970, Mahfound (1992), that the children should replace their parents since they were the most similar individuals to them. Therefore a distance measure between individuals should be used that gives an idea of the closeness that exists between them.

The main element of the deterministic crowding is the distance comparison between parents and children in the replacement process. This distance can be defined on the codified string or on the decoded solution. For example, considering two individuals with a binary code:

| | Codified solutions | Decoded solution | Fitness [$f(x)=x^2$] |
|---|---|---|---|
| Ind1 | [1 0 **0 0** 1 1 1] | 71 | $71^2 = 5041$ |
| Ind2 | [1 0 **1 1** 1 1 1] | 95 | $95^2 = 9025$ |

The distance will be the number of different bits between both strings. In this case, there are two different bits; thus the distance is two. On the other hand, if we suppose that these individuals are solutions to the problem: maximize the objective function ($x^2$), then their decoded solutions will be, (*71*) and (*95*) respectively, going the traditional decoding of binary code. Then, the distance between these two decoded solutions will be: $d_{Ind1,Ind2} = \sqrt{(71-95)^2} = 24$.

The generic form, the distance between decoded solutions to parametric problem will be:

$$d_{i,j} = \sqrt{\sum_{k=1}^{p} \left(x_{k,i} - x_{k,j}\right)^2} \qquad with \quad p \quad the \quad number \quad of \quad parametre$$

With this simplest example, we have can verify that the most realist distance is the defined on the decoded solutions. The first distance between these other two individuals is the same that previously calculated: 2. However, the distance on the decoded solutions is: $d_{Ind3,Ind4} = \sqrt{(71-68)^2} = 3$ (see individuals 3 and 4 for this example).

| | Codified solutions | Decoded solution | Fitness [$f(x)=x^2$] |
|---|---|---|---|
| Ind3 | [1 0 0 0 1 **1 1**] | 71 | $71^2 = 5041$ |
| Ind4 | [1 0 0 0 1 **0 0**] | 68 | $68^2 = 4624$ |

Therefore, the same distance on the codified strings leads to very different distance on the decoded solutions. Following these ideas, we need to define the distance measure between individuals, taking account the characteristic of the specific problem. The distance for the permutation flow shop problem will be the number of elements that are in different position between the two solutions. See an example on Table 2.

| Solutions | Distance |
|---|---|
| **1** 3 **2** 5 4 **9** 8 7 6 | **3** |
| **9** 3 **1** 5 4 **2** 8 7 6 | |

**Table 2: Example of decoded distance for permutation flow shop problem**

In this case, the distances in the decoded and codified solution are the same since we are considering a direct codification. That is, the codified solution indicate directly the solution of the problem, there is not a decoding process. In our case, the main thing is the number of sleepers scheduled in different positions.

Once the distance is defined, we can described the method. In the deterministic crowding GA (DCGA), there is not a selection process, in its place, the individuals of the population are randomly grouped in couples. Then, each couple goes over the crossover and mutation process following the probability functions, as in original GA. This is the main difference between GA and DCGA. Now, the each group in the process has four individuals: the two parents and the two new children, and the replacement process must select two of these four individuals to next population. This replacement is based in the distance between these four individuals. In first place, the process search which parent is the nearest to each child (that is: father 1 with child 1 and father 2 with child 2, or father 1 with child 2 and father 2 with child 1). The next comparison it is based on the fitness between the nearest individuals, the winner is the individual with the highest fitness (see Figure 9). The process will go on with the next couple.

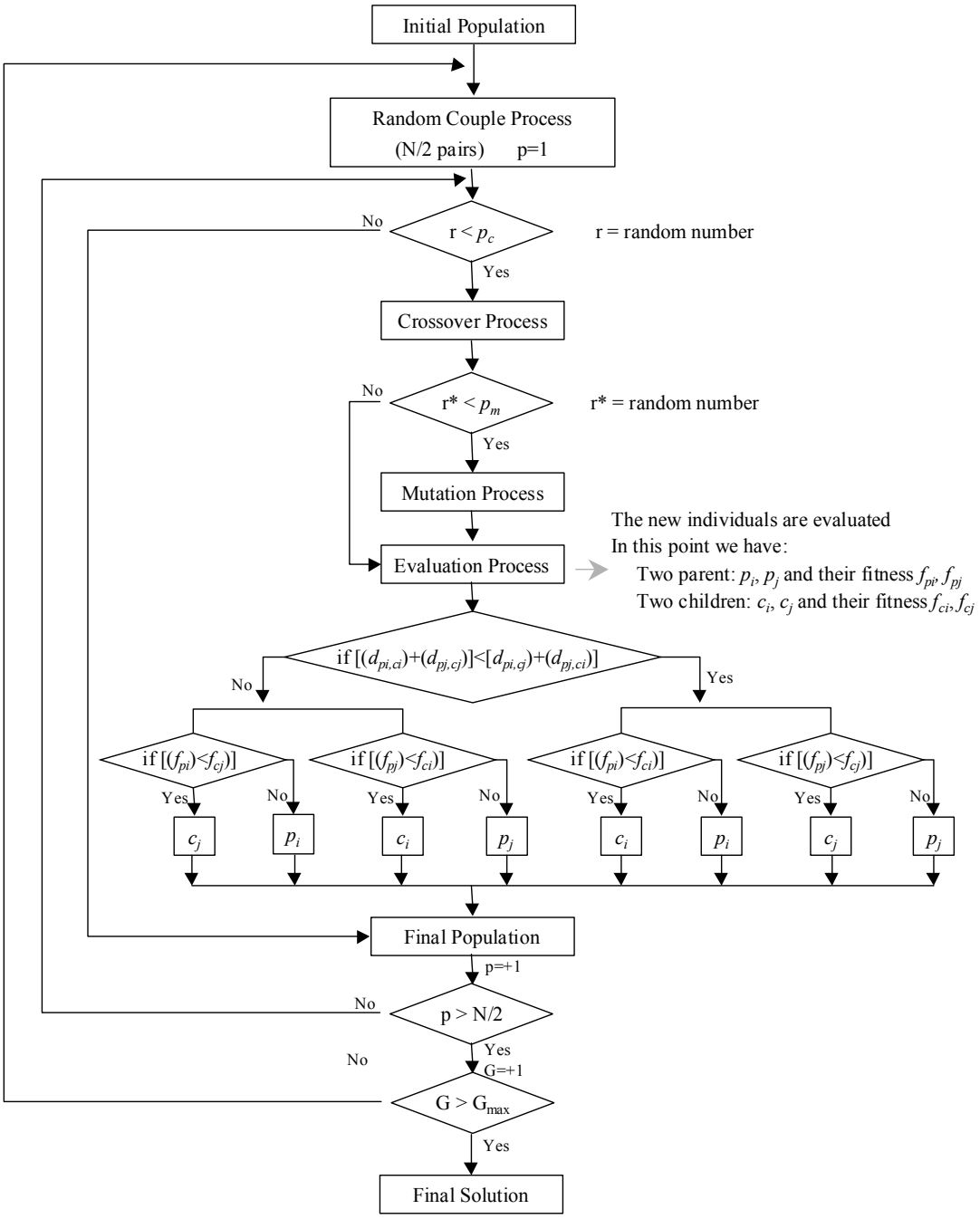


**Figure 9. Deterministic crowding method**

## 4. The DCGA for the Concrete Sleepers Production Scheduling Problem

In this section, we will describe the details of the Deterministic Crowding Genetic Algorithm (DCGA) implementation for the Concrete Sleepers Production Scheduling Problem. The optimization process has been divided into two phases: in the first phase we pretend to optimize the sequence of sleepers on the manufacturing lines (plank moulds), and in the second phase, we will optimize the sequence of the plank mould on different series.

In the first phase, the plank moulds are considered all together in only one sequence of sleepers. The objective is to obtain a permutation of the sleepers just that, after dividing this sequence in plank moulds, we obtain their maximum utilization. Therefore, in the second phase we consider the plank moulds that we have obtained in the previous phase. The DCGA must find the order of the plank moulds within the series to minimize the number of changes of mould between consecutive plank moulds and to maximize the utilization of the manufacturing lines.

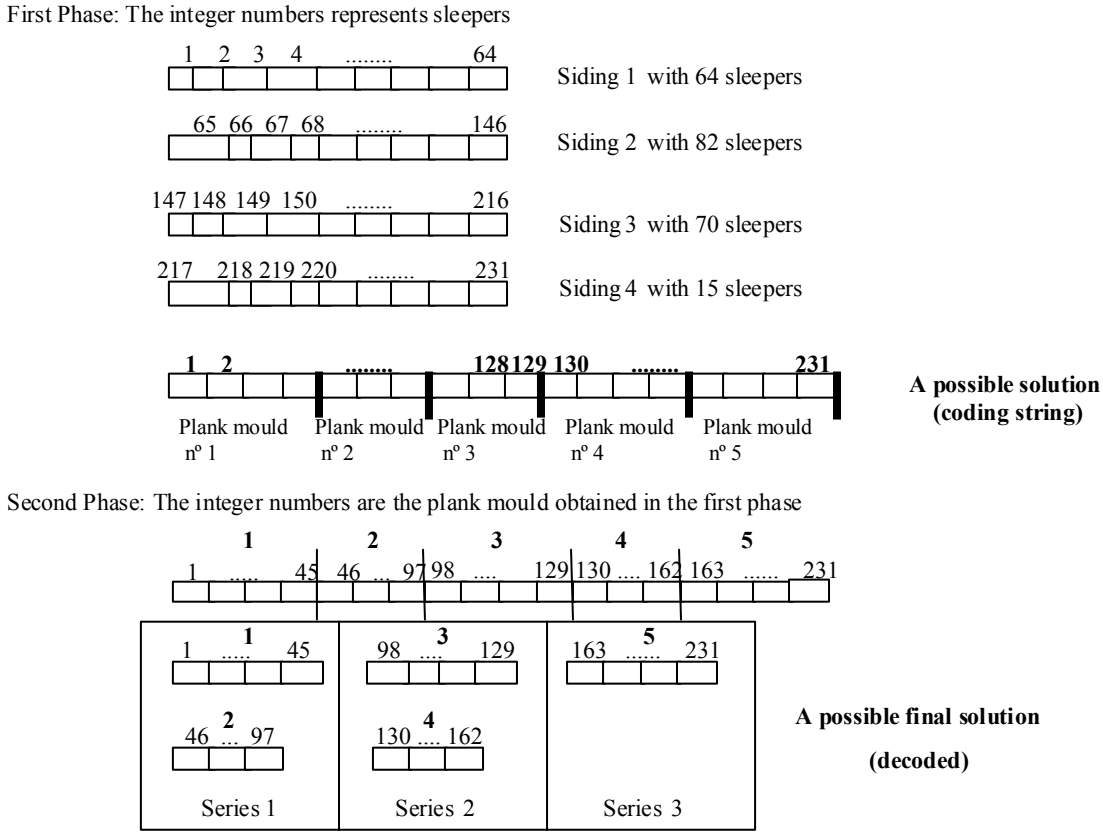


**Figure 10: Total process of the implementation.**

In the Figure 10, we show a small example with five siding and 231 sleepers to be manufactured. In the first phase, the codified string will be formed to 231 integer numbers, and the solutions will be all possible permutations of these integer number. One of these solutions is shown in the phase one of the figure. In the second phase, it is realized a division of this solution in plank moulds. This division depends of the length of the sleepers on the previous sequence. Begging with the first sleeper on the solution, sleepers will add to the same plank mould until the total length would be bigger than 104 meters, then we remove this last sleepers and so, we have the first plank mould. The removed sleeper will be the first sleeper of the next plank mould. The process will go on until all sleepers will be in plank mould.

## 4.1. The First Phase of the DCGA

The codification is permutational, where each integer number corresponds with a sleeper to produce. Associated with each feasible solution, we must calculate its fitness or the value of the objective function. In this case, the main objective function is the

average percentage utilization of the manufacturing lines. The highest is this value the better is the solution.

The initial population will be generated in a random process and we consider the size of the population, say *N*, as a parameter of the algorithm. The process is quite simple, suppose we need to produce in this period *m* sleepers, so *N* random permutations of the integer numbers from 1 to *m* will be created. Note that not all permutations of the *m* number is a feasible solution to our problem, since we must take into account the resource capacity constraints associated with the number of existing moulds for each sleeper. During the generation process of the initial population, we check the solution feasibility and if this constraint is not accomplished, a new solution must be generated. To check if a solution is feasible we proceed as follows: divide the solution (list of sleepers) in sub-list of sleepers whose length sum less than the length of the manufacturing lines, in our cases 104 meters; then, within each plank mould, check that there is no more moulds than the ones available, i.e. the resource constraint. If the solution accomplishes this constraint is feasible, if not, generate another one.

The iterative process will go on *G* times, where *G* is the number of generations or iterations that we intend to repeat. When the process finishes, we can obtain *N* different solutions as maximum and 1 as minimum. But due to the big size of the problem (m! possible solutions), the multimodality of the scheduling problems and the diversity process of the DCGA, we will obtain more than one different solutions with the highest fitness. That is, with the maximum average percentage utilization, therefore we must study the second phase with all ones.

However, we can not only take account the solutions of the last generation (*G*) because we do not know how the filling percentage affects the placing in the series. At first sight, the relation will be inverse, if the filling of the plank moulds is higher, then the allocation could be harder than with others less filling plank moulds. Perhaps, with worse solutions of the first phase, we can obtain best results in the second phase. Therefore, in the second phase, the process takes the *x* best solutions in different iterations of the first phase. For example, if we use 10.000 generations in the first DCGA, the process can keep in reserve from 1 to 50 (*x*) best solutions in the 1000, 3000, 5000, 7000 and 10000 generations, that is, for the second phase we have *5x* starting point for the second DCGA.

### 4.2. The Second Phase of the DCGA

To solve the assigning problem of the plank moulds to production series, we have implemented a very similar DCGA. Each different solution of sleepers sequence obtained in the previous phase constitutes a starting point for this current phase. In this way the process will go on with as many genetic algorithms as different solutions suggested in phase 1.

Following this strategy, the method can propose to the final user different final solutions with different fitness so he or she can make a final decision based on his/her experience and evaluation of other production constraints not included in the method as breakdown machine or others non-quantifiable objectives (see Figure 11).

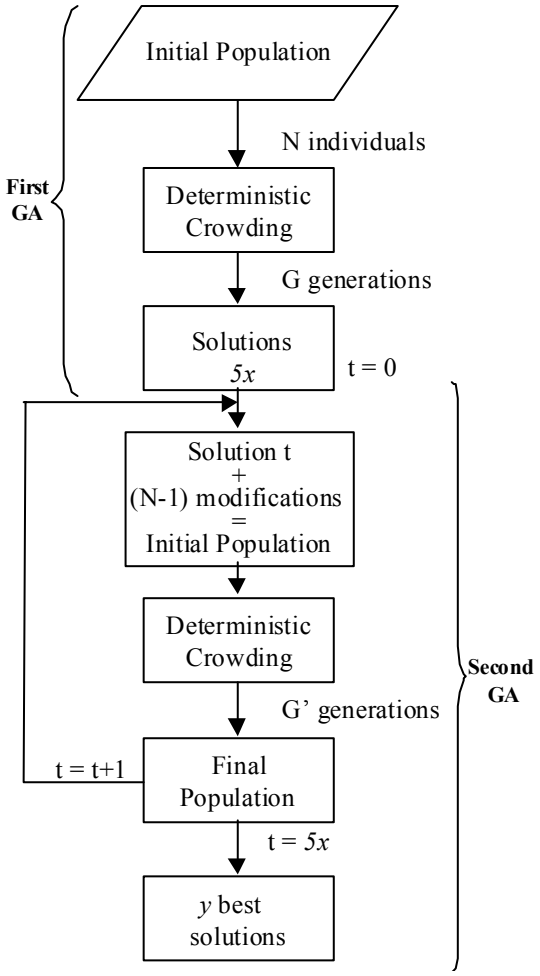


**Figure 11: Global process of the two concatenated DCGAs**

The codification process implemented in this phase is similar to the one of the previous crowding genetic algorithm. The main difference is that in this phase each integer number represents a plank mould, that is, the set of sleepers that forms a manufacturing line (see Figure 10).

To each DCGA, corresponding to a $x_i$ obtained in phase 1, an initial population is generated, which consists in the solution $x_i$ (say, with K plank moulds) and (*K-1*) possible permutations of the plank moulds. For example, in the Figure 10, we obtained 5 plank moulds [1 2 3 4 5] at the end of phase 1, within each of them the sleepers are already ordered and this order will not be changed. In the second phase, the objective is to group these plank moulds in production series. An example of an initial population will be [1 2 3 4 5] or [2 1 3 4 5] or [3 1 2 4 5].

The objective function of this phase is the movements of the moulds between series and the utilization factor of all manufacturing lines. As mentioned, the mould movements are complicated and time-consuming operations that must be avoided. Therefore, the number of movements of moulds between series is a measure of the quality of the solution. To calculate this value for each solution in this phase, we group the plank moulds in production series taking into account the maximum number of available manufacturing lines. Afterwards, we calculate the number of necessary moulds of each type for each series. If this number is higher than the existing number of moulds, the

solution will be not considered since it is not feasible. Another solution will be generated as in the previous phase. Finally, the number of moulds movements between production series is calculated and the value of the first objective function, or fitness, will be obtained as the percentage of common existing moulds between production series. The lower is this value the better is the solution.

Moreover, we must use a second objective related with the utilization of all manufacturing lines expressed as a percentage: (total number of occupied/total available manufacturing lines). In the example of the Figure 10, this percentage will be 5/6.

The final objective will be the sum of these two objectives.

## 5. Computational experiment

To test the performance of the proposed optimization method and to study the characteristics of the obtained solutions, a computational experiment was performed. This experiment was designed to reflect the real practical situation of the PRECON S.A. company, and done with collaboration of production director and production planner of the company. However, given the agreements with the company, we are only allowed to explain and publish a limited number of instances and results that we will explain next. This instances were selected in a way that represent the type of instances faced by the production planner in the real situation.

The heuristics were implement in C and the test were run in a Pentium 3-1000Mhz. In our experiments we used N equal to 50 individuals and G equal to 10.000. The Order Crossover and Order Based Mutations operators have probability 80% and 20% percentage, and the replacement process based on the distance measure as described previously (see Figure 9). The user can choose between 1 to 50 solutions of the first phase to initiate the second phase.

Next, we describe in detail the characteristics and data of two instances, and afterwards we will comment on the results obtained. Note that both instances are obtained from the company and we use all the time real data.

The first instance considers a set of sleepers that must be manufactured for a certain month (in this case May, 2000). This instance is composed of three different configurations of sleepers. See Tables 3 and 4 for the description of the data of the instance. The dark shaded rows indicate the sleepers that have a mould constraint.

In the Table 4, the first column shows the number of different sleepers with the characteristics: number of available moulds (constraint) and number of necessary units of each one of these sleepers. For example, the first row shows us that we must fabricate 2 units of each one of these 12 different sleepers which have only 1 available mould of them, that is, there exist a constraint to manufacture these 12 sleepers.

| Type (Configuration) | 1 | 2 | 3 |
|---|---|---|---|
| **Meters** | 286.42 | 1554.18 | 913.60 |
| **Sleepers Number** | 89 | 461 | 352 |

**Table 3: Total meters and number of sleepers of each configuration**

| | Sleepers | Number of available moulds | Units of each sleeper |
|---|---|---|---|
| **Type 1** | 12 | 1 | 2 |
| | 18 | 1 | 3 |
| | 1 | 5 | 2 |
| | 1 | 2 | 4 |
| | 1 | 5 | 5 |
| **Type 2** | 96 | 1 | 1 |
| | 103 | 1 | 2 |
| | 3 | 1 | 3 |
| | 31 | 1 | 4 |
| | 2 | 2 | 2 |
| | 1 | 10 | 20 |
| | 1 | 3 | 1 |
| | 1 | 2 | 1 |
| **Type 3** | 1 | 5 | 1 |
| | 1 | 78 | 351 |

**Table 4: Characteristic of the first instance**

The second instance is a very specific problem that the enterprise has to face many times since this instance is related with the construction of railway siding for AVE. In this case, the sleepers are only of type 3 (AVE). See Table 5 and 6 for the description of the data of this instance.

| Type (Configuration) | 3 |
|---|---|
| **Metres** | 7591.19 |
| **Sleepers Number** | 3095 |

**Table 4: Total meters and number of sleepers for the AVE problem**

| | Sleepers | Number of available moulds | Number of necessary sleepers |
|---|---|---|---|
| **Type 3** | 1 | 1 | 20 |
| | 338 | 1 | 5 |
| | 1 | 2 | 10 |
| | 1 | 2 | 20 |
| | 1 | 4 | 85 |
| | 1 | 5 | 50 |
| | 1 | 5 | 70 |
| | 1 | 6 | 55 |
| | 1 | 15 | 165 |
| | 1 | 17 | 125 |
| | 1 | 78 | 805 |

**Table 5: Characteristic of the AVE instance**

### 6.1. Results for the First Instance

In Table 6, we present a summary of the results for the first instance. As we can see, for the sleepers of type one and two, the planner did not ever used the four available manufacturing lines at any time. Due to the moulds constrains, the planner used three series with only one manufacturing line, and ten series with two manufacturing lines, that is, 13 series and 23 plank moulds. For the third configuration of the AVE, the real needs are 351 units of the same sleeper of which we have 78 moulds. To fill up one plank mould with this sleeper is needed 38 moulds, that is, in each plank mould we can produce 34 equals sleepers, so we can produce these 351 sleepers in 9 plank moulds. The different sleeper can be contained in any of these 9 plank moulds.

| Dates | Manufacturing lines for 1 and 2 types | | | | Manufacturing lines for AVE | | | |
|---|---|---|---|---|---|---|---|---|
| | L1 | L2 | L3 | L4 | L1 | L2 | L3 | L4 |
| 25-4-2000 | | | | | X | X | | |
| 27-4-2000 | | | | | X | X | | |
| 28-4-2000 | | | | X | | | | |
| 2-5-2000 | | | | X | | | | |
| 3-5-2000 | | | | | X | X | | |
| 4-5-2000 | | | X | X | | | | |
| 5-5-2000 | X | X | | | | | | |
| 8-5-2000 | | | X | X | | | | |
| 9-5-2000 | X | X | | | | | | |
| 11-5-2000 | | | X | X | | | | |
| 12-5-2000 | X | X | | | | | | |
| 13-5-2000 | | | | X | | | | |
| 15-5-2000 | | | X | X | X | X | | |
| 17-5-2000 | X | X | | | | | | |
| 18-5-2000 | | | X | X | | | | |
| 19-5-2000 | | X | | X | | X | | |

**Table 6: Real dates of the first instance**

The convergence for the third configuration (AVE) was obtained in the 3 or 4 generation in the first phase (for the 12 runs performed). However, the convergence for the one and two configurations was more slow due to the moulds constrains, being needed the 10.000 iterations for the first phase and 300 iterations for the second phase.

For the first phase, the plank moulds are filled up with the objective of maximizing their utilization. The average best percentage obtained for 12 replicas is shown in the Table 7. We remember that the four best solutions obtained by the first phase after 1000, 3000, 5000, 7000 and 10000 generations will be used as start point of the second phase, where the plank moulds will be allocated in the series. For this, we will consider the best 20 solutions for the second phase.

| Number Generations | Type 1 | | Type 2 | | Type 3 | |
|---|---|---|---|---|---|---|
| | Av. | Var. | Av. | Var. | Av. | Var. |
| 100 | 0.5458 | 0.0002 | 0.9406 | 0.0001 | 0.99 | 0.000 |
| 200 | 0.6031 | 0.0004 | 0.9615 | 0.0000 | 0.99 | 0.000 |
| 400 | 0.6681 | 0.0005 | 0.9762 | 0.0000 | 0.99 | 0.000 |
| 600 | 0.7219 | 0.0015 | 0.9815 | 0.0000 | 0.99 | 0.000 |
| 800 | 0.7617 | 0.0019 | 0.9841 | 0.0000 | 0.99 | 0.000 |
| **1000** | **0.7960** | 0.0020 | **0.9854** | 0.0000 | **0.99** | 0.000 |
| 1200 | 0.8176 | 0.0015 | 0.9864 | 0.0000 | 0.99 | 0.000 |
| 1400 | 0.8345 | 0.0014 | 0.9872 | 0.0000 | 0.99 | 0.000 |
| 1700 | 0.8552 | 0.0015 | 0.9880 | 0.0000 | 0.99 | 0.000 |
| 2000 | 0.8698 | 0.0018 | 0.9886 | 0.0000 | 0.99 | 0.000 |
| **3000** | **0.8954** | 0.0022 | **0.9897** | 0.0000 | **0.99** | 0.000 |
| 4000 | 0.9087 | 0.0026 | 0.9904 | 0.0000 | 0.99 | 0.000 |
| **5000** | **0.9155** | 0.0028 | **0.9909** | 0.0000 | **0.99** | 0.000 |
| 6000 | 0.9190 | 0.0029 | 0.9912 | 0.0000 | 0.99 | 0.000 |
| **7000** | **0.9218** | 0.0030 | **0.9915** | 0.0000 | **0.99** | 0.000 |
| 8000 | 0.9242 | 0.0031 | 0.9917 | 0.0000 | 0.99 | 0.000 |
| 9000 | 0.9263 | 0.0032 | 0.9919 | 0.0000 | 0.99 | 0.000 |
| **10000** | **0.9284** | 0.0033 | **0.9921** | 0.0000 | **0.99** | 0.000 |

**Table 7: Average best percentages of filling for each type of configuration (12 replicas)**

The results for the 1 and 2 types are summarized in table 8.

| Generation Number | Average Percentage | Average moulds | Group 1 | Group 2 | Group 3 | Group 4 | Series |
|---|---|---|---|---|---|---|---|
| 1000 | 46.35 | 74.73 | 3.54 | 5.28 | 1.75 | 0.16 | 10.73 |
| 3000 | 45.51 | 77.01 | 3.80 | 5.14 | 1.60 | 0.17 | 10.71 |
| 5000 | 45.32 | 77.79 | 3.87 | 5.07 | 1.60 | 0.17 | 10.71 |
| 7000 | 44.99 | 74.80 | 4.06 | 4.93 | 1.58 | 0.20 | 10.77 |
| 10000 | 44.70 | 74.33 | 3.93 | 5.30 | 1.47 | 0.13 | 10.82 |

**Table 8: Average results of all solutions (20 solutions for 12 replicas), first instance**

The column of average percentage shows the average occupation of the used series. The column of average moulds shows the average number of common moulds between series. The groups show the number of series with: only one occupied manufacturing line (group 1); two occupied manufacturing line (group 2); three (group 3); and four, all occupied manufacturing lines (group 4). The column of series shows us the number of different necessary series to fabricate all plank moulds obtained in second phase.

Just as we have commented before, if the percentage of utilization within each plank mould is higher, then the difficulty to fill the four available manufacturing lines in each series will be higher. Thus, with the obtained solutions after the 5000 generation the results are better than with the last solutions (10.000 generations). The number of manufacturing lines that are used within each series is lesser and therefore, the number of common moulds also decreases.

However, as the second phase of the DCGA is directed by two objectives, it seems more reasonable to consider only the non-dominated 20 best solutions. In this case, the results are shown in the Table 9.

| Generation Number | Average Percentage | Average moulds | Group 1 | Group 2 | Group 3 | Group 4 | Series |
|---|---|---|---|---|---|---|---|
| 1000 | 49.39 | 85.97 | 2.76 | 5.07 | 2.01 | 0.26 | 10.10 |
| 3000 | 47.66 | 89.94 | 3.16 | 5.02 | 2.04 | 0.09 | 10.30 |
| 5000 | 47.98 | 87.88 | 2.81 | 5.47 | 1.75 | 0.16 | 10.16 |
| 7000 | 47.16 | 85.61 | 3.64 | 4.49 | 2.00 | 0.24 | 10.37 |
| 10000 | 46.57 | 84.47 | 2.98 | 5.84 | 1.67 | 0.00 | 10.43 |

**Table 9: Average results of non-dominated solutions (12 replicas), first instance**

As we can see, if we take only into account the non-dominated solution, the average quality will be higher, and the number of series will be lesser.

For example, in the table 10 we show one solution with the quality similar to the average in the 5000 generation of the first phase of the DCGA and 300 average generation in the second phase of the DCGA, other solution with the lesser number of series and plank mould used (Best), and others two with only two manufacturing lines used as in the real dates (Two lines):

| Solution | Average Percentage | Average moulds | Group 1 | Group 2 | Group 3 | Group 4 | Series | Plank mould |
|---|---|---|---|---|---|---|---|---|
| Average | 47 | 93 | 5 | 1 | 4 | 0 | 10 | 19 |
| Best | 52 | 108 | 2 | 4 | 3 | 0 | 9 | 19 |
| Two lines | 50 | 65 | 0 | 10 | 0 | 0 | 10 | 20 |
| | 47 | 104 | 1 | 9 | 0 | 0 | 10 | 19 |

**Table 10: Results of a representative solution.**

The number of necessary days to manufacture all sleepers obtained from the solution of the DGCA is 9, against the 13 that the enterprise needed before the implementation of the method. the official salary of the different works in the plant is:

| Category | Number of persons | Salary |
|---|---|---|
| Specialized unskilled | 2 | 10.9 €/h. |
| Assistant of plant | 1 | 12.0 €/h. |
| First Officer | 1 | 12.9 €/h. |

**Table 11: Salary of the works**

In this way, the cost of the daily salary is 373.6 €. Therefore, the results of the DCGA proposed to solve the Concrete Sleepers Production Scheduling Problem allow us an approximate salary saving of 1500 €. However, this is not the only saving possible. We must take into account that the decrease of the number of realized plank moulds is associated to the decrease of the number of used steel bar. In our case, the steel has a diameter of 55 mm and a cost per lineal meter of 0.75 €. As in each plank mould we use 9 or 12 bar of the 104 metres, the real fixed cost in steel bar is about 13600 €. With the solution proposed my the DCGA method the fixed cost will be about 12000 €. That represents a saving of 1600 €. Therefore the total saving of salary and steel bar is about 3100 €.

### 6.2. Results for the Second Instance

In this case, the problem is a very common one in the company, it consists on five units of the same siding.

The convergence of the evolution in this instance is very fast (Table 12). After 1000 generations the filling percentage is of the 90.4% and the increase after 6000 generations is the 0.97%. For this, the fixed generation number is less than for the first instance.

The enterprise expects about to obtain less than 78 plank moulds for this problem. In this case, the second optimization phase (allocating these plank moulds in series) is not necessary, and so, we have only to run the first phases, i.e. the first DCGA. The number of solutions, which number of plank moulds is 77, 76 and 75 for each 1000 generations is shown in the table 13.

| Number Generations | Type 3 | |
|---|---|---|
| | Av. | Var. |
| 1-1000 | 0.9042 | 0.0738 |
| 1001-2000 | 0.9087 | 0.0745 |
| 2001-3000 | 0.9106 | 0.0748 |
| 3001-4000 | 0.9051 | 0.0812 |
| 4001-5000 | 0.9122 | 0.0751 |
| **5001-6000** | 0.9126 | 0.0752 |
| 6001-7000 | 0.9130 | 0.0752 |

**Table 12: Filling percentage for the second instance**

| | Generations | | | | | | |
|---|---|---|---|---|---|---|---|
| **Plank Moulds** | **1000** | **2000** | **3000** | **4000** | **5000** | **6000** | **7000** |
| **75** | 11.50 | 15.58 | 26.41 | 33.41 | 37.33 | 40.50 | 42.08 |
| **76** | 36.50 | 33.91 | 23.50 | 16.58 | 12.66 | 10.36 | 8.63 |
| **77** | 8.45 | 1.50 | 0.50 | 0.00 | 0.00 | 0.00 | 0.00 |
| **Av.** | 75.94 | 75.72 | 75.48 | 75.33 | 75.25 | 75.20 | 75.17 |

**Table 13: Average number of necessary plank moulds (12 replicas)**

We can see that after 1000 generations the need of the enterprise has been accomplished. All of the solutions in the population have a number of plank moulds less than 78, being very important the evolution of the number of solutions with 75 plank moulds. These solutions have never been obtained before in the company with the old scheduling system.

In last row, the average number of necessary plank moulds for the 50 solutions of the 1000, 2000, ..., 7000 generations are shown. We see as this number decreases according the process evolves.

The main cost saving is refereed to the number of steel bar used in the fabrications of the plank moulds. The salary saving is not known since we do not obtain how many series are necessary to do for this solution.

## 7. Concluding Remarks

In this work, a complex real production problem from the concrete industry is described. The problem consists in scheduling the concrete sleepers production of a large Spanish company, PRECON S.A., where the basic constraints are related with the quantity of available moulds. The main objectives of the company when scheduling are the maximization of manufacturing lines utilization and the minimization of the moulds movements.

We propose a heuristic solution method based on the Deterministic Crowding Genetic Algorithms. Given the good results obtained by this technique to other scheduling problems, the complexity of the problem and the presence of multiobjective functions, we believe that the genetic algorithms can be an appropriate technique to solve the concrete sleepers production scheduling problem. The DCGAs can also handle in a simple way different constraints that can be imposed by the company, as well the introduction of new resources and products.

The results obtained by the crowding genetic algorithm improve the obtained with the previous scheduling system of the company. Therefore, the production director of PRECON was very satisfied with the approach developed and explained in this paper. We could also proved that this improvement arises at cost saving of salary and utilized resources that allow increasing the profit of the enterprise and accomplishing the global strategy of rise and quality of PRECON.

# References


Bagchi, T.P. (1999). *Multiobjective scheduling by genetic algorithms*. Kluwer Academic Publishing.

Baker, J.E. (1985). "Adaptive selection methods for genetic algorithms". *Proceedings of the first International Conference on Genetic Algorithms and their applications*. John J. Gefenstette (Ed.). Lawrence Erlbaum. New Jersey, (101-111).

Beasley D., D.R. Bull, R.R. Martin (1993). "A sequential niche technique for multimodal function optimization". *Evolutionary Computation*, **1**:2 (101-125).

Brucker (1998). *Scheduling Algorithms*. Springer-Verlag.

Chrétienne P., E.G. Coffman, Jr., J.K. Lenstra and Z. Liu, editors (1995). *Scheduling Theory and its Applications*, John Wiley & Sons.

Davis L. (1989). "Adapting operator probabilities in genetic algorithms". *Proceedings of the third international conference on Genetic Algorithms*. J. David Schaffer (Ed.). Kaufmann. San Mateo (375-378).

Davis L. ed. (1991). *Handbook of Genetics Algorithms*, Van Nostrand, (New York).

K. Deb, D.E. Goldberg, "An investigation of niche and species formation in genetic function optimization", *Proceedings of the Third International Conference on Genetic Algorithms*, J. David Schaffer, Ed., Kaufmann, San Mateo, (1989), 42-50.

Deb, K. (2001). *Multi-objective optimization using evolutionary algorithms*. John Wiley.

Escalada-Imaz G., R. Torres-Velázquez (1997). "Algoritmos genéticos genéricos y basados en orden. Conceptos fundamentales y mecanismo de base", *Inteligencia Artificial* **3**.

Fang H. (1994). *Genetic Algorithms in Timetabling and Scheduling*, Doctoral dissertation, Department of Artificial Intelligence, University of Edinburg.

Fogel D.B. ed. (1998). *Evolutionary computation. The fossil record. (Selected readings on the history of evolutionary computation)*, IEEE Press.

French S. (1982). *Sequencing and scheduling: and introduction to the mathematics of the job shop*. Ellis Horwood.

Glover F. and M. Laguna (1997). *Tabu search*. Kluwer Academic.

Goldberg D. and J. Richardson (1987). "Genetic algorithms with sharing for multimodal function optimization". *Proceedings of the second international conference on Genetic Algorithms*. Grefensette, J. (Ed.). Lawrence Erlbaum. (41-49).

Goldberg D.E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley.

Grefenstette J., R. Gopal, B. Rosmaita, D.V. Gucht (1985). "Genetic algorithms for the traveling salesman problem", *Proceedings of the First International Conference on Genetic Algorithms and Their Applications*, John J. Grefenstette (ed.). Lawrence Erlbaum, New Jersey (160-168).

Horn J., N. Nafpliotis, and D. Goldberg (1994). "A niched pareto genetic algorithm for multiobjetive optimization". *Proceedings of the first IEEE Conference on Evolutionary Computation*, 1 ( 82-87).

Lourenço H.R., O.C. Martin, and T. Stützle (2003) *Iterated Local Search*. in the Handbook of Metaheuritics book, F. Glover and G. Kochenberger (eds), Kluwer (321- 353).

Kirkpatrick S., C.D. Jr. Gelatt, and M.P. Vecchi (1983). "Optimization by simulated annealing". *Science*, 220 (671-680).

P.J.M. Van Laarhoven, E.H.L. Aarts, J.K. Lenstra (1992). "Job shop scheduling by simulated annealing", *Operations Research*, **40** (112-129).

Mahfoud S.W. (1992). "Crowding and preselection revisited", *Parallel Problem Solving from Nature II*, R. Männer, B. Manderick, (eds.), Elsevier, Amsterdam (27-36).

Mahfoud S.W. (1995). *Niching methods for genetic algorithms*. Ph. D. Dissertation, Univ. of Illinois, Urbana-Champaign.

Mattfeld D.C. (1995). *Evolutionary search and the job shop. Investigations on genetic algorithms for production scheduling*. Springer.

Morton and Pentico (1993). *Heuristic Scheduling Systems,* John Wiley & Sons.

Michalewicz Z. (1995). *Genetic Algorithms + Data Structures = Evolution Programs*, Springer, (Berlin).

Oei C.K., D.E. Goldberg and S.J. Chang (1991). *Tournament selection, niching and the preservation of diversity*. IlliGAL Report Nº 91011. University of Illinois at Urbana-Champaign.

Pérez, E., Herrera, F. and Hernández, C. (2001). "Finding multiple solutions in job shop scheduling by niching genetic algorithms". Technical Report #DECSAI-010110, Dept. of Computer Science and Artificial Intelligence, University of Granada, Spain.

Pétrowski A. (1996). "Clearing procedure as a niching method for genetic algorithms. *In Proc. 1996 IEEE Int. Conf. Evolutionary Computation*, Nagoya, Japan (798-803).

Pinedo M. and X. Chao (1999). *Operations Scheduling with Applications in Manufacturing and Services*. McGraw Hill.

Van Laarhoven P.J.M., E.H.L. Aarts, and J.K. Lenstra (1992). "Job shop scheduling by simulated annealing". *Operations research* 40 (112-129).

Whitley L.D., T. Starkweather, D'A. Fuquay(1989). "Scheduling problems and travelling salesmen: The Genetic Edge Recombination", *Proceedings of the Third International Conference on Genetic Algorithms*, J. David Schaffer (ed.), Kaufmann, San Mateo (133-140).

Wang, L., and Zheng, D-Z. (2001). An effective hybrid optimization strategy for job-shop scheduling problems. *Computers & Operations Research*, 28 (585-596).