Chapter from the book *Advances in Evolutionary Algorithms*
Downloaded from:
http://www.intechopen.com/books/advances_in_evolutionary_algorithms

Interested in publishing with InTechOpen?
Contact us at book.department@intechopen.com
Design of an Efficient Genetic Algorithm to Solve the Industrial Car Sequencing Problem

A. Zinflou¹, C. Gagné² and M. Gravel²

¹Département des sciences appliquées, Université du Québec à Chicoutimi,
²Département d’informatique et de mathématique, Université du Québec à Chicoutimi,
Canada

1. Introduction

In many industrial sectors, decision makers are faced with large and complex problems that are often multi-objective. Many of these problems may be expressed as a combinatorial optimization problem in which we define one or more objective functions that we are trying to optimize. Thus, the car sequencing problem in an assembly line is a well known combinatorial optimization problem that cars manufacturers face. This problem involves scheduling cars along an assembly line composed of three consecutive shops: body welding and construction, painting and assembly. In the literature, this problem is most often treated as a single objective problem and only the capacity constraints of the assembly shop are considered (Dincbas et al., 1988). In this workshop, each car is characterized by a set of different options and the workstations where each option is installed are designed to handle a certain percentage of cars requiring the same options. To smooth the workload at the critical assembly workstations, cars requiring high work content must be dispersed throughout the production sequence. Industrial car sequencing formulation subdivides the capacity constraints into two categories, that are the capacity constraints linked to the high-priority options and the capacity constraints linked to the low-priority options.

However, the reality of industrial production does not only take into account the assembly shop requirements. The industrial formulation proposed by French automobile manufacturer Renault, in the context of the ROADEF 2005 Challenge, also takes into account the paint shop requirements. In this workshop, the minimization of the amount of solvent used to purge the painting nozzles for colour changeovers, or when a known maximum number of vehicle bodies of the same colour have been painted, is an important objective to consider. Indeed, long sequences of cars of the same colour tend to render visual quality controls inaccurate. To ensure this quality control, the number of cars of the same colour must not exceed an upper limit.

The industrial car sequencing problem (ICSP) is thus a multi-objective problem in nature, with three conflicting objectives to minimize. In the assembly shop, one tries to minimize the number of violations of capacity constraints related to high-priority options (HPO) and to low-priority options (LPO). In the paint shop, one tries to minimize the number of colour changes (COLOUR). In the 2005 ROADEF Challenge, the Renault automobile manufacturer proposes to tackle the problem by treating the three objectives lexicographically.
Among the resolution methods proposed by the participants of the challenge, one finds essentially neighbourhood search methods as simulated annealing, iterative tabu search, iterative local search and variable neighbourhood search (Briant et al., 2007; Cordeau et al., 2007; Estellon et al., 2007; Ribiero et al., 2007a; Gavranović, 2007; Benoist, 2007), an ant colony optimization algorithm (ACO) (Gagné et al., 2006) and a genetic algorithm (GA) (Jaszkiewicz et al., 2004). Since the work of all the participating teams was not published, the previous enumeration is not exhaustive. After the challenge, other authors proposed to solve the problem using an integer linear programming model (Estellon et al., 2005; Gagné et al., 2006; Prandtstetter and Raidl, 2007), an algorithm hybridizing variable neighbourhood search and integer linear programming (Prandtstetter and Raidl, 2007) or an iterative local search approach (Ribeiro et al., 2007b).

One may note that few authors proposed GAs to solve this multi-objective problem, except for Jaszkiewicz et al. (Jaszkiewicz et al., 2004). Moreover, this team was not amongst the twelve finalists of the 2005 ROADEF Challenge that included 55 teams from 15 countries at the beginning. As for the ICSP, one may only find the GAs proposed by Warwick and Tsang (1995), Terada et al. (2006) and Zinflou et al. (2007) in the literature for the standard version of the car sequencing problem. Among them, only Zinflou et al. (2007) succeeded in proposing an efficient GA, suggesting that this metaheuristic is not well suited to deal with the specificities of this problem.

The main purpose of this chapter is to show that GAs can be efficient approaches for solving the ICSP when the different mechanisms of the algorithm are specially design to deal with the specificities of the problem. To achieve this, we present the different choices made during the design of the genetic operators. In particular, we propose two new crossover operators dedicated to the multi-objective characteristic of the problem. The performance of the proposed approaches is assessed experimentally using the different instances of the 2005 ROADEF Challenge and compared with the best results obtained during the challenge.

This chapter is organized as follows: Section 2 briefly defines the industrial car sequencing problem and Section 3 describes the new crossover operators proposed for this multi-objective problem. The basic features of the proposed GA are presented in Section 4. Section 5 is dedicated to computational experiments and comparisons with previous results from literature. Finally, the conclusion of this research work is given in Section 6.

2. The industrial car sequencing problem

This section provides the main elements to describe the ICSP. The reader may consult Nguyen & Cung (2005) and Solnon et al. (2007) for a complete description of the problem. On each production day, customer orders are sent in real time to the assembly plant. The daily task of the planners is then: (1) to assign a production day to each ordered vehicle, according to production line capacities and delivery dates that were promised to customers; and (2) to schedule the cars within each production day while satisfying as many of the requirements as possible of the three manufacturing workshops, as illustrated in Figure 1. The sequence thus found is then applied to the whole assembly line.

In the definition of ICSP proposed during the 2005 ROADEF Challenge, the Renault car manufacturer stated that technologies used in the plants are such that the body shop does not set requirements for the daily schedule. The ICPS then consists in scheduling a set of cars (Nb_cars) for a production day taking into consideration the paint shop and assembly shop requirements.
In the paint shop, production scheduler tries to group cars by paint colour to minimize the number of colour changes. Painting nozzles must be purged with solvent when changing car colour, or after a maximum number of cars ($r_{\text{max}}$) painted the same colour, to ensure quality. Each purge requires a colour change. Then, each solution with more consecutive cars than $r_{\text{max}}$ to be painted the same colour must be considered unfeasible.

In the assembly shop, many elements are added to the painted body to complete the car assembly. Each car is characterized by a set of different options $O$ for which the workstations, where these options are installed, are designed to handle up to a certain percentage of the cars requiring the same options. These capacity constraints may be expressed by a ratio $r_0/s_0$, that means that any consecutive subsequence of $s$ cars must include at most $r$ cars with option $o$. Cars requiring the same configuration of options must be dispersed throughout the production sequence to smooth out the workload at various critical workstations. If, for a subsequence of length $s$, it is impossible to satisfy the capacity constraint for option $o$, the number of cars that exceeds $r$ defines what is called conflicts or violations. As mentioned previously, the ICSP subdivides the capacity constraints of the assembly shop into two groups; the constraints related to the high-priority options and those related to the low-priority options. In this shop, production scheduler tries to optimize two different objectives: the number of capacity constraint violations related to the high-priority options (HPO) and the number of capacity constraints violations related to the low-priority options (LPO).

We choose to cluster the cars requiring the same configuration of high-priority and low-priority options into $V$ car classes, for which we know the exact number to produce ($c_v$). These quantities represent the production constraints of the problem. Table 1(a) shows an example of the industrial problem for producing 25 cars ($\text{Nb}_\text{cars}$) having 5 options ($O$) with 6 car classes ($V$) and a possibility of 4 different colours across each class. One defines a production sequence $Y$ by two vectors representing respectively the car classes ($\text{Classes}$) and the car colour codes ($\text{Colours}$) as shown in Figure 1(b). A production sequence will be designated by $Y = \{\text{Classes}/\text{Colours}\}$ in the remainder of the chapter and the element at position $i$ of the sequence will be defined by $Y(i) = \text{Classes}(i)/\text{Colours}(i)$.

Another interesting feature of the ICSP is that it links the different production days. Thus, the evaluation of a solution must take into account the end of the previous production day and must extrapolate the minimum number of conflicts generated with the next production day. Similarly, a colour change will be added if the colour of the first car of the current day is different from the colour of the last car of the previous day.

To evaluate the number of conflicts for each option, we first construct binary matrix $S$ of size $O \times \text{Nb}_\text{cars}$ using solution vector $Y$. We have $S_{oi} = 1$ if the class of car assigned to position $i$ of the solution vector requires option $o$, otherwise it is equal to 0. The decomposition of the
Classes vector of solution Y from Table 1 into its different options to obtain S is given in Table 2. In Table 2(a), we also report the end of the previous production day sequence to allow to evaluate the number of conflicts related to the link of these two production days. In Table 2(b), we also evaluate the solution based on the next day, assuming cars without any option.

| Class # | o | r | s | 1 | 2 | 3 | 4 | 5 | 6 |
|---------|---|---|---|---|---|---|---|---|---|
| 1       | 1 | 1 | 2 | 0 | 1 | 1 | 0 | 0 | 0 |
| 2       | 2 | 5 | 1 | 0 | 1 | 0 | 1 | 1 |
| 3       | 1 | 3 | 0 | 1 | 0 | 0 | 0 | 0 |
| 4       | 3 | 5 | 0 | 0 | 0 | 1 | 0 | 1 |
| 5       | 2 | 3 | 0 | 1 | 1 | 0 | 1 | 0 |

\[ c_v \]

| Class # | 1 | 2 | 1 | 1 | 2 | 1 | 1 |
|---------|---|---|---|---|---|---|---|
| 2       | 1 | 1 | 0 | 2 | 1 | 1 |
| 3       | 3 | 1 | 3 | 2 | 0 | 0 | 2 |
| 4       | 4 | 1 | 0 | 1 | 0 | 1 | 0 |

Table 1. Example and solution of an ICSP

| Y | 1 | 2 | 3 | 4 | 5 | 6 | …… | 21 | 22 | 23 | 24 | 25 |
|---|---|---|---|---|---|---|-----|-----|-----|-----|-----|-----|
| Classes | 3 | 5 | 5 | 4 | 6 | 4 | 6 | 4 | 3 | 1 | 4 | 5 | 1 |
| Colours | 4 | 4 | 2 | 2 | 2 | 2 | 3 | 3 | 1 | 1 | 1 | 1 |

Table 2. Evaluation of the solution shown in Table 1
For the current production day $D$, options 1, 3 and 4 do not cause any violation in this part of the solution. Indeed, for each of these three options, we never have a subsequence of size $s$, with more than $r$ cars with the option. However, for option 2, there are two conflicts located between positions 1 to 5 since we have 4 cars having the option while the capacity constraint limits the maximum to 2. In addition, there is one conflict located between positions 2 to 6, another conflict between positions 20 to 24 and two other conflicts between positions 21 to 25, since capacity constraint 2/5 is not satisfied. For option 5, we also have one conflict as capacity constraint 2/3 is not satisfied between positions 1 to 3.

For the link with previous production day $D-1$, we have one conflict located between positions -1 to 1 for option 1, two conflicts located between positions -2 to 3 and positions -1 to 4 for option 2, and another conflict between positions -1 to 2 for option 5. For the link with next production day $D+1$, we only have one conflict located between positions 22 to 26 for option 2.

Considering that the first three options are high-priority and that the other two are low-priority, we therefore have 10 conflicts for the HPO objective and 2 conflicts for the LPO objective for this solution $Y$. Then, we only have to count the number of colour changes (COLOUR) to complete the evaluation of solution $Y$.

The 2005 ROADEF Challenge proposed to tackle the problem using a weighted sum method that assigns different weights $w_1$, $w_2$ and $w_3$ to each objective according to their priority level, in order to evaluate a solution $Y$. The quality of solution $Y$ is then given by:

$$F(Y)=w_1*obj_1+w_2*obj_2+w_3*obj_3$$

where $obj_1$, $obj_2$ and $obj_3$ correspond respectively to the values obtained for a solution $Y$ on each objective according to the priority level assigned. The weights $w_1$, $w_2$ and $w_3$ are respectively set at 1000000, 1000 and 1 (Nguyen & Cung, 2005). According to the different configurations of the Renault plants, the three following objective hierarchies are possible: HPO-COLOUR-LPO, HPO-LPO-COLOUR and COLOUR-HPO-LPO.

3. Introducing problem knowledge in crossover design for the industrial car sequencing problem

Traditional crossover operators are not well suited to deal with the specificities of the car sequencing problem. Indeed, Warwick and Tsang (1995), and Terada et al. (2006) used such operators to solve the single objective car sequencing problem found in the literature and their results were not competitive. However, Zinfrou et al. (2007) obtained very competitive results using two highly-specialized crossover operators for the same problem.

For the multi-objective ICSP, Jaszkiewicz et al. (2004) proposed to use a common sequence preserving crossover. Basically, the purpose of this operator is to create an offspring using the common maximum subsequence of the indices of the groups in two given solutions (parents). However, even if the results of this approach are promising, they did not allow the authors to be part of the twelve finalists during the 2005 ROADEF Challenge.

The crossover operators proposed by Zinfrou et al. (2007) for the single objective car sequencing problem, called non-conflict position crossover (NCPX) and interest based crossover (IBX), use problem-knowledge to perform recombination. The concept used by NCPX and IBX crossovers to use problem-knowledge is called interest. The idea behind this concept is to penalize the conflicting car classes, by counting the number of new conflicts caused by the addition of these classes as a cost. Conversely, if the addition of a car class does not cause
new conflicts, then this is counted as a profit equal to the difficulty of class \( D_v \) as proposed by Gottlieb et al. (2003). Basically, NCPX crossover tries to minimize the number of relocated cars by emphasizing non conflict position information from both parents. The IBX crossover, in turn, rather tries to keep the cars in the same area of the chromosome as it occupied with one of the two parents. For more details about these two crossover operators, the reader may consult Zinflou et al. (2007).

The following sections will show how to adapt the two NCPX and IBX crossover operators to the multi-objective ICSP.

### 3.1 Adaptation of the interest calculation for the industrial car sequencing problem

To present the different adaptation of the crossover operators, we must redefine the interest concept to be able to take into account the multi-objective nature of the ICSP. We define the total weighted interest (TWI) to establish if it is interesting to add a car of class \( v \), of colour \( \text{colour} \) at a position \( i \) in the sequence. The total weighted interest is expressed by:

\[
TWI_{v,\text{colour},i} = I_{v,i,HPO} \cdot w_{HPO} + I_{v,i,\text{COLOUR}} \cdot w_{\text{COLOUR}} + I_{v,i,LPO} \cdot w_{LPO}
\]

(2)

where \( w_{HPO}, w_{\text{COLOUR}} \) and \( w_{LPO} \) correspond respectively to the weight of each objective (1000000, 1000 or 1 according to their priority levels) and \( I_{v,i,HPO}, I_{v,i,\text{COLOUR}} \) and \( I_{v,i,LPO} \) correspond to the interest in inserting a car of class \( v \) at the position \( i \) for each objective. The interest concept may be defined according to each objective.

According to Equation 3, the interest \( I_{v,i,\text{COLOUR}} \) to insert a car of class \( v \) at position \( i \) to minimize objective COLOUR is set at 1 if it is possible to complete the current colour subsequence with a car of class \( v \). If it isn’t possible, the interest is set to -1.

\[
I_{v,i,\text{COLOUR}} = \begin{cases} 
1 & \text{if } \text{nb}(v_{\text{colour}(i-1)}) > 0 \text{ and } \text{run}_\text{length} < \text{rl}_{\text{max}} \\
-1 & \text{otherwise}
\end{cases}
\]

(3)

\( \text{nb}(v_{\text{colour}(i-1)}) \) indicates the number of cars of class \( v \) painted the same colour as the car in position \( i-1 \), \( \text{run}_\text{length} \) indicates the size of the consecutive subsequence of cars of the same colour as the car in position \( i-1 \) and \( \text{rl}_{\text{max}} \) indicates the maximum length of a subsequence of the same colour. This notion serves to favour the classes of cars that have the same colour as the car located in the previous position, to lengthen the colour subsequence to the maximum size. Conversely, we penalize the car classes for which the addition implies a colour change. \( I_{v,i,HPO} \) and \( I_{v,i,LPO} \) indicate the interest to insert a car of class \( v \) at position \( i \) in the sequence to minimize objectives HPO and LPO respectively. According to Equation 4, the interest corresponds to the difficulty for class \( v \) if the addition of this class does not cause new conflicts respectively on high-priority options (\( k = \text{HPO} \)) and on low-priority options (\( k = \text{LPO} \)). In the opposite case, we will define the cost that corresponds to the number of new conflicts produced on the high-priority or low-priority options, to discourage the insertion of this class at position \( i \).

\[
I_{v,i,k} = \begin{cases} 
D_{v,k} & \text{if } \text{NbNewConflicts}_{v,i,k} = 0 \\
-Nb\text{NewConflicts}_{v,i,k} & \text{otherwise}
\end{cases}
\]

(4)
**3.2 The multi-objective NCPX crossover operator (NCPXM)**

The NCPXM procedure for the ICSP is inspired by the NCPX crossover proposed for the single objective car sequencing problem (Zinfou et al., 2007) and is carried out in two main steps. Step 1 consists of selecting a parent \(P_1\) and establishing in this chromosome the number of positions that are not part of a conflict for objectives HPO (\(nbpos_{HPO}\)) and LPO (\(nbpos_{LPO}\)) and the number of positions where there is no colour change (\(nbpos_{COLOUR}\)). Then, we randomly select a number \(nbg_s\) between 0 and \(nbpos\) for each objective \(k\) (\(k = HPO, LPO, COLOUR\)). These three numbers are used to determine, for each objective \(k\), the number of "good" genes that will maintain in offspring \(P_1\) the same position they had in \(P_1\). To take into account the priority of the objectives, we must make sure that the number of "good" genes kept for the main objective is greater or equal to the number of "good" genes selected for the secondary objective, and so forth. Once we establish these numbers, starting position \((sPos)\) that is between 1 and \(Nb_{cars}\), is randomly selected in the offspring to be created. The process of copying the good genes of \(P_1\) to the offspring being created starts from \(sPos\) by first considering the main objective. If we reach the end of the chromosome and the number of genes copied for objective \(k\) is less than its corresponding \(nbg_s\), the copy process restarts this time from the beginning of the offspring up to \(sPos - 1\). The same process is repeated for the other objectives, taking into account the already copied genes. Thereafter, the remainder of the genes from \(P_1\) are used to constitute a non-orderly list \(L\) for the cars that must still be placed. We then randomly determine a position \((Pos)\) from which the remaining positions of chromosome \(E_1\) will be completed.

In Step 2, the cars in \(L\) are sorted according to their TWI. In case of a tie in TWI, if one of the cars is in \(P_2\) at the position to be completed, this car is then selected. In the opposite case, we randomly select a car amongst those of equal ranking.

The operation of this cross operator is illustrated in Figure 2 for two parents \(P_1 = [21352446/62224622]\) and \(P_2 = [32621454/26242622]\) with the following objective hierarchy HPO-LPO-COLOUR. Let us assume that the evaluation of \(P_i\) gives 5 positions without conflicts for objective HPO and for objective LPO (expressed by 0 in vectors “conflicts on HPO and LPO” below chromosome \(P_1\)), 4 positions where there is no colour change (expressed by 0 in vector the “colour changes” below chromosome \(P_1\)) and the values for numbers \(nbg_{HPO} = 4\), \(nbg_{LPO} = 2\), \(nbg_{COLOUR} = 1\) and \(sPos = 3\) by random setting. Starting with \(sPos\) and considering objective HPO, we may copy genes 5/2, 4/6, 4/2 and 2/6 in the
Advances in Evolutionary Algorithms

offspring. Repeating the same procedure with LPO, one notes that the three good genes 5/2, 4/2 and 2/6 are already transferred to the offspring, that corresponds to the number of good genes to transfer for this objective. Also, the two good genes 5/2 and 2/6 are already present in the offspring for the COLOUR objective, that corresponds to the number of good genes to transfer for this objective. Genes 1/2, 3/2, 2/4 and 6/2 of P₁ are then used to constitute non-orderly list L. In Step 2, assuming that Pos = 7 and that the TWI calculation places the genes in the order 3/2, 2/4, 6/2, 1/2 with equal TWI value on genes 2/4 and 6/2. We then place 3/2 gene in position 8 and favour placing gene 6/2 in position 3 since it occupies this position in P₂ and genes 2/4 and 1/2 are placed in positions 2 and 5 respectively. In this example, genes 1/2 and 6/2 are directly inherited from P₂ since they have the same position in the second parent. The offspring produced from P₁ and P₂ is then E₁ = {22651443/64222622}.

A second offspring is created similarly, this time starting with parent P₂.

Fig. 2. Schematic of the NCPX<sup>MO</sup> crossover

3.3 The multi-objective IBX crossover operator (IBX<sup>MO</sup>)

The IBX<sup>MO</sup> crossover procedure for the ICSP is inspired by the functioning of the IBX for the single objective car sequencing problem (Zinflou et al., 2007) and proceed in three main steps. Step 1 consists in randomly determining two cut-off points for both parents P₁ and P₂. Once these temporary cut-off points are determined, the colours of the preceding cars at the 1<sup>st</sup> cut-off point and the colour of the cars immediately after the 2<sup>nd</sup> cut-off point in P₁ are verified so as not to interrupt an ongoing colour subsequence. As long as the colour of the cars located before the 1<sup>st</sup> cut-off point is the same as the colour of the car located at the cut-off point, we move the cut-off point to the left. Inversely, as long as the colour of the car at the 2<sup>nd</sup> cut-off point is identical to the colour of the car after that cut-off point, we move the 2<sup>nd</sup> cut-off point to the right.

In Figure 3, once the cut-off points are set for both parents P₁ = {22351446/46222622} and P₂ = {32421465/24662222}, the genes subsequence [351/222] included between the two cut-off points of the first parent (a₁ ∈ P₁) is directly recopied in the offspring. Thereafter, two non-
orderly lists ($L_1$ and $L_2$) are created from subsequence $b_3 = \{32/24\}$ and $b_4 = \{465/222\}$ of $P_2$ and will be used to complete the beginning and the end of offspring $E_1$. However, during this operation, part of the information may be lost by the addition of duplicates. One effect of this process is that the production requirements will not always be satisfied. In the example in Figure 3, we may thus notice that the production constraints for the 2, 3, 4 and 5 car classes are no longer met. To restore all the genes and to produce exactly $c_v$ cars of the $v$ class, replacement of genes $3/2$ and $5/2$ (obtained from $a_1-a_2$) whose number exceeds the production constraints are replaced by genes $4/6$ and $2/6$ (obtained from $a_2-a_1$) whose number is now lower than the production constraints. This replacement is done randomly in the second step to adjust the $L_1$ and $L_2$ lists.

Fig. 3. Schematic of the IBX$^{mo}$ crossover

Finally, the last step consists in rebuilding the beginning and the end of the offspring using the two corrected lists $L_1$ and $L_2$ by using TWI as defined in Equation 2. In both cases, the reconstruction starts from the cut-off point towards the beginning or the end of the offspring, depending on the situation. For example, we calculate the TWI for each car $\in L_1$ to reconstruct the beginning of the offspring. The car class $v$ to place is then chosen deterministically in 95% of the cases and in the remaining 5% of the cases the car class $v$ to be placed is chosen probabilistically using the roulette wheel principle (Goldberg, 1989). The second vector of the solution for this position is then completed by the colour associated to this class. We then remove this class from list $L_1$ and restart the calculations for the next position. The same process is repeated to reconstruct the end of the offspring from list $L_2$.

A second offspring is created by using the same process, but this time starting from parent $P_2$.

4. Genetic algorithm for the industrial car sequencing problem

In this section, we present the complete description of the genetic algorithm (GA) used to solve the multi-objective ICSP.
4.1 Representation of the chromosome
As shown previously in Table 1(b), instead of choosing classical bit-string encoding, that seems ill-suited for this type of problem, a chromosome is represented using two vectors of size Nb_cars corresponding respectively to the class and the colour of the car.

4.2 Creating the initial population
In the proposed implementation, the individuals of the initial population are generated in two ways: 70 % randomly and 30 % using a greedy heuristic based on the concept of interest. Two greedy heuristics are used according the main objective. If the main objective is to minimize the number of colour changes (COLOUR), the greedy heuristic used is greedy_colour. If the main objective is to minimize the number of conflicts on high-priority options (HPO), the greedy heuristic used is greedy_ratio. Figure 4 resumes the operation of these two heuristics. Notice that in both cases, one ensures that the individuals produced are feasible solutions.

| greedy_colour heuristic | greedy_ratio heuristic |
|-------------------------|------------------------|
| 1: Start with an individual Y consisting of the D-I production day cars | 1: Start with an individual Y consisting of the D-I production day cars |
| 2: \(i=1; \) \(run\_length =1\) | 2: \(i=1; \) \(run\_length =1\) |
| 3: \(previous\_colour = Colours(-1)\) | 3: \(previous\_colour = Colours(-1)\) |
| 4: While there are cars to place | 4: While there are cars to place |
| 5: If \(run\_length < n_{max}\) and there remain cars with \(previous\_colour\) then | 5: If \(run\_length = n_{max}\) then |
| 6: \(colour = previous\_colour\) | 6: Exclude the cars for which \(colour = previous\_colour\) from the candidates cars list |
| 7: \(run\_length ++\) | 7: End If |
| 8: Else | 8: For each candidate car class \(v\) |
| 9: Choose randomly \(previous\_colour \neq colour\) | 9: Evaluate the interest \(I_{v,i,HPO}\) of adding a car class \(v\) at position \(i\) |
| 10: \(run\_length = 1\) | 10: End For |
| 11: End If | 11: Choose randomly a number \(rnd\) between 0 and 1 |
| 12: Restricted the choice to the \(m\) car classes having the selected colour | 12: If \(rnd < 0.95\) Then |
| 13: For each of these \(m\) car classes | 13: Choose class \(v\) according to Arg Max \(I_{v,i,Colour}\) |
| 14: Evaluate the interest \(I_{v,i,Colour}\) of adding a car class \(v\) at position \(i\) | 14: In case of a tie, break the tie lexicographically by using the interest of the second objective and then the third objective \((I_{v,i,HPO} or I_{v,i,Colour})\). In case of ties for the 3 objectives, choose a class randomly |
| 15: End For | 15: Else |
| 16: Choose randomly a number \(rnd\) between 0 and 1 | 16: Choose car class \(v\) using the roulette wheel principle |
| 17: If \(rnd < 0.95\) then | 17: End If |
| 18: Choose car class \(v\) according to Arg Max \(I_{v,i,Colour}\) | 18: For the selected car class \(v\), choose colour with Arg Max \(I_{v,Colour}\). In case of a tie, choose colour randomly |
| 19: In case of a tie, choose car class \(v\) randomly | 19: \(Y(i) = v / colour\) |
| 20: Else | 20: If \(run\_length = n_{max}\) or \(colour \neq previous\_colour\) then |
| 21: Choose \(v\) using the roulette wheel principle | 21: \(run\_length= 1\) |
| 22: End If | 22: Else |
| 23: \(Y(i) = v / colour\) | 23: \(run\_length = run\_length +1\) |
| 24: \(i=i+1\) | 24: End If |
| 25: End While | 25: \(previous\_colour = colour\) |
| 26: \(i=i+1\) | 27: End While |

Fig. 4. Greedy construction of an individual us the greedy_colour or greedy_ratio heuristic
Design of an Efficient Genetic Algorithm to Solve the Industrial Car Sequencing Problem

Greedy_colour begins with an initial solution composed of the cars planned the previous production day. In fact, to link with the previous production day, we only need to know the maximum value of $s_i$ for all the options and this value determines the length of the sequence required at the end of the previous day to evaluate the current solution. Then, we initialize the counter for positions $i$ at 1, the length of the current colour subsequence ($run_length$) that is also at 1 and the colour of the last car produced the previous day ($previous_colour$) (lines 2-3). The selection iteration process for the next car to place in the building sequence (lines 4-25) begins by selecting a current colour ($colour$) according to $r_{\text{max}}$ and $previous_colour$ (lines 5-11). Once the colour of the next car to place is determined, we limit the selection process to the $m$ car classes having that colour. At this step, for each of the $m$ classes, we evaluate the interest $I_{v,i,\text{COLOUR}}$ to place a car of class $v$ at the current position $i$. In 95% of the cases, the selected class is the one with the largest $I_{v,i,\text{COLOUR}}$ ($\text{Arg Max } \{ I_{v,i,\text{COLOUR}} \}$). For the remaining 5% of the cases, the car class to place is selected using the roulette wheel principle. Once the colour and the car class are selected, we add the selected car class $v$ and the selected colour at position $i$ of sequence $Y$ being built (line 23). This process is thus repeated until an entire sequence of cars is built. The main purpose of this greedy_colour heuristic is thus to minimize, in a greedy way, the number of colour changes.

The second proposed construction heuristic, called greedy_ratio, also uses a greedy approach to build an individual $Y$. However, for this heuristic, the main greedy criterion used to select the car to add in the next position of sequence $Y$ being built is the interest $I_{v,i,\text{HPO}}$. Just as for the greedy_colour heuristic, the greedy_ratio procedure starts with an initial solution consisting of cars already sequenced the previous production day. We then initialize the various counters and the colour of the previous car produced on day D-1 the same way as for the greedy_colour heuristic. The main loop of the algorithm (lines 4-27) first checks if the maximum length for a subsequence of identical colour, $r_{\text{max}}$, has not been reached. If $r_{\text{max}}$ is reached, we withdraw all the cars of colour $previous\_colour$ from the list of classes that may be added at current position $i$ (list of candidate car classes). This step ensures that the generated solution is feasible. Then, for each candidate car class $v$, we calculate the interest $I_{v,i,\text{HPO}}$ to place a car of class $v$ at the current position $i$ according to the HPO objective. Then, the selection of the next car class to place in the sequence is made in 95% of the cases by selecting the class with the largest $I_{v,i,\text{HPO}}$. Note that in case of a tie for the $I_{v,i,\text{HPO}}$, the tie is broken using the highest interest for the second objective and then the third objective, respectively. In 5% of the cases, the car class to place is selected using the roulette wheel principle. Once the car class is selected, we choose the colour of the car to add from the colours available for this class according to $I_{v,i,\text{COLOUR}}$. If all the colours for this class of cars are of the same interest, we choose a colour randomly. Thereafter, we add the selected car class and colour at position $i$ in sequence $Y$ being built. Finally, we update the various counters ($run\_length$ and $i$) and $previous\_colour$. This process is repeated until a complete sequence of cars is done.

4.3 Selection

Several selection strategies could have been considered in the GA based algorithm to solve the multi-objective ICSP. However, since it is easy to implement and that it is efficient for the standard car sequencing problem (Zinflou et al., 2007), the selection procedure chosen to solve the multi-objective ICSP is a binary tournament selection.
4.4 Mutation operator

According to the objective hierarchy, four mutation operators are used here: reflection, random_swap, group_exchange and block_reflection. Note that these four operators have often been used in the literature for the ICSP to explore the neighbourhood within a local search method (Solnon et al., 2007). For problems with HPO-COLOUR-LPO and HPO-LPO-COLOUR objective hierarchies, the mutation operators used are reflection and random_swap. A reflection consists in randomly selecting two positions and reversing the subsequence included between these two positions. A random_swap simply consists in randomly exchanging the positions of two cars belonging to different classes. For problems with COLOUR-HPO-LPO objective hierarchy, the mutation operators used are the group_exchange and the block_reflection. The group_exchange mutation consists in randomly exchanging the position of two subsequences of consecutive cars painted the same colour. The block_reflection consists in selecting a subsequence of consecutive cars painted the same colour and in inverting the position of the cars included in this subsequence.

4.5 Replacement strategy

The proposed GA is an elitist approach in that it has explicit mechanisms that keep the best solution found during the search process. To ensure that elitism, the replacement strategy used is a \((\lambda + \mu)\) type of deterministic replacement. In this replacement strategy, the parent and offspring populations are combined and sorted and only the \(\lambda\) best individuals are kept to form the next generation.

```
1: Generate randomly or using the two greedy heuristics of the initial population POP₀
2: Evaluate each individual \(Y \in POP₀\) and sort POP₀
3: While no stop criterion is reached
4:   While |\(Q_t\)| < \(N\)
5:     Choose randomly a number \(\text{rnd}\) between 0 and 1
6:     If \(\text{rnd} < p_c\) then
7:       Select parents \(P_1\) and \(P_2\)
8:       Create two offspring \(E_1\) and \(E_2\) using NCPX_MO or IBX_MO crossover
9:       Evaluate the generated offspring
10:   else
11:     Generate random migrant using the greedy heuristic
12:   End If
13: Choose randomly a number \(\text{rnd}\) between 0 and 1
14: If \(\text{rnd} < p_m\) then
15:   Mutate and evaluate the offspring or the migrant
16: End If
17: Add \(E_1\) and \(E_2\) or the migrant to \(Q_t\)
18: End While
19: Sort \(Q_t \cup POP_t\)
20: Choose the first \(N\) individuals of \(Q_t \cup POP_t\) to the next generation \(POP_{t+1}\)
21: \(t = t + 1\)
22: End while
23: Return the best individual found so far
```

Fig. 5. The proposed GA procedure for ICSP

Figure 5 describes the general procedure of our GA for the ICSP. The GA starts building an initial population \(POP₀\) in which each individual \(Y \in POP₀\) is evaluated. Then it performs a
series of iterations called generations. At each generation $t$, a limited number of individuals are selected to perform recombination according to a crossover probability ($p_c$). Notice that, occasionally, a new individual is introduced in the offspring population to maintain diversity and avoid stagnation. This individual called random migrant is created using the greedy heuristic used to create the initial population according to the objective hierarchy of the problem to solve. After the crossover, the generated offspring or the migrant is mutated according to mutation probability ($p_m$). Finally, the current population is updated by selecting the best individuals from the pool of parents ($POP_t$) and offspring ($Q_t$). This process is repeated until a stop criterion is reached.

5. Computational experiments

The GA proposed in this chapter was implemented in C++ and compiled with Visual Studio .Net 2005. The computational experiments were run on a Dell Pentium with a Xeon 3.6 GHz processor and 1 Gb of RAM, with Windows XP. For all the experiments performed, the parameters $N$, $p_c$, $p_m$, $T_{max}$ that represent respectively the population size, crossover probability, mutation probability and time limit allowed for the GA are set at the following values: 5, 0.8, 0.35 and 350 seconds. The small population size and the mutation and crossover probabilities were determined using the theoretical results of Goldberg (1989) and the work of Coello Coello and Pulido (2001). According to these authors, a very small population size is sufficient to obtain convergence, regardless of the chromosome length. Thus, the use of a small population with a high crossover probability allows, on one hand, to increase the efficiency of the GA for the ICSP by limiting the computation time required to evaluate the fitness of each individual. In fact, the evaluation of the fitness of a solution for the ICSP requires considerable computation time. On the other hand, a high crossover probability usually allows better exploration of the search space (Grefenstette, 1986). In addition to the difficulties related to the multi-objective nature of the ICSP, a 600 second time limit was set for a Pentium 4/1.6 GHz/Win2000/1 Go RAM computer for the 2005 ROADEF Challenge. To meet this time limit, we set the running time of our GA at 350 seconds, that corresponds roughly to the time limit defined in the Challenge, considering the differences in hardware.

Three versions of our GA will be used for the numerical experiments. The first version integrates the NCPX$^{MO}$ crossover operator (AG-NCPX$^{MO}$), the second uses the IBX$^{MO}$ crossover operator (AG-IBX$^{MO}$) and the third version integrates the NCPX$^{MO}$ crossover operator with a local search procedure (AG-NCPX$^{MO}$+LS).

5.1 Benchmark problems

The performance of the proposed multi-objective GAs is evaluated using three test suites provided by the Renault car manufacturer and that are available from the Challenge website at: http://www.prism.uvsq.fr/~vdc/ROADEF/CHALLENGES/2005/. The first set (SET A) includes 16 sets of data to sequence 334 to 1314 cars that have from 6 to 22 options that create from 36 to 287 cars classes with 11 to 24 different colours. This set allowed to evaluate the teams during the qualification phase and thus to determine the 18 teams who qualified for the next phase of the Challenge. The second set (SET B) consists of a wide range of 45 instances each consisting of 65 to 1270 cars having from 4 to 25 options, with 11 to 339 cars classes and 4 to 20 different colours. This set was used by the qualified teams to improve and tune their algorithms. Finally, the last set (SET X) consists of 19 instances having from...
65 to 1319 cars to sequence, with 5 to 26 options, 10 to 328 car classes and 5 to 20 different colours. This set remained unknown to the teams until the last phase of the Challenge and was used by the jury to establish the final ranking.

In comparison with the standard car sequencing problem whose largest instances included 400 cars, 5 options and from 18 to 24 car classes, the resolution of the multi-objective ICSP thus represents a large challenge.

5.2 Experimental comparison

To evaluate the performance of the algorithms proposed in this chapter, we compare our results with the best results obtained during the 2005 ROADEF Challenge for the 61 instances of SET A and SET B. All the results of the 2005 ROADEF Challenge are available online from the Challenge website. Thus, Tables 3 to 5 report the comparative results of GA-NCPX\textsuperscript{MO}, GA-IBX\textsuperscript{MO} and GA-NCPX\textsuperscript{MO+LS} with those of the Challenge Winning Team and those of the GLS (Jaszkiewicz et al., 2004) which is the best evolutionary algorithm proposed during the Challenge. The rank of the solution found by each algorithm for the same instance is listed in Tables 3 to 5 and is based on the results of the 18 qualified teams and the results of the three GAs proposed here.

In these tables, we group instances in three categories:

- those for which the main objective is the minimization of the number of conflicts on high-priority options (HPO) and where the requirements for these high-priority options are considered “easy” according to Renault (Table 3);
- those for which the main objective is the minimization of the number of conflicts on high-priority options (HPO) and where the requirements for these high-priority options are considered “difficult” according to Renault (Table 4) ; and
- those for which the main objective is the minimization of the number of colour changes (COLOUR) (Table 5).

Each row of Tables 3 to 5 indicates the name of the instance, the value and the rank of the solution found respectively by the Winning Team, the GLS (Jaszkiewicz et al., 2004), the GA-IBX\textsuperscript{MO}, the GA-NCPX\textsuperscript{MO} and the GA-NCPX\textsuperscript{MO+LS}. The best results obtained for each instance are highlighted in bold in the different tables. It is important to note that as for the Challenge results, the GAs proposed were run once only and what we report is the solution value obtained for this execution. The results reported in the different tables indicate the objectives weighted sum value ($F(X)$) of the solution as calculated in Equation 1.

Table 3 reports the results for instances with “easy” high-priority options according to Renault. These instances have two possible hierarchies that are HPO-LPO-COLOUR or HPO-COLOUR-LPO. By examining the results of Table 3, one may note that GA-NCPX\textsuperscript{MO} outperforms GA-IBX\textsuperscript{MO} for all the instances of SET A and SET B, except for instance 028\_ch2\_S23\_J3 with HPO-COLOUR-LPO objective hierarchy where the two algorithms obtain equal results. These results seem to highlight the superiority of the NCPX\textsuperscript{MO} crossover operator over the IBX\textsuperscript{MO} crossover operator for the ICSP. The best performance of the NCPX\textsuperscript{MO} crossover operator may probably be explained by its ability to use information about non-conflict positions. Thus, this crossover is able to do a better search intensification during the allowed time.

Except for instance 028\_ch2\_S23\_J3 with HPO_LPO-COLOUR objective hierarchy, that is trivially solved by all algorithms, GA-IBX\textsuperscript{MO} ranks between 11\textsuperscript{th} and 19\textsuperscript{th} while GA-NCPX\textsuperscript{MO} ranks between 1\textsuperscript{st} and 17\textsuperscript{th} according to the instances. It should be noted that, contrary to
most algorithms of the Challenge, GA-IBX\textsuperscript{MO} and GA-NCPX\textsuperscript{MO} do not use a local search procedure in their algorithm.

By comparing the results of GA-NCPX\textsuperscript{MO} and GA-IBX\textsuperscript{MO} to those of GLS, one may note for SET A that GLS globally outperforms GA-IBX\textsuperscript{MO} but GA-NCPX\textsuperscript{MO} clearly outperforms GLS. Indeed, GLS outperforms GA-IBX\textsuperscript{MO} for 3 instances of Set A, is worse for one instance while it obtains identical results for the remaining instance. By contrast, GLS is worse than GA-NCPX\textsuperscript{MO} for 4 of the 5 instances of SET A shown in Table 3. These results are confirmed with a few slight differences for the instances of SET B. Thus, GLS outperforms GA-IBX\textsuperscript{MO} for 10 instances, is worse for 7 instances while obtaining identical results for the remaining instance. Compared to GA-NCPX\textsuperscript{MO}, GLS achieves better results for 6 instances, is worse for 8 instances while obtaining identical results for the 4 remaining instances. We may therefore notice a slight advantage for GA-NCPX\textsuperscript{MO} for the instances of SET B with easy high-priority options. These results are very promising considering that GLS is a memetic algorithm, that is, an approach hybridizing GA with local search method.

When we now compare the results of GA-NCPX\textsuperscript{MO} and GA-IBX\textsuperscript{MO} to those of the Winning Team for the 2005 ROADEF Challenge, one may notice that the results of the two proposed GAs are clearly lower than the results of the Winning Team in terms of solution quality. We believe that this gap may be explained by the lack of intensification of the search for this type of approach. By combining GA-NCPX\textsuperscript{MO} with a local search procedure inspired from the one proposed by Estellon et al. (2007) and using the mutation operators presented in Section 4.4 to explore the neighbourhood, we obtain the results shown in the last column of Table 3. We mention here that GA-NCPX\textsuperscript{MO}+LS was executed with the same time limit as the other algorithms presented in this chapter. We observe that adding the local search procedure clearly improves the performance of the algorithm. Indeed, GA-NCPX\textsuperscript{MO}+LS clearly outperforms GA-NCPX\textsuperscript{MO} and achieves competitive results compared to those of the Challenge Winning Team for all instances of SET A with easy high-priority options. In fact, GA-NCPX\textsuperscript{MO}+LS ranks first for all these instances and even finds new minimums for instance 022_3_4 with HPO_COLOUR_LPO objective hierarchy and for instance 25_38_1 with HPO_LPO_COLOUR objective hierarchy. For the instances of SET B, GA-NCPX\textsuperscript{MO}+LS obtains similar results as those of the Challenge Winning Team for 10 of the 16 instances. For the remaining instances, we observe a small gap that comes from the results of the second or the third objective. Indeed, GA-NCPX\textsuperscript{MO}+LS is always ranked between 1\textsuperscript{st} and 3\textsuperscript{rd}, except for instance 064_ch1_S22_J3 with HPO_COLOUR_LPO objective hierarchy where it ranks 7\textsuperscript{th}. Table 4 reports the results obtained by the different algorithms for the instances of SET A and SET B considered by Renault as “difficult “ high-priority options. The two possible objective hierarchies for these instances are HPO-LPO-COLOUR and HPO-COLOUR-LPO.

We may notice again that GA-NCPX\textsuperscript{MO} clearly outperforms GA-IBX\textsuperscript{MO}. Therefore, for the instances of SET A, GA-NCPX\textsuperscript{MO} obtains better results than GA-IBX\textsuperscript{MO} for 6 of the 7 instances while GA-IBX\textsuperscript{MO} is better for the only remaining instance. The results are quite the same for the instances of SET B where, this time, GA-NCPX\textsuperscript{MO} always outperforms GA-IBX\textsuperscript{MO}. GA-IBX\textsuperscript{MO} ranks between 12\textsuperscript{th} and 20\textsuperscript{th} while GA-NCPX\textsuperscript{MO} ranks between 1\textsuperscript{st} and 19\textsuperscript{th} depending on the instances. Despite the fact these two algorithms do not use a local search procedure, they are quite competitive with the global results of the teams that qualified for the Challenge. However, for the instances with easy high-priority options, we notice that the results of the two proposed algorithms are not competitive with those of the Challenge Winning Team.
Table 3. Results of the Winning Team, GLS, GA-IBX\textsuperscript{MO}, GA-NCPX\textsuperscript{MO} and GA-NCPX\textsuperscript{MO}+LS for  "easy" high-priority options instances with HPO as the main objective

If we now compare the performance of GA-IBX\textsuperscript{MO} and GA-NCPX\textsuperscript{MO} with those of GLS, we notice that GLS clearly outperforms GA-IBX\textsuperscript{MO}, both for the instances of SET A and SET B. Thus, GLS obtains better results than GA-IBX\textsuperscript{MO} for 6 of the 7 instances of SET A and for 11 of 12 instances of SET B. We believe that the poor performance of GA-IBX\textsuperscript{MO} may be explained by the difficulty of these instances which, combined with the time limit, more highlight the lack in terms of intensification of the search process of the crossover operator. However, when we compare the results of GLS with those of GA-NCPX\textsuperscript{MO}, we observe essentially the same results as those obtained in Table 3 for the instances of SET A. Indeed, GA-NCPX\textsuperscript{MO} outperforms GLS for 6 of the 7 instances of SET A. But, for the SET B instances, the results slightly favour GLS. Thus, GA-NCPX\textsuperscript{MO} is better than GLS for 4 instances, is worse for 5 instances while obtaining identical results for the 3 remaining instances.

These results confirm the previous observations made and once again highlight the need to incorporate more explicit intensification mechanisms in our GA. By analyzing the results of adding a local search procedure to GA-NCPX\textsuperscript{MO} (last column of Table 4), we notice a clear
improvement of the performance for all the instances. In fact, the results of GA-NCPX\textsuperscript{MO}+LS are competitive with those of the Challenge Winning Team by obtaining equal or better results for 9 of the 19 instances of the two sets, while obtaining significantly closer results for the remaining instances. GA-NCPX\textsuperscript{MO}+LS always ranks between 1\textsuperscript{st} and 6\textsuperscript{th} except for instance 024_38_5 with HPO_COLOUR_LPO hierarchy where it ranks 12\textsuperscript{th}. Compared to GLS, GA-NCPX\textsuperscript{MO}+LS always obtains better result except for two instances for which the two algorithms obtain identical results.

**Table 4.** Results of the Winning Team, GLS, GA-IBX\textsuperscript{MO}, GA-NCPX\textsuperscript{MO} and GA-NCPX\textsuperscript{MO}+LS for “difficult” high-priority options instances with HPO as the main objective

Table 5 lists the results of the different algorithms for the instances of SET A and SET B with COLOUR-HPO-LPO objective hierarchy. By comparing first GA-IBX\textsuperscript{MO} and GA-NCPX\textsuperscript{MO}, we observe once again that GA-NCPX\textsuperscript{MO} globally outperforms GA-IBX\textsuperscript{MO}. GA-NCPX\textsuperscript{MO} obtains better results for 18 instances out of 19 and identical results for the remaining instance. However, contrary to the previous observation, the gap between the two algorithms is smaller for this group of instances. Except for three instances, the two algorithms give the same value for the main objective. For these instances, the gap between the two algorithms is observed for the second and third objective. However, we notice again that the results of the two algorithms are not competitive with those of the Challenge.
Winning Team, except for instance 35_ch2_S22_J4 with COLOUR_HPO_LPO objective hierarchy for which all algorithms obtain the same result. GA-IBXMO ranks between 12th and 20th while GA-NCPXMO ranks between 1st and 17th. We also notice that, except for one instance for GA-NCPXMO and three instances for GA-IBXMO, the two algorithms obtain the same value for the main objective as the Challenge Winning Team did. We can make this conclusion considering that the weight of the main objective is set at 1000000 and that the gap between the algorithms is less than this value.

|                   | Winning Team | GLS (rank) | AG-IBXMO (rank) | AG-NCPXMO (rank) | AG-NCPXMO+LS (rank) |
|-------------------|--------------|------------|-----------------|------------------|---------------------|
| **SET A**         |              |            |                 |                  |                     |
| COLOUR_HPO_LPO    |              |            |                 |                  |                     |
| 022_3.4           | 11039001 (1) | 11041001 (15) | 11039131 (12) | 11039098 (11) | 11039001 (1)       |
| 039_38.4_ch1      | 68161000 (3) | 68265000 (15) | 68265000 (15) | 68249000 (12) | 68155000 (1)       |
| 064_38.2_ch1      | 63423782 (1) | 63435799 (15) | 63443831 (17) | 63423782 (1) | 63423782 (1)       |
| 064_38.2_ch2      | 27367052 (1) | 27367076 (15) | 27367052 (1) | 27367052 (1) | 27367052 (1)       |
|                   |              |            |                 |                  |                     |
| **SET B**         |              |            |                 |                  |                     |
| COLOUR_HPO_LPO    |              |            |                 |                  |                     |
| 022_S22_J1        | 13022148 (1) | 13022154 (19) | 13022189 (19) | 13022178 (17) | 13022148 (1)       |
| 023_S23_J3        | 51327031 (1) | 5439063 (21) | 51735264 (20) | 51393130 (17) | 51343070 (9)       |
| 024_V2_S22_J1     | 134023158 (1) | 135226676 (20) | 134230457 (14) | 134057341 (4) | 134023158 (1)     |
| 025_S22_J3        | 126127589 (1) | 126300350 (18) | 126136839 (12) | 126127589 (1) | 126127589 (1)     |
| 028_ch1_S22_J2    | 38098210 (1) | 38098251 (9) | 38099330 (16) | 38098334 (12) | 38098188 (1)       |
| 028_ch2_S23_J3    | 4000071 (1) | 4000071 (1) | 5000078 (18) | 4000071 (1) | 4000071 (1)       |
| 029_S21_J6        | 52711171 (1) | 52905570 (20) | 52763341 (15) | 52717428 (8) | 52711171 (1)       |
| 035_ch1_S22_J3    | 6156090 (1) | 6156092 (10) | 615609 (18) | 6156092 (10) | 6156090 (1)       |
| 035_ch2_S22_J3    | 7651671 (1) | 7651671 (1) | 7651671 (1) | 7651671 (1) | 7651671 (1)       |
| 039_ch1_S22_J4    | 55045096 (1) | 55045235 (9) | 55046737 (18) | 55045235 (9) | 55045096 (1)       |
| 039_ch3_S22_J4    | 59214671 (1) | 59214698 (12) | 59214783 (15) | 59214681 (9) | 59214671 (1)       |
| 048_ch1_S22_J3    | 64115670 (1) | 64135847 (14) | 64153806 (15) | 64124687 (12) | 64115670 (1)       |
| 048_ch2_S22_J3    | 58283180 (1) | 58288194 (12) | 58312194 (19) | 58290183 (13) | 58283180 (1)       |
| 064_ch1_S22_J3    | 62095288 (1) | 62108458 (10) | 63116379 (15) | 62113381 (12) | 62097307 (3)       |
| 064_ch2_S22_J4    | 31052178 (1) | 31052184 (9) | 32052158 (16) | 31053188 (13) | 31052178 (1)       |

Table 5. Results of the Winning Team, GLS, GA-IBXMO, GA-NCPXMO and GA-NCPXMO+LS for instances with COLOUR as the main objective

By comparing the results of our algorithms with those of GLS, we again notice that GLS outperforms GA-IBXMO for 2 of 4 instances of SET A, is worse for only one instance while obtaining an identical result for the remaining instance. However, for the SET B instances, GLS clearly outperforms GA-IBXMO by obtaining better results for 11 instances, worse results for 3 instances and identical results for the remaining instance. By comparing the results of GA-NCPXMO with those of GLS, one notes that GA-NCPXMO obtains better results for all instances of SET A except one where the two algorithms achieve identical results. For the SET B instances, GA-NCPXMO obtains better results than GLS for 5 instances, is worse for 6 instances while obtaining identical results for the 4 remaining instances. Again, we observe very close performance between the two algorithms.

By now comparing the results of the two GAs to those of the Challenge Winning Team, we notice on one hand that GA-NCPXMO always reaches the same value for the main objective.
On the other hand, GLS doesn’t always reach these values. GLS even obtains the worst solution for instances 023_S23_J3 and 025_S22_J3 with COLOUR_HPO_LPO objective hierarchy.

By analysing the results of GA-NCPX\textsuperscript{MO}+LS, we observe a clear performance improvement for all the instances. Thus, for SET A instances, GA-NCPX\textsuperscript{MO}+LS always obtains identical or better results than those of the Challenge Winning Team. For SET B instances, GA-NCPX\textsuperscript{MO}+LS obtains identical or better results than those of the Challenge Winning Team for 11 of the 15 instances. GA-NCPX\textsuperscript{MO}+LS always ranks between 1\textsuperscript{st} and 4\textsuperscript{th} except for instances 023_S23_J3 and 029_S21_J6 with COLOUR_HPO_LPO objective hierarchy, where it ranks 9\textsuperscript{th} and 8\textsuperscript{th} respectively. Compared to GLS, GA-NCPX\textsuperscript{MO}+LS gets better results for 16 of the 19 instances while obtaining identical results for the 3 remaining ones.

Finally, Table 6 gives the results of the different algorithms for the 19 instances of SET X that was used in the 2005 ROADEF Challenge to determine the final ranking. Here, instead of executing the algorithms once as we did in the previous results, we executed the algorithms 5 times as was done for the qualified teams in this phase of the Challenge. The values reported in this table are thus the average results of 5 runs.

| SET X                  | Winning Team | GLS (rank) | AG-IBX\textsuperscript{MO} (rank) | AG-NCPX\textsuperscript{MO} (rank) | AG-NCPX\textsuperscript{MO}+LS (rank) |
|------------------------|--------------|------------|-----------------------------------|------------------------------------|--------------------------------------|
| HPO_COLOUR_LPO         |              |            |                                   |                                    |                                      |
| 023_S49_J2             | 192466 (1)   | 246268.20 (17) | 246268.40 (18) | 211879 (12) | 193077 (3) |
| 024_S49_J2             | 337006 (1)   | 421425 (8) | 270462.20 (18) | 506015 (11) | 346202.20 (2) |
| 029_S49_J5             | 110442.60 (2) | 120855 (11) | 150969.20 (17) | 123029.20 (12) | 111093.20 (3) |
| 034_VU_S51_J1_J2_J3    | 56386.80 (1) | 76217.60 (17) | 74354.20 (15) | 66750 (12) | 57577.40 (5) |
| 034_VU_S51_J1_J2_J3    | 8087037 (4) | 809145.20 (10) | 8112049 (16) | 8103064 (15) | 8087035.80 (1) |
| 039_CH1_S49_J1         | 69239 (1)    | 69455.60 (6) | 69705 (9) | 69479.60 (7) | 69355.20 (2) |
| 039_CH3_S49_J1         | 231030.20 (2) | 239593.20 (16) | 250670 (17) | 235475.40 (13) | 231030.40 (3) |
| 048_CH1_S50_J4         | 197044.80 (3) | 206509.60 (16) | 207634 (17) | 204182 (14) | 197045.40 (4) |
| 048_CH2_S49_J5         | 31077916.20 (1) | 31104598.80 (12) | 31128931 (18) | 31106266.2 (13) | 31078317.20 (2) |
| 064_CH1_S49_J1         | 61187229.80 (1) | 61229518.80 (12) | 61309246 (17) | 61223429 (10) | 61190429 (2) |
| 064_CH2_S49_J4         | 37000 (1)    | 40400 (14) | 42000 (15) | 39000 (12) | 37000 (1) |
| 655_CH1_S51_J2_J3_J4   | 30000 (1)    | 30000 (1) | 30000 (1) | 30000 (1) | 30000 (1) |
| 655_CH2_S52_J1_J2_S01_J1 | 153034000 (1) | 153035200 (8) | 153047000 (12) | 153041000 (11) | 153034000 (1) |
| LPO_COLOUR_HPO         |              |            |                                   |                                    |                                      |
| 022_S49_J2             | 12002003 (1) | 12002003 (1) | 12002008 (16) | 12002003 (1) | 12002003 (1) |
| 035_CH1_S50_J4         | 5010000 (1)  | -          | 5010000 (1) | 5010000 (1) | 5010000 (1) |
| 035_CH2_S50_J4         | 6056000 (1)  | 6056000 (1) | 6056000 (1) | 6056000 (1) | 6056000 (1) |
| LPO_COLOUR_HPO         |              |            |                                   |                                    |                                      |
| 025_S49_J1             | 160407.60 (2) | 189390.20 (15) | 188118.20 (13) | 176454.60 (10) | 160407.20 (1) |
| 028_CH1_S50_J4         | 36370094.20 (4) | 36377907.20 (5) | 49863125.80 (20) | 39634315.20 (12) | 36360092.40 (2) |
| 028_CH2_S51_J1         | 3 (1)        | 3 (1)      | 3 (1) | 3 (1) | 3 (1) |

Table 6. Results of the Winning Team, GLS, GA-IBX\textsuperscript{MO}, GA-NCPX\textsuperscript{MO} and GA-NCPX\textsuperscript{MO}+LS for SET X instances

When we compare the average results of GA-IBX\textsuperscript{MO} and GA-NCPX\textsuperscript{MO}, we again notice for this set that GA-NCPX\textsuperscript{MO} clearly outperforms GA-IBX\textsuperscript{MO} by obtaining better results except for 4 instances for which the two algorithms obtain the same average results. We also notice for these 4 instances that the two algorithms always find the same solution for each run. Moreover, the results obtained by the two GAs are the same as those of the Winning Team.
By looking more closely at the characteristics of these 4 instances, we notice that they are small instances where the number of cars to schedule is between 65 and 376. These small sizes probably explain why the two algorithms solve these 4 instances trivially. As shown in the previous results, the gap between the two algorithms seems to be related to the size of the instances. Indeed, GA-IBX\textsuperscript{MO} seems to have more difficulty to converge towards a good solution for large instances. This situation is again confirmed using instance 024_S49_J2 with HPO\_COLOUR\_LPO objective hierarchy and 1319 cars to schedule. For this instance, the gap between the average results of the two algorithms for the main objective is over 26 conflicts. Except for the 4 small size instances solved trivially, GA-IBX\textsuperscript{MO} ranks between 9\textsuperscript{th} and 20\textsuperscript{th} while GA-NCPX\textsuperscript{MO} ranks between 7\textsuperscript{th} and 15\textsuperscript{th}.

If we now compare the results of our two algorithms to those of GLS, we observe similar results to those obtained for SET A et SET B. GA-IBX\textsuperscript{MO} is worse than GLS for 13 instances, better for 3 instances while identical for the 3 other instances. We notice that among the 3 instances for which GA-IBX\textsuperscript{MO} achieves better average results than GLS, there is one instance (035_CH1_S50_J4 with COLOUR\_HPO\_LPO hierarchy) for which GLS did not provide a feasible solution during this phase of the Challenge. When we now compare GLS to GA-NCPX\textsuperscript{MO}, we notice that GA-NCPX\textsuperscript{MO} outperforms GLS for 8 instances, is worse for 7 instances while identical for the 4 remaining instances.

We also notice that the results of GA-IBX\textsuperscript{MO} and GA-NCPX\textsuperscript{MO} are not competitive with the average results of the Winning Team. However, by adding a local search procedure to GA-NCPX\textsuperscript{MO}, we considerably improve the performance of the algorithm by obtaining the best average results for 10 instances while obtaining very close average results for the other instances. GA-NCPX\textsuperscript{MO}+LS ranks between 1\textsuperscript{st} and 5\textsuperscript{th} for all the instances of SET X.

Now, to compare the performance of the proposed approaches with the results of the teams that qualified for the Challenge, we used the ranking procedure described in the Challenge description, that consists in calculating a mark for each instance of SET X according to Equation 5. The mark of each algorithm is calculated according to the best and the worst solution found by the 18 teams that qualified for the Challenge and the 3 proposed algorithms. The score is a normalized measure of solution quality that necessarily lies between 0 and 1.

$$ mark(Alg) = \frac{result_{Alg} - Best\_result}{Best\_result - worst\_result} $$  

In Equation 5, Best\_result and Worst\_result indicate respectively the best and the worst average result found for an instance while result\_Alg indicates the average result found by the algorithm for which we compute the mark for the same instance. Then, each row of Table 7 lists the mark of the Winning Team, the GA-IBX\textsuperscript{MO}, the GA-NCPX\textsuperscript{MO} and GA-NCPX\textsuperscript{MO}+LS for each instance of SET X. The last row of this table lists the total mark of each algorithm for the whole set. On analysing the results of Table 7, we notice that they confirm the results of Tables 3 to 6, namely that GA-NCPX\textsuperscript{MO} is a better performer than GA-IBX\textsuperscript{MO} and that GA-NCPX\textsuperscript{MO}+LS is the best performer compared to the two other algorithms. It is important to mention that, according to the final rank of the Challenge that is published by the organizers and that is available online from the Challenge website, GLS ranks 13\textsuperscript{th} with a mark of 16.8937 while the Winning Team has a mark of 18.9935. Based on these results, we may conclude that the difference between the results of our best genetic approach and those of the Winning Team is rather small (0.0345). We also notice that both GA-NCPX\textsuperscript{MO} obtain a
better mark than GLS, with and without local search procedure. We may then conclude that the methods proposed in this chapter achieve competitive results for the multi-objective ICSP. Thus, we demonstrate that GAs are well suited to address this category of problem if they incorporate specific knowledge of the problem to design dedicated genetic operators.

| SET X | Winning Team | AG-IBX^{MO} | AG-NCX^{MO} | AG-NCX^{MO}+LS |
|-------|--------------|-------------|-------------|----------------|
| HPO_COLOUR_LPO | 023_S49_J2 | 1 | 0.5575 | 0.8403 | 0.9950 |
| | 024_S49_J2 | 1 | 0.4605 | 0.9966 | 0.9998 |
| | 029_S49_J5 | 0.9980 | 0.4249 | 0.8200 | 0.9888 |
| | 034_VP_S51_J1_J2_J3 | 0.9996 | 0.7949 | 0.8799 | 0.9823 |
| | 034_VU_S51_J1_J2_J3 | 1 | 0.9998 | 0.9999 | 1 |
| | 039_CH1_S49_J1 | 1 | 0.9755 | 0.9873 | 0.9939 |
| | 039_CH3_S49_J1 | 0.9999 | 0.6368 | 0.9178 | 1 |
| | 048_CH1_S50_J4 | 0.9999 | 0.9952 | 0.9968 | 1 |
| | 048_CH2_S49_J5 | 1 | 0.9868 | 0.9927 | 0.9999 |
| | 064_CH1_S49_J1 | 1 | 0.9799 | 0.9940 | 0.9995 |
| | 064_CH2_S49_J4 | 1 | 0.8588 | 0.9435 | 1 |
| | 655_CH1_S51_J2_J3_J4 | 1 | 1 | 1 | 1 |
| | 655_CH2_S52_J1_J2_S01_J1 | 1 | 0.9999 | 0.9999 | 1 |
| COLOUR_HPO_LPO | 022_S49_J2 | 1 | 0.9999 | 1 | 1 |
| | 035_CH1_S50_J4 | 1 | 1 | 1 | 1 |
| | 035_CH2_S50_J4 | 1 | 1 | 1 | 1 |
| HPO_LPO_COLOUR | 025_S49_J1 | 1 | 0.9983 | 0.9990 | 1 |
| | 028_CH1_S50_J4 | 0.9999 | 0.9553 | 0.9891 | 0.9999 |
| | 028_CH2_S51_J1 | 1 | 1 | 1 | 1 |
| **Total** | **18.9935** | **16.6241** | **18.3569** | **18.9590** |

Table 7. Marks of the Winning Team, GA-IBX^{MO}, GA-NCX^{MO} and GA-NCX^{MO}+LS for SET X instances.

6. Conclusion

In this chapter, we have introduced a GA based on two specialized crossover operators dedicated to the multi-objective nature of the ICSP proposed by French automobile manufacturer Renault for the ROADEF 2005 Challenge. If GAs are known to be well suited for multi-objective optimization (Barichard, 2003; Basseur, 2004; Zinflou et al., 2006), few researchers and industrials decided to use this category of algorithms to solve the ICSP. Among the 18 teams that qualified for the second phase of the Challenge, only one proposed a genetic algorithm based approach. This situation may be explained by the difficulty in defining specific and efficient genetic operators that take into account the specificities of the problem. The approach proposed in this chapter is essentially based on adapting highly specialized genetic crossover operators to the specificities of the industrial version of the single objective car sequencing problem, for which we have three conflicting objectives to optimize. The numerical experiments allowed us to demonstrate the efficiency of the
proposed approach for this industrial problem. A natural conclusion of these experimental results is that GAs may be robust and efficient alternative to solve the multi-objective ICSP. These results also again highlight the importance of incorporating specific problem knowledge into genetic operators, even if classical genetic operators could be used. We are also aware of the fact that having known the solutions found by the algorithms of the different qualified teams has facilitated improving and tuning our algorithms. However, the main purpose of this study was to demonstrate that GAs can be an efficient alternative to solve this kind of industrial problem.

The lexicographical treatment of the objectives proposed by Renault is such that it can eliminate several “interesting” solutions for the manufacturer. Indeed, the relaxation of the importance granted to the main objective can highlight other attractive solutions for the company. For example, if an additional violation on the HPO objective allows to avoid 5 colour changes, the production scheduler could then be interested to a such solution to make his final schedule. We therefore believe that the industrial problem introduced by Renault would benefit to be treated to obtain so-called “compromise solutions”. In this context, the GAs proposed in this chapter represent very interesting alternatives to find these compromise solutions. In fact, GAs are well suited for multi-objective optimization in the Pareto sense and these approaches have proven their ability to generate compromise solutions in a single optimization step. Since the mid-nineties, an increasing number of approaches exploit the principle of dominance (Zitzler and Thiele, 1998; Deb, 2000; Knowles and Corne, 2000a; Knowles and Corne, 2000b; Coello Coello and Pulido, 2001) in the Pareto sense as defined by Goldberg (1989). These evolutionary multi-objective algorithms use the concepts of dominance, niches and elitism (Deb, 2000; Knowles and Corne, 2000b; Deb and Goel, 2001; Zitzler et al., 2001). The NSGAII algorithm (Deb, 2000), the SPEA2 algorithm (Zitzler et al., 2001) and the PMS\textsuperscript{MO} algorithm (Zinflou et al., 2007) are recognized as amongst the best performing of the elitist multi-objective evolutionary algorithms. These algorithms are said to be elitist because they include one or several mechanisms allowing the memorization of the best solutions found during the execution of the GA.

For future work, we will use this type of approaches to consider the objectives simultaneously, without assigning priority or weight. A set of compromise solutions may then be found for comparison to the solution by considering the objectives in lexicographical order. It will thus be possible to highlight different solutions that are much more financially interesting for a manufacturer and that are better suited to industrial reality.

7. References

Barichard, V. (2003). Approches hybrides pour les problèmes multiobjectifs, Ph.D. Thesis, Université d’Angers, France.

Basseur, M. (2004). Conception d’algorithmes coopératifs pour l’optimisation multi-objectifs : Application aux problèmes d’ordonnancement de type flow-shop, Ph.D. Thesis, Université des Sciences et Technologies de Lille, France.

Benoit, T. (2007). Soft car sequencing with colors: Lower bounds and optimality proofs, European Journal of Operational Research: doi:10.1016/j.ejor.2007.04.035.

Briant, O.; Naddef, D. & Mounié, G. (2007). Greedy approach and multi-criteria simulated annealing for the car sequencing problem, European Journal of Operational Research: doi:10.1016/j.ejor.2007.04.052.
Coello Coello, A. C. & Pulido, G. T. (2001). Multiobjective optimization using a micro-genetic Algorithm, Proceedings of the Genetic and Evolutionary Computation Conference (GECCO’2001), 274-282, San Francisco, California.

Cordeau, J.-F.; Laporte, G. & Pasin, F. (2007). An iterated local search heuristic for the car sequencing problem, European Journal of Operational Research: doi:10.1016/j.ejor.2007.04.048.

Deb, K. (2000). A fast elitist non-dominated sorting genetic algorithm for multiobjective optimization : NSGA II, Proceedings of Parallel problem Solving from Nature – PPSN VI, Lecture Notes in Computer Science, M. Schoenauer et al. (Eds), Springer, 849-858.

Deb, K. & Goel, T. (2001). Controlled elitist non-dominated sorting genetic algorithms for better convergence, Proceedings of Evolutionary Multi-Criterion Optimization, Lecture Notes in Computer Science 1993, E. Zitzler et al. (Eds), Springer-Verlag.

Dincbas, M.; Simonis, H. & van Hentenryck, P. (1988). Solving the car sequencing problem in constraint logic programming, Proceedings of the European Conference on Artificial Intelligence (ECAI-88), Munich, Germany, Pitmann Publishing, London, 290-295.

Estellon, B.; Gardi, F. & Nouioua, K. (2005). Ordonnancement de véhicules: une approche par recherche locale à grand voisinage, Proceedings of Journées Francophones de Programmation par Contraintes, 21-28, Lens, France.

Estellon, B.; Gardi, F. & Nouioua, K. (2007). Two local search approaches for solving real-life car sequencing problem, European Journal of Operational Research, doi:10.1016/j.ejor.2007.04.043.

Gagné, C.; Gravel, M. & Price, W. L. (2006). Solving real car sequencing problems with ant colony optimization, European Journal of Operational Research, 174(3), 1427-1448.

Gavranović, H. (2007). Local search and suffix tree for car-sequencing problem with colors, European Journal of Operational Research, doi:10.1016/j.ejor.2007.04.051.

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

Gottlieb, J.; Puchta, M. & Solnon, C. (2003). A study of greedy, local search and ant colony optimization approaches for car sequencing problems, Computers Science, 246-257.

Grefenstette, J. J. (1986). Optimization of Control parameters for genetic algorithms. IEEE Transactions on Systems, Man, and Cybernetics, 16(1), 122-128.

Jaszkiewicz, A.; Kubiak, M. & Kominek, P. (2004). The application of the genetic local search algorithm to Car Sequencing Problem, Proceedings of the 7th National Conference on Evolutionary Algorithms and Global Optimization, Kazimierz Dolny, Poland.

Knowles, J. D. & Corne, D. W. (2000a). M-PAES : A Memetic Algorithm for Multiobjective Optimization, Proceedings of the 2000 Congress on Evolutionary Computation, 325-332.

Knowles, J. D. & Corne, D. W. (2000b). The pareto-envelope based selection algorithm for multiobjective optimization, Proceedings of the Sixth International Conference on Parallel Problem Solving from Nature (PPSN VI), 839-848, Berlin.

Nguyen, A. & Cung, V.-D. (2005). Le problème du Car Sequencing Renault et le challenge ROADEF 2005, Proceedings of Journées Francophones de Programmation par Contraintes, 3-10, 2005.

Prandtstetter, M. & Raidl, G. R. (2007). An integer linear programming approach and a hybrid variable neighborhood search for the car sequencing problem, European Journal of Operational Research, doi:10.1016/j.ejor.2007.04.044.
Ribeiro, C. C.; Aloise, D. Noronha, T. F. Rocha, C. & Urrutia, S. (2007a). An efficient implementation of a VNS/ILS heuristic for a real-life car sequencing problem, *European Journal of Operational Research*, doi:10.1016/j.ejor.2007.02.003.

Ribiero, C. C.; Aloise, D. Noronha, T. F. Rocha, C. & Urrutia, S. (2007b). A hybrid heuristic for a multi-objective real-life car sequencing, *European Journal of Operational Research*, doi:10.1016/j.ejor.2007.04.034.

Solnon, C.; Cung, V.-D. & Artigues, C. (2007). The car sequencing problem: overview of state-of-the-art methods and industrial case-study of the ROADEF’2005 challenge problem, *European Journal of Operational Research*, doi:10.1016/j.ejor.2007.04.033.

Terada, J.; Vo, H. & Joslin, D. (2006). Combining genetic algorithms with squeaky-wheel optimization, *Proceedings of Genetic and Evolutionary Computation Conference (GECCO) 2006*, Seattle.

Warwick, T. & Tsang, E. (1995). Tackling car sequencing problem using a generic genetic algorithm, *Evolutionary Computation*, 3(3), 267-298.

Zinflou, A., Gagné, C. & Gravel, M. (2007). Crossover operators for the car-sequencing problem, *Proceedings of the Seventh European Conference on Evolutionary Computation in Combinatorial Optimisation (EvoCOP 2007)*, LNCS 4446, C. Cotta and J. van Hemert (Eds.), Springer-Verlag Berlin Heidelberg, 229-239.

Zinflou, A.; Gagné, C. Gravel, M. & Price, W. L. (2006). Pareto memetic algorithm for multiple objectives optimization with an industrial application, *Journal of Heuristics*, doi: 10.1007/s10732-007-9042-2.

Zitzler, E.; Laumanns, M. & Thiele, L. (2001). SPEA2: Improving the strength pareto evolutionary algorithm, *Technical Report 103*, Computer Engineering and Communication Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich, Switzerland.

Zitzler, E. & Thiele, L. (1998). An evolutionary algorithm for multiobjective optimization: the strength pareto approach, *Technical Report 43*, Computer Engineering and Communication Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich, Switzerland.
With the recent trends towards massive data sets and significant computational power, combined with evolutionary algorithmic advances evolutionary computation is becoming much more relevant to practice. Aim of the book is to present recent improvements, innovative ideas and concepts in a part of a huge EA field.

How to reference
In order to correctly reference this scholarly work, feel free to copy and paste the following:

A. Zinflou, C. Gagné and M. Gravel (2008). Design of an Efficient Genetic Algorithm to Solve the Industrial Car Sequencing Problem, Advances in Evolutionary Algorithms, Xiong Zhihui (Ed.), ISBN: 978-953-7619-11-4, InTech, Available from: http://www.intechopen.com/books/advances_in_evolutionary_algorithms/design_of_an_efficient_genetic_algorithm_to_solve_the_industrial_car_sequencing_problem