Adaptation of the Genetic Local Search Algorithm to a Car Sequencing Problem

Andrzej Jaszkiewicz, Paweł Kominek, Marek Kubiak
Institute of Computing Science, Poznan University of Technology, Piotrowo 3a, 60-965 Poznan
e-mail: {Andrzej.Jaszkiewicz; Pawel.Kominek; Marek.Kubiak}@cs.put.poznan.pl
www-idss.cs.put.poznan.pl

1 Introduction

The paper describes an adaptation of the genetic local search algorithm (GLS) to a car sequencing (CS) problem. The problem has been formulated by a French car manufacturer for the ROADEF Challenge 2005 [2]. It is based on a real problem coming from production planning. Its goal is to schedule the order of cars to be put on a production line for each production day, while satisfying as many factory requirements as possible. The main element of this adaptation is a systematic approach to construction of recombination operators for the given optimization problem. The use of this approach allowed designing a well-performing algorithm avoiding intensive tests in the trials and error method. Another improving parts of this adaptation are efficient initial solutions generation heuristics and a fast local search procedure.

2 Description of the car sequencing problem

In the CS problem car factories have to schedule the order of cars to be put on the line for a production day, while satisfying as many requirements of a paint shop and an assembly line as possible. The paint shop requirement is to minimize the cost of painting the vehicles. The assembly line requirements are more complicated: vehicles which require special assembly operations (‘hard to assemble’) have to be evenly distributed throughout the all sequence of vehicles, in order to smooth the workload on the line. Therefore, the number of such vehicles may not exceed a given quota in any (continuous) subsequence of vehicles. Each type of this ‘hardness’ is defined as one ratio constraint (RC). All the assembly line requirements are expressed in a set of ratio constraints.

More specifically, each vehicle in the given set is described by the following parameters:

- a unique identifier,
- a paint color code,
- a ratio constraints activity vector.

The constraints activity vector shows whether the vehicle is considered ‘hard to assemble’ with respect to all ratio constraints. A 0 in the vector means the specific constraint is not active for the vehicle (‘easy to assemble’); a 1 means the opposite.

A solution of the CS problem is a sequence of vehicles (a permutation of their identifiers).
There is one hard constraint in the problem, namely the upper paint batch limit \( (PBL) \): if in a solution there is a subsequence of vehicles with the same paint color which is longer than \( PBL \), then the solution is regarded as infeasible.

The color is also used as a subcriterion of the overall objective function. Actually, the number of paint color changes \( (PCC(s)) \) between consecutive vehicles in solution \( s \) is counted, as it reflects the cost of spray guns washing each time colors of vehicles change (the painting cost).

The mentioned RCs are soft constraints and their violations are also reflected in the value of the objective function. Each RC is defined by a ratio \( N/P \). The denominator \( (P) \) specifies the length of a subsequence of vehicles, while the numerator \( (N) \) is the upper limit on the number of vehicles in this subsequence which may have the ratio constraint active.

More formally, let \( RC_i \) be the \( i \)th ratio constraint in the problem, defined by \( N_i/P_i \). Let \( sub_j \) be a continuous subsequence of vehicles of length \( P_i \) in a solution, starting at position \( j \), and \( V_i(sub_j) \) be the number of vehicles in \( sub_j \) which have \( RC_i \) active. Then the number of violations of this ratio constraint for subsequence \( sub_j \), \( VN_i(sub_j) \), is defined as:

\[
VN_i(sub_j) = \begin{cases} 
V_i(sub_j) - N_i & \text{if } V_i(sub_j) > N_i \\
0 & \text{otherwise}
\end{cases}
\]

The value of \( VN_i(s) \), the total number of violations of \( RC_i \) in solution \( s \), is the sum of \( VN_i(sub_j) \) for all possible subsequences \( sub_j \) of solution \( s \).

There is some additional \( VN_i(s) \) computation at the beginning and at the end of the sequence of vehicles, which is taken into consideration in the algorithm outlined later. However, since this issue is very specific, and not really different from the idea of ratio constraints, it is not presented here; for the detailed description of this computation refer to [2].

The ratio constraints form two disjoint sets which differ by a priority level. High priority ratio constraints (HPRCs) are those considered more important in the production process than others, with low priority (LPRCs). The numbers of violations \( VN_i(s) \) are summed up in these sets and constitute two subcriteria of the overall objective function: \( VN_{HPRC}(s), VN_{LPRC}(s) \).

The objective function \( f \) is defined as the weighted sum of the three mentioned subcriteria:

\[
f(s) = W_{PCC} \cdot PCC(s) + W_{HPRCs} \cdot VN_{HPRCs}(s) + W_{LPRCs} \cdot VN_{LPRCs}(s)
\]

The vector of weights \( W=(W_{PCC}, W_{HPRCs}, W_{LPRCs}) \) is constant for each instance and taken from the set \{ \( (100, 10000, 1), (1, 10000, 100), (10000, 100, 1) \) \}. The objective function is to be minimized in order to obtain the lowest production cost.

In instances of the problem there is also one additional piece of information available, about the hardness of HPRCs. If such ratio constraints are not very tight, they are considered easy to satisfy. Otherwise, if numbers of constrained vehicles are higher than possible to schedule without any violations, HPRCs are considered difficult to satisfy.

3 Initial reduction of the search space size

At the first approach to the problem an important observation was made: in an instance there may exist vehicles which have exactly the same paint color code and constraints activity vector, and are, therefore, indiscernible from the point of view of the objective function. In fact, such subsets of vehicles, called groups later in the paper, were found in all instances. This discovery, which is independent on the type of algorithm used to solve the problem, results in a considerable reduction of the search space size, since a solution is no longer perceived as a permutation (a sequence of identifiers), but as a permutation with repetitions (a sequence of groups indices).
4 Implementation of the genetic local search algorithm

4.1 Initial solutions

Initial solutions are generated by two heuristic algorithms. The first one, a profit-driven heuristic, iteratively adds one of available group indices at the current position, starting from the beginning of an empty sequence. The group index to be added in each iteration is chosen on the basis of the heuristic evaluation of the group:

\[
EG = \text{Profit} - \text{Cost}
\]

If addition of a group index at the current position does not increase the number of violations of some RC, then this fact is counted as a unit of profit (1), and also multiplied by the weight of this RC and a quotient \(k_i\). On the other hand, if this addition increases the number of violations \(VN_i\), the weighted change in \(VN_i\) is counted as cost. The quotient \(k_i\) is computed as:

\[
k_i = \frac{\text{number of not assigned vehicles with RC}_i \text{ active}}{\text{maximum number of not assigned vehicles which may have RC}_i \text{ active and cause no violation}}
\]

The idea behind this quotient is to measure which RCs are more difficult to satisfy than others, and to emphasize the importance of difficult constraints in the heuristic evaluation \(EG\). Basically the same idea of profit and cost applies to the color of a group. If addition of a group index does not increase the number of paint color changes (PCC), then this fact, multiplied by \(W_{PCC}\), is counted in \(EG\) as profit. Otherwise it is counted as cost.

The group chosen to be added at an iteration is the one with the highest \(EG\) (steepest approach). The obtained heuristic algorithm is very fast and allows for construction of multiple initial solutions.

Second heuristic algorithm inserts a randomly chosen group index into the created solution at one chosen position in the current sequence. The position of this insertion is determined on the basis of the objective value for the obtained solution (the solution may be not full, but the objective value of such a solution is computed simply in the same manner as for an ordinary, feasible one). The chosen insertion position is the one with the lowest value of \(f\) after insertion (steepest approach). The obtained algorithm is rather a time-consuming one, so it creates only one initial solution. However, this solution is sometimes the best in the initial population.

4.2 Local search procedure

Local search uses a shift neighborhood operator. The operator shifts a group index at a position in the sequence to another position in this sequence. In other words, it removes a group index form the sequence and inserts it at another position. When an index is removed, all successive indices are moved by one position back. When an index is inserted at position \(l\) the index at position \(l\) and all successive indices are moved by one position forward.

The local search procedure is a mixture of greedy and steepest approaches. Firstly, the procedure chooses (in a random manner; the greedy approach) one removal position. The index at this position is then removed from it. Secondly, the local search chooses an insertion position for the removed index. The best insertion position is chosen among all possible ones (the steepest approach). If the insertion at the best position results in an overall improvement of the objective function, taking into account both changes due to the removal and the insertion, the shift move is performed.
Since the size of the neighborhood defined by the shift operator is relatively large (~1 000 000 in our case), a significant effort has been made to improve the efficiency of the algorithm. First of all, the neighbor solutions are not constructed explicitly, but only modifications of the objective function caused by particular moves are evaluated. In our case, since each group index may be inserted at about 1000 positions, most of the CPU time is consumed by the search for the best insertion position, while the time of evaluation of removals is practically meaningless. In addition, the CPU time is consumed mainly by evaluation of changes to ratio constraints violations. Thus, the effort has been focused on efficient evaluation of a group index insertions from the point of view of ratio constraints violations.

Assume that the insertion of an index at position \( l \) is evaluated with respect to ratio constraint \( RC_i \), defined by ratio \( N_i/P_i \). It may modify the number of vehicles associated with this ratio constraint in subsequences of indices starting from positions \([l-N_i+1, \ldots, l]\). Therefore, only the modified subsequences have to be taken into account.

Furthermore, evaluation of insertion at position \( l+1 \) may be further improved by taking into account results calculated for position \( l \). Insertion at position \( l \) modifies subsequences of indices starting from positions \([l-N_i+1, \ldots, l]\); insertion at position \( l+1 \) modifies subsequences of indices starting from positions \([l-N_i+2, \ldots, l+1]\). Thus, it is enough to exclude from the calculations the change due to subsequence starting from position \( l-N_i+1 \) and add the change due to subsequence starting from position \( l+1 \).

The efficiency of local search is further improved by storing information about modifications of ratio constraints violations at particular positions. For each insertion position \( l \) and ratio constraint \( RC_i \), the evaluation of the insertion of both an index of a group active and not active on this constraint is stored when calculated for the first time. When insertion of next indices is considered these stored values are used. Of course, the stored values are cleared in a range of positions defined by \( P_i \) when a move is performed.

Finally, the efficiency of local search is improved by the concept of candidate moves. The idea is to avoid evaluations of not promising moves. In our case, the definition of candidate moves is strictly related to crossover and mutation operators (see sections 4.3 and 4.4). When local search is applied to a solution constructed by one of the operators, only a subset of removal positions is considered by local search.

### 4.3 Design of a crossover operator on the basis of global convexity tests

#### 4.3.1 Rationale for testing global convexity

Global convexity is a tendency in the solution space that solutions become more similar to each other (with respect to some similarity/distance function) as they are getting better in quality.

Such tendency was found by Boese et al. [1] in some instances of TSP and the graph bisection problem, and also confirmed to exist in instances of some test functions defined for binary strings [4]. In the research mentioned the tendency was measured (tested) mainly by a value of the correlation coefficient between solution quality and similarity to a global optimum (known in these cases). Generally speaking, the conclusion was that the higher the correlation, the more globally convex the solution space is. If the correlation did not reveal the existence of a globally convex landscape (i.e. the shape of a set of points was not regular), the tendency was also observed on scatter plots of quality and similarity [1], [4].

However, global convexity is not only a theoretical feature of the solution space. As it was shown by Merz and Freisleben [6], [7], Jaszkiewicz and Kowalek [3], and Kubiak [5] it may be efficiently used in an evolutionary algorithm by so-called distance-preserving crossover operators (DPX). Such an operator creates an offspring which preserves important features of
parents (identified in global convexity tests) in such a way, that it is at least as similar to the parents as they are to each other. This method enables an evolutionary algorithm to efficiently exploit the 'path to the goal' [4] (the tendency of increasing similarity of better solutions) found by global convexity tests. It also allows a designer to avoid not promising paths of development of an algorithm (e.g. implementation of badly performing crossover operators) and testing them in the trials and errors method.

4.3.2 Global convexity tests for the car sequencing problem

Based on the analysis of the problem two potentially important features of solutions were identified:

- assignment of a group index to a certain position; this feature reflects the intuition that good solutions should tend to have the same group indices at the same positions;
- existence of groups indices in a certain subsequence; this feature reflects the intuition that good solutions should have groups indices put in very similar subsequences, not necessarily at the same positions; it is also presumed that longer common subsequences are more important that short ones.

For each of the proposed features a similarity function was defined:

- \( sim_{CP}(s_1, s_2) \): counts the number of groups indices assigned to the same positions in solutions \( s_1 \) and \( s_2 \),
- \( sim_{CS}(s_1, s_2) \): firstly, it counts the sum of square lengths of all maximum subsequences of groups indices common to solutions \( s_1 \) and \( s_2 \) (the square of lengths prefers one long common subsequence to several short ones); finally, the square root of the sum is computed.

Global convexity tests were conducted for each problem’s instance [3], [5]:

- 1000 different local optima were generated, starting from random solutions and using the local search procedure with the shift neighborhood operator,
- for each local optimum its average similarity to not worse solutions was computed,
- the value of the correlation coefficient between a similarity function and the objective function was computed,
- scatter plots of the objective and similarity functions were generated, showing the actual tendency in the set of local optima.

Here we present results of the tests just for a part of the 19 instances available at [2]: 6 representative instances, 2 of each of the main types (one large, one small). The computed values of correlation coefficients are shown in Table 1.

<table>
<thead>
<tr>
<th>Instance name</th>
<th>Correlation with ( sim_{CP} )</th>
<th>Correlation with ( sim_{CS} )</th>
<th>Correlation with ( sim_{CSWG} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>024_38_5_EP_ENP_RAF</td>
<td>-0,012</td>
<td>-0,157</td>
<td>-0,920</td>
</tr>
<tr>
<td>025_38_1_EP_ENP_RAF</td>
<td>-0,082</td>
<td>-0,164</td>
<td>-0,123</td>
</tr>
<tr>
<td>039_38_4_RAF_EP_ch1</td>
<td>0,025</td>
<td>-0,893</td>
<td>-</td>
</tr>
<tr>
<td>048_39_1_EP_ENP_RAF</td>
<td>-0,041</td>
<td>-0,145</td>
<td>-0,816</td>
</tr>
<tr>
<td>064_38_2_EP_RAF_ENP_ch2</td>
<td>-0,077</td>
<td>-0,652</td>
<td>-</td>
</tr>
<tr>
<td>064_38_2_RAF_EP_ENP_ch2</td>
<td>-0,003</td>
<td>-0,960</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 1. Values of the correlation between the objective function and the similarity functions.
Very low absolute values of the correlation for the $sim_{cp}$ function indicate that there is no global convexity while taking common positions of groups indices into account.

In case of 3 instances the correlation for the $sim_{cs}$ function has high absolute value. These values are negative, so they reflect a very beneficial tendency: smaller values of the objective function (better solutions) tend to be connected with high similarity. It means that there exists global convexity in these instances. The 3 other ones have the values of the correlation rather small (no convexity).

Closer investigation into the nature of these nonconvex instances revealed the fact that their most important subcriterion (HPRCs violations, with weight 10000) was defined as difficult. It means that the less important ones (LPRCs violations and color changes in these cases) do not really affect the value of the objective function (they are totally dominated by the value of HPRCs violations). This observation lead to the conclusion that the similarity function $sim_{cs}$ may be mislead by differences in indices of groups which differ only on those less important constraints, so the concept of ‘weaker’ groups was introduced: groups of vehicles which have only the most important constraints, HPRCs, different. The corresponding similarity function, $sim_{CSWG}$ (common subsequences using ‘weaker’ groups indices only), allowed finding of global convexity in 2 more instances (see the last column in Table 1).

Graphical examples of covariation of the solution quality and the similarity functions are shown on scatter plots in Figure 1.

![Figure 1. Examples of scatter plots of the objective function and the similarity functions for one instance (024_38_5_EP_ENP_RAF).](image)

To summarize, global convexity tests clearly indicate that a distance preserving crossover operator for the CS problem should preserve maximum common subsequences of groups indices (or ‘weaker’ groups indices) common to mated parents. Moreover, these tests also show that preservation of common positions would lead to a rather poor operator.

### 4.3.3 Common sequences preserving crossover

Firstly, this operator finds all common maximum subsequences of groups indices in two given solutions (parents). Then the set of common subsequences is taken from one randomly chosen parent (the donor) and the set of groups indices which are not in the common subsequences is computed (the one-vehicle set). Finally, the common subsequences and the indices from the
one-vehicle set are inserted into the offspring in random order, starting from the beginning of the sequence.

After this operation all the positions with indices inserted from the one-vehicle set are marked as possible to be changed by the local search procedure. All the inserted common subsequences are marked as forbidden to change (to be strict: no removal of a group index from such a subsequence is possible; an insertion in this part is allowed). This way the promising candidate moves for the local search are determined.

4.4 Random subsequence mutation

In order to introduce more diversity to the genetic search, a mutation operator was devised. Its role is to make only a slight perturbation in a mutated solution, so the local search procedure applied afterwards may lead to a different local optimum, still being quite similar to the original solution (not loosing its important features).

Based on these premises, the operator changes in a random manner one short subsequence of the mutated solution. At the beginning, the length and the place of the mutated subsequence is randomly drawn. The place of the mutation may be anywhere in the sequence, while the length of this subsequence is at most 5% of its total length. Finally, the chosen subsequence is randomly reordered.

Candidate moves for the local search procedure are then defined: the mutated subsequence is marked as possible to be changed by this procedure, while the other part of the sequence is marked as forbidden to change (in the same manner as by the crossover operator).

5 Computational experiments

In order to evaluate the implemented algorithm and its components, several experiments were conducted using exactly the same time limitation for one run as defined by the challenge organizers: 600 seconds [2].

The genetic local search algorithm was run in 3 different setups:

- GLS using the mutation only (GLS-M),
- GLS using the crossover only (GLS-C)
- GLS using both of the operators (GLS-MC).

In addition to this, the multiple start local search algorithm (MSLS) was run, starting from the heuristic initial solutions and using the shift neighborhood operator.

Each of the algorithm types was run 5 times for the selected instances, on a PC with Intel Celeron 1.33 GHz processor, running MS Windows XP. Median results of these runs are presented in Table 2, together with reference solutions quality provided by the challenge organizers in instances of the CS problem.

<table>
<thead>
<tr>
<th>Instance name</th>
<th>No. of all vehicles</th>
<th>Reference solution</th>
<th>MSLS</th>
<th>GLS-M</th>
<th>GLS-C</th>
<th>GLS-MC</th>
</tr>
</thead>
<tbody>
<tr>
<td>024_38_5_EP_ENP_RAF</td>
<td>1329</td>
<td>1073792</td>
<td>238190</td>
<td>159267</td>
<td>146686</td>
<td>128016</td>
</tr>
<tr>
<td>025_38_1_EP_ENP_RAF</td>
<td>1232</td>
<td>207790</td>
<td>34097</td>
<td>32086</td>
<td>35505</td>
<td>34394</td>
</tr>
<tr>
<td>039_38_4_RAF_EP_ch1</td>
<td>981</td>
<td>729200</td>
<td>71400</td>
<td>711700</td>
<td>717700</td>
<td>711500</td>
</tr>
<tr>
<td>048_39_1_EP_ENP_RAF</td>
<td>618</td>
<td>430134</td>
<td>65255</td>
<td>35567</td>
<td>26864</td>
<td>27859</td>
</tr>
<tr>
<td>064_38_2_RAF_EP_ENP_ch2</td>
<td>434</td>
<td>284650</td>
<td>4837</td>
<td>3656</td>
<td>5519</td>
<td>3745</td>
</tr>
<tr>
<td>064_38_2_RAF_EP_ENP_ch2</td>
<td>434</td>
<td>319761</td>
<td>307595</td>
<td>306752</td>
<td>311676</td>
<td>306753</td>
</tr>
</tbody>
</table>

Table 2. Median results of 5 experiments for all types of algorithms and the selected instances.
The computation results show that solutions may be easily ameliorated, compared to the reference solutions, simply with the use of the MSLS algorithm. This fact indicates that the initial solutions generation heuristics and the shift neighborhood operator used in the local search are quite efficient.

Comparison of the metaheuristics described in this paper (MSLS and 3 types of GLS) results in a conclusion that GLS is even better than MSLS. Rather surprisingly, it seems that it is the mutation operator which is mainly responsible for the best quality solutions: for 5 instances of 6 it is either GLS-M or GLS-MC which provides the best solutions. However, the crossover operator is also responsible for some best results and its presence in the algorithm may not be neglected (its low speed is perhaps the deteriorating factor).

### 6 Conclusions

In this paper the adaptation of the genetic local search algorithm to the car sequencing problem has been described.

One of the elements of the GLS algorithm, namely the local search procedure, has been carefully designed in order to shorten the time of computation needed to attain a local optimum, as it is often used (after each recombination).

The conducted global convexity tests have revealed an important structure in most of instances of the CS problem, and allowed for the justified construction of recombination operators which aim at preservation of important features of good solutions. The obtained GLS algorithm generates better solutions than MSLS, given the same computation time. This fact sustains the hypothesis that systematic construction of recombination operators based on global convexity tests is a good adaptation pattern for the genetic local search algorithm.

### Acknowledgments

The work of the authors was supported by State Committee for Scientific Research, KBN research grant no. 3 T11F 004 26.

### References


