Tabu Search for a Car Sequencing Problem

Nicolas Zufferey
Faculté des Sciences de l'Administration,
Université Laval, Québec (QC),
G1K 7P4, Canada,
nicolas.zufferey@fsa.ulaval.ca

Martin Studer
Independent consultant,
Lausanne, Switzerland

Edward A. Silver
Operations Management,
Haskayne School of Business,
University of Calgary,
edward.silver@haskayne.ucalgary.ca

Abstract

The goal of this paper is to propose a tabu search heuristic for the car sequencing problem (CSP) used for the ROADEF 2005 international Challenge. This NP-hard problem was proposed by the automobile manufacturer Renault. The first objective of the car industry is to assign a production day to each customer-ordered car and the second one consists of scheduling the order of cars to be put on the line for each production day, while satisfying as many requirements as possible of the plant shops: paint shop and assembly line.

Introduction

The car sequencing problem is well described in (Perron and Shaw 2004). Car sequencing is a standard feasibility problem in the constraint programming community (Dincbas et al. 1997), (Gent 1998), (Hentenryck et al. 1992), (Parrello et al. 1986), (Régin and Puget 1997), (Smith 1997), (Warwick and Tsang 1995). It is known for its difficulty and there exists no definitive method to solve it. Some instances are part of the *Constraint Satisfaction Problem Lib* repository (see www.csplib.org). As with any difficult problem, we can use two approaches to solve it. The first is based on complete search and has the ability to prove the existence or the non-existence of a solution. This approach uses the maximum amount of constraint propagation (Régin and Puget 1997). The second approach is based on local search methods and derivatives. We mention local search (Davenport and Tsang 1999), (Lee et al. 1998), (Puchta and Gottlieb 2002), genetic algorithms (Warwick and Tsang 1995) or ant colony optimization approaches (Solnon 2000). These methods are, by nature, built to find feasible solutions and are not able to prove the non-existence of feasible solutions. Recent efforts have shown important improvements in this area (Michel and Hentenryck 2002).

The problem was proposed by the automobile manufacturer Renault and was the subject of the Challenge ROADEF 2005 (Roadef 2005). In such a problem, a set of cars has to be scheduled minimizing a given objective function. Two additional goals are very important in an industrial context: the algorithm should be quick and robust. Not more than 10 minutes on a PC Pentium4 (1.6Ghz/785 Mo RAM) are allowed to generate a solution, and the mean value over 10 runs is used to measure the quality of a method (i.e. not just the best value over 10 runs). The CSP is NP-complete (Gent 1998) and could be formulated as a constraint satisfaction problem (Gottlieb et al. 2003). Therefore heuristics are appropriate to solve it. In this work, we propose a robust meta-heuristic for the CSP, which is mainly based on tabu search. We will see that our method provides competitive results on the given benchmark instances. See (Zufferey et al. 2004) for more details on this work.

Description of the car sequencing problem

The objectives of the car industry are the following (Roadef 2005): **(1)** assign a production day to each ordered car (which was not part of the Challenge); **(2)** schedule the order of cars to be put on the line for each production day, while satisfying as many requirements as possible of the plant shops (paint shop and assembly line).

The paint shop has to minimize the consumption of paint solvent, which is used to wash spray guns. If two consecutive scheduled cars are not of the same color, a spray gun clean is needed. Therefore, we would like to minimize the number of paint color changes in the sequence of scheduled vehicles. A *solution* consists of a sequence of the last cars of production day $D-1$ (which are already scheduled) and all cars of day D. In any particular day D being scheduled, after B (batch size) consecutive cars with the same color, it is necessary (i.e. a hard constraint) to change color, even if it seems natural to clean the spray guns and continue with the same color. If the solution satisfies the hard constraint, it is said to be *admissible*.

In order to smooth the workload of the assembly line, vehicles that require special assembly operations have to be evenly distributed throughout the total processed cars. These vehicles should not exceed a given quota over any sequence of vehicles. This requirement is modelled by a ratio constraint N/P. Ratio constraints are associated with car characteristics which require extra operations on the assembly line (for instance, sun-roof, air conditioning, etc). The meaning of a ratio constraint N_i/P_i is that at most

N_i cars, in each consecutive sequence of P_i cars, can be associated with constraint i.
There are two classes of ratio constraints: *high priority level* constraints and *low priority level* constraints. High priority level constraints are due to car characteristics that require a heavy workload on the assembly line. Low priority level constraints result from car characteristics that cause small inconvenience to production. Let H (resp. L) be the set of all high (resp. low) level ratio constraints. High priority level ratio constraints must be satisfied preferentially to low priority level constraints.
Ratio constraints are soft constraints: the complete satisfaction of all the ratio constraints cannot be ensured beforehand when a production day is scheduled. The problem may be over-constrained. Hence the optimization objective is to minimize the number of violations of ratio constraints.
Violations of a specific high level ratio constraint H_i are said to be H_i*-violations*, whereas violations of any high level ratio constraint are called H*-violations*. A solution without any H_i-violations is said to be H_i*-feasible*, and one without any H-violations is called H*-feasible* (similar definitions hold for low level ratio constraints). Renault classifies the high priority level constraints of each instance into two sub-classes: *Easy* to satisfy (i.e. Renault found an H-feasible solution), and *Difficult* to satisfy (i.e. Renault failed to find an H-feasible solution, but an H-feasible solution may exist).

When production day D is scheduled, the ultimate scheduled vehicles of production day $D-1$ must be taken into account. Vehicles of production day $D-1$ are already scheduled, therefore their positions cannot be changed. The computation of the number of violations (of ratio constraints) on production day D must take into account the last cars of production day $D-1$.
Production day $D+1$ is ignored while scheduling production day D. For each ratio constraint, we compute the violations concerning the ultimate vehicles of production day D, as if the first scheduled cars of production day $D+1$ are not associated with this ratio constraint.

We will now formulate the problem in a mathematical way. A solution s is a vector composed of the ultimate n' cars of production day $D-1$ (which are already scheduled) and the n cars of production day D. Thus, "$s(i) = j$" means that we have car j at position i in the solution s. The following information is provided by Renault for each car c: $PD(c)$, which is the production day of car c, where $PD(c) \in \{D-1, D\}$; $Color(c)$, which is the color of car c (integer), where $Color(c) \in \{1, \ldots, 50\}$; $H_i(c) \in \{0, 1\}$, where $H_i(c) = 1$ indicates that car c is involved in constraint H_i; $L_i(c) \in \{0, 1\}$, where $L_i(c) = 1$ indicates that car c is involved in constraint L_i.
The main objectives are: for the paint shop to minimize the number of color changes, and for the assembly line to minimize the number of violations of the ratio constraints. This leads to the overall objective function $F(s) := \alpha \cdot FH(s) + \beta \cdot FL(s) + \gamma \cdot FC(s)$, where $FH(s) =$ number of H-violations in solution s, $FL(s) =$ number of L-violations in solution s, and $FC(s) =$ number of paint color changes in solution s. The sets of weights α, β, γ were specified by Renault for three different types of instances of the problem. The nature of the problem depends on the values of α, β, γ that appear in the definition of $F(s)$. For instances of type A, we have $\alpha, \beta, \gamma \in \{0; 1; 100; 10,000\}$ and for instances of type B and X, we have $\alpha, \beta, \gamma \in \{0; 1; 100; 1,000,000\}$. We describe below how to compute these functions. Note that the ways of computing the various components of F were not provided by the organizers of the Challenge.

In order to compute $FH(s)$, we need the following quantities. $FH_i(s)$ is the number of H_i-violations (of a specified constraint H_i) in solution s. W_k is a window of size k, i.e. a sequence of k consecutive cars. $FH_i(s, W_k)$ is the number of H_i-violations in a specified window W_k in solution s, which is equal to $\max\{$(number of cars concerned with H_i in window W_k)$- N_i; 0\}$. In most of the cases we have $k = P_i$. However, for cars scheduled near the end of day D, we have to consider values of k smaller than P_i in order to give proper weight to associated violations. For a ratio constraint N/P, the last windows will have a decreasing length between $P-1$ and $N+1$. If in these windows, the number of cars associated with the ratio constraint is strictly greater than N, there will be violations, which are independent of the first vehicles of production day $D+1$. In the following, expression "$W_P = a$" means that the window W of size P ends at position a. These considerations lead to the following equations: $FH_i(s, W_k) = \max[-N_i + \sum_{c \in W_k} H_i(c); 0]; FH_i(s) = \sum_{W_{P_i}=a}^{b} FH_i(s, W_{P_i}) + \sum_{k=N_i+1}^{P_i-1} FH_i(s, W_k = n'+n); FH(s) = \sum_{i|H_i \in H} FH_i(s)$. We set a and b as follows. The first window contains the $P_i - 1$ last scheduled cars of production day $D-1$ and the first scheduled car of production day D. The last window of size P_i contains the P_i last scheduled cars of production day D. Consequently, $a = (n'+1)$ and $b = (n'+n)$. Note that the computation of $FL(s)$ is done in exactly the same way, and $FC(s)$ is simply equal to the number of color changes in the sequence, counted from the last car of day $D-1$.

Tabu search for the CSP

Let X be the set of all solutions of a given problem, and let F be an objective function which has to be minimized on X. A set $N(x)$, called *neighborhood* of x, is associated with each solution $x \in X$. The solutions in $N(x)$ (also called *neighbors* of x) are obtained from x by performing local changes called *moves*. A very well-known local search method is tabu search, which was proposed in (Glover 1989). Its basic version can be described as follows. Tabu search needs an initial solution x_0 in X as input. It then successively generates solutions $x_1, x_2, \ldots$ in X such that $x_{i+1} \in N(x_i)$. When a move is performed from x_i to x_{i+1}, it is forbidden (tabu) to perform the reverse of such

a move (with some exceptions) for a certain number of iterations. Solution x_{i+1} is set equal to arg $\min_{x \in N'(x_i)} F(x)$, where $N'(x)$ is a subset of $N(x)$ containing all solutions x' which can be obtained from x by performing a move that is not tabu, or such that $F(x') < F(x^*)$, where x^* is the best solution found so far. The process is stopped when a fixed number $Iter$ (parameter) of iterations without improving x^* have been performed. Many variants and extensions of this basic algorithm can be found in (Glover and Laguna 1997).

We first group cars with the same characteristics in equivalent classes. We define a *H-distance* between two cars x and y as the number of high level constraints they do not have in common. The *DistH* matrix contains the H-distances of all couples (x,y). Each element is computed as follows: $DistH(x,y) := \sum_i [H_i(x) + H_i(y)] \bmod 2$. According to the H-distance, we define *H-classes*. Two cars x and y are in the same H-class if $DistH(x,y) = 0$.

We will propose a tabu search for each component of F: TABUFH, TABUFL and TABUFC will respectively focus on FH, FL and FC. The common points are described as follows. A neighbor solution s' is obtained from a solution s by swapping two cars x and y, which are in the car set associated with day D. We denote this move by $m(x,y)$. Note that such a type of move is not new as it was also used, for example, in the local search proposed in (Puchta and Gottlieb 2002). However, we will see below that the proposed way of evaluating a neighbor solution is original and very efficient. We will only consider moves leading to a solution s' which respects the hard constraint B. At each iteration, we generate X^{cand} (parameter) different *x-candidates*, where an x-candidate is a possible candidate for the move $m(x,y)$. For each x-candidate, we then generate Y^{cand} (parameter) different *y-candidates*, and finally we choose the best y-candidate according to the minimization of $\Delta FC(s,m)$, $\Delta FH(s,m)$ or $\Delta FL(s,m)$. If we focus on FH, preliminary experiments led us to use $X^{cand} = 2$ and $Y^{cand} = n - \mid \{\text{car } z \mid DistH(x,z) = 0\} \mid$. In other words, we use a small number of x-candidates and consider all possible relevant y-candidates for each x-candidate. We select the best of the $X^{cand} \cdot Y^{cand}$ moves and break ties randomly. If we focus on FL, preliminary experiments led us to $X^{cand} = 100$ and $Y^{cand} = n - \mid \{\text{car } z \mid DistL(x,z) = 0\} \mid$, i.e. a much larger number of x-candidates. If we focus on FC, preliminary experiments showed that $X^{cand} = 1$ seems appropriate in this case.

In order to generate an initial solution when FH is the most important component of F, we build a solution step by step (i.e. car by car). Let s_p be the current partial solution. At each step, we choose the next car c such that it minimizes $FH(s_p + \{c\})$. More precisely, we compute $\Delta FH(c) = \sum_i \Delta FH_i(c)$, where $\Delta FH_i(c)$ is the number of additional H_i-violations (over all possible windows which include c) induced by car c, when scheduled. If more than one car provides a minimum value for $\Delta FH_i(c)$, we break ties by choosing c with larger value of $SatH(c) := \frac{1}{|H|} \cdot \sum_{i=1}^{|H|} H_i(c)$, in order to avoid having many cars involved in lots of ratio constraints at the end of the sequence. If there are still several possibilities, we then break ties with the next most important component of the objective function (i.e. $\Delta FL(c)$ if FL is more important than FC, $\Delta FC(c)$ otherwise), and finally with the third component of the objective function (if any).
Note that if FC is the most important component, it is easy to generate an FC-optimal solution by simply grouping cars of the same color and respecting the hard constraint B.

Description of TabuFC

We first determine a car x of the solution s, that is not tabu and that induces a maximum number of color changes. In order to decrease the value of $FC(s)$, we have to choose a non tabu car y carefully. Suppose we have the following situation involving six cars $a-x-b$ and $c-y-d$. It is promising to switch x and y if $Color(x) \in \{(Color(c);\ Color(d)\}$ and $Color(y) \in \{(Color(a);\ Color(b)\}$. If we switch two cars x and y, we forbid moving cars x and y for t iterations, where we randomly generate $t \in \{t_{min}, \ldots, t_{max}\}$ after each move. Preliminary experiments showed that $t_{min} = 5$ and $t_{max} = 30$ seem to be appropriate values.

Description of TabuFH (or TabuFL)

In order to reduce the FH-value of the solution s, we would like to move the car that currently induces the highest number of H-violations. We define a *H_i-violated* window as a window for which we have at least one violation according to H_i. Let $VH_i(s,c)$ be the number of H_i-violations induced by car c (and only car c!) in s. This number is actually the number of times that c is in a H_i-violated window and $H_i(c) = 1$. We set $VH(s,c) = \sum_{i=1}^{|H|} VH_i(s,c)$. It is promising to move a car c such that $\sum_{i|FH_i(s)>0} VH_i(s,c)$ is large. In order have a small CPU time in the generation and selection of a neighbor solution s' of s, we propose a refined delta computation, i.e. a way to compute $\Delta FH(s,m) = \sum_{i|H_i(x)\neq H_i(y)}^{|H|} \Delta FH_i(s,m)$ very quickly. Let $PVH_i(s,c)$ be the number of H_i-violations a car c' such that $H_i(c') = 1$ will create at position $p(c,s)$ when $H_i(c) = 0$ in solution s. We set $PVH(s,c) = \sum_{i=1}^{|H|} PVH_i(s,c)$. We can now define the advantage $AdvH(y \to x)$ of putting y at position $p(x,s)$ of solution s as the number of violations that y will eliminate in the area of position $p(x,s)$ of car x if we switch x and y. $AdvH(y \to x) := \sum_{i|H_i(x)=1}^{|H|} VH_i(s,x) \cdot [1 - H_i(y)]$. The disadvantage $DisadvH(y \to x)$ should be proportional to the number of violations y will add in the area of position $p(x,s)$ of car x if we put y instead of x. $DisadvH(y \to$

$x) := \sum_{i|H_i(x)=0}^{|H|} PVH_i(s,x) \cdot H_i(y)$. This leads to $\Delta FH(s,m) = DisadvH(y \leftrightarrow x) - AdvH(y \leftrightarrow x)$ which is very quick to evaluate given the matrices VH_i and PVH_i, which are updated only one time at the end of each iteration of tabu search. Moreover, we actually do not need to update the whole matrices VH_i and PVH_i, but only the parts near the components associated with cars x and y.

After a move $m(x,y)$, we forbid putting a car c at position $p(x,s)$ if $DistH(x,c) = 0$, i.e. if c and x belong to the same H-class (the same holds for c and y). The duration t of the tabu status depends on the quality of the move. Preliminary experiments showed that the following way of updating t is appropriate: $t = -10 \cdot \Delta FH(s,m)$ if $\Delta FH(s,m) < 0$; $t = 3$ if $\Delta FH(s,m) = 0$; $t = 1$ if $\Delta FH(s,m) > 0$. This dynamic tabu tenure, where the duration of the tabu status depends on the quality of the move m, performs better than if we randomly choose $t \in \{t_{min}, \ldots, t_{max}\}$.

General strategy and results

The organizers of the Challenge provided test set A at the very beginning of the Challenge in order to select, in what we call the first round, a pool of 24 candidates (out of 55) for the final round. We placed 6th at this stage of the competition. Such a test set was, by far, the most studied by the candidates, and we only tuned our algorithm according to it. Thus, we focus only on test set A in this paper. Test set B was given at the end of the first round only for extra tuning purposes, and not for another selection round. Finally, test set X was used to rank the best candidates in the final round. Only the organizers ran the algorithms (provided by the teams) on these X instances, for which we were ranked at position 17. One could argue that performance on set X is a better independent measure of the different approaches. However, we believe that some useful insights are obtained by focusing on the test set where we did so well. Moreover, we think that there may have been some unforeseen glitches when our algorithm was run by the organizers on set X.

Remember that, in the most general case, we have to minimize an objective function $F(s) = 10000\,F_1(s) + 100\,F_2(s) + F_3(s)$, where $F_i(s) \in \{FH, FL, FC\}$. Basically, the general strategy described below (steps A, B and C) is applied on every instance. Note that if we reach 600 seconds in any procedure, we immediately stop the process and return the best visited solution.

A Set $F^* = \infty$

B While $600 - t_3$ seconds are not reached, do

1. generate an initial solution by applying the greedy algorithm focusing on F_1;
2. if $F_1 \neq FC$, apply the tabu search focusing on F_1 (i.e. TABUFH) during t_1 seconds or q_1 iterations; note that we stop if we get $F_1(s) = 0$; in order to save time, we ignore F_2 and F_3;
3. without augmenting F_1, apply the tabu search focusing on F_2 during t_2 seconds or q_2 iterations; note that we stop if we get $F_2(s) = 0$; in order to save time, we ignore F_3;
4. let s be the so obtained solution; if $F(s) < F^*$, set $F^* = F(s)$ and $s^* = s$

C Without augmenting F_1 and F_2, apply the tabu search on s^* focusing on F_3 during t_3 seconds;

Preliminary experiments showed that it is slightly better to avoid to increase F_i when focusing on F_{i+1}, for $i \in \{1,2\}$. With such a strategy, when focusing on F_{i+1}, we forbid all neighbors that would worsen objective F_i. Thus, we rule out solutions where it is conceivable that there is enough improvement in F_2 and F_3 to more than compensate for a slight deterioration in F_1. However, the strategy worked well, on average, perhaps because the very limited computational time is not conducive to investigating many poorer solutions. In the following, we will specify the chosen values for t_i and q_i, for $i \in \{1,2,3\}$. Note that if we give a value for t_i, we do not need to give any value for q_i (and vice versa). In our best strategy (i.e. the best one we tune according to preliminary experiments) for the paint shop, we set $q_2 = 100$ and $t_3 = 10$ seconds if F_3 exists, otherwise we set $t_3 = 0$ (i.e. for the PHE instance below). For the easy assembly line instances, because the high priority ratio constraints are easy, we set $t_1 = 10$ seconds, which is always enough to have $FH = 0$. We set $q_2 = 1000$ and $t_3 = 10$ seconds. Finally, for the difficult assembly line instances, we set $q_1 = 5000$, $q_2 = 2000$, $t_3 = 10$ if the third component exists, $t_3 = 0$ otherwise.

Note first that any name of instance is straightforward. For example, PHEL1 means that FC (i.e. the **P**aint shop) is more important than FH, which is more important than FL. The "E" after the "H" indicates that the H ratio constraints are considered as **E**asy by Renault. There will be a "D" if the H ratio constraints are considered as **D**ifficult by Renault. When we have to consider more than one instance of such a type, we indicate it by a number at the last position of the name of the instance. For example, we have to consider below three instances of type PHEL, namely PHEL1, PHEL2 and PHEL3. For each instance of test set A, we provide, in Tables 1, 2 and 3, the results obtained by Renault (they mention in (Roadef 2005) that they used a simulated annealing method), and we give the average (over 10 runs) result obtained by our best strategy (denoted by $Best$). In addition, we provide the best average result obtained by a participant of the Challenge (denoted by $BestComp$ for "best competitor"), and the best average result obtained with our algorithm in the case it is performed by the organizers (they use different seeds than us) of the Challenge (this is referred to as $InChallenge$). Because $BestComp$ may differ from one instance to another (thus the best methods may differ too), we will not give any name of competitors (or any name of best methods), and we refer to the web site of the Challenge for more details (Roadef 2005). The last line of Tables 1, 2 and 3 (labelled "Our Rank") indicates the rank we obtained in the Challenge for the considered instance. In brackets, we indicate our average rank over the associated instances. In addition, we

indicate for each instance: the numbers n' and n of cars respectively associated with day $D-1$ and D, the number $|H|$ (resp. $|L|$) of ratio constraints of type H (resp. L), and the number $|C|$ of colors.

First, we can observe that the gap between the results obtained with our best strategy and the ones obtained by the organizers of the Challenge if they run our program with other seeds is very small. The organizers even get better results than us on some instances. Such an observation shows that our algorithm is robust, i.e. does not strongly depend on the seed of the random generator. We respectively obtained rank 12, 8 and 1 on the paint shop instances (4 instances), the easy assembly line instances (5 instances), and the difficult assembly line instances (7 instances). Such a good performance led us to the general 6th rank among 55 candidates. We observed that our algorithm always obtained the best results on the first component of the objective function F. As we did, several teams were able to generate FC-optimal solutions on the paint shop instances, and H-feasible solutions on the easy assembly line instances. For these instances, the second and third components of F were helpful to rank the teams. On the contrary, only a few teams performed well on FH when considering the difficult assembly line instances. This explains the first rank reached by our algorithm on such seven instances. The large number of visited solutions probably explains the success of our method on the first component of F, and we think that our heuristic worked not as well on the other component because of the additional constraints we have to deal with.

Results for the paint shop instances

	PHEL1	PHEL2	PHEL3	PHE
$(n'; n)$	(99;335)	(14;485)	(29;875)	(27;954)
$(B; \mid C \mid)$	(15;12)	(450;12)	(15;14)	(15; 14)
$(\mid H \mid; \mid L \mid)$	(4;2)	(3;6)	(7;2)	(5;-)
Renault FC	30	11	64	69
Best FC	27	11	63	68
InChallenge FC	27	11	64	68
BestComp FC	27	11	63	68
Renault FH	197	48	462	392
Best FH	368.7	39.4	433.8	236.2
InChallenge FH	367.8	39.4	436	244.6
BestComp FH	367	39	423	156
Renault FL	61	5	883	-
Best FL	106.9	168.2	830.9	-
InChallenge FL	101.2	151.4	832.4	-
BestComp FL	52	1	782	-
Our Rank (12)	15	7	15	7

Table 1: Results for the paint shop instances.

The obtained results are detailed in Table 1. First, we would like to mention that the code we gave to Renault contained an error in our greedy method focusing on FC, it is why we use 64 colors instead of 63 for instance PHEL3. This error was removed from the code of the best strategy proposed here. We can see that for instances PHEL1 and PHE, Renault is not able to generate FC-optimal solutions. On instance PHEL1, Renault obtains a poor FC-value, but a very good FH-value. This is not surprising: the worse the results are on a component of F, the better they may be on another component. For instances on which Renault and our strategy get FC-optimal solutions, we get lower FH-values than Renault. The FL-values of Renault are surprisingly low on the first two instances, which probably means that they give more importance to FL than we did. If we compare the results labelled by $InChallenge$ with the $BestComp$ ones, we can observe the following. For PHEL1 and PHEL2, the main differences occur in the last component of the objective function, whereas for PHEL3 and PHE, $BestComp$ performs better from the second component of the objective function. More generally, we will again see in Tables 2 and 3 that we always get the best results on the first component of F, and that $BestComp$ generally performs better from the second component onward. Such an observation might indicate that we should augment t_2 (or resp. t_3). However, such an action will indirectly reduce the time that we spend on F_1 (or resp. on F_2). Therefore, it will probably decrease our global performance. This kind of behavior was confirmed by additional experiments not shown in this paper.

Results for the easy assembly line instances

	HEPL1	HEPL2	HEPL3	HEPL4	HELP
$(n'; n)$	(99;335)	(14;485)	(29;875)	(228;1004)	(228;1004)
$(B; \mid C \mid)$	(15; 12)	(450; 12)	(15;14)	(10; 24)	(10; 24)
$(\mid H \mid; \mid L \mid)$	(4; 2)	(3; 6)	(7;2)	(4;18)	(4;18)
Renault FH	28	2	2	0	0
Best FH	0	0	0	0	0
InChallenge FH	0	0	0	0	0
BestComp FH	0	0	0	0	0
Renault FC	46	70	195	290	290
Best FC	38.2	39.6	136.3	232.9	840.7
InChallenge FC	38.8	38.8	137.4	232.8	833.2
BestComp FC	34	31	113	232.8	754.6
Renault FL	50	2	787	2075	2075
Best FL	95.1	63.4	802.9	3694.5	377.4
InChallenge FL	107.8	49.4	801.4	3705.2	377.2
BestComp FL	51	0	761	3705.2	133.6
Our Rank (8)	17	14	9	1	15

Table 2: Results for the easy assembly line instances.

We can see in Table 2 that for the instances HEPL1, HEPL2 and HEPL3, Renault is not able to generate H-feasible solutions. This is surprising because these instances are classified as easy, i.e. Renault should get H-feasible solutions (by definition). Another strange thing is that Renault generates the same solution for the instances HEPL4 and HELP, even though they have different priorities. Note that in general Renault has good values for the last component of the objective function, this is, again, an indicator that Renault gives more importance (or weight) to this last component than we did (and that is indicated by the associated relative weights in the objective function). We can remark that our method obtained very good results on instance HEPL4 (the best ones in the Challenge!).

Results for the difficult assembly line instances

For the first component of the objective function (i.e. for FH) in Table 3, the value in brackets indicates the number of times (over 10 runs) the associated algorithm reaches what we suspect to be the lower bound. We can see that the FH-values obtained by the best strategy are far better

	HDP	HDPL1	HDPL2	HDPL3
$(n'; n)$	(27;954)	(18;600)	(14;1315)	(14; 1260)
$(B; \mid C \mid)$	(20; 14)	(10;12)	(10; 13)	(10;13)
$(\mid H \mid; \mid L \mid)$	(5; -)	(5; 12)	(5; 8)	(5;8)
Renaut FH	115	35	98	73
Best FH	13.1 (9)	0.1 (9)	4 (10)	4 (10)
InChallenge FH	13.4 (6)	0 (10)	4.2 (8)	4 (10)
BestComp FH	13.4 (6)	0 (10)	4 (10)	4 (10)
Renault FC	229	182	468	363
Best FC	138.5	179.9	302.8	296.1
InChallenge FC	137.2	180.8	303.2	296.2
BestComp FC	137.2	179	292.4	267.2
Renault FL	-	861	99	205
Best FL	-	1225.4	165.2	311.8
InChallenge FL	-	1181.4	168.8	293.4
BestComp FL	-	642	106.4	160.4
Our Rank (1)	1	3	10	3

	HDLP1	HDLP2	HDLP3	
$(n'; n)$	(18;600)	(14;1315)	(14;1260)	
$(B; \mid C \mid)$	(10;12)	(10;13)	(10;13)	
$(\mid H \mid; \mid L \mid)$	(5;12)	(5;8)	(5;8)	
Renaut FH	42	106	82	
Best FH	0.5 (5)	4 (10)	4 (10)	
InChallenge FH	0.2 (8)	4 (10)	4 (10)	
BestComp FH	0 (10)	4 (10)	4 (10)	
Renault FC	334	392	464	
Best FC	369.1	398.4	398.7	
InChallenge FC	368.2	394	366.8	
BestComp FC	351	359.8	366.8	
Renault FL	98	134	77	
Best FL	111.7	64	30.7	
InChallenge FL	99.4	63.6	29.6	
BestComp FL	64	49.8	10.4	
Our Rank (1)	12	4	2	

Table 3: Results for the difficult assembly line instances.

than the ones obtained by Renault. Moreover, we proved the H-feasibility of instances HDPL1 and HDLP1.

Conclusion

In this paper, we presented a heuristic mainly based on three tabu search procedures in order to solve an NP-complete car sequencing problem. Due to the different weights of the components that appear in the global objective function F, and to the small amount of time allowed for each instance, we decided to work on one component at a time. Therefore, we had to be careful not to degenerate a solution according to a component of F on which we already worked. Furthermore, at each iteration of the proposed tabu search procedures, it is important to consider a set of neighbor candidate solutions that can be evaluated quickly, in order to visit as many solutions as possible. The obtained results showed that our method is competitive and robust in comparison with the algorithms proposed by the other participants of the Challenge. As we usually obtain the best results according to the first component of F, an interesting avenue of research would be to find a way to better consider the other components of F.

Acknowledgement. The research leading to this paper was partially carried out while the first author was a postdoctoral assistant in the University of Calgary and the research was supported by the Swiss National Science Foundation and by the Natural Sciences and Engineering Research Council of Canada under Grant A1485. We also would like to thank the four anonymous reviewers for their helpful comments on the paper.

References

Davenport A., and Tsang, E. 1999. Solving Constraint Satisfaction Sequencing Problems by Iterative Repair, *In: Proceeding of the First International Conference on the Practical Applications of Constraint Technologies and Logic Programming*: 345–357

Dincbas, M., Simonis, H., and Hentenryck, P.V. 1997. Solving the Car-Sequencing Problem, *In: Constraint Logic Programming*, Kodratoff, Y., (Eds), Proceedings ECAI-88, 290–295

Glover, F. 1989a. Tabu Search - Part I, *ORSA Journal on Computing **1***: 190–205

Glover, F., and Laguna, M. 1997. *Tabu Search*, Kluwer Academic Publishers, Boston

Gent, I.P. 1998. Two Results on Car Sequencing Problems, Technical Report APES APES-02-1998, University of St. Andrews

Gent, I.P., and Walsh, T. 1999. CSPLib: A Benchmark Library for Constraints, Technical Report APES-09-1999, available from http://4c.ucc.ie/∼tw/csplib

Gottlieb, J., Puchta, M., and Solnon, C. 2003. A Study of Greedy, Local Search and Ant Colony Optimization Approaches for Car Sequencing Problems, G. Raidl et al. (Eds), Springer, LNCS 2611, *Applications of Evolutionary Computing*: 246–257

Hentenryck, P.V., Simonis, H., and Dincbas, H. 1992. Constraint Satisfaction using Constraint Logic Programming, *Artificial Intelligence **58***: 113–159

Lee, J.H.M., Leung, H.F., and Won, H.W. 1998. Performance of a Comprehensive and Efficient Constraint Library using Local Search, Springer, LNAI 1502, *In: Proceeding of 11th Australian Joint Conference on Artificial Intelligence*: 191–202

Michel, L., and Hentenryck, P.V. 2002. A Constraint-Based Architecture for Local Search, *In: Proceedings of the 17th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications, ACM Press*: 83–100

Parrello, B., Kabat, W., and Wos, L. 1986. Job-Shop Scheduling using Automated Reasoning: A Case Study of the Car-Sequencing Problem, *Journal of Automated Reasoning **2***: 1–42

Perron, L., and Shaw, P. 2004. Combining Forces to Solve the Car Sequencing problem, *J.-C. Régin and M. Rueher (Eds): CPAIOR 2004, LNCS 3011*: 225–239, Springer-Verlag Berlin Heidelberg

Gottlieb, J., Puchta, M., 2002. Solving Car Sequencing Problems by Local Optimization, Springer, LNCS 2279, *In: Applications of Evolutionary Computing*: 132–142

Régin, J.C., and Puget, J.F. 1997a. A Filtering Algorithm for Global Sequencing Constraints, *In Smolka, G., (Eds): Principles and Practice of Constraint Programming - CP97*: 32–46, Springer Verlag LNCS 1330

ROADEF Challenge, 2005. Description of the problem, http://www.prism.uvsq.fr/∼vdc/ROADEF/CHALLENGES/2005/challenge_2005_html

Solnon, C. 2000. Solving Permutation Constraint Satisfaction Problems with Artificial Ants, *In: Proceedings of ECAI2000*: 118–122, Presented at the ILOG Solver and ILOG Scheduler 2nd International Users Conference, Paris, July 1996

Smith, B. 1997. Succeed-First or Fail-First: A Case Study in Variable and Value Ordering Heuristics, *In: Proceedings of PACT1997*: 321–330, IOS Press,

Warwick, T., and Tsang, E. 1995. Tackling Car Sequencing Problems Using a Generic Genetic Algorithm Strategy, *Evolutionary Computation **3(3)***: 267–298

Zufferey, N., Studer, M., and Silver, E. A. 2004. A General Heuristic based on Greedy Algorithms and Tabu Search for a Car Sequencing Problem, Technical Report, Haskayne School of Business, University of Calgary, Canada