archive ouverte UNIGE
http://archive-ouverte.unige.ch

Article

A solution method for a car fleet management problem with maintenance constraints

HERTZ, Alain, SCHINDL, David, ZUFFEREY, Nicolas

Abstract

The problem retained for the ROADEF'99 international challenge was an inventory management problem for a car rental company. It consists in managing a given fleet of cars in order to satisfy requests from customers asking for some type of cars for a given time period. When requests exceed the stock of available cars, the company can either offer better cars than those requested, subcontract some requests to other providers, or buy new cars to enlarge the available stock. Moreover, the cars have to go through a maintenance process at a regular basis, and there is a limited number of workers that are available to perform these maintenances. The problem of satisfying all customer requests at minimum cost is known to be NP-hard. We propose a solution technique that combines two tabu search procedures with algorithms for the shortest path, the graph coloring and the maximum weighted independent set problems. Tests on benchmark instances used for the ROADEF'99 challenge give evidence that the proposed algorithm outperforms all other existing methods (thirteen competitors took part to this contest).

Reference

HERTZ, Alain, SCHINDL, David, ZUFFEREY, Nicolas. A solution method for a car fleet management problem with maintenance constraints. *Journal of Heuristics*, 2009, vol. 15, p. 425-450

DOI : 10.1007/s10732-008-9072-4

Available at:
http://archive-ouverte.unige.ch/unige:26168

Disclaimer: layout of this document may differ from the published version.

UNIVERSITÉ DE GENÈVE

J Heuristics (2009) 15: 425–450
DOI 10.1007/s10732-008-9072-4

A solution method for a car fleet management problem with maintenance constraints

Alain Hertz · David Schindl · Nicolas Zufferey

Received: 26 October 2006 / Revised: 4 February 2008 / Accepted: 19 February 2008 /
Published online: 6 March 2008
© Springer Science+Business Media, LLC 2008

Abstract The problem retained for the ROADEF'99 international challenge was an inventory management problem for a car rental company. It consists in managing a given fleet of cars in order to satisfy requests from customers asking for some type of cars for a given time period. When requests exceed the stock of available cars, the company can either offer better cars than those requested, subcontract some requests to other providers, or buy new cars to enlarge the available stock. Moreover, the cars have to go through a maintenance process at a regular basis, and there is a limited number of workers that are available to perform these maintenances.

The problem of satisfying all customer requests at minimum cost is known to be NP-hard. We propose a solution technique that combines two tabu search procedures with algorithms for the shortest path, the graph coloring and the maximum weighted independent set problems. Tests on benchmark instances used for the ROADEF'99 challenge give evidence that the proposed algorithm outperforms all other existing methods (thirteen competitors took part to this contest).

Keywords Car fleet management · Graph optimization · Integer linear programming · Tabu search

A. Hertz · D. Schindl
Département de Mathématiques et de Génie Industriel, École Polytechnique, Montréal, Canada

A. Hertz
e-mail: Alain.Hertz@gerad.ca

D. Schindl
e-mail: David.Schindl@gerad.ca

N. Zufferey (✉)
Faculté des Sciences Économiques et Sociales, Section des Hautes Études Commerciales (HEC),
Université de Genève, Genève, Switzerland
e-mail: Nicolas.Zufferey@hec.unige.ch

1 Introduction

The problem retained for the ROADEF'99 international challenge was an inventory management problem for a car rental company. A complete description of the ROADEF'99 international challenge can be found in ROADEF (1999). We give below a shorter description. The reader interested in a survey on inventory management may refer to Silver et al. (1998).

A car rental company manages a stock of cars of different types. It receives requests from customers asking for cars of specific types for a given time horizon. Basically a request is characterized by its start and end times, by a required car type, and by the number of required cars. All requests are supposed to be known for a given time horizon (for example, a few months or a year). The satisfaction of all customer requests is mandatory. If there are not enough cars available in stock, the company can react in three different ways:

- it can offer an *upgrade* to the customer, which means that some desired cars are replaced by cars of a "better" resource type (e.g., larger, more comfortable and more expensive cars). The customer of course receives a bill according to his request, and the company has to pay for the upgrade cost;
- the company can decide to subcontract some requests to other providers, which is generally a very expensive alternative;
- the third possibility is to purchase new cars, which then belong to the stock of the company for the rest of the time horizon.

Each one of the above alternatives has a known cost, and the problem is to satisfy all requests at minimum cost.

Notice that the set of cars assigned to a customer cannot be changed in the course of its request. For example, suppose that customers X and Y both need a car of type A, customer X from May 7 to 17, and customer Y from May 13 to 26. Assume also there is only one car of type A in stock at this period. It is then forbidden to assign the available car to X from May 7 to 17, and to Y from May 18 to 26, and to satisfy customer Y from May 13 to 17 with another car (i.e., an upgrade, or a car obtained from another provider). This would be too inconvenient for customer Y.

The problem is further complicated with constraints that impose to perform a maintenance on each car on a regular basis. More precisely, a maximum time of use without maintenance is given for each car type, and each maintenance is characterized by a duration and a number of workers needed to perform it. The company has a fixed number of maintenance workers which means that the maintenances should be scheduled so that the capacity of the workshop is never exceeded.

The overall problem is NP-hard in the strong sense (see ROADEF 1999). Thirteen competitors took part to the ROADEF'99 challenge, and each one has developed a heuristic algorithm to solve the problem. We propose in this paper a new heuristic and show that it outperforms all existing algorithms.

The literature that considers the car rental business mainly deals with revenue management and pool control systems (e.g., Edelstein and Melnyk 1997; Geraghty and Johnson 1997; Pachon et al. 2003). Fink and Reiners (2006) have recently proposed a model and a solution method for a car rental logistics problem that

involves short-term decisions about the transportation and deployment of cars with regard to optimizing fleet utilization while maintaining a high service level. Our paper complements the above developments by proposing optimization techniques for the assignment of cars to customers, taking into account maintenance constraints.

The paper is organized as follows. In the next section, we define the notation and give a mathematical formulation of the problem. An overview of the proposed solution method is given in Sect. 3. Various subproblems are analyzed in Sect. 4, and we propose integer linear models for some of them, while others are modeled as shortest path, graph coloring, or maximum weighted independent set problems. Section 5 contains the description of our solution method, which combines two tabu search procedures with graph optimization techniques. Computational experiments on a set of benchmark instances are reported and discussed in Sect. 6. Finally, some conclusions are drawn.

2 Notation and formal description

The time horizon is a finite discrete interval, denoted $H = [Start, End]$, the interval unit being the day. It is assumed that the number W of workers that can simultaneously work on maintenances is constant during the whole time horizon H. At the beginning of the time horizon, the cars in stock may have been used without maintenance, and some requests are possibly being handled. For the sake of simplicity, we will not consider this initial state. Moreover, we suppose all costs to be constant during the time horizon. These assumptions have no effect on the complexity of the problem.

Let $\mathcal{T}$ be the set of car types. For two different car types t and t', we say that t is an upgrade of t', and we write $t \gg t'$, if any car of type t can be offered to a customer requiring a car of type t'. For a car type $t \in \mathcal{T}$, we denote:

- $C(t)$ the set of cars of type t available at the beginning of the time horizon;
- $u(t)$ the maximum time of use that can separate two maintenances on a car of type t;
- $m(t)$ the duration of a maintenance on a car of type t;
- $w(t)$ the number of workers required for a maintenance on a car of type t;
- $b(t)$ the maximum number of cars of type t that can be bought during the time horizon;
- $F_a(t)$ the fixed cost of assigning a car of type t to a customer;
- $k_a(t)$ the cost per day of the assignment of a car of type t to a customer;
- $k_b(t)$ the cost of buying a car of type t;
- $k_d(t)$ the cost per day of a car of type t in stock (rented or not);
- $k_s(t)$ the cost per day of subcontracting a car of type t to another provider;
- $k_m(t)$ the cost of a maintenance on a car of type t (this cost being typically much smaller than the other costs).

Let $\mathcal{R}$ be the set of requests during the time horizon. For a request $r \in \mathcal{R}$, we denote:

- $\alpha(r)$ and $\beta(r)$ the start and end times of r (we suppose that both $\alpha(r)$ and $\beta(r)$ belong to the time horizon H); also, we denote $\delta(r) = \beta(r) - \alpha(r) + 1$ the number of days of the request;
- $t(r)$ the car type of the request, and $T(r)$ the set of car types that can satisfy r (i.e., $T(r) = \{t(r)\} \cup \{t \in \mathcal{T} \text{ such that } t \gg t(r)\}$);
- $n(r)$ the number of required cars.

When a car c is used to satisfy a request r, we say that c *covers* r. For a car type $t \in \mathcal{T}$, we denote $R(t)$ the set of requests which can be covered with cars of type t (i.e., $R(t) = \{r \in \mathcal{R} \text{ such that } t \in T(r)\}$), and for a day $d \in H$, we denote $R_d(t)$ the set of requests $r \in R(t)$ with $d \in [\alpha(r), \beta(r)]$. Finally, the type of a car c is denoted t_c.

Four types of decisions must be taken to build a solution s, and these are represented by the four types of variables defined below:

- for a request r, we denote $\sigma_r(s)$ the number of cars that are subcontracted to another provider;
- for a car type t and a day $d \in H$, we denote $B_{t,d}(s)$ the set of cars of type t bought by the company at day d. Also, $C_{t,d}(s) = C(t) \cup (\bigcup_{d' \leq d} B_{t,d}(s))$ denotes the set of cars of type t available in stock at day d;
- for a request r and a car type t, we denote $A_{r,t}(s)$ the set of cars of type t that are assigned from the stock to request r (we suppose $A_{r,t}(s) \subseteq C_{t,\alpha(r)}(s)$). Also, for a car c, we denote $O_c(s)$ the ordered set of requests covered by c in s;
- for a car c, we denote $M_c(s)$ the ordered set of days at which a maintenance is started on c, and $D_c(s)$ the set of days at which a maintenance occurs on c. Hence, $d \in D_c(s)$ if and only if there exists a day $d' \in M_c(s)$ with $d \in [d', d' + m(t_c) - 1]$.

The following constraints must be satisfied:

(C1) the cars taken from the stock to satisfy a customer's request are of the desired type or upgrades:

$$\text{if } |A_{r,t}(s)| > 0, \quad \text{then } t \in T(r);$$

(C2) each customer receives the desired number of cars:

$$\sigma_r(s) + \sum_{t \in T(r)} |A_{r,t}(s)| = n(r) \quad \text{for all } r \in \mathcal{R};$$

(C3) at most $b(t)$ cars of type t are purchased during the time horizon:

$$\sum_{d \in H} |B_{t,d}(s)| \leq b(t) \quad \text{for all } t \in \mathcal{T};$$

(C4) a car cannot be assigned to two requests that intersect in time:

$$\text{if } A_{r,t}(s) \cap A_{r',t}(s) \neq \emptyset \quad \text{for } r \neq r', \quad \text{then } [\alpha(r), \beta(r)] \cap [\alpha(r'), \beta(r')] = \emptyset;$$

(C5) a car cannot be assigned to a request and maintained at the same time:

$$\text{if } c \in A_{r,t}(s), \quad \text{then } [\alpha(r), \beta(r)] \cap [d, d + m(t) - 1] = \emptyset \quad \text{for all } d \in M_c(s);$$

(C6) every car c has a maintenance after at most $u(t_c)$ days of effective use. Such a constraint cannot be satisfied when a car c covers a request r with duration $\delta(r) > u(t_c)$. In such a case, we only impose that a maintenance occurs between the end of r and the beginning of the next request covered by c (if any). Formally, let c be a car, let d and d' be two consecutive elements in $M_c(s)$, and let R' be the subset of requests r covered by c such that $[\alpha(r), \beta(r)] \subseteq [d + m(t_c), d'[$:

$$\sum_{r \in R'} \delta(r) \leq \max\left\{u(t_c), \max_{r \in R'} \delta(r)\right\};$$

(C7) the capacity of the maintenance workshop is never exceeded:

$$\sum_{c : d \in D_c(s)} w(t_c) \leq W \quad \text{for all } d \in H.$$

There are five different costs associated with a solution s. The first one is the cost of assigning cars from the stock:

$$f_1(s) = \sum_{r \in \mathcal{R}} \sum_{t \in T(r)} |A_{r,t}(s)|(F_a(t) + \delta(r)k_a(t)).$$

The second one is the cost that the company has to pay for each car in stock (rented or not):

$$f_2(s) = \sum_{t \in \mathcal{T}} k_d(t) \left(|H||C(t)| + \sum_{d \in H} (End - d + 1)|B_{t,d}(s)| \right).$$

The third one is the subcontracting cost:

$$f_3(s) = \sum_{r \in \mathcal{R}} k_s(t(r))\delta(r)\sigma_r(s).$$

The fourth one is the purchase cost:

$$f_4(s) = \sum_{t \in \mathcal{T}} \sum_{d \in H} k_b(t)|B_{t,d}(s)|.$$

The last one is the maintenance cost:

$$f_5(s) = \sum_{t \in \mathcal{T}} \sum_{c \in C_{t,End}(s)} k_m(t)|M_c(s)|.$$

The cost $f(s)$ to be minimized is the sum of these five functions (i.e., $f(s) = \sum_{i=1}^{5} f_i(s)$). As shown in ROADEF (1999), the problem of finding a solution that satisfies all the above constraints with a minimum cost is NP-hard in the strong sense. We call this problem the *Car Fleet Management Problem*, or CFMP for short.

3 Overview

The proposed solution technique for the CFMP, which we call ACM (for Algorithm for Car fleet management problem with Maintenance constraints), is rather complex. Before getting into details, we give in this section an overview of the method and its ingredients.

The general scheme of ACM is given in Fig. 1. It contains three main steps. An initial solution is generated at Step 1, using one of the proposed initialization procedures. The two next steps are applied repeatedly until a time limit is reached. Step 2 tries to improve the current solution without changing the set of purchased cars, while Step 3 generates a new solution with a different set of purchased cars. The stopping criteria of ACM is a time limit which we fix to one hour, as imposed by the organizers of the challenge.

The three steps of ACM use subroutines which we have developed for the solution of various subproblems of the CFMP. Figure 2 illustrates how these subroutines are used in ACM. The central rectangle with bold lines represents ACM while the three steps are shown with dashed lines. The other small rectangles represent subroutines. An arc from a box A to a box B means that A is used as a subroutine in B. We give here below a short non formal description of each subroutine. More details will be provided in the next sections.

- (IP_1) is an integer linear program that models the CFMP without maintenance constraints.
- (IP_2) is an integer linear program for finding an optimal maintenance strategy, assuming that the assignment of cars to the requests is fixed.
- Assume that the assignment of cars to the requests is fixed, and suppose that the maintenance schedules are known for a set V of cars. MAINTENANCE is a procedure that determines, if possible, a feasible maintenance schedule of minimum cost for a car $c \notin V$, without modifying the maintenance schedules of the cars in V.
- Let Q be a subset of requests that can be satisfied with cars of type $t \in \mathcal{T}$. If the maintenance constraints are relaxed, then COLOR is a procedure that determines the minimum number of cars of type t needed to cover all requests in Q.
- Let Q be a subset of requests that can be satisfied with ℓ^* cars of type $t \in \mathcal{T}$, and assume that only $\ell < \ell^*$ cars of this type are available in stock, and that no upgrade and no purchase is allowed. If the maintenance constraints are relaxed, then SUBCONTRACT is a heuristic procedure that aims to determine a subset $Q' \subseteq Q$ of minimum total subcontracting cost such that ℓ cars are sufficient to

ACM: An algorithm for the CFMP
Repeat the following until a time limit is reached

1. Generate an initial solution s and set $s^* \leftarrow s$;
2. Try to improve s without changing the set of purchased cars and update s^*;
3. Determine a solution s with a different set of purchased cars and go to 2.

Fig. 1 General scheme of the proposed solution technique

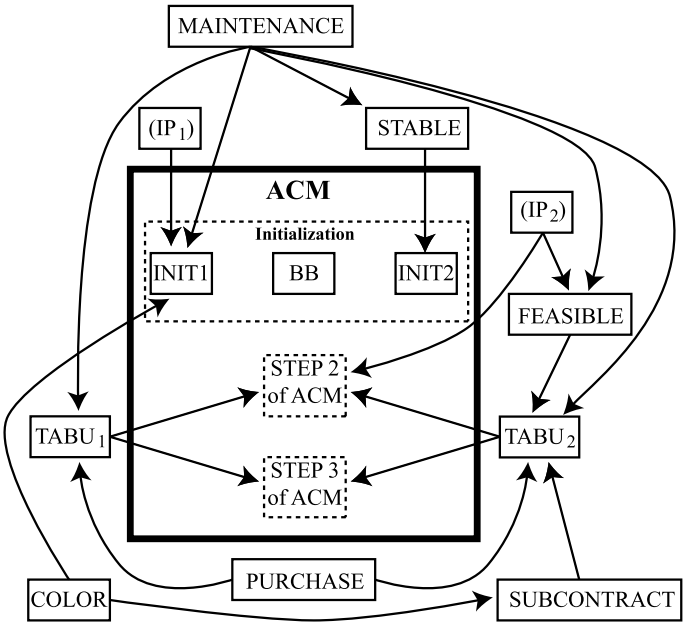

Fig. 2 Overview of the proposed solution method

cover all requests in $Q - Q'$. If $\ell^* = \ell + 1$ then the procedure always finds the optimal solution.

- Consider a car c and a subset Q of requests which can be covered by c. Assume that the maintenance schedules are known for a subset V of cars with $c \notin V$. STABLE is a heuristic procedure that aims to determine a subset $Q' \subseteq Q$ of maximum total duration so that Q' can be covered by c and a feasible maintenance schedule can be found for c without changing those of the cars in V.
- INIT_1 is an initialization procedure used at Step 1 of ACM. It first builds a solution for the CFMP without maintenance constraints. The maintenance schedules are then built by considering the cars c by non increasing order of the total duration of the requests covered by c. Some requests are possibly subcontracted when no feasible maintenance schedule can be found for a car.
- INIT_2 is another initialization procedure used at Step 1 of ACM. It first orders the cars so that cars with a high subcontracting cost and a low assignment cost appear first in the order. Requests are then assigned to the cars according to this order, using the STABLE procedure.
- BB is the algorithm proposed by Briant and Bouzgarrou (ROADEF 1999) for the CFMP.
- PURCHASE is a procedure that determines the optimal purchase dates, assuming that the number of purchased cars of each type is known.
- Let V be a set of cars that covers a set Q of requests, and for which a maintenance schedule is known. Let $Q' \subseteq \mathcal{R} - Q$ be a set of requests to be covered by a car $c \notin V$, and assume that no feasible maintenance schedule can be found for c without exceeding the capacity W of the maintenance workshop (constraint (C7)).

FEASIBLE is a procedure that determines a feasible maintenance schedule for c by modifying the maintenance schedules and possibly subcontracting requests for some cars in V.

- TABU_1 is a tabu search algorithm (Glover and Laguna 1997), where a neighbor s' of a solution s is obtained by assigning a car c to a subcontracted request r. To make such a change feasible, requests covered by c that overlap with r are subcontracted, and the maintenance schedule for c is possibly modified.
- TABU_2 is a tabu search algorithm that tries to reduce the total cost not only by assigning cars to subcontracted requests, but also by avoiding upgrades. A neighbor s' of a solution s is obtained by assigning a car of type t to a request r, where r is subcontracted or covered by a car of type $t' \neq t$ in s. To make such a change feasible, we modify the sets of requests covered by the cars of type t (some requests being moved from one car to another, or possibly subcontracted in s'), and also possibly modify the maintenance schedules of these cars.

4 Subproblems

In this section, we study various subproblems of the CFMP and show how they can be formulated with integer linear programs and graph models.

4.1 Integer linear programs

The CFMP without maintenance constraints (i.e., without constraints (C5), (C6) and (C7)) can be formulated as an integer linear program as follows. In addition to variables $\sigma_r(s)$ that indicate the number of cars of type $t(r)$ that are subcontracted to another provider (see previous section), consider the following variables:

- For each pair (r, t) with $r \in R(t)$, we define $a_{r,t}(s)$ as the number of cars of type t that are assigned from the stock to satisfy request r (i.e., $a_{r,t}(s) = |A_{r,t}(s)|$). We do not consider the variables $a_{r,t}(s)$ with $r \notin R(t)$.
- For each car type t and each day d, we define $b_{t,d}(s)$ as the number of cars of type t bought by the company at day d (i.e., $b_{t,d}(s) = |B_{t,d}(s)|$).

The objective of the CFMP without maintenance constraints is to minimize $\sum_{i=1}^{4} f_i(s)$. Notice that the cost $\sum_{t \in \mathcal{T}} k_d(t)|H||C(t)|$ in $f_2(s)$ is due to cars in stock at the beginning of the time horizon, and can therefore not be avoided. By removing this constant cost from the objective function, and by replacing $|A_{r,t}|$ with $a_{r,t}(s)$ and $|B_{t,d}(s)|$ with $b_{t,d}(s)$, one gets the integer linear program (IP_1) of Fig. 3 for the CFMP without maintenance constraints.

Constraint (C1) (see previous section) is implicitly taken into account by the fact that the variables $a_{r,t}(s)$ are not defined for $r \notin R(t)$. Constraint (1) of the above integer linear program is equivalent to (C2), and constraint (2) is equivalent to (C3). Constraint (3) imposes that there are enough cars in stock every day to satisfy the requests.

The purchase strategy associated with the solution of (IP_1) is to buy a set $B_{t,d}(s)$ of $b_{t,d}(s)$ new cars of type t at each day $d \in H$. An assignment of cars to the requests

$$
(IP_1)\quad
\begin{cases}
\min \sum_{t\in\mathcal{T}} \sum_{r\in R(t)} (F_a(t) + \delta(r)k_a(t))a_{r,t}(s) \\
\qquad + \sum_{t\in\mathcal{T}} \sum_{d\in H} (k_b(t) + k_d(t))(End - d + 1)b_{t,d}(s) \\
\qquad + \sum_{r\in\mathcal{R}} k_s(t(r))\delta(r)\sigma_r(s) \\
\text{subject to} \\
\qquad \sigma_r(s) + \sum_{t\in T(r)} a_{r,t}(s) = n(r) \qquad \forall r \in \mathcal{R}, \qquad (1) \\
\qquad \sum_{d\in H} b_{t,d}(s) \le b(t) \qquad \forall t \in \mathcal{T}, \qquad (2) \\
\sum_{r\in R_d(t)} a_{r,t}(s) - \sum_{d'\le d} b_{t,d'}(s) \le |C(t)| \quad \forall t \in \mathcal{T},\ \forall d \in H \qquad (3) \\
\quad \text{all } \sigma_r(s),\ a_{r,t}(s) \text{ and } b_{t,d}(s) \text{ are integers.}
\end{cases}
$$

Fig. 3 Integer linear program for the CFMP without maintenance constraints

satisfying constraint (C4) (i.e., the sets $A_{r,t}(s)$) can be obtained by applying the sequential coloring algorithm described in the next section. So consider now the sets $A_{r,t}(s)$ as fixed. We show how to determine the best maintenance strategy by solving an integer linear program (IP_2). We first need to define some notation.

- We denote $\mathcal{C}$ the set of cars in stock at day *End*. Hence, $\mathcal{C} = \bigcup_{t\in\mathcal{T}} C_{t,End}$.
- We denote $\Delta(c)$ the set of days d at which a maintenance on c can start. Hence, according to constraint (C5), $d \in \Delta(c)$ if and only if there is no request covered by c during the time interval $[d, d + m(t_c) - 1]$. For each car c and each day $d \in \Delta(c)$, we define $x_{d,c} = 1$ if a maintenance on car c starts at day d, and $x_{d,c} = 0$ otherwise (i.e., $x_{d,c} = 1$ if and only if $d \in M_c(s)$).
- Let $O_c = \{r_1, \ldots, r_q\}$ denote the ordered set of requests covered by c, and let P_c denote the set of ordered pairs (i, j) with $j > i$ and such that the total duration of the requests $r_i, \ldots, r_j$ is strictly larger than $u(t_c)$ while the total duration of the requests $r_i, \ldots, r_{j-1}$ is smaller than or equal to $u(t_c)$. In case a request r_i with $i < q$ has a duration $\delta(r_i) > u(t_c)$, then we also include the pair $(i, i + 1)$ in P_c. For every pair $(i, j) \in P_c$, constraint (C6) imposes a maintenance on c between the end of r_i and the beginning of r_j. We therefore define $\Delta_{(i,j)}(c) = [\beta(r_i) + 1, \alpha(r_j) - m(t_c)] \cap \Delta(c)$ and impose the start of at least one maintenance at a day $d \in \Delta_{(i,j)}(c)$.
- For a day d and a car c, we define $\underline{d}_c = \max\{Start, d - m(t_c) + 1\}$. A maintenance can then occur at day d only if it was started during the time interval $[\underline{d}_c, d]$.

Assuming that the sets $A_{r,t}(s)$ are fixed, the integer linear program (IP_2) of Fig. 4 has a feasible solution if and only if there is a maintenance strategy that satisfies constraints (C5), (C6) and (C7). Indeed, constraint (C5) is implicitly taken into account by the fact that the variables $x_{d,c}$ are not defined for $d \notin \Delta(c)$, and constraints (4)

$$(IP_2)\quad \begin{cases} \min \sum_{c\in\mathcal{C}} \sum_{d\in\Delta(c)} k_m(t_c) x_{d,c} \\ \\ \text{subject to} \\ \qquad \sum_{d\in\Delta_{(i,j)}(c)} x_{d,c} \geq 1 \qquad \forall c \in \mathcal{C},\ \forall (i,j) \in P_c, \qquad (4) \\ \\ \sum_{c\in\mathcal{C}} \sum_{d'=\underline{d}_c}^{d} w(t_c) x_{d',c} \leq W \qquad \forall d \in H. \qquad (5) \end{cases}$$

Fig. 4 Integer linear program for the maintenance strategy

Algorithm MAINTENANCE(V, $O_c = \{r_1, \ldots, r_q\}$)

1. Set $i \leftarrow 1$ and $M_c \leftarrow \emptyset$;
2. If $\delta(r_i) > u(t_c)$ then set $p \leftarrow i+1$; else determine the smallest integer p such that $p = q+1$ or $\sum_{\ell=i}^{p} \delta(r_\ell) > u(t_c)$;
3. If $p \leq q$ then set $j \leftarrow p-1$ and go to 4; else STOP: return M_c;
4. If $\alpha(r_{j+1}) - \beta(r_j) \leq m(t_c)$ then go to 6; else determine the day $d \in [\beta(r_j)+1, \alpha(r_{j+1}) - m(t_c)]$ which minimizes $w_d = \max_{d'\in[d,d+m(t_c)-1]} W_{d'}(V)$;
5. If $w_d + w(t_c) > W$ then go to 6; else add d to M_c, set $i \leftarrow j+1$, and go to 2;
6. Set $j \leftarrow j-1$; if $j \geq i$ then go to 4; else STOP: return (i, p).

Fig. 5 Finding the maintenance schedule for a single car c

and (5) of (IP_2) correspond to constraints (C6) and (C7), respectively. Moreover, if a feasible solution exists, then the optimal solution to (IP_2) is an optimal maintenance strategy (with $A_{r,t}(s)$ fixed) since the objective of (IP_2) is to minimize the maintenance costs.

Finding an optimal maintenance schedule for a single vehicle is a much simpler problem. More precisely, assume again that all sets $A_{r,t}(s)$ are fixed and suppose that the maintenance schedules are known for a set V of cars. Let c be a car not in V and let $O_c = \{r_1, \ldots, r_q\}$ be the ordered set of requests covered by c. We consider the problem of determining, if possible, a feasible maintenance schedule of minimum cost for c without modifying the maintenance schedules of the cars in V. Let $W_d(V)$ denote the total number of workers which are already busy in maintaining cars of V at day d. The algorithm MAINTENANCE in Fig. 5 produces two possible outputs: if there exists a feasible maintenance schedule for c, then the algorithm returns an optimal one M_c; otherwise, it returns a pair (i, p) with the meaning that no maintenance can be scheduled between the end of r_i and the beginning of r_p, while there should be one.

Step 2 of MAINTENANCE takes care of the fact that a request r_i possibly has a duration $\delta(r_i) > u(t_c)$, in which case constraint (C6) imposes a maintenance between the end of r_i and the beginning of r_{i+1}.

The choice made at Step 2 guarantees that as many requests as possible are satisfied before scheduling a maintenance for c. Hence, the obtained schedule (if any) has a minimum maintenance cost. Notice that, at Step 4, we could have chosen any day d with $w_d + w(t_c) \leq W$. By choosing d so that the maximum value of $W_{d'}(V)$ over all $d' \in [d, d + m(t_c) - 1]$ is minimized, we try to balance the load of the workshop during the time horizon.

Notice also that if V is empty (i.e., no maintenance has been scheduled yet), and if the output of MAINTENANCE($\emptyset$, O_c) is a pair (i, p), then constraint (C6) cannot be satisfied for c, and this does not depend on the number W of workers in the workshop. We say that the set O_c of requests covered by c is (C6)-*feasible* if and only if MAINTENANCE($\emptyset$, O_c) produces a feasible maintenance schedule.

4.2 Graph models

In what follows, we find it convenient to replace every request r for n_r cars by n_r requests for a car. This does not change the problem since there is no constraint linking the different cars to be assigned to a request. As a consequence, variables $\sigma_r(s)$ are equal to one or zero, and we replace them by a set $\mathcal{S}(s)$ containing all requests subcontracted to another provider.

4.2.1 A graph coloring problem

Let Q be a set of requests that we would like to satisfy with cars of type $t \in \mathcal{T}$, and assume that we want to use as few cars as possible. If the maintenance constraints are relaxed, then this is a well-known interval scheduling problem (see Kolen et al. 2006 for a survey on this subject) that can be solved using a graph coloring algorithm. More precisely, consider the graph G_Q with vertex set Q, and where two vertices r and r' are linked by an edge if and only if the intervals $[\alpha(r), \beta(r)]$ and $[\alpha(r'), \beta(r')]$ intersect. The considered problem then reduces to finding a coloring of the vertices of G_Q that uses as few colors as possible, and such that no two vertices linked by an edge have the same color. Each color corresponds to a car.

The graph coloring problem is NP-complete for general graphs (Garey and Johnson 1979). However, the graph defined above belongs to the well-known class of interval graphs (Golumbic 1980) and polynomial algorithms have been designed to color the vertices of such graphs with a minimum number of colors. For example, Gupta et al. (1979) and Kolen and Lenstra (1995) have described an $O(n \log n)$ algorithm (where n stands for the number of vertices) which is summarized in Fig. 6, and where each color is represented by a strictly positive integer.

4.2.2 A shortest path problem

The next subproblem considers again a set Q of requests to be satisfied with cars of a given type $t \in \mathcal{T}$, but we assume here that we have only ℓ cars of this type in

Algorithm COLOR(G_Q)

Repeat
  Choose a non colored vertex r in G_Q with minimum value $\alpha(r)$ (break ties at random);
  Color r with the smallest color not yet used on a vertex adjacent to r;
Until all vertices are colored.

Fig. 6 A coloring algorithm for interval graphs

stock, and that no upgrade and no purchase is allowed. If needed, some requests may be subcontracted to another provider, but this should be done at minimum cost. The maintenance constraints are relaxed here again.

Let ω_d denote the number of requests $r \in Q$ with $d \in [\alpha(r), \beta(r)]$. It is well-known that the size of a maximum clique (i.e., the maximum number of vertices which are pairwise adjacent) in an interval graph is equal to the minimum number of colors needed to color its vertices (Golumbic 1980). This property applied to G_Q means that the maximum value ω_d over H is equal to the minimum number ℓ^* of cars needed to satisfy all requests in Q (which can be determined with the above COLOR algorithm). If ℓ cars are sufficient to cover all requests in Q (i.e. $\ell^* \leq \ell$), then no subcontracting is needed. Otherwise, we define an oriented graph $\vec{G}(Q, \ell^*)$ as follows:

- create a vertex r for every $r \in Q$ such that there is a day $d \in [\alpha(r), \beta(r)]$ with $\omega_d = \ell^*$, and add two vertices u and v corresponding to two artificial requests with $\alpha(u) = \beta(u) = Start - 1$ and $\alpha(v) = \beta(v) = End + 1$;
- create an arc from a vertex r to a vertex r' if and only if $\alpha(r) < \alpha(r')$ and $[\beta(r) + 1, \alpha(r') - 1]$ contains no day d with $\omega_d = \ell^*$ (notice that there is no arc leaving the artificial vertex v);
- define $(k_s(t(r)) - k_a(t))\delta(r) - F_a(t)$ as the length of all arcs originating from a vertex $r \neq u$ (this value corresponds to the cost of subcontracting r instead of covering it with a car of type t); the arcs originating from the artificial vertex u have a length 0.

It then follows from the definition of $\vec{G}(Q, \ell^*)$ that the optimal subcontracting strategy for using only $\ell^* - 1$ cars can be determined by finding a shortest path P from u to v in $\vec{G}(Q, \ell^*)$. The internal vertices of P are the requests to be subcontracted, and the length of P is the cost of the corresponding subcontracting strategy. Since $\vec{G}(Q, \ell^*)$ has no circuit, a shortest path can be determined using a linear algorithm (see for example Ahuja et al. 1993).

By construction, the remaining requests in Q (i.e., those which are not subcontracted) can be covered with $\ell^* - 1$ cars, and one can repeat this process until only ℓ cars are needed. This procedure, called SUBCONTRACT, is summarized in Fig. 7. It produces two outputs: a set Q' of requests to be subcontracted, and an assignment of the remaining requests to the ℓ available cars.

Algorithm SUBCONTRACT(Q, ℓ)

1. Set $Q' \leftarrow \emptyset$ and $\ell^* \leftarrow \max_{d \in H} \omega_d$; if $\ell^* \leq \ell$ then go to 4;
2. Determine a shortest path P from u to v in $\vec{G}(Q, \ell^*)$, add the internal vertices of P to Q', and remove them from Q;
3. If $\ell^* = \ell + 1$ then go to 4; else set $\ell^* \leftarrow \ell^* - 1$ and go to 2;
4. Determine an optimal coloring o of G_Q by applying COLOR(G_Q) and STOP: Q' is the set of requests to be subcontracted and o corresponds to an assignment of the remaining requests to the ℓ available cars.

Fig. 7 Subcontracting strategy

4.2.3 A maximum weighted stable set problem

Consider a car c and a subset $Q \subseteq R(t_c)$ of requests which can be covered by c. The last subproblem studied in this subsection is to assign requests from Q to c so that the total duration of the assigned requests is maximum. If we relax the maintenance constraints, then an optimal solution can be determined in polynomial time. Indeed, the problem is equivalent to finding a stable set (i.e., a set of pairwise non adjacent vertices) of maximum weight in G_Q, where the weight of a vertex is the duration of its corresponding request. Finding a stable set of maximum weight in a graph is equivalent to finding a clique of maximum weight in the complementary graph. Since the complementary graph of G_Q is a comparability graph (Golumbic 1980), and since linear algorithms are known for finding cliques of maximum weight in comparability graphs (see for example Shamir 1994), this problem can be solved in linear time.

The problem is much more difficult if the maintenance constraints have to be satisfied. More precisely, assume that the maintenance schedules are known for a subset V of cars with $c \notin V$, and suppose that we want to assign requests from Q to c so that the total duration of the assigned requests is maximum, and a feasible maintenance schedule can be found for c without changing those of the cars in V. Since no polynomial algorithm is known for this problem, we propose a heuristic procedure, described in Fig. 8, that produces a subset Q' of requests to be covered by c. It uses the notation Q_r for the subset of requests in Q that overlap with r (i.e., $r' \in Q_r$ if and only if $r \in Q$ and $[\alpha(r), \beta(r)] \cap [\alpha(r'), \beta(r')] \neq \emptyset$).

The choice made at Step 4 is to favor requests with a long duration and which overlap with requests having a small total duration (since these will be removed from Q).

5 An algorithm for the CFMP

In this section, we describe the proposed algorithm, called ACM, for the solution of the CFMP. It uses the subroutines of the previous sections (see Fig. 2 of Sect. 3) as well as other procedures described below. The general scheme of ACM is shown in Fig. 1 of Sect. 3. In the next subsections, we give more details for each step of this general scheme.

Algorithm STABLE(V, Q)

1. Set $Q' \leftarrow \emptyset$;
2. For all $r \in Q$ do
   If MAINTENANCE(V, $Q' \cup \{r\}$) does not produce a feasible schedule then remove r from Q;
3. If $Q = \emptyset$ then STOP: Q' is the set of requests to be covered by c;
4. Choose $r \in Q$ with maximum value $\frac{\delta(r)}{\sum_{r' \in Q_r} \delta(r')}$;
5. Set $Q' \leftarrow Q' \cup \{r\}$, remove $\{r\} \cup Q_r$ from Q, and go to 2.

Fig. 8 Algorithm for a stable set of maximum weight

5.1 Initial solution

We propose two different procedures for generating an initial solution. Procedure INIT_1 first generates a solution s that satisfies constraints (C1) to (C4) by solving (IP_1) and applying procedure COLOR (see Sect. 4.2.1). The set of purchased cars is then considered as fixed and we denote O_c the ordered set of requests covered by car c. The maintenance schedules are then built by considering the cars c by non increasing order of the total duration of the requests in O_c. For each car c, we apply procedure MAINTENANCE(V, O_c) of Sect. 4.1, where V is the set of cars preceding c in the above order (i.e., the set of cars for which a maintenance schedule is already known). If MAINTENANCE(V, O_c) returns a pair (i, p), we remove from $O_c = \{r_1, \ldots, r_q\}$ the request with smallest duration among $\{r_i, \ldots, r_{p-1}\}$ and repeat this process until a feasible maintenance schedule can be found for c. The procedure is described in Fig. 9.

The second initialization procedure builds a solution with no upgrade and no purchase. It considers the cars c in stock by non increasing order of $k_s(t_c) - k_a(t_c)$. Hence, we first consider cars with a high subcontracting cost and a low assignment cost. A schedule for each car is built using procedure STABLE (see Sect. 4.2.3), as described in Fig. 10.

5.2 Improvement procedures

We now describe several improvement procedures used at Step 2 of ACM. Two of them are tabu search algorithms that follow the general scheme of Fig. 11, where $N(s)$ denotes the *neighborhood* of a solution s, which is defined as the set of *neighbor solutions* obtained from s by performing a local change, called *move*, and *TL* is a list of forbidden moves. A more detailed description of tabu search and its concepts can be found in Glover and Laguna (1997).

The two tabu search algorithms we have developed share some common features. For example, their tabu list *TL* contains requests with the meaning that it is forbidden for several iterations to remove them from the cars covering them. Also, a neighbor solution is obtained for both algorithms by modifying the set $\mathcal{S}(s)$ of subcontracted requests, the assignments $A_{r,t}(s)$ of requests to cars, and the maintenance schedules

Algorithm INIT_1

1. Solve (IP_1); buy a set $B_{t,d}(s)$ of $b_{t,d}(s)$ new cars of type t at each day $d \in H$ (where the values $b_{t,d}(s)$ come from the solution of (IP_1));
   For all $t \in \mathcal{T}$ do
   (a) let Q_t denote the set of requests to be covered by cars of type t according to the output of (IP_1);
   (b) apply $\text{COLOR}(G_{Q_t})$ to determine an assignment of cars to the requests in Q_t;
   Subcontract all non covered requests and let s denote the resulting solution satisfying constraints (C1) to (C4); set $V \leftarrow \emptyset$ and let O_c denote the ordered set of requests covered by a car c in s;
2. Sort the cars by non increasing order of the total duration of the requests in O_c;
3. Consider each car c according to the above order and build a feasible maintenance schedule for c as follows:
   (a) let $O_c = \{r_1, \ldots, r_q\}$; if the output of $\text{MAINTENANCE}(V, O_c)$ is a pair (i, p) then choose r_j in $\{r_i, \ldots, r_{p-1}\}$ with minimum value $\delta(r_j)$, transfer r_j from O_c to the set $\mathcal{S}(s)$ of subcontracted requests, and repeat 3a; else fix the obtained maintenance schedule for c in s, add c to V, and go to 3b;
   (b) if c is the last car in the ordered list then STOP; else go to Step 3a and consider the next car.

Fig. 9 First generator of an initial solution

Algorithm INIT_2

1. For all $t \in \mathcal{T}$ do set $Q_t \leftarrow \{r \text{ with } t(r) = t\}$;
   Sort the cars $c \in \bigcup_{t \in \mathcal{T}} C(t)$ in non increasing order of $k_s(t_c) - k_a(t_c)$;
2. Consider the cars c according to the above order and build a feasible schedule for each of them as follows;
   (a) apply procedure $\text{STABLE}(V, Q_{t_c})$ to determine the set O_c of requests assigned to c;
   (b) remove O_c from Q_{t_c};
3. Set $\mathcal{S}(s) \leftarrow \bigcup_{t \in \mathcal{T}} Q_t$ (i.e, the set of requests not covered by any car).

Fig. 10 Second generator of an initial solution

$M_c(s)$. However, all neighbor solutions have the same set $\mathcal{C}$ of purchased cars, and we denote by $\mathcal{C} = \bigcup_{t \in \mathcal{T}} C_{t,End}(s)$ the set of cars in stock at day *End*. While $\mathcal{C}$ does not change, the cars can be bought as late as possible to reduce the storage costs. More precisely, let $p(t) = |C_{t,End}(s)| - |C(t)|$ denote the number of purchased cars of type t, let $O_c(s)$ denote the set of requests covered by c in s, and let $e_c(s)$ denote the earliest start time $\alpha(r)$ of a request $r \in O_c(s)$. The optimal purchase dates for s are determined with the very straightforward PURCHASE procedure described in Fig. 12.

Tabu Search

Choose an initial solution s; set $TL \leftarrow \emptyset$ (tabu list); set $s_{best} \leftarrow s$ (best solution);
Repeat the following until a stopping criterion is met

1. Determine a best solution $s' \in N(s)$ such that either s' is obtained from s by performing a move $m \notin TL$ or s' is better than s_{best};
2. If s' is better than s_{best} then set $s_{best} \leftarrow s'$;
3. Set $s \leftarrow s'$ and update TL.

Fig. 11 General scheme of a tabu search

Procedure PURCHASE(s)

For all car types $t \in \mathcal{T}$ do
(a) Set $B_{t,d} = \emptyset$ for all days $d \in H$;
(b) Determine the set E of $p(t)$ cars $c \in C_{t,End}(s)$ with largest value $e_c(s)$ (ties are broken randomly);
(c) Set $B_{t,e_c(s)} \leftarrow B_{t,e_c(s)} \cup \{c\}$ for every car $c \in E$.

Fig. 12 Optimization of the purchase dates

Procedure Neighborhood_1(t, s)

1. Set $N_t^1(s) \leftarrow \emptyset$;
2. For all requests $r \in \mathcal{S}(s)$ with $t(r) = t$ and all cars $c \in C_{t,End}(s)$ do
   (a) Set $Q = \{r' \in O_c(s)$ with $[\alpha(r), \beta(r)] \cap [\alpha(r'), \beta(r')] \neq \emptyset\}$;
   (b) Set $O_c(s') \leftarrow (O_c(s) \cup \{r\}) - Q$ and $\mathcal{S}(s') \leftarrow (\mathcal{S}(s) - \{r\}) \cup Q$;
   (c) If MAINTENANCE($\mathcal{C} - \{c\}$, $O_c(s')$) produces a feasible maintenance schedule, then fix it for c, apply PURCHASE(s'), and add the so obtained new solution to $N_1^t(s)$.

Fig. 13 First neighborhood

Our first tabu search algorithm, called TABU$_1(t)$, focuses on a given type $t \in \mathcal{T}$ of cars. In order to build a neighbor solution s' from s, we first choose a request $r \in \mathcal{S}(s)$ with $t(r) = t$ and a car c in $C_{t,End}(s)$, and cover r with c. We then subcontract all requests r' currently covered by c such that $[\alpha(r), \beta(r)] \cap [\alpha(r'), \beta(r')] \neq \emptyset$. The next step consists in rescheduling the maintenances on c by applying the MAINTENANCE procedure. If no feasible maintenance schedule can be found, then the neighbor s' is not considered. The set of so obtained neighbors of s is denoted $N_1^t(s)$, and its generation is summarized in Fig. 13.

When a move is performed from the current solution s to a neighbor $s' \in N_t^1(s)$, we introduce in TL the request r that was in $\mathcal{S}(s)$, and is now in $O_c(s')$.

The second tabu search algorithm, called TABU_2, performs much more complex moves. It tries to reduce the total cost not only by assigning subcontracted requests to cars, but also by avoiding upgrades. We first describe a procedure which builds a feasible maintenance schedule for a car c given a (C6)-feasible ordered set $O_c(s)$ of requests covered by c. Such a construction may involve changes in the sets $M_{c'}(s)$ and $O_{c'}(s)$ of other cars c'.

Consider a scheduled maintenance m for a car c, and let r and r' be the two requests surrounding m (i.e., r and r' are consecutive in $O_c(s)$ and m intersects $[\beta(r)+1, \alpha(r')-1]$). We define $r_{(c,m)}(s)$ as the request among r and r' which induces the smallest increase in cost if it is subcontracted instead of being covered by c. More formally, $r_{(c,m)}(s) = r$ if $(k_s(t(r)) - k_a(t))\delta(r) < (k_s(t(r')) - k_a(t))\delta(r')$, else $r_{(c,m)}(s) = r'$. The solution obtained from s by transferring $r_{(c,m)}(s)$ from $O_c(s)$ to the set $\mathcal{S}(s)$ of subcontracted requests is denoted $s \oplus_1 (c, m)$. Recursively, if c contains two requests surrounding m in $s \oplus_\ell (c, m)$, then we define $s \oplus_{\ell+1} (c, m) = (s \oplus_\ell (c, m)) \oplus_1 (c, m)$. We also denote $s \oplus_0 (c, m) = s$.

Assume that a feasible maintenance schedule has already been built for a set $V \subseteq \mathcal{C} - \{c\}$ of cars, let $O_c(s) = \{r_1, \ldots, r_q\}$, and suppose that the output of MAINTENANCE(V, $O_c(s)$) is a pair (i, p) (meaning that no maintenance can be scheduled on c between the end of r_i and the beginning of r_p). Let V_p be the set of cars c' in V with at least one maintenance $m_{c'}$ intersecting the interval $[\beta(r_{p-1})+1, \alpha(r_p)-1]$, and such that MAINTENANCE(V, $O_c(s)$) produces a feasible maintenance schedule for c if $m_{c'}$ is removed from c'. In order to allow the schedule of a maintenance on c between r_{p-1} and r_p, we remove requests around $m_{c'}$ for a car c' in V_p. This may involve modifications of the maintenance schedules of other cars in V_p.

More precisely, we denote $IP_3(s, c, p, V)$ the integer linear program obtained from (IP_2) by considering only the cars in V, by fixing the maintenance schedules for the cars in $V - (V_p \cup \{c\})$ (i.e., $x_{d,c'} = 1$ for all $c' \in V - (V_p \cup \{c\})$ and for all $d \in M_{c'}(s)$), and by replacing $O_c(s)$ with $\{r_1, \ldots, r_p\}$ (hence no maintenance is scheduled for c after r_p). The optimal value of $IP_3(s, c, p, V)$ is denoted $g(s, c, p, V)$.

Procedure FEASIBLE in Fig. 14 solves $IP_3(s \oplus_\ell (c', m_{c'}), c, p, V)$ with increasing values of ℓ, until a maintenance schedule can be found for c up to request r_p. Then MAINTENANCE(V, $O_c(s)$) is again applied with $O_c(s) = \{r_1, \ldots, r_q\}$. If the output is a feasible maintenance schedule for c, then the algorithm stops, else the above process is repeated with the new output (i, p) of MAINTENANCE(V, $O_c(s)$).

Observe that FEASIBLE is a finite procedure. Indeed, notice first that if $s \oplus_\ell (c', m_{c'})$ does not have two requests surrounding $m_{c'}$, then $m_{c'}$ can be removed from c' and a feasible maintenance schedule can then be obtained for c, by definition of V_p. Hence, ℓ cannot be larger than $|O_{c'}(s)|$. Also, the successive values of p in Step 2 are strictly increasing since the schedule found at Step 4 is feasible for c up to r_p.

We can now describe the construction of the second neighborhood $N_2(s)$. For a request r and a car type $t \in T(r)$ with $A_{r,t}(s) = \emptyset$, we build a neighbor solution of s by first removing r from $\mathcal{S}(s)$ if r is subcontracted in s, or from $O_c(s)$ if r is currently covered by c in s. Let Q denote the set of requests covered by cars of type t. We apply SUBCONTRACT($Q \cup \{r\}$, $|C_{t,End}(s)|$) (see Sect. 4.2) to determine

Procedure FEASIBLE(s, V, c)

1. Apply MAINTENANCE($V, O_c(s)$); if the output is a feasible maintenance schedule for c then STOP: add this schedule to c and return the so obtained solution as output;
2. Let (i, p) be the output of MAINTENANCE($V, O_c(s)$); determine the set V_p of cars c' in V having at least one maintenance $m_{c'}$ that intersects the interval $[\beta(r_{p-1}) + 1, \alpha(r_p) - 1]$, and such that MAINTENANCE($V, O_c(s)$) produces a feasible maintenance schedule for c if $m_{c'}$ is removed from c'; set $\ell \leftarrow 0$;
3. If there is no car $c' \in V_p$ such that $IP_3(s \oplus_\ell (c', m_{c'}), c, p, V)$ has a feasible solution, then set $\ell \leftarrow \ell + 1$ and repeat 3;
4. Choose a car, say c^*, which produces a feasible solution with minimum cost $g(s \oplus_\ell (c^*, m_{c^*}), c, p, V)$, set $s \leftarrow s \oplus_\ell (c^*, m_{c^*})$ and go to 1.

Fig. 14 Finding a feasible maintenance schedule for a car c

a subset of requests which can be covered with the $|C_{t,End}(s)|$ cars of type t in stock, without considering the maintenance constraints. We denote s' the resulting solution.

If a car c' with $O_{c'}(s') = \{r_1, \ldots, r_q\}$ is not (C6)-feasible in s', then let (i, p) be the output of MAINTENANCE($\emptyset, O_{c'}(s')$). We artificially increase the duration of r_{p-1} and r_p so that these two requests overlap and must therefore be covered by different cars. A new assignment of the requests covered by the cars of type t is then obtained by applying again the SUBCONTRACT procedure. This is repeated until all cars of type t cover a (C6)-feasible set of requests. We then use procedure FEASIBLE to build a feasible maintenance schedule for all cars.

We finally try to avoid upgrades and subcontracted requests by making simple insertions, even if this induces additional maintenances on several cars. The reason for such final modifications is that the maintenance costs are typically much smaller than the upgrade and subcontracting costs. All this process is described in more details in procedure Neighborhood_2(s) of Fig. 15, where L denotes the set of requests with an artificially modified duration and $\mathcal{C} = \bigcup_{t \in \mathcal{T}} C_{t,End}(s)$ is the set of cars in stock at day *End*.

Let $Q' \subset Q \cup \{r\}$ be the subset of requests that procedure SUBCONTRACT chooses to subcontract at Step 2b. In order to have $r \notin Q'$ (since the basic idea of this second neighborhood is to cover an additional request r with a car of type t) and $Q' \cap TL = \emptyset$ (since tabu requests should not be subcontracted), the shortest path problems are solved in the graph obtained from $\vec{G}(Q \cup \{r\}, \ell^*)$ by removing the vertices in $\{r\} \cup (TL \cap Q)$.

Also, as explained in Sect. 4.2, $\vec{G}(Q \cup \{r\}, \ell^*)$ contains an arc from a vertex r' to a vertex r'' if and only if $\alpha(r') < \alpha(r'')$ and $[\beta(r') + 1, \alpha(r'') - 1]$ contains no day where ℓ^* requests of $Q \cup \{r\}$ overlap. We use the modified durations of r' and/or r'' if they belong to the list L. However, we compute the cost $(k_s(t(r') - k_a(t))\delta(r') - F_a(t)$ of an arc leaving r' by using the original durations (since we want to know the real cost impact of subcontracting r').

Procedure Neighborhood_2(s)

1. Set $N_2(s) \leftarrow \emptyset$ and $L \leftarrow \emptyset$;
2. For all $r \in \mathcal{R}$ and $t \in T(r)$ such that $A_{r,t}(s) = \emptyset$ do
   (a) Remove r from $\mathcal{S}(s)$ if r is subcontracted in s, or from $O_c(s)$ if r is covered by a car c in s;
   (b) Let Q be the set of requests covered by cars of type t in s; apply SUBCONTRACT($Q \cup \{r\}$, $|C_{t,End}(s)|$) and let s' denote the resulting solution; if all cars in s' cover a (C6)-feasible set of requests then go to 2d;
   (c) Consider a car $c' \in C_{t,End}(s)$ which covers a (C6)-infeasible set $O_{c'}(s') = \{r_1, \ldots, r_q\}$ of requests, and let (i, p) be the output of MAINTENANCE($\emptyset$, $O_{c'}(s')$);
   - set $x \leftarrow \frac{1}{2}(\alpha(r_p) - \beta(r_{p-1}))$;
   - set $\beta(r_{p-1}) \leftarrow \beta(r_{p-1}) + \lceil x \rceil$ and $\alpha(r_p) \leftarrow \alpha(r_p) - \lfloor x \rfloor$;
   - add r_{p-1} and r_p to L and go to 2b;

   (d) Give back the original durations to all requests $r' \in L$ and let $V = \{c_1, \ldots, c_k\}$ be the set of cars of type t; set $M_{c'}(s') = M_{c'}(s)$ for all cars $c' \in \mathcal{C} - V$;
   (e) For $i = 1$ to k do
   apply FEASIBLE($s', \mathcal{C} - V, c_i$); remove c_i from V and set s' equal to the resulting solution;
   (f) For every subcontracted request r' in s' do
   if there is a car $c' \in C_{t,End}(s)$ with $t \in T(r')$ which does not cover any request overlapping with r', and such that MAINTENANCE($\mathcal{C} - \{c'\}$, $O_{c'}(s') \cup \{r'\}$) produces a feasible maintenance schedule for c', then add r' to $O_{c'}(s')$ and update the maintenance schedule for c' according to the output of MAINTENANCE;
   (g) For every request r' covered in s' by a car c' with $t_{c'} \neq t(r')$ do
   if there is a car $c'' \in C_{t(r'),End}(s)$ which does not cover any request overlapping with r', and such that MAINTENANCE($\mathcal{C} - \{c''\}$, $O_{c''}(s') \cup \{r'\}$) produces a feasible maintenance schedule for c'', then transfer r' from $O_{c'}(s')$ to $O_{c''}(s')$ and update the maintenance schedule for c'' according to the output of MAINTENANCE;
   (h) Apply PURCHASE(s') and add the resulting solution to $N_2(s)$.

Fig. 15 Second neighborhood

To complete this section, we finally describe Step 2 of ACM in Fig. 16. It uses two parameters MAX_1 and MAX_2 that limit the number of iterations performed without improvement of the best solution s_{best} encountered so far. Parameter LIM at Step 4 limits the time spent trying to improve the maintenance schedules. Notice that s^* denotes the best solution encountered by ACM, using any purchase strategy, while s_{best} is the best solution found during one visit of Step 2 of ACM.

Step 2 of ACM

1. Apply TABU_2 until MAX_2 iterations have been performed without improvement of s_{best};
2. For all $t \in \mathcal{T}$ do
   apply $\text{TABU}_1(t)$ with s_{best} as initial solution, until MAX_1 iterations have been performed without improvement of s_{best};
3. If s_{best} was modified during Steps 1 and/or 2 then go to 1;
4. Solve (IP_2) with a time limit of LIM seconds to possibly obtain a better maintenance schedule for s_{best}, and update s_{best} if it has been improved;
5. If s_{best} is better than s^* then set $s^* \leftarrow s_{best}$.

Fig. 16 Step 2 of ACM

5.3 Modification of the purchases

The tabu search procedures described in the previous subsections work with a fixed set of purchased cars. We now explain how to modify that set. Each time we enter Step 3 of ACM, we either buy one new car, or we remove a car from the set of purchased cars.

- To evaluate the cost of buying a new car c of type t, we generate a solution s from s_{best} by adding c to $C_{t,End}(s_{best})$, setting $O_c(s) = \emptyset$, and then applying TABU_2 with parameter MAX_3 instead of MAX_2, and then $\text{TABU}_1(t)$ (with parameter MAX_1). We choose MAX_3 very small when compared to MAX_2 (more details will be given in the next section).
- If $C(t) \subset C_{t,End}(s_{best})$ for a car type t, we evaluate the cost of not buying a car of type t by first choosing $c \in C_{t,End}(s_{best})$ with minimum total duration of the requests covered by c. We then generate a solution s from s_{best} by subcontracting all requests covered by c (i.e., we add $O_c(s_{best})$ to $\mathcal{S}(s_{best})$) and removing c from $C_{t,End}(s_{best})$. We finally apply TABU_2 on the resulting solution with parameter MAX_3 instead of MAX_2, and then $\text{TABU}_1(t)$ (with parameter MAX_1).

Among all these solutions, we choose one with minimum cost and use it as initial solution when entering Step 2 of ACM again. On the one hand, if this best solution is obtained by purchasing a new car of type t, we then forbid for 5 iterations of ACM to remove a car of type t from the set of purchased cars. On the other hand, if the new solution is obtained by removing a car of type t from the set of purchased cars, we then forbid for 5 iterations of ACM to buy a new car of type t. All this process is summarized in Fig. 17.

6 Computational experiments

In this section we report the computational experiments we have made on the set of 16 benchmark instances of the ROADEF'99 international challenge. We compare our results with those obtained by the competitors of the challenge.

Step 3 of ACM

1. Do the following for all $t \in \mathcal{T}$ such that no car of type t was removed from the set of purchased cars during the last 5 iterations of ACM:
   create s from s_{best} by adding a new car c to $C_{t,End}(s_{best})$ and setting $O_c(s) \leftarrow \emptyset$; apply TABU_2, starting from s, with parameter MAX_3 instead of MAX_2; apply $\text{TABU}_1(t)$ starting from the resulting solution (with parameter MAX_1); if the resulting solution s has a cost $f(s) < BestCost$ then set $BestCost \leftarrow f(s)$ and $BestSolution \leftarrow s$;
2. Do the following for all car types t such that $C(t) \subset C_{t,End}(s_{best})$ and no car of type t was bought during the last 5 iterations of *ACM*:
   choose a car c of type t with minimum total duration of the requests covered by c; create s from s_{best} by adding $O_c(s_{best})$ to $\mathcal{S}(s_{best})$ and removing c from $C_{t,End}(s_{best})$; apply TABU_2, starting from s, with parameter MAX_3 instead of MAX_2; apply $\text{TABU}_1(t)$ starting from the resulting solution (with parameter MAX_1);
   if the resulting solution s has a cost $F(s) < BestCost$ then set $BestCost \leftarrow F(s)$ and $BestSolution \leftarrow s$;
3. Set $s \leftarrow BestSolution$ and use s as initial solution for TABU_2 in Step 2 of ACM.

Fig. 17 Step 3 of ACM

6.1 Existing algorithms

Among the thirteen competitors, only the four best teams have given a complete description of their solution method. We summarize below each of these four algorithms. More detailed descriptions (but no publication to our knowledge) of these algorithms can be found in ROADEF (1999).

The winners of the contest are Briant and Bouzgarrou. Their algorithm has 6 steps. In the first step, they solve a linear program similar to (IP_1) that produces an initial solution which satisfies constraints (C1) to (C4). In a second step, they make each car (C6)-feasible by subcontracting some requests, if necessary. The third step consists in finding a feasible maintenance schedule for each car. Here again, some requests are possibly subcontracted to make this possible. In the fourth step, they try to reduce the costs by adding a subcontracted request r to a car c, and removing from c all requests that overlap with r. Such a change is done only if a feasible maintenance schedule can be found for c. This is similar to our first neighborhood $N_t^1(s)$, but they move to such a neighbor solution only if it has a better cost than the original solution. The fifth step consists in optimizing the purchase dates as in our PURCHASE procedure. They also decide to subcontract all requests covered by a purchased car if the resulting solution (with one car less) has a smaller cost. Finally, if additional cars can still be bought, they try to cover some subcontracted requests with new cars, if this leads to a better solution.

Asdemir, Karslioğlu, Gürbüz, and Ünal have designed an algorithm that combines three procedures. The first one constructs a solution with a fixed set of purchased cars,

and with fixed dates of purchase. There is initially no purchase and the requests are ordered by non decreasing starting times. Each request r is then considered according to the above order, and assigned, if possible, to an available car c of type $t_c \in T(r)$ so that the total cost is minimized and no maintenance constraint is violated. If no assignment is possible then the request is subcontracted. The second procedure tries to reduce the subcontracting costs by exchanging a subcontracted request r with a request r' covered by a car c of type $t_c \in T(r)$. This is done only if a feasible maintenance schedule can be found for c and the total cost decreases. Such an exchange is a special case of our first neighborhood $N_t^1(s)$ where the subcontracted request r can be added to c only if it overlaps with at most one request covered by c. The third procedure evaluates the cost of buying a new car of type t, if the limit $b(t)$ is not yet attained. This is done by fixing a purchase date for a car of type t at the starting time of the first subcontracted request r with $t(r) = t$, and by using the two previous procedures to generate a solution with this new set of purchases. All car types t are evaluated, and if such a purchase improves the current best solution, then it is implemented.

Bayrak has developed a three phase heuristic. He first orders the car types so that t appears before t' if $t \gg t'$. He then considers each car type t according to this order, and assigns as many requests r with $t(r) = t$ to cars of type t. This is done car by car. More precisely, let $R_t = \{r \text{ with } t(r) = t\}$. The procedure first determines a subset Q of R_t with longest total duration which can be covered by a car without violating constraints (C4), (C5) and (C6). This is repeated for the second car with the remaining requests in R_t, and so on until all requests of R_t are assigned. This gives a partition $(Q_1, \ldots, Q_p)$ of R_t. If $p > |C(t)|$, then the first sets $Q_1, \ldots, Q_{|C(t)|}$ are covered by the $|C(t)|$ cars of type t available in stock. For each additional set Q_i, a new car is bought if the limit $b(t)$ is not yet reached and if the cost induced by such a purchase is smaller than the cost of subcontracting all requests in Q_i. The second phase of the algorithm consists in scheduling the maintenances on the cars. This is done by considering the cars in the same order as above. When no feasible schedule can be found for a car, requests are subcontracted until constraint (C7) is satisfied. The third phase consists in trying to replace subcontracted requests r with upgrades. This is only done if a feasible maintenance schedule can be found and the total cost decreases.

Dhaenens-Flipo and Durand have also developed a three phase heuristic. In the first phase, they order the car types t by non increasing subcontracting cost. They then consider each car type t according to this order, and assign requests from R_t (defined as above) to cars of this type so that the total duration of the assigned requests is maximized. The maintenance constraint (C6) is partially taken into account at this stage by imposing that two requests r and r' having a total duration $\delta(r) + \delta(r') > \frac{2}{3}u(t)$ and such that $(\alpha(r') - \beta(r)) \leq m(t)$ should not be covered by the same car. The number of available cars is supposed equal to $b(t) + |C(t)|$. This problem is formulated as an integer linear program. A maintenance schedule is then built for each car, and requests are subcontracted if necessary. In a second phase, they decide to subcontract all requests covered by a purchased car if the resulting solution (with one car less) has a smaller cost. The third phase consists in trying to replace subcontracted requests r with upgrades. This is only done if a feasible maintenance schedule can be found and the total cost decreases.

Notice that none of the above four algorithms uses a meta-heuristic (e.g. tabu search, simulated annealing, genetic or ants algorithms, adaptive memory programming).

6.2 Comparisons

In this section, we compare our results with those obtained using the above four algorithms. All tests are made on the sixteen instances used in the challenge and which are derived from real-world problems. The name of an instance is coded with a quadruplet (x, y, z, w), where x is the number $|\mathcal{R}|$ of requests, y is the number $|\mathcal{T}|$ of car types, z is the capacity W of the workshop, and w is equal to b if purchases are allowed, and to nb otherwise. So, for example, the instance $(80, 8, 2, b)$ has 80 requests, 8 car types, 2 workers available for the maintenance, and purchases are allowed. The sixteen benchmark instances are divided into two groups depending whether purchases are allowed or not. For each instance without purchase (i.e., $b(t) = 0$ for all car types t), there is a corresponding instance with possible purchase. For example, the instances $(80, 8, 2, nb)$ and $(80, 8, 2, b)$ differ only in the possibility of buying new cars for the second one. The time horizon of all instances is $[0, 730]$ corresponding to a period of 2 years.

We have run our algorithms with the time limit equivalent to one hour on a PC Pentium Pro 200 MHz with 64 Mb memory, as imposed by the organizers of the challenge. The integer linear programs (IP_1), (IP_2), and (IP_3) are solved using CPLEX 8.1. We have made some preliminary tests to determine the length of the tabu lists and the maximum numbers of iterations without improvement of s_{best}. On the basis of these tests, we use $MAX_1 = 10{,}000$, $MAX_2 = 100$, and $MAX_3 = 40$, and $|TL|$ is generated randomly at each iteration in the interval $[1, 12]$ for TABU_1, and in the interval $[1, 40]$ for TABU_2. Also, since TABU_2 is relatively slow, we only generate 30% of the neighbors in $N_2(s)$, these being chosen randomly. Finally, we use $LIM = 10$ at Step 4 of Fig. 15.

The results are shown in Tables 1 and 2. The first table contains the results for the instances without purchase, while the second one is for the other instances. The column labeled *Best* contains the best known solution for each instance. We put an asterisk when we have been able to strictly improve the previous best known solution, or when we could equal it (without of course using it as initial solution for ACM). Some of these best results have been obtained when using different parameters from those mentioned above (for tuning purposes) or by running our algorithms for more than 1 hour. The next 4 columns contain the relative error with respect to *Best* obtained by the 4 methods described in the previous section. These columns are labeled with the first letter of the names of the authors. So, for example, the algorithm proposed by Briant and Bouzgarrou is labeled BB. The next three columns contain the relative error with respect to *Best* obtained with ACM with three different initial solutions. We either start with INIT_1 (column ACM-1), with INIT_2 (column ACM-2), or with the solution produced by the BB algorithm (column ACM-BB). For each instance, ten runs of our algorithms have been executed, and we report average results. The last line of each column indicates average results.

Since ACM was run ten times on each instance, we also report in Table 3 the worst and best solutions obtained with the three different initial solutions. Again, results are

Table 1 Results for the instances without purchase

Instance	Best	BB	AGHKU	H	DD	ACM-1	ACM-2	ACM-BB
(80, 8, 2, *nb*)	1162285*	0.00%	0.05%	1.31%	8.47%	0.00%	0.00%	0.00%
(150, 7, 2, *nb*)	3280230*	0.87%	5.85%	5.03%	9.73%	0.00%	0.00%	0.00%
(160, 12, 2, *nb*)	3333599*	14.63%	19.68%	29.56%	20.51%	0.81%	0.42%	0.32%
(200, 12, 2, *nb*)	5450785*	7.77%	22.56%	25.52%	26.02%	2.59%	3.01%	1.87%
(200, 7, 2, *nb*)	5156915*	6.36%	12.61%	21.02%	31.93%	3.62%	2.28%	1.85%
(200, 7, 4, *nb*)	4558728*	0.00%	0.66%	3.26%	14.45%	0.00%	0.00%	0.00%
(210, 9, 2, *nb*)	5810288*	5.67%	10.76%	18.48%	27.11%	2.42%	2.37%	2.07%
(210, 9, 4, *nb*)	5135237*	1.82%	1.44%	3.46%	13.09%	0.12%	0.04%	0.04%
Average	4236008	4.64%	9.20%	13.46%	18.91%	1.20%	1.02%	0.77%

Table 2 Results for the instances with purchase

Instance	Best	BB	AGHKU	H	DD	ACM-1	ACM-2	ACM-BB
(80, 8, 2, *b*)	1145181*	0.00%	1.55%	2.82%	7.23%	0.04%	0.02%	0.00%
(150, 7, 2, *b*)	2811138	0.00%	9.40%	3.98%	13.23%	0.01%	0.01%	0.00%
(160, 12, 2, *b*)	3064397*	11.99%	29.40%	26.08%	21.98%	1.47%	1.22%	1.09%
(200, 12, 2, *b*)	4517706*	12.42%	34.97%	35.93%	38.13%	4.88%	13.80%	2.25%
(200, 7, 2, *b*)	4990499*	6.88%	15.36%	27.76%	31.38%	4.98%	4.18%	3.19%
(200, 7, 4, *b*)	4092002*	0.00%	3.00%	3.46%	12.76%	0.01%	0.01%	0.00%
(210, 9, 2, *b*)	5380588*	7.38%	18.91%	29.31%	34.15%	3.52%	7.32%	2.55%
(210, 9, 4, *b*)	4147087*	8.71%	10.92%	9.19%	14.91%	0.01%	5.86%	0.06%
Average	3787302	5.92%	15.44%	17.32%	21.72%	1.86%	4.05%	1.14%

relative errors according to *Best* (which can be found in Tables 1 and 2). We can first observe from column *Best* in Tables 1 and 2 that the rental company does not have enough vehicles in stock since it can reduce the total cost by about 10% in purchasing new cars.

The three versions of ACM give on average better results than those obtained by the competitors of the challenge. There are however some instances for which we get solutions with a slight higher total cost. For example, for the instance (200, 12, 2, *b*), ACM-2 deviates by about 1% from the solution produced by BB while ACM-1 is more than 7% better than BB for this instance. For three other instances, (e.g., (80, 8, 2, *b*), (150, 7, 2, *b*), and (200, 7, 4, *b*)), both ACM-1 and ACM-2 deviate from the solution produced by BB by less than 0.04%. For all other instances, ACM-1 and ACM-2 produce results which are equal to or better than BB, the improvement being larger than 10% for the instance (160, 12, 2, *b*). We can also observe in Table 3 that if we take the best of the 10 runs, both ACM-1 and ACM-2 have a cost increase of 0.01% on two instances when compared to BB, and are never worse than BB on the other instances. Also, if we take the worst run, both ACM-1 and ACM-2 are on average better than BB.

Table 3 Minimum and maximum deviations for ACM

Instance	Minimum deviation			Maximum deviation		
	ACM-1	ACM-2	ACM-BB	ACM-1	ACM-2	ACM-BB
(80, 8, 2, *nb*)	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
(150, 7, 2, *nb*)	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
(160, 12, 2, *nb*)	0.18%	0.00%	0.00%	2.28%	1.80%	0.83%
(200, 12, 2, *nb*)	0.00%	1.21%	0.72%	4.56%	7.35%	2.66%
(200, 7, 2, *nb*)	2.31%	1.34%	0.00%	5.48%	3.70%	3.14%
(200, 7, 4, *nb*)	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
(210, 9, 2, *nb*)	0.70%	1.37%	1.24%	4.93%	4.06%	2.59%
(210, 9, 4, *nb*)	0.00%	0.04%	0.03%	0.27%	0.04%	0.04%
Average	0.40%	0.50%	0.25%	2.19%	2.12%	1.16%
(80, 8, 2, *b*)	0.00%	0.00%	0.00%	0.13%	0.13%	0.00%
(150, 7, 2, *b*)	0.01%	0.01%	0.00%	0.01%	0.01%	0.00%
(160, 12, 2, *b*)	0.70%	0.52%	0.11%	3.00%	2.19%	1.76%
(200, 12, 2, *b*)	3.65%	10.22%	0.19%	6.26%	16.69%	4.12%
(200, 7, 2, *b*)	3.81%	1.77%	1.89%	6.54%	6.12%	3.70%
(200, 7, 4, *b*)	0.01%	0.01%	0.00%	0.01%	0.01%	0.00%
(210, 9, 2, *b*)	2.06%	5.62%	1.30%	4.84%	9.41%	4.02%
(210, 9, 4, *b*)	0.00%	5.24%	0.00%	0.07%	6.57%	0.01%
Average	1.28%	2.92%	0.44%	2.61%	5.14%	1.70%

ACM produces on average better results when starting with INIT_1 rather than INIT_2. This is not so clear for instances with no possible purchase since there are instances in Table 1 for which ACM-2 is slightly better than ACM-1. For instances with possible purchase, ACM-1 is on average more than 2% better than ACM-2. This is probably due to the fact that ACM-2 starts with an initial solution without purchase, and therefore spends a lot of time in determining which cars should be bought, while ACM-1 starts with the solution of (IP_1) which has already a good purchase policy.

ACM-BB starts from the solution obtained by the best competitor to the challenge. Since this solution is obtained in about one minute (and cannot be improved by running BB 60 times since BB is totally deterministic), we improve it with ACM for the remaining 59 minutes. As can be observed in Tables 1 and 2, we are able to decrease the total cost by 4% to 5% on average, this improvement being larger than 10% for instance (200, 12, 2, *b*) (and even larger than 12% if we consider the best of the 10 runs of ACM-BB).

ACM can be considered as a robust algorithm since, as can be observed in Table 3, the cost difference between the best and the worst solution over the 10 runs is on average less than 2%.

7 Conclusion

We have developed an efficient algorithm for a car fleet management problem with maintenance constraints. The previous known algorithms combine the solution of integer linear programs with simple constructive or descent techniques. We have added graph optimization procedures and two tabu search algorithms. In particular, we have implemented solution techniques for the graph coloring, the shortest path, and the maximum weight stable set problems. These algorithms make it possible to consider several cars simultaneously when assigning requests to cars or determining maintenance schedules. Also, the neighborhoods used in our two tabu search algorithms allow to move from one solution to a neighbor one by modifying the assignments and maintenance schedules of many cars. This is particularly important when the maintenance constraints (particularly constraint (C7)) are tight, since a modification of a maintenance schedule on a car is often only possible if the maintenance schedules on other cars are changed at the same time (and it may then be necessary to subcontract some requests to be able to move a maintenance to another period).

The computational experiments demonstrate that ACM outperforms all previous known algorithms. In particular, we are able to improve the solutions obtained by the winner of the ROADEF'99 challenge by 4% to 5%, on average. For some instances, we are even able to get a 12% improvement of the best known solution. The best version of ACM is ACM-BB which uses the method proposed by Briant and Bouzgarrou (the winners of the contest) to generate an initial solution.

References

Ahuja, R.K., Magnanti, T.L., Orlin, J.B.: Network Flows: Theory, Algorithms, and Applications. Prentice-Hall, Englewood Cliffs (1993), pp. 107–108

Edelstein, M., Melnyk, M.: The pool control system. Interfaces **8**(1), 21–36 (1997)

Fink, A., Reiners, T.: Modeling and solving the short-term car rental logistics problem. Transp. Res. Part E Logist. Trans. Rev. **42**, 272–292 (2006)

Geraghty, M.K., Johnson, E.: Revenue management saves national car rental. Interfaces **27**(1), 107–127 (1997)

Garey, M., Johnson, D.: Computers and Intractability: A Guide to the Theory of NP-Completeness. Freeman, New York (1979)

Glover, F., Laguna, M.: Tabu Search. Kluwer Academic, Dordrecht (1997)

Golumbic, M.C.: Algorithmic Graph Theory and Perfect Graphs. Academic Press, San Diego (1980)

Gupta, U.I., Lee, D.T., Leung, J.Y.-T.: An optimal solution for the channel-assignment problem. IEEE Trans. Comput. C **28**, 807–810 (1979)

Kolen, A.W.J., Lenstra, J.K.: Combinatorics in operations research. In: Graham, R.L., Groetschel, M., Lovász, L. (eds.) Handbook of Combinatorics, pp. 1875–1910. North-Holland, Amsterdam (1995)

Kolen, A.W.J., Lenstra, J.K.L., Papadimitriou, C.H., Spieksma, F.C.R.: Interval scheduling: A survey. Nav. Res. Logist. **54**, 530–543 (2007)

Pachon, J.R., Iakovou, E., Ip, C., Aboudi, R.: A synthesis of tactical fleet planning models for the car rental industry. IIE Trans. **35**, 907–916 (2003)

ROADEF Challenge: Description of the problem. http://www.prism.uvsq.fr/~vdc/ROADEF/CHALLENGES/1999/ (1999)

Shamir, R.: Advanced topics in Graph Algorithms. Technical Report, Har-Peled, S. (ed.), Tel Aviv University, Israel, pp. 111–113 (1994)

Silver, E.A., Pyke, D.F., Peterson, R.: Inventory Management and Production Planning and Scheduling. Wiley, New York (1998)