Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2014, Article ID 483734, 11 pages
http://dx.doi.org/10.1155/2014/483734

Research Article

An Iterative Algorithm for the Management of an Electric Car-Rental Service

J. Alberto Conejero,[1] Cristina Jordán,[2] and Esther Sanabria-Codesal[3]

[1] *Instituto Universitario de Matemática Pura y Aplicada, Universitat Politècnica de València, 46022 València, Spain*
[2] *Instituto Universitario de Matemática Multidisciplinar, Universitat Politècnica de València, 46022 València, Spain*
[3] *Departamento de Matemática Aplicada, Universitat Politècnica de València, 46022 València, Spain*

Correspondence should be addressed to J. Alberto Conejero; aconejero@upv.es

Received 21 February 2014; Accepted 24 April 2014; Published 25 May 2014

Academic Editor: Juan R. Torregrosa

The management of a car-rental service becomes more complex as long as one-way bookings between different depots are accepted. These bookings can increase the operational costs due to the necessity of moving vehicles from one depot to another by the company staff in order to attend previously accepted bookings. We present an iterative model based on flows on networks for the acceptance of bookings by a car-rental service that permits one-way reservations. Our model lets us also recover the movement of the fleet of vehicles between the depots over the time. In addition, it also permits including restrictions on the amount of cars managed at every single depot. These results can be of interest for an electric car-rental service that operates at different depots within a city or region.

1. Introduction

Car rentals can admit bookings where the pick-up and drop-off location can coincide or not. If bookings are only admitted when both locations coincide, their management is not as complicated as if bookings are only accepted when the locations are different (one-way bookings). When this happens, these companies face much more difficulties in order to cope with the management of their fleet of vehicles, since these bookings could entail a huge increment of operational costs due to deadhead times. Nevertheless, in order to offer a high level of service thier policy is to accept bookings without a deep examination of its consequences on the operational costs [1].

The use of fully electric vehicles (FEV) is fostered by government authorities in order to contribute to the lowering of the current pollution levels [2]. This comes jointly with a gradual phasing out of conventionally-fuelled vehicles from the urban environment [3]. Some references concerning the adoption of these new transportation devices can also be found in [4, 5].

The easiest way to implement a rental service of electric vehicles is to make the users return them at the place where the cars have been picked off. This is done in order to be sure that every car has its own parking place with a charging point. Therefore, an improvement in the process of acceptance of bookings that consider one-way trips between different stations and try to avoid, as far as possible, the amount of staff dedicated to the rearrangement of vehicles between the depots is of particular interest in order to facilitate and extend their use among the potential users. Therefore, a social benefit is obtained as long as the number of reservations of these shared vehicles is increased and the number of private cars running is reduced.

Flows and networks were firstly used in the air industry for solving fleet routing problems, see for instance [6–9]. The problems that can be solved using them range from schedule design, flight assignment [10], and crew scheduling [11] to scheduling of air cargo alliances [12]. Their techniques have been also considered in the management of car-rental services in order to answer questions of strategic and tactical

decisions, revenue and capacity management, and pricing, see for instance [1, 13–15].

We suppose that a simulation of bookings for a car-rental service is given. We model the process of acceptance/rejection of those petitions. This depends on having a car available at the departure point (without leaving unattended any other previously scheduled booking from that place) and on having an empty parking space at the arrival destination. We assume that an answer should be given as fast as possible to the client. Therefore, we will answer each petition before considering the next one. We do not consider the case that a rejected booking can turn to be admissible after some other booking has been accepted. As we said before, the general practice in the car-rental industry is to accept bookings and later to optimize the management of the fleet of vehicles.

The car-rental services considered in the frame of this paper are managed with a little different policy. We assume that one of these services does not need to compete with other services and it is just an optional element inside the mobility network of an urban area, where other means of transport are offered to the citizens. Therefore, if a booking should be rejected because of the lack of available cars or empty parking slots, then we reject it since we consider that this will have a low impact on the service perception of prospective users, as long as other means of transport can be used. In this way we determine which bookings should be admitted without falling on deadhead times due to the rearrangement of cars by the car-rental service staff.

The paper is organized as follows. In Section 2 we introduce some basics of graph theory about flows and networks. We also present time-expanded networks. Our model for managing the bookings using a network is presented in Section 3. Section 4 is devoted to explain the theoretical background that backs our computational method. A first algorithmic approach based on Ford-Fulkerson algorithm is presented in Section 5. We will see that the problem that arises can be stated in terms of finding an admissible flow on a network with minimum capacities at certain edges. Later, an iterative method based on a simplification of the auxiliary network defined for considering the admissibility of the flow will be given in Section 6. This method lays on a solution of certain shortest path problem. An analysis of the results is reported in Section 7. Finally in Section 8 we discuss the contribution of the model to the management of a service in order to offer a cheaper rental service, which will contribute to extend its use for the movements within an urban area. Our algorithm helps to determine if it is advisable to increase the fleet of vehicles or the size of the parking at a certain depot, attending to the increments of cost and to the number of additional bookings accepted.

2. Preliminaries

A *directed graph* $G = (V, E)$ is given by a set of nodes V and a set of ordered pairs of nodes (arcs). If an arc connects the node u with the node v, we will simply denote it by (u, v). We recall that a 5-tuple $N = (V, E, s, t, c)$ is named *network* if it is a directed weakly connected graph where V is the set of nodes, E is the set of arcs, s is a node with outgoing degree positive (at least one edge departs from it) usually called *source*, t is a node with ingoing degree positive (at least one edge arrives to it) usually called *sink*, and c is a function from E to $\mathbb{N}$ that assigns to every edge $(u, v) \in E$ a value $c(u, v)$ that will represent the maximum capacity permitted on this edge. Given a network N, we can consider flows on them. A *flow* f is a function $f : E \to \mathbb{N}$ such that $f(u, v) \leq c(u, v)$ for every $(u, v) \in E$, and for every $v \in V \setminus \{s, t\}$ the sum of the values of the flow on the edges with v as initial node coincides with the sum of the flows on the edges with v as final node; that is,

$$\sum_{u \in V, (u,v) \in E} f(u, v) = \sum_{u \in V, (v,u) \in E} f(v, u). \tag{1}$$

Condition (1) is usually known as the *conservation law of the flow*. The value of the flow f on N, $f(N)$, is defined as the sum of the flows on the adjacent arcs to s (or to t):

$$f(N) = \sum_{u \in V, (s,u) \in E} f(s, u) = \sum_{u \in V, (u,t) \in E} f(u, t). \tag{2}$$

A flow f on N is said to be *maximum* if it has the maximum value among all possible flows that can be defined on N. We refer the reader to [16–24] for general information on flows and networks and graph theory.

In particular, we are going to consider *time-expanded* networks, also called *time-space* networks [18, 25]. In these networks, the inner nodes of the network (not the sink nor the source) represent locations at certain times. The source is connected with all locations at the initial time and all locations at the final time are connected with the sink. This is also the structure used for dealing with problems in which the flow emerges from several sources and leaves at several sinks.

Now, we transform this network into a one time-expanded network in which we reply cities as many times as time periods we are going to consider, see Figure 1. Nodes in each column represent the same city and each level represents the same time period for different cities.

In this model we also consider another value associated with each arc named its *minimum capacity*. The minimum capacity is a function $m : E \to \mathbb{N}$. The flow through every edge $(u, v) \in E$ must verify $m(u, v) \leq f(u, v) \leq c(u, v)$. We denote a network N with minimum capacities as $N = (V, E, s, t, m, c)$, where V is the set of nodes, E the set of edges, s the source, t the sink, and m, c the minimum and maximum capacities, respectively. Networks with maximum and minimum capacities cannot always admit a flow that respects the upper and lower bounds. It is compulsory that, for every node $v \in V$, the sum of the maximum capacities of all the edges that arrive to v was greater than or equal to the sum of the minimum capacities of all the edges that depart from it. Consider

$$\sum_{u \in V, (v,u) \in E} m(v, u) \leq \sum_{u \in V, (u,v) \in E} c(u, v). \tag{3}$$

If not, the conservation law never holds.

In the next section, we proceed to explain in detail the steps that let us construct a model for deciding whether to

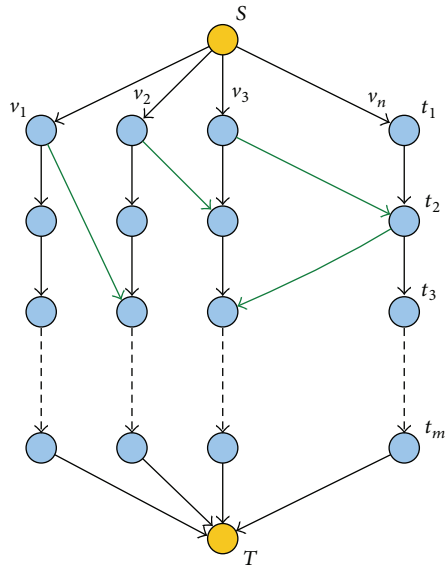

FIGURE 1: A time-expanded network with n depots at m times. Arcs in green represent bookings of cars and arcs in black represent staying at the parking of each depot.

accept or reject a booking attending to the distribution of cars at the departure depot and the existence of empty parking places at the arrival one.

3. A Car-Rental Booking Model

Suppose that our car-rental service operates with only one type of car. Let n be the number of locations (depots) and assume that depot i starts with a given number of cars k_i. Let k_0 be the total number of cars of the company; that is, $k_0 = \sum_{i=1}^{n} k_i$. In addition, we suppose that depot i has p_i parking slots at its location to park the cars that are not used by any client and are recharging and waiting to be reserved. We assume that this number cannot be increased along the time.

We also consider the bookings along m consecutive days, h possible times a day. Then the sequence $\{t_j : 1 \le j \le hm\}$ represents all the possible time periods. For instance, if we consider the times 9:00, 10:00,. . ., 20:00 then $h = 12$. So that $v_{i,12(a-1)+b}$, for some $1 \le a \le m$ and $1 \le b \le 12$, represents (8+b):00 a.m. of the day a at the location i.

The set of nodes V will consist of s, t (the source and the sink) and all the possible combinations of locations and times $\{v_{i,j} : 1 \le i \le n;\ 1 \le j \le hm\}$ where $v_{i,j}$ represents the location i at time t_j.

The set E is formed by four types of arcs. The first three are the following ones.

(i) $(s, v_{i,1})$, $1 \le i \le n$, that connect the source with the depots at the first instant of time.

(ii) $(v_{i,j}, v_{i,j+1})$, $1 \le i \le n$, $1 \le j \le hm - 1$, that represent staying at a parking between two consecutive times.

(iii) $(v_{i,hm}, t)$, $1 \le i \le n$, that connect the depots at the last time to the sink.

Our objective is to know if we can accept a booking of w cars between two determined depots at two different time moments under the restrictions of having enough vehicles at the departure depot and free space at the arrival one. We define a booking by a 5-tuple $r = (i_p, t_p, i_d, t_d, w)$, where $1 \le i_p, i_d \le n$, $1 \le t_p < t_d \le hm$, and $n \in \mathbb{N}$, with i_p being the pick-up depot, t_d being the pick-up time, i_d being the drop-off city, t_d being the drop-off time, and w being the number of cars to be reserved. Each booking $r = (i_p, t_p, i_d, t_d, w)$ is converted into a new edge from node v_{i_p,t_p} to node v_{i_d,t_d}. This will be the fourth type of arcs of our network.

Now, we analyze which capacities are necessary to model the movement of the cars along the time and to accept only admissible bookings.

We have supposed that depot i has p_i parking spaces. To set this restriction we assign p_i as maximum capacity to the edge $(v_{i,j}, v_{i,j+1})$ $1 \le i \le n$, $1 \le j \le hm$, and every node of the form $v_{i,hm}$, $1 \le i \le n$, must be connected with the sink t by an edge with p_i as maximum capacity.

Finally we must assign a minimum capacity to arcs $(s, v_{i,1})$. If not, the cars at a depot i could not be considered in the parking in the future. In Figures 2 and 3 every edge $(u, v) \in E$ has a 3-tuple $(m(u, v), f(u, v), c(u, v))$ associated with it where m and c are the minimum and maximum capacities, respectively, and f denotes the flow. If we suppose that arcs $(s, v_{i,1})$ have assigned minimum capacity equal to the number of cars of every depot, k_i, we will avoid the aforementioned problem. The following example shows the necessity of a minimum capacity for the edges that represent the starting time at every parking.

Example 1. Suppose that we start with 6 cars at depot 1 at time $t = 1$ and with 7 cars at depot 2 at time $t = 1$ too. We also consider that we have 7 and 8 parking slots, respectively, at each one of these depots. Let us suppose that we have to decide whether the booking $(2, 1, 1, 2, 3)$ could be accepted or not.

In Figure 2 we have a piece of a network without minimum capacities in the edges $(s, v_{1,1})$ and $(s, v_{2,1})$. In this case, there is a flow of 3 cars across the edge $(v_{2,1}, v_{1,2})$, and therefore the booking would be accepted. Nevertheless, there are 6 cars parked at slot 1 during all periods of time and this is not taken into account. If they were, then the proposed booking cannot be accepted as it is indicated in Figure 3, since we will have a total number of 9 cars that arrive to node $v_{1,2}$. This is not possible since the conservation law fails due to the fact that the maximum capacity of the only edge that departs from $(v_{1,2})$ is 7.

For edges representing bookings, we also have to consider that the minimum and maximum capacity must coincide in order to be sure that the cars follow the booking itinerary and do not stay at the parking of the picking up depot. Next example shows the necessity of minimum capacity for edges associated with bookings.

Example 2. Suppose that we want to represent a booking of the form $(1, 2, 2, 3, 1)$. This is done by considering a flow of 1 car along the edge $(v_{1,2}, v_{2,3})$.

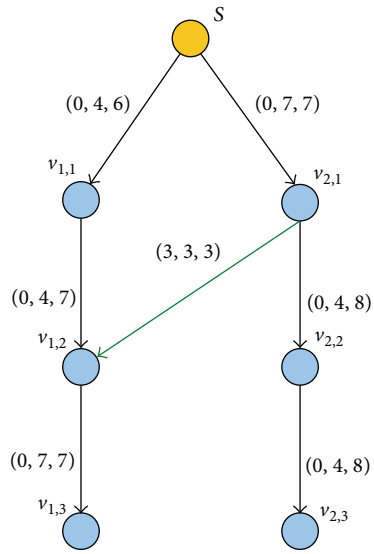

Figure 2: Without minimum capacities on the arcs that depart from S.

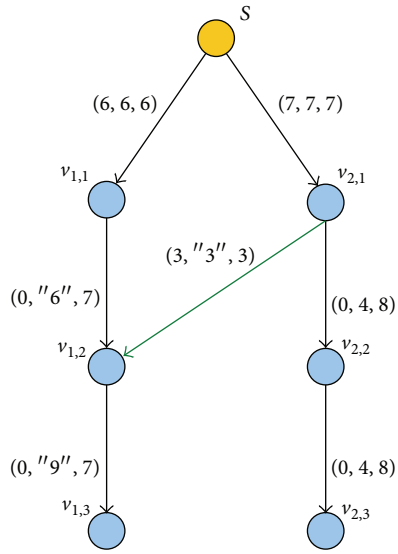

Figure 3: With minimum capacities on the arcs that depart from S.

In Figure 4 we have a piece of a network without minimum capacities at the edge $(v_{1,2}, v_{2,3})$. In this case, we cannot be sure that the flow of 1 unit that corresponds to this booking will traverse the edge, and therefore the car will go from depot 1 to depot 2 at those times. We can only ensure this if we assign a minimum capacity of 1 unit to this edge, as it is shown in Figure 5.

4. Existence of an Admissible Flow on the Network

The flow over the network that we have already presented simulates the movement of the cars over time. As we said before, a flow through a network with minimum capacities could not be admissible. Firstly, the initial number of cars at every depot must be smaller or equal to the number of parking slots at each depot; that is, $k_i \leq p_i$ for $i = 1, \ldots, n$. Secondly, a list of bookings, which is in fact a list of new edges to be added to the former network, could result into a network without any admissible flow.

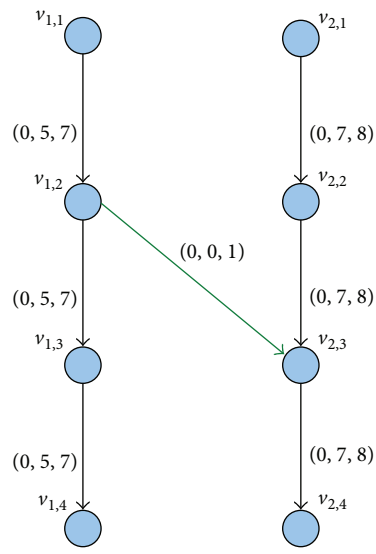

Figure 4: Without minimum capacities on the arcs that represent a booking.

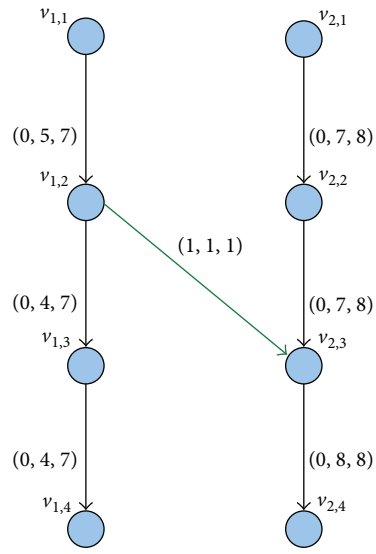

Figure 5: With minimum capacities on the arcs that represent a booking.

Now, we start with the initial network N that contains only the first three types of edges that were previously mentioned. This network has the following values as maximum capacities at their edges:

$$c(u,v) = \begin{cases} k_i & \text{for } u = s,\ v = v_{i,1},\ i = 1,\ldots,n \\ p_i & \text{for } u = v_{i,j},\ \ v = v_{i,j+1},\ \ i = 1,\ldots,n, \\ & \quad j = 1,\ldots,hm-1, \\ p_i & \text{for } u = v_{i,hm},\ v = t,\ i = 1,\ldots,n. \end{cases} \tag{4}$$

In addition, we also set the following minimum capacities:

$$m(u,v) = \begin{cases} k_i & \text{for } u = s,\ v = v_{i,1},\ i = 1,\ldots,n \\ 0 & \text{for } u = v_{i,j},\ \ v = v_{i,j+1},\ \ i = 1,\ldots,n, \\ & \quad j = 1,\ldots,hm-1, \\ 0 & \text{for } u = v_{i,hm},\ v = t,\ \ i = 1,\ldots,n. \end{cases} \tag{5}$$

Then, the following flow results in being admissible on this network:

$$f(u,v) = \begin{cases} k_i & \text{for } u = s,\ v = v_{i,1},\ i = 1,\ldots,n \\ k_i & \text{for } u = v_{i,j},\ v = v_{i,j+1},\ i = 1,\ldots,n, \\ & \quad j = 1,\ldots,hm-1, \\ k_i & \text{for } u = v_{i,hm},\ v = t,\ i = 1,\ldots,n. \end{cases} \tag{6}$$

Now, let us suppose that we have an ordered list of L bookings requests $\{r_1,\ldots,r_L\}$, where each booking is given by its corresponding 5-tuple. Consider that these bookings have arrived to us in advance to t_1, for instance through the website of the car-rental company. We have to decide whether we can accept them or not. The acceptance of each booking depends on the acceptance or not of the previous ones. We assume that all bookings must be answered following a first-in/first-out criterion, which is the closest approach to the management of bookings that arrive through a web service.

We start with the first booking request $r_1 = (i_{p_1}, t_{p_1}, i_{d_1}, t_{d_1}, w_1)$ and we add the corresponding edge from node $v_{i_{p_1},t_{p_1}}$ to $v_{i_{d_1},t_{d_1}}$ with maximum and minimum capacities equal to w_1.

In order to know if there exists an admissible flow in the network $N = (V, E, s, t, m, c)$ with the additional edge corresponding to booking r_1, we define an auxiliary network N' and apply Ford-Fulkerson algorithm on it. The explanation of Ford-Fulkerson algorithm can also be found in [16, 18, 19, 21].

The use of this network N' in order to determine the existence of an admissible flow on a network with minimum capacities is given by the following result that can be found either in [16, page 83] or in [21, page 92]. Its application provides us with a theoretical support for modeling our problem.

Theorem 3. *Let $N = (V, E, s, t, m, c)$ be a network and let one consider its auxiliary network N' defined as follows.*

(1) *The set of nodes of N' consists of the nodes of N and two new auxiliary nodes s', t' that will be the source and the sink, respectively, of the network N'.*

(2) *To define E' we consider all the edges of E and we add a new one from t to s, with infinite capacity.*

(3) *For every edge $(u, v) \in E$ with capacity $m(u, v) \geq 0$ we also change its maximum capacity $c(u, v)$ by $c(u, v) - m(u, v)$ and we add a new edge (s', v) with maximum capacity $c'(s', v) = m(u, v)$ and another one (u, t') with maximum capacity $c'(u, t') = m(u, v)$.*

(4) *We also set all the minimum capacities of N' to zero.*

Let f' be a flow of maximum value $f'(N')$ on N', and let $f'(t, s)$ be the flow of f' through the edge (t, s). There exists an admissible flow of value $f'(t, s)$ on N if and only if $f'(N') = \sum_{(u,v)\in E} m(u, v)$.

Remark 4. The proof of Theorem 3 shows how to construct the admissible flow; we just have to define $f(u, v) = f'(u, v) + m(u, v)$ for every edge (u, v) in the initial network N, and its value will be $f'(t, s)$.

To sum up, the network N' obtained when considering also the first booking r_1 has $V \cup \{s', t'\}$ as the set of nodes, s' as the source, t' as the sink, and the following edges with their corresponding maximum capacities. These new capacities are defined as:

$$c(u,v) = \begin{cases} \infty, & \text{for } u = t,\ v = s, \\ \sum_{i=1}^{n} k_i & \text{for } u = s,\ v = t', \\ k_i & \text{for } u = s',\ v = v_{i,1},\ \ i = 1,\ldots,n, \\ p_i & \text{for } u = v_{i,j},\ \ v = v_{i,j+1},\ \ i = 1,\ldots,n, \\ & \quad j = 1,\ldots,hm-1, \\ p_i & \text{for } u = v_{i,hm},\ \ v = t,\ \ i = 1,\ldots,n, \\ w_1 & \text{for } u = v_{i_{p_1},t_{p_1}},\ v = t', \\ w_1 & \text{for } u = s',\ v = v_{i_{d_1},t_{d_1}}. \end{cases} \tag{7}$$

We have eliminated the edges that should appear on N' with maximum capacity equal to 0. If there exists a maximum flow on this network of value $f'(N') = \sum_{i=1}^{n} k_i +$

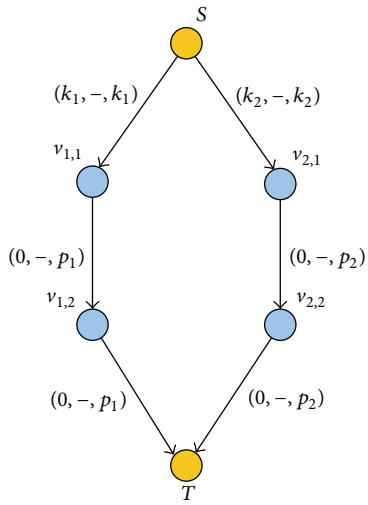

Figure 6: Network N.

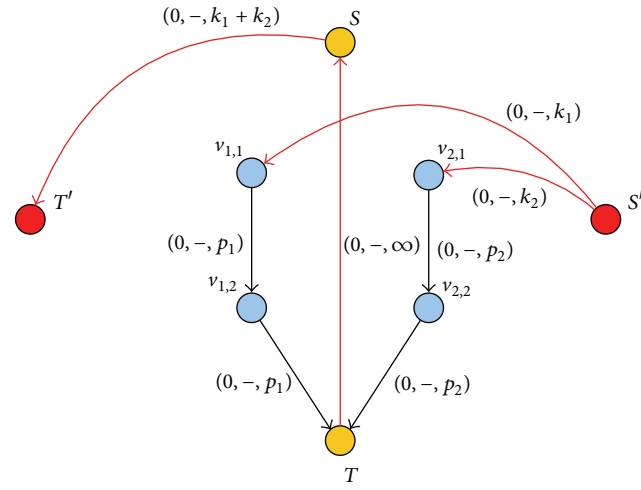

Figure 7: Auxiliary network N'.

w_1, then the booking should be accepted by Theorem 3. If it is not accepted, we remove the edges $(v_{i_{p_1},t_{p_1}}, t')$ and $(s', v_{i_{d_1},t_{d_1}})$. If yes, then we keep these two edges on N'.

Example 5. Suppose that we have two depots, 1 and 2, with k_1 cars at depot 1 and k_2 cars at depot 2. We consider each depot at two different times, $t = 1$ and $t = 2$. Suppose that we have a booking of w_1 cars from depot 1 at time t_1 to depot 2 at time t_2. In Figure 6 we have the initial network N with the minimum and maximum capacities at every edge. The flow is not indicated at any arc. In Figure 7 we have the corresponding auxiliary network N' where we have reduced the minimum capacities 0 of the arcs as it is indicated in (7). We recall that the arcs with maximum capacity equal to 0 are also removed.

After dealing with r_1, we pass to next booking $r_2 = (i_{p_2}, t_{p_2}, i_{d_2}, t_{d_2}, w_2)$ and we proceed as before, adding two new edges to N' with capacity w_2:

$$c(u, v) = \begin{cases} w_2, & \text{for } u = v_{i_{p_2},t_{p_2}}, \quad v = t', \\ w_2, & \text{for } u = s', \ v = v_{i_{d_2},t_{d_2}}. \end{cases} \tag{8}$$

If for dealing with an arbitrary booking $r = (i_p, t_p, i_d, t_d, w)$, one (or two) of the edges (v_{i_p,t_p}, t') and (s', v_{i_p,t_p}) are already defined on N'; what we have to do is to sum w to the existing maximum capacity of that edge and test the admissibility of N'. If the booking should not be accepted, then we just have to decrease the maximum capacity of that edge on w, leaving the edge with the previous capacity.

Remark 6. Our model lets us cancel a booking some time after it was already accepted. It is sufficient to decrease the maximum capacity along the corresponding edge in as many units as the booking indicates, removing it if, afterwards, its maximum capacity is 0. But we have to check if doing this we get a network without an admissible flow. On one hand, this could happen with one-way bookings when the car was required to be at another city in the future for being used for another booking departing from there. In this case, the car must be moved there by the company staff. On the other hand, the lack of an admissible flow on the network can also come from the necessity of needing more parking spaces available at the departure city during part of the time that the car was reserved.

5. Algorithm for Computing the Admissibility of the Network N

In this section we explain how we define the graph of N' in order to maximize the flow on this network. The Ford-Fulkerson algorithm provides a solution for the quest of a maximum flow between a source and a sink on a network.

As we have mentioned in the previous section, the acceptance of a booking depends on verifying whether a certain network N with minimum capacities admits a flow or not, and this can be reduced to find a maximum flow on an auxiliary network N'. The following algorithm shows how to define N' and how to use it in order to accept or reject a reservation.

Step 1. Notation

(1.1) n, number of depots.

(1.2) For every $1 \le i \le n$, p_i stands for the number of parking places available at depot i.

(1.3) For every $1 \le i \le n$, k_i denotes the initial number of cars at depot i.

(1.4) hm is the total number of time moments considered, $t_1, \ldots, t_{hm}$.

(1.5) The network N' will have $hmn + 4$ nodes; that is, n depots at hm different times plus 4 nodes for s, t, s', and t'.

(1.6) $A = (a_{i,j})$ is matrix of capacities in N', whose dimensions are $(hmn+4)\times(hmn+4)$. The nodes are enumerated in the adjacency matrix as follows:

(i) For $1 \le i \le n, 1 \le j \le hm$, $v_{i,j} \to n(j-1)+i$,

(ii) $t \to hmn+1$,

(iii) $s \to hmn+2$,

(iv) $t' \to hmn+3$,

(v) $s' \to hmn+4$.

(1.7) Set $\{r_1, \ldots, r_L\}$ as a list of bookings. Set $\alpha = 1$, which will be the counter for the reservations.

(1.8) Set $\Lambda = 0$, which will accumulate the number of cars in accepted bookings.

Step 2. Initilialize A

For every $1 \le i, j \le hmn+4$, we define the following:

$$a_{i,j} = \begin{cases} p_\ell & \text{for } i = kn+\ell,\ j = i+n,\ 0 \le k \le hm-2,\\ & 1 \le \ell \le n,\\ p_\ell & \text{for } i = (hm-1)n+\ell,\ j = hmn+1,\\ & 1 \le \ell \le n,\\ k_\ell & \text{for } i = hmn+4,\ \ j = \ell,\ 1 \le \ell \le n,\\ \sum_{\ell=1}^{n} k_\ell & \text{for } i = hmn+2,\ \ j = hmn+3,\\ \infty & \text{for } i = hmn+1,\ \ j = hmn+2,\\ 0 & \text{elsewhere.} \end{cases} \tag{9}$$

Remark. $a_{i,j} = 0$ can represent the lack of an edge or the existence of an edge with capacity equal to 0.

Step 3. Analyze the bookings.

While $\alpha \le L$, take $r_\alpha = (i_{p,\alpha}, t_{p,\alpha}, i_{d,\alpha}, t_{d,\alpha}, w_\alpha)$.

(3.1) Update the following elements of A:

$$a_{i,j} = \begin{cases} a_{i,j} + w_\alpha & \text{for } i = hmn+4,\ \ j = n\left(t_{d,\alpha}-1\right)+i_{d,\alpha},\\ a_{i,j} + w_\alpha & \text{for } i = n\left(t_{p,\alpha}-1\right)+i_{p,\alpha},\ \ j = hmn+3. \end{cases} \tag{10}$$

(3.2) Maximize the flow $f(N')$ of the network N' using Ford-Fulkerson algorithm.

(3.3) If $f(N') = \Lambda + \sum_{i=1}^{n} k_i + w_\alpha$, then

(i) accept the booking r_α,

(ii) update $\Lambda = \Lambda + w_\alpha$

elsewhere,

(iii) undo the assignments in A of Step (3.1).

(3.4) Set $\alpha = \alpha + 1$ and return to Step 3.

Step 4. We finish when all the bookings have been processed.

In order to reduce the computational cost of the above algorithm we have analyzed its implementation based on the notion of a residual network. Let $N = (V, E, s, t, c)$ be a network with a flow f over N. Without loss of generality, we assume that each pair of nodes is connected in just one sense, either $(u,v) \in E$ or $(v,u) \in E$ but not at the same time. We define the *residual network* N_f as the network $N_f = (V, \mathscr{E}, s, t, \widetilde{c})$, with the same nodes, source, and sink as N. The set of arcs $\mathscr{E}$ consists of all the arcs of the form (u,v) such that either (u,v) or (v,u) belong to E; that is, every arc of E is maintained and it has also an associated arc in the opposite sense. Besides, the new capacities, $\widetilde{c}$, on this new set $\mathscr{E}$ are defined as

$$\widetilde{c}(u,v) = \begin{cases} c(u,v) - f(u,v), & \text{if } (u,v) \in E,\\ f(u,v), & \text{if } (v,u) \in E. \end{cases} \tag{11}$$

We define an *f-augmenting path* on N_f as a path from s to t, namely, $\widetilde{p}$, such that all the edges in this path have positive capacity. We define the *capacity of* $\widetilde{p}$, namely, $\widetilde{c}_{\widetilde{p}}$ as the minimum of the capacities of all the arcs in $\widetilde{p}$. The flow is maximum when there is not an *f-augmenting path* on N_f.

Let us consider the network N_f' associated with the network N' defined in previous algorithm with the initial flow defined in (6), which results in being admissible in N'. Then, the new adjacency matrix $A_f = (a_{i,j}^f)$ that corresponds to the network N_f' will be

$$a_{i,j}^f = \begin{cases} p_\ell - k_\ell & \text{for } i = kn+\ell,\ j = i+n,\ \ 0 \le k \le hm-2,\\ & 1 \le \ell \le n,\\ k_\ell & \text{for } j = kn+\ell,\ i = j-n,\ 1 \le k \le hm-1,\\ & 1 \le \ell \le n,\\ p_\ell - k_\ell & \text{for } i = (hm-1)n+\ell,\ j = hmn+1,\\ & 1 \le \ell \le n,\\ k_\ell & \text{for } i = hmn+1,\ \ j = (hm-1)n+\ell,\\ & 1 \le \ell \le n,\\ k_\ell & \text{for } i = \ell,\ j = hmn+4,\ \ 1 \le \ell \le n,\\ \sum_{\ell=1}^{n} k_\ell & \text{for } i = hmn+3,\ j = hmn+2,\\ \sum_{\ell=1}^{n} k_\ell & \text{for } i = hmn+2,\ j = hmn+1,\\ \infty & \text{for } i = hmn+1,\ \ j = hmn+2,\\ 0 & \text{elsewhere.} \end{cases} \tag{12}$$

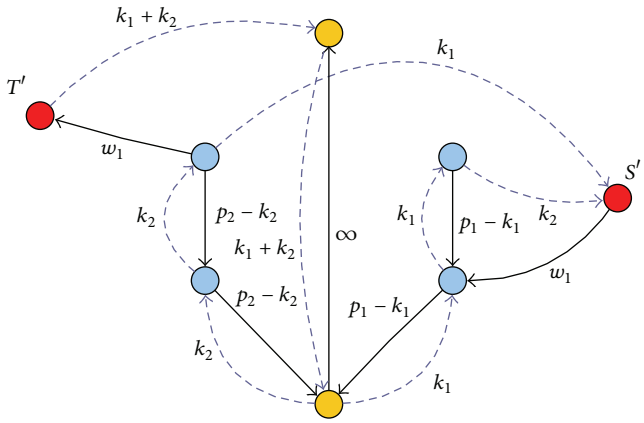

FIGURE 8: The network of Example 5 after the first iteration of Step (3.1) in the algorithm. The arcs in black belong to N', and the arcs with a purple (dashed) line represent the additional edges to be considered with the former ones of N' in order to have the residual network of N'.

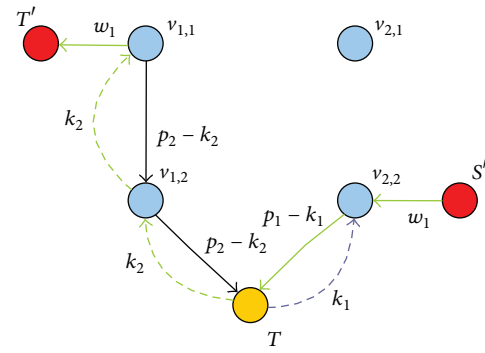

FIGURE 9: A simplified version of the residual network of Example 5 is to be considered in order to decide whether to accept or not the booking $(v_1, 1, v_2, 2, w_1)$. This is accepted if we find an f-augmenting path from S' to T' to be increased in w_1 units. The arcs in the f-augmenting path are represented in green.

After each booking $r_\alpha = (i_{p,\alpha}, t_{p,\alpha}, i_{d,\alpha}, t_{d,\alpha}, w_\alpha)$, we look for an f augmenting path $\tilde{p}$ from s' to t' in the residual network associated with the network N'_f. The residual one is updated following the indications in (10). For the sake of clarity, we provide, in Figure 8, the residual network of N' in Example 5 after the first iteration of Step (3.1) in the algorithm. The edges with 0 capacity and the labels of all the nodes except s' and t' have been removed in order to get a neater picture.

6. A Simplified Algorithm for Computing the Admissibility of a Flow on a Network

We refer again to Example 5. The nodes that stand for the drop-off depot before the drop-off time, the nodes that refer to the pick-up depot before the pick-up time, and s can be removed because if some of them appear in an f-augmenting path, they do it in a cycle, and this cycle can be erased from the f-augmenting path without decreasing its capacity. Therefore, the only way to find an f-augmenting path without cycles from s' to t' is to consider the following path: $\{s', v_{2,2}, t, v_{1,2}, v_{1,1}, t'\}$. Therefore, a simplification of the residual network in what concerns to this booking can be seen in Figure 9. Such a path exists if the following conditions are fulfilled whether $w_1 \leq p_1 - k_1$ and $w_1 \leq k_2$. So by checking them we can omit the restrictions given by the arcs $(s', v_{2,2})$ and $(v_{1,1}, t')$, and the network N' is reduced to the time-space part plus the initial sink t, see Figure 10.

Taking this into account, Step 3 in the algorithm of Section 5 can be significantly simplified. Let us take a booking (i_p, t_p, i_d, t_d, w). In order to accept it, we only have to find, if there exists, a path that connects the node v_{i_d,t_d} with the node v_{i_p,t_p} with capacity greater than or equal to w. Such a path can be easily found using a shortest path algorithm, like Dijkstra's,

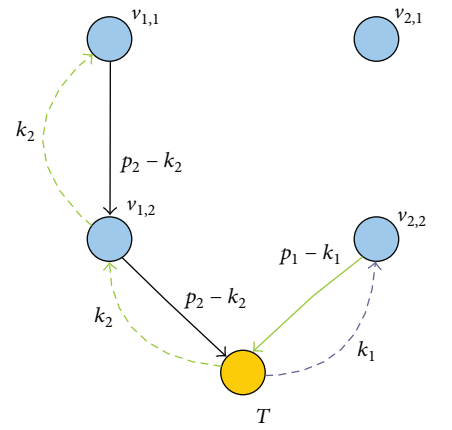

FIGURE 10: A more simplified version of the residual network of Example 5. The booking $(v_1, 1, v_2, 2, w_1)$ will be accepted if there is an f-augmenting path from v_2 at time $t = 2$ to v_1 at time $t = 1$ to be increased in w_1 units. The arcs in the f-augmenting path are represented in green.

in the residual network associated with N'. This path will be of the form

$$v_{i_p,t_p}, v_{i_p,t_{p+1}}, \ldots, v_{i_p,t_{hm}}, t, v_{i_d,t_{hm}}, \ldots, v_{i_d,t_{d+1}}, v_{i_d,t_d}. \quad (13)$$

Further information concerning Dijkstra's algorithm can be found, for instance, in [16, 18, 19, 21, 24].

We point out that the use of a shortest path algorithm prevents us from getting an f-augmenting path with cycles. With respect to the previous algorithm, we modify the steps indicated below but considering the matrix A_f previously defined.

Step 1′. Notation

(1.6.bis) $A_f = (a^f_{i,j})$ a matrix of dimensions $(hmn+1)\times(hmn+1)$. The nodes are enumerated in the adjacency matrix as follows:

(i) $v_{i,j} \rightarrow n(j-1)+i$,

(ii) $t \to hmn + 1$;

that is, we have removed the vertex s, t', and s' and the adjacent arcs to them. In addition, Step (1.8) is removed.

Step 2′. Initialization of A_f

In the same way as in (12), but restricting ourselves to the new dimensions of A_f, we omit s, s', t', and their adjacent arcs.

Step 3′. Analyze the bookings

While $\alpha \le L$, take $r_\alpha = (i_{p,\alpha}, t_{p,\alpha}, i_{d,\alpha}, t_{d,\alpha}, w_\alpha)$.

(3.1)′ Find a shortest path $\widetilde{p}$ between the nodes $n(t_{d,\alpha} - 1) + i_{d,\alpha}$ and $n(t_{p,\alpha} - 1) + i_{p,\alpha}$.

(3.2)′ Compute the capacity of $\widetilde{p}$, namely, $\widetilde{c}_{\widetilde{p}}$.

(3.3)′ If $w_\alpha \le \widetilde{c}_{\widetilde{p}}$ then do the following.

(i) Accept the booking r_α.

(ii) Update the following elements of A_f:

$$a_{i,j}^f = \begin{cases} a_{i,j}^f - w_\alpha \\ \text{for } i = n\left(t_{d,\alpha} - 1\right) + i_{d,\alpha} + kn, \\ \qquad j = i + (k+1)\,n,\ k = 0, \ldots, hm - t_{d,\alpha}, \\ a_{i,j}^f + w_\alpha \\ \text{for } i = j + (k+1)\,n,\ j = n\left(t_{d,\alpha} - 1\right) + i_{d,\alpha} + kn, \\ \qquad k = 0, \ldots, hm - t_{d,\alpha}, \\ a_{i,j}^f - w_\alpha \\ \text{for } i = n\,(hm - 1) + i_{d,\alpha},\quad j = hmn + 1, \\ a_{i,j}^f + w_\alpha \\ \text{for } i = hmn + 1,\quad j = n\,(hm - 1) + i_{d,\alpha}, \\ a_{i,j}^f + w_\alpha \\ \text{for } i = n\,(hm - 1) + i_{p,\alpha},\quad j = hmn + 1, \\ a_{i,j}^f - w_\alpha \\ \text{for } i = hmn + 1,\quad j = n\,(hm - 1) + i_{p,\alpha}, \\ a_{i,j}^f + w_\alpha \\ \text{for } i = n\left(t_{p,\alpha} - 1\right) + i_{p,\alpha} + kn, \\ \qquad j = i + (k+1)\,n,\ k = 0, \ldots, hm - t_{p,\alpha}, \\ a_{i,j}^f - w_\alpha \\ \text{for } i = j + (k+1)\,n,\ j = n\left(t_{p,\alpha} - 1\right) + i_{p,\alpha} + kn, \\ \quad k = 0, \ldots, hm - t_{p,\alpha}. \end{cases} \tag{14}$$

elsewhere

(iii) Reject the booking r_α.

(3.4)′ Set $\alpha = \alpha + 1$ and return to Step 3.

Step 4′. We finish when all the bookings have been processed.

7. Some Additional Considerations about the Model

In our assumptions we have considered that we start the scheduling of bookings from scratch starting on time t_1, but this is only the real case once in the company lifetime. The interval of time $t_1, \ldots, t_{hm}$ must be finite. Nevertheless, we can consider that before running the algorithm on the model we have already confirmed some bookings that started before t_1 and some that will finish after t_{hm}. In order to introduce them in our model:

(i) for every existing booking (i_p, t_p, i_d, t_d, w) with $t_p < t_1$ and $t_1 \le t_d \le t_{hm}$, we add a new edge (s, v_{i_d,t_d}) with maximum and minimum capacity equal to w,

(ii) for every existing booking (i_p, t_p, i_d, t_d, w) with $t_1 \le t_p \le t_{hm}$ and $t_{hm} < t_d$, we add a new edge (v_{i_p,t_p}, t) with maximum and minimum capacity equal to w.

We have also considered that we are dealing with only one type of car. If our fleet has several types of cars ordered in increasing category as $x_1 \prec x_2 \prec \cdots \prec x_l$, we will propose a model with l-layers. In the lower one we have a network for cars of type x_1, in the one above of it we deal with the cars of type x_2, and so on. We should point out that in this case the number of parking spaces is shared by several types of cars. Sometimes the clients could ask for a car of low type, for instance x_1, and the company could have none of those cars available of this layer for renting. However, we can try to see if we can offer him a car of type x_2. If this is the case, we can confirm the booking and grant the client with an upgrade when he arrives to the pick-up depot. The client will be happy and the service have accepted a booking that otherwise could be lost.

8. Results and Discussion

The proposed model simulates the flow of the fleet of vehicles of a car-rental service without staff dedicated to move cars from one location to another one, as could sometimes be needed to do due to one-way bookings. This model has the advantage that let us recover the flow of cars across the depots along the time, as it is indicated in Remark 6.

The model is based on determining the admissibility of a flow on a network with minimum capacities. Its implementation is backed by Theorem 3. In our first model every new booking only requires defining at most one new edge on the auxiliary network and applying the well-known Ford-Fulkerson algorithm on it. Later, a simplification of the network permits us to solve the problem using a shortest path algorithm.

On one hand, Ford-Fulkerson algorithm can be easily computed on a network with *Mathematica*© by using the command *NetworkFlow* [*network, source, sink*] of the *Combinatorica* package. In our case, *network* would be the graph associated with N', s' would be the source, and t' would be the sink. The complexity of Ford-Fulkerson algorithm is $\mathcal{O}(|\mathcal{E}| \cdot f(N'))$, where $\mathcal{E}$ is the set of edges in the network N' and $f(N')$ is the maximum flow on it, see for instance [19,

Table 1: Comparison between the 2 algorithms for 6 cities at 20 different times.

Number of bookings	With NetworkFlow (s.)	With FindShortestPath (s.)
1	0.125	0.062
5	0.718	0.266
10	1.828	0.515
15	3.265	0.782
30	9.235	1.562
75	34.609	4.078
125	61.547	6.765
200	100.265	11.00
300	139.593	16.657
400	163.484	22.422
500	207.984	28.281

20, 26]. Therefore, every time a booking $r = (i_p, t_p, i_d, t_d, w)$ is accepted we add at most two more new edges and the flow is increased in w, which is the number of cars in the booking. An implementation using Edmonds-Karp algorithm has a complexity $\mathcal{O}(|V| \cdot |\mathcal{E}|^2)$, which is independent of the number of cars considered and only depends on the set of vertex V' and the set of edges $\mathcal{E}$, [19, 20, 26].

On the other hand, the shortest path between two nodes in a network can be easily computed with *Mathematica* using *FindShortestPath* [*network, source, sink*] of the *Combinatorica* package, with the same values for *network, source*, and *sink* as before. There are several algorithms for solving the shortest path problem between two points. The well-known Dijkstra algorithm has a cost $\mathcal{O}(|V|^2)$ when it is implemented with lists. The order of the cost can be reduced if it is implemented with binary heaps, $\mathcal{O}((|\mathcal{E}|) \log |V|)$, or with Fibonacci heaps, $\mathcal{O}(|\mathcal{E}| + |V| \log |V|)$, see [19, 20, 26]. Therefore, the simplification proposed of the algorithm shown in Section 6 can provide much more better computational results.

This can be seen on a simulation carried out over time-expanded network of 6 depots at 20 different times and 500 bookings. The computations were performed on a computer with a processor of 2.50 GHz and 3.25 Gb of RAM using *Mathematica*© *8.0*. The comparison of the execution times of each one of these methods is presented in Table 1. We observe a significant improvement in the computational time required to solve the problem if we use the simplified algorithm based on the search of shortest paths using the residual network.

Finally, this model let us simulate if it is profitable for the company that manages the service to invest or not, either in more cars or in more parking slots, taking into account the historic of petitions of bookings received. This can be done using a list of all the requests of bookings received during a period of time, despite they were accepted or not. We analyze, with the historical data of petitions, if when we increase the maximum and minimum capacities at one or some of the edges $(s, v_{i,1})$ the number of accepted bookings is increased, and whether the new expected incomes coming from these bookings are enough to invite us to invest. Analogously, we can consider if there is a significant increment of benefits if we augment the number of parking spaces at one or some depots. Both types of analysis can be jointly considered.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

J. Alberto Conejerois is supported by MEC Project MTM2013-47093-P. The authors thank Víctor Fernández and Raúl Urbano from the Mobincity Project for their helpful discussions on the topic. Esther Sanabria-Codesal is supported by MEC Project MTM2012-33073.

References

[1] A. Fink and T. Reiners, "Modeling and solving the short-term car rental logistics problem," *Transportation Research: E*, vol. 42, pp. 272–292, 2006.

[2] H2020, Workprogramme 2014-2015, "Objective GV.8-2015: Electric vehicles' enhanced performance & integration into the transport system & the grid," http://ec.europa.eu/research/participants/portal/desktop/en/opportunities/h2020/topics/2608-gv-8-2015.

[3] The Mobincity Project: Smart Mobility in Smart city, http://www.mobincity.eu/.

[4] European Vehicle Market Statistics, Pocketbook 2013, 2014, http://www.theicct.org/sites/default/files/publications/EU_vehiclemarket_pocketbook_2013_Web.pdf.

[5] I. Steizen, V. Gintner, L. Suhl, and N. Kliewer, "Electric vehicles: charging into the future," *German Institute For Economic Research*, vol. 6, no. 27, 2010.

[6] J. Abara, "Applying integer linear programming to the fleet assignment problem," *Interfaces*, vol. 19, no. 4, pp. 20–28, 1989.

[7] D. Teodorovic, *Airline Operation Research*, Gordon and Breach Science, New York, NY, USA, 1988.

[8] S. Yan, C. H. Tang, and M. C. Lee, "A flight scheduling model for Taiwan airlines under market competitions," *Omega*, vol. 35, pp. 175–186, 2007.

[9] S. Yan and C. H. Tseng, "A passenger demand model for airline flight scheduling and fleet routing," *Computers and Operations Research*, vol. 29, pp. 1559–1581, 2002.

[10] M. Lohatepanont and C. Barnhart, "Airline schedule planning: integrated models and algorithms for schedule design and fleet assignment," *Transportation Science*, vol. 38, pp. 19–32, 2004.

[11] I. Steizen, V. Gintner, L. Suhl, and N. Kliewer, "A time-space network approach for the integrated vehicle-and crew-scheduling with multiple depots," *Transportation Science*, vol. 44, pp. 367–382, 2010.

[12] S. Yan and C. H. Chen, "Optimal flight scheduling models for cargo airlines under alliances," *Journal of Scheduling*, vol. 11, no. 3, pp. 175–186, 2008.

[13] A. T. Ernst, E. O. Gavriliouk, and L. Marquez, "An efficient Lagrangean heuristic for rental vehicle scheduling," *Computers & Operations Research*, vol. 38, no. 1, pp. 216–226, 2011.

[14] A. Hertz, D. Schindl, and N. Zufferey, "A solution method for a car fleet management problem with maintenance constraints," *Journal of Heuristics*, vol. 15, pp. 425–450, 2009.

[15] Y. Yang, W. Jin, and X. Hao, "Car rental logistics problem: a review of literature," *IEEE International Conference on Service Operations and Logistics, and Informatics*, vol. 2, pp. 2815–2819, 2008.

[16] A. Dolan and J. Aldous, *Networks and Algorithms: An Introductory Approach*, John Wiley & Sons, Chichester, UK, 1993.

[17] J. A. Conejero and C. Jordán, "Aplicaciones de la Teoría de Grafos a la vida real," Massive Open Online Course (MOOC) offered by UPV[x], 2014, http://cursografos.upvx.es/ficha.

[18] J. R. Evans and E. Minieka, *Optimization Algorithms for Networks and Graphs*, Marcel Dekker, New York, NY, USA, 1992.

[19] J. L. Gross and J. Yellen, *Graph Theory and Its Applications*, Discrete Mathematics and its Applications (Boca Raton), Chapman & Hall/CRC, New York, NY, USA, 2006.

[20] J. L. Gross and J. Yellen, *Handbook of Graph Theory*, Chapman & Hall/CRC, New York, NY, USA, 2003.

[21] M. C. Hernández-Ayuso, *Introducción a la teoría de redes*, Sociedad Matemática Mexicana, Tlalpan, México, 2005.

[22] C. Jordán, "Estructuras Matemáaticas para la Informáatica II Open Course Ware (OCW)," (Look at Información & Materiales) http://www.upv.es/ocwasi/2010/6024.

[23] C. Jordán and J. R. Torregrosa, "Herramientas de la teoría de grafos para la modelización," *Modelling in Science Education and Learning*, vol. 4, no. 22, pp. 275–289, 2011.

[24] C. Jordán and J. R. Torregrosa, *Introducción a la teoría de grafos y sus algoritmos*, Universitat Politècnica de València, Valencia, Spain, 1996.

[25] L. Bodin, B. Golden, A. Assad, and M. Ball, "Routing and scheduling of vehicles and crews: the state of the art," *Computers & Operations Research*, vol. 10, no. 2, pp. 63–211, 1983.

[26] S. Pemmaraju and S. Skiena, *Computational Discrete Mathematics*, Cambridge University Press, Cambridge, UK, 2003.