Journal of the Operational Research Society (2010) 61, 1133–1143

www.palgrave-journals.com/jors/

Dynamic scheduling of recreational rental vehicles with revenue management extensions

AT Ernst[1], M Horn[2]*, P Kilby[3] and M Krishnamoorthy[4]

[1]CSIRO Mathematical and Information Sciences, Victoria, Australia; [2]CSIRO Mathematical and Information Sciences, New South Wales, Australia; [3]National ICT Australia, Australian Capital Territory, Australia; and [4]Monash University, Victoria, Australia

The rental fleet scheduling problem (RFSP) arises in vehicle-rental operations that offer a wide variety of vehicle types to customers, and allow a rented vehicle to 'migrate' to a setdown depot other than the pickup depot. When there is a shortage of vehicles of a particular type at a depot, vehicles may be relocated to that depot, or vehicles of similar types may be substituted. The RFSP involves assigning vehicles to rentals so as to minimise the costs of these operations, and arises in both static and online contexts. The authors have adapted a well-known assignment algorithm for application in the online context. In addition, a network-flow algorithm with more comprehensive coverage of problem conditions is used to investigate the determination of rental pricing using revenue management principles. The paper concludes with an outline of the algorithms' use in supporting the operations of a large recreational vehicle rental company.

Journal of the Operational Research Society (2010) **61,** 1133–1143. doi:10.1057/jors.2009.78
Published online 29 July 2009

Keywords: scheduling; networks and graphs; revenue management; vehicle routing

Introduction

This paper reports on a study of rental operations in which vehicles of different types are made available for hire from several depots to retail customers. A customer's request for a rental includes the time and the depot where a vehicle is to be picked up, and the time and the depot where the vehicle is to be returned to the rental company. Before accepting a rental request, the company must be confident that an acceptable vehicle will be available at the pickup depot, and this feasibility must be maintained until the requested pickup time. More broadly, the rental company will wish to assign vehicles to rentals in an efficient manner, while maintaining a reliable service to customers and orderly management of related activities such as vehicle maintenance.

Tourism Holdings Limited (THL) is the leading company in the recreational rentals market in Australia and New Zealand. In Australia, THL maintains depots in 10 cities and a fleet comprising several thousand vehicles—four-door cars, sport utility vehicles, and campervans of various sizes. THL's business is characterised by long booking horizons, long booking durations, a diverse range of vehicle types, and a policy that allows return of a vehicle at a depot other than the pickup point at no extra cost. These conditions imply a much more complex scheduling task than in conventional rental operations (eg where vehicles must be returned to their origin depots), and they have provided the main practical motivation for the research described in this paper.

In THL's rental operations, a *fleet schedule* is a plan for the deployment of each vehicle in the fleet so as to honour all currently booked rentals, with provision for vehicle-maintenance activities and changes in fleet composition. A schedule may include adjustments called *substitutions* and *relocations* that in effect extend the fleet's capacity in response to local peaks of demand. In both cases, the direct costs are borne by the company, not by the customer.

In a *substitution*, a rental is planned with a vehicle type different to the one requested by the customer. Associated with each rental is a *product*, specifying a preferred vehicle type (ie make and model), a set of alternative types, and other attributes. Substitution replaces the preferred type with one of the alternatives, usually of higher value.

A *relocation* shifts a vehicle from one location to another in order to satisfy a planned (or anticipated) rental pickup at the destination. Relocations help to ensure the availability of vehicles at the places and times they are needed, but obviously can be expensive for the company. Relocations of differing durations may be available, with shorter times at greater cost.

Details that need to be considered in practice include vehicle *turnaround*, where a vehicle is cleaned and undergoes mechanical servicing before being passed to the next customer. In addition, there may be times when vehicles are unavailable for longer periods of time in order to undergo

*Correspondence: M Horn, Locked Bag 17, North Ryde, NSW 1670, Australia.
E-mail: mark.horn@csiro.au

maintenance, or where part of the fleet is turned over with *disposal* of old vehicles and arrival of new vehicles.

The main scheduling objective is to minimise the costs associated with substitutions and relocations. We call this the *rental-fleet scheduling problem* (RFSP). The static problem—solved each day—is to find a solution to the RFSP. A critical *dynamic* form of the problem is posed when a response to a customer's rental request is required in real time.

The problem as outlined above is similar in several respects to that addressed by Hertz *et al* (2009), which was formulated as a test problem by the French Society for Operational Research (ROADEF). Like the RFSP considered here, the ROADEF problem is concerned with the management of a vehicle rental fleet, with provision for substitutions. The ROADEF formulation also incorporates maintenance scheduling, subcontracting, and vehicle purchase decisions, but makes no reference to location, and thus excludes provision for relocations. Hertz *et al* describe a combination of tabu search and other techniques for the ROADEF problem. These methods appear to be very effective for the problems tested, which however are considered only in a static context, and are much smaller than those addressed here (eg with 80 rather than 6000 or more bookings in the schedule).

Fink and Reiners (2006) discuss the management of vehicle inventory by large European rental companies. They present a decision-support framework, together with network-flow based optimisation procedures that provide for relocation between depots, but they do not seek the operational capability (eg tracking of particular vehicles) developed in the present research, and their time horizon is very short (1 week). Pachion *et al* (2003) describe vehicle-rental operations as seen in the United States, including 'pooling' arrangements within which relocations can be made. These arrangements form the basis of a tiered decision framework, by contrast with the unified approach taken here.

Parallels with the RFSP may be seen in the pickup and delivery problem (PDP), where a fleet of vehicles is deployed to handle a set of transport requests between designated pickup and setdown points (Dumas *et al*, 1991; Ioachim *et al*, 1995; Savelsbergh and Sol, 1998; Currie and Salhi, 2003). Related to the PDP are the multi-depot vehicle scheduling problems (MDVSP) discussed by Desaulniers *et al* (1998). There are significant differences between these problems and the RFSP, notably the PDP's assumption of a homogeneous fleet (heterogeneous in the RFSP); the short time-horizon in the PDP compared with a horizon of around a year in the RFSP; and the combination of few depots and many other locations in the PDP, as opposed to the larger number of depots and an absence of non-depot locations in the RFSP. The present research has however made use of some of the network-flow concepts developed for the MDVSP.

There are parallels also in the scheduling of airline operations, where the RFSP can be viewed as encompassing both fleet assignment and aircraft rotation (also called 'tail assignment' or 'aircraft routing') (Clarke *et al*, 1997; Gopalan and Talluri, 1998; Ahmed and Poojari, 2008). Again, these precedents have been instructive but have not foreclosed the need for substantial fresh thinking for the RFSP (Ernst *et al*, 2007a).

It is apparent then that the RFSP embodies complexities which are not encountered in conventional vehicle-rental operations, and have not been addressed directly in the previous research literature. This paper presents two approaches to the problem, namely network-flow and assignment models. Although the two approaches are essentially equivalent, an exposition of this kind is useful in demonstrating the different strengths of the two formulations. A well-known algorithm for the assignment model has particular advantages in the dynamic scheduling context, where there is a need to maintain a near-optimal solution under strict computational time restrictions. The network-flow formulation is more comprehensive than the assignment model, and is well-suited to application in the static context.

The network-flow model has proved useful also as a basis for applying Revenue management (RM) ideas to the vehicle-rental scheduling business. Revenue management (also called Yield Management) involves setting prices in response to expected demand, and is particularly relevant in businesses—notably in the tourism and travel sectors—where there are substantial fixed costs and each 'product' is instantiated within a limited period of time (Botimer and Belobaba, 1999; Yeoman *et al*, 1999). This description matches quite closely the conditions of the recreational rentals business, so that there is at least a *prima facie* case for the investigation of RM undertaken here.

It should be noted that for the assignment and network-flow approaches discussed above this paper gives two separate problem formulations, which are equivalent except for their treatment of substitutions. For further details (eg with respect to delays and expected profits), see Ernst *et al* (2007b).

The remainder of the paper is set out as follows. We formulate the RFSP as an assignment problem considered in both static and dynamic contexts; we develop a network-flow formulation of the RFSP; and we report on simulation tests performed on the assignment and network-flow models. We then show how the network-flow formulation can be adapted for RM purposes, and report on a set of tests devised to validate this approach. We conclude with an outline of the implementation of the algorithms and the benefits they have delivered to THL.

An assignment approach

We commence with an assignment model that fully addresses the RFSP. After presenting the model and defining a solution procedure we will focus on its use in a dynamic context, where a new rental (or an event such as a vehicle breakdown) is to be incorporated in the current schedule, the latter having been constructed initially by a static algorithm.

As a basis for the assignment model we partition the company's vehicles into sub-fleets, one for each vehicle type,

and associate each rental with the sub-fleet corresponding to its preferred vehicle type. The scheduling task then involves two parts: assigning rentals to vehicles efficiently in each sub-fleet, and (for substitutions) providing for transfers of rentals between fleets. Note, however, that the formulation as presented here omits the substitution stage. That is, given the breakdown of work between static and dynamic models outlined above, all substitution decisions are made by the static algorithm and are preseved in the assignment solution.

Problem formulation

We define sets I and J with $|I|=|J|$, representing rentals, maintenance sessions, and other activities. Set I represents the end-time-slots and locations of all activities, while set J represents the corresponding start-time-slots and locations, and each time-slot is defined in practice as the morning or afternoon of a given calendar day. An assignment (i, j) $(i \in I,\ j \in J)$ indicates that i and j appear as a successive pair in the list of activities to be carried out by a particular vehicle; that is, the schedule proceeds from an 'end-event' i to a 'start-event' j. The period between the end of an activity i and the start of another activity j may include a turnaround and a relocation to ensure that the vehicle is ready and in the right location to commence j.

The sequence of activities assigned to a vehicle v commences with a start-activity $first_v \in I$, indicating the time and place when the vehicle first becomes available, and is terminated by a notional end-activity $last_v \in J$ (or a generic *last*), indicating when the vehicle becomes idle or is to be taken out of service. Thus the sequence of rentals and other activities for vehicle v is $first_v \rightarrow i,\ i \rightarrow j \ldots k \rightarrow last_v$. To ensure that a sequence is feasible (eg with chronological timings to avoid cycles), we define a set of feasible activity-pairs $A \subset I \times J$.

For each $i \in I$ and $j \in J$ there is an assignment cost c_{ij}. A comprehensive statement of costs is given by Ernst *et al* (2007b). In summary, the main potential costs are a relocation component (ie the cost of shifting the vehicle used by i from the location of i to the location of j); a penalty for violation of the commencement time associated with j; and, in principle (see below), a substitution cost if the vehicle used by j is not a preferred type for j.

The task then is to find a set of assignments $I \times J$, pairing each $i \in I$ with a $j \in J$ so as to minimise total costs. Let $x_{ij} = 1$ if assignment $i \rightarrow j$ is in the solution, 0 otherwise. The scheduling problem can then be stated as follows.

$$\text{Minimize} \quad \sum_{i \in I} \sum_{j \in J} x_{ij} c_{ij} \tag{1}$$

$$\text{Subject to} \quad \sum_{i:(i,\,j) \in A} x_{ij} = 1 \qquad \forall j \in J \tag{2}$$

$$\sum_{j:(i,\,j) \in A} x_{ij} = 1 \qquad \forall i \in J \tag{3}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i \in I,\ j \in J \tag{4}$$

Equations (1)–(4) define a well-known assignment problem (Dell'Amico and Toth, 1999). With arbitrarily high self-assignment costs (ie $c_{ii} = \infty\ \forall i \in I \cup J$), an $x_{ii} = 1$ in the optimal solution indicates that activity i cannot be scheduled with the available vehicle fleet. Note that this formulation requires vehicles of each type to be indistinguishable; for example, an activity cannot be assigned in advance to a particular vehicle, and there is no provision for individual vehicles to terminate at specific locations at the end of the scheduling period.

The complete problem across all vehicle types can be specified as a set of assignment problems with interlinking constraints. Let $x^t_{ij} = 1$ indicate that the end of activity i will follow the start of activity j as serviced by a vehicle of type $t \in T$, where T is the set of vehicle types.

We then have a block of constraints for each t of the form (1)–(4), linked by constraints of the form

$$\sum_{t \in T} x^t_{ii} = |T| - 1 \qquad \forall i \in I \cap J \tag{5}$$

Equation (5) requires that each activity can be assigned to a single vehicle type $t \in T$: for every other type $t' \in T$, $t' \neq t$, $x^{t'}_{ii} = 1$ indicates that activity i is not serviced by t'. We ensure this in practice by solving the assignment problem separately for each vehicle type, with each rental activity assigned to the most-preferred vehicle type for the rental. Thus the assignment procedures at present do not handle vehicle substitutions, which are delayed instead for periodic re-optimisation by the static algorithm (see also the following account of schedule repair and improvement).

Assignment algorithms

There are several algorithms for solving assignment problems of the type defined in (1)–(4). For our purposes the successive shortest-path algorithm of Engquist (1982) is particularly useful. Engquist's algorithm begins by assigning every activity i to its least-cost successor j, ignoring the requirement Equation (3), that each j should have exactly one predecessor. In this initial assignment, normally some j are *deficient* (with no predecessor assigned) and some *abundant* (with more than one predecessor assigned). A *dummy* activity is added, with an assignment to one of the deficient activities (the choice in this respect is arbitrary). The dummy is a notational convenience to provide uniformity in the treatment of activities.

The main body of the algorithm involves iteration of a *reassignment step*. In this step the current set of assignments is treated as a *meta-graph*, with a *meta-node* representing each current assignment. The arcs of the meta-graph indicate possible changes to the activity-assignments represented by the meta-nodes; in particular, an arc from $(i \rightarrow j)$ to $(m \rightarrow n)$ means 'delete assignment $(i \rightarrow j)$ and replace it with assignment $(m \rightarrow j)$'. An arc from $(i \rightarrow j)$ to $(m \rightarrow n)$ exists if $(m \rightarrow j)$ is permitted, and has a cost of $c_{mj} - c_{ij}$.

The focus for the reassignment step is the tree of shortest paths from the dummy meta-node to the abundant meta-nodes. The least-cost path in the tree is chosen, and the changes specified by the chosen path are then applied to the schedule. The reassignment step is repeated until no more abundant nodes remain. A conclusion is guaranteed because at each iteration, exactly one deficient node receives an assignment, reducing the number of abundant nodes by one. Engquist (1982) shows that the solution obtained in this way is optimal.

The operation of Engquist's algorithm is illustrated by a problem with $I = \{a, b, c, d, e\}$ and $J = \{A, B, C, D, E\}$. Suppose that the meta-nodes are initially as follows;

$$a \to B$$

$$b \to A$$

$$c \to A$$

$$d \to C$$

$$e \to B$$

Here, A and B are abundant, while D and E are deficient. A dummy activity is assigned to the deficient node E, and this assignment is represented by an additional meta-node $(dummy \to E)$. Now suppose that the least-cost path from the dummy activity is $(dummy \to E) \Rightarrow (c \to A) \Rightarrow (b \to A)$. Making the changes implied by this path replaces the corresponding assignments with $(c \to E)$ and $(b \to A)$. The result (see below) leaves only A abundant, and only D deficient.

$$a \to B$$

$$b \to A$$

$$c \to E$$

$$d \to C$$

$$e \to B$$

An assignment algorithm for the static RFSP commences with an empty schedule, that is with $x_{ii} = 1$ for all activities. Now instead of reducing deficiency and abundance in each iteration, we now try to find a shortest path for any unscheduled activity. It is easy to show that this also leads to an optimal solution of the assignment problem.

Adding rental bookings

We have adapted the algorithm outlined above for the online task of adding a rental to the schedule. A rental query is handled by using the same technique to accommodate the requested rental in the schedule (if possible), and then discarding the adjusted schedule.

The task is to find an efficient assignment for a rental-request r for preferred vehicle type t in the sub-schedule for V_t. We refer to the existing successor of an activity i as S_i. An arc $(i \to S_i) \Rightarrow (j \to S_j)$ in the meta graph is legal only if S_i is a feasible successor for j in terms of timing and other conditions. To commence, we define two dummy meta-nodes $(dummy \to r)$, $(r \to dummy)$. We then find the least-cost path from $(dummy \to r)$ to $(r \to dummy)$, as indicated earlier. This path represents an optimal insertion of r in the sub-schedule for V_t.

The above technique can lead to radical changes in the schedule, and is well suited to an operation with limited execution time. The paths originating at $(dummy \to r)$ are explored in a breadth-first manner; that is, all paths of length 2 are explored before moving on to paths of length 3, and so on. Because any path from $(dummy \to r)$ to $(r \to dummy)$ represents a feasible insertion, the algorithm can return a useful result—indicating for example that r is feasible—if it is terminated after a limited time (eg with a time-limit for validating each rental request).

One consequence of a time-limited search is that the implementation found for a rental request may not be fully optimal, which is quite acceptable in practice. However, if a sub-schedule thus becomes sub-optimal, there will be cycles in the corresponding meta-network, each cycle defining a path from an activity back to itself, with a negative cost. The possibility of negative-cost cycles precludes the use of some standard shortest-path algorithms (eg the Dijkstra algorithm (Dijkstra, 1959)). Instead, we have used the Bellman–Ford algorithm (Bellman, 1958; Ford and Fulkerson, 1962), with the implementation described by Cormen *et al* (1989).

Repairing and improving the schedule

Sub-optimality may be introduced in a schedule due to a limitation on execution time as indicated above, or through real-time contingencies, such as breakdowns or late returns to depot. Techniques are described below for restoring optimality, through the use of *improvement* and *repair* procedures. Both procedures make use of the fact that a potential improvement in the schedule is manifested as a negative-cost cycle in the meta-network. The *improvement* procedure can be invoked at any time for a given vehicle type t. The meta-network corresponding with the sub-schedule for t is constructed, and an existing assignment is chosen at random as the root for a shortest-path tree. The procedure begins constructing the tree, and when it discovers a negative-cost cycle, the cycle is implemented and the procedure is restarted for t. If no cycles are found, processing can re-start for another vehicle type t'. This procedure obviously is very well suited for implementation as a continuous 'background' process. Further improvement might be obtained through a higher-level heuristic procedure allowing transfer of rentals between sub-schedules, so as to implement substitutions. In practice, however, this is not required because substitutions are handled very effectively in the periodic re-optimisation of the schedule using the network flow procedure described in the following section.

The *repair* procedure is invoked after a disruption, such as the late return of a vehicle, which may introduce delays and other inefficiencies in the schedule. Again the meta-network is created for the relevant vehicle-type t, but the root of the shortest path is defined as an activity close to the disruption instead of the one chosen at random. In this way the procedure focuses on the most severely affected parts of the meta-network, and useful repairs are more likely to be discovered.

A network-flow approach

A network-flow formulation provides an alternative to the assignment approach given above. As before, time is represented as a succession of morning and afternoon time-slots, and scheduling is limited initially to a single vehicle type. We use a time-layered network in which each node represents a particular location at a particular time, identified by a (location time) pair. We represent each possible activity that a vehicle can perform as an arc, with a predefined cost and an upper bound on flow, that is, on the number of vehicles carrying out the activity associated with the arc. For example, a *rental arc* connects the two nodes representing the start location/time and end location/time for a rental, and has an upper bound on flow of one. An *idle arc* represents a period when a vehicle stands idle, and connects nodes representing the same location on successive dates, with zero cost and no upper bound on flow. Activities such as relocations and turnarounds are represented as arcs in a similar fashion.

We denote the set of all nodes as $\mathcal{N}$, the set of all arcs as $\mathcal{A}$, the idle arcs as $\mathcal{I}$, and the origin and destination nodes of an arc a as $\alpha(a)$ and $\omega(a)$, respectively. Each node n has a supply S_n, which is positive if there are vehicles available at this location/time, or negative for location/times where vehicles are taken off the fleet, and with all other nodes having $S_n = 0$. A dummy sink node is introduced to take all remaining flow at the end of the planning horizon (an alternative would be to pre-specify the number of vehicles that are to end up at each location). Let x_a represent the flow on arc a, and let C_a represent the cost of the activity represented by a. The RFSP can then be formulated as a network-flow problem of the form

$$\text{Minimise} \quad \sum_{a \in \mathcal{A} \setminus \mathcal{I}} C_a x_a \tag{6}$$

$$\text{Subject to} \quad \sum_{a \in \mathcal{A}:\alpha(a)=n} x_a - \sum_{a \in \mathcal{A}:\omega(a)=n} x_a = S_n \quad \forall n \in \mathcal{N} \tag{7}$$

$$0 \leqslant x_a \leqslant u_a \qquad \forall a \in \mathcal{A} \tag{8}$$

This can be extended to incorporate multiple vehicle types by using a multi-commodity flow formulation. Denoting the set of vehicle types again as T, and the flow of vehicle type $t \in T$ on arc a as x_a^t, we supplement Equations (6)–(8) with linking constraints:

$$\sum_{t \in T} x_a^t \leqslant 1 \qquad \forall \text{ arcs } a \text{ representing rentals} \tag{9}$$

In the present research, the problem defined above is solved—with a very low incidence of fractional variables—using CPLEX, a general purpose Integer Linear Programming (ILP) solver (an alternative would be to use a specialised multi-commodity flow solver). To construct a schedule for individual vehicles from the optimal network flow, it is necessary then to trace a path through the network for each vehicle. In a simple greedy heuristic, the path is traced from the vehicle's start location to any sink node (or a particular sink node, as discussed below), along arcs that carry positive flow: the residual flow is then updated along the path chosen for the vehicle, and the procedure is repeated for each remaining vehicle in the fleet. If this is possible for each vehicle, the solution is optimal; in fact, when the vehicles are indistinguishable it is always possible to create a path for every vehicle from the remaining arcs in the residual network. However, when some vehicles have fixed destinations (eg for maintenance or pre-assigned bookings), then it may not be possible to find such a path. Our practice is to handle these fixed-destination vehicles (FDVs) first, in each case finding and fixing in place a shortest path through the full network, but using arcs in the residual network where possible. Once all schedules for FDVs are fixed, the ILP is re-solved, and the schedules for all other vehicles extracted from the residual network are determined by the new solution. In practice this produces very good solutions because most vehicles do not have fixed destinations.

While the network-flow approach is a little more complicated than the assignment formulation in some respects, it has two important advantages. Firstly, because there are generally multiple different ways of extracting individual vehicle schedules from the network-flow solution, it may be possible to deal with the issue of a small number of FDVs without explicitly including this condition in the formulation. An alternative is to create a separate network-flow/assignment problem not just for each vehicle type, but also for each vehicle end point location and end-time pair. However, this *significantly* increases the overall formulation size for only very limited improvement in solution quality.

The second advantage of the network-flow approach is that it generally requires fewer variables than the assignment formulation. For example, if there are 1000 rentals to be scheduled on 500 vehicles without fixed destinations, the assignment formulation implies 500 000 arcs connecting the end-node of each rental ($last_i \in I$) to the end-node of each vehicle ($last_j \in J$). By comparison, the network-flow formulation for a problem of the same size will normally require fewer than 10 000 arcs; for example, with 10 locations and a 1-year scheduling horizon, there are 1000 rental arcs, 3650 idle arcs, plus a few thousand more arcs representing relocations and turnarounds. While the large number of arcs in the assignment formulation poses no difficulty for the dynamic rescheduling task—where generally only a small part of the overall problem is considered—the complete assignment problem

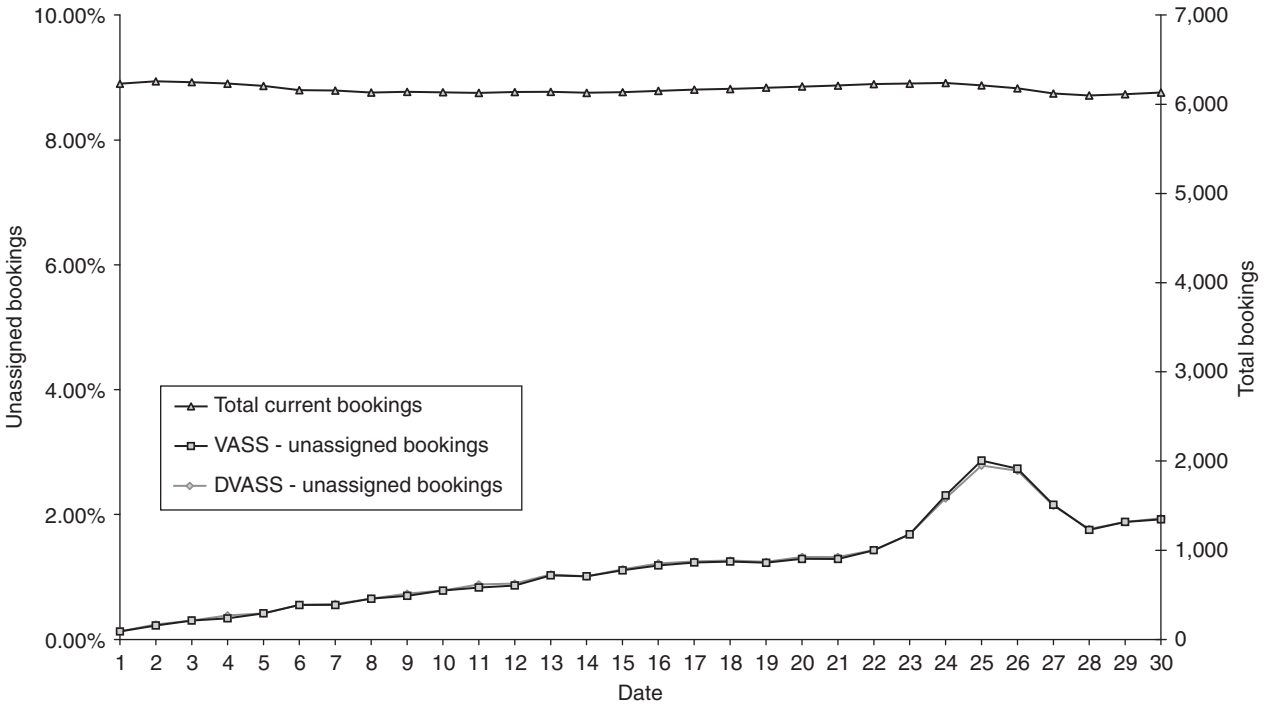

Figure 1 Simulation tests: current and unassigned bookings.

would be too large to be solved using currently available ILP solvers.

Computational tests

The assignment and network-flow algorithms described earlier in this paper are embodied in software modules called, respectively, D-VASS and VASS. The latter solves the RFSP to create an optimised schedule during an overnight run, while the former dynamically updates the schedule whenever a change is made during the day. Both modules have been tested extensively through simulation tests involving different assumptions with regard to aggregate demand, fleet size, and so on. The results of the tests are broadly consistent, and for brevity we shall focus mainly on a single test run whose scale and complexity broadly reflect conditions encountered in practice. This involves 20 pickup locations and 2123 vehicles of 52 different types, marketed as 336 different products. The simulation commenced with 6168 rentals, with random attributes, which were scheduled by an initial run of the static VASS optimisation procedure.

From the initial setup described above, scheduling operations were modelled over a period of 31 days, with incoming rental requests randomised with respect to time of issue, desired product, rental date and duration, and pickup and setdown locations. Excluded from the simulation were removals and additions to the vehicle fleet, and unexpected events such as late returns (effective handling of these events has been confirmed in other simulations and through several years' practical application). To obtain an approximately uniform load on the system over the course of the simulation period, 63 new rental requests were modelled per day, approximately matching the rate at which rentals were dispatched. By this means the total number of bookings in the system was maintained at a fairly steady level, as indicated in Figure 1 (the range is between 6100 at the end of day 30, and 6258 at the end of day 4).

As previously indicated, D-VASS is the module that seeks to incorporate online requests into the current schedule; while VASS is a static optimiser which is run late every night. To compare the performance of the two algorithms, we have considered revenue and costs at the end of each simulated day: for D-VASS, after the last rental request or improvement session of the day, and for VASS, just after its evening run.

The performance indicators reported here should be read in the light of a subtle difference in the way optimisation is carried out by the two modules. For D-VASS, the decision as to whether a rental is accepted initially is normally determined by whether the rental can be fitted into the current schedule. However, once a rental has been accepted, it must be retained in the schedule at all cost, even if unforeseen conditions (eg due to a late return of a vehicle by another customer) make the original assignment no longer feasible. In order to stress-test the optimisation, we forced D-VASS to accept all rentals, which may include some for which it cannot find feasible assignments; that is, D-VASS's

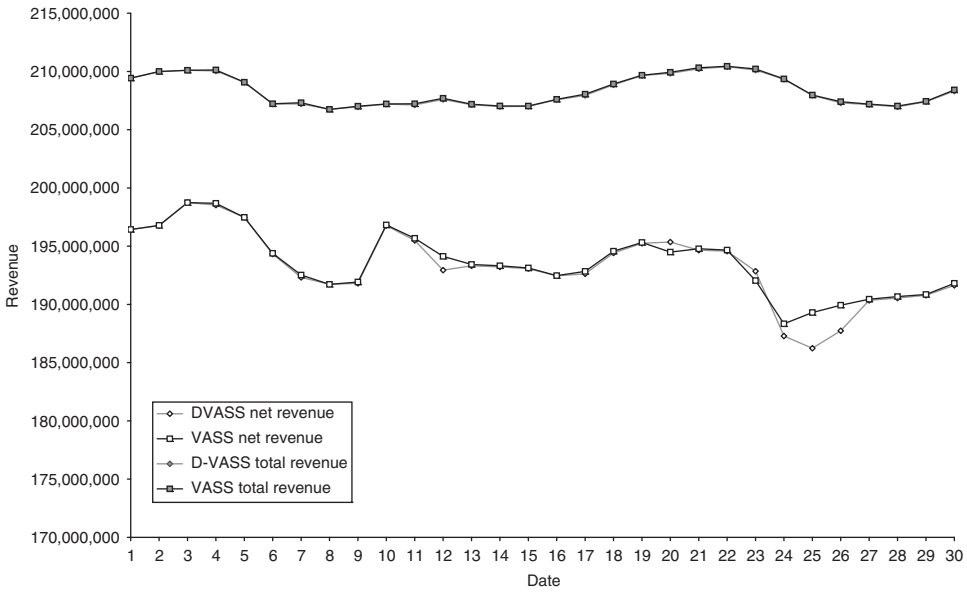

Figure 2 Simulation tests: total and net revenue.

task here was to fit the rentals into the schedule at the least possible incremental cost.

VASS on the other hand takes a global view: if it is not possible to schedule all rentals, VASS tries to accommodate as many as possible, dropping those of least value. This 'value' is based on how much the customer is paying, with a discount on rentals that occur late in the planning period. The discount is applied because later rentals are more likely to be accommodated through future changes to the schedule, and hence present a less severe problem in practice than earlier unscheduled rentals. The objectives applied by D-VASS and VASS are thus not quite identical when there are some unscheduled rentals.

As indicated in Figure 1, the number of unassigned rentals in the simulation runs is small (average 1.2%, at most 2.8%), and differs very little between D-VASS and VASS. An interesting aspect of these results is the apparent overall increase in scheduling difficulty over the course of the simulation period, as indicated by the general upward trend in the numbers of unassigned rentals, with a marked peak in this respect between days 22 and 28. This pattern is reflected also in the computational times (Figure 3), and to some extent in the revenue plots (Figure 2), where there is an inverse dip corresponding with the peak in unassigned rentals.

Total and net revenues are shown in Figure 2; note that the revenues and costs shown here are provided for illustrated purposes only and bear no relation to dollar values used by THL. Because of the close relationship between revenue and number of assigned rentals, the differences between the revenue obtained from the two modules were less than 0.5% in each case. The differences with respect to net revenue—defined as revenue minus costs—are somewhat greater than for raw revenue, due to differences in costs. Here the results from VASS are superior in nearly every case to those from D-VASS, as one would expect, with significant exceptions only at days 20 and 23. These apparent anomalies can be explained by the differences between the objective functions applied in the two algorithms, as discussed above. Overall, the most interesting result here is the ability of D-VASS to maintain the schedule in a near-optimal state between runs of the more exhaustive optimisation procedure embodied in VASS.

Figure 3 compares Central Processor Unit (CPU) times for the two procedures. For D-VASS the daily value shown here is the total time elapsed time consumed in all D-VASS operations during the day (including background improvement), while for VASS the value shown is simply the elapsed time for the nightly run of the model. These plots show that VASS's CPU requirement is less than one-fifth that of D-VASS throughout the simulation period. However given that D-VASS's execution was spread out through each working day when—with a dedicated hardware platform for schedule optimisation—there is no other demand on the system, these times still represent less than 5% CPU utilisation. It is notable also that in both cases the CPU time appears to be dependent more on the 'difficulty' of the problems addressed (see the preceding discussion of unassigned rentals), than on problem size as such (eg total bookings in the system).

RM algorithms

Revenue management is concerned with the formulation of profit-maximising pricing policies, not merely with

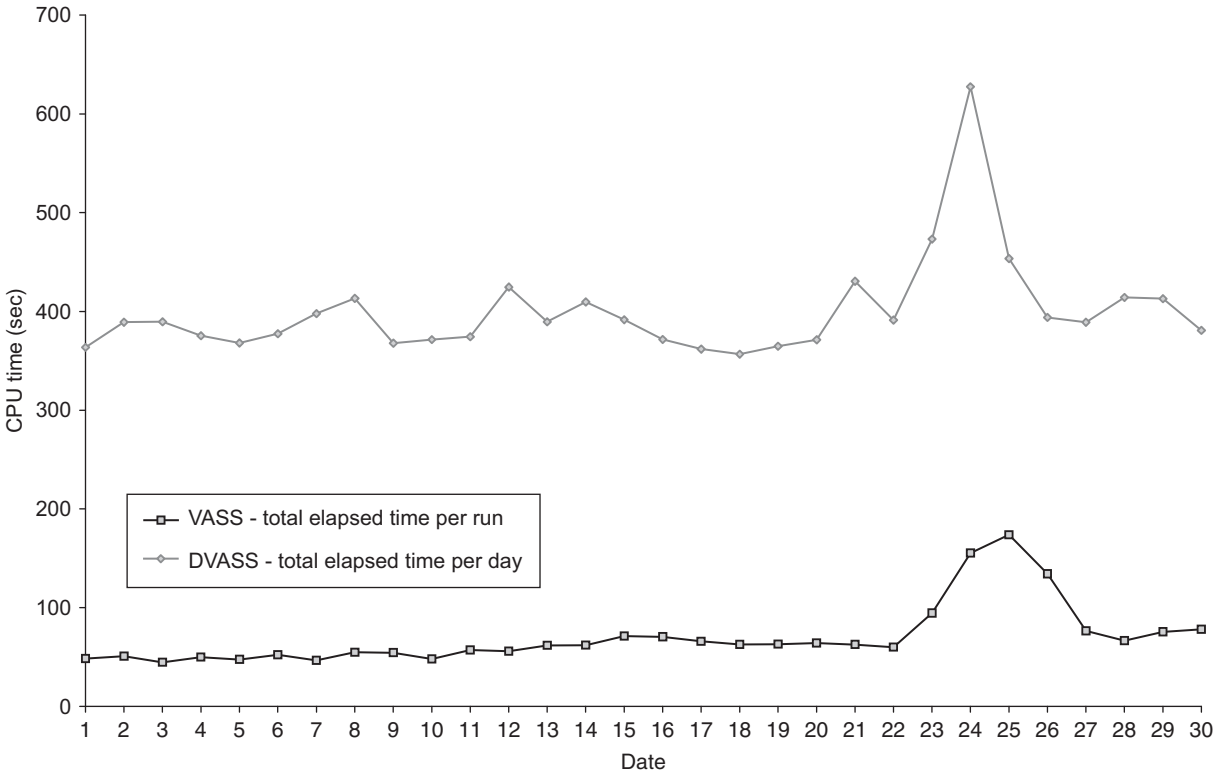

Figure 3 Simulation tests: CPU times.

operational tasks such as planning schedules. For example, with respect to a rental-request r, the scheduling task is concerned with the implementation of r; but from an RM perspective the more pertinent questions are whether r should be accepted, and if so, at what price. The network-flow formulation provides a convenient basis for investigating these questions. In the investigation reported here it is assumed that each customer responds to a specific, individually priced offer, and that customers can be grouped into classes, each customer class having some specified upper limit on what they are willing to pay (per day) for a rental.

We consider a simplified situation where there is only one vehicle type and a single location, turnaround times are zero, and all rental requests occur on a single day, 'today'. If we knew exactly how many customers of each type would be requesting rentals between today and the day of the rental, we would accept only the most profitable $|V|$ rentals (where V is the set of available vehicles) and refuse the rest. Although a deterministic approach of that sort is obviously impossible, we can use predictions of future demand to estimate the profit that could be obtained from a given number of vehicles if we serviced only the highest-paying customers.

The predictive analysis then focusses on the calculation of a cut-off price; that is, we accept rentals only from customers willing to pay at least this amount. The cut-off price depends on the number of unassigned vehicles and the time remaining during which more customers may arrive, and thus is represented in aggregate as a *pricing matrix* (see Figure 4). This matrix specifies for each day (prior to the departure day) what the price should be depending on how many vehicles are still available. An obvious but important property of such a pricing matrix is the decline in prices as the number of available vehicles increases.

Outlined below are two alternative techniques for incorporating such prices into the network-flow model. In each case substitutions are not allowed, and a rental over multiple days is modelled by defining separate rentals on successive days.

Location-dependent pricing: A price matrix is calculated for each location based on the expected value of rentals departing that location. The idle arcs in the network are split into many parallel idle arcs, each with a capacity of one, and with the cost of the ith arc on day t given by the negative value of the price matrix for i vehicles remaining with t days of lead time. In the optimal solution, any idle vehicle collects a profit equivalent to the expected revenue to be gained from the vehicle during the time that it is idle. The solution may include relocations to pre-position vehicles for expected rentals at more 'profitable' locations. In practice this could provide the opportunity to accommodate lucrative last-minute bookings at locations that are short of vehicles.

Fleet-based pricing: By ignoring the locational attributes of rentals, the whole fleet can be treated as a single pool of vehicles. We introduce a dummy location 'anywhere', so as to

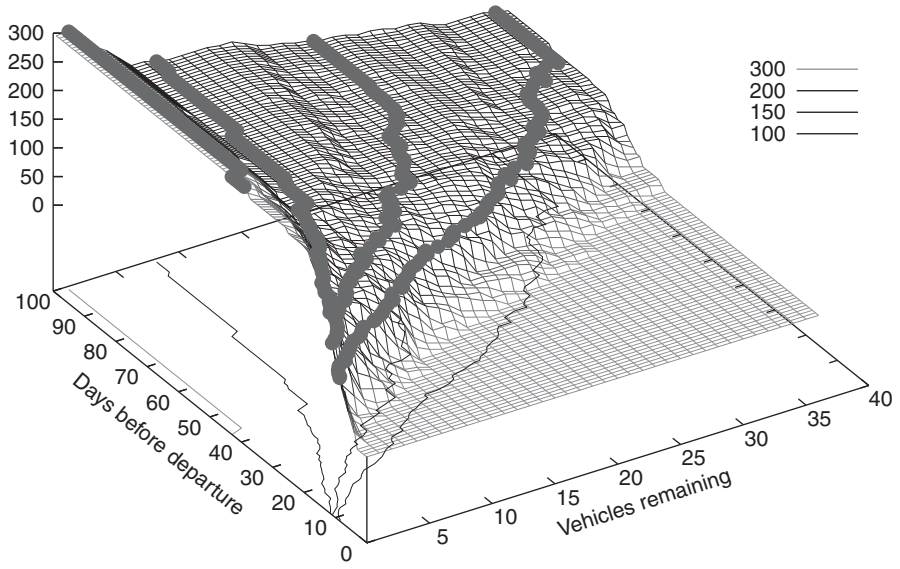

Figure 4 Minimum prices for today's rentals, from randomised data. Price is determined by the number of vehicles still available and the number of days remaining before departure. Expected demand comprises four customer classes paying \$100 to \$300, the fleet size is 40 vehicles, and the booking horizon is 100 days.

capture the total set of vehicles available anywhere. Idle arcs with negative costs from the pricing matrix are added between successive nodes of 'anywhere', while all other locations have zero-cost idle arcs. Relocation arcs to and from the other nodes in the network are connected to the 'anywhere' nodes. The costs and durations of these arcs are sufficiently high that a vehicle cannot be relocated faster or at lower cost by going through 'anywhere' nodes than through a direct relocation. Hence in this model a vehicle collects a 'revenue management bonus' only along idle arcs to 'anywhere' if the arc has a considerable duration. This reflects the fact that when most rentals are for periods of a week or more, there is limited value in making a vehicle available for only a day or two.

The procedure to decide whether a given rental should be accepted is the same in both cases. An arc representing the new rental is added to the network with lower bound of zero and upper bound of one, and the flow is re-optimised. The rental should be accepted only if the new optimal solution has a unit of flow across the new arc. A very quick approximation can be obtained by using dual prices to estimate the change in objective due to the new arc and calculating a shortest-flow augmenting path to ensure feasibility. This is similar in principle to the way bookings are added in the assignment model described earlier.

RM—test results

To validate the approach outlined above, we have carried out some simple numerical experiments. We present results from scenarios involving a single vehicle type and randomly generated customer arrivals from four classes of customers: Budget (\$100/day), Discount (\$150/day), International (\$200/day), and Full (\$300/day). We refer to the consequent distribution of rental requests in any given case as the *historical* demand, and to a randomised re-generation of requests with the same distribution as *instantiated* demand. This is used to distinguish between the apparent demand used for planning purposes and the 'actual' demand by which the effectiveness of the planning is evaluated. The distributions were generated such that rental requests from the lower-paying customers and for longer durations are biassed to generally arrive in the earlier part of the booking period.

We tested the randomised scenarios with the following algorithms:

Non-RM: This is the base case: rental requests are accepted whenever they can feasibly be included in the schedule. The results from the other scenarios are reported in terms of a percentage improvement in profit over this case.

L-D: Pricing matrices for each location are calculated using the Location-dependent method with *historical* data. The simulation is based on *instantiated* demand.

F-B: A single pricing matrix is calculated using the Fleet-based method with *historical* data. The simulation is based on *instantiated* demand.

UB: An upper bound on performance is obtained through retrospective optimisation, that is, by determining the rental requests that we would accept if we had perfect foreknowledge of all rental requests and customers of the instantiated demand.

Table 1 Improvements in revenue using RM methods, relative to the Non-RM base case

$\|B\|$	$\|L\|$	$\|V\|$	*UB* (%)	*L-D* (%)	*F-B* (%)
500	3	51	17.4	6.3	14.0
500	5	58	19.0	8.6	14.8
500	10	60	19.9	11.9	15.0
1000	3	106	16.0	3.3	12.9
1000	5	105	19.3	6.3	13.8
1000	10	108	18.9	8.8	14.1

Table 2 Improvements in revenue using RM methods, with less-than-expected demand

$\|B\|$	*UB* (%)	*L-D* (%)	*F-B* (%)
900	11.4	5.7	6.8
800	8.4	3.9	2.8
700	5.1	2.3	−1.2
600	3.7	1.5	−2.3

The results are shown in Table 1: $|B|$ is the number of rental requests, $|L|$ the number of locations, and $|V|$ the number of vehicles in the fleet. The results given are the average percentage improvement in revenue over the base case, from tests with five randomly generated 100-day scenarios with varying numbers of locations, rental requests, and vehicles. The fleet size is randomly generated based on the number of rental requests, in such a way that demand always exceeds supply.

These results show that RM can yield significant improvements in profitability. It is surprising that the Fleet-based approach appears to perform better than the Location-dependent method, although the difference becomes less pronounced as the number of locations increases. The improvements achieved using the Fleet-based approach are not far from the respective upper bounds, indicating that there would be little to gain through the use of more elaborate methods.

What impact does the accuracy of the forecast have on the effectiveness of the RM methods? We have investigated this question by running simulations with demand 10–40% less than the forecast (ie *historical*) data used to calculate prices. This simulates the case where an unexpected downturn in tourism or other economic conditions has a negative impact on demand—the opposite case is much less of an issue, since in periods of unexpectedly high demand the fleet be fully utilised, albeit with perhaps somewhat less profit than if the increase in demand had been anticipated. Each test scenario involved 128 vehicles and 10 locations over a period of 100 days. The results (see Table 2) indicate that RM can yield substantial benefits to the fleet operator, even with inaccurate forecasts. Of the two methods, it appears that the Location-dependent procedure is more robust with respect to errors in the forecast of the kind modelled here.

Conclusions

In this paper we have defined a significant and complex problem arising from current practice in recreational vehicle-rental operations, and we have presented assignment and network-flow algorithms to solve the problem. We have shown how the assignment approach can be applied to the problem in a dynamic context, and we have explored the use of the network-flow algorithm for RM purposes.

The assignment and network-flow algorithms described in this paper are now established as vital elements of THL's rental operations (Ernst *et al*, 2007b), embodied in the D-VASS and VASS modules (see the outline of computational tests, above). D-VASS uses the assignment algorithms to support THL's reservations system. D-VASS answers availability queries submitted online by reservations staff and inserts confirmed rentals into the current schedule, while running an improvement procedure as a background process. The static task is carried out by the VASS module, which is based on the network-flow algorithm. VASS is invoked each night to rebuild the schedule de novo, correcting sub-optimalities introduced during the day.

The computational tests demonstrate the effectiveness of the software in managing a full-scale schedule at optimal or near-optimal efficiency, and indicate the reliability of the D-VASS module in handling incremental changes to the schedule without substantially impairing its efficiency. THL estimates that the systems described here have produced an improvement in fleet utilisation of approximately 20% with total direct savings amounting to more than 5% of the company's AUD4 million annual operating costs. Besides these direct impacts on operational efficiency, VASS and D-VASS have yielded greatly improved flexibility in the management of the rental fleet and in the company's ability to meet the desires of customers.

The results of the RM experiments show that the proposed approach could yield significant further improvements in profitability. To gain confidence that the estimated benefits could be achieved in practice, a more comprehensive study would be required, taking into account issues such as competition, leakage between customer classes, cancellations, prediction and monitoring techniques, and other considerations that lie beyond the scope of this paper.

References

Ahmed A and Poojari C (2008). An overview of the issues in the airline industry and the role of optimization models and algorithms. *J Opl Res Soc* **59**: 267–277.

Bellman R (1958). On a routing problem. *Q Appl Math* **16**: 87–90.

Botimer T and Belobaba P (1999). Airline pricing and fare product differentiation: A new theoretical framework. *J Opl Res Soc* **50**: 1085–1097.

Clarke L, Johnson E, Nemhauser G and Zhu Z (1997). The aircraft rotation problem. *Ann Opns Res* **69**: 33–46.

Cormen TH, Leiserson CE and Rivest RL (1989). *Introduction to Algorithms. The MIT Electrical Engineering and Computer Science Series*. The MIT Press: Cambridge, Massachusetts.

Currie R and Salhi S (2003). Exact and heuristic methods for a full-load, multi-terminal, vehicle scheduling problem with backhauling and time windows. *J Opl Res Soc* **54**: 390–400.

Dell'Amico M and Toth P (1999). Algorithms and codes for dense assignment problems: The state of the art. *Discrete Appl Math* **100**(1–2): 17–48.

Desaulniers G, Lavigne J and Soumis F (1998). Multi-depot vehicle scheduling problems with time windows and waiting costs. *Eur J Opl Res* **111**: 479–494.

Dijkstra E (1959). A note on two problems in connexion with graphs. *Numer Math* **1**: 269–271.

Dumas Y, Desrosiers J and Soumis F (1991). The pickup and delivery problem with time windows. *Eur J Opl Res* **54**: 7–22.

Engquist M (1982). A successive shortest path algorithm for the assignment problem. *Infor* **20**: 370–384.

Ernst A, Horn M, Kilby P and Krishnamoorthy M (2007a). *Static and real-time algorithms for recreational vehicle fleet scheduling*. Technical Report 2007/46, CSIRO Mathematical and Information Sciences.

Ernst A, Horn M, Kilby P, Krishnamoorthy M, Degenhardt P and Moran M (2007b). Static and dynamic order scheduling for recreational rental vehicles at Tourism Holdings Limited. *Interfaces* **37**: 334–341.

Fink A and Reiners T (2006). Modeling and solving the short-term car rental logistics problem. *Transport Res E* **42**: 272–292.

Ford L and Fulkerson DR (1962). *Flows in Networks*. Princeton University Press: Princeton, NJ.

Gopalan R and Talluri K (1998). Mathematical models in airline schedule planning: A survey. *Ann Opns Res* **76**: 155–185.

Hertz A, Schindl D and Zufferey H (2009). A solution method for a car fleet management problem with maintenance constraints. *J Heuristics* DOI: 10.1007/S10732-008-9072-4.

Ioachim I, Desrosiers J, Dumas Y, Solomon MM and Villeneuve D (1995). A request clustering algorithm for door-to-door handicapped transportation. *Transport Sci* **29**: 63–78.

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

Savelsbergh M and Sol M (1998). DRIVE: Dynamic routing of independent vehicles. *Opns Res* **46**: 474–490.

Yeoman I, Ingold A and Kimes S (1999). Yield management: Editorial introduction. *J Opl Res Soc* **50**: 1083–1084.

Received February 2008;
accepted April 2009 after one revision