PERGAMON

Transportation Research Part C 10 (2002) 35–63

TRANSPORTATION RESEARCH PART C

www.elsevier.com/locate/trc

Fleet scheduling and dispatching for demand-responsive passenger services

Mark E.T. Horn [*]

CSIRO Mathematical and Information Sciences, G.P.O. Box 664, Canberra, ACT 2601, Australia

Received 21 January 2000; accepted 28 June 2000

Abstract

This paper describes a software system designed to manage the deployment of a fleet of demand-responsive passenger vehicles such as taxis or variably routed buses. Multiple modes of operation are supported both for the fleet and for individual vehicles. Booking requests can be immediate (i.e. with zero notice) or in advance of travel. An initial implementation is chosen for each incoming request, subject to time-window and other constraints, and with an objective of minimising additional travel time or maximising a surrogate for future fleet capacity. This incremental insertion scheme is supplemented by post-insert improvement procedures, a periodically executed steepest-descent improvement procedure applied to the fleet as a whole, and a "rank-homing" heuristic incorporating information about future patterns of demand. A simple objective for trip-insertion and other scheduling operations is based on localised minimisation of travel time, while an alternative incorporating occupancy ratios has a more strategic orientation. Apart from its scheduling functions, the system includes automated vehicle dispatching procedures designed to achieve a favourable combination of customer service and efficiency of vehicle deployment. Provision is made for a variety of contingencies, including travel slower or faster than expected, unexpected vehicle locations, vehicle breakdowns and trip cancellations. Simulation tests indicate that the improvement procedures yield substantial efficiencies over more naïve scheduling methods and that the system will be effective in real-time applications. © 2001 Elsevier Science Ltd. All rights reserved.

Keywords: Scheduling; Dispatching; Demand-responsive; Multi-modal; Vehicle routing; Public transport; Passenger transport; Dial-a-ride; Ride-sharing; Taxis

1. Introduction

This paper describes a scheduling and dispatching system called L2sched, which is designed to address the needs of a variety of demand-responsive, road-based passenger transport fleets and

[*] Tel.: +61-2-6216-7000; fax: +61-2-6216-7111.
E-mail address: mark.horn@cmis.csiro.au (M.E.T. Horn).

operating modes. At present L2sched is a component of the LITRES-2 modelling system, where it deals with simulated vehicles, travellers and so on. The LITRES-2 system architecture (Horn, 2001) will facilitate a transfer to an operational setting.

The transport modes considered in this research may include special services for disabled or aged people, general-purpose "maxi-taxi" services, ride-sharing arrangements, and conventional taxis. Responsiveness to demand in each case requires effective communication between prospective travellers and a scheduling centre, and between the centre and the drivers of the various vehicles. These communications can be effected by telephone and wireless, respectively, supplemented perhaps by automated location-reporting technologies.

In operational terms there can be significant overlap amongst the various demand-responsive modes; for example, a taxi may be available for single- and multiple-hire service simultaneously, while a larger vehicle may be run alternately under a pre-booked charter arrangement and as a variably routed bus. For flexibility in such cases, and to maximise the opportunities for achieving efficient deployment, L2sched embraces all vehicles in a single fleet, the various operating modes being regarded as characteristics of individual vehicles rather than of the fleet. While this arrangement allows the possibility of a mode-by-mode partitioning of the fleet (e.g. for marketing purposes), it highlights the economies of scale available to travellers and fleet operators when demand-responsive services are managed centrally. For travellers, the advantages of central management lie in the convenience of a single point of access and the likelihood (other things being equal) of prompt service. For a fleet operator, the main benefits derive from the opportunities for achieving an efficient use of resources, notably through high occupancy levels in multiple-hire vehicles.

In many cases a transport fleet is operated with a deadline on bookings, so that the pattern of demand is fully defined before scheduling takes place. The scheduling task then involves the assignment of a pre-defined set of passenger trips to a set of vehicles and the construction of routes for individual vehicles, subject to time and capacity constraints. This Pickup and Delivery Problem with Time Windows (PDPTW) may be regarded as a generalisation of the Vehicle Routing Problem (VRP), which involves deliveries between pickup or setdown points and a central depot (inter-locational connectivity is many-to-many in the PDPTW, as against many-to-one in the VRP). Several authors have investigated the PDPTW, with applications in goods transport (Dumas et al., 1991), transport for handicapped persons (Ioachim et al., 1995), and dial-a-ride operations (Jaw et al., 1986). The work of Jaw et al. is of particular interest in relation to the present work, involving as it does a time-windowed incremental-insertion heuristic similar in some respects to heuristics in L2sched. Other optimisation approaches are discussed in the concluding section of this paper.

By contrast with the statically defined PDPTW, a dynamic context for demand-responsive transport involves requests, issued in real time, for more-or-less immediate service. This corresponds with a typical taxi-dispatching environment, for which the vendors of radio-dispatch hardware now provide various kinds of automated assistance. This assistance however is generally of a rudimentary nature, involving for example the notification of work opportunities to drivers on the basis of the pickup stop's location in relation to a pre-defined zonal tessellation, with no attempt to plan future work assignments. Although there are strong grounds for questioning the efficiency of such procedures (Horn et al., 1999), they are understandable in the light of institutional arrangements which accord a substantial degree of independence to individual taxi drivers.

Similarly "short-sighted" techniques have been adopted in other contexts, for example for a real-time ride-sharing application (Kowshik et al., 1994), and for variable-route bus services (Smith et al., 1998).

L2sched is intended to bridge the gap between the static and dynamic approaches described above. Demand is realised as a stream of trip-requests, which are scheduled as they arrive. Travel requirements are temporally elaborated so as to distinguish between the times when a request is issued and service is required, thus permitting a long-sighted view of fleet management in which opportunities for wider optimisation can be exploited. Similarly, scheduling objectives are designed to obtain fleet-wide efficiency of deployment while satisfying the travel requirements inscribed in each trip-request. This approach obviously implies a strong centralisation of dispatching and scheduling functions (cf. the decentralised system propounded by Dial, 1995).

The following three sections of this paper outline scheduling concepts, terminology, and routing conventions. Section 5 discusses scheduling objectives, and on this basis develops two main formulations for the objective function. Section 6 describes the main trip-insertion and optimisation strategies embodied in L2sched, and these strategies are elaborated in Sections 7–9. Sections 10 and 11 describe provisions for real-time dispatching and contingency-handling, respectively. Section 12 reports computational tests on the procedures developed in Sections 5–9, and Section 13 discusses some possible directions for further research. Additional technical information is provided in the appendices (for further details see Horn, 1999).

2. The scheduling context

L2sched is provided at the outset with a set of parameters, a description of a road network, and a file specifying the characteristics and availability of each vehicle in the fleet. Communication between L2sched and the outside world or simulator involves messages of the following kinds.

2.1. Real-time inputs

(a) *Information about fleet activity*, such as vehicle arrivals, departures, trip cancellations and breakdowns. This information may include vehicle-location information issued at intervals by automated vehicle location (AVL) systems.
(b) *Trip requests*. Each request applies to a group of one or more prospective passengers, and includes geographical locations and time-windows for a pickup and a setdown stop. Requests are issued by a request-broker (Horn, 2001), and by taxi-drivers wishing to exploit opportunistic pickup opportunities (see Section 3).
(c) *Booking commands*. A booking command calls for the scheduling of a trip which an immediately preceding trip request has found to be feasible. A booking implies a promise that the trip will be carried out, although perhaps not in the implementation planned when it was booked.
(d) *Booking cancellations*. A booking cancellation message informs L2sched that a previously made booking has been cancelled.

2.2. Real-time outputs

(a) *Dispatch messages*. A dispatch directs the driver of a specified vehicle to proceed at a specified time to a specified stop.
(b) *Responses to trip-requests*, indicating success or failure in each case, and if successful, recording a possible implementation.
(c) *Information messages*. These provide information about the state of the system and about actions such as new bookings and trip restarts (i.e. re-bookings of stranded passengers).

3. Definitions

A *mode* is a type of service requested by passengers, and may be one of the following:

TaxiSingle, TaxiMulti	A pre-booked door-to-door taxi, single- or multiple-hire.
TaxiOPSingle,TaxiOPMulti	A conventional "opportunistic-pickup" taxi service.
RovingBus	A taxi-like service, operating between pre-defined stops.
SmartShuttle	A timetabled taxi/bus service, deployed on demand.

These categories might be further elaborated to meet specialised requirements; for example transport services for disabled persons would require only minor extensions to the scheduling framework, so as to record special features such as wheelchair accessibility and to match these with travellers' requirements. A distinction is made between *single-hire* modes (TaxiSingle, TaxiOPSingle) and *multiple-hire* modes (the others). The modes acceptable to each vehicle are specified in advance, and a vehicle's cargo at any given time may include one party of passengers in a single-hire mode, or perhaps several parties in a mixture of multiple-hire modes.

The deployable fleet is defined as a set of *shifts*, each of which indicates the availability of a given vehicle during a given time-period, and is represented by a deployment plan or *itinerary*. The latter essentially is a list of *stops*, each one representing a place to be visited by the vehicle, together with a time-window for the visit. The sequencing of stops in an itinerary corresponds with the order in which the stops are to be visited, and the trajectory of a vehicle from one stop to the next is called a *leg*. The following notation is used in referring to the fleet and to individual itineraries:

$\mathscr{F}$	The fleet managed by the scheduler, defined as a set of itineraries.
$\mathscr{S}_I$	The sequence of stops currently assigned to an itinerary I.
$\mathscr{L}_I$	The set of legs connecting $\mathscr{S}_I$.
$\mathscr{M}_I$	The set of modes that can be served by itinerary I.
veh_I	The vehicle represented by itinerary I.
cap_I	Maximum number of passengers that can be carried by veh_I.
loc_I	Cursor indicating current location of veh_I in $\mathscr{S}_I$.
z_I	Current system cost for I, as criterion for optimisation (see Eq. (5.2)).

A *trip-request* specifies a pickup and a setdown stop for a group of one or more passengers wishing to travel together in a given mode. If a request is accepted, its implementation is called a *trip*. The realisation of a trip t is defined by its assignment to an itinerary $I(t)$ and the placement of its pickup and setdown stops in the stop-list $\mathscr{S}_{I(t)}$. A trip-request or trip t is specified as follows:

$mode_t$	Transport mode for trip t.
$npass_t$	Number of passengers travelling together on trip t.
$I(t)$	Itinerary to which trip t is currently assigned, implying $S_{\mathrm{PU}(t)}, S_{\mathrm{SD}(t)} \in \mathscr{S}_{I(t)}$.
$S_{\mathrm{PU}(t)}$	Pickup stop specifying location and time-window.
$S_{\mathrm{SD}(t)}$	Setdown stop specifying location and time-window.

In practice, trips are usually requested in simple temporal terms that can be recast easily in the uniform "fully windowed" form described here; for example, "Pick me up at A after 11:00, and get me to B by 11:30" implies a time-window {11:00–11:30} at both A and B.

3.1. Stop attributes

Just as a trip is delimited by a Pickup and a Setdown stop, each itinerary is delimited by a StartShift and an EndShift stop. Other types of stops are defined for specialised purposes: a Rankloc is a cab-rank chosen as a destination by a rank-homing heuristic (see Section 9 below), while a *wayside stop* defines a vehicle's current location with respect to its itinerary (procedures for establishing wayside and other stop locations are given in Horn, 1999). The notation for permanent attributes of a stop S is summarised below.

$type_S$	The type of stop, as discussed above.
x_S, y_S	Spatial location.
t_S^{start}	Start of the time-window for S.
t_S^{end}	End of the time-window for S.
t_S^{serv}	Duration of service interval at S, normally for loading or unloading passengers.

For a pickup or setdown stop S, the time-window limits $\{t_S^{\mathrm{start}}, t_S^{\mathrm{end}}\}$ embody travellers' requirements (see also Appendix C). Other attributes of S depend on its current implementation, that is, on the itinerary I to which S is assigned and the ordering of stops $\mathscr{S}_I$. The maintenance and use of these attributes is explained in subsequent parts of this paper and in the appendices.

$sprev_S$	Immediate predecessor of S in $\mathscr{S}_I$.
$snext_S$	Immediate successor of S in $\mathscr{S}_I$.
$npass_S$	Number of passengers on board the vehicle on departure from S.
t_{ST}^{leg}	Travel time on the leg connecting S to its successor T (i.e. $snext_S$).
EAT_S	Earliest arrival time, assuming earliest feasible service at S's predecessors.
LDT_S	Latest departure time at S to ensure feasible service at S's successors.
EAT'_S	Earliest feasible commencement of a service interval at S.
LDT'_S	Latest feasible termination of a service interval at S.

3.2. Slack time

The slack time in an itinerary is time that is allocated neither to travelling nor to servicing stops. In a narrow sense this is unproductive waiting time, but it provides leeway for the insertion of additional stops. For a single stop S, the maximum slack can be defined as $t_S^{\text{slack}} = \text{LDT}_S - \text{EAT}_S - t_S^{\text{serv}}$. This time is in effect shared between stops; for example, if a vehicle is running early over a sequence of stops $S \to S' \to S''$ with $t_S^{\text{slack}} = t_{S'}^{\text{slack}} = t_{S''}^{\text{slack}} = s$, the total slack is merely s. The available slack in an itinerary and the time actually spent waiting at each stop depend on the dispatching strategy adopted (see Section 10).

3.3. Real-time attributes

L2sched's sense of time passing is embodied in a current time t^{now}, obtained from time-stamps on incoming messages (see Section 2). Arrival and departure messages provide the basis for estimating a vehicle's current position with respect to its stop-sequence, and its current status (i.e. stopped or in transit between stops). A *movable trip* is one that is not yet operationally committed; that is, a trip t such that $veh_{I(t)}$ has not yet reached $S_{\text{PU}(t)}$. Such a trip can be transferred to an itinerary other than $I(t)$ without violating the logical requirement that $S_{\text{PU}(t)}$ and $S_{\text{SD}(t)}$ should be assigned to the same itinerary.

3.4. Opportunistic pickups

The TaxiOPSingle and TaxiOPMulti modes represent traditional taxi services that pick up passengers at cab-ranks or from the side of the road. A vehicle that can operate in one of these modes is called a *TaxiOP vehicle*. A trip in such a vehicle differs from a phone-booked trip in that its initiation requires the conjunction of the vehicle and the travelling party in time and space (i.e. at the pickup point). The modelling of these conjunctions is carried out not in L2sched but in the LITRES-2 simulator, where special treatment is given to parties seeking OP service. In brief, the simulator sends a trip-request to L2sched whenever such a party arrives at a network node where a TaxiOP vehicle is waiting, and whenever a TaxiOP vehicle arrives at a network node where an OP party is waiting. L2sched includes mechanisms designed to ensure adequate service at nodes (normally cab-ranks) where OP parties are *likely* to be waiting: these mechanisms can be useful also in directing vehicles of all kinds to localities where demand is likely to be concentrated (see Section 9).

4. Routing of vehicles between stops

The road system is specified as a network model, typically with minor roads filtered out: pickup and setdown locations on the other hand are defined as points in continuous space. Since it can be problematic for a vehicle to stop between the ends of a major network link, access from the network to a given pickup or setdown stop is assumed to occur by way of a "gateway" to the stop, namely the network node nearest to the stop.

Vehicles thus are assumed to travel by shortest-time paths between the nodes of the network, with off-net detours (possibly of zero length) from nodes to reach pickup and setdown stops. The road network model performs network calculations with the aid of a pre-calculated all-pairs shortest-path-time matrix, on the assumption that link speeds are invariant with respect to the time of day (techniques for networks with time-varying link speeds are explored in Horn, 2000;

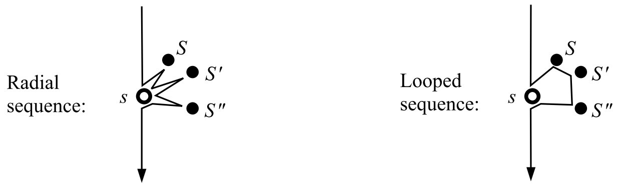


Fig. 1. Travel between stops around a common network node.

these techniques have been incorporated in L2sched subsequent to the research described in the present paper). Off-network travel components are estimated by assuming a constant off-net travel speed. Planar distances are calculated using either the Pythagorean or Manhattan metric, the choice being determined by a run-time parameter.

The fastest trajectory between a pair of stops P and Q is normally $\{P \rightarrow p \cdots \rightarrow q \rightarrow Q\}$, where p and q are the nearest network nodes to P and Q, respectively, and $p \rightarrow q$ takes the shortest-time network path between these two nodes. If another stop is required between p and q, say at a point S nearest another node s, the fastest trajectory will be $\{P \rightarrow p \cdots \rightarrow s \rightarrow S \rightarrow s \cdots \rightarrow q \rightarrow Q\}$. Special treatment is required in the case where an itinerary includes a succession of stops at points $\{S, S', \ldots\}$ with a common nearest-node s.

A naïve extension of the convention outlined above would lead to a radial pattern of travel such as that shown at the left in Fig. 1, but the "looped" sequence shown on the right is clearly more efficient because of the triangle inequality in continuous space (e.g. $t_{SS'} \leqslant t_{Ss} + t_{sS'}$). The trip-insertion procedures (see Section 7) tend to induce looped sequences, due to the emphasis on minimising travel duration. The time-window and capacity constraints may however require exceptions to this; for example, if S and S' in Fig. 1 are the pickup stops and S'' is a setdown, the capacity constraint may enforce $\{s \rightarrow S'' \rightarrow S \rightarrow S' \rightarrow s\}$, even though the minimum-time sequence is $\{s \rightarrow S \rightarrow S' \rightarrow S'' \rightarrow s\}$.

5. The objective function

L2sched recognises an explicitly passenger-oriented criterion for fleet deployment, namely that of minimising the time taken to reach the destination, which is applicable mainly to conventional taxi services (see Appendix C). The following analysis however is focussed on the complexities associated with fleets whose economic viability is dependent on efficient operation and high occupancy levels. Service quality of course is still important for such fleets, but it need not be embodied explicitly in the objective function (e.g. it can be encapsulated in the temporal constraints applied to trips, see Appendix C). We are concerned thus with objectives that are important from the point of view of a fleet operator, such as the following.

- Minimise *total vehicle travel time* or *distance*. This implies direct benefits for both passengers (short average trip times) and the fleet operator (low running costs).
- Maximise *total ridership*, defined as the total number of passengers carried by the fleet.

A tactical element is apparent in the case of ridership, which can be maximised only as the outcome of a series of scheduling decisions. The indirect nature of this criterion is due to the fact that each incoming trip-request is always accepted if any feasible insertion can be found for it: the selection or rejection of requests on the basis of their impact on ridership (e.g. to favour large parties travelling short distances over single travellers travelling long distances) would almost certainly be unacceptable in practice.

The relations between travel time and ridership are illustrated in Fig. 2, which shows the current itineraries of two vehicles v_1 and v_2, and a trip-request $i \rightarrow j$. The request could be accommodated feasibly by either vehicle, the additional travel time being approximately the same in both cases. Suppose however that v_1 is currently empty of passengers (e.g. its only commitment is its eventual return to a shift-changeover point), while v_2 is almost at full capacity. Under normal loading conditions, it would probably be best to assign the trip to v_1, so as to ensure utilisation of v_1 (whose current empty state is possibly due to its being located far from most current demand), and to avoid an unnecessary augmentation of travel time for passengers currently carried by v_2. On the other hand, if the fleet is spatially well-aligned with the current pattern of demand and is operating at or close to a saturated (i.e. fully-loaded) condition, an assignment to v_2 may be preferable on the grounds that it preserves more flexibility for future deployments of v_1. Thus the significance of individual vehicles' passenger loadings can depend on current and expected fleet-wide loadings, and on the location of vehicles in relation to current and expected demand.

The current research has involved tests with two objective functions, one involving simply the minimisation of travel time, and a composite form incorporating passenger loadings as well as travel time. In both cases, the objective function $Z_{\mathscr{F}}$ is defined as the sum of costs z_I applying to each itinerary (Eq. (5.1)); z_I in turn is defined in terms of a function *zleg* applied to each leg of I (Eq. (5.2)); while *zleg* is designed to embody the travel time and passenger-loading considerations discussed above (5.3) or (5.4). That is, minimise $Z_{\mathscr{F}}$, where

$$Z_{\mathscr{F}} = \underset{\forall I \in \mathscr{F}}{\bullet}\, z_I \tag{5.1}$$

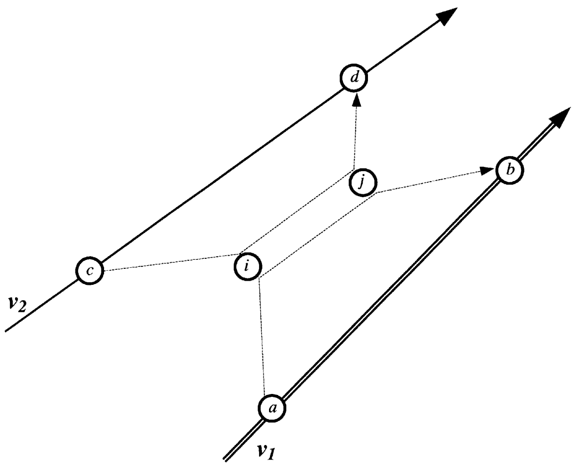


Fig. 2. Criteria for trip-insertion.

and

$$z_I = \bullet\ zleg_I(tleg_k, r_k) \qquad (5.2)$$

$$\forall k \in \mathscr{L}_I$$

And $zleg_I$ has one of the following forms.

$$zleg_{I(time)}(tleg_k, r_k) = t_k^{leg} \qquad (5.3)$$

$$zleg_{I(tload)}(tleg_k, r_k) = t_k^{leg}(tfact + rfact\sqrt{2r_k - r_k^2}) \qquad (5.4)$$

Additional quantities in Eq. (5.4) are defined as follows.

r_k	Passenger occupancy on leg k of itinerary I, defined as $npass_k/cap_I$.
$rfact$	Parameter indicating weight attached to ridership, with range $0 < rfact < 1$.
$tfact$	Weight attached to travel time, defined as $tfact = 1 - rfact$.

Eq. (5.4) includes an expression describing the northwest quadrant of a circle, with values in the range zero to one and a monotonic-decreasing gradient: this function has been chosen on the basis of considerations outlined below, informed by preliminary experiments with other formulations. The parameter $rfact$ determines the relative weight to be given to the quadrant function: in tests carried out so far, the most favourable results in terms of ridership and efficiency have been obtained with $rfact \bullet 0.5$ (see also Section 12).

The impact of the quadrant function may be illustrated with reference to the example discussed earlier in relation to Fig. 2. Although this illustration is somewhat simplified, it will be apparent that a similar analysis can be applied in more complex cases, where the setdown stop j of the new trip $i \rightarrow j$ is interleaved with other trips (e.g. subsequent to b in the itinerary of v_1). Assuming that v_1 is currently empty, v_2 is carrying three passengers, and that the passenger capacity is four for both v_1 and v_2, the marginal costs of carrying trip $i \rightarrow j$ (one passenger) can be written as:

$$\Delta z(v_1) = zleg_I(ij, 1/4) - zleg_I(ij, 0) + zleg_I(ai + ij + jb - ab, 0)$$

$$\Delta z(v_2) = zleg_I(ij, 1) - zleg_I(ij, 3/4) + zleg_I(ci + ij + jd - cd, 3/4)$$

In both cases the marginal cost comprises a "direct" component associated with the increased passenger load for the trip itself (the first two terms), plus a component incurred by deviations from the current itinerary. Comparing vehicles v_1 and v_2, it is evident that the direct cost component is less for v_2, while the deviational component is less for the lightly-loaded v_1. In general then, the quadrant function can be expected to favour the assignment of trips to lightly-loaded vehicles when the geographical distribution of the fleet is sparse or is poorly matched with the distribution of demand, since under these conditions the deviational components will tend to outweigh the direct components. Eq. (5.4) is thus broadly consonant with the tactical considerations discussed earlier in this Section.

6. Procedural scheme

Because the full set of trips to be scheduled is not known in advance, the scheduling task in principle is to find a succession of optimal plans for the deployment of the fleet. The difference between one optimal plan and the next is induced by a (typically) small change in scheduling conditions, namely the accommodation of an additional trip. Thus the optimal plan does not necessarily change radically in any single step, but instead may be regarded as evolving over time. This evolutionary view is reflected in the three-tier strategy used in L2sched:

(a) Each trip is inserted in the schedule in such a way as to minimise the marginal cost of insertion.

(b) Immediately following a trip-insertion, an attempt may be made to find local improvements made possible by the insertion.

(c) At intervals an attempt may be made to find improvements on a broader basis than that attempted in (b).

The following sections describe these procedures, which are implemented using "atomic" insert-test, insertion and transfer operations (see Appendix B). Because they are not designed to exploit the properties of any particular formulation of $zleg_I$, the procedures are discussed in terms of the generic objective of minimising total cost, of which Eqs. (5.3) and (5.4) are possible formulations.

7. Choosing trip-insertions

An implementation for a given trip or trip-request t involves the choice of an itinerary and the insertion of t in the chosen itinerary's stop-list. A *future* insertion in an itinerary involves inserting both stops of t after the current leg of the itinerary (i.e. the leg just traversed or currently being traversed), and thus does not require the immediate re-routing of an in-transit vehicle. Either of two procedures can be used for finding a future insertion. The *exhaustive-search* procedure is guaranteed to find the least-cost feasible future insertion, if any. The *heuristic* procedure offers no such guarantee. Analysis of possible insertion configurations indicates that for an unloaded objective function (as in Eq. (5.3)), the heuristic procedure will find the least-cost insertion in almost all cases (see also Section 12).

The search for an insertion of t is largely concerned with the insertion of S_{PU} and $S_{\mathrm{SD}}(S_{\mathrm{PU}} = S_{\mathrm{PU}(t)}, S_{\mathrm{SD}} = S_{\mathrm{PU}(t)})$. The following notation is used to describe the insertion procedures:

π^* Currently selected insertion for t in $\mathscr{F}$, with initially $\pi^* = \emptyset$.
Δ^* Net additional cost incurred by π^*, with initially $\Delta^* = \infty$.
π_i Insertion of S_{PU} in a leg $i \rightarrow snext_i$.
Δ_i Lower bound on net additional cost incurred by π_i.
It can be shown (Horn, 1999) that for $i' = snext_i$ and $P = S_{\mathrm{PU}}$,
$\Delta_i = zleg_I(t_{iP}^{\mathrm{leg}} + t_{Pi'}^{\mathrm{leg}} - t_{ii'}^{\mathrm{leg}}, npass_{ii'})$
π_{ij} Insertion of S_{PU} and S_{SD} in legs $i \rightarrow snext_i$ and $sprev_j \rightarrow j$.
Δ_{ij} Net additional cost incurred by π_{ij}.

7.1. Exhaustive-search insertion – explore all pickup-setdown combinations

For each itinerary $I \in \mathscr{F}$ with $mode_t \in \mathscr{M}_I$, do the following.
(1) For each feasible insertion π_i of S_{PU} in I with $\Delta_i < \Delta^*$, do the following:
For each feasible insertion π_{ij} of S_{SD} in I after i with $\Delta_{ij} < \Delta^*$, set $\pi^* = \pi_{ij}$ and $\Delta^* = \Delta_{ij}$.

7.2. Heuristic insertion – choose "optimal" pickup, then "optimal" setdown

For each itinerary $I \in \mathscr{F}$ with $mode_t \in \mathscr{M}_I$, do the following:
1. (a) Set $\pi_{i^*} = \emptyset$ and $\Delta_{i^*} = \Delta^*$. (b) For each feasible insertion π_i of S_{PU} in I with $\Delta_i < \Delta_{i^*}$, set $\pi_{i^*} = \pi_i$ and $\Delta_{i^*} = \Delta_i$. (c) If $\pi_{i^*} = \emptyset$, stop.
2. For each feasible insertion π_{i^*j} of S_{SD} in I after i^* with $\Delta_{i^*j} < \Delta^*$, set $\pi^* = \pi_{i^*j}$, $\Delta^* = \Delta_{i^*j}$.

7.3. Pre-emptive insertion – pickup after wayside, then setdown

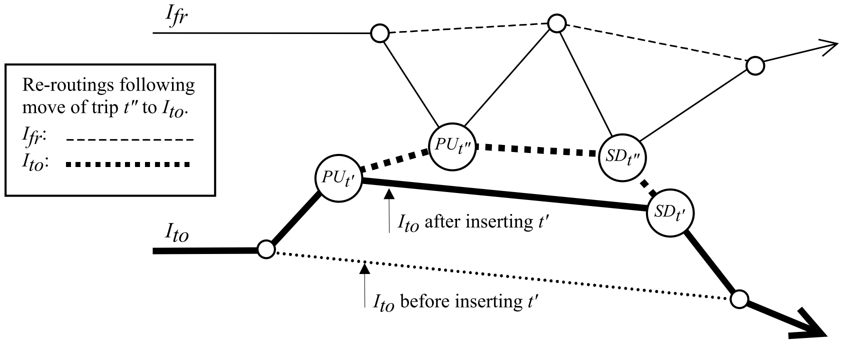
A *pre-emptive* insertion involves inserting the pickup stop in the leg that is currently being traversed by a vehicle veh_I, requiring an immediate dispatch to re-direct veh_I to the new pickup. Insertions of this kind may complement the future insertions outlined above, subject to operational circumstances (e.g. the willingness of drivers to accept short-term re-routings). The pre-emptive insertion procedures run as follows:
For each itinerary $I \in \mathscr{F}$ with $mode_t \in \mathscr{M}_I$ and $status_I = in\text{-}transit$, do the following:
1. Define a wayside stop w in the current leg $R \rightarrow T$ of $\mathscr{L}_I$. The wayside divides $R \rightarrow T$ into an "all-past" leg $R \rightarrow w$ and an "all-future" leg $w \rightarrow T$.
2. If π_i $(i = R \rightarrow T)$ is infeasible or $\Delta_i \geqslant \Delta^*$, stop.
3. The stop-list for possible insertion of S_{SD} comprises $\{S_{\mathrm{PU}}, T, snext_T \ldots\}$. For each feasible insertion π_{ij} of S_{SD} in this list with $\Delta_{ij} < \Delta^*$, set $\pi^* = \pi_{ij}$ and $\Delta^* = \Delta_{ij}$.

8. Improvement procedures

8.1. Post-insertion transfers

A transfer procedure seeks to improve the scheduling configuration after a trip-insertion in an itinerary I_{to}, by means of improving shifts of trips into I_{to}. In Fig. 3, trip t' has just been inserted in I_{to}, and the idea is that the deviation thus induced in I_{to} may bring the latter close enough to the pickup and setdown stops of a trip t'' in another itinerary I_{fr} to make it worthwhile to move t'' from I_{fr} to I_{to}. The procedure embodies a steepest-descent strategy, as follows:

1. For each itinerary I_{fr} in $\{\mathscr{F} - \{I_{\mathrm{to}}\}\}$, consider each movable trip currently assigned to I_{fr} whose transfer to I_{to} would not impair the feasibility of I_{to} and whose net cost would be less than zero. Out of all such feasible transfers, record the trip t^* with the least net transfer cost.
2. If a least-cost transfer was found, carry it out and repeat from step 1.



I_{fr}
Re-routings following move of trip t'' to I_{to}.
I_{fr}:
I_{to}:
$PU_{t''}$
$SD_{t''}$
$PU_{t'}$
$SD_{t'}$
I_{to} after inserting t'
I_{to}
I_{to} before inserting t'

Fig. 3. Local trip transfer after trip-insertion.

8.2. Post-insertion shuffle

A local shuffle procedure can be applied alone or in combination with the transfer procedure described above. Following the insertion of a trip in an itinerary I, the procedure shuffles the trips in I as follows:

1. Form a set $\mathscr{T}$ of movable trips assigned to I. Stop if any such trip is infeasible or $|\mathscr{T}| < 2$.
2. Record the current cost $z_I^{\text{init}} = z_I$. Then for each trip $t \in \mathscr{T}$ (in arbitrary sequence): remove t from I, then choose and carry out a re-insertion of t in $\{\mathscr{S}_I - \{t\}\}$. To ensure that a re-insertion is found, the exhaustive-search procedure (see Section 7) is always used here.
3. If $z_I < z_I^{\text{init}}$, repeat step 2. Otherwise stop.

8.3. Periodic optimisation

The purpose of the heuristic optimisation procedure described here is to overcome inefficiencies that may accumulate as a result of the incrementally optimal choices made when inserting a series of trips. The procedure is invoked after every k new bookings, where k is a run-time parameter; or by means of an explicit real-time command (e.g. at fixed time-intervals).

A simple steepest-descent improvement strategy is applied. An improvement is defined as a scheduling change involving a *re-location* or *swap*. A re-location involves removing a trip from its current itinerary, and re-inserting it in the same or another itinerary; while a swap involves exchanging a pair of trips between two different itineraries. The optimisation procedure is outlined below (for its efficient implementation see Horn, 1999).

1. Record a set Λ of possible improvements to the schedule. This set comprises every feasible *re-location* or *swap* for which the net cost impact is less than zero.
2. If Λ is now empty, stop.
3. Select the least-cost improving change λ^* from Λ, and implement λ^*.
4. Update Λ so that it correctly records improvements that are possible following the changes made in step 3; and go to step 2.

8.4. Permutation

A procedure to exhaustively permute stops within an itinerary has been implemented on an experimental basis. The procedure can be invoked just before or just after each application of the periodic optimisation described above, and finds the least-cost of all possible permutations of movable stops in a given itinerary, without disturbing trips currently in transit (Horn, 1999). In the contexts tested so far (typically with 5–10 trips per itinerary) it finds few improvements, and very few when the heuristic shuffle is applied. This suggests that the heuristic methods are adequate for practical purposes, so far as the ordering of stops within itineraries is concerned.

9. Rank-homing heuristics

A rank-homing heuristic is a rule governing the deployment of vehicles that have no work currently scheduled. The target of such a deployment is called a Rankloc stop because the heuristics were designed initially with the aim of ensuring adequate attendance to travellers waiting at cab-ranks for TaxiOP service (see Section 3). By clustering vehicles in localities of anticipated demand, the rank-homing mechanism may be useful also in improving the efficiency of phone-booked service modes (see Section 12); thus Rankloc stops may be defined broadly as locations in whose vicinity travel requests are likely to arise.

A set $\mathscr{R}$ of network nodes designated (at least notionally) as cab-ranks is specified in advance. The task addressed by the rank-homing heuristics is to choose a cab-rank $r^*(r^* \in \mathscr{R})$ to which a given "empty" vehicle v is to be dispatched from its current location S. The opportunity to do this can arise either upon the discharge of v's last-scheduled passengers at a setdown stop, or in response to a dispatch-request from a timed-out Rankloc stop (see Section 10). In general then, r^* is to be inserted in a leg $S \rightarrow T$, where $type_S \in \{\text{Setdown, Rankloc}\}$ and $type_T = \text{EndShift}$. The rank-homing heuristics are based on the following scheme:

1. Define $\mathscr{R}' = \mathscr{R}$, or if S is a rank, $\mathscr{R}' = \{\mathscr{R} - \{S\}\}$. The latter exclusion is made because any opportunity for pickup upon arrival at a rank is exploited immediately, thus precluding invocation of the rank-homing heuristic.
2. Define $\mathscr{R}''$ as comprising every rank $r(r \in \mathscr{R}')$ for which the estimated number of waiting trip-requests $nreqs_r$ (see below) exceeds the number of vehicles currently dispatched to r.
3. If $\mathscr{R}'' \neq \emptyset$, define r^* as the nearest rank in $\mathscr{R}''$ to S; otherwise define r^* as the nearest rank to S in $\mathscr{R}'$.

The heuristics apply different conventions to obtain the rank-demand estimates $nreqs_r$:

(a) The *pseudo-omniscient* heuristic obtains $nreqs_r$ directly from the LITRES-2 simulator through a message sent whenever the number of waiting passengers increases or decreases at any $r \in \mathscr{R}$. This might represent a situation where cab-ranks are equipped with push-button call kiosks. Note that this method, unlike (b) below, fails to take into account the delay t^{leg}_{Sr} in getting from S to r.

(b) The *historical-experience* heuristic obtains $nreqs_r$ by reference to a table of estimated aggregate demand at each cab-rank. For each cab-rank $r(r \in \mathscr{R})$, a vector of demand $\{ntrips_{r,t}\}$ is defined in advance for each of a series of times $\mathscr{T}$. For a current time t^{now} and cab-rank r, the value of $nreqs_r$ upon arrival from S is estimated by linear interpolation between $ntrips_{r,t1}$ and $ntrips_{r,t2}$ at time $t_r = t^{\text{now}} + t^{\text{leg}}_{Sr}$, where $t_1, t_2 \in \mathscr{T}$ and $t_1 \leqslant t_r \leqslant t_2$.

When a rank r^* has been selected, it is inserted as a Rankloc stop R in the final leg $S \rightarrow T$, provided that this does not delay arrival at T beyond the end of the shift. The time-window at R is then defined as $\{t_R^{\text{start}} \rightarrow t_R^{\text{end}}\}$, where $t_R^{\text{start}} = t^{\text{now}} + t_{SR}^{\text{leg}}, t_R^{\text{end}} = t_R^{\text{start}} + trmax$, and $trmax$ is a limit on the time a driver is willing to wait at a rank. The vehicle is then dispatched immediately to R.

It may be noted that travel from S to r^* incurs costs in terms of fuel and time but is (by itself) unproductive; furthermore, because $nreqs_r$ is merely an estimate, we cannot be certain that a pickup will be found at or near r^*. It would be logical then for the decision to proceed to r^* to be subject to a criterion embodying the cost of the unproductive leg, and the relative probabilities of finding work at S and r^*. These probabilities could be obtained through generalisation of the demand-estimation methods, for instance (as an extension of the historical-experience heuristic) by providing vectors of estimated demand at every node. A further refinement pertinent to environments where ride-sharing occurs to a substantial extent would be to use the number of passengers (rather than requests) in the assessment of rank-homing prospects.

10. Dispatching of vehicles

10.1. Dispatching strategy

The *visit period* $\{t_S^{\text{arr}*} \rightarrow t_S^{\text{dep}*}\}$ for a stop S is the period when a vehicle is present at S, encompassing (if S is an ordinary pickup or setdown stop) a *handling period* $\{t_S^{\text{arr}'} \rightarrow t_S^{\text{dep}'}\}$ during which actual boarding or disembarkation is to take place. A dispatching strategy involves a set of rules for determining a time $t_S^{\text{dep}*}$ for dispatch from S. Such a strategy may be called *uniform* if it is applied at every stop in the same way, as in the alternatives considered below. For a leg $S \rightarrow T$, determining $t_S^{\text{dep}*}$ implies an arrival-time estimate $t_T^{\text{arr}*} = t_S^{\text{dep}*} + tleg_{ST}$; this leaves open the possibility of using AVL equipment to obtain increasingly accurate leg-time estimates $t_{S'T}^{\text{leg}}, t_{S''T}^{\text{leg}} \ldots$ at intermediate wayside stops $S'S'' \ldots$

The assessment of a dispatching strategy may encompass the allocation of slack time, promptness of service and risk of tardiness. In particular:

(a) When waiting at S occurs *after* the handling period, slack is afforded for the possible insertion of an additional stop or stops subsequent to S. By contrast, any waiting at S *before* the handling period would be available as slack only under a scheduling strategy allowing side-loops before handling S, or by re-allocating S (if a pickup) to a different vehicle. Such a strategy seems inherently inefficient and is not provided for in L2sched.

(b) Promptness refers to the handling of a pickup or setdown early rather than late in the designated time-window $\{t_S^{\text{start}} \rightarrow t_S^{\text{end}}\}$. This is obviously desirable from the travellers' point of view.

(c) A risk of tardiness arises because of uncertainty over the actual travel times on future legs of an itinerary. For example, departure at LDT_S would leave no leeway for longer-than-estimated travel times on the next leg $S \rightarrow T$ or any of its successors: in other words, a delay on $S \rightarrow T$ would cause the vehicle to run late thereafter, with at least one time-window violation unless the delay could be overcome by speeding in another leg.

The "early service" dispatching strategy adopted in L2sched aims to strike an effective balance between the above criteria. In general, the idea is to make the vehicle serving a given stop arrive as

Table 1
Dispatching strategies

Strategy	$t_S^{\mathrm{arr}*}$	$t_S^{\mathrm{dep}*}$	Waiting, if any	Other characteristics
Early service	EAT'_S	$\mathrm{EAT}'_T - t_{ST}^{\mathrm{leg}}$	After t_S^{start}	Prompt, medium risk
Minimum delay	EAT_S	$\mathrm{EAT}'_S + t_S^{\mathrm{serv}}$	Before t_S^{start}	Prompt, low risk
Maximum delay	$\mathrm{LDT}'_S - t_S^{\mathrm{serv}}$	LDT_S	After t_S^{start}	Not prompt, high risk

soon as is necessary to provide the earliest useful service at the stop, and depart as late as possible to maintain similar early service at the next stop. Table 1 defines this strategy and compares it with two simpler (and clearly inferior) approaches.

10.2. Dispatch requests

An explicit dispatch request asks permission from the scheduling system for a given vehicle to proceed immediately to some place other than the vehicle's next scheduled stop. Unlike other scheduling actions, this type of request may be speculative or indeterministic: it might represent a driver's wish to take a lunch-break at a certain place, or to proceed to a place where new business is expected.

Two types of dispatch request are recognised. In the first, L2sched is asked to define a Rankloc destination for the nominated vehicle: this can occur when the waiting period at a current Rankloc stop has expired, and is handled using a rank-homing heuristic (see Section 9). The second type of request involves dispatch to a nominated destination s. This involves constructing a wayside stop w at the current location and inserting s immediately after w: insertion is assessed (see Appendix A), and if feasible is followed by a dispatch to s.

10.3. Contents and timing

The protocol for communication between L2sched and drivers is minimal in that dispatches specify *where* and *when* to go, not *how*: each driver is expected to navigate effectively (i.e. via the fastest path) to the dispatch destination. The protocol is also "short-sighted", in that each driver is given instructions only for the leg currently being traversed (if in transit) or the next leg (if currently waiting at a stop). Each dispatch command is sent as soon as the need for it becomes known, and without subsequent duplication. Upon arrival at a stop S, vehicle veh_I notifies L2sched, which replies immediately with a dispatch command for the next leg $S \rightarrow T$. This may be superseded by another dispatch command at any time prior to the arrival of veh_I at T, as the result of operations affecting the timing or sequencing of $S \rightarrow T$ (e.g. trip-insertions, trip-removals, or optimisations). With reference to the insertion types distinguished earlier (see Section 7), a future insertion may cause a re-dispatch before departure from S, while a pre-emptive insertion involves a re-dispatch while in transit on $S \rightarrow T$.

11. Contingencies

A contingency is a real-time event or condition that is inconsistent with a prediction made by L2sched about the behaviour of a vehicle or a group of passengers. When L2sched detects a contingency, it must at least adjust timing and other variables so as to bring its representation of itineraries and other entities into a consistent state, even when the contingency involves a violation of scheduling constraints (e.g. late arrival at a stop). Beyond this, there may be opportunities to adjust vehicle deployments so as to eliminate infeasibilities induced or threatened by the contingency. The adjustments, which are outlined below, are independent of the dispatching policy outlined in Section 10.

11.1. Arrival and departure times

When a vehicle represented by an itinerary I arrives at a stop S it is possible (indeed likely) that the arrival time $t_S^{\mathrm{arr}*}$ does not correspond exactly with the arrival time EAT_S anticipated when the vehicle was dispatched from $sprev_S$. A discrepancy of this kind may require no more than a resetting of EAT_S to $t_S^{\mathrm{arr}*}$, with a forward-propagation of EAT_S along S_I to reflect the consequent increase or decrease in slack time. Beyond this however it is possible that the arrival time is so late as to fail the time window at S (with $t_S^{\mathrm{arr}*} > t_S^{\mathrm{end}} - t_S^{\mathrm{serv}}$), or to threaten the feasibility of service at S's successors ($t_S^{\mathrm{arr}*} > \mathrm{LDT}_S - t_S^{\mathrm{serv}}$), or both. Here again the revised EAT must be propagated forward along $\mathcal{S}_I$, and it may be possible to avert threats to feasibility by transferring one or more trips from I into other itineraries.

A discrepant departure time $t_S^{\mathrm{dep}*}$ from a stop S is one for which the implied arrival time at the next stop T is discrepant in the sense outlined above. The implied arrival time is estimated as $t_T^{\mathrm{arr}*} = t_S^{\mathrm{dep}*} + t_{ST}^{\mathrm{leg}}$, and if it differs from the current estimate for EAT_T, T and its successors are treated just as they would in the case of problematic arrival at T.

11.2. Unexpected locations

A locational contingency occurs when the location reported for a vehicle (e.g. by an AVL system) is unexpected. A stop at an unexpected location would imply a human misunderstanding or faulty equipment, requiring action within a wider context than that of L2sched. When a vehicle is en route between stops, a locational discrepancy is defined as a mismatch between the reported location and the wayside location estimated for the current time t^{now}. This can be handled by re-estimating the arrival time at the next stop, followed (if necessary) by the application of the arrival-time adjustments described above.

11.3. Trip-cancellations

The nature of a trip-cancellation depends on the location of the vehicle assigned to the cancelled trip, relative to the trip's pickup stop when the cancellation order is received. In summary, the vehicle may be anterior to the pickup stop (a simple cancellation), or at the pickup (a no-show), or it may have made the pickup already (a truncated trip). In each case, the trip must be removed from the vehicle's itinerary, and if the vehicle is headed directly for (or located at) the pickup or setdown stop of the cancelled trip, a wayside stop must be inserted in the itinerary to mark the vehicle's location. These changes are followed by adjustments to the timing variables (EAT and LDT) in what remains of the itinerary, so as to reflect the increase in slack time arising from the cancellation.

11.4. Breakdowns

The breakdown of a vehicle is handled by treating the trips that were assigned to it as trip-insertion requests on the now-reduced fleet. The breakdown location is identified as a wayside stop in the vehicle's itinerary, and attempts are then made to provide alternative service for its current trips (i.e. passengers currently on board) and for all trips that it is scheduled to pick up in the future. For trips that cannot be re-allocated in this way, the setdown windows are stretched progressively (e.g. in 5 min increments); and any trip that is still unserviced when the extension reaches a pre-defined upper limit (e.g. 30 min) is abandoned.

12. Simulation tests

This section reports on a series of tests that have been designed to explore procedural aspects of L2sched, which for simulation purposes is embedded within the LITRES-2 modelling framework (see Section 1). The focus here is on system components that are directly concerned with scheduling efficiency. Of the other components, contingency-handling and dispatching have proved robust in practice, and their logic appears to be well justified on a priori grounds. The rank-homing heuristics are discussed at the end of this section.

The tests are based on data collected for an earlier study of taxi operations in the Gold Coast, an urban region of southern Queensland (Horn et al., 1999). That study modelled an existing taxi fleet providing both phone-booked and rank-based service, as a basis for investigating fleet utilisation during the course of the day and options such as multiple hiring. The present tests are focussed on more technical issues, and consequently they involve significant departures from the realism sought in conventional case studies.

Demand data were obtained from booking records of the Regent Taxi Company, the main taxi operator in the Gold Coast area. These records cover a single 24-h period (1 December 1999 with 4282 trips), which for scaling purposes were aggregated to form origin-destination (OD) matrices at hourly intervals; when a simulation is run, the matrices are disaggregated using randomisation techniques so as to generate a time-ordered stream of travel-requests for input to L2sched. The Regent fleet comprises 220 taxis, run in three shifts, with temporally staggered shift changeovers. The road network model has 1782 nodes. The tests were conducted on a Sun UltraSparc workstation.

The original OD matrices (see above) were scaled up by a factor of two, in order to study the scheduler's operation under fairly heavily loaded conditions; the effective load in fact is heavier in the single-hire cases, as discussed below (the measurement of passenger load and fleet capacity is discussed in Horn et al., 1999). Initial experiments showed that in single-hire mode the fleet could accommodate approximately 95% of this scaled-up demand with an upper limit of 15 min on waiting time at pickup. The remaining cases (i.e. "difficult" trip-requests) were accommodated by means of the "stretched window" mechanism described in Appendix C, so as to standardise loadings and thus to permit comparisons of results obtained in each of the four scenarios tested. The standardisation between scenarios is not quite exact, due to edge-effects in trip-generation at the start and end of the simulation period. In particular, 8564 trips were handled in the zero-notice scenarios, while in the advance-notice scenarios the number of trips is 8462. Differences between the sets of trips tested in the two scenarios have led also to a slight discrepancy between the average time per trip (10 min and 46 s as compared with 10 min and 47 s).

The test scenarios embody differing assumptions about two major conditions of fleet operation, namely hiring conventions and advance notice of travel. These factors have significant influence on potential efficiency of fleet deployment: multiple-hiring augments opportunities for productive deployment (e.g. using one instead of two vehicles for a pair of trips with requirements in common), while advance notice helps to overcome mismatches between demand and supply (e.g. for service to a pickup stop located far from any vehicle at the time when a request is issued). In a more general sense, the number of possible implementations for a set of trips – hence the opportunities for optimisation and prompt service – is greater (other things being equal) when multiple hiring is permitted and bookings are made in advance, as compared with single hiring and immediate bookings, respectively.

In each of the scenarios the fleet comprises the base-case fleet of 220 vehicles, with a capacity of four passengers each. Every trip-request is phone-booked, and pertains to a single passenger (i.e. $npass_t = 1$ for each trip). The scenarios represent the four possible combinations of single versus multiple hiring, and immediate service versus advance notice. They are referred to below through the use of acronyms composed of the letters S/M and Z/N (Single/Multiple, Zero/Notice); thus the fleet is exclusively single-hire in Scenarios SZ and SN, and exclusively multiple-hire in Scenarios MZ and MN. Scenarios SZ and MZ assume that every passenger wishes to depart immediately after the time t when travel is requested (i.e. the pickup window commences at time t in every case). By contrast, Scenarios SN and MN assume that after the issue-time t there is a notice-period of average duration 30 min, the specific duration of this period being obtained by uniform random interpolation between zero and 60 min.

The parameters explored in each scenario are listed in Table 2. Of the results reported in Tables 3–6, the average travel distance per vehicle (directly related to running costs) is a measure of fleet efficiency, while average travel times per passenger provide some insight into the behaviour of the fleet. The latter is constant in the single-hire scenarios for obvious reasons. In the multiple-hire scenarios the travel time provides an indication of ride-sharing; for example, the proportion of an "average trip" shared with others in run MN.01 is 38% (i.e. 1 – 10:47/17:21 = 1 – 647/1041). In these tests the quality of service (e.g. the amount of time spent waiting after the commencement of

Table 2
Parameters for tests

Category	Section reference	Abbreviation	Description
Objective function	5	F*rfact*	Value of rfact in Eq. (5.4). For time-based objective, $rfact = 0$. For load-weighted objective, $0 < rfact \leqslant 1$.
Trip-insertions	7	E	Exhaustive-search insertion.
		H	Heuristic insertion.
		P	Pre-emptive insertion.
Post-insertion heuristics	8	T	Transfer.
		S	Shuffle.
Periodic improvement	8	I*n*	Improvement is applied after every *n* insertions.

Table 3
Scenario SZ: TaxiSingle fleet, without notice

Run	Parameters see Table 2	Travel distance (km)/per vehicle	CPU (min:s)
SZ.01	F0 E	943.1	2:21
SZ.02	F0 H	940.4	2:12
SZ.03	F0 E P	939.3	2:33
SZ.04	F0 E T	938.6	2:33
SZ.05	F0 E S	939.3	2:44
SZ.06	F0 E TS	943.1	2:21
SZ.07	F0 E I50	927.9	6:06
SZ.08	F0 E I100	930.2	4:29
SZ.09	F0 E I150	934.7	3:38
SZ.10	F0 E TS I100	928.9	4:51
SZ.11	F0.5 E	940.1	2:15
SZ.12	F0.5 E I100	932.4	4:22
SZ.13	F0.5 E TS I100	930.2	5:04

Note: average travel time is 10 min and 46 s in each run.

Table 4
Scenario SN: TaxiSingle fleet, with notice

Run	Parameters see Table 2	Travel distance (km)/per vehicle	CPU (min:s)
SN.01	F0 E	950.6	2:31
SN.02	F0 H	949.4	2:24
SN.03	F0 E P	939.3	5:09
SN.04	F0 E T	948.9	2:46
SN.05	F0 E S	914.4	4.21
SN.06	F0 E TS	915.0	3:01
SN.07	F0 E I50	897.1	26:01
SN.08	F0 E I100	898.5	17:48
SN.09	F0 E I150	901.4	14:25
SN.10	F0 E TS I100	893.9	17:44
SN.11	F0.5 E	951.2	2:35
SN.12	F0.5 E I100	900.0	18:11
SN.13	F0.5 E TS I100	893.3	17:42

Note: average travel time is 10 min and 47 s in each run.

the pickup window) is assumed to be expressed adequately in the time-window and other constraints (waiting time is considered briefly in connection with the supplementary tests of blue-ribbon service, below). The CPU figures reported for each run are actual execution times (system plus user), and do not include time consumed in the simulator module or in setting up the scheduler.

12.1. Aggregate performance

A broad comparison between the different scenarios bears out the observations made earlier about the role of ride-sharing conventions and advance notice. The shortest per-vehicle dis-

Table 5
Scenario MZ: TaxiMulti fleet, without notice

Run	Parameters see Table 2	Travel distance (km)/per vehicle	Travel time (min:s)/per trip	CPU (min:s)
MZ.01	F0 E	681.8	15:13	2:31
MZ.02	F0 H	688.8	15:10	2:24
MZ.03	F0 E P	679.2	15:13	5:09
MZ.04	F0 E T	670.9	15:42	2:46
MZ.05	F0 E S	687.2	15:12	4.21
MZ.06	F0 E TS	686.0	15:09	3:01
MZ.07	F0 E I50	663.8	15:26	6:18
MZ.08	F0 E I100	676.7	15:24	4:31
MZ.09	F0 E I150	681.8	15:14	3:49
MZ.10	F0 E TS I100	665.8	14:28	3:22
MZ.11	F0.5 E	673.1	14:32	2:34
MZ.12	F0.5 E I100	657.5	14:36	4:40
MZ.13	F0.5 E TS I100	646.8	14:36	5:27

Table 6
Scenario MN: TaxiMulti fleet, with notice

Run	Parameters see Table 2	Travel distance (km)/per vehicle	Travel time (min:s)/per trip	CPU (min:s)
MN.01	F0 E	611.2	17:21	2:31
MN.02	F0 H	622.5	17:20	2:24
MN.03	F0 E P	530.6	17:13	5:09
MN.04	F0 E T	616.4	17:22	2:46
MN.05	F0 E S	531.1	17:17	4:21
MN.06	F0 E TS	614.4	17:18	3:01
MN.07	F0 E I50	493.1	17:19	46:40
MN.08	F0 E I100	507.8	17:19	33:17
MN.09	F0 E I150	516.6	17:18	25:58
MN.10	F0 E TS I100	497.3	17:14	29:55
MN.11	F0.5 E	586.8	15:58	2:45
MN.12	F0.5 E I100	493.8	15:46	34:26
MN.13	F0.5 E TS I100	484.7	15:49	30:51

tance found for Scenario SZ (in run SZ.07, see Table 3) is 91% longer than the corresponding distance in Scenario MN (run MN.13 in Table 6), and the scope for optimisation is similarly differentiated. Measuring this scope as the difference between the shortest and longest travel distance obtained in each scenario, it is 15.2 km in Scenario SZ (runs SZ.01 and SZ.07), 57.3 km in Scenario SN (SN.01, SN.13), 42.0 km in Scenario MZ (MZ.02, MZ.13) and 137.8 km in Scenario MN (MN.02, MN.13). These figures provide at least a preliminary justification for the more elaborate scheduling procedures described in this paper, even under inflexible conditions such as Scenario SZ; for example, if running costs are \$1 per kilometre, the daily fleet-wide savings in the four scenarios are \$3,344, \$12,606, \$9,240 and \$30,316, respectively.

12.2. Trip-insertion procedures

The tests show a small but perhaps consistent differentiation in performance between the two main insertion procedures: the *heuristic* procedure produces shorter aggregate distances in the single-hire scenarios (Tables 3 and 4, runs SZ.01, SZ.02, SN.01, SN.02), while *exhaustive-search* is superior in the multiple-hire cases (Tables 5 and 6, runs MZ.01, MZ.02, MN.01, MN.02). Because the procedures are applied incrementally (see Section 6), there is no reason to expect superior performance from (say) the exhaustive-search procedure under all conditions; it is an interesting question whether the differentiation found here has any fundamental significance. *Pre-emptive* insertion produces significant improvements in fleet efficiency, especially in Scenario 4 (run MN.03 in Table 4). In the single-hire scenarios (runs SZ.03 and SN.03 in Tables 2 and 3), the improvement is small because of the limited scope for pre-emption in this case (a pre-emptive insertion is possible here only if the whole "pre-empting" trip can be served without delaying the vehicle's arrival at the stop to which it was already heading). To simplify analysis the remaining tests apply exhaustive search without pre-emption, but future work might well be addressed to other combinations involving heuristic and pre-emptive insertion.

12.3. Post-insertion procedures

The post-insertion procedures were exercised separately and in combination in runs SZ.04-SZ.06 et cetera (the baseline cases here are runs SZ.01, etc.). Their impact appears to be fairly small but generally positive, except for some elementary cases in Scenarios MZ and MN (runs MZ.05, MZ.06, MN.04, MN.06). The *transfer* heuristic outperforms the alternatives in the zero-notice scenarios, but is of doubtful value elsewhere. Conversely, the *shuffle* heuristic is very effective in the advance-notice scenarios, but less so in the others. In combination (i.e. in run SZ.06, etc.), the two heuristics generally have an advantage over the simpler cases, except in Scenario MN (Table 6, cf. runs MN.01 and MN.06). This advantage is sustained consistently when the two are combined with periodic optimisation (e.g. in Table 6, compare runs MN.08 and MN.10, and runs MN.12 and MN.13).

12.4. Periodic improvement

The periodic improvement procedure was applied at intervals of 50, 100 and 150 trip-insertions in each scenario (runs SZ.07-SZ.09, etc.), significantly reducing travel distances in every scenario, especially those involving advance notice. This reduction was however won at the cost of higher CPU times, notably in Scenario MN (see Table 6). For practical purposes, the critical figures here are the durations of individual applications of the improvement procedure, each such application representing a pause while L2sched waits for completion of the procedure. Examination of detailed results shows that the pauses are longest in the run where improvement is applied least frequently (MN.09), presumably because in this case there is a correspondingly high level of excess distance to be dealt with by each improvement run. Here the range of pause-durations during periods of substantial passenger loadings lies between 20 and 48 s; the corresponding range in run MN.07 is 10–28 s. Pauses of this magnitude in practice would very probably lead to problems of synchronisation between the scheduler and the outside world; such problems may however be avoided by providing the improvement module with an interrupt mechanism.

12.5. The objective function

Three tests were run in each scenario (runs SZ.11-SZ.13 etc.) in order to test the impact of the load-weighted objective, as opposed to the simple time-minimising objective applied elsewhere (see Section 5). The results generally confirm the *a priori* expectation of advantages obtainable from the load-weighted objective in a multiple-hire context. In particular, in Scenarios MZ and MN this objective produces both more an efficient deployment of the fleet (i.e. shorter travel distances) and a lower incidence of ride-sharing (i.e. shorter travel times per passenger) than the time-minimising objective. The relatively low rate of ride-sharing indicates that (at least in these tests) the load-weighted objective is achieving efficient deployment not by inducing higher vehicle occupancies, but by drawing into play vehicles that would otherwise be under-utilised (e.g. empty vehicles whose location in outlying areas places them at a "competitive disadvantage" with respect to other vehicles, as discussed in Section 5). For the single-hire scenarios (SZ and SN), the two objective functions are approximately equivalent in terms of fleet efficiency. This result appears to justify the use of the load-weighted objective in mixed-mode fleets (e.g. there seems to be no need to tailor the objective to individual trip-insertions in such cases).

12.6. Blue-ribbon service

Four additional test runs were designed to assess the impact of the blue-ribbon service convention (see Appendix C), when applied to the main test scenarios. Table 7 shows distance, time, and CPU results from the blue-ribbon tests, together with a column showing average waiting times for blue-ribbon service and for the counterpart "baseline" runs (i.e. SZ.01 et cetera). The baseline times (parenthesised in Table 7) may be regarded as typical of the waiting times in each of the four test series reported in Tables 3–6 (e.g. in Scenario SZ the baseline value is 12:21, and the range of average waiting times is 12:16 to 12:31). The effectiveness of the blue-ribbon convention in minimising waiting time (see column 3 of Table 7) is not surprising, since the convention applies this criterion explicitly; by contrast, the insertion techniques used in the main tests are indifferent to the duration of waiting time, provided it does not violate the 15 min service standard, applying a blue-ribbon style rule beyond this limit.

Table 7
Blue-ribbon tests

Run	Travel distance (km)/per vehicle	Waiting time[a] (min:s)/per trip	Travel time (min:s)/per trip	CPU (min:s)
SZ.00	893.1	6:06 (12:21)	10:46	2:39
SN.00	960.9	0:47 (8:21)	10:47	2:25
MZ.00	893.1	6:06 (8:21)	10:46	2:59
MN.00	800.7	1:01 (6:48)	11:55	2:59

[a] The figures in parenthesis are "baseline" waiting times, from runs SZ.01, SN.01, MZ.01 and MN.01.

The freezing of blue-ribbon time windows and the consequent loss of scheduling flexibility explain the relatively long travel distances in three of the four blue-ribbon runs (see SN.00, MZ.00 and MN.00 in Table 7, compared with SN.1, MZ.1 and MN.1 in Tables 4–6). There is however an anomaly apparent in the short aggregate distance obtained in the single-hire, zero-notice case (893.1 km in run SZ00, compared with 940.4 km in run SZ.01). This may reflect a strategic benefit obtained from "early activation" of vehicles under peak loading conditions; that is, the blue-ribbon convention forces demand to be dealt with as soon as possible, so (perhaps) maximising the flexibility of response available at any given time. The processes involved in this case merit more detailed investigation, given the pervasiveness in practice of the conventions modelled (i.e. blue-ribbon service for single-hire taxis, with little advance notice of travel). Further enquiry might be addressed also to the impact of the waiting time limit in the stretchable service standard, of which blue-ribbon may be regarded as simply a limiting case (see Appendix C).

12.7. Further investigations

The simulation tests may be regarded as having indicative value with respect to both the usage of the scheduling procedures and several points requiring further investigation. The foregoing analysis is admittedly based on fairly primitive assumptions regarding fleet composition, and has examined outcomes only at an aggregate level (ignoring, for example, the incidence of excessively stretched time-windows at different times of day). A more nuanced view of such issues will clearly be desirable when L2sched is tuned for application in a specific urban context.

It may be noted finally that further research on the rank-homing procedures might be valuable in situations quite apart from conventional rank-style pickup conventions. In particular, such a procedure may provide a broadly applicable mechanism for bringing supply within effective striking range of demand. This assertion is supported by preliminary tests indicating that the *historical-experience* heuristic has the effect of keeping uncommitted vehicles in intermittent motion across the study region, thus avoiding situations where vehicles may become stranded in remote locations (see also the foregoing remarks on the quadratic objective). This motion in turn tends to bring more vehicles into play in areas where they are most needed, at the cost (in the advance-notice scenarios) of greater aggregate travel distances. Further research might well be directed at ways of overcoming these difficulties, and (in broader terms) of obtaining a more precisely forward-looking mobilisation of the fleet (e.g. see the rank-homing extensions canvassed at the end of Section 9).

13. Conclusions

The tests reported in the preceding section show that the improvement procedures yield substantial operational efficiencies over more naïve scheduling methods, and that these efficiencies are augmented when the procedures are used in combination. The tests also show that L2sched makes modest use of computational resources under operating conditions approximating those of a large demand-responsive fleet, while the provisions for handling contingencies underwrite the system's robustness in such conditions. There is thus good reason to suppose that the system will be effective in an operational setting. A practical application will of course entail integration with real-time booking and communications modules, and a systematic tuning of the optimisation parameters. It remains now to consider some broader avenues for future investigation.

13.1. Optimisation procedures

For periodic optimisation it may be worthwhile to investigate procedures with a wider scope of search than those currently implemented in L2sched. Research on the Vehicle Routing Problem suggests several possible starting points for work in this respect, including exact methods (Desrochers et al., 1992; Laporte, 1992), and heuristic approaches such as tabu search (Gendreau et al., 1994), simulated annealing (Osman, 1993), genetic algorithms (Potvin and Bengio, 1994) and guided local search (Kilby et al., 1997). It is obvious however that in urban situations like those for which L2sched is intended, and with similar hardware, such procedures would entail much longer execution times than those required for the periodic-improvement procedure used at present; yet the latter is already somewhat problematical in this respect (see Section 12). In practice then, the application of more wide-ranging search procedures will probably be limited to "overnight" optimisation of bookings made in advance, and the value of such an approach will depend on the proportion of trips for which such notice is given.

13.2. Scheduling objectives

The formulation of the objective function may also merit further investigation. The load-weighted objective function is "heuristically" formulated, but it can be justified on broad *a priori* grounds, and its impact has been shown to be favourable in the tests carried out so far (see Sections 5 and 12). A fundamental research task here would be to obtain a firm theoretical basis for such an objective, by extending the reasoning in Section 5 to a probabilistic analysis of the current and future distributions of trip-requests and vehicles in space and time. As a simpler alternative, it may be useful to conduct further simulations to investigate the effectiveness of the load-weighted form defined here under a variety of conditions (e.g. different networks, different fleets, different fleets, different times of day), so as to facilitate detailed tuning of the weighting parameter *rfact* (see Eq. (5.4)); a possibility in this respect would be to apply different values of this parameter at different times of the day on the basis of expectations of aggregate passenger loading conditions (e.g. peak versus off-peak).

A second possibility would be to build a measure of "driver equity" into the objective function. This reflects dispatching practice in many taxi fleets, where informal efforts are made to allocate work such that over time, all drivers are kept more or less equally busy (see Shrivastava et al., 1997). The quadrant function (Eq. (5.4)) favours equity in this sense, although in a somewhat indirect fashion, and not necessarily in all scheduling situations (see Section 5).

A third area of interest concerns the treatment of the temporal quality of the service provided to passengers. In this respect it is interesting to compare L2sched with the work of Jaw et al. (1986), where a compound objective function includes "disutility" components for both the operator and passengers, with eight different weighting parameters. By contrast, L2sched involves at most two "heuristic" methodological choices (see Section 5), without reference to the service quality required by passengers. Provision however is made for variants to this arrangement, notably a "blue-ribbon" service, and a progressive stretching of time-windows under near-saturated conditions (see Appendix C). A further variant might involve the application of a duration constraint to travel time as such (Horn, 1999).

13.3. Condensed time-windows

The scheduling and dispatching conventions currently applied in L2sched can lead to indistinctnesses in travel timing which some travellers are likely to find unattractive. For example, having specified pickup and setdown time-windows {10:00:00–10:30:00} and {10:15:00–10:45:00}, respectively, for a 15-min taxi trip, a traveller may find it unacceptable to be told that his/her presence is required at the pickup stop for up to 30 min. This kind of situation could be rectified at the time of booking by guaranteeing reduced pickup and setdown windows $\{t_{\mathrm{PU}}^{\mathrm{start}*} \rightarrow t_{\mathrm{PU}}^{\mathrm{end}*}\}$ and $\{t_{\mathrm{SD}}^{\mathrm{start}*} \rightarrow t_{\mathrm{SD}}^{\mathrm{end}*}\}$, to replace the windows originally specified by the traveller. From the traveller's point of view, the ideal windows in this respect would be those implied by the booking's initial implementation (e.g. $t_{\mathrm{PU}}^{\mathrm{start}*} = \mathrm{EAT}'_{\mathrm{PU}}$ and $t_{\mathrm{PU}}^{\mathrm{end}*} = \mathrm{EAT}'_{\mathrm{PU}} + t_{\mathrm{PU}}^{\mathrm{serv}}$). Because "freezing" the condensed windows in this way has the effect of reducing slack, it would be reasonable to impose a fare surcharge for such guarantees. A generalisation of this approach would embody a yield-management principle, with the fare based on the time of day, the duration of travel, and the locations of the origin and destination. The aim would be to match the fare to the actual cost of providing the service, so as to encourage efficient use of the system.

Acknowledgements

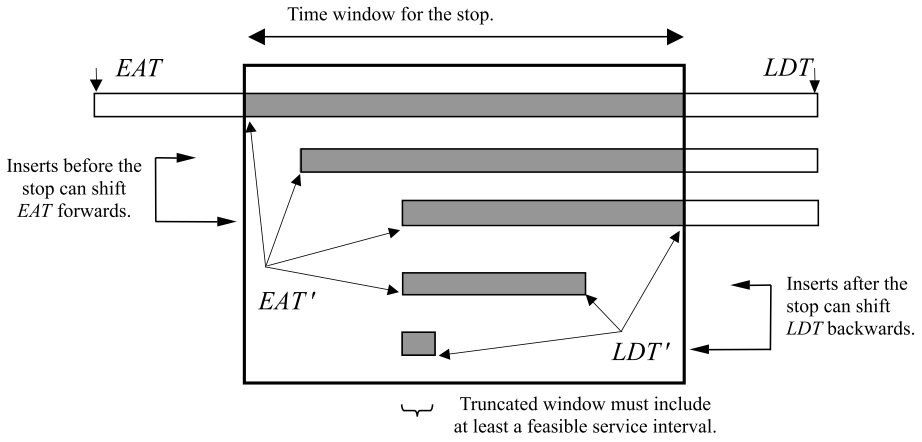
The author thanks his CSIRO colleagues Peter Lamb and Phil Kilby, and the anonymous reviewers, for valuable comments on an earlier draft of this paper.

Appendix A. Constraints and their application

This appendix outlines the concepts and methods used by L2sched in maintaining feasibility, that is, in ensuring that passengers are picked up and delivered in accordance with the requirements expressed in trip-requests. We note first of all that if the temporal conditions outlined in Section 3 are to be satisfied in a stop-list $\mathscr{S}_I$, any insertion of a stop S in a leg $R \rightarrow T$ of $\mathscr{S}_I$ must make possible an arrival time t_S^{arr} and departure time t_S^{dep} consistent with the following conditions, both at S and at every other stop in $\mathscr{S}_I$.

$t_S^{\mathrm{arr}} + t_S^{\mathrm{serv}} \leqslant t_S^{\mathrm{end}}$	A feasible service interval must follow arrival.
$t_S^{\mathrm{start}} + t_S^{\mathrm{serv}} \leqslant t_S^{\mathrm{dep}}$	A feasible service interval must precede departure.
$t_S^{\mathrm{arr}} + t_S^{\mathrm{serv}} \leqslant t_S^{\mathrm{dep}}$	A service interval must be possible between arrival and departure.
$t_R^{\mathrm{dep}} + t_{RS}^{\mathrm{leg}} \leqslant t_S^{\mathrm{arr}}$	S must be reachable by its nominated arrival time.
$t_S^{\mathrm{dep}} + t_{ST}^{\mathrm{leg}} \leqslant t_T^{\mathrm{arr}}$	T must be reachable by its nominated arrival time.

These conditions have recursive ramifications because changes to the arrival or departure times at any stop can affect the feasibility of its predecessors or successors. The situation is substantially



Time window for the stop.
EAT
LDT
Inserts before the stop can shift EAT forwards.
EAT'
LDT'
Inserts after the stop can shift LDT backwards.
Truncated window must include at least a feasible service interval.

Fig. 4. Time constraints at a stop.

simplified by defining at each stop the EAT, LDT, and related variables introduced in Section 3 (see also Fig. 4). These variables are defined as follows:

(a) EAT_S is the earliest possible arrival time at S if arrivals at all preceding stops are also as early as possible. EAT'_S is the earliest feasible commencement of a service interval at S.

$$\mathrm{EAT}_S = \max(t_R^{\mathrm{start}}, \mathrm{EAT}_R) + t_R^{\mathrm{serv}} + t_{RS}^{\mathrm{leg}}, \tag{A.1}$$
$$\mathrm{EAT}'_S = \max(t_S^{\mathrm{start}}, \mathrm{EAT}_S). \tag{A.2}$$

(b) LDT_S is the latest possible departure time from S in order to provide feasible service to succeeding stops. LDT'_S is the latest feasible termination of a service interval at S.

$$\mathrm{LDT}_S = \min(t_T^{\mathrm{end}}, \mathrm{LDT}_T) - t_T^{\mathrm{serv}} - t_{ST}^{\mathrm{leg}}, \tag{A.3}$$
$$\mathrm{LDT}'_S = \min(t_S^{\mathrm{end}}, \mathrm{LDT}_S). \tag{A.4}$$

We now outline feasibility tests for the proposed insertion of a trip t with stops S_{PU} and $S_{\mathrm{SD}}(S_{\mathrm{PU}} = S_{\mathrm{PU}(t)}, S_{\mathrm{SD}} = S_{\mathrm{SD}(t)})$ in an itinerary I, with reference to the following additional quantities:

$npshare_t$	Maximum number of other passengers sharing travel with trip t: if $mode_t$ is single-hire, $npshare_t = 0$; otherwise $npshare_t = cap_I - npass_t$.
$npres_S$	Number of seats reserved on departure from S: if operating single-hire, $npres_S = cap_I$; otherwise $npshare_t = npass_S$.
lat_S	Latest feasible arrival time at $S \in \{S_{\mathrm{PU}}, S_{\mathrm{SD}}\} : lat_S = t_S^{\mathrm{end}} - t_S^{\mathrm{serv}}$.

The insertion can be ruled out immediately if $mode_t \notin \mathscr{M}_I$ or $cap_I < npass_t$. After this, the task is to establish a feasible insertion of S_{PU} in $\mathscr{S}_I$, and then a subsequent insertion of S_{SD}, the search in both cases proceeding along $\mathscr{S}_I$ in chronological sequence (see Section 7). The tests for an insertion of S_{PU} in a leg $R \to T$ of $\mathscr{S}_I$ are as follows:

1. If $\mathrm{EAT}_R > lat_{S_{\mathrm{PU}}}$, the vehicle cannot reach R by a feasible time to service S_{PU}, so the insertion of S_{PU} in $R \to T$ or in any subsequent leg of $\mathscr{S}_I$ would be infeasible.

2. If $npshare_t > npres_R$, this insertion would be infeasible.
3. Calculate $\mathrm{EAT}_{S_{\mathrm{PU}}}$; then if $\mathrm{EAT}_{S_{\mathrm{PU}}} > lat_{S_{\mathrm{PU}}}$, this insertion would be infeasible.
4. Calculate $\mathrm{LDT}_{S_{\mathrm{PU}}}$; then if $\mathrm{LDT}_{S_{\mathrm{PU}}} < \mathrm{EAT}'_{S_{\mathrm{PU}}} + t^{\mathrm{serv}}_{S_{\mathrm{PU}}}$, this insertion would be infeasible.

Similarly, the tests for insertion of $\mathscr{S}_{\mathrm{SD}}$ in S_I are as follows, assuming that a feasible insertion-leg $R^* \to S^*$ has been found in $\mathscr{S}_I$ for the pickup S_{PU}. The insertion to be tested is for a leg $R \to T$, where $R \in \{S_{\mathrm{PU}}, S^*, snext_{S^*} \ldots\}$, and $S \in \{S^*, snext_{S^*} \ldots\}$, with a re-calculated EAT carried forward over the putative successors of S_{PU}, (i.e. $S^*, snext_{S^*} \ldots R$).

1. If $\mathrm{EAT}_R > lat_{S_{\mathrm{SD}}}$, the vehicle cannot reach R by a feasible time to service S_{SD}, so the insertion of S_{SD} in $R \to T$ or in any subsequent leg of $\mathscr{S}_I$ would be infeasible.
2. If $npshare_t > npres_R$, the vehicle could not carry t past R without becoming overloaded; so the insertion of S_{SD} in $R \to T$ or in any subsequent leg would be infeasible.
3. Calculate $\mathrm{EAT}_{S_{\mathrm{SD}}}$; then if $\mathrm{EAT}_{S_{\mathrm{SD}}} > lat_{S_{\mathrm{SD}}}$, this insertion would be infeasible.
4. Calculate $\mathrm{LDT}_{S_{\mathrm{SD}}}$; then if $\mathrm{LDT}_{S_{\mathrm{SD}}} < \mathrm{EAT}'_{S_{\mathrm{SD}}} + t^{\mathrm{serv}}_{S_{\mathrm{SD}}}$, this insertion would be infeasible.

Appendix B. Removing and inserting trips

B.1. Trip-insertion

The insertion of a trip t in an itinerary I follows the identification of insert locations for S_{PU} and $S_{\mathrm{SD}}(S_{\mathrm{PU}} = S_{\mathrm{PU}(t)}, S_{\mathrm{SD}} = S_{\mathrm{SD}(t)})$, using one of the insertion-finding procedures outlined in Section 7. The insertion is carried out as follows.

1. Insert S_{PU} and S_{SD} in their chosen positions in $\mathscr{S}_I$.
2. Propagate changes to EAT and LDT from each inserted stop $S \in \{S_{\mathrm{PU}}, S_{\mathrm{SD}}\}$:
   (a) EAT on stops subsequent to S may be delayed. Changes to EAT are therefore propagated forward along S_I, until the amount of the change decays to zero.
   (b) LDT on stops preceding S may need to be advanced. Changes to LDT are therefore propagated backward along $\mathscr{S}_I$, until the amount of the change decays to zero.
3. Define values of the passenger-loading variables initially as
4. $npass_S = npass_R$, $npres_S = npres_R$, for $S \in \{S_{\mathrm{PU}}, S_{\mathrm{SD}}\}$, where $R = sprev_S$. Then adjust for each stop S' in $\mathscr{S}_I$ from S_{PU} up to and including $sprev_{\mathrm{SD}}$, as follows: $npass_{S'} = npass_{S'} + npass_t$ and $npres_{S'} = npres_{S'} + cap_I - npshare_t$.

B.2. Trip-removal

The procedure to remove a trip t from an itinerary I is essentially the converse of trip-insertion as outlined above. It involves the following steps:

1. Remove t's pickup and setdown stops $\{S_{\mathrm{PU}}, S_{\mathrm{SD}}\}$ from S_I.
2. Propagate changes to EAT and LDT forward and backward from the successors and predecessors of S_{PU} and S_{SD}.
3. Adjust the values of $npass_S$ and $npres_S$ on each stop between S_{PU} and S_{SD}.

Appendix C. Traveller-oriented objectives

L2sched assumes generally that the urgency of travel is embodied in the time-windows specified for a trip's pickup and setdown stops, these windows being based on mode-specific service standards (see below). By contrast, an "early-service" criterion involves choosing insertions so that travellers arrive as early as possible at their destinations. That is, when inserting a trip t the objective is to minimise $\mathrm{EAT}'_{S_{\mathrm{SD}(t)}}$: for single-hire travel this obviously entails minimising also $\mathrm{EAT}'_{S_{\mathrm{PU}(t)}}$. The difference that this can make when scheduling even a conventional taxi fleet is illustrated in Fig. 5, which shows current itineraries for two vehicles V_1 and V_2, bound for setdowns at a and b, respectively, and thence for nearby cab-ranks. Suppose now that a call is received requesting pickup from point p. Under a travel-time minimising objective the new trip would be allocated to V_1, but it is apparent that the pickup time at p would be earlier if the trip were carried by V_2.

For a given mode m, the service standards comprise a maximum waiting time $twmax_m$ and a travel-factor $kmax_m$ (Horn, 1999). For a trip $i \to j$ with minimum travel duration $tt_{ij}^{\min}$, these parameters determine the following service-based upper limits on the pickup and setdown times:

$$t_{S(i,m)}^{\mathrm{end}} = t_{S(i)}^{\mathrm{start}} + t_{S(i)}^{\mathrm{serv}} + twmax_m, \text{ and} \tag{C.1}$$

$$t_{S(j,m)}^{\mathrm{end}} = t_{S(i)}^{\mathrm{start}} + t_{S(i)}^{\mathrm{serv}} + twmax_m + tt_{ij}^{\min} kmax_m + t_{S(j)}^{\mathrm{serv}}. \tag{C.2}$$

Then for limits $\{t_{S(i)}^{\mathrm{end}}, t_{S(j)}^{\mathrm{end}}\}$ specified in the trip-request, the windows are terminated by the minimum value in each case:

$$t_{S(i)}^{\mathrm{end}'} = \min\left(t_{S(i)}^{\mathrm{end}}, t_{S(i,m)}^{\mathrm{end}}\right), \text{ and} \tag{C.3}$$

$$t_{S(j)}^{\mathrm{end}'} = \min\left(t_{S(j)}^{\mathrm{end}}, t_{S(j,m)}^{\mathrm{end}'}\right). \tag{C.4}$$

Reasonably tight service-quality constraints for a mode can ensure generally early service for trips that can be feasibly carried by the mode, while leaving some leeway for optimisation with respect to the efficiency and ridership objectives. An explicit early service criterion can however be useful in defining special types of service, as follows:

(a) *Blue-ribbon service*. For a blue-ribbon mode, trip-insertions are chosen so as to minimise $\mathrm{EAT}'_{S_{\mathrm{SD}(t)}}$. Furthermore, when such an insertion is carried out for a given trip t, the service quality is locked-in by setting $t_S^{\mathrm{end}} = \mathrm{EAT}'_S + t_S^{\mathrm{serv}}$ for $S \in \{S_{\mathrm{PU}(t)}, S_{\mathrm{SD}(t)}\}$. This ensures that the early service obtained when the trip is first inserted will not be impaired in subsequent optimisations, even with different objectives. Blue-ribbon service is equivalent to stretchable windows (see below) with $twmax_m = 0$.

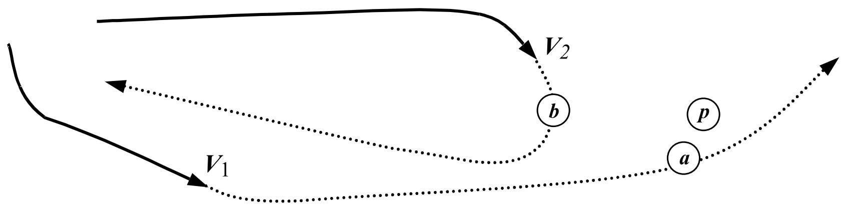


Fig. 5. Waiting time versus travel time as an objective.

(b) *Stretchable windows*. Window-stretching is applicable where the aim is to provide good service within the service standards, while attempting to accommodate travellers in cases where these standards cannot be adhered to. This corresponds with the way taxis operate in practice, and is useful in a modelling context where a realistic estimate of fleet capacity is sought. The "stretched" trip-insertion process runs as follows. First, search for an insertion in the normal way (i.e. with the standard objectives and the constraints as in (C.3) and (C.4)). If a feasible insertion is found, no stretching is required. Otherwise, re-define the time-windows so as to ignore service standards (i.e. with $t^{\text{end}}_{S(i)}, t^{\text{end}}_{S(j)}$ as specified in the original travel request); then apply the early-service criterion in a search for an insertion satisfying these stretched constraints. If this second search succeeds, implement the insertion and lock in the window-termination times as in (a) above.

References

Desrochers, M., Desrosiers, J., Solomon, M., 1992. A new optimization algorithm for the vehicle routing problem with time windows. Operations Research 40 (2), 342–354.

Dial, R.B., 1995. Autonomous dial-a-ride transit: intoductory overview. Transportation Research C 3 (5), 261–275.

Dumas, Y., Desrosiers, J., Soumis, F., 1991. The pickup and delivery problem with time windows. European Journal of Operational Research 54, 7–22.

Gendreau, M., Hertz, A., Laporte, G., 1994. A tabu search heuristic for the vehicle routing problem. Management Science 40 (10), 1276–1290.

Horn, M.E.T., 1999. A booking management and fleet scheduling system for demand-responsive passenger services. CSIRO Mathematical and Information Sciences Technical Report CMIS 99/63.

Horn, M.E.T., 2000. Efficient modelling of travel in networks with time-varying link speeds. Networks 36 (2), 80–90.

Horn, M.E.T., 2001. Multi-modal and demand-responsive passenger transport systems: a modelling framework with embedded control systems. Transportation Research A, forthcoming.

Horn, M.E.T., Smith, J.R., Robinson, B., 1999. Taxi fleet performance under flexible operating conditions and with improved scheduling procedures. In: Proceedings of the Fourth International Conference of ITS Australia, Adelaide SA.

Ioachim, I., Desrosiers, J., Dumas, Y., Solomon, M., Villeneuve, D., 1995. A request clustering algorithm for door-to-door handicapped transportation. Transportation Science 29 (1), 63–78.

Jaw, J.-J., Odoni, A.R., Psaraftis, H.N., Wilson, N.M., 1986. A heuristic algorithm for the multi-vehicle advance request dial-a-ride problem with time windows. Transportation Research 20B (3), 243–257.

Kilby, P., Prosser, P., Shaw, P., 1997. Guided local search for the vehicle routing problem. In: Proceedings of the Second International Conference on Metaheuristics, Sophia-Antipolis, France.

Kowshik, R.R., Reddy, P.D.V.G., Gard, J., Jovanis, P.P., Kitamura, R., 1994. Real-time rideshare matching using GIS. In: Proceedings of the Annual Meeting of IVHS America, vol. 1, Atlanta, GA, pp. 119–125.

Laporte, G., 1992. The vehicle routing problem: an overview of exact and approximate algorithms. European Journal of Operational Research 59, 345–358.

Osman, I.H., 1993. Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. Annals of Operational Research 41, 421–451.

Potvin, J.-Y., Bengio, S., 1994. A genetic approach to the vehicle routing problem with time windows. Technical Report CRT-953, Centre de Recherche sur les Transports, University of Montreal.

Shrivastava, M., Chande, P.K., Monga, A.S., Kashigawi, H., 1997. Taxi despatch: a fuzzy rule approach. In: Proceedings of the IEEE Conference on Intelligent Transportation Systems, Boston.

Smith, B.L., Durvasula, P.K., Turochy, R.E., Brich, S.C., Demetsky, M.J., 1998. Supporting demand responsive transit operations: a prototype system designed using the national ITS architecture. In: Proceedings of the 1998 ITS America Annual Meeting and Exposition, Detroit.