Lees-Miller, J. D., & Wilson, R. E. (2011). Proactive Empty Vehicle Redistribution for Personal Rapid Transit and Taxis.

Proactive Empty Vehicle Redistribution for Personal Rapid Transit and Taxis

John D. Lees-Miller

Department of Engineering Mathematics, University of Bristol, Bristol, UK

R. Eddie Wilson

Transportation Research Group, University of Southampton, Southampton, UK

Corresponding Author:
John D. Lees-Miller
University of Bristol, Engineering Mathematics
Merchant Venturers Building
Woodland Road
Bristol, UK BS8 1UB
enjdlm@bristol.ac.uk

Proactive Empty Vehicle Redistribution for Personal Rapid Transit and Taxis

The empty vehicle redistribution (EVR) problem is to decide when and where to move empty vehicles in a Personal Rapid Transit or taxi system. These decisions are made in real time by an EVR algorithm. A reactive EVR algorithm moves empty vehicles only in response to known requests; in contrast, a proactive EVR algorithm moves empty vehicles in anticipation of future requests. This paper describes two new proactive EVR algorithms, called Sampling and Voting (SV) and Dynamic Transportation Problem (DTP), that move empty vehicles proactively based on demand estimates from historical data. It also develops methods for assessing the performance of EVR algorithms absolutely in terms of both throughput and passenger waiting times. In simulation tests, the proposed algorithms provide lower passenger waiting times than other algorithms in the literature, and proactive movement of empty vehicles significantly reduces waiting times, usually with a modest increase in empty vehicle travel.

Keywords: personal rapid transit; empty vehicle redistribution; empty vehicle management; taxi dispatch; operations

1. Introduction

Personal Rapid Transit (PRT) is an emerging urban transport mode. It will use small, computer-guided vehicles to carry individuals and small groups between pairs of stations on a dedicated network of guideways. The vehicles will operate on-demand and provide direct service from origin station to destination station, much like conventional taxis.

The world's first PRT system is in the final stages of commissioning at London's Heathrow Airport (ULTra PRT 2010). It is a `last mile' circulator with three stations and twenty-one driverless vehicles (Figure 1), connecting a business car park with Terminal 5. Many other recently proposed PRT systems provide connections between train stations, bus stations or off-site car parks and a wide range of destinations (Bly and Teychenne 2005). Used in this way, PRT can increase the efficacy of other public transport modes and create new possibilities for urban planning. In order for PRT to play this role, it must provide a high-quality service,

which means low passenger waiting times, low travel times and high levels of safety and comfort.

The focus of this paper is on methods for moving empty vehicles to provide low passenger waiting times. Deciding which vehicles to move and where to move them is known as the empty vehicle redistribution (EVR) problem. It is assumed that passengers request immediate service (that is, they do not book ahead) from their origin station to their chosen destination station. A central control system can move empty vehicles *reactively*, in response to a request that has just been received, and *proactively*, in anticipation of future requests. Without proactive movements, empty vehicles tend to wait idle at stations where there is a net inflow of occupied vehicles. This leads to long passenger waiting times at stations where there is a net outflow of occupied vehicles (Lees-Miller et al. 2010). Clearly, proactive movement of empty vehicles in anticipation of future arrivals can reduce waiting times, but there is a risk of unnecessary empty vehicle travel.

This paper proposes two EVR algorithms that can move vehicles proactively, and it develops methods for assessing the performance of EVR algorithms absolutely, both in terms of throughput and in terms of passenger waiting times. The proposed algorithms are evaluated with these methods and in simulation, and they provide lower waiting times than other algorithms in the literature. The results show that proactive movement of empty vehicles significantly reduces mean passenger waiting times, typically with a modest increase in empty vehicle travel.

Figure 1. Photograph of PRT vehicle and at-grade station at London Heathrow Airport. PRT vehicles, stations and infrastructure are smaller than typical Automated People Mover and urban rail systems. Vehicle length, width and height are 3.7m, 1.4m and 1.8m, respectively. Photo courtesy of ULTra PRT Ltd.

The basic PRT system model is adapted from the urban taxi model of Bell and Wong (2005) (section 2). When a request is received, the vehicle that minimizes the request's waiting time is immediately assigned to serve the request. This algorithm is here called Bell and Wong Nearest Neighbours (BWNN). BWNN is a reactive algorithm, and it assumes no knowledge of future requests. In order to improve on this, it is assumed that, while future requests are not known with certainty, the average request rates between each pair of stations are known from historical data, in the form of an origin-destination demand matrix. In vehicle routing terminology, this makes the EVR problem *dynamic* (because requests are received while the system is operating) and *stochastic* (because statistical information about future requests is available).

However, it proves useful to first consider two deterministic versions of the EVR problem. Firstly, in the *fluid limit* problem, the variables are long run average

flows of vehicles, rather than individual vehicle movements, and the objective is to minimise the number of vehicles used to balance the flows of occupied and empty vehicles; this is a classical transportation problem. Analysis in the fluid limit gives a way of benchmarking EVR algorithms in terms of throughput (section 3). Secondly, in the *static* problem, every individual request is assumed to be known in advance. An optimal solution to the static problem provides a benchmark in terms of passenger waiting time (section 4).

The two proposed algorithms are extensions to BWNN that allow proactive movements. The Sampling and Voting (SV) algorithm works by generating an ensemble of possible sequences of future requests from the demand matrix over a given finite horizon. Each sequence, together with the current state of the system, defines an instance of the static EVR problem whose (approximate) solution suggests a plan of empty vehicle movements. Features of these plans that are common across the ensemble are then extracted to determine which empty vehicles should actually be moved (section 5). In contrast, the Dynamic Transportation Problem (DTP) algorithm works by attempting to maintain a given *target* number of vehicles inbound to each station. The problem of satisfying the targets with minimum empty vehicle movement is a classical transportation problem (section 6). The SV algorithm was first introduced by Lees-Miller and Wilson (2011), but the presentation here is different; the DTP algorithm has not been described previously.

PRT has much in common with a conventional taxi service, and the principle of proactive empty vehicle movement also applies to conventional taxis. Most existing work on taxi dispatch focuses on reactive algorithms (Horn 2002, Bell and Wong 2005, Seow et al. 2010) and related operational challenges such as travel time estimation (Lee et al. 2004) and the handling of both advanced bookings and

immediate requests (Horn 2002, Wang et al. 2009). Approaches to proactive movement include random roaming (Lee et al. 2004), the 'go to hotspot' heuristics of Li (2006), and the 'rank homing' heuristics of Horn (2002). A heuristic similar to that of Horn (2002) is included in the simulation tests in section 7. Similar problems that involve the repositioning of idle vehicles arise in the control of automated guided vehicle (AGV) systems (Vis, 2006) and elevators (Wesselowski and Cassandras, 2007).

2. The Model

The three main factors that determine passenger waiting times are congestion on the guideway, congestion in stations and the availability and locations of empty vehicles. For very large systems with many vehicles, congestion effects will often be significant, but most systems proposed for the near and medium term will operate well below the congested limit. This motivates the following simplifying assumptions.

(1) Congestion on the guideway is ignored, so vehicles take quickest paths, and the travel times between stations are constants.

(2) Congestion in stations is ignored; any delays in stations are constants included in the travel times.

(3) In addition, it is assumed that the requested average flows of occupied vehicles between stations are known, in the form of an origin-destination matrix derived from historical data, and that these averages are steady (they do not change with time).

The relevant input data are then as follows. Let S be the set of stations. For each pair of stations i and j, let T_{ij} be the travel time from i to j, in minutes, with $T_{ij} = 0$ if $i = j$. Similarly, let D_{ij} be the average number of occupied vehicle trips per minute from i to j, with $D_{ij} = 0$ if $i = j$.

The PRT system model used in this paper is based on the urban taxi model of Bell and Wong (2005). Let K be the set of vehicles, and let n_K be the fleet size ($n_K = |K|$). Each vehicle $k \in K$ has a planned route, which consists of a list of stations that it must visit in order. Each pair of adjacent stations defines a *trip*, during which the vehicle may be occupied or empty. A vehicle's route changes over time: completed trips are deleted from the head, and new occupied or empty vehicle trips are appended to the tail as they are assigned. If a vehicle completes all of the trips in its list, it becomes *idle* at the last station on its route. For simplicity, it is assumed that the lists are not reordered, so for each vehicle $k \in K$, it is enough to know the last station, d_k, that vehicle k was assigned to visit, and the time, a_k, at which the vehicle will arrive (or has already arrived) at d_k. With this notation, an idle vehicle is one whose a_k is in the past.

When a new *request* for travel is received, a vehicle is immediately assigned to serve it. Each request, r, has associated with it an origin station i_r, a destination station j_r and the time e_r at which the system receives the request. It is assumed that each request is for immediate travel from its origin station, so the *waiting time* of the request is the delay between e_r and when the assigned vehicle *picks up* the request at i_r.

The following heuristic, here called Bell and Wong nearest neighbours (BWNN), is used to decide which vehicle to assign. When a request r is received at time e_r, BWNN immediately assigns the vehicle

$$k^* = \underset{k \in K}{\operatorname{argmin}}[\max\{0, a_k - e_r\} + T(d_k, i_r)] \quad (1)$$

where the travel times T_{ij} are written $T(i,j)$ here for readability. The first term, $\max\{0, a_k - e_r\}$, is the delay before the vehicle can start a new trip; note that the vehicle cannot begin its trip in the past, before e_r. The second term, $T(d_k, i_r)$, is the

required empty vehicle trip time; if k is already going to the request's origin station ($d_k = i_r$), then the empty vehicle trip time is zero, because no empty trip is required. Figure 2 gives an illustrated example.

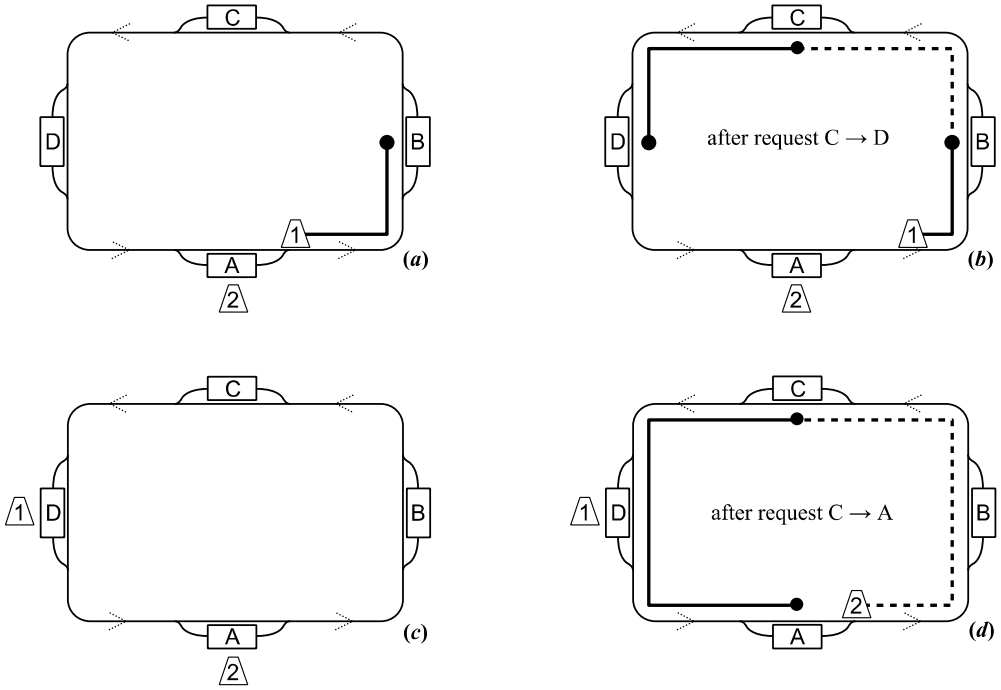

Figure 2. Illustration of the BWNN algorithm. There are four off-line stations (labelled *A* - *D*) in a ring and two vehicles (labelled *1* and *2*). Traffic flow is anticlockwise. (*a*) Vehicle *1* is initially moving to station *B*, and vehicle *2* is idle at station *A*. (*b*) When a request for travel from *C* to *D* is received, vehicle *1* is assigned, because it gives a smaller waiting time than vehicle *2*. Both an empty trip (dashed line) from *B* to *C* and an occupied trip (solid line) from *C* to *D* are appended to vehicle *1*'s route. Note that vehicle *1* stops at station *B* and station *C* (filled circles), but it does not become idle at either station, because it has not finished its route. (*c*) However, vehicle *1* does become idle at *D*, because no further requests are assigned to it. (*d*) When another request is received from *C* to *A*, vehicle *2* is assigned, and it begins an empty trip to *C*. Note that vehicle *2* was idle at station *A*, so the passenger's waiting time might have been reduced by moving vehicle *2* to station *C* proactively, before the request was received.

The BWNN algorithm is *reactive*, because it moves vehicles only in response to requests. The proposed SV and DTP algorithms (see sections 5 and 6) extend BWNN to allow for proactive movements. Before introducing these, we first develop some related theory.

3.The Fluid Limit EVR Problem

The *fluid limit* deals with long run average flows of vehicles, rather than individual vehicle trips. In particular, the aim here is to find the flows of empty vehicles, X_{ij}, that are needed to balance the given flows D_{ij} of occupied vehicles. The result is a classical transportation problem that is well-known in the PRT literature (Anderson 1978) and can also be derived from the urban taxi economics model of Yang et al. (2002). The main output of this fluid limit analysis is a measure of the theoretical maximum throughput of the system, which provides a benchmark against which EVR algorithms can be compared (Lees-Miller et al. 2010).

The transportation problem in the fluid limit is formulated as follows. Let $s_i = \sum_j (D_{ji} - D_{ij})$ be the occupied vehicle flow *surplus* at station i, and partition the stations into $S^+ = \{i \in S:\ s_i \geq 0\}$ and $S^- = \{i \in S:\ s_i < 0\}$. The stations in S^+ have a net inflow of occupied vehicles on average, and the stations in S^- have a net outflow on average. The flows of empty vehicles (X_{ij}) required to balance the total inflows and outflows can then be found from the following transportation problem:

$$\min \quad \sum_{i \in S^+} \sum_{j \in S^-} T_{ij} X_{ij} \tag{2}$$

$$\text{s.t.} \quad \sum_{j \in S^-} X_{ij} = s_i \qquad \forall\ i \in S^+ \tag{3}$$

$$\sum_{i \in S^+} X_{ij} = -s_j \qquad \forall\ j \in S^- \tag{4}$$

$$X_{ij} \geq 0 \qquad \forall\ i \in S^+ \ \text{ and } \ \mathrm{j} \in \mathrm{S}^-$$

The objective (2) is to minimise the average number of moving empty vehicles (minutes, times vehicles per minute, gives vehicles), and constraints (3, 4) ensure that the empty vehicle flows balance the inflows and outflows of occupied vehicles. This problem can be solved efficiently using standard techniques (Bertsimas and Tsitsiklis 1997).

An important use of this analysis is to define the *intensity* of the demand (Lees-Miller et al. 2010), which is the total number of vehicles (occupied and empty) required, according to the fluid limit analysis, divided by the number of vehicles actually available, n_K. In particular, the objective (2) gives the number of empty vehicles required, and the number of occupied vehicles required can be computed similarly to give the intensity of the demand as

$$\rho = \frac{1}{n_K}\left(n_X^* + \sum_{i,j} T_{ij} D_{ij}\right) \tag{5}$$

where n_X^* is the optimal objective value (2). Intensity one corresponds to maximum possible throughput. When $\rho > 1$, requests are being received faster than the system can serve them, because it does not have enough vehicles. This means that both the number of passengers waiting and their waiting times will grow indefinitely, so long as the demand is held constant. When $\rho < 1$, the system may (depending on how efficiently it uses empty vehicles) be able to keep up with demand. So, while this fluid limit analysis does not give a direct measure of passenger waiting times, the intensity of the demand is an important factor. The term 'intensity' is motivated by similar definitions in the theory of queueing systems, where it is also known as 'utilisation'.

The fluid limit problem provides a benchmark for throughput, but it does not directly provide estimates for waiting times; this requires a more detailed model, such as the one in the next section.

4. The Static EVR Problem

Unlike the fluid limit problem, the static problem deals with individual vehicles and requests. In the static problem, it is assumed that all requests are known in advance, instead of being revealed while the system is operating. This problem is of interest because an optimal solution for an instance of the static problem provides a

benchmark for waiting times; that is, in particular, no dynamic EVR algorithm running on the same instance can produce a lower mean passenger waiting time. The static problem also motivates the SV algorithm described in section 5.

The static EVR problem can be formulated as a *multivehicle truckload pickup and delivery problem with time windows*, which is a well-studied (Yang et al. 2004) vehicle routing problem. An instance of the static EVR problem requires that the initial state of the vehicles and all requests over a given finite horizon be known. For simplicity, let the current time be time zero, adjust the vehicles' a_k times accordingly, and set $a_k = 0$ for vehicles that are currently idle. Let R be the set of requests with e_r within the horizon. All of the i_r, j_r and e_r are known at time zero, and e_r is the earliest time that request r can be picked up.

To translate the static EVR problem into a vehicle routing problem, we construct an auxiliary *vehicle-request graph*, G, with one node for each vehicle, one node for each request, and a *sink* node, s. The routing problem is to find the best routes in G that serve all of the requests. Each route must start from a vehicle node, go through zero or more request nodes, and end at the sink node. More formally, the node set of G is $K \cup R \cup \{s\}$, and the edge set is $\{(u, v): u \in K \cup R, v \in R \cup \{s\}, u \neq v\}$. The routing objective is to minimize mean request (passenger) waiting time, which is known as a *minimum latency* objective. The requirement that request r be picked up only after e_r is a *one-sided time window* constraint. The edge costs encode the required travel times in the PRT network, with

$$c_{uv} = \begin{cases} a_u + T(d_u, i_v), & u \in K, v \in R \\ T(i_u, j_u) + T(j_u, i_v), & u, v \in R, u \neq v \\ 0, & \text{otherwise} \end{cases} \qquad (6)$$

The cost (6) of an edge from vehicle k to request r includes the time required for k to finish serving previously assigned requests (if any) plus the required empty vehicle

trip time (if any). Similarly, when both u and v are requests, the cost includes the occupied travel time $T(i_u, j_u)$ for request u and any empty travel time $T(j_u, i_v)$ required in order to serve v directly after u.

The static EVR problem is NP-hard, because the corresponding routing problem on G generalises the minimum latency version of the asymmetric travelling salesman path problem (ATSPP), which is NP-hard (Nagarajan and Ravi 2008). In particular, the routing problem is a minimum latency ATSPP when there is one vehicle and $e_r = 0$ for all requests. Small instances can be solved exactly using standard techniques, but very large instances must be solved in order to benchmark EVR algorithms, because waiting times must be averaged over thousands of requests. These large instances can only be solved approximately, at present.

In this paper, large instances of the static EVR problem are solved using the following static nearest neighbours (SNN) heuristic (Lees-Miller and Wilson 2011). For each request r, in ascending order by e_r, SNN chooses the vehicle

$$k^* = \underset{k \in K}{\operatorname{argmin}} \max\{0, a_k + T(d_k, i_r) - e_r\} \tag{7}$$

that minimizes the waiting time for request r. SNN is similar to BWNN (1), and in fact for vehicles with $a_k \geq e_r$, the objective values in (1) and (7) are the same. However, if a vehicle has $a_k < e_r$, SNN allows the vehicle to start its empty trip before e_r, whereas BWNN does not. In this sense, SNN moves empty vehicles proactively. If there are several vehicles that minimize (7), one is chosen using the tie-breaking rules in Lees-Miller and Wilson (2011). The vehicle state for k^* (a_{k^*} and d_{k^*}) is then updated accordingly, and the next request is considered.

The solutions produced by SNN are not provably optimal, so other methods may produce lower mean waiting times for a given instance; in this sense, the solutions are not, strictly speaking, benchmarks. However, nearest neighbour

heuristics have been found to produce high quality solutions to other minimum latency routing problems (Larsen et al. 2004, Swihart and Papastavrou 1999). This is particularly true when the fleet size is large, because each vehicle's route tends to be short, and fleet sizes in PRT are large relative to those usually studied in the vehicle routing literature (the case study systems in section 7 have 200 vehicles). The intensity of the demand is also important in determining the difficulty of a given instance; when the intensity of the demand is low, there are long delays between requests, so most requests will simply be served in the order in which they are received. In fact, the SNN heuristic finds solutions with zero waiting time (which are clearly optimal) when the intensity of the demand is below about 80% on the case study systems.

Another use for the static problem is in the proposed Sampling and Voting algorithm described in the next section.

5. New EVR Algorithm: Sampling and Voting (SV)

The sampling and voting (SV) algorithm moves idle vehicles proactively by generating an ensemble of static problems, solving them approximately using SNN, and extracting common features from the solutions in order to decide which idle vehicles to actually move.

When a new request is received at time t, SV assigns a vehicle using the BWNN algorithm. Immediately after a vehicle has been assigned to serve the request, SV may then move idle vehicles proactively. To decide which idle vehicles to move, an ensemble of n_E possible sequences of n_R future requests each is generated from the demand matrix. Each sequence in the ensemble, together with the current state of the system, defines an instance of the static EVR problem. Each of these instances is solved approximately using the SNN algorithm, and each resulting solution prescribes

a sequence of empty vehicle trips, which constitutes ‘advice’ on which idle vehicles the system should actually move. However, because each solution is for a different sequence of requests, they may offer conflicting advice. To determine which action should actually be taken, a voting system is used. The system adopted here is that at most one idle vehicle at each station may be moved. So, each solution casts one vote on the best destination (as defined below) for an idle vehicle at each station i with idle vehicles; note that it may vote for i as the best destination, which means that it votes not to move any idle vehicles from i at this decision point. If the destination with the most votes is not i, an idle vehicle at i is selected and moved.

For each station i with idle vehicles, the destination to vote for is determined as follows. If all vehicles now idle at i were used for requests from i, vote for i. Otherwise, if a vehicle now idle at i was moved empty to another station, j, vote for j (for the first such trip). Otherwise, if any vehicle was moved empty from i to another station, j, vote for j (for the first such trip). Otherwise, vote for i. These voting rules are described in more detail in Lees-Miller and Wilson (2011).

6. New EVR Algorithm: Dynamic Transportation Problem (DTP)

The dynamic transportation problem (DTP) algorithm requires a *target* for the number of vehicles that should be inbound to each station at any given time. If the number of vehicles inbound to a station is below its target, idle vehicles should be moved there; conversely, if a station has inbound vehicles in excess of its target, some of its idle vehicles may be moved elsewhere. The problem of moving idle vehicles to meet targets with minimum empty vehicle running time is a classical transportation problem. The idea of maintaining a target number of vehicles at each station is common in the PRT literature (Andréasson 1998, Anderson 1998). The decision on which vehicles to move is typically made according to rules that can be viewed as

approximation algorithms for the transportation problem. For example, when a station has a deficit, it may call the nearest idle vehicle at a station with a deficit not greater than its own (Andréasson 1998). Also, Andréasson (2003) formulates a different transportation problem that can be used to reroute PRT vehicles while they are moving.

More formally, the DTP algorithm is as follows. For each station i, let θ_i be the target number of inbound vehicles at station i; the θ_i are parameters. When a new request is received at time t, let b_i be the number of vehicles that are inbound to i (that is, $d_k = i$), and let l_i be the number of vehicles that are idle at i (that is, $d_k = i$ and $a_k \leq t$); note that $b_i \geq l_i \geq 0$. Define $u_i = \min\{b_i - \theta_i, l_i\}$ as the *surplus* of vehicles at station i. If $u_i > 0$, station i has a surplus of inbound vehicles, but only l_i of these are currently idle at i, so at most l_i vehicles can be moved now. If $u_i < 0$, station i has a *deficit* of inbound vehicles. In general, the surpluses and deficits need not balance, so we introduce an extra dummy node q with $u_q = -\sum_i u_i$. Let $S' = S \cup \{q\}$ and partition S' into sets $S_t^+ = \{i \in S' : u_i \geq 0\}$ and $S_t^- = \{i \in S' : u_i < 0\}$. Note that this partition may change over time. For each $i \in S_t^+$ and each $j \in S_t^-$, let x_{ij} be the number of vehicles to send from node i to node j, which is to be solved for. If i and j are both stations, then $x_{ij} > 0$ means that x_{ij} vehicles which are currently idle at i are to move to station j. If either $i = q$ or $j = q$, then no vehicles are actually moved, regardless of the value of x_{ij}; in other words, a decision to move idle vehicles from or to the dummy node means that they should be left where they are until the next decision point. The costs for sending vehicles to or from the dummy node are zero (define $T_{iq} = T_{qi} = 0$), because these vehicles do not actually move.

The transportation problem to be solved has the same form as that in the fluid limit (2), but the variables are x_{ij} instead of X_{ij}, the surpluses are u_i instead of s_i, and

the stations (and q) are partitioned into S_t^+ and S_t^- instead of S^+ and S^-. When the targets θ_i are integers, there is an optimal solution in which all of the x_{ij} are integer, because it is a special case of the minimum cost network flow problem and the u_i are integers (Bertsimas and Tsitsiklis 1997, ch. 7). The problem is solved exactly with the integer RELAX IV code (Bertsekas and Tseng 1998) every time a request is received, and also every time a vehicle becomes idle.

The targets, θ_i, must be chosen to suit the network, demand matrix and fleet size. We use two metaheuristics for this purpose, namely simulated annealing (Galassi et al. 2003) and the cross-entropy method (de Boer et al. 2005). For simulated annealing, an initial estimate for the targets can be obtained from the fluid limit problem (2), namely

$$\hat{\theta}_i = \left[\left(\sum_{\substack{\mathrm{j} \\ \mathrm{i} \neq \mathrm{j}}} (D_{ji} + X_{ji}^*) T_{ji} \right) \left(\frac{\sum_{\substack{\mathrm{j} \\ \mathrm{i} \neq \mathrm{j}}} D_{ij}}{\sum_{\substack{\mathrm{j} \\ \mathrm{i} \neq \mathrm{j}}} (D_{ij} + X_{ij}^*)} \right) \right] \tag{8}$$

where the X_{ij}^* are obtained from the optimal solution to (2), and the square brackets $[\cdot]$ denote rounding to the nearest integer. The first factor is the number of vehicles that are expected to be inbound to station i on average, and the second factor is the fraction of vehicles that leave station i occupied. The rationale for the second factor is that if most of the vehicles leaving station i are empty, on average, then the station should not attempt to retain many idle vehicles. Neighbouring solutions are generated by adding -1, 0 or 1 with equal probability to each target, and each target is constrained to remain in $[0, n_K]$. For the cross entropy method, each target takes an integer value in $[0, \theta_{\max}]$ where θ_{max} is an arbitrary upper bound set less than n_K, in order to reduce the problem size. The initial probability distribution for the value of each target is uniform over the integers in $[0, \theta_{\max}]$. The cost of a given solution is the

corresponding mean request waiting time, which is estimated using 'Monte Carlo' simulation.

7. Results

In this section, the proposed algorithms are evaluated in simulations on two case study systems. The main simulation outputs are passenger waiting times and empty vehicle use. The steady state distributions of these outputs are estimated by running long simulations with the demand matrix held constant for each run. Passenger requests from station i to station j are generated from a Poisson process with rate D_{ij}. For convenience, passenger arrival and travel times are rounded to the nearest integer second.

The input data used here are the 'Grid' and 'Corby' networks and their corresponding demand matrices from Lees-Miller et al. (2010) (for the Grid network, the demand matrix with dispersion parameter $\theta = 0.01$ is used). The fleet size is set at $n_K = 200$ vehicles. Intensity one corresponds to a total demand of 1414 requests/hour for the Corby system and 2035 requests/hour for the Grid system.

For comparison, another EVR algorithm, here called the Surplus / Deficit (SD) algorithm, is also evaluated. It is an algorithm for the dynamic EVR problem that moves vehicles proactively. The general approach in SD is similar to several other published EVR algorithms (Anderson 1998); it is most similar to that of Andréasson (1998). Each station i has an associated call time τ_i, which is the cumulative average of all previous empty vehicle trip times to that station. The surplus of vehicles at station i is the number of inbound vehicles minus the expected number of requests over the call time, namely $\tau_i \sum_j D_{ij}$. When a new request is received, a vehicle is assigned using BWNN. Immediately afterward, SD may move idle vehicles proactively, as follows. For each station i with idle vehicles, in descending order by

number of idle vehicles, if the surplus of vehicles at i is greater than or equal to one, an idle vehicle at i is sent to the nearest station with surplus less than zero (if any). Additionally, when a vehicle becomes idle at station i, the above actions are taken for station i only.

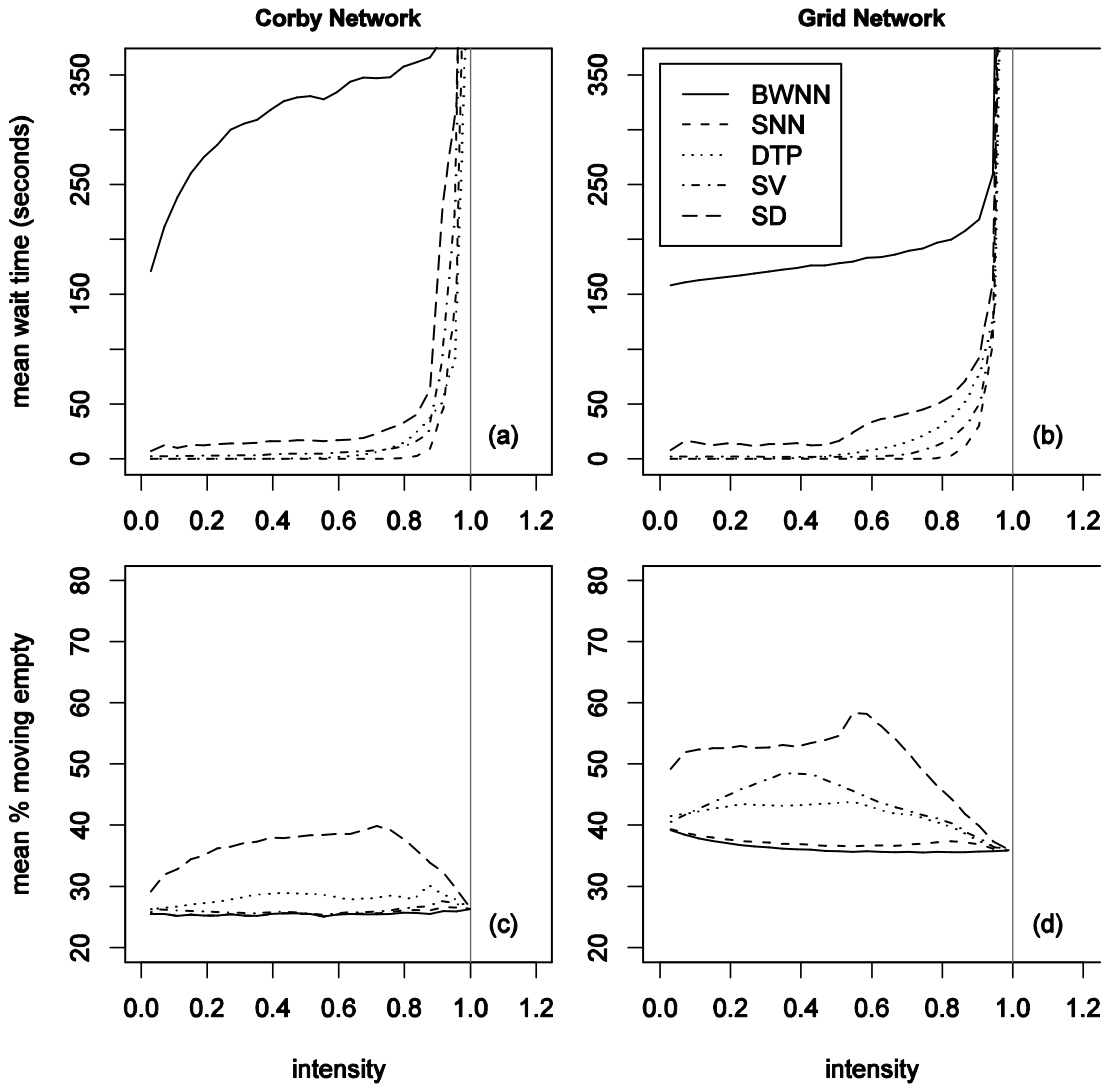

Figure 3. Mean passenger waiting times (*a*, *b*) and empty vehicle use (*c*, *d*) for the Grid and Corby networks for the five heuristics. Here there are $n_E = 50$ sequences, each with $n_R = 300$ requests for SV. The targets used for the DTP algorithm at each intensity are those that produced the lowest mean waiting time at that intensity, whether they were found by simulated annealing or cross-entropy. Each point is averaged over 10 independent runs of 50000 simulated passengers each.

Figure 3(*a*) compares the mean waiting times observed for the five heuristics on the Corby system. An important observation is that waiting times increase rapidly as intensity approaches one, regardless of which EVR algorithm is used, as is

expected based on the definition of intensity (5). In practice, we are most interested in the system's performance at intensities from around 0.7 to around 0.9, because in this range the system is well-utilized, but acceptably low passenger waiting times may still be obtained. At intensity 0.8, for example, mean waiting times are 355s for BWNN, 41s for SD, 20s for DTP, and 15s for SV. By moving vehicles proactively, SV reduces mean waiting times by 96% from the BWNN baseline. The relative reduction decreases as intensity increases, however, and in fact the SV, DTP and SD algorithms become increasingly similar to the BWNN algorithm at higher intensities, because there are fewer idle vehicles to redistribute. Figure 3(*c*) shows that the reduction in passenger waiting times comes from a modest increase in the average number of moving empty vehicles, or equivalently in empty vehicle travel time. The largest increase occurs at intensity 0.91, and this is from 47 concurrently moving empty vehicles with BWNN to 51 with SV (out of 200 vehicles). With perfect information about future arrivals, SNN finds routes with average waiting times less than those for the dynamic case, as expected; at intensity 0.8, the mean waiting time for SNN is 3s. Only mean waiting times are reported, but the ranking of the five algorithms is the same at the 90th percentile of the waiting time distribution; at intensity 0.8, 90% of passengers wait less than 51s with SV, 76s with DTP, and 106s with SD.

Results for the Grid network are qualitatively similar, as shown in Figures 3(*b*, *d*). However, it is notable that waiting times diverge at around intensity 0.95, which is below the theoretical maximum (intensity one). It thus appears that nearest neighbour algorithms of the type studied here do not deliver maximum possible throughput; it is not yet known whether there is any practical algorithm that does.

The following parameters are used to choose the targets for the DTP algorithm. The mean waiting time for each solution is estimated with a simulation of

20000 requests. For simulated annealing, the initial temperature is 10; the temperature decay factor is 1.01; the final temperature is 0.1; 10 evaluations are performed at each temperature; the Boltzmann constant is 1. Five independent simulated annealing runs are performed for each DTP point in Figure 3, for a total of 23150 simulations per point. For cross-entropy, $\theta_{\max} = 50$; each iteration generates 2400 solutions on Grid, or 1500 on Corby; the rarity parameter is 0.1; the probability update smoothing factor is 0.5; the maximum number of allowed iterations is 50, but the search terminates if the maximum likelihood solution does not change for five iterations. Five independent cross-entropy runs are performed for each DTP point in Figure 3, giving an average of 421 thousand simulations per point.

Computation times for the BWNN, SNN and SD algorithms are negligible. Computation times for SV are larger, but SV is still fast enough for real time use with the case study networks: the mean computing time per passenger request for the SV results in Figure 3 is 0.04s (user plus system time on a 2.0GHz Intel Xeon E5405). The computation times for DTP are small once the targets (θ_i) are set (4×10^{-5}s per request on the case study networks). However, a potentially large amount of offline computation is required to find targets that give low waiting times. The offline computation times for the DTP results in Figure 3 are around 100h per point (including both simulated annealing and cross-entropy), which is impractically long. However, there are many possible improvements to the target search method used here. For example, rather than treating each demand matrix (that is, each point in Figure 3) separately, targets obtained for one demand matrix could be used as the initial targets for similar demand matrices. This is a topic for future work.

The results and conclusions presented here are consistent with simulations that have been conducted on nine case study systems in total, with between 15 and 60

stations, between 50 and 600 vehicles, and total demand at intensity one between 360 and 5050 requests/hour. However, it is possible to construct pathological systems in which the SV and DTP heuristics perform poorly. For example, SV gives waiting times much higher than those for DTP (though still lower than those for the BWNN baseline) on a three-station `star' network with equal demand from both 'satellite' stations to the 'hub' station, and no demand from the hub to the satellites. It is possible to reduce SV's waiting times on this star network by allowing it to make proactive movements every time a vehicle becomes idle (like DTP), but this tends to increase waiting times and empty vehicle travel on larger networks. Several such minor variants of SV, DTP and SD have been tested, and the ones presented here performed the best on the case study systems.

8. Conclusions

This paper introduces two new algorithms for the Empty Vehicle Redistribution (EVR) problem in PRT, namely the Sampling and Voting (SV) and Dynamic Transportation Problem (DTP) algorithms. These algorithms move vehicles proactively, in anticipation of future requests. In the case studies considered here, SV and DTP produce much lower passenger waiting times than other heuristics in the literature (up to 96% lower than the BWNN baseline). These benefits are obtained at the cost of some increase in empty vehicle running time (around 22% at intensity 0.8 on the Grid network, for example), and some increase in computational cost.

Studying the EVR problem in the fluid limit, at the level of long run average flows of vehicles, gives a benchmark for throughput. Studying the static version of the EVR problem, in which all requests are known in advance, gives a benchmark for passenger waiting times, and it shows that the EVR problem is hard in a strong sense (NP-hard) when the demand is heavy. These benchmarks indicate the potential for

other algorithms to increase throughput or decrease passenger waiting time. The results obtained here do not rule out improvements over SV and DTP.

This suggests several items for future work. The details of the SV algorithm might be improved along the lines suggested in Lees-Miller and Wilson (2011). More efficient ways of setting the targets for the DTP algorithm would make it more practical. Algorithms combining the ideas from SV and DTP are also possible. The fidelity of the model could be improved by including congestion effects on the line and at stations (variable travel times), and by allowing reoptimisation of the vehicles' request lists, possibly including vehicle rerouting. The empirical evaluations done here are for PRT systems; similar evaluations could be done with network and demand data from taxi systems.

Proactive empty vehicle movement can substantially reduce passenger waiting times in demand responsive transport systems that serve immediate requests, such as PRT and taxi systems. Moreover, the particular EVR algorithm used to make proactive movements strongly affects the obtained waiting times. Improvements to these algorithms may therefore be of considerable value to practitioners.

References

Anderson, J.E., 1978. *Transit systems theory*. Lexington, MA: Lexington Books.

Anderson, J.E., 1998. Control of personal rapid transit systems. *Journal of Advanced Transportation*, 32 (1), 57–74.

Andréasson, I., 1998. Quasi-Optimum Redistribution of Empty PRT Vehicles. *In*: W. J. Sproule, E. S. Neumann and S. W. Lynch, eds. *Automated People Movers VI: Creative Access for Major Activity Centers*, 9–12 April 1997 Las Vegas, NV. Reston, VA: American Society of Civil Engineers, 541–550.

Andréasson, I., 2003. Reallocation of empty personal rapid transit vehicles en route. *Transportation Research Record: Journal of the Transportation Research Board*, 1838, 36–41.

Bell, M.G.H. and Wong, K.I., 2005. A Rolling Horizon Approach to the Optimal Dispatching of Taxis. *In*: H. S. Mahmassani, ed. *Transportation and traffic*

theory : flow, dynamics and human interaction: proceedings of the 16th International Symposium on Transportation and Traffic Theory. Amsterdam: Elsevier, 629–648.

Bertsekas, D.P. and Tseng, P., 1988. Relaxation Methods for Minimum Cost Ordinary and Generalized Network Flow Problems. *Operations Research*, 36, 93–114.

Bertsimas, D. and Tsitsiklis, J., 1997. *Introduction to Linear Optimization*. Belmont, MA: Athena Scientific.

Bly, P.H. and Teychenne, P., 2005. Three Financial and Socio-Economic Assessments of a Personal Rapid Transit System. *In*: *Proceedings of the Tenth International Conference on Automated People Movers*, 1–4 May 2005 Orlando, FL. [CD-ROM]. American Society of Civil Engineers.

de Boer, P.-T., et al., 2005. A Tutorial on the Cross-Entropy Method. *Annals of Operations Research*, 134 (1), 19–67.

Galassi, M., et al., 2003. *GNU Scientific Library: Reference Manual*. Bristol: Network Theory Ltd.

Horn, M., 2002. Fleet scheduling and dispatching for demand-responsive passenger services. *Transportation Research Part C: Emerging Technologies*, 10 (1), 35–63.

Larsen, A., Madsen, O.B.G. and Solomon, M.M., 2004. The A Priori Dynamic Traveling Salesman Problem with Time Windows. *Transportation Science*, 38 (4), 459–472.

Lee, D.-H., et al., 2004. Taxi Dispatch System Based on Current Demands and Real-Time Traffic Conditions. *Transportation Research Record: Journal of the Transportation Research Board*, 1882, 193–200.

Lees-Miller, J.D., Hammersley, J.C. and Wilson, R.E., 2010. Theoretical Maximum Capacity as Benchmark for Empty Vehicle Redistribution in Personal Rapid Transit. *Transportation Research Record: Journal of the Transportation Research Board*, 2146, 76–83.

Lees-Miller, J.D. and Wilson, R.E., 2011. Sampling for Personal Rapid Transit Empty Vehicle Redistribution. *Transportation Research Record: Journal of the Transportation Research Board*, submitted.

Li, S., 2006. *Multi-attribute taxi logistics optimization*. Thesis (S. M.). Massachusetts Institute of Technology.

Nagarajan, V. and Ravi, R., 2008. The Directed Minimum Latency Problem. *In:* A. Goel, et. al, eds. *Approximation, Randomization and Combinatorial Optimization: Algorithms and Techniques*. Berlin: Springer, 193–206.

Seow, K.T., Dang, N.H. and Lee, D.-H., 2010. A Collaborative Multiagent Taxi-Dispatch System. *IEEE Transactions on Automation Science and Engineering*, 7 (3), 607–616.

Swihart, M.R. and Papastavrou, J.D., 1999. A stochastic and dynamic model for the single-vehicle pick-up and delivery problem. *European Journal of Operational Research*, 114 (3), 447–464.

ULTra PRT, 2010. *ULTra at London Heathrow Airport*. Available from: http://www.ultraprt.com/applications/existing-systems/heathrow [Accessed 8 February 2011].

Vis, I. F. A., 2006. Survey of research in the design and control of automated guided vehicle systems. *European Journal of Operational Research*, 170 (3), 677–709.

Yang, H., Wong, S. and Wong, K., 2002. Demand-supply equilibrium of taxi services in a network under competition and regulation. *Transportation Research Part B: Methodological*, 36 (9), 799–819.

Yang, J., Jaillet, P. and Mahmassani, H., 2004. Real-Time Multivehicle Truckload Pickup and Delivery Problems. *Transportation Science*, 38 (2), 135–148.

Wang, H., Lee, D.-H. and Cheu, R., 2009. PDPTW Based Taxi Dispatch Modeling for Booking Service. *In: Fifth International Conference on Natural Computation*, 14–16 August 2009 Tianjian. Los Alamitos, CA: IEEE Computer Society, 242–247.

Wesselowski, K., and Cassandras, C. G., 2007. The Elevator Dispatching Problem: Hybrid System Modeling and Receding Horizon Control. *In: Analysis and Design of Hybrid Systems 2006: a Proceedings volume from the 2nd IFAC Conference*, 7–9 June 2006 Alghero, Italy. Elsevier, 136–141.