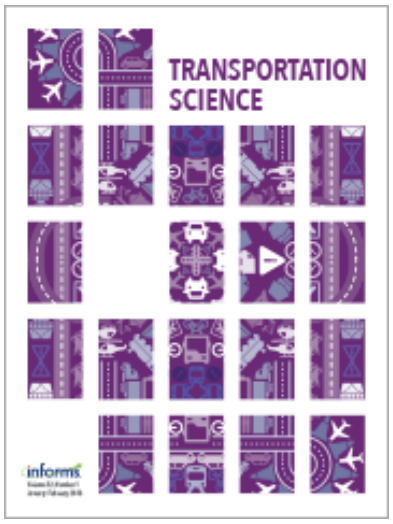

Transportation Science

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

The General Pickup and Delivery Problem

M. W. P. Savelsbergh, M. Sol,

To cite this article:
M. W. P. Savelsbergh, M. Sol, (1995) The General Pickup and Delivery Problem. Transportation Science 29(1):17-29. https://doi.org/10.1287/trsc.29.1.17

The General Pickup and Delivery Problem[1]

M. W. P. SAVELSBERGH

Georgia Institute of Technology, Atlanta, Georgia 30032-0205

M. SOL

Eindhoven University of Technology, 5600 MB Eindhoven, The Netherlands

In pickup and delivery problems vehicles have to transport loads from origins to destinations without transshipment at intermediate locations. In this paper, we discuss several characteristics that distinguish them from standard vehicle routing problems and present a survey of the problem types and solution methods found in the literature.

INTRODUCTION

In the general pickup and delivery problem (GPDP) a set of routes has to be constructed in order to satisfy transportation requests. A fleet of vehicles is available to operate the routes. Each vehicle has a given capacity, a start location and an end location. Each transportation request specifies the size of the load to be transported, the locations where it is to be picked up (the origins) and the locations where it is to be delivered (the destinations). Each load has to be transported by one vehicle from its set of origins to its set of destinations without any transshipment at other locations.

Three well-known and extensively studied routing problems are special cases of the GPDP. In the pickup and delivery problem (PDP), each transportation request specifies a single origin and a single destination and all vehicles depart from and return to a central depot. The dial-a-ride problem (DARP) is a PDP in which the loads to be transported represent people. Therefore, we usually speak of clients or customers instead of transportation requests and all load sizes are equal to one. The vehicle routing problem (VRP) is a PDP in which either all the origins or all the destinations are located at the depot.

The GPDP is introduced in order to be able to deal with various complicating characteristics found in many practical pickup and delivery problems, such as transportation requests specifying a set of origins associated with a single destination or a single origin associated with a set of destinations, vehicles with different start and end locations, and transportation requests evolving in real time.

Many practical pickup and delivery situations are demand responsive, i.e., new transportation requests become available in real-time and are immediately eligible for consideration. As a consequence, the set of routes has to be reoptimized at some point to include the new transportation requests. Observe that at the time of the reoptimization, vehicles are on the road and the notion of depots becomes void.

The purpose of this paper is to present a general model that can handle the practical complexities mentioned above, to isolate and discuss some of the characteristics that differentiate pickup and delivery problems from traditional vehicle routing problems, and to give an overview of the literature on pickup and delivery problems. In our survey of the literature, we focus primarily on deterministic models. For a discussion of issues concerning stochastic vehicle routing problems, the reader is referred to the survey paper by DROR, LAPORTE and TRUDEAU.[12]

With respect to the literature on routing and scheduling problems, it is interesting to observe that although pickup and delivery problems are as important from a practical point of view and as interesting from a theoretical point of view as vehicle routing problems, they have received far less attention. We hope that by identifying and dis-

[1]Accepted by M. S. Daskin.

Transportation Science
Vol 29, No. 1, February 1995

0041-1655/95/2901-0017 $01 25

cussing important issues regarding pickup and delivery problems and by presenting a survey of the literature, we stimulate further research in the area.

1. PROBLEM FORMULATION

LET N BE the set of transportation requests. For each transportation request $i \in N$, a load of size $\bar{q}_i \in \mathbb{N}$ has to be transported from a set of origins N_i^+ to a set of destinations N_i^-. Each load is subdivided as follows: $\bar{q}_i = \sum_{j \in N_i^+} q_j = -\sum_{j \in N_i^-} q_j$, i.e., positive quantities for pickups and negative quantities for deliveries. Define $N^+ := \bigcup_{i \in N} N_i^+$ as the set of all origins and $N^- := \bigcup_{i \in N} N_i^-$ as the set of all destinations. Let $V := N^+ \cup N^-$. Furthermore, let M be the set of vehicles. Each vehicle $k \in M$ has a capacity $Q_k \in \mathbb{N}$, a start located k^+, and an end location k^-. Define $M^+ := \{k^+ | k \in M\}$ as the set of start locations and $M^- := \{k^- | k \in M\}$ as the set of end locations. Let $W := M^+ \cup M^-$.

For all $i, j \in V \cup W$ let d_{ij} denote the travel distance, t_{ij} the travel time, and c_{ij} the travel cost. Note that the dwell time at origins and destinations can be easily incorporated in the travel time and therefore will not be considered explicitly.

DEFINITION 1. *A pickup and delivery route R_k for vehicle k is a directed route through a subset $V_k \subset V$ such that:*

1. *R_k starts in k^+.*
2. *$(N_i^+ \cup N_i^-) \cap V_k = \emptyset$ or $(N_i^+ \cup N_i^-) \cap V_k = N_i^+ \cup N_i^-$ for all $i \in N$.*
3. *If $N_i^+ \cup N_i^- \subseteq V_k$, then all locations in N_i^+ are visited before all locations in N_i^-.*
4. *Vehicle k visits each location in V_k exactly once.*
5. *The vehicle load never exceeds Q_k.*
6. *R_k ends in k^-.*

DEFINITION 2. *A pickup and delivery plan is a set of routes $\mathscr{R} := \{R_k | k \in M\}$ such that:*

1. *R_k is a pickup and delivery route for vehicle k, for each $k \in M$.*
2. *$\{V_k | k \in M\}$ is a partition of V.*

Define $f(\mathscr{R})$ as the price of plan $\mathscr{R}$ corresponding to a certain objective function f. We can now define the general pickup and delivery problem as the problem:

$$\min\{f(\mathscr{R}) | \mathscr{R} \text{ is a pickup and delivery plan}\}$$

The special cases of the GPDP mentioned in the introduction can be characterized as follows:

The pickup and delivery problem: $|W| = 1$ and $|N_i^+| = |N_i^-| = 1$ for all $i \in N$. In this case we define i^+ as the unique element of N_i^+ and i^- as the unique element of N_i^-.

The dial-a-ride problem: $|W| = 1$ and $|N_i^+| = |N_i^-| = 1$ and $\bar{q}_i = 1$ for all $i \in N$.

The vehicle routing problem: $|W| = 1$, $|N_i^+| = |N_i^-| = 1$ for all $i \in N$, and $N^+ = W$ or $N^- = W$.

Although the literature merely covers pickup and delivery problems with $|N_i^+| = |N_i^-| = 1$, In many practical situations $|N_i^+| > 1$ or $|N_i^-| > 1$. In some of these situations, a transportation request with $|N_i^+| > 1$ or $|N_i^-| > 1$ can be decomposed into several, independent requests with a single pickup point and a single delivery point. However, in many other situations, a request with multiple pickup or delivery points has to be served by a single vehicle and therefore cannot be decomposed.

Although we are not aware of any real-life applications where requests occur with both $|N_i^+| > 1$ and $|N_i^-| > 1$, this case is part of the GPDP definition for the sake of symmetry.

To formulate the GPDP as a mathematical program, we introduce four types of variables: z_i^k ($i \in N$, $k \in M$) equal to 1 if transportation request i is assigned to vehicle k and 0 otherwise, x_{ij}^k ($(i,j) \in (V \times V) \cup \{(k^+, j) | j \in V\} \cup \{(j, k^-) | j \in V\}$, $k \in M$) equal to 1 if vehicle k travels from location i to location j and 0 otherwise, D_i ($i \in V \cup W$), specifying the departure time at vertex i, and y_i ($i \in V \cup W$), specifying the load of the vehicle arriving at vertex i. Define $q_{k^+} = 0$ for all $k \in M$. The problem is now to

minimize $f(x)$

subject to

$$\sum_{k \in M} z_i^k = 1 \qquad \text{for all } i \in N, \quad (1)$$

$$\sum_{j \in V \cup W} x_{lj}^k = \sum_{j \in V \cup W} x_{jl}^k = z_l^k \qquad \text{for all } i \in N, l \in N_i^+ \cup N_i^-, k \in M, \quad (2)$$

$$\sum_{j \in V \cup \{k^-\}} x_{k^+ j}^k = 1 \qquad \text{for all } k \in M, \quad (3)$$

$$\sum_{i \in V \cup \{k^+\}} x_{ik^-}^k = 1 \qquad \text{for all } k \in M, \quad (4)$$

$$D_{k^+} = 0 \qquad \text{for all } k \in M, \quad (5)$$

$$D_p \le D_q \qquad \text{for all } i \in N, p \in N_i^+, q \in N_i^-, \quad (6)$$

$$x_{ij}^k = 1 \Rightarrow D_i + t_{ij} \leq D_j \quad \text{for all } i, j \in V \cup W, k \in M, \quad (7)$$

$$y_{k^+} = 0 \quad \text{for all } k \in M, \quad (8)$$

$$y_l \leq \sum_{k \in M} Q_k z_i^k \quad \text{for all } i \in N, l \in N_i^+ \cup N_i^-, \quad (9)$$

$$x_{ij}^k = 1 \Rightarrow y_i + q_i = y_j \quad \text{for all } i, j \in V \cup W, k \in M, \quad (10)$$

$$x_{ij}^k \in \{0, 1\} \quad \text{for all } i, j \in V \cup W, k \in M, \quad (11)$$

$$z_i^k \in \{0, 1\} \quad \text{for all } i \in N, k \in M, \quad (12)$$

$$D_i \geq 0 \quad \text{for all } i \in V \cup W, \quad (13)$$

$$y_i \geq 0 \quad \text{for all } i \in V \cup W. \quad (14)$$

Constraint (1) ensures that each transportation request is assigned to exactly one vehicle. By constraint (2) a vehicle only enters or leaves a location l if it is an origin or a destination of a transportation request assigned to that vehicle. Constraints (3) and (4) make sure that each vehicle starts and ends at the correct place. Constraints (5), (6), (7) and (13) together form the precedence constraints. Constraints (8), (9), (10) and (14) together form the capacity constraints.

2. PROBLEM CHARACTERISTICS

2.1. Transportation Requests

A very important characteristic of routing problems is the way in which transportation requests become available. In a static situation, all requests are known at the time the routes have to be constructed. In a dynamic situation, some of the requests are known at the time the routes have to be constructed and the other requests become available in real time during execution of the routes. Consequently, in a dynamic situation, when a new transportation request becomes available, at least one route has to be changed in order to serve this new request. Most vehicle routing problems are static, whereas most pickup and delivery problems are dynamic.

In practice, a dynamic problem is often solved as a sequence of static problems. In its simplest form, each time a new request becomes available the current set of routes is updated. This is commonly referred to as an on-line algorithm. However, it is usually possible and beneficial to buffer incoming rquests and only update the current set of routes if the buffer size exceeds a certain preset value. Another important issue concerning (buffered) on-line algorithms is how to incorporated information that may be known about the spatial or time distribution of future requests.

A depot is another important concept in routing problems. In the routing literature, the depot is usually the place where vehicles start and end their routes. Since most pickup and delivery problems are dynamic, often with a long planning horizon, the concept of a depot vanishes. Drivers sleep at the last location they visited or at the first location they have to visit the next day. Even for problems with a short planning horizon, such as a single day, where vehicles start and end at a central depot, a demand responsive situation leads to problems without depots. When new transportation requests become available and the current set of routes has to be updated the vehicles are spread out over the planning area.

The general pickup and delivery model is well suited for dealing with the subproblems that occur in dynamic, demand responsive, routing problems.

2.2. Time Constraints

Apart from the vehicle capacity constraints and the intrinsic precedence constraints related to pickup and delivery, side constraints related to time arise in almost every practical pickup and delivery situation. Although time constraints have become an integral part of models for vehicle routing problems (for recent surveys on the vehicle routing problem with time windows see DESROCHERS ET AL.[8] and SOLOMON and DESROSIERS[38]), they play an even more prominent role in pickup and delivery problems. Among other reasons, because the most studied pickup and delivery problem is the dial-a-ride problem, which deals with the transportation of people who specify desired pickup or delivery times.

The presence of time constraints complicates the problem considerably. If there are no time constraints, finding a feasible pickup and delivery plan is trivial: arbitrarily assign transportation requests to vehicles, arbitrarily order the transportation requests assigned to a vehicle and process each transportation request separately. In the presence of time constraints the problem of finding a feasible pickup and delivery plan is $\mathcal{NP}$-hard. Consequently, it may be difficult to construct a feasible plan, especially when time constraints are restrictive. On the other hand, an optimization method

may benefit from the presence of time constraints, since the solution space may be much smaller.

2.2.1. Time Constraints Related to Transportation Requests

For each $i \in V \cup W$ a time window $[e_i, l_i]$ is introduced denoting the time interval in which service at location i must take place. Given a pickup and delivery plan and departure times of the vehicles, the time windows define for each $i \in V$ the arrival time A_i and the departure time D_i. Note that $D_i = \max\{A_i, e_i\}$. If $A_i < e_i$, then the vehicle has to wait at location i.

Next to the explicit time windows mentioned above there also exist implicit time windows. Implicit time windows originate from controlling customer inconvenience. In dial-a-ride systems, people usually specify either a desired delivery time $\overline{A}_{i^-}$ or a desired pickup time $\overline{D}_{i^+}$. Because people do not want to be late at their destination, in a pickup and elivery plan the actual delivery time A_{i^-} must satisfy $A_{i^-} \leq \overline{A}_{i^-}$. Analogously, we require $D_{i^+} \geq \overline{D}_{i^+}$. This defines half open time windows. To prevent clients from being served long before (after) their desired delivery (pickup) time, we can either construct closed time windows or take an objective function that penalizes deviations from the desired service time. Closed time windows can, for example, be constructed by defining a maximum deviation from the desired pickup or delivery time.

Another problem in which customer inconvenience can be controlled by time windows emerges from a demand responsive, immediate request, dial-a-ride system. In such a system, clients that request service want to be picked up as soon as possible. If client i has a pickup location that is geographically far away, this client will be scheduled last on a vehicle route. This situation will not change even when new clients request service. To prevent the client from suffering indefinite deferment, a closed pickup time window $[0, l_{i^+}]$ is defined where l_{i^+} is an input to the system.

Other time constraints that originate from customer incovenience restrictions in dial-a-ride systems are maximum ride time restrictions, i.e., a bound on the time a client is in the vehicle, and deadhead restrictions, i.e., no waiting time is allowed when a client is in the vehicle. Deadhead restrictions introduce the concept of schedule blocks, i.e., working periods between two successive slack periods.

2.2.2. Time Constraints Related to Vehicles

Vehicles are usually not available all day. Drivers have to eat and sleep and vehicles are subjected to service plans. These constraints can be modeled as time windows for vehicles. Typically a vehicle has multiple time windows defining all the periods in which it is available.

2.3. Objective Functions

A wide variety of objective functions is found in pickup and delivery problems. The most common ones are discussed below.

First, we present objective functions related to single-vehicle pickup and delivery problems.

Minimize duration. The duration of a route is the total time a vehicle needs to execute the route. Route duration includes travel times, waiting times, loading and unloading times, and break times.

Minimize completion time. The completion time of a route is the time that service at the last location is completed. In case the start time of the vehicle is fixed at time zero, the completion time coincides with the route duration.

Minimize travel time. The travel time of a route refers to the total time spent on actual traveling between different locations.

Minimize route length. The length of a route is the total distance traveled between different locations.

Minimize client inconvenience. In dial-a-ride systems, client inconvenience is measured in terms of pickup time deviation, i.e., the difference between the actual pickup time and the desired pickup time, delivery time deviation, i.e., the difference between the desired delivery time and the actual delivery time, and excess ride time, i.e. the difference between the realized ride time and the direct ride time. In demand responsive situations where clients request immediate service, i.e., as soon as possible, the difference between the time of pickup and the time of request placement may also contribute to the definition of client inconvenience. Different kinds of functions, linear as well as nonlinear, have been proposed to model client inconvenience.

Second, we present objective functions related to multiple-vehicle pickup and delivery problems.

Minimize the number of vehicles. This function is almost always used in dial-a-ride systems, combined with one of the above functions to optimize the single-vehicle subproblems. Dial-a-ride systems are normally highly subsidized systems for the transportation of the elderly and handicapped. Therefore the objective is to minimize cost (mostly together with customer inconvenience). Because drivers and vehicles are the most expensive parts in a dial-a-ride system, minimizing the number of vehicles to serve all requests is usually the main objective.

Maximize profit. This function, which can use all of the above functions, can be used in a system where the dispatcher has the possibility of rejecting a transportation request when it is unfavorable to transport the corresponding load. Note that, for example, in a dial-a-ride system, rejecting a transportation request is not allowed. A model based on this objective function should not only incorporate the costs, but also the revenues associated with the trasportation of loads.

For dynamic pickup and delivery problems it is not clear what kind of objective functions should be used. In a demand responsive environment pickup and delivery routes may be open ended. Therefore, objectives such as duration, completion time, and travel time have no clear meaning. Intuitively, an objective function for dynamic problems should emphasize decisions that affect the near future more than decisions regarding the remote future.

Note that if a dynamic problem is solved as a sequence of static problems, the objective function for the static subproblem does not necessarily have to be equal to the objective function for the dynamic problem. The objective function for the static subproblem may reflect some knowledge or anticipation of future requests.

3. SOLUTION APPROACHES

IN THIS SECTION, we review the literature on pickup and delivery problems. The survey is organized as follows. The class of pickup and delivery problems has been divided into static and dynamic problems, since their characteristics as well as their solution approaches differ considerably. Within either of these classes, we distinguish single-vehicle and multiple-vehicle problems. Obviously, in the single-vehicle PDP all transportation requests are handled by the same vehicle, whereas in the multiple-vehicle PDP the transportation requests have to be divided over the set of vehicles. Assigning transportation requests to vehicles in the PDP is much more difficult than assigning transportation requests to vehicles in the VRP. In the VRP, all the origins of transportation requests are located at the depot. Therefore, transportation requests with geographically close destinations are likely to be served by the same vehicle. In the PDP, geographically close destinations may have origins that are geographically far apart and we cannot conclude that they are likely to be served by the same vehicle.

For each of the resulting subclasses, one or more papers are discussed. The level of detail depends on the originality, viability and importance of the desired solution approach. A performance evaluation of the solution methods is only provided if such information is present in the corresponding paper. Furthermore, we do not cover iterative improvement methods in any detail. For these the interested reader is referred to KINDERVATER and SAVELSBERGH.[22]

3.1. The Static Pickup and Delivery Problem

3.1.1. The Static Single-Vehicle Pickup and Delivery Problem

The static single-vehicle pickup and delivery problem is probably the most studied variant of the PDP. First of all because the dial-a-ride problem belongs to this class, but also because it appears as a subproblem in multiple-vehicle pickup and delivery problems. We discuss solution approaches for problems with and without time windows separately.

(A) The static 1-PDP without time windows

Optimization

PSARAFTIS[29] considers immediate request dial-a-ride problems. In these problems, every client requesting service wishes to be served as soon as possible. The objective is to minimize a weighted combination of the time needed to serve all clients and the total degree of 'dissatisfaction' clients experience until their delivery. Dissatisfaction is assumed to be a linear function of the time each client waits to be picked up and of the time he spends riding in the vehicle until his delivery. This leads to the following objective function:

$$\min w_1 T + w_2 \sum_{i \in N} (\alpha W_i + (2 - \alpha) R_i),$$

where T denotes the route length, W_i the waiting time of client i from the departure time of the vehicle until the time of pickup, R_i the riding time of client i from pickup to delivery, and $0 \leq \alpha \leq 2$.

The problem is solved using a straightforward dynamic programming algorithm. The state space consists of vectors $(L, k_1, k_2, \ldots, k_n)$, where L denotes the location currently being visited ($L = 0$ at the starting location, $L = i$ at the origin of client i and $L = i + n$ at the destination of client i) and k_i denotes the status of client i ($k_i = 3$ if client i has not been picked up, $k_i = 2$ if client i has been picked up but has not been delivered and $k_i = 1$ if client i has been delivered). Starting with state vector $(0, 3, 3, \ldots, 3)$ the algorithm explores new states preserving feasibility of the partial routes constructed so far. The algorithm has a time complexity of $O(n^2 3^n)$, where $n = |N|$, and can solve problems with up to 10 clients, i.e., 20 locations.

KALANTARI, HILL and ARORA[21] propose a method based on the branch and bound algorithm for the

TSP developed by LITTLE, MURTY, SWEENEY and KAREL.[23] The method handles precedence constraints by precluding in each branch all the arcs that violate the active precedence constraints. Problems with up to 18 clients were solved with this algorithm.

FISCHETTI and TOTH[16] develop an additive bounding procedure that can be used in a branch-and-bound algorithm for the TSP with precedence constraints, i.e., a single-vehicle dial-a-ride problem without capacity constraints. The idea behind additive bounding is to use several bounds for a problem type additively rather than separately. Let $\mathscr{L}^{(k)}$ $(k = 1, \ldots, r)$ be a procedure which, when applied to problem P with cost vector c, returns a lower bound $\delta^{(k)}$ and a residual cost vector $c^{(k)}$, such that $\delta^{(k)} + c^{(k)}x \leq cx$ for all feasible x. Apply $\mathscr{L}^{(1)}$ to problem P with cost vector c and subsequently apply $\mathscr{L}^{(k)}$ to problem P with cost vector $c^{(k-1)}$ $(k = 2, \ldots, r)$. Then $\sum_{k=1}^{r} \delta^{(k)} + c^{(k)}x \leq cx$ for all feasible x. So $\sum_{k=1}^{r} \delta^{(k)}$ is a lower bound for P.

The bounds used by FISCHETTI and TOTH for the TSP with precedence constraints are based on the assignment relaxation, the shortest spanning 1-arborescence relaxation, disjunctions, and on variable decomposition, in that order.

Approximation

STEIN[39] presents a probabilistic analysis of a simple approximation algorithm for the single-vehicle dial-a-ride problem without capacity constraints. The algorithm constructs a TSP tour through all origins and a TSP tour through all the destinations and then concatenates them.

Stein shows that if n origin-destination pairs are randomly chosen in a region of area a using a uniform distribution, there exists a constant b such that the length y'_n of the tour constructed by the algorithm satisfies

$$\lim_{n \to \infty} \frac{y'_n}{\sqrt{n}} = 2b\sqrt{a} \text{ with probability 1,}$$

where b is the constant that appears in the theorem of BEARDWOOD, HALTON and HAMMERSLEY[1] on the length of an optimal traveling salesman tour in the Euclidean plane. Recent experiments by JOHNSON[18] indicate that $b \approx 0.713$. STEIN also proves that if n origin-destination pairs are randomly choosen in a region of area a, using a uniform distribution, the length y_n^* of the optimal tour satisfies

$$\lim_{n \to \infty} \frac{y_n^*}{\sqrt{n}} = 1.89b\sqrt{a} \text{ with probability 1.}$$

This implies that the algorithm has an asymptotic performance bound of 1.06 with probability 1.

PSARAFTIS[31] presents a worst-case analysis of a simple two-phase approximation algorithm for the single-vehicle dial-a-ride problem in the plane. In the first phase, a TSP tour through all locations is constructed. In the second phase, this tour is traversed clock-wise, beginning at the start location of the vehicle and skipping any location that has been visited before, any origin where the pickup of a client would violate the capacity constraints, and any destination whose origin has not yet been visited, until a feasible dial-a-ride solution has been obtained.

PSARAFTIS shows that if there are no capacity constraints, the TSP tour has to be traversed at most twice. Furthermore, if in addition to the absence of capacity constraints the triangle inequality holds and CHRISTOFIDES' heuristic[5] is used to construct the TSP tour, the algorithm has a worst case ratio of 3.

FIALA TIMLIN and PULLEYBLANK[15] consider a slightly different problem. There are groups of customers, each group with its own priority. Customers have to be served in order of increasing priority. FIALA TIMLIN and PULLEYBLANK develop an algorithm based on insertion techniques and iterative improvement methods.

(B) The static 1-PDP with time windows

Optimization

PSARAFTIS[30] modifies the dynamic programming algorithm discussed above to solve the static single-vehicle dial-a-ride problems with time windows. The major difference between the new and the original algorithm is the use of forward instead of backward recursion. Time windows are handled by the elimination of time infeasible states. The new algorithm still has a time complexity of $O(n^2 3^n)$.

DESROSIERS, DUMAS and SOUMIS[9] also present a dynamic programming approach. Their algorithm uses states (S, i) with $S \subseteq V$ and $i \in V$. State (S, i) is defined only if there exists a feasible path that visits all nodes in S and ends in i. Elaborate state elimination criteria, based not only on S, but also on the state (S, i), are used to reduce the state space. These elimination criteria are very effective when the time windows are tight and the vehicle capacity is small. The algorithm has been tested successfully on problems with up to 40 transportation requests.When capacity constraints are rather tight, then, despite the fact that a dynamic programming algorithm is exponential, the running

time of this algorithm seems to increase only linearly with problem size.

Approximation

SEXTON and BODIN[36,37] consider the dial-a-ride problem with desired delivery times specified by the clients. The objective is to minimize client inconvenience, which is defined as a weighted combination of delivery time deviation and excess ride time.

The presented solution approach applies Benders decomposition to a mixed 0–1 non-linear programming formulation, which separates the routing and scheduling component.

The concept of space-time separation indicators between tasks plays an important role in the algorithm. Such an indicator measures the travel time between locations at which the tasks are performed, i.e., their spatial separation, and the difference between the latest feasible times at which they can be performed, i.e., their temporal separation.

Let $\bar{A}_{k^-}$ be the desired delivery time of client k. The latest possible pickup time $\bar{D}_{k^+}$ is then defined as $\bar{D}_{k^+} := \bar{A}_{k^-} - t_{k^+k}$. The space-time separation indicators between tasks i and j are now defined as follows:

$$\sigma_{i^+j^+} = t_{i^+j^+} + \bar{D}_{j^+} - \bar{D}_{i^+}$$

$$\sigma_{i^+j^-} = t_{i^+j^-} + \bar{A}_{j^-} - \bar{D}_{i^+}$$

$$\sigma_{i^-j^+} = t_{i^-j^+} + \bar{D}_{j^+} - \bar{A}_{i^-}$$

$$\sigma_{i^-j^-} = t_{i^-j^-} + \bar{A}_{j^-} - \bar{A}_{i^-}.$$

The algorithm has been tested on real-life problem instances with up to 20 clients. The solutions produced by the algorithm were compared with the solutions used in practice and the results were encouraging.

VAN DER BRUGGEN, LENSTRA and SCHUUR[4] develop a solution procedure based on the Lin-Kernighan variable-depth local search method for the TSP. They use the techniques introduced by SAVELSBERGH[34,35] to handle time windows and precedence constraints efficiently. The algorithm has a construction and an improvement phase. An initial route is obtained by visiting the locations in order of increasing centers of their time window, i.e., $(e_i + l_i)/2$ for location i, taking precedence and capacity constraints into account. The resulting route may be time infeasible. Using an objective function measuring total infeasibility, the route is made feasible by iterative improvement methods. The same procedures, with a different objective function, are then applied to find a better feasible route. The method has been tested on problem instances with up to 50 clients. For all these instances the optimal solutions were known and the values of the solutions produced by the method were always within one percent of the optimal value; in many cases the method produced the optimal solution.

3.1.2. The Static Multiple-Vehicle Pickup and Delivery Problem

(A) The static *m*-PDP without time windows

Approximation

CULLEN, JARVIS and RATLIFF[6] propose an interactive approach for the multiple-vehicle dial-a-ride problem with a homogeneous fleet, i.e., equal vehicle capacities. The problem is decomposed into a clustering part and a chaining part. Both parts are solved in an interactive setting, i.e., man and machine cooperate to obtain high quality solutions. The algorithmic approach in both parts is based on set partitioning and column generation.

A cluster consists of a seed arc and a set of clients assigned to this seed arc. The total number of clients assigned to a seed arc may not exceed the vehicle capacity Q. Let (u^+, u^-) denote the seed arc of the cluster and let S denote the set of clients assigned to this seed arc. The cost c of serving this cluster is approximated by $c = 2\sum_{i \in S} d_{u^+i^+} + d_{u^+u^-} + 2\sum_{i \in S} d_{u^-i^-}$, i.e., it is assumed that a vehicle serving a cluster starts in u^+, makes a round trip to each pickup location in the cluster, travels to u^- and makes a round trip to each delivery location in the cluster.

The clustering problem, i.e., the problem of constructing and selecting clusters to serve all the clients, can be formulated as a set partitioning problem. Let J be the set of all possible clusters, i.e., seed arcs and assignments of clients to seed arcs. For each $j \in J$, let c_j denote the approximate cost of serving the cluster, and for each $i \in N$, $j \in J$ let a_{ij} be a binary constant indicating whether client i is a member of cluster j or not. Furthermore, introduce a binary decision variable y_j to indicate whether a cluster is selected or not. The clustering problem is now to

$$\text{minimize} \quad \sum_{j \in J} c_j y_j$$

subject to

$$\sum_{j \in J} a_{ij} y_j = 1 \quad \text{for all } i \in N,$$

$$y_j \in \{0, 1\} \quad \text{for all } j \in J.$$

Because the set of all possible clusters is extremely large, a column generation scheme is used to solve the linear programming relaxation of this set partitioning problem.

The master problem is initialized with all columns corresponding to clusters consisting of a single client. The row prices $(\pi_1, \pi_2, \ldots, \pi_n)$ are computed and used to define a subproblem to generate columns with negative reduced costs, i.e., clusters that correspond to attractive columns for the master problem. The master problem now heuristically tries to improve the current solution by using (some of) the new columns. Then new row prices are calculated and the subproblem is solved again.

The subproblem that has to be solved is a location-allocation problem. Let $c_{ij} = 2d_{u_j^+ i^+} + 2d_{u_j^- i^-}$ and $f_j = d_{u_j^+ u_j^-}$. For each $j \in J$ the binary variable x_j denotes whether cluster j is used or not. For each $i \in N$, $j \in J$ the binary variable z_{ij} denotes whether client i is assigned to cluster j or not. The subproblem has the following mathematical programming formulation:

$$\text{minimize} \quad \sum_{j \in J} \sum_{i \in N} (c_{ij} - \pi_i) z_{ij} + \sum_{j \in J} f_j x_j$$

subject to

$$\sum_{i \in N} z_{ij} \leq Q x_j \quad \text{for all } j \in J,$$

$$\sum_{j \in J} z_{ij} \leq 1 \quad \text{for all } i \in N,$$

$$z_{ij} \in \{0, 1\} \quad \text{for all } i \in N, j \in J,$$

$$x_j \in \{0, 1\} \quad \text{for all } j \in J.$$

Because of the size of J, the subproblem is still very difficult to solve. Therefore, it is solved approximately: Choose a set of clients to form the seed arcs. With these seed arcs fixed, solve the resulting assignment problem. With these assignments fixed, solve the resulting location problem. Continue alternating between the assignment problem and the location problem for some specified number of iterations, or until no further improvement in the objective function is found.

The clusters in the solution to the clustering problem, together with some promising clusters that did not appear in the final solution, are input to the chaining problem. In the chaining problem a subset of these clusters, partitioning the set of clients, is selected and linked to form pickup and delivery routes. In the set partitioning formulation of this problem the rows correspond to clients again, but columns now correspond to vehicle routes. The column generation procedure links a subset of the seed arcs of the selected clusters. Each linking of seed arcs is then translated into a column for the set partitioning problem by placing a 1 in row i if client i is part of one of the clusters in the linking.

Preliminary computational results, based on three problem instances with up to 100 points, show that the method is very effective. For the three instances used in the computational experiment the best known solution was either found or improved.

(B) The static *m*-PDP with time windows

Optimization

DUMAS, DESROSIERS and SOUMIS[14] present a set partitioning formulation for the static pickup and delivery problem with time windows and a column generation scheme to solve it to optimality. The approach is very robust in the sense that it can be adapted easily to handle different objective functions and variants with multiple depots and an inhomogeneous fleet of vehicles.

Let Ω be the set of all feasible pickup and delivery routes. For each route $r \in \Omega$, we denote the cost of the route by c_r. For all $i \in N$, $r \in \Omega$ let a_{ir} be a binary constant indicating whether transportation request i is served on route r ($a_{ir} = 1$) or not ($a_{ir} = 0$). Furthermore, introduce a binary variable x_r indicating whether route r is used in the optimal solution ($x_r = 1$) or not ($x_r = 0$). The static PDP can now be formulated as follows:

$$\text{minimize} \quad \sum_{r \in \Omega} c_r x_r$$

subject to

$$\sum_{r \in \Omega} a_{ir} x_r = 1 \quad \text{for all } i \in N,$$

$$\sum_{r \in \Omega} x_r = |M|,$$

$$x_r \in \{0, 1\} \quad \text{for all } r \in \Omega.$$

The cardinality of Ω is much too large to allow an exhaustive enumeration. Therefore, a column generation method is used to solve the linear relaxation of the set partitioning problem. Columns, defining admissible routes, are generated as needed by solving a constrained shortest path problem on a perturbed distance matrix. The perturbation of the distance matrix depends on the dual variables of the set partitioning problem. The lower bound found in this way is usually excellent and constitutes a

good starting point for a branch and bound approach to obtain an integral solution. The constrained shortest path problem is solved by dynamic programming. The dynamic programming algorithm employed is essentially the same as the one in DESROSIERS, DUMAS, and SOUMIS[9] (see Section 4.1.1.), except for the fact that not all transportation requests have to be processed. With this approach problem instances with up to 55 clients and 22 vehicles have been solved.

Approximation

DUMAS, DESROSIERS and SOUMIS[13] develop an approximation algorithm for the dial-a-ride problem based on their optimization algorithm discussed above. The basic idea is to create route segments for small groups of clients, called *mini-clusters*. A minicluster is a segment of a route starting and ending with an empty vehicle. Each minicluster is then treated as a transportation request that entirely fills a vehicle. The optimization algorithm is now applied to this set of transportation requests. This reduces the number of rows in the set partitioning matrix. The subproblem is now much easier to solve because each transportation request corresponds to a full truck load. Note that in the approach of CULLEN, JARVIS and RATLIFF[6] clusters cannot be identified with rows because they do not partition the set of clients.

Miniclusters are constructed simply by taking a known pickup and delivery plan, constructed using any existing algorithm, and cutting it into pieces, such that each piece starts and ends with an empty vehicle. In this setup, the approach can best be viewed as an improvement method.

Computational results are reported for problems with up to 31 vehicles and 190 clients (resulting in problems with up to 85 miniclusters). Larger problems, with up to 53 vehicles and 880 clients (up to 282 miniclusters), are solved using a decomposition scheme. First, the problem is decomposed by dividing the planning period into successive time slices with 30 to 40 miniclusters per time slice. Second, the problems associated with the time slices are all solved separately. Finally, the solutions for the successive time slices are combined into one overall solution.

An alternative method to construct mini-clusters is presented by DESROSIERS ET AL. [10] A parallel insertion heuristic creates mini-clusters. The insertion criteria are based on the concept of *neighboring requests*. Two requests are considered to be neighbors if they satisfy all of the following temporal and spatial restrictions: the time intervals $[e_{i^+}, l_{i^-}]$ and $[e_{j^+}, l_{j^-}]$ overlap, $t_{i^+j^+} + l_{j^+i^-} \leq \alpha t_{i^+i^-}$ or $t_{j^+i^+} + t_{i^+j^-} \leq \alpha t_{j^+j^-}$ for some constant α, the angle between the arcs (i^+, i^-) and (j^+, j^-) is less than some constant β, and the difference in cost between serving i and j together and serving i and j separately is larger than some constant γ.

JAW ET AL.[19,20] present two different approaches to the multiple-vehicle dial-a-ride problem, in which clients that are to be picked up and delivered have the following types of service constraints: Each client i specifies either a desired pickup time $\bar{D}_{i^+}$ or a desired delivery time $\bar{A}_{i^-}$, and a maximum ride time $\bar{T}_i$. The objective is to minimize a combination of customer dissatisfaction and resource usage.

JAW ET AL.[19] present a three-phase algorithm. The first phase, the grouping phase, decomposes the problem by dividing the time horizon into intervals and then assigning clients to groups according to the time interval into which their desired pickup or delivery time falls. The time intervals are chosen in such a way that it is possible to fully serve a client in two consecutive time intervals. The second phase, the clustering phase, partitions the set of clients in each time group into clusters and assigns a vehicle to each cluster. The number of clusters created is equal to the number of available vehicles. The third phase, the routing phase, constructs a route for each vehicle using a standard approximation algorithm for the single-vehicle dial-a-ride problem.

A more traditional insertion algorithm is presented in JAW ET AL.[20] Time windows on both the pickup time and delivery time of a client are defined based on a prescribed tolerance U and the specified desired pickup or delivery time. if client i has specified a desired pickup time, the actual pickup time D_{i^+} should fall within the time window $[\bar{D}_{i^-}, \bar{D}_{i^+} + U]$; if he has specified a desired delivery time, the actual delivery time A_{i^-} should fall within the window $[\bar{A}_{i^-} - U, \bar{A}_i]$. Moreover, his actual travel time should not exceed his maximum ride time: $A_{i^-} - D_{i^+} \leq \bar{T}_i$. Note that this information suffices to determine two time windows $[e_{i^+}, l_{i^+}]$ and $[e_{i^-}, l_{i^-}]$ for each customer i. Finally, waiting time is not allowed when the vehicle is carrying passengers.

The selection criterion is simple: customers are selected for insertion in order of increasing e_{i^+}. The insertion criterion is as follows: among all feasible points of insertion of the customer into the vehicle schedules, choose the cheapest; if no feasible point exists, introduce an additional vehicle.

For the identification of feasible insertions, the notion of an *active period* is introduced. This is a period of time a vehicle is active between two successive periods of slack time. For a given route, let

$\hat{A}_1, \hat{A}_2, \ldots$ be the arrival times at the consecutive locations visited on the route. For each visit to an address i during an active period, we define the following possible forward and backward shifts:

$$P_i^- = \min\left\{ \min_{j \le i} \left\{ \hat{A}_j - e_j \right\}, \varrho \right\},$$

$$P_i^+ = \min_{j \le i} \left\{ l_j - \hat{A}_j \right\},$$

$$S_i^- = \min_{j \ge i} \left\{ \hat{A}_j - e_j \right\},$$

$$S_i^+ = \min\left\{ \min_{j \ge i} \left\{ l_j - \hat{A}_j \right\}, \sigma \right\},$$

where ϱ and σ are the durations of the slack periods immediately preceding and following the active period in question. P_i^- (P_i^+) denotes the maximum amount of time by which every stop preceding but not including i can be advanced (delayed) without violating the time windows, and S_i^- (S_i^+) denotes the maximum amount of time by which every stop following but not including i can be advanced (delayed). These quantities thus indicate how much each segment of an active period can be displayed to accommodate an additional customer. Once it is established that some way of inserting the pickup and delivery of customer i satisfies the time window constraints, it must be ascertained that it satisfies the maximum travel time constraints.

PSARAFTIS[32] compares and tests these two approaches and concludes that the three-phase algorithm can be best used as a fast planning tool or as a device to produce good starting solutions in an operational situation, whereas the insertion algorithm can form the basis of an operational scheduling system that would assist the dispatcher in the actual execution of the schedule. These algorithms can handle fairly large problem instances effectively and efficiently. Solutions are reported for problem instances with up to 2600 customers and 20 simultaneously active vehicles.

BODIN and SEXTON[3] extend their algorithm for the single-vehicle dial-a-ride problem with desired delivery times to a two-phase algorithm for the multiple-vehicle dial-a-ride problem with desired delivery times. In the first phase, the clients are assigned to vehicles and a single-vehicle dial-a-ride problem is solved to construct an initial feasible solution. In the second phase, the algorithm attempts to reassign clients with bad service to other vehicles in order to find a better solution. This swapping of clients is done in the following way: the algorithm passes repeatedly through the client list, each time reassigning clients with a client inconvenience larger than some threshold. This threshold depends on the pass the algorithm is on. The algorithm has been tested on problems with up to 85 clients and 7 vehicles.

3.1.3. The Static Full-Truck-Load Pickup and Delivery Problem

The full-truck-load problem is the special case of the GPDP in which each load has to be transported directly from its origin to its destination, i.e., $|N_i^+| = |N_i^-| = 1$ for all $i \in N$, $\bar{q}_i = 1$ for all $i \in N$, and $Q_k = 1$ for all $k \in M$. In these problems, a transportation request is usually called a trip.

The full-truck-load problem is an interesting special case since it can be easily transformed into an asymmetric TSP in the single-vehicle case and an m-TSP in the multiple-vehicle case. The set of vertices corresponds to the set of transportation requests, and the distance between two vertices i, j is taken to be $d_{i^-j^+}$.

We discuss two papers that address a version of the full-truck-load problem in which the number of physical locations is small with respect to the number of transportation requests, i.e., each physical location is origin or destination of many requests.

BODIN, ET AL.[2] present two different route first-cluster second approximation procedures for the static full-truck-load problem. Both procedures construct one giant route servicing all transportation requests, and then divide this route into feasible vehicle routes. The first procedure constructs the giant route by solving a rural postman problem. The objective in this problem is to find a minimum cost cycle that traverses each arc in a given subset of arcs, in this case all arcs corresponding to traveling from an origin to a destination, at least once. The second procedure applies the above suggested transformation and constructs a giant route by solving an asymmetric TSP. On test problems with up to 800 requests and 45 vehicles the first procedure performed better than the second procedure.

DESROSIERS ET AL.[11] develop an optimization algorithm for a multiple-depot full-truck-load problem in which the objective is to minimize the total distance traveled and there is a restriction on the length of the routes. The problem is transformed into an asymmetric TSP with a reduced set of subtour elimination constraints, prohibiting only subtours not including any depot, and distance constraints, prohibiting illegal subpaths of the Hamiltonian cycle. The transformed problem is solved by an algorithm of LAPORTE, NOBERT and TAILLEFER,[24] which is a direct extension of the algorithm of CARPANETO and TOTH[7] for the asymmetric TSP. Test problems with 15 physical locations up to 104

requests, 1, 2 or 3 depots and up to 18 vehicles per depot were solved with this algorithm.

3.2. The Dynamic Pickup and Delivery Problem

As in most combinatorial optimization problems, dynamic aspects of the pickup and delivery problem are not very well studied.

3.2.1. The Dynamic Single-Vehicle Pickup and Delivery Problem

PSARAFTIS[29] extends the dynamic programming algorithm described in Section 3.1.1 for the static immediate request dial-a-ride problem to the dynamic case. Indefinite deferment of customers, i.e., continuously reassigning service of a customer to the last position in the pickup and delivery sequence, is prevented with a special priority constraint.

The times at which requests for service are received define a natural order among the clients. In general, the position that a particular customer holds in the sequence of pickups will not be the same as his position in this natural order. The difference between these two positions defines a *pickup position shift*. A *delivery position shift* can be similarly defined as the difference between the position in the sequence of deliveries and the position in the natural order. A priority constraint bounding the two position shifts from above prevents indefinite deferment.

The dynamic problem is solved as a sequence of static problems. Each time a new request for service is received, a slightly modified instance of the static problem is solved to update the current route. Obviously, all clients that have already been picked up and delivered can be discarded and the new client has to be incorporated. The starting location of the vehicle and the origins of the clients that have been picked up but not yet delivered have to be set to the location of the vehicle at the time of the update.

3.2.2. The Dynamic Multiple-Vehicle Pickup and Delivery Problem

PSARAFTIS[33] develops an algorithm for the dynamic multiple-vehicle problem in which the vehicles are in fact ships. In this case, the capacity of the ports also has to be considered in order to avoid waiting times when loads are to be picked up or delivered. The algorithm is based on a rolling horizon principle. Let t_k be the 'current time', i.e., the time at the kth iteration of the procedure. At time t_k the algorithm only considers those known loads i whose earliest pickup time e_{i^+} falls between t_k and $t_k + L$, where L, the length of the rolling horizon, is a user input. The algorithm then makes a tentative assignment of loads to eligible ships. However, only loads with e_{i^+} within the interval $[t_k, t_k + \alpha L]$ for some $\alpha \in (0,1)$ are considered for permanent assignment. In this way the algorithm places less (but still some) emphasis on the less reliable information about the future. Iteration $k+1$ will move the 'current time' to t_{k+1}, which is equal to the time a significant input update has to be made, or to the lowest e_{i^+} of all yet unassigned loads, whichever of the two is the earliest.

The tentative assignment of loads to ships is calculated in two phases. First the utility u_{ij} of assigning load i to ship j is calculated for each load i and ship j. This utility is a complicated function measuring the assignment's effect on (1) the delivery time of load i, (2) the delivery times of all other loads already assigned to ship j, (3) the efficiency of use of ship j, and (4) the system's port resources. In the second phase the following assignment problem is solved:

$$\text{maximize} \qquad \sum_i \sum_j u_{ij} x_{ij}$$

subject to

$$\sum_i x_{ij} \le K \qquad \text{for all } j$$

$$\sum_j x_{ij} \le 1 \qquad \text{for all } i$$

$$x_{ij} \in \{0,1\} \qquad \text{for all } i, j,$$

where K is a user specified integer denoting that no more than K loads are assigned to each ship.

3.2.3. The Dynamic Full-Truck-Load Pickup and Delivery Problem

POWELL[25,26,27,28] and FRANTZESKAKIS and POWELL[17] consider a dynamic full-truck-load pickup and delivery problem in which transportation requests may be rejected. An algorithm is developed based on network flow representation of the problem. In this model the planning area is divided into regions R and the time axis with time horizon P is divided into days $\{0, 1, 2, \ldots, P\}$. The network has node set $\{R \times \{0, 1, 2, \ldots, P\}) \cup S \cup T$, where S represents a source and T represents a sink. The arc set consists of arcs representing loaded moves that are known at time $t = 0$, arcs represention empty moves, all arcs $((r,t),(r,t+1))$, where $r \in R$ and $0 \le t < P$, and all arcs $(S,(r,0))$ and $((r,P),T)$, where $r \in R$. Arcs representing a loaded move have capacity 1, arcs $(S,(r,0))$ have capacity equal to the number of vehicles available

in region r on day 0. All other arcs have capacity ∞. The profit of an arc representing a move is the net profit or cost of the corresponding move. The profit of all other arcs is 0. Note that a maximum profit flow in this network corresponds to an allocation of vehicles to transportation requests, but that it is not required to honor all transportation requests.

To anticipate future transportation requests, the network is extended with stochastic links. These stochastic links correspond to future uncertain trajectories of vehicles. They emanate from each node (r, t) and end in T. The profit of the kth stochastic link emanating from (r, t) reflects the expected marginal value of another vehicle in region r on day t given that there are already k vehicles in region r on day t. These expected marginal values of an extra vehicle are based on historical data. Each stochastic link has capacity 1. A maximum profit flow in this extended network not only represents a deterministic allocation of vehicles to loads known at $t = 0$, but also assigns vehicles to regions in order to be able to serve furture transportation requests at minimal cost. The use of a stochastic link from (r, t) to T indicates that at time t the vehicle will be available in region r to serve some request that is not yet known. Because the vehicles are only allocated to known loads, the network has to be reoptimized a few times each day.

4. CONCLUSION

IN THIS PAPER, we have discussed various characteristics of pickup and delivery problems and have given an overview of the solution approaches that have been proposed. In the process, we have identified several important research topics.

Most real-life pickup and delivery problems that we are aware of are demand responsive. Currently, very little is known about on-line algorithms for dynamic pickup and delivery problems. Besides the obvious practical importance of such algorithms, it seems to be a fascinating and challenging research area as well.

Although the single-vehicle pickup and delivery problem is $\mathcal{NP}$-hard, it can be solved very efficiently with dynamic programming as long as the number of transportation requests is relatively small, which is the case in many practical situations. Therefore, the main problem in solving multiple-vehicle pickup and delivery problems is the assignment of transportation requests to vehicles.

Consequently, pickup and delivery problems, as well as many other routing and scheduling problems, seem to be well suited for solution approaches based on set partitioning with column generation. Although this approach has already been explored successfully, it is likely to remain an active research area for the next decade.

An important obstacle in the development of effective and efficient approximation algorithms is the lack of a reliable measure of closeness. In vehicle routing problems, customers that are geographically close to each other are likely to be served by the same vehicle. A concept similar to that of geographical closeness in vehicle routing problems does not exist for pickup and delivery problems. Although several alternatives have been proposed, none of them has turned out to be entirely satisfactory. The development of a useful closeness concept seems crucial for progress in approximation algorithms.

Many interesting questions arise in the context of interactive planning systems for pickup and delivery problems. Representing solutions, for instance, is nontrivial: an optimal solution to a single vehicle pickup and delivery problem may look very bad when drawn on a map, even without time windows and with infinite vehicle capacity.

REFERENCES

1. J. BEARDWOOD, J. H. HALTON AND J. M. HAMMERSLEY, "The Shortest Path through Many Points," *Proc. Cambridge Philos. Soc.* **55**, 299–327 (1959).
2. L. BODIN, B. GOLDEN, A. ASSAD AND M. BALL, "Routing and Scheduling of Vehicles and Crews—The State of the Art," *Comput. Oper. Res.* **10**, 63–211 (1983).
3. L. BODIN AND T. SEXTON, *The Multi-Vehicle Subscriber Dial-a-Ride Problem*, Management Science & Statistics Working Paper No. 83-009, University of Maryland at College Park, MD (1983).
4. L. J. J. VAN DER BRUGGEN, J. K. LENSTRA AND P. C. SCHUUR, *A Variable Depth Approach for the Single-Vehicle Pickup and Delivery Problem with Time Windows*, COSOR Memorandum 90-48, Eindhoven University of Technology (1990).
5. N. CHRISTOFIDES, *Worst Case Analysis of a New Heuristic for the Travelling Salesman Problem*, Report 388, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, PA (1976).
6. F. H. CULLEN, J. J. JARVIS AND H. D. RATLIFF, "Set Partitioning Based Heuristics for Interactive Routing," *Networks* **11**, 125–143 (1981).
7. G. CARPANETO AND P. TOTH, "Some New Branching and Bounding Criteria for the Asymmetric Travelling Salesman Problem," *Management Sci.* **26**, 736–743 (1980).
8. M. DESROCHERS, J. K. LENSTRA, M. W. P. SAVELSBERGH AND F. SOUMIS, "Vehicle Routing with Time Windows: Optimization and Approximation," in *Vehicle Routing: Methods and Studies*, B. L. Golden and A. A. Assad (eds.), North-Holland, Amsterdam, 1988.

9. J. DESROSIERS, Y. DUMAS AND F. SOUMIS, "A Dynamic Programming Solution of the Large-Scale Single-Vehicle Dial-a-Ride Problem with Time Windows," *Amer. J. Math. Management Sci.* **6**, 301–325 (1986).
10. J. DESROSIERS, Y. DUMAS, F. SOUMIS, S. TAILLEFER AND D. VILLENEUVE, *An Algorithm for Mini-Clustering in Handicapped Transport*, Report G-91-02, Ecole des Hautes Etudes Commerciales, Montreal (1991).
11. J. DESROSIERS, G. LAPORTE, M. SAUVE, F. SOUMIS AND S. TAILLEFER, "Vehicle Routing with Full Loads," *Comput. Oper. Res.* **15**, 219–226 (1988).
12. M. DROR, G. LAPORTE AND P. TRUDEAU, "Vehicle Routing with Stochastic Demands: Properties and Solution Frameworks," *Trans. Sci.* **23**, 166–176 (1989).
13. Y. DUMAS, J. DESROSIERS AND F. SOUMIS, Large Scale Multi-Vehicle Dial-a-Ride Systems, Report G-89-30, Ecole des Hautes Etudes Commerciales, Montreal (1989).
14. Y. DUMAS, J. DESROSIERS AND F. SOUMIS, "The Pickup and Delivery Problem with Time Windows," *European J. Oper. Res.* **54**, 7–22 (1991).
15. M. T. FIALA TIMLIN AND W. R. PULLEYBLANK, *Precedence Constrained Routing and Helicopter Scheduling: Heuristic Design*, Report UW/ICR 90-02, Institute for Computer Research, University of Waterloo, Ontario, Canada (1990).
16. M. FISCHETTI AND P. TOTH, "An Additive Bounding Procedure for Combinatorial Optimization Problems," *Oper. Res.* **37**, 319–328 (1989).
17. L. F. FRANTZESKAKIS AND W. B. POWELL, "A Successive Linear Approximation Procedure for Stochastic, Dynamic Vehicle Allocation Problems," *Trans. Sci.* **24**, 40–57 (1990).
18. D. S. JOHNSON, Private Communication, (1992).
19. J. J. JAW, A. R. ODONI, H. N. PSARAFTIS AND N. H. M. WILSON, *A Heuristic Algorithm for the Multi-Vehicle Many-to-Many Advance-Request Dial-a-Ride Problem*, Working Paper MIT-UMTA – 82-3, M.I.T., Cambridge Massachusetts (1982).
20. J. J. JAW, A. R. ODONI, H. N. PSARAFTIS AND N. H. M. WILSON, "A Heuristic Algorithm for the Multi-Vehicle Advance-Request Dial-a-Ride Problem with Time Windows," *Trans. Res.* **20B**, 243–257 (1986).
21. B. KALANTARI, A. V. HILL AND S. R. ARORA, "An Algorithm for the Traveling Salesman Problem with Pickup and Delivery Customers," *European J. Oper. Res.* **22**, 377–386 (1985).
22. G. A. P. KINDERVATER AND M. W. P. SAVELSBERGH, *Local Search in Physical Distribution Management*, COSOR Memorandum 92-30, Eindhoven University of Technology (1992).
23. J. D. C. LITTLE, K. G. MURTY, D. W. SWEENEY AND C. KAREL, "An Algorithm for the Traveling Salesman Problem," *Oper. Res.* **11**, 972–989 (1963).
24. G. LAPORTE, Y. NOBERT AND S. TAILLEFER, "A Branch and Bound Algorithm for the Asymmetrical Distance Constrained Vehicle Routing Problem," *Math. Modelling* **9**, 857–868 (1987).
25. W. B. POWELL, "An Operational Planning Model for the Dynamic Vehicle Allocation Problem with Uncertain Demands," *Trans. Res.* **21B**, 217–232 (1987).
26. W. B. POWELL, "Maximizing Profits for North American Van Lines' Truckload Division: A New Framework for Pricing and Operations," *Interfaces* **18**, 21–41 (1988).
27. W. B. POWELL, "A Comparative Review of Alternative Algorithms for the Dynamic Vehicle Allocation Problem," in *Vehicle Routing: Methods and Studies*, B. L. Golden and A. A. Assad (eds.), North-Holland, Amsterdam, 1988.
28. W. B. POWELL, "Optimization Models and Algorithms: An Emerging Technology for the Motor Carrier Industry," *IEEE Trans. Vehicular Tech.* **40**, 68–80 (1991).
29. H. PSARAFTIS, "A Dynamic Programming Solution to the Single Vehicle Many-to-Many Immediate Request Dial-a-Ride Problem," *Trans. Sci.* **14**, 130–154 (1980).
30. H. PSARAFTIS, "An Exact Algorithm for the Single Vehicle Many-to-Many Dial-a-Ride Problem with Time Windows," *Trans. Sci.* **17**, 351–357 (1983).
31. H. PSARAFTIS, "Analysis of an $O(N^2)$ Heuristic for the Single Vehicle Many-to-Many Euclidean Dial-a-Ride Problem," *Trans. Res.* **17B**, 133–145 (1983).
32. H. PSARAFTIS, "Scheduling Large-Scale Advance-Request Dial-a-Ride Systems," *Amer. J. Math. Management Sci.* **6**, 327–367 (1986).
33. H. N. PSARAFTIS, "Dynamic Vehicle Routing Problems," in *Vehicle Routing: Methods and Studies*, B. L. Golden and A. A. Assad (eds.), North-Holland, Amsterdam, 1988.
34. M. W. P. SAVELSBERGH, *Computer Aided Routing*, CWI Tract 75, CWI, Amsterdam (1992).
35. M. W. P. SAVELSBERGH, "An Efficient Implementation of Local Search Algorithms for Constrained Routing Problems," *European J. Oper. Res.* **47**, 75–85 (1990).
36. T. SEXTON AND L. BODIN, "Optimizing Single Vehicle Many-to-Many Operations with Desired Delivery Times: I Scheduling," *Trans. Sci.* **19**, 378–410 (1985).
37. T. SEXTON AND L. BODIN, "Optimizing Single Vehicle Many-to-Many Operations with Desired Delivery Times: II Routing," *Trans. Sci.* **19**, 411–435 (1985).
38. M. M. SOLOMON AND J. DESROSIERS, "Time Window Constrained Routing and Scheduling Problems," *Trans. Sci.* **22**, 1–13 (1988).
39. D. M. STEIN, "An Asymptotic Probabilistic Analysis of a Routing Problem," *Math. Oper. Res.* **3**, 89–101 (1978).

(Received, December 1992; revised May 1993, February 1994; accepted February 1994)