Time Dependent Vehicle Routing Problems: Formulations, Properties and Heuristic Algorithms

CHRYSSI MALANDRAKI

Roadnet Technologies, Inc., Timonium, Maryland 21093

MARK S. DASKIN

Northwestern University, Evanston, Illinois 60208

The time dependent vehicle routing problem (TDVRP) is defined as follows. A vehicle fleet of fixed capacities serves customers of fixed demands from a central depot. Customers are assigned to vehicles and the vehicles routed so that the total time of the routes is minimized. The travel time between two customers or between a customer and the depot depends on the distance between the points and time of day. Time windows for serving the customers may also be present. The time dependent traveling salesman problem (TDTSP) is a special case of the TDVRP in which only one vehicle of infinite capacity is available. Mixed integer linear programming formulations of the TDVRP and the TDTSP are presented that treat the travel time functions as step functions. The characteristics and properties of the TDVRP preclude modification of most of the algorithms that have been developed for the vehicle routing problem. Several simple heuristic algorithms are given for the TDTSP and TDVRP without time windows based on the nearest-neighbor heuristic. A mathematical-programming-based heuristic for the TDTSP without time windows using cutting planes is also briefly discussed. Test results on small, randomly generated problems are reported.

INTRODUCTION

The time dependent vehicle routing problem (TDVRP) is defined as follows. A vehicle fleet of fixed capacities has to serve customers of fixed demands from a central depot. Customers must be assigned to vehicles and the vehicles routed so that the total time spent on the routes is minimized. The travel time between two customers or between a customer and the depot *depends on the distance between the points and the time of day*. Time windows for serving the customers may also be given as well as a maximum allowable duration of each route (work day of the driver). The time dependent traveling salesman problem (TDTSP) is a special case of the TDVRP in which only one vehicle of infinite capacity is available.

The TDVRP extends the vehicle routing problem (VRP) (CHRISTOFIDES[9, 10, 12] and BODIN et al.[7]) to account for urban congestion. Similarly, the TDTSP is an extension of the traveling salesman problem (TSP) (LAWLER et al.[44] and CHRISTOFIDES[11]). The VRP considers the cost or travel time between two points as known and constant. Usually the VRP assumes that the costs or travel times are a scalar transformation of distances. For real-life applications some composite or modified measure of cost may be used (FISHER et al.[23] and BELL et al.[5]).

The assumption that costs are deterministically known and constant is an approximation of actual conditions. In a congested urban environment, the travel time between two points is usually not a function of distance traveled alone since speeds are not constant. Fluctuations in traffic density may cause fluctuations in travel speed that result in variations in travel times. One component is the variation due to accidents, weather conditions or other random events. Another component of this variation, which may cause travel times to increase dramatically during rush hours, is the temporal

Transportation Science
Vol. 26, No. 3, August 1992

0041-1655/92/2603-185 $01.25

variation that results from the hourly, daily, weekly or seasonal cycles in the average traffic volumes.

If the major variation in travel times results from the time-of-day variation, the travel times between two points may be represented by a deterministic function of the distance between the two points and also the time of day the travel takes place. If we ignore the time-of-day dependence of travel times we may get a suboptimal solution, with a different route structure and different number of vehicles needed than would result from the time dependent optimal solution. Also, we may obtain a solution that violates time windows or maximum permissible times for each route.

As a generalization of the TSP, the TDVRP belongs to the class of NP-complete problems (LENSTRA and RINNOOY KAN[45]) for which it seems unlikely that polynomial-time exact algorithms can be developed (GAREY and JOHNSON,[26] JOHNSON and PAPADIMITRIOU[39]).

Considerable research has been devoted to the TSP (DANTZIG, FULKERSON and JOHNSON,[19, 20] LIN and KERNIGHAN,[46] HELD and KARP,[35-37] CARPANETO and TOTH,[8] CROWDER and PADBERG,[17] BALAS and CHRISTOFIDES,[4] Lawler et al.[44]), the multiple TSP (GAVISH and SRIKANTH[29]), and the VRP (CLARKE and WRIGHT,[16] GILLETT and MILLER,[30] FISHER and JAIKUMAR,[24] CHRISTOFIDES, MINGOZZI and TOTH,[13-15] GAVISH and GRAVES,[27, 28] MAGNANTI,[47] LAPORTE, NOBERT and DESROCHERS,[42] GOLDEN and ASSAD[31]).

PICARD and QUEYRANNE[52] examine the TSP with costs associated with each node depending not only on the node that precedes it but also on its position (time) in the sequence. FOX, GAVISH and GRAVES[25] give a new formulation of the previous time dependent TSP. They assume that the cost of traveling between city i and city j depends on the time period and that the travel time between any two cities is one time period. The time dependent TSP discussed in these papers[25, 52] is conceptually the closest to the TDTSP presented here. In our model, the assumption that the travel time between any two cities is one time period is relaxed and the travel times depend on the time of day and not on the sequence of node visitations for both the TDTSP and the TDVRP.

The TSP and VRP with time windows are examined by BAKER,[2] BAKER and SCHAFFER,[3] DESROSIERS, SOUMIS and DESROCHERS,[21] SOLOMON,[53, 54] and KOLEN et al.[41] SOLOMON and DESROSIERS[55] survey several types of vehicle routing problems with time windows.

The remainder of this paper is organized as follows. The TDVRP is formulated mathematically in the next section with the TDTSP as a special case. Section 2 discusses properties of the travel time functions and problem characteristics. Simple heuristic algorithms for the TDTSP and the TDVRP without time windows based on the nearest-neighbor heuristic are described in Section 3. A mathematical-programming-based heuristic for the TDTSP without time windows is presented in Section 4 and test results of the heuristics on randomly generated problems in Section 5. Section 6 summarizes the paper and presents conclusions and directions for future work.

1. PROBLEM FORMULATION

A DIRECTED graph $G(V, E)$ is given with V the set of nodes and E the set of directed links. A directed link is represented by an ordered pair of nodes (i, j) in which i is called the origin and j is called the destination of the link. The network is assumed complete and an $n \times n$ time dependent matrix $C(t) = [c_{ij}(t_i)]$ is also given representing the travel times on link $(i, j) \in E$, where $c_{ij}(t_i)$ is a function of the time of day, t_i, at the origin node i of the link.

The TDVRP is formulated as a mixed integer linear programming (MILP) problem for the case in which the travel time, $c_{ij}(t_i)$, is a known step function of the time of day, t_i, at the origin node i. In this way, the day is divided into time intervals. Once the time interval during which the salesman starts traversing link (i, j) is known, the travel time of traversing link (i, j) is a known constant.

The problem is formulated on an expanded network. Each link (i, j), from node i to node j, is replaced by M_{ij} parallel links from i to j where M_{ij} is the number of distinct time intervals considered in the step function $c_{ij}(t_i)$ representing the travel time for the link. The number M_{ij} may differ from link to link. For clarity of exposition in the following formulation we simply denote the number of time intervals by M instead of M_{ij} as if the number of time intervals is the same for all physical links in the network. A travel time step function for link (i, j) with three time intervals is shown in Figure 1.

The formulation concerns a TDVRP with exactly K vehicles that includes time windows and permits waiting at the customer nodes. It considers collections from all the customers but it can also be used for deliveries. The total route time (travel time plus service time plus waiting time) is minimized. The starting time from the depot is given. Each customer is served by one visit of one vehicle. Time windows are included that express a single time interval during which visiting a node is permitted.

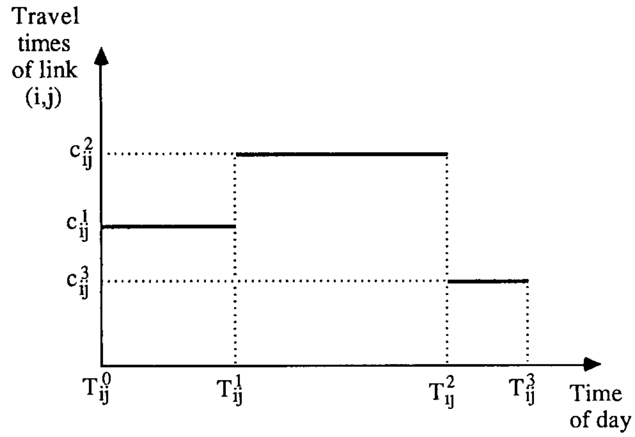


Fig. 1. Travel time step function for link (i, j) with three time intervals ($M_{ij} = 3$).

Multiple time windows can also be modeled but this will make the formulation more complicated. We assume:

1. The travel time from node i to node j during time interval m is independent of the type of vehicle. This is a reasonable assumption in an urban environment.
2. The collection (or delivery) time for each vehicle is independent of the type of vehicle and depends only on the customer. This is also a reasonable assumption.

These assumptions permit us to formulate the TDVRP without considering which vehicle visits a node.

The depot node is also expanded as follows. Consider node 1 as the starting depot; that is, delete all the inbound links to node 1. Augment the network by K nodes $(n + 1, \dots, n + K)$. These nodes correspond to returning depot nodes for each of the K vehicles respectively. Delete all outbound links from nodes $n + 1, \dots, n + K$ and all the links interconnecting the depot nodes. Thus all vehicles start from the depot node 1 and each vehicle has to return to its specified return depot. Also set $c_{ij}(t_i) = c_{i1}(t_i)$ for every node $i = 2, \dots, n$ and $j = n + 1, \dots, n + K$.

The notation used in the formulation is summarized below:

Constants

n = number of nodes including the depot
M = number of time intervals considered for each link
K = number of vehicles
c_{ij}^m = travel time from node i to j if starting at i during time interval m; $c_{ii}^m = \infty$ for all i, m
c_i = service time at node i (e.g., delivery time); $c_i = 0$ for $i = 1, n + 1, \dots, n + K$
T_{ij}^m = upper bound for time interval m for link (i, j)—see Figure 1
t = the starting time from the depot node 1
b_k = weight (or volume) capacity of vehicle k
d_i = weight (or volume) to be collected at customer i; $d_i = 0$ for $i = 1, n + 1, \dots, n + K$
B_1 = a large number
B_2 = a large number
$B = \max_k b_k$ = capacity of largest vehicle
L_i = earliest time that the salesman can arrive at node i
U_i = latest time that the salesman can arrive at node i

Decision variables

$$x_{ij}^m = \begin{cases} 1 & \text{if any vehicle travels directly from node } i \text{ to node } j \text{ starting from } i \text{ during time interval } m \\ 0 & \text{otherwise} \end{cases}$$

t_j = departure time of any vehicle from node j
w_j = weight (or volume) larger than or equal to that carried by a vehicle when departing from node j.

With this notation the TDVRP may be formulated as follows:

$$\text{Min} \sum_{k=1}^{K} t_{n+k} \tag{1}$$

subject to

$$\sum_{\substack{i=1 \\ i \neq j}}^{n} \sum_{m=1}^{M} x_{ij}^m = 1 \qquad (j = 2, \dots, n + K) \tag{2}$$

$$\sum_{\substack{j=2 \\ j \neq i}}^{n+K} \sum_{m=1}^{M} x_{ij}^m = 1 \qquad (i = 2, \dots, n) \tag{3}$$

$$\sum_{j=2}^{n} \sum_{m=1}^{M} x_{1j}^m = K \tag{4}$$

$$t_1 = t \tag{5}$$

$$t_j - t_i - B_1 x_{ij}^m \geqslant c_{ij}^m + c_j - B_1 \tag{6}$$

$$(i = 1, \dots, n; j = 2, \dots, n + K; i \neq j; m = 1, \dots, M)$$

$$\left.\begin{aligned} t_i + B_2 x_{ij}^m &\leqslant T_{ij}^m + B_2 \quad (7) \\ t_i - T_{ij}^{m-1} x_{ij}^m &\geqslant 0 \quad (8) \end{aligned}\right\}$$

$$(i = 1, \ldots, n;\ j = 2, \ldots, n + K;\ i \neq j;\ m = 1, \ldots, M)$$

$$L_i + c_i \leqslant t_i \leqslant U_i + c_i \qquad (i = 1, \ldots, n + K) \quad (9)$$

$$w_j - w_i - B \sum_{m=1}^{M} x_{ij}^m \geqslant d_j - B \quad (10)$$

$$(i = 1, \ldots, n;\ j = 2, \ldots, n + K;\ i \neq j)$$

$$w_1 = 0 \quad (11)$$

$$w_{n+k} \leqslant b_k \qquad (k = 1, \ldots, K) \quad (12)$$

$$x_{ij}^m = 0 \text{ or } 1 \qquad \text{for all } i, j, m \quad (13)$$

$$t_i \geqslant 0 \qquad \text{for all } i \quad (14)$$

$$w_i \geqslant 0 \qquad \text{for all } i. \quad (15)$$

The objective function (1) minimizes the total route time of all vehicles (travel time plus service time at all nodes plus the waiting time at all nodes). Constraints (2) to (4) ensure that each customer is visited exactly once and exactly K vehicles are used. If we want to permit less than K vehicles to be used we can replace constraint (4) by constraint $\sum_{j=2}^{n} \sum_{m=1}^{M} x_{1j}^m \leqslant K$ and modify constraints (2) to read $\sum_{i=1}^{n} \sum_{m=1}^{M} x_{ij}^m \leqslant 1$ for $j = n + 1, \ldots, n + K$. Equivalently, we can replace constraint (4) by constraint $\sum_{j=2}^{n+K} \sum_{m=1}^{M} x_{1j}^m = K$ to allow a vehicle to go directly from the starting depot 1 to its own return depot (after including links $(1, n + k)$, for $k = 1, \ldots, K$ in the network).

Constraint (5) sets the starting time from the depot node 1 equal to t for all vehicles. If we omit constraint (5) the starting time of all the vehicles will be the same but it will be a variable determined by the optimization. To permit vehicles to start at different times, we split the starting depot node 1 into K nodes, one for each vehicle, in a manner similar to that in which the return depot was expanded, and use a different variable for the starting time of each vehicle.

Constraints (6) compute the departure time at node j (GOLDEN, MAGNANTI and NGUYEN[32]). The objective function (1) ensures that constraints (6) apply with equality when $x_{ij}^m = 1$ except in cases when waiting at j decreases the objective function value. This point will be discussed again in Section 2. If we want constraints (6) to apply always with equality (no waiting permitted) we must add the following constraints

$$t_j - t_i + B_1 x_{ij}^m \leqslant c_{ij}^m + c_j + B_1$$

$$(i = 1, \ldots, n;\ j = 2, \ldots, n + K;\ i \neq j;\ m = 1, \ldots, M) \quad (16)$$

that operate in a manner similar to constraints (6). The inclusion of constraints (16) is not indicated when time windows are present. We can set the large number B_1 equal to the total route time of a set of feasible vehicle tours plus $\max_{i,j,m} c_{ij}^m$ plus $\max_j c_j$.

The temporal constraints (7) and (8) ensure that the proper parallel link m is chosen between nodes i and j according to the departure time from node i. We want to model the following constraints

$$\text{if } x_{ij}^m = 1 \text{ then } T_{ij}^{m-1} \leqslant t_i \leqslant T_{ij}^m \text{ for all } i, j, m.$$

This means that t_i belongs to the time interval m defined by the above inequalities if the link used in leaving node i corresponds to the same time interval m. We can set the large number B_2 equal to the latest possible return time of a vehicle. If for some particular i, j and m, $x_{ij}^m = 1$ then the relevant link corresponding to the mth interval is used. If $x_{ij}^m = 0$ though, the relevant interval is not m and constraints (7) and (8) are not binding. Note that consecutive time intervals share a boundary point since all time intervals are considered closed. Thus, if $t_i = T_{ij}^m$ either x_{ij}^m or x_{ij}^{m+1} may be equal to one.

Constraints (9) impose the time windows that are defined in terms of the *arrival* times at the nodes while the variables t_i for $i = 1, \ldots, n + K$ represent the *departure* times from the nodes. Inclusion of time windows may actually facilitate the solution by excluding some time interval choices and decreasing the total number of links in the expanded network.

Constraints (10) to (12) impose the capacity restrictions. Constraint (11) states that all vehicles leave the depot node 1 empty. Constraints (10) ensure that the weight carried by the vehicle leaving customer j is at least equal to the weight when leaving the previously visited customer i plus the weight of the commodity picked up at customer j. Constraints (12) ensure that the capacity of each vehicle is not surpassed.

The formulation does not require subtour elimination (SE) constraints because both constraints (5) and (10) operate as (separate) SE constraints. These constraints are a generalization of the following SE constraints for the TSP (MILLER, TUCKER and ZEMLIN[50]):

$$u_i - u_j + n x_{ij} \leqslant n - 1 \qquad (2 \leqslant i \neq j \leqslant n)$$

where $x_{ij} = 1$ if link (i, j) is used and $x_{ij} = 0$ otherwise, and the u_i for $i = 2, \ldots, n$ are arbitrary real numbers.

Restrictions on the duration of any vehicle tour may easily be imposed by adding the constraint

$t_{n+k} \leqslant T_k^{\text{end}}$ for all k where T_k^{end} is the latest time vehicle k must return to the depot.

The TDVRP formulation can be easily modified to model both deliveries from the depot and collections from the customers by the same vehicle of a nonhomogeneous commodity for the case where all the vehicles have the same capacities B. This and other extensions of existing formulations of the VRP to model the TDVRP can be found in MALANDRAKI.[48]

The TDTSP is obtained from the TDVRP formulation as a special case if we set $K = 1$ and omit the capacity constraints (10) to (12) and (15). The MILP formulation is very large involving many binary variables and a large number of temporal and capacity constraints.

2. PROPERTIES AND CHARACTERISTICS

2.1. Properties of the Travel Time Functions

In reality, travel times change continuously as a function of time and not in discrete jumps. When waiting is permitted at the nodes, the step functions used in the formulation presented in Section 1 will generally behave as if they were piecewise linear *continuous* functions when the travel time in period $m + 1$ is less than that in period m.

To illustrate this, consider finding the time dependent shortest path (a shortest path with time dependent travel times) from node 1 to node 3 in Figure 2a. Figure 2b gives the travel time function for link (2, 3). The scales of the vertical and horizontal axes are identical. If the traveler is ready to

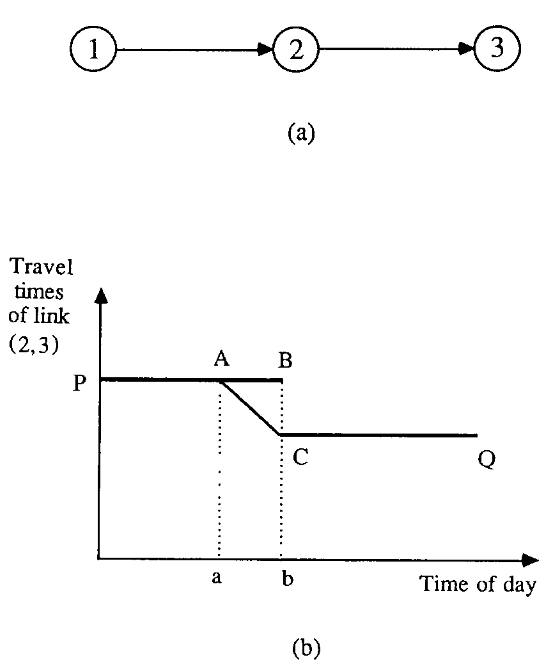


Fig. 2. (a) Example network. (b) Travel time step function.

depart from node 2 at any time t_2 such that $a < t_2 < b$, it is advantageous to wait and depart at time b. Thus, the effective travel time function for link (2, 3) is given by the piecewise continuous function P-A-C-Q instead of the step function P-B, C-Q.

In the presence of time windows (constraint (9)), scenarios can be constructed in which it is infeasible for the model to allow waiting at a node by implicitly using the more realistic continuous travel time function, even if waiting would decrease the objective function. Also, when the travel time function increases from period m to period $m + 1$, the effective travel time functions remain discontinuous. These and related properties of the travel time functions are discussed in Malandraki.[48] Finally, we note that the heuristics for the TDVRP and TDTSP without time windows presented below as well as the dynamic programming algorithm for the TDTSP may all use continuous (nonlinear) travel time functions. However, direct extension of shortest path or dynamic programming TSP algorithms to include time dependence is not valid for decreasing step functions or decreasing continuous travel time functions with a slope smaller than minus one, if waiting is not permitted. The travel time step functions are needed in the cutting-plane-based heuristic for the TDTSP presented in Section 4.

2.2. Problem Characteristics

The TDVRP is an *asymmetric problem.* Suppose that the travel time functions are symmetric, that is, the travel time function for link (i, j) is the same as the travel time function for link (j, i). This does not imply that two tours that traverse the same physical links but in the opposite direction have the same total tour times.

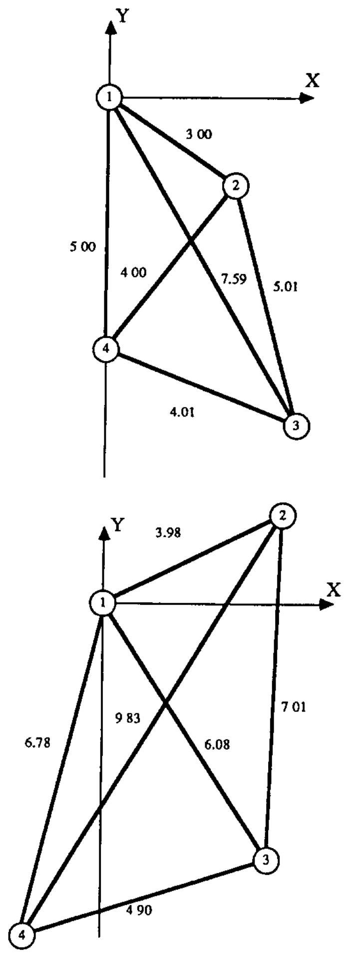
Another characteristic of the TDVRP is that *the k-opt exchange or insertion heuristics cannot be easily extended to solve the problem.* The k-opt exchange solution technique tries to improve a solution by exchanging k links while retaining feasibility. When the method is applied to the symmetric TSP only the costs of the exchanged links need to be taken into consideration. When the method is applied to the TDTSP though, the travel times of more links may change either because the traversal direction changes or because the starting time at the origin node of the link changes. Consequently, the application of the method becomes expensive computationally. Similarly, the application of insertion heuristics is expensive because the travel times of links subsequent to the inserted link may change.

Two properties (LARSON and ODONI[43]) of the optimal solution of the Euclidean TSP (where travel times are Euclidean) do not extend to the TDTSP.

These are: (1) the fact that the optimum traveling salesman tour does not intersect itself, and (2) the convex hull property.

The "Euclidean" TDTSP is defined as follows. Without loss of generality we assume that the same time periods are applicable for all links, that is, $M_{ij} = M$ for all i, j. We also assume that the travel times on the links for each time period separately are Euclidean; that is, each network defined only by the links of the same time period is Euclidean. Note that each of these networks represents "positions" of nodes during a particular time period and the links represent travel times that differ from period to period. Hence, the relative positions of nodes may "change" for different time periods.

The convex hull is not well defined for the "Euclidean" TDTSP. Since the "positions" of the nodes change, the convex hull may differ from period to period. But even when the points defining the convex hull as well as their order of appearance on the hull are the same for all periods, the two-period, four-node example of Figure 3 shows that neither of the above properties hold for the "Euclidean" TDTSP. Time is measured in hours and the time periods are the same for all the links. The first period (Fig. 3a) applies for $0{:}00 \leqslant t_i < 6{:}00$ for all nodes i, where t_i is the departure time from origin node i. The second time period (Fig. 3b) applies for $t_i \geqslant 6{:}00$ for all i. The starting node is node 1 and the starting time is 0:00. The coordinates are shown next to the figures. The numbers next to the links denote the travel times which are the same in both directions.



$t_i < 6{:}00$ ∀ node i

Node	X	Y
1	0.00	0.00
2	2.40	-1.80
3	3.65	-6.65
4	0.00	-5.00

(a) First time period.

$t_i \geq 6{:}00$ ∀ node i

Node	X	Y
1	0.00	0.00
2	3.55	1.80
3	3.15	-5.20
4	-1.55	-6.60

(b) Second time period.

Fig. 3. Example for TDTSP convex hull properties. Starting node: 1. Starting time: 0:00. Numbers next to links denote travel times.

There are six possible tours for a four-node network with a given starting node. The tour with the minimum total travel time is 1-2-4-3-1 with a total travel time of 17.98 (rounded to the second digit). The shortest tour crosses itself and violates the convex hull property. Since the convex hull property may be violated by the optimal solution of the TDTSP, the convex hull heuristic cannot be extended to solve the TDTSP.

The "Euclidean" TDTSP is a restrictive case of the problem. Since the properties do not hold for the "Euclidean" TDTSP, they do not hold for the general TDTSP.

3. HEURISTIC ALGORITHMS FOR THE TDTSP AND TDVRP

THE CHARACTERISTICS of the TDTSP and TDVRP discussed in Section 2, preclude the efficient extension of most of the heuristics for the nontemporal problem. This section presents simple heuristic algorithms for the TDTSP (Section 3.1) and the TDVRP (Section 3.2) without time windows. The algorithms are based on the nearest-neighbor (greedy) heuristic for the TSP (Bodin et al.[7]) that can easily be extended to account for time dependence.

3.1. Heuristic Algorithms for the TDTSP

The nearest-neighbor heuristic is extended in a straightforward manner to solve the TDTSP with a given starting time and without considering waiting and time windows. When step functions are used, the algorithm must find the proper parallel link every time a physical link is examined. If the parallel links are examined sequentially, this algorithm (heuristic NN1) requires $O(n^2M)$ time in the worst case. A variation of the heuristic is to apply the algorithm $n - 1$ times with each node (apart from the depot) to be visited second. The best solution, among the $n - 1$ solutions, is selected. Using this extension (heuristic NN2), the solution tour can be found in $O(n^3M)$ time.

Since it is well known that the nearest-neighbor heuristic does not give good solutions for the TSP, we describe next a variation of the nearest-neighbor TDTSP heuristic that attempts to ameliorate the results of the greedy approach. It selects the next node to be visited randomly according to a user-specified probability distribution. Similar probabilistic approaches have been used before. For example, FEO and BARD[22] use it in a probabilistic set covering heuristic. The probabilistic, nearest-neighbor heuristic (NNR for Nearest-Neighbor Random) may be described as follows:

Step 1. Start from the depot at the given starting time.

Step 2. Sort in nondecreasing order up to k feasible links with the k best times of traversal that originate at the last-visited node and terminate at a nonvisited node and do not violate the temporal requirements (i.e., a feasible parallel link is chosen). From this list, select one link randomly according to a given probability distribution. Add this link to the tour.

Step 3. Repeat Step 2 until all nodes are visited. Then return to the depot from the last node.

Step 4. Repeat Steps 1, 2 and 3 L times (with a new starting random number each time) and select the best tour.

For the implementation of this algorithm $L = n$ and $k = 3$ were used. The k best elements out of n can be sorted in $O(kn)$ time when insertion sorting is used (AHO, HOPCROFT and ULLMAN[1]). Hence, using the NNR heuristic the solution can be found in $O(Ln^2(M + k))$ time in the worst case and for $L = n$ in $O(n^3(M + k))$ time. Since L is a user-specified parameter, an important feature of the probabilistic approach is that a better solution may be found if more computational time is used.

3.2. Heuristic Algorithms for the TDVRP

3.2.1. Sequential Route Construction TDVRP Heuristic

In this heuristic, a new vehicle is introduced when no more customers can be accommodated using the current vehicle.

Step 1. Start from the depot at the given starting time with the first available vehicle in the input file.

Step 2. Find a nonvisited node such that the link from the last-visited node to this node has the smallest time of traversal and does not violate the temporal requirements and the capacity restrictions for the current vehicle. Add this node to the tour of the current vehicle.

Step 3. Repeat Step 2 until all nodes are visited or the current vehicle is filled and return to the depot. If all nodes are visited, stop. Otherwise, if no more vehicles are available indicate insufficient capacity and stop. If more vehicles are available start from the depot at the given starting time with the next available vehicle in the input file. Go to Step 2.

The heuristic may indicate insufficient capacity when in fact a feasible solution does exist. It will always find a feasible solution, though, if $n - 1$ vehicles are available, each with capacity larger than or equal to the largest customer demand.

Two versions of the last heuristic have been coded. One version examines the vehicles from first to last in the input file and the other from last to first vehicle. Obviously the two versions coincide for a homogeneous fleet of vehicles.

Using this heuristic, a solution can be found in $O((n + K)nM)$ time in the worst case, where K is the number of available vehicles and parallel links are examined sequentially. Since each customer is served by one vehicle, at most $n - 1$ vehicles will be used and the worst time performance is bounded by $O(n^2M)$ where K has been replaced by n.

3.2.2. Simultaneous Route Construction TDVRP Heuristic

In this heuristic, a new vehicle is introduced if doing so leads to the use of the shortest feasible link at any stage of the algorithm. All available vehicles may be used even if the total available capacity greatly exceeds the needed capacity.

Step 1. Start from the depot at the given starting time with the first available vehicle in the input file.

Step 2. Find a nonvisited node such that the link from the last-visited node of any used vehicle (or the depot if there are still unused vehicles) to this node has the smallest travel time and does not violate the temporal requirements and the capacity restrictions of the corresponding vehicle. Add this node to the tour of the corresponding vehicle.

Step 3. Repeat step 2 until all nodes are visited or no more capacity is available. If all nodes are visited, return to the depot for each used vehicle and stop. Otherwise, indicate insufficient capacity and stop.

This heuristic also may indicate infeasibility when the problem does have a feasible solution. Since all vehicles may be used even if there is no need to do so, the use of this heuristic should be avoided when the total capacity is much larger than the total demand. Using this heuristic, a solution can be obtained in $O(n^2KM)$ time in the worst case. Both TDVRP heuristics evaluate a solution on the basis of its total route time. However, other factors may also be important; e.g., fixed costs for the vehicles, the number of vehicles used, labor costs including overtime, and the route times of the individual tours, none of which are considered here.

Additional variations of these heuristics are possible. For example, the link to be included in a vehicle route can be selected randomly as in the probabilisitc TDTSP heuristic. Also, for a nonhomogeneous fleet, the next vehicle to be used can be selected randomly. Repeated execution of the probabilistic heuristic can give us a set of solutions from which to choose the preferred solution. Another variant would be to penalize links from the depot (i.e., the introduction of new vehicles) using a constant or multiplicative penalty term. A probabilistic heuristic for the TDVRP with time windows that includes waiting and considers $n - 1$ identical vehicles to be available can be found in MALANDRAKI.[48]

No polynomial-time heuristic can provide performance guarantees for the TSP when the triangle inequality does not hold unless there exists an exact polynomial-time algorithm for the problem (JOHNSON and PAPADIMITRIOU[40]). Since we do not assume that the triangle inequality holds for the TDVRP the above applies to the heuristics presented.

4. CUTTING PLANE HEURISTIC FOR THE TDTSP

THE TDTSP with time windows and a given starting time can be solved using dynamic programming (DP) by extending the DP algorithm for the TSP (BELLMAN[6] and Held and Karp[37]). The travel time functions may then be continuous or step functions if waiting is allowed before departing from a node. If waiting is not allowed, the DP algorithm cannot be applied for functions that may result in earlier arrival at a destination when departing later from the origin. Such functions include discontinuous decreasing functions or decreasing functions with a slope smaller than minus one. The DP approach for the TSP is of limited value due to the exponential size of its space and time requirements. It may compare well though at this point with other exact algorithms for the TDTSP both in the maximum size of the problems solved and in the computational times.

A cutting plane heuristic algorithm for the TDTSP with a given starting time from the depot and without time windows is briefly described in this section, based on the MILP formulation for the TDTSP presented in Section 1. The algoithm does not use all the constraints in the formulation but adds the temporal constraints only as needed. Cutting plane constraints, not included in the formulation, are also used. A preparatory phase precedes the application of the algorithm and excludes links that cannot participate in an optimal solution. Both the DP approach and the cutting plane algorithm are described in detail in Malandraki.[48]

Figure 4 presents a flowchart of the TDTSP algorithm which solves an LP relaxation of the TDTSP formulation that includes the assignment constraints but not the temporal constraints. This solution may produce subtours. Constraints (6) operate as SE constraints. They correspond to the Miller-Tucker-Zemlin[50] constraints for the TSP which are weaker (Wong[56]) than the usual SE constraints for the TSP (introduced by Dantzig,

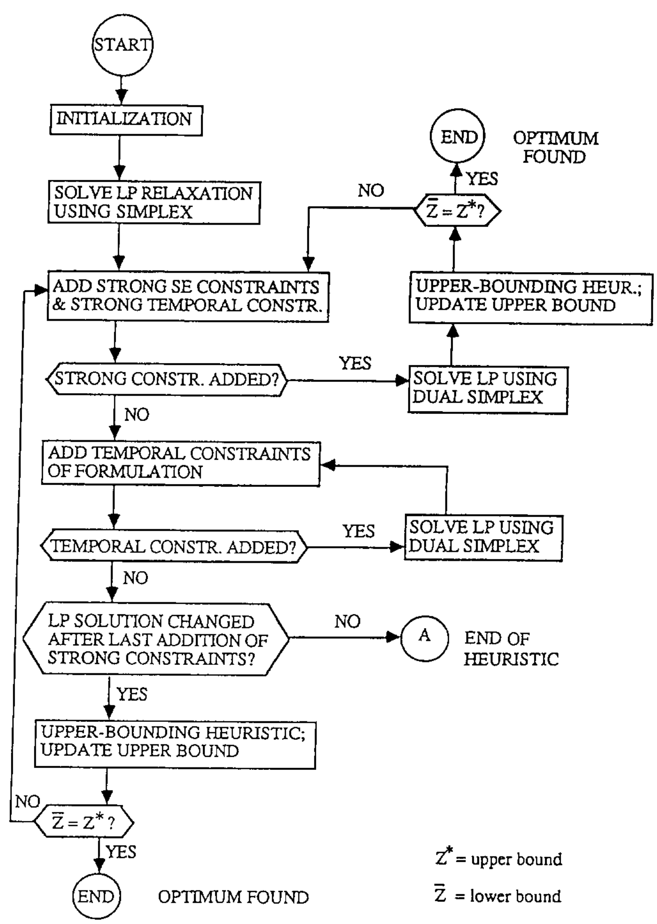


Fig. 4. Flowchart of phase 1 of TDTSP algorithm.

Fulkerson and Johnson[20]) that extend for the TDTSP as follows.

$$\sum_{i \in S} \sum_{j \in S} \sum_{m=1}^{M} x_{ij}^{m} \leq |S| - 1 \qquad (17)$$

where S is the node set of a subtour and $|S|$ is the cardinality of S.

Constraints (17) are much tighter than constraints (6). Only those constraints that are violated in the last LP solution are identified and added to the relaxation. The problem of identifying violated strong SE constraints for the TDTSP can be reduced (Malandraki[48]) to the problem of identifying violated SE constraints for the symmetric TSP. This can be acheived in $O(n^4)$ time (PADBERG and GRÖTSCHEL[51]) using an algorithm by GOMORY and HU.[33]

The temporal restrictions are imposed by constraints (6) to (8) which involve "big" numbers and generally result in weak LP relaxations. Strong temporal constraints may also be added. The strong temporal constaints used in the algorithm are of the form

$$x_{ij}^{m} + \sum_{\substack{k=2 \\ k \neq j}}^{n+1} \sum_{p \in A_{ij}^{m}} x_{jk}^{p} \leqslant 1$$

$$\text{for every link } (i, j) \text{ in period } m \qquad (18)$$

and

$$\sum_{\substack{i=1 \\ i \neq j}}^{n} \sum_{m \in B_{jk}^{p}} x_{ij}^{m} + x_{jk}^{p} \leqslant 1$$

$$\text{for every link } (j, k) \text{ in period } p \qquad (19)$$

where

$$A_{ij}^{m} = \Big\{\text{period } p \text{ for every link } (j, k) \,|\, \big(T_{jk}^{p} < T_{ij}^{m-1} + c_{ij}^{m} + c_{j}\big) \text{ or } \big(T_{jk}^{p-1} > T_{ij}^{m} + c_{ij}^{m} + c_{j} + \text{DIFF}\big)\Big\}$$

$$B_{jk}^{p} = \Big\{\text{period } m \text{ for every link } (i, j) \,|\, \big(T_{jk}^{p} < T_{ij}^{m-1} + c_{ij}^{m} + c_{j}\big) \text{ or } \big(T_{jk}^{p-1} > T_{ij}^{m} + c_{ij}^{m} + c_{j} + \text{DIFF}\big)\Big\}$$

and

$$\text{DIFF} = \max\big\{0, c_{jk}^{p-1} - c_{jk}^{p}\big\}.$$

The inclusion of the term DIFF allows the travel time step function on link (j, k) to behave as if it was a piecewise linear continuous function when the travel time in period p is less than that in the preceding period $p - 1$, as discussed in Section 2.1

To identify all the violated strong temporal constraints of the current LP solution, each link with a positive solution value is examined sequentially, first with respect to its succeeding links for constraint (18) and then with respect to its preceding links for constraint (19).

The algorithm adds strong SE and temporal constraints, re-solves and iterates until no more such constraints can be added. At that point, temporal constraints (6) to (8) of the formulation are added to the current LP but only for those links with positive flows in the current LP solution since constraints (6) to (8) for links whose solution values are zero are not binding and do not influence the solution. The algorithm iterates until no more constraints of any type can be added. The strong SE and strong temporal constraints are generally not sufficient to ensure an integer solution when the LP relaxation is solved. Additional valid and facet-defining inequalities known for the TSP should be extended to the TDTSP although few of them can be easily identified (GRÖTSCHEL and PADBERG[34]).

Two upper-bounding heuristics are used to convert the LP solution of the current relaxation to an integer solution. They start from the depot and construct a tour sequentially, using links having nonzero values or zero reduced costs in the last LP solution. The first heuristic uses a stack (HOROWITZ and SAHNI[38]) to store promising alternative routes in order to return and investigate them after examining the current tour. A parameter is also used to limit the number of alternatives that the algorithm examines. The second upper-bounding heuristic is a variation of the previous one in which the node to be visited next is chosen randomly from among the best candidates according to a given probability distribution.

The primal or dual simplex method (DANTZIG[18]) from the XMP mathematical programming library (MARSTEN[49]) was used to solve the LP problems. If at any time the upper bound equals the lower bound, the algorithm terminates with an optimal tour. Otherwise, it terminates at point A of the flowchart with an upper bound Z^* (best known tour) and a lower bound $\overline{Z} < Z^*$.

5. COMPUTATIONAL RESULTS

THIS SECTION presents the results of the application of the nearest-neighbor TDTSP and TDVRP heuristics and the cutting plane algorithm to randomly generated test problems. Unfortunately, there are no preceding research results to compare with the results of this paper. The algorithms were coded in FORTRAN and implemented on a VAX/VMS-11/785.

5.1. Generation of Random Test Problems

Table I summarizes the characteristics of the TDTSPs that were generated randomly to test the heuristics. Altogether 96 TDTSPs were generated and solved with: (1) 10, 15, 20 and 25 nodes; (2) step functions as travel time functions with two or three time intervals per link on average; and (3) four distributions of the number of time intervals across the links of the network (i.e., the number of parallel links between each pair of nodes). One deterministic and three probabilistic distributions were used with different variances.

For each generated problem, the number of nodes, the distribution of the number of time periods, and the seed for the generation of the random numbers were given. The travel time for the earliest time period of link (i, j), c_{ij}^1, was generated from a uniform [20, 80] distribution with the result rounded to the next integer. The travel times of subsequent time periods for link (i, j) were generated from a uniform $[c_{ij}^1 - 20, c_{ij}^1 + 20]$ distribution with the result rounded to the next integer. Since the travel time on a link depends on the link's distance, it is reasonable to generate correlated travel times for parallel links.

The duration of a time period was also generated randomly so that the upper bound of the last time period was not too large but also so that the length of any time period was not smaller than its travel time. The delivery time for each customer was generated from a uniform [10, 20] distribution with the results rounded to the next integer.

In addition, for the TDVRPs the number of vehicles was generated from a uniform $[n/6, n/2]$ distribution with the results rounded to the next integer. The number of vehicles and the vehicle capacities were generated so that the TDVRP test problems vary in the number of vehicles and the "tightness" of the capacity constraints. The vehicle capacities were generated so that the average vehicle capacity, AVCAP, was

$$\text{AVCAP} = \left(\sum_{i=1}^{n} d_i + d_{\max} \right) \Big/ K \qquad (20)$$

where $d_{\max} = \max_i d_i$. If AVCAP was less than $d_{\max}$ or if the total capacity divided by the total demand was less than 1.1 when AVCAP was computed using (20), equation (21) below was used instead

$$\text{AVCAP} = 2 \sum_{i=1}^{n} d_i / K \qquad (21)$$

The capacity of a vehicle was then generated from a uniform $[d_{\max}, 2\text{AVCAP} - d_{\max}]$ distribution. More details about the generation of the problems and additional test results can be found in Malandraki.[48]

TABLE I

Characteristics of TDTSP Test Problems

Factors	Levels
No. of nodes	10 15 20 25
Average No. of intervals per link	2 3
Distribution of No. of intervals per link	Det Pr1 Pr2 Pr3
No. of cases	32
Replications per case	3
Total No. of problems	96

5.2. Nearest-Neighbor TDTSP Heuristic Results

The NN1, NN2 and NNR heuristics were applied to the 96 test problems. Three versions (NNR1, NNR2, NNR3) of the probabilistic NNR heuristic were tested using different distributions for the probability with which a link is chosen for inclusion from among the best links. These distributions are shown in Table II.

The results from the five heuristics are shown in Table III. Each entry in the table is the average over the twelve problems with the same number of nodes and the same average number of time intervals per link. The average number of links for the test problems is shown in the third column. For each heuristic, the average ratio of the heuristic solution over the best known solution is shown. The best known solution may have been obtained by methods other than the nearest-neighbor heuristics.

All the nearest-neighbor TDTSP heuristics perform similarly except for the NN1 heuristic which exhibits the worst results. The three versions of the probabilistic heuristic perform similarly, with versions NNR2 and NNR3 slightly better on the average.

The execution times of the five heuristics are shown in Table IV. The CPU time does not include the time needed to read the input file of a problem. As expected, the NN1 heuristic needs the smallest amount of CPU time and the NNR heuristics require the most CPU time. Note that all average times are less than one second. Overall, comparing both the solution results from Table III and the CPU times from Table IV, the NN2 heuristic seems to give the best results. The probabilistic heuristics

TABLE II

Probabilities of Links Being Chosen in Probabilistic Nearest-Neighbor TDTSP Heuristic

Version	Best Link	2nd Best	3rd Best
NNR1	0.75	0.15	0.10
NNR2	0.80	0.15	0.05
NNR3	0.85	0.11	0.04

though may find a better solution if they are applied for a longer time.

5.3. Nearest-Neighbor TDVRP Heuristic Results

The nearest-neighbor heuristics for the TDVRP without time windows were applied to 32 randomly generated test problems. The problems are the same as those generated for the TDTSP in terms of their common characteristics (network configuration, travel times, time periods, etc.) but in addition they have the number of available vehicles, demands and capacities generated as described in Section 5.1. Four distributions of the number of time intervals per link were again employed but only one replication per case was used.

Two versions (SEQ-SL and SEQ-LS) of the sequential nearest-neighbor TDVRP heuristic were tested as well as the simultaneous TDVRP heuristic (SIM). The SEQ-SL and SIM heuristics fill the vehicles from smallest to largest and the SEQ-LS from largest to smallest. The heuristics were applied as follows. First, all three heuristics were applied for the original number of available vehicles. If the solution of the SEQ-SL and SEQ-LS heuristics used less than the available number of vehicles, the SIM heuristic was applied again using those vehicles used in the best solution found so far. The best solution found by the two applications of the simultaneous heuristic is reported.

Table V shows the results of the TDVRP heuristics. Each entry is the average of the four problems with the same number of nodes and average number of time periods per link. There were a few cases for which the heuristics failed to obtain a feasible solution (although a feasible solution was found by inspection). These cases are not considered in the computation of the averages. The third column of Table V shows the average number of links of the problems. The fourth column shows the average ratio of total demand over total capacity of the available vehicles. The remaining columns show the average ratio of the heuristic solution over the best solution and the average number of vehicles used. The SEQ-LS heuristic finds consistently better solutions than the SEQ-SL. Fewer vehicles of larger capacities are used in the SEQ-LS solutions with smaller total travel times. The order in which the vehicles are filled seems to influence the solution considerably. The results of the SIM heuristic are comparable to those of SEQ-LS but not better for out sample of problems.

Table VI summarizes the execution times for the heuristics. The CPU times are extremely small for problems with up to 25 nodes. On the average, the SEQ-LS heuristic appears to perform best considering both the average CPU times and the ratios of Table V.

5.4. Results of the Cutting Plane Algorithm

At the beginning of the cutting plane algorithm an optimal solution is bounded from above by the

TABLE III

Results of Nearest-Neighbor TDTSP Heuristics

Intervals per Link	n	No of Links	Heuristic Solution/Best Known Solution				
			NN1	NN2	NNR1	NNR2	NNR3
2	10	181.33[a]	1.159	1.104	**1.063**[b]	1.080	1.076
	15	424.50	1.152	**1.070**	1.098	1.072	1.082
	20	764.00	1.116	**1.070**	1.096	1.084	1.072
	25	1208.00	1.114	1.050	1.091	**1.045**	1.053
Avg		644.46	1.135	1.073	1.087	**1.070**	1.071
3	10	270.25	1.136	**1.050**	1.076	1.085	1.073
	15	634.50	1.164	1.098	1.099	1.103	**1.093**
	20	1150.25	1.161	1.070	1.097	1.062	**1.058**
	25	1806.08	1.090	**1.043**	1.083	1.045	1.046
Avg		965.27	1.138	**1.065**	1.089	1.074	1.068
Overall Avg		804.86	1.137	**1.069**	1.088	1.072	**1.069**

[a] Values shown are averages over 12 problems (i.e., four distributions of number of intervals per link and three replications per distribution).

[b] Values in bold indicate the best results in each row.

TABLE IV

CPU (VAX-11 / 785) Times (Sec) for the TDTSP Heuristics

Intervals per Link	n	NN1	NN2	NNR1	NNR2	NNR3
2	10	0.03[a]	0.05	0.10	0.09	0.09
	15	0.05	0.10	0.22	0.23	0.23
	20	0.06	0.19	0.47	0.48	0.47
	25	0.08	0.36	0.89	0.89	0.91
Avg		0.06	0.18	0.42	0.42	0.42
3	10	0.03	0.05	0.09	0.10	0.11
	15	0.04	0.11	0.24	0.24	0.25
	20	0.06	0.23	0.49	0.49	0.51
	25	0.08	0.41	0.91	0.90	0.91
Avg		0.05	0.20	0.43	0.43	0.44
Overall Avg		0.05	0.19	0.43	0.43	0.43

[a] Values shown are averages over 12 problems (i.e., four distributions of number of intervals per link and three replications per distribution).

original upper bound (OUB) (best known solution of the nearest-neighbor heuristics) and from below by the original lower bound (OLB) (solution of the original LP relaxation). After the application of the cutting plane algorithm, a new final upper bound (FUB) on an optimal solution may be obtained (by the upper-bounding heuristics) and a higher final lower bound (FLB) (from the addition of SE and temporal constraints). If the FUB and FLB are equal, an optimal solution is obtained. Otherwise, the effectiveness of the algorithm is indicated by the "gap" between these bounds. The lower bound generally increases monotonically at every iteration but the upper bound does not necessarily decrease at every iteration.

Table VII shows the average number of links after the preparatory phase (the number before the preparatory phase is shown in Table III). Each entry is the average value over the twelve problems with the same number of nodes and the same average number of intervals per link. The table also shows the average and maximum values of the gap which is a percent measure computed as GAP = 100*(FUB-FLB)/FLB. The average gap tends to increase with the number of nodes and the number of time periods. It is generally quite large, about 20% on the average for the largest problems. A large gap may be the result of omitted cutting plane constraints and/or the inability of the upper-bounding heuristics to find good solutions.

Ratio1 and Ratio2 are additional effectiveness measures computed as follows: Ratio1 = 100*(FLB-OLB)/(OUB-OLB); Ratio2 = 100*(OUB-FUB)/(OUB-OLB); and Ratio = Ratio1 + Ratio2. Ratio1 and Ratio2 measure the percentage of the original difference between the upper and lower bounds that is eliminated as a result of the increase in the lower bound and the decrease in the upper bound correspondingly. Ratio represents the combined result. The three Ratio measures depend on the OUB. Ratio1 tends to decrease with the number of nodes and to increase with the number of time intervals. Ratio2 is inconsistent, probably reflecting differences in the accuracy of the OUB. On average, more than 35% of the difference between the upper and the lower bounds is eliminated.

The last column of Table VII shows that the upper-bounding heuristics found an improved or at least as good a solution as the OUB in 2/3 of the problems. This measure exhibits no trend in terms

TABLE V

Results of Nearest-Neighbor TDVRP Heuristics

Intervals per Link	n	No of Links	Total Demand over Total Capacity	SEQ-SL Heur sol/ Best sol	SEQ-SL No of vehicles	SEQ-LS Heur sol/ Best sol	SEQ-LS No. of vehicles	SIM Heur sol/ Best sol	SIM No. of vehicles
2	10	182.75[a]	0.62	1.148[b]	2.67	1.000[b]	2.00	1.123[b]	2.67
	15	427.25	0.57	1.130	4.00	1.001	3.00	1.037	3.25
	20	763.25	0.71	1.182[b]	4.67	1.062[b]	2.67	1.000[b]	2.67
	25	1201.25	0.73	1.126	6.00	1.029	5.00	1.016	5.50
Avg		643.63	0.66	1.144	4.43	1.022	3.29	1.042	3.64
3	10	272.75	0.76	1.077	3.00	1.005	2.75	1.015[b]	2.00
	15	630.50	0.73	1.040	4.75	1.005	4.00	1.012	4.25
	20	1153.25	0.64	1.193	5.00	1.074	3.50	1.041	3.50
	25	1823.25	0.54	1.202	7.50	1.045	4.75	1.003	4.75
Avg		969.94	0.66	1.128	5.06	1.032	3.75	1.018	3.73
Overall		806.78	0.66	1.135	4.77	1.027	3.53	1.029	3.69

[a] Values shown are averages over four problems (four distributions of number of intervals per link).
[b] The heuristic failed to obtain a feasible solution in one problem out of four.

TABLE VI

CPU (VAX-11 / 785) Times (Sec) of TDVRP Heuristics

Intervals per Link	n	No of Links	SEQ-SL	SEQ-LS	SIM
2	10	182.75[a]	0.05	0.05	0.06
	15	427.25	0.08	0.08	0.08
	20	763.25	0.11	0.10	0.10
	25	1201.25	0.12	0.13	0.18
Avg		643.63	0.09	0.09	0.11
3	10	272.75	0.06	0.06	0.06
	15	630.50	0.09	0.09	0.12
	20	1153.25	0.13	0.09	0.12
	25	1823.25	0.13	0.11	0.17
Avg		969.94	0.10	0.09	0.11
Overall Avg		806.78	0.10	0.09	0.11

[a] Values shown are averages over four problems (four distributions of number of intervals per link).

of the number of nodes, although results for problems with two time intervals per link seem to be better than those with three intervals per link.

The average and the maximum CPU times are shown in Table VIII both with and without the use of the probabilistic upper-bounding heuristic which has very high CPU times. The time needed for the preparatory phase, to read the problem inputs, and to set up the data structures of the LP is not included. The average CPU times increase with the number of links, as expected , since the number of links corresponds to the number of binary variables of the LP. The CPU time increases quickly with the number of time periods. The maximum CPU times seem to be influenced more than the average times by the number of nodes.

6. CONCLUSIONS

THIS PAPER defines, formulates, and develops heuristic algorithms for both the time dependent traveling salesman problem (TDTSP) and the time dependent vehicle routing problem (TDVRP). Properties of the problems are examined indicating that most of the existing heuristics for the nontemporal problems cannot be easily extended to the time dependent problems. Nearest-neighbor heuristics for the TDTSP and the TDVRP without time windows are presented as well as a cutting plane heuristic algorithm based on a mixed integer linear programming formulation. The algorithms are tested on randomly generated problems with 10 to 25 nodes and with travel times represented by step functions of two or three time periods per link on average.

The nearest-neighbor heuristics require very low computation times but the cutting plane algorithm is much more expensive computationally. The cutting plane algorithm exhibits large gaps between the value of the best feasible solution obtained and the solution of the final LP relaxation. The gaps increase with the number of nodes and the average number of time periods per link. The cutting plane algorithm solves only small problems but finds a solution better than or at least as good as the solution obtained by the nearest-neighbor heuristics in 2/3 of the problems tested.

The time dependent problems represent an urban, congested environment more accurately than do their nontemporal counterparts but they are more difficult to solve. More research needs to be devoted to the development of algorithms for the time dependent problems. The algorithm need also

TABLE VII

Effectiveness Indicators of TDTSP Cutting Plane Algorithm

Intervals per Link	n	Links after Preparatory Phase	Percent Gap Avg	Percent Gap Max	Ratio 1 %	Ratio 2 %	Ratio %	Percent Runs with Cutting Plane Heuristic at Least as Good as NN
2	10	169.25[a]	9.19	20.05	20.53	18.82	39.35	58.33
	15	408.25	11.48	16.79	12.93	22.32	35.25	66.67
	20	744.17	11.15	16.04	13.82	26.98	40.80	83.33
	25	1181.83	13.27	17.30	12.66	15.43	28.09	66.67
Partial		625.88	11.27	20.05	14.98	20.89	35.87	68.75
3	10	246.00	14.83	20.22	22.40	13.38	35.78	50.00
	15	603.17	16.49	25.85	15.75	25.84	41.59	66.67
	20	1110.25	18.86	25.55	17.60	14.77	32.37	75.00
	25	1758.92	20.65	31.81	14.21	13.27	27.48	66.67
Partial		929.59	17.70	31.81	17.49	16.81	34.30	64.58
Overall		777.73	14.49	31.81	16.24	18.85	35.09	66.67

[a] Values shown are computed over 12 problems (i.e., four distributions of number of intervals per link and three replications per distribution).

TABLE VIII
CPU Times of TDTSP Cutting Plane Algorithm

Intervals per Link	n	CPU (VAX) Time (Sec)			
		Total Time		Without Probabilistic Heuristic	
		Avg	Max	Avg	Max
2	10	24.22*	37.67	22.43	36.15
	15	58.62	68.14	52.32	61.47
	20	117.36	153.23	101.52	135.76
	25	223.67	355.31	195.48	328.29
Partial		105.97	355.31	92.94	328.29
3	10	39.39	58.00	36.90	55.29
	15	97.70	118.19	88.22	109.19
	20	207.19	286.29	184.96	262.40
	25	431.38	537.98	385.07	494.25
Partial		193.91	537.98	173.79	494.25
Overall value		149.94	537.98	133.36	494.25

[a] Values shown are computed over 12 problems (i.e., four distributions of number of intervals per link and three replications per distribution).

to be tested on real data collected from congested networks.

ACKNOWLEDGMENTS

THIS RESEARCH was performed at Northwestern University and was supported, in part, by a National Science Foundation Presidential Young Investigator Award No. 8352298, a matching grant from the General Motors Corporation and a fellowship from the Transportation Center, Northwestern University. This support is gratefully acknowledged. The authors thank Professor Robert Fourer of Northwestern University for providing the XMP linear programming code. The authors also thank three anonymous referees for their helpful comments on an earlier draft of the paper.

REFERENCES

1. A. V. AHO, J. E. HOPCROFT AND J. D. ULLMAN, *Data Structures and Algorithms*, Addison-Wesley, Reading, Mass., 1983.
2. E. K. BAKER, "An Exact Algorithm for the Time-Constrained Traveling Salesman Problem," *Opns. Res.* **31**, 938–945 (1983).
3. E. K. BAKER AND J. R. SCHAFFER, "Solution Improvement Heuristics for the Vehicle Routing and Scheduling Problem with Time Window Constraints," Working Paper Department of Management Science and Computer Information Systems, University of Miami, 1984.
4. E. BALAS AND N. CHRISTOFIDES, "A Restricted Lagrangean Approach to the Traveling Salesman Problem," *Math. Program.* **21**, 19–46 (1981).
5. W. J. BELL, L. M. DALBERTO, M. L. FISHER, A. J. GREENFIELD, R. JAIKUMAR, P. KEDIA, R. G. MACK AND P. J. PRUTZMAN, "Improving the Distribution of Industrial Gases with an On-Line Computerized Routing and Scheduling Optimizer," *Interfaces* **13**, 4–23 (1983).
6. R. BELLMAN, "Dynamic Programming Treatment of the Traveling Salesman Problem," *J. Assoc. Comput. Mach.* **9**, 61–63 (1962).
7. L. BODIN, B. GOLDEN, A. ASSAD AND M. BALL, "The State of the Art in the Routing and Scheduling of Vehicles and Crews," Office of Policy Research, Urban Mass Transportation Administration, U.S. Department of Transportation, Report UMTA/BMGT/MSS#81-001, Washington, D.C., 1981.
8. G. CARPANETO AND P. TOTH, "Some New Branching and Bounding Criteria for the Asymmetric Traveling Salesman Problem," *Mgmt. Sci.* **26**, 736–743 (1980).
9. N. CHRISTOFIDES, "Vehicle Routing," in *Combinatorial Optimization: Annotated Bibliographies*, pp. 148–163, M. O'hEigeartaigh, J. K. Lenstra and A. H. G. Rinnooy Kan (eds.), Centre for Mathematics and Computer Science, Amsterdam, 1985.
10. N. CHRISTOFIDES, "Vehicle Routing," in *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, Ch. 12, pp. 431–448, E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan and D. B. Shmoys (eds.), John Wiley & Sons, New York, 1985.
11. N. CHRISTOFIDES, "The Travelling Salesman Problem," in *Combinatorial Optimization*, Ch. 6, pp. 131–149, N. Christofides, A. Mingozzi, P. Toth and C. Sandi (eds.), John Wiley & Sons, New York, 1979.
12. N. CHRISTOFIDES, "The Vehicle Routing Problem," *R.A.I.R.O. Rech. Opnl.* **10**, 55–70 (1976).
13. N. CHRISTOFIDES, A. MINGOZZI AND P. TOTH, "State-Space Relaxation Procedures for the Computation of Bounds to Routing Problems," *Networks* **11**, 145–164 (1981).
14. N. CHRISTOFIDES, A. MINGOZZI AND P. TOTH, "Exact Algorithms for the Vehicle Routing Problem, Based

on Spanning Tree and Shortest Path Relaxations," *Math. Program.* **20**, 255–282 (1981).

15. N. CHRISTOFIDES, A. MINGOZZI AND P. TOTH, "The Vehicle Routing Problem," in *Combinatorial Optimization*, Ch. 11, pp. 315–338, N. Christofides, A. Mingozzi, P. Toth and C. Sandi (eds.), John Wiley & Sons, New York, 1979.
16. G. CLARKE AND W. WRIGHT, "Scheduling of Vehicles from a Central Depot to a Number of Delivery Points," *Opns. Res.* **12**, 568–582 (1964).
17. H. CROWDER AND M. PADBERG, "Solving Large-Scale Symmetric Traveling Salesman Problems to Optimality," *Mgmt. Sci.* **26**, 495–509 (1980).
18. G. B. DANTZIG, *Linear Programming and Extensions*, Princeton University Press, Princeton, N.J., 1963.
19. G. B. DANTZIG, D. R. FULKERSON AND S. M. JOHNSON, "On a Linear Programming Combinatorial Approach to the Traveling-Salesman Problem," *Opns. Res.* **7**, 58–66 (1959).
20. G. B. DANTZIG, D. R. FULKERSON AND S. M. JOHNSON, "Solution of a Large-Scale Traveling-Salesman Problem," *Opns. Res.* **2**, 393–410 (1954).
21. J. DESROSIERS, F. SOUMIS AND M. DESROCHERS, "Routing with Time Windows by Column Generation," *Networks* **14**, 545–565 (1984).
22. T. A. FEO AND J. F. BARD, "A Network Approach to Flight Scheduling and Maintenance Base Planning," Department of Mechanical Engineering Working Paper, University of Texas, Austin, 1987.
23. M. L. FISHER, A. J. GREENFIELD, R. JAIKUMAR AND J. T. LESTER, "A Computerized Vehicle Routing Application," *Interfaces* **12**, 42–52 (1982).
24. M. L. FISHER AND R. JAIKUMAR, "A Generalized Assignment Heuristic for Vehicle Routing," *Networks* **11**, 109–124 (1981).
25. K. R. FOX, B. GAVISH AND S. C. GRAVES, "An n-Constraint Formulation of the (Time-Dependent) Traveling Salesman Problem," *Opns. Res.* **28**, 1018–1021 (1980).
26. M. R. GAREY AND D. S. JOHNSON, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman, New York, 1979.
27. B. GAVISH AND S. C. GRAVES, "Scheduling and Routing in Transportation and Distribution Systems: Formulations and New Relaxations," Working Paper 8202 Graduate School of Management, University of Rochester, 1981.
28. B. GAVISH AND S. C. GRAVES, "The Traveling Salesman Problem and Related Problems," Working Paper 7906 Graduate School of Management, University of Rochester, 1979.
29. B. GAVISH AND K. SRIKANTH, "An Optimal Solution Method for Large-Scale Multiple Traveling Salesmen Problems," *Opns. Res.* **34**, 698–717 (1986).
30. B. GILLET AND L. MILLER, "A Heuristic Algorithm for the Vehicle Dispatching Problem," *Opns. Res.* **22**, 340–349 (1974).
31. B. L. GOLDEN AND A. A. ASSAD, "Perspectives on Vehicle Routing: Exciting New Developments," *Opns. Res.* **34**, 803–810 (1986).
32. B. L. GOLDEN, T. L. MAGNANTI AND H. NGUYEN, "Implementing Vehicle Routing Algorithms," *Networks* **7**, 113–148 (1977).
33. R. E. GOMORY AND T. C. HU, "Multi-Terminal Network Flows," *SIAM J. Appl. Math.* **9**, 551–570 (1961).
34. M. GRÖTSCHEL AND M. W. PADBERG, "Polyhedral Theory," in *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, Ch. 8, pp. 251–305, E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan and D. B. Shmoys (eds.), John Wiley & Sons, New York, 1985.
35. M. HELD AND R. M. KARP, "The Traveling Salesman Problem and Minimum Spanning Trees—Part II," *Math. Program.* **1**, 6–25 (1971).
36. M. HELD AND R. M. KARP, "The Traveling Salesman Problem and Minimum Spanning Trees," *Opns. Res.* **18**, 1138–1162 (1970).
37. M. HELD AND R. M. KARP, "A Dynamic Programming Approach to Sequencing Problems," *J. SIAM* **10**, 196–210 (1962).
38. E. HOROWITZ AND S. SAHNI, *Fundamentals of Computer Algorithms*, Computer Science Press, Rockville, Md., 1978.
39. D. S. JOHNSON AND C. H. PAPADIMITRIOU, "Computational Complexity," in *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, Ch. 3, pp. 37–85, E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan and D. B. Shmoys (eds.), John Wiley & Sons, New York, 1985.
40. D. S. JOHNSON AND C. H. PAPADIMITRIOU, "Performance Guarantees for Heuristics," in *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, Ch. 5, pp. 145–180, E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan and D. B. Shmoys (eds.), John Wiley & Sons, New York, 1985.
41. A. W. J. KOLEN, A. H. G. RINNOOY KAN AND H. W. J. M. TRIENEKENS, "Vehicle Routing with Time Windows," *Opns. Res.* **35**, 266–273 (1987).
42. G. LAPORTE, Y. NOBERT AND M. DESROCHERS, "Optimal Routing under Capacity and Distance Restrictions," *Opns. Res.* **33**, 1050–1073 (1985).
43. R. C. LARSON AND A. R. ODONI, *Urban Operations Research*, Prentice-Hall, Englewood Cliffs, N.J. 1981.
44. E. L. LAWLER, J. K. LENSTRA, A. H. G. RINNOOY KAN AND D. B. SHMOYS (eds.), *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, John Wiley & Sons, New York, 1985.
45. J. K. LENSTRA AND A. H. G. RINNOOY KAN, "Complexity of Vehicle Routing and Scheduling Problems," *Networks* **11**, 221–227 (1981).
46. S. LIN AND B. KERNIGHAN, "An Effective Heuristic Algorithm for the Traveling Salesman Problem," *Opns. Res.* **21**, 498–516 (1973).
47. T. L. MAGNANTI, "Combinatorial Optimization and Vehicle Fleet Planning: Perspectives and Prospects," *Networks* **11**, 179–213 (1981).
48. C. MALANDRAKI, "Time Dependent Vehicle Routing Problems: Formulations, Solution Algorithms and Computations Experiments," Ph.D. dissertation, Northwestern University, Evanston, Ill., 1989.

49. R. E. MARSTEN, "The Design of the XMP Linear Programming Library," *ACM Transact. Math. Software* **7**, 481–497 (1981).
50. C. E. MILLER, A. W. TUCKER AND R. A. ZEMLIN, "Integer Programming Formulation of Traveling Salesman Problems," *J. Assoc. Comput. Mach.* **7**, 326–329 (1960).
51. M. W. PADBERG AND M. GRÖTSCHEL, "Polyhedral Computations," in *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, Ch. 9, pp. 307–360, E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan and D. B. Shmoys (eds.), John Wiley & Sons, New York, 1985.
52. J. C. PICARD AND M. QUEYRANNE, "The Time-Dependent Traveling Salesman Problem and its Applications to the Tardiness Problem in One-Machine Scheduling," *Opns. Res.* **26**, 86–110 (1978).
53. M. M. SOLOMON, "Algorithms for the Vehicle Routing and Scheduling Problem with Time Window Constraints," *Opns. Res.* **35**, 254–265 (1987).
54. M. M. SOLOMON, "On the Worst-Case Performance of Some Heuristics for the Vehicle Routing and Scheduling Problem with Time Window Constraints," *Networks* **16**, 161–174 (1986).
55. M. M. SOLOMON AND J. DESROSIERS, "Time Window Constrained Routing and Scheduling Problems," *Trans. Sci.* **22**, 1–13 (1988).
56. R. T. WONG, "Integer Programming Formulations of the Traveling Salesman Problem," *Proceedings of the IEEE International Conference on Circuits and Computers*, pp. 149–152, 1980.

(Received, July 1989; revisions received October 1990, March 1991; accepted April 1991 by Amedeo Odoni)