MANAGEMENT SCIENCE
Vol. 36, No. 2, February 1990
Printed in U.S.A.

ROUTING AND SCHEDULING ON A SHORELINE WITH RELEASE TIMES*

HARILAOS N. PSARAFTIS, MARIUS M. SOLOMON, THOMAS L. MAGNANTI AND TAI-UP KIM

Department of Ocean Engineering, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139
Management Science Department, Northeastern University, Boston, Massachusetts 02115
Sloan School of Management, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139
Kion Technology Inc., Seoul, Republic of Korea

In this paper we examine computational complexity issues and develop algorithms for a class of "shoreline" single-vehicle routing and scheduling problems with release time constraints. Problems in this class are interesting for both practical and theoretical reasons. From a practical perspective, these problems arise in several transportation environments. For instance, in the routing and scheduling of cargo ships, the routing structure is "easy" because the ports to be visited are usually located along a shoreline. However, because release times of cargoes at ports generally complicate the routing structure, the combined routing and scheduling problem is nontrivial. For the straight-line case (a restriction of the shoreline case), our analysis shows that the problem of minimizing the maximum completion time can be solved exactly in quadratic time by dynamic programming. For the shoreline case we develop and analyze heuristic algorithms. We derive data-dependent worst-case performance ratios for these heuristics that are bounded by constant. We also discuss how these algorithms perform on practical data.
(ROUTING PROBLEMS; DYNAMIC PROGRAMMING; ANALYSIS OF HEURISTICS)

1. Introduction

In this paper we examine a class of time-constrained vehicle routing and scheduling problems that may be encountered in several transportation/distribution environments. Our basic variant assumes one vehicle (initially located at a prescribed point) that must pick up a number of cargoes, which are located at some other known points. We assume that each cargo is available for pickup at or after a given earliest pickup time (or, as we shall refer to from now on, at or after a prescribed "release time"). Given the vehicle is allowed to wait at any point if doing so proves desirable or necessary, we wish to find the schedule that will permit us to pick up all cargoes as soon as possible (that is, the schedule that minimizes the maximum completion time).

As defined so far, this problem is a generalization of the classical Traveling Salesman Problem (TSP). As such, it is *NP*-complete, even if the interpoint distance metric is restricted to be Euclidean (Papadimitriou 1977). In this paper we further assume a special network topology which we refer to as the shoreline topology. Throughout most of the paper we also permit a somewhat broader interpoint distance metric, a triangle-inequality metric, rather than a Euclidean metric (though in places, we assume that distances are Euclidean). We call this problem SLP (SL for shoreline). The shoreline network is defined as follows:

An ordered set of points $i = 1, \ldots, n$ is located on the shoreline if the interpoint distance matrix $[t_{ij}]$ of these points satisfies the following conditions:

For all $1 \le i \le k \le j \le n$,

(i) $t_{ii} = 0$,

(ii) $t_{ij} = t_{ji}$,

* Accepted by Alexander H. G. Rinnooy Kan; received October 1986. This paper has been with the authors 17 months for 2 revisions.

0025-1909/90/3602/0212$01.25

(iii) $t_{ij} \geq t_{ik}$,
(iv) $t_{ij} \geq t_{kj}$,
(v) $t_{ij} \leq t_{ik} + t_{kj}$.

Figure 1.1 shows typical forms of shoreline instances. Note that (a) due to conditions (iii) and (iv) a shoreline network imposes a topological restriction on the triangle-inequality metric, (b) a path along the shoreline need not be convex (i.e., lie on the boundary of a convex region), (c) given a nonordered set of n points and their corresponding distance matrix, it is possible to check in $O(n^2)$ time whether these points are located on some shoreline, and (d) a further restriction of the shoreline problem is the straight-line case that replaces condition (v) by $t_{ij} = t_{ik} + t_{kj}$.

We have adopted the name "shoreline" because problems with this metric arise in the routing and scheduling of ships. In these applications, the ports to be visited by a ship are often located on a (real-world) shoreline (many real-world shorelines obey the definition). In the presence of shorelines, the underlying routing structure is "easy", in the sense that in the absence of time constraints the optimal ship schedule is obvious.

However, planners may also have to deal with earliest and/or latest pickup times at each port. These times are often independently determined from cargo availability requirements at the land side, and, as such, usually possess little or no "regularity" (for instance, they need not be monotonically increasing along the shoreline). This lack of regularity generally destroys the "easy" routing structure and makes the resulting routing and scheduling problem nontrivial. Similar situations may be encountered in other routing and scheduling environments.

The literature on vehicle routing and scheduling, in general, and problems with time windows, in particular, has been growing at an explosive rate over the last few years (see the comprehensive survey by Bodin et al. 1983). Algorithms for different variants of the routing problem in the presence of time windows have been developed and analyzed by Baker (1983), Desrosiers et al. (1984), (1986), Jaw et al. (1986), Psaraftis (1983), (1986), Sexton and Bodin (1985a, b), Solomon (1986a, b), (1987), and Solomon et al. (1988), to name just a few. A recent survey is provided by Solomon and Desrosiers (1988). The literature on applications for the routing and scheduling of ships is considerably less rich, but also growing: see for instance, Psaraftis (1985), Fisher and Rosenwein (1985) and Ronen (1983).

The scope of this paper is to introduce an important routing structure, the shoreline structure, examine the complexity of different SLP variants, and design and analyze exact and approximate algorithms that exploit this special structure.

Without loss of generality, we assume a unit vehicle speed, so that all interpoint direct travel times, t_{ij} for $i, j = 1, \ldots, n$, equal the corresponding distances. Also without loss

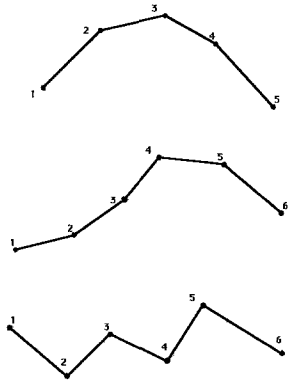

FIGURE 1.1. Typical Shoreline Instances.

of generality, as in our previous definition, we number the points according to the order that they are encountered when a vehicle traverses the shoreline from one endpoint (point 1) to the other (point n). Finally, we assume that the vehicle is initially located (at time $t = 0$) at point 1. (We shall relax this assumption in §2.) A vehicle schedule can be represented by an array $(C_1, C_2, \ldots, C_n)$ with C_i defined as the completion time (or visit time) of point i. A point's completion time is the time that the vehicle *departs* from that point, which is not necessarily equal to the time the vehicle may arrive at (or pass by) that point. *Release times* r_i impose the restriction that $C_i \geq r_i$ for all i in a feasible schedule; note that the vehicle is allowed to *wait* at a point if doing so proves desirable or necessary.

Our objective for this problem is to minimize the maximum completion time, $C_{\max} = \max_i C_i$; that is, complete all visits as soon as possible. Although our basic scenario assumes that the vehicle does not return to point 1 at the end of its schedule (we call this basic version the "path" case (pSLP)), we shall also selectively examine the "tour" case, (tSLP) in which the vehicle returns to point 1 at the end of the trip (and must arrive there as soon as possible). In all cases, we denote the optimal value of the problem, that is, the minimum value of $C_{\max}$ over all schedules, as $C^*_{\max}$.

We focus on developing algorithms that take advantage of the shoreline structure, in general, and the straight line structure, in particular. The paper is organized as follows. In §2, we show that the straight-line variant of the SLP is polynomial by developing an $O(n)$ exact algorithm for the tSLP and an exact $O(n^2)$ dynamic programming algorithm for the pSLP, respectively. In §3, we develop several heuristics for the general shoreline SLP. These heuristics are based on the exact algorithms of §2, which are not necessarily optimal for the general shoreline case. We derive data-dependent worst-case performance bounds for these heuristics and show that these are bounded by a constant. We also discuss how these algorithms perform in practice. Finally, §4 presents our conclusions.

2. The Straight-Line Case

We begin our investigation by analyzing the straight-line case with release times. In this case, all points $1, 2, \ldots, n$ to be visited are on a linear segment of length L and no point i can be visited before a prespecified "release time" r_i. Our basic underlying scenario assumes that for all i and j, the direct travel time t_{ij} from point i to point j, is equal to the corresponding interpoint distance $|x_i - x_j|$ defined by the x-coordinates x_i and x_j of points i and j along the line (therefore $L = t_{1n}$). Note that the straight-line case does not necessarily require a geometric linear segment, but any metric that can be transformed to a linear segment, that is, one in which condition $t_{ij} = t_{ik} + t_{kj}$ for all $1 \leq i \leq k \leq j \leq n$ is satisfied. As such, this case can occur in a wide range of applications (railroads, rivers, highways, or even ship problems where ports are located along a convex hull with land at the interior of the hull). We begin by assuming that the initial location of the vehicle at $t = 0$ is at point 1 (later in this section we see what happens when we relax this assumption).

It is clear that in the absence of release times, or even in the case in which all release times are "agreeable" (that is $r_i \leq r_j$ for all $i < j$), the solution of this problem is trivial: a straight "traversal" from point 1 to point n is optimal. However, if the release times are not agreeable, it becomes less clear what the optimal schedule should be. Figure 2.1 shows a typical schematic representation of a schedule for $n = 5$ (in the figure the distance axis is horizontal and the time axis is vertical with time increasing in the downward direction). As illustrated in this figure, if the release times are not agreeable, the optimal schedule can be anything but trivial (it may, for instance, involve several direction *reversals*, such as those that occur at points 4 and 6 in the example of Figure 2.1).

Before we solve the basic ("path") version of this problem in which we do not require

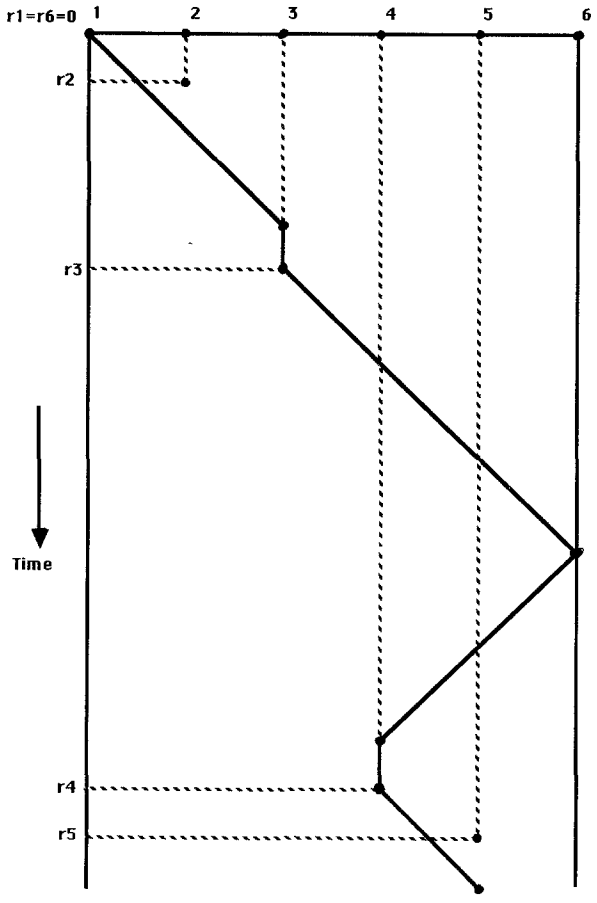

FIGURE 2.1. Straight-Line Case: A Typical Schedule.

the vehicle to return to point 1 at the end of the schedule, let us briefly examine the equivalent "tour" version. If the vehicle is required to return to point 1, the problem can be solved very easily by the following simple algorithm:

"*TRAVERSE*" *Algorithm* (Solves Tour Version)

Step 1. Without waiting at any intermediate point, go straight from point 1 to point n (that is, arrive at point n at time L).

Step 2. Wait at point n for an amount of time equal to $BW_{\max} = \max_i [\max (0, r_i - L - t_{in})]$.

Step 3. Depart from point n at time $L + BW_{\max}$ and return to point 1 through point $n-1, n-2, \ldots, 2$ without waiting at any intermediate point (that is, arrive at point 1 at time $2L + BW_{\max}$).

THEOREM 2.1. *"TRAVERSE" solves exactly the tour version of the straight-line case in $O(n)$ time.*

PROOF. First, the route and schedule produced by the "TRAVERSE" algorithm is *feasible*. Indeed, the quantity $BW_i = \max (0, r_i - L - t_{in})$ (see Figure 2.2) is the amount of time the vehicle would have to wait at point i on its way back from point n to point 1, if the only enforced release time were r_i. By waiting at point n for $BW_{\max} = \max_i BW_i$, the vehicle schedule obeys the condition $C_i \geq r_i$ for all i and is therefore feasible.

To see that this schedule is *optimal*, we note that since it is feasible, its total duration,

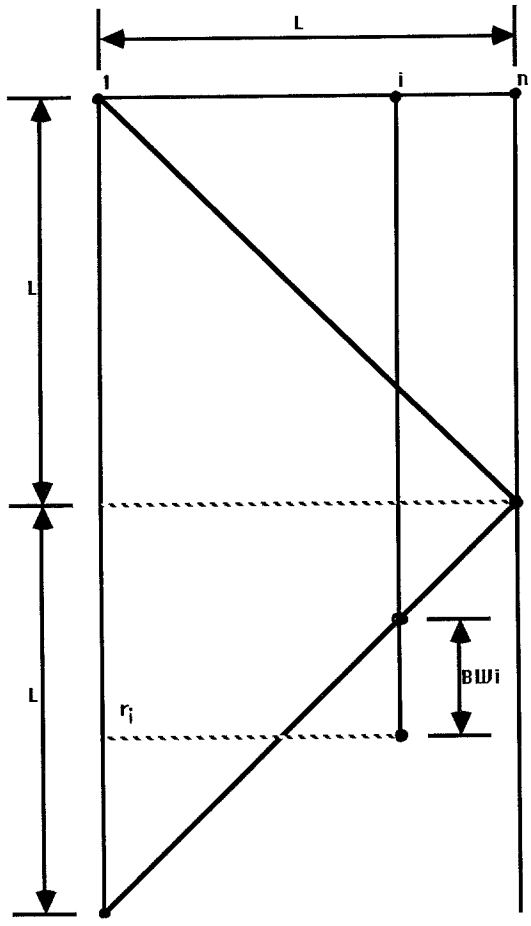

FIGURE 2.2. Definition of BW_i in "TRAVERSE" Heuristic.

$2L + BW_{\max}$, is an upper bound on $C^*_{\max}$. Hence $C^*_{\max} \leq 2L + BW_{\max}$. On the other hand, $C^*_{\max}$ cannot be smaller than the minimum trip duration if we ignore *all* release times, except the one of point i, for any choice of i. Given that such a trip has a minimum duration of $2L + BW_i$, we conclude that $C^*_{\max} \geq 2L + BW_i$ for all i, or $C^*_{\max} \geq 2L + BW_{\max}$. Thus, $C^*_{\max} = 2L + BW_{\max}$, and "TRAVERSE" is optimal for the tour version.

Finally, the $O(n)$ complexity of "TRAVERSE" is obvious.

An $O(n^2)$ Algorithm For the Path Version

We now return to our basic (path) scenario: the vehicle need not return to point 1, but can terminate its route at any point. A cursory investigation shows that although it is always possible to convert a tour problem to a path problem, the opposite is not necessarily possible, even if the last point to be visited in the path version is prescribed. Moreover, although in the tour case the schedule always has a simple pattern, the optimal schedule in a path problem can be fairly complicated (see Figure 2.1 again).

To motivate the solution approach for this problem, let us first present the classical dynamic programming recursion that solves the general TSP with release times and then see how we can exploit the special structure of the problem at hand.

Let $N = \{1, 2, \ldots, n\}$, $S \subseteq N$ and $i \in S$, $1 \in S$. Define $V(i, S)$ as the minimum

time to visit all points in S starting from point 1 and terminating at point i subject to the release time constraints. Then $V(i, S)$ obeys the following recursion:

$$S \neq \{1\}: V(i, S) = \min_{y \in S - \{i\}} \{\max (r_i, t_{yi} + V(y, S - \{i\}))\},$$

$$S = \{1\}: V(1, \{1\}) = r_1.$$

This recursion is, in fact, true for any form of distance matrix $[t_{ij}]$ and can be extended to the case that includes deadlines. The computational effort associated with solving this well-known recursion is exponential: the number of possible subsets S of N is $O(2^n)$, the number of possible states (i, S) is $O(n2^n)$, and the overall running time is $O(n^2 2^n)$.

As we next show, in the straight-line case we can exploit the problem structure to reduce the computational effort from exponential to polynomial. The theorem to follow essentially states that at any time along the optimal schedule the set of visited points is the union of two disjoint sets S_1 and S_2, both of which are "contiguous": S_1 includes all points from point 1 to point j, and S_2 is either empty or includes all points from point k to point n (see Figure 2.3). The theorem also states that one need consider only points j or k to represent the last visited point i along the route, for any given S_1 and S_2 (or point j only if $S_2 = \varnothing$).

THEOREM 2.2. *In the path version of the straight-line case, the dynamic programming recursion need consider only states* (i, S) *with* i *and* S *defined as follows*:

(a) $S = S_1 \cup S_2$ *with* $S_1 = \{x: 1 \leq x \leq j\}$ *and* $S_2 = \{x: k \leq x \leq n\}$ *for some indices* j *and* k *satisfying* $1 \leq j < k \leq n + 1$ (*with the convention that* $S_2 = \varnothing$ *if* $k = n + 1$).

(b) *If* $S_2 = \varnothing$ *then* $i = j$. *Otherwise*, $i = j$ *or* $i = k$.

PROOF. Let R be an optimal route that does not satisfy properties (a) and (b) of the theorem. Then at some time t and for some indices j and k, this route will have visited all points $1 \leq x \leq j$ and $k \leq x \leq n$, but then depart to visit a point p with $j + 1 < p < k - 1$. Among all routes that violate (a) and (b), let R maximize t. Also, let q be the point in R visited after point p.

At some time $t' > t$, route R must for the first time visit either point $j + 1$ or point $k - 1$. After t' route R must visit some point l to the left of p followed by a point r to the right of p, or vice versa (see Figure 2.4). Now consider another route R' that at time t travels to point q rather than point p and then waits at point q to depart at the same time as route R. Route R' will also visit point p on its way traveling from l to r. Otherwise R and R' are identical. (Since $t_{lr} = t_{lp} + t_{pr}$, the timing of routes R and R' coincide after they both visit l, p and r.)

Now either $q = k - 1$ or $j + 1$, or we may repeat the argument and find another route R'' with both p and q visited after time t'. Continuing in this way, we can find another optimal route $\bar{R}$ that visits every point $j + 1 < p < k - 1$ after time t'. Our assumption that route R maximizes t among all routes that violate (a) and (b) implies that $\bar{R}$ satisfies these properties.

Theorem 2.2 implies that the choice of y, the decision variable of the recursion (best immediate predecessor of i) is limited to at most two alternatives, depending on i: if i

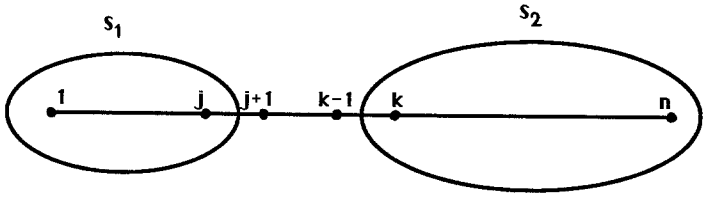

FIGURE 2.3. Sets S_1 and S_2 for the Straight-Line Case.

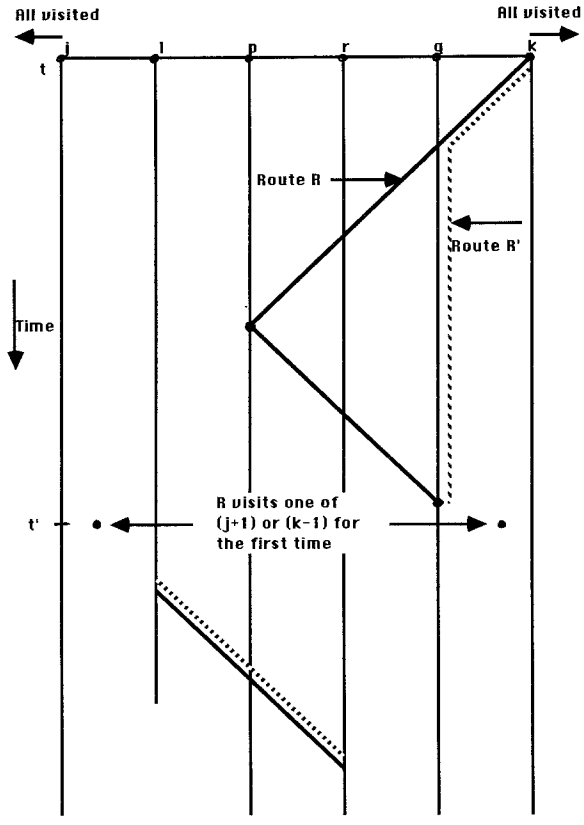

FIGURE 2.4. The Dashed Line Route Can Replace the Solid Line Route with No Additional Delay.

$= j$, then y can be equal to either $j - 1$ (if $S_1 \neq \{1\}$) or k (if $S_2 \neq \varnothing$); if $i = k$, then y can be equal to either $k + 1$ (if $S_2 \neq \{n\}$) or j.

As a result of this observation, it is clear that *only two indices, j and k, are needed to represent the set S of the recursion*. Consequently, it is possible to rewrite the recursion as follows ($1 \leq j < k \leq n + 1$):

$$V(j, j, k) = \min \{\max (r_j, t_{j-1,j} + V(j-1, j-1, k)), \max (r_j, t_{kj} + V(k, j-1, k))\}$$

$$V(k, j, k) = \min \{\max (r_k, t_{k+1,k} + V(k+1, j, k+1)), \max (r_k, t_{jk} + V(j, j, k+1))\}$$

with $V(1, 1, n + 1) = r_1$.

By convention, in the recursion we set $V(0, 0, k) = V(n + 1, j, n + 1) = +\infty$ for all $k > 1, j < n$.

This recursion can be executed in $O(n^2)$ time and the optimal value of the problem is:

$$C^*_{\max} = \min_{1<j\leq n} V(j, j, j+1) = \min_{1<k\leq n+1} V(k, k-1, k)$$

The prior arguments are easily extended to examine the case in which the vehicle starts at time $t = 0$ not at point 1, but at some interior point p between 1 and n. It is straightforward to see that in the tour case the optimal schedule is a traversal of the form $(p, 1, n, p)$ or $(p, n, 1, p)$, with the vehicle waiting at points 1 and/or n for the shortest

amount of time that would permit it to return to point p without waiting at any other point. In the path case, it is also straightforward to see that the optimal schedule always visits an end point first (point 1 or point n), and then exhibits a pattern similar to the one examined earlier in this section.

We also mentioned earlier that the classical dynamic programming recursion can be easily extended to the case that includes ("hard") *deadlines*, that is to situations in which the schedule is required to satisfy the restrictions $C_i \leq d_i$ for a prescribed set of deadlines d_i for the points i. One might wonder, therefore, whether a similar extension is possible for the $O(n^2)$ algorithm as well. Unfortunately, the answer to this question is no, since in the presence of deadlines Theorem 2.2 will not be valid in general. Indeed, if there are deadlines, feasibility conditions might imply that the route shown as the dashed line in Figure 2.4 *cannot* substitute for the one depicted as a solid line. Thus, the *general* case that includes deadlines cannot be solved by an obvious extension of the $O(n^2)$ algorithm. Indeed, at this time, the status of the computational complexity of this case is *open*, and we conjecture the case to be *NP*-complete. *Special* cases with deadlines that *are* solvable in polynomial time include (a) the case of *one common deadline d*, which is solved by applying the $O(n^2)$ algorithm as if no deadline were present and then checking whether $C^*_{\max} \leq d$ (if yes, the $O(n^2)$ algorithm produces the optimal solution, and if no, the problem is infeasible), and (b) the case of *nonoverlapping time windows*, in which the points are visited by increasing order of r_i's (this problem could be infeasible as well, but checking feasibility requires only $O(n)$ time once we have sorted the r_i's, which requires $O(n \log n)$ time).

An extension to the case in which the vehicle is required to spend a known "service" or "handling" time of h_i at each point i would encounter similar obstacles. The status of this case is also open, and we conjecture it to be *NP*-complete as well.

Having examined the straight-line case, we now consider the shoreline case.

3. The Shoreline Case

In the absence of release time constraints, or when these are agreeable, it is easy to see that the shoreline problem can be solved optimally by simply traversing the points in order from 1 to n. In particular, the traveling salesman problem for shoreline problems without time constraints is polynomially solvable.

On the other hand, as is well known, the Euclidean traveling salesman problem is *NP*-complete even in the absence of time windows (Papadimitriou 1977). Nevertheless, the computational complexity of the general shoreline case with release times remains open at this time. In the following discussion, we shall design and analyze approximate algorithms for its solution.

3.1. *Worst-Case Analysis of Heuristics*

In this section, we present a class of simple heuristics and we derive data-dependent worst-case performance ratios for them. We also show that these ratios are bounded by constants. For this purpose, let us first introduce some notation. Let $v_{ij} = \sum_{k=i}^{j-1} t_{k,k+1}$ be the travel time along the shoreline between i and j, $1 \leq i < j < n$. Denote the length of the shoreline, v_{1n}, by S. In addition let $L = t_{1n}$ denote the length of the direct segment between point 1 and point n. Furthermore, define $FW_i = \max\{0, r_i - v_{1i}\}$ and let $FW_{\max} = \max_{1 \leq i \leq n} FW_i$. The quantity FW_i is the amount of time the vehicle would have to wait at point i when traveling along the shoreline from point 1 to point n, if r_i were the only release time constraint enforced. Let also $BW_i = \max\{0, r_i - L - v_{in}\}$ and $BW_{\max} = \max_{1 \leq i \leq n} BW_i$. Finally, let b represent a point on the shoreline for which $BW_b = BW_{\max}$. Note that these latter quantities are the extensions to the shoreline case of the similar quantities defined in §2. We now present a class of heuristics for the shoreline problem.

The (Tour) "TRAVERSE" Algorithm

This procedure is the same one we discussed in §2 as an exact algorithm for the straight-line case; "TRAVERSE" is not necessarily optimal for the general shoreline case. Its computational complexity is still $O(n)$; the following theorem establishes its worst-case behavior.

THEOREM 3.1. *The "TRAVERSE" algorithm has a worst-case performance ratio of* $2S/(S+L)$.

PROOF. First note that $C^T_{\max} = BW_{\max} + L + S$, where $C^T_{\max}$ is the value of the "TRAVERSE" heuristic. If $BW_{\max} = 0$, then $C^T_{\max} = L + S = C^*_{\max}$, i.e., the "TRAVERSE" heuristic is optimal. Assume now that $BW_{\max} > 0$.

Let the tSLPA be a relaxation of the tSLP with no release time constraints. Consider also the problem tSLPB which is a tSLP with only three points: point 1, with $r_1 = 0$, point b with $r_b > r_{1b}$ and point n with $r_n = 0$. Then $C^*_{\max}(A) = L + S$ and $C^*_{\max}(B) = r_b + t_{1b}$, where $C^*_{\max}(A)$ and $C^*_{\max}(B)$ are the optimal completion times for the tSLPA and tSLPB problems, respectively. Since $C^*_{\max} \geq \max\{C^*_{\max}(A), C^*_{\max}(B)\} = \max\{L + Sr_b + t_{1b}\}$ and $C^T_{\max} = BW_{\max} + L + S = r_b + v_{1b}$, we obtain

$$C^T_{\max}/C^*_{\max} \leq (r_b + v_{1b})/\max\{L + S, r_b + t_{1b}\}.$$

Assume first that $L + S \geq r_b + t_{1b}$. Then

$$C^T_{\max}/C^*_{\max} \leq (r_b + v_{1b})/(L + S) \leq (L + S - t_{1b} + v_{1b})/(L + S)$$

$$= 1 + (v_{1b} - t_{1b})/(L + S).$$

Since $S - v_{1b} = v_{bn} \geq t_{bn} \geq L - t_{1b}$, we conclude that $C^T_{\max}/C^*_{\max} \leq 1 + (S - L)/(L + S) = 2S/(L + S)$. Assume now that $L + S < r_b + t_{1b}$. Then

$$C^T_{\max}/C^*_{\max} < (r_b + v_{1b})/(r_b + t_{1b}) < 1 + (v_{1b} - t_{1b})/(L + S) \leq 2S/(L + S).$$

Therefore, the ratio $C^T_{\max}/C^*_{\max}$ is bounded by $2S/(L + S)$.

To show that this bound is tight, we need consider only an example with $r_i = 0$ for $i = 1, \ldots, n - 1$ and $r_n = S$.

Note that if $S/L = 1$ (the straight-line case), then the "TRAVERSE" algorithm's worst-case performance ratio becomes 1, i.e., this method solves the problem exactly. We have already proved this result in §2.

Consider now the path problem. The "TRAVERSE" algorithm for the path problem consists of two parts, each generating a different tour: in the description, Steps 1, 2, and 3 constitute part 1 and Steps 4 and 5 constitute part 2.

The (Path) "TRAVERSE" Algorithm

Step 1. Go straight from point 1 to point n.

Step 2. Wait at point n for an amount of time equal to $BW_{\max} = \max_i\{\max\{0, r_i - L - v_{in}\}\}$.

Step 3. Visit all the points sequentially from point n to point 2. The total duration of the trip is $L + S - t_{12} + BW_{\max}$.

Step 4. Wait at point 1 for an amount of time equal to $FW_{\max}$.

Step 5. Visit all the points sequentially from point 1 to point n. The total duration of the trip is $FW_{\max} + S$.

The value of the heuristic is $C^T_{\max} = \min\{L + S - t_{12} + BW_{\max}, FW_{\max} + S\}$.

Note that in Step 4, by waiting for $FW_{\max}$ at point 1, the vehicle is certain to visit all the points on or after each point's release time. The following theorem specifies the worst-case behavior of this heuristic.

THEOREM 3.2. *The "TRAVERSE" algorithm solves the path problem in* $O(n^2)$ *time and its worst-case performance ratio is* $\min\{2(L+S)/3L, (4S-L)/2S\}$.

Since the proof is similar to that of Theorem 3.1, we will omit it (the reader may refer to Kim 1985 for more details).

The worst-case performance ratios we have derived are data-dependent. Specifically, they are strictly increasing functions of S/L. This ratio can be thought of as a measure of how far a given shoreline is from the straight line (for which $S/L = 1$). Furthermore, it is easy to see that these worst-case performance ratios are bounded by the constant 2, which means that the relative error of these heuristics cannot exceed 100%.

3.2. *Computational Performance of Heuristics*

In this section we summarize the results of a computational study of several heuristics for the "path" version of the problem. The heuristics examined include the "TRAVERSE" algorithm described earlier and what we call the "ZOOM" heuristic. This heuristic is the dynamic programming algorithm developed for the straight-line pSLP. It is easy to see that this algorithm is not necessarily optimal for the general shoreline case. However, its worst-case performance cannot be worse than that of the "TRAVERSE" heuristic, since the "ZOOM" heuristic always checks all the paths considered by the "TRAVERSE" heuristic.

Our computational experiments have been conducted within the context of cargo ship scheduling. We have first established a travel time matrix on a real-world shoreline. The number of points considered was $n = 8$, 10, 20, 30. All travel time matrices satisfied the shoreline requirements but were *not* Euclidean. The corresponding S/L ratios were 1.986, 2.055, 2.075 and 2.60. Then, for each port on the shoreline we have generated random release times that are mutually independent and uniformly distributed between 0 and $R_{\max}$, where $R_{\max}$ is a user-specified parameter. For each given $R_{\max}$ value, we have created 10 problem instances for each problem size. To simulate different degrees of tightness for the release times, we made a series of runs with five different values for $R_{\max}$.

The computational experiments were carried out on an IBM PC and on a SANYO MBC-555 which are personal computers with 256 K RAM. The largest problem for which we were able to obtain the optimal solution was $n = 8$. In the other cases, we used a lower bound on the optimal value given by $\max\{S, \max_i r_i\}$. This bound is, in many cases, very loose.

The computational performance of the "ZOOM" heuristic (average over the 10 problems instances for each $R_{\max}$ value) seemed to be consistently very good for any range of release times and any problem size even though the underlying routing structure was far from the straight line. For $n = 8$, the algorithm never deviated more than 10% from the optimal value, with the average error around 4%. For $n > 8$, the algorithm performed better for the upper range of release times for which it was within 20% of the lower bound in most cases. For the lower range of release times, the deviations were larger (up to 54%), but this result might be attributable to the looseness of the lower bound rather than to the performance of the algorithm. As expected, "TRAVERSE" was consistently inferior to "ZOOM", but never by more than 10% (and about 6% on the average). Further details on these runs and additional approaches we have considered can be found in Kim (1985).

4. Conclusions

In this paper we have introduced a class of single-vehicle routing and scheduling problems with time constraints. These problems share one common feature: because of their special topology, in the absence of time constraints, they are trivial to solve. We have

introduced the "shoreline" topology, a generalization of the straight-line case and a restriction of the triangle-inequality case, and have focused on the problem of minimizing $C_{\max}$ (the time the last point is visited), subject to "release time" constraints. For the straight-line case we have seen that this problem can be solved exactly in $O(n^2)$ time by a dynamic programming algorithm. For the shoreline case we have presented some heuristics and analyzed their worst-case and practical performances.

The problems examined in this paper are members of a much broader family of problems, each having a different objective function, type of distance matrix, and constraints. For instance, instead of minimizing $C_{\max}$ we might wish to minimize $\sum C_i$, the sum of completion times (or, equivalently, the average completion time). Or, we might also impose deadlines (or time windows) on the times each point can be visited. The pertinent question is whether the results of this paper can be extended to these other variants.

Kim (1985) carried out a cursory investigation to answer this question. He describes a taxonomy of 28 different variants of this problem class, broken down by type of objective, type of distance matrix and type of time constraints. His analysis provides reducibility relationships among these variants, and a preliminary examination of their complexity. The analysis of many of these variants is still inconclusive, for the complexity status of many of them is still open. To our knowledge, the only variant among this class that is provably *NP*-complete is the problem of minimizing $\sum C_i$ on a straight line, subject to time window constraints on each point (release times and "hard" deadlines). This result is due to a transformation from the "traveling repairman problem," examined by Afrati et al. (1986). As with machine scheduling problems, the resolution of exactly which among this class of problems are polynomially solvable and which are *NP*-complete is an effort that would probably require a nontrivial number of man-years to complete (not to mention the many more extensions, such as the multi-vehicle case, that could be considered). We think that such a research effort would be worthwhile for two reasons. First, from a practical perspective, in many routing and scheduling situations, the imposition of time constraints destroys a relatively simple routing structure. Second, from a more theoretical perspective, progress in this area would shed more light on the exact location of the boundary between problems in *P* and *NP*-complete problems and would enhance the state of the art in routing and scheduling of vehicles with time windows (in itself an already rapidly growing area).[1]

[1] Work on this paper was supported in part by contract No. N00016-83-K-0220 of the Office of Naval Research, by an internal grant of the MIT Center for Transportation Studies, by an internal grant from Northeastern University's Research and Scholarship Development Fund, and by Grant # ECS-83-16224 of the National Science Foundation.

References

AFRATI, F., S. COSMADAKIS, C. PAPADIMITRIOU, G. PAPAGEORGIOU AND N. PAPAKOSTANTINOU, "The Complexity of the Travelling Repairman Problem," *Information Theory Appl.* (France), 20 (1986), 79–87.

BAKER, E., "An Exact Algorithm for the Time-Constrained Traveling Salesman Problem," *Oper. Res.*, 31 (1983), 938–945.

BODIN, L., B. GOLDEN, A. ASSAD AND M. BALL, "Routing and Scheduling of Vehicles and Crews: The State of the Art," *Computers and Operations Res.*, 10 (1983), 62–212.

CHRISTOFIDES, N., A. MINGOZZI AND P. TOTH, "State-Space Relaxation Procedures for the Computation of Bounds to Routing Problems," *Networks*, 11 (1981), 2.

DESROSIERS, J., F. SOUMIS AND M. DESROCHERS, "Routing with Time Windows by Column Generation," *Networks*, 14 (1984), 545–565.

———, Y. DUMAS AND F. SOUMIS, "A Dynamic Programming Method of the Large-Scale Single-Vehicle Dial-A-Ride Problem with Time Windows," *Amer. J. Math. and Management Sci.*, 6 (1986), 301–325.

FISHER, M. AND M. ROSENWEIN, "An Interactive Optimization System for Bulk Cargo Ship Scheduling," Working Paper, 85-08-07, University of Pennsylvania, 1985.

GAREY, M. AND D. JOHNSON, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman and Company, San Francisco, 1979.

JAW, J., A. ODONI, H. PSARAFTIS AND N. WILSON, "A Heuristic Algorithm for the Multi-Vehicle Advance-Request Dial-A-Ride Problem with Time Windows," *Transportation Res.*, 20B (1986), 243–257.

KIM, TAI-UP, "Solution Algorithms for Sealift Routing and Scheduling Problems," PhD. Dissertation, Massachusetts Institute of Technology, 1985.

LENSTRA, J., A. RINNOOY KAN AND P. BUCKER, "Complexity of Machine Scheduling Problems, Studies in Integer Programming," *Annals of Discrete Mathematics*, 1 (1977), 343–362, North-Holland Publishing Company, Amsterdam.

PAPADIMITRIOU, C., "The Euclidean Traveling Salesman Problem is *NP*-Complete," *Theoret. Comput. Sci.*, 4 (1977), 237–244.

PSARAFTIS, H., "An Exact Algorithm for the Single Vehicle Dial-A-Ride Problem with Time Windows," *Transportation Sci.*, 17 (1983), 351–357.

———, "Scheduling Large Scale Advance Request Dial-A-Ride Systems," *Amer. J. Math. and Management Sci.*, 6 (1986), 327–367.

———, J. ORLIN, B. BIENSTOCK AND P. THOMPSON, "Analysis and Solution Algorithms of Sealift Routing and Scheduling Problems: Final Report," Working Paper, No. 1700-85, Sloan School of Management, MIT, 1985.

RONEN, D., "A Review of Cargo Ships Routing and Scheduling Models," *European J. Oper. Res.*, 12 (1983).

SEXTON, T. AND L. BODIN, "Optimizing Single Vehicle Many-to-Many Operations with Desired Delivery Times: I. Scheduling," *Transportation Sci.*, 19 (1985a), 378–410.

——— AND ———, "Optimizing Single Vehicle Many-to-Many Operations with Desired Delivery Times: II. Routing," *Transportation Sci.*, 19 (1985b), 411–435.

SOLOMON, M., "Algorithms for the Vehicle Routing and Scheduling Problem with Time Window Constraints," *Oper. Res.*, 35 (1987), 254–265.

———, "On the Worst-Case Performance of Some Heuristics for the Vehicle Routing and Scheduling Problem with Time Window Constraints," *Networks*, 16 (1986a), 161–174.

———, "The Minimum Spanning Tree Problem with Time Window Constraints," *Amer. J. Math. and Management Sci.*, 6 (1986b), 499–421.

———, E. BAKER AND J. SCHAFFER, "Vehicle Routing and Scheduling Problems with Time Window Constraints: Efficient Implementations of Solution Improvement Procedures," In *Vehicle Routing: Methods and Studies*, B. Golden and A. Assad (Eds.), North-Holland Publishing Co., Amsterdam, 1988.

——— AND J. DESROSIERS, "Time Window Constrained Routing and Scheduling Problems: A Survey," *Transportation Sci.*, 22 (1988), 1–13.