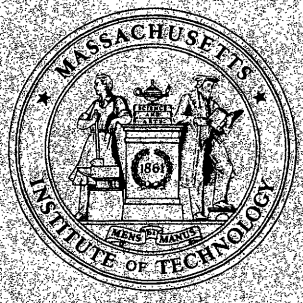
OPERATIONS RESEARCH CENTER

working paper



MASSACHUSETTS INSTITUTE OF TECHNOLOGY

ROUTING AND SCHEDULING ON A SHORELINE WITH RELEASE TIMES

by

Harilaos N. Psaraftis
Marius M. Solomon
Thomas L. Magnanti
Tai-Up Kim

OR 152-86 September 1986

ABSTRACT

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 resulting routing and scheduling problem is nontrivial. From a theoretical perspective, this class of problems lies on the borderline between problems in P and those that are NP-complete. For the straight-line case (a restriction of the shoreline case), our analysis shows that the problem of minimizing 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. We also discuss how these algorithms perform on practical data. Finally, we examine the computational complexity of other problem variants involving alternative objective functions and different types of time window constraints.

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 this proves desirable or necessary, we wish to find the schedule by which all cargoes are picked up 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 $1 \leq i \leq k \leq j \leq n$,

(i) $t_{ii} = 0$

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

(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 non-ordered set of n points and their corresponding distance matrix, it is possible to check in $O(n^2)$ time whether there is a shoreline on which these points are located, 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 aplications, 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, one 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 non-trivial. Similar situations may be encountered in other routing and scheduling environments.

From a theoretical perspective, this class of problems is interesting because it lies on the borderline between problems in P and NP-complete

problems. As we shall see, seemingly minor variations in the definition of a problem can transform it from polynomially solvable to NP-complete.

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), Baker and Schaffer (1984), Desrosiers et al. (1983), (1985), Jaw et al. (1984), Psaraftis (1983), Sexton and Bodin (1983a), (1983b), Solomon (1983), (1984a), (1984b), and Solomon et al. (1986), to name just a few. The literature on applications for the routing and scheduling of ships is considerably less rich, but also growing: see for instance, Psaraftis et al. (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,..., n, equal the corresponding distances. Also without loss of generality, as in our previous definition, we number the points according to the order that they are encountered when one 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. 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 <u>departs</u> 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 this 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).

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 Section 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. Section 3 is concerned with the general shoreline SLP. Here we develop several heuristics, derive data-dependent worst-case performance bounds for them and conjecture that for Euclidean distances these are bounded by constants. We also discuss how these algorithms perform in practice. In Section 4 we examine the computational complexity of and the relation among other problem variants involving alternative objective functions and different types of time window constraints. Finally, Section 5 presents our conclusions.

2. THE STRAIGHT LINE CASE

We begin our investigation by analysing the straight-line case with release times. In this case, all points 1, 2,...,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).

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 vertical with increasing time downward). 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 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,...,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 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, $2L + BW_{max}$, is an upper bound on C^*_{max}, the minimum possible duration. Hence $C^*_{max} \leq 2L + BW_{max}$. On the other hand, C^*_{max} cannot be smaller than the minimum trip duration if all release times, except the one of point i, are ignored, 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, $C^*_{max} = 2L + BW_{max}$, that is, "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 again Figure 2.1).

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 $S_2 = \emptyset$).

Theorem 2.2 In the straight line case (the "path" version with release times only) one need consider only states (i,S) for which i and S are defined as follows (see also Figure 2.3):

(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 $S_2 = \emptyset$ if $k = n+1$).

(b) If $S_2 = \emptyset$ then $i = j$. Otherwise, $i = j$ or 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 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 1 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 1 to r. Otherwise R and R' are identical. (Since $t_{1r} = t_{1p} + t_{pr}$, the timing of routes R and R' coincide after they both visit 1, 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 = j then y can be equal to either j - 1 (if $S_1 \neq \{1\}$) or k (if $S_2 \neq \emptyset$); 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:

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

We mentioned earlier that the classical DP recursion can be easily extended to the case that includes deadlines, that is, to the case in which we further require that the schedule be such that $C_i \leq d_i$ for all i. Here d_i is a prescribed deadline for point 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 extension of the $O(n^2)$ algorithm. Indeed, and as we note in our discussion of other problem variants in Section 4, the status of the computational complexity of this case is at this time 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 the r_i's have been sorted in O(nlogn) time).

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 general shoreline case with release times has so far resisted our attempts to either solve it in polynomial time, (possibly as an

extension of the straight-line case) or prove that it is an NP-complete problem. We conjecture, however, that this problem is NP-complete. In this light, we shall design and analyze approximate algorithms for its solution.

3.1 Worst-Case Analysis of Heuristics

In this section, we present two classes of simple heuristics and derive data-dependent worst-case performance ratios for them. 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, $1 \leq i < j \leq n$. 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. Let also $r^* = r_m = \max_{1 \leq i \leq n} r_i$. Furthermore, define $FW_i = \max\{0, r_i - v_{1i}\}$ and let $FW^* = FW_s = \max_{1 \leq i \leq n} FW_i$. The quantity $FW_i = \max\{0, r_i - v_{1i}\}$ 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. Finally, let $BW_i = \max\{0, r_i - L - v_{in}\}$ and $BW^* = BW_b = \max_{1 \leq i \leq n} BW_i$. Note that these latter quantities are the extensions to the shore-line case of the similar quantities defined in Section 2. We present now two classes of heuristics for the shoreline problem.

The (Tour) "TRAVERSE" Heuristic

The description of this procedure for the tour problem has been given in Section 2. Its worst-case behavior is established in the following theorem.

Theorem 3.1 The "TRAVERSE" heuristic solves the tour problem in O(n) time and its worst-case performance ratio is $2S/(S + L)$.

Proof: If $BW^* = 0$, it is obvious that $C^T_{max} = L + S$, where C^T_{max} is the value of the "TRAVERSE" heuristic. Assume now that $BW^* > 0$. Then $r_b > L + v_{bn}$.

Let now the tSLPA be the relaxation of the tSLP where there are no release time constraints. Then, it is clear that C^*_{max} (A) = L + S, where C^*_{max} (A) is the optimal completion time for the tSLPA.

Consider now problem tSLPB which is a tSLP with only three points: point 1, with $r_1 = 0$, point b with $r_b > t_{1b}$ and point n with $r_n = 0$. Then, C^*_{max} (B) = $r_b + t_{1b}$. Hence $C^*_{max} \geq \max \{C^*_{max}(A), C^*_{max}(B)\} = \max \{L+ S, r_b + t_{1b}\}$. Furthermore,

$C^T_{max} = BW^* + L + S = r_b + v_{1b}$, and hence

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

Now , if $L + S \geq r_b + t_{1b}$

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

$= 1 + (v_{1b} - t_{1b})/(L+S) \leq 1 + (S-L)/(L+S) = 2S/(L+S)$, since

$S - v_{1b} = v_{bn} \geq t_{bn} \geq L - t_{1b}$. Assume now that $L + S < r_b + t_{1b}$. Then:

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

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

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

Consider now the path problem. The "TRAVERSE" heuristic is:

(Path) "TRAVERSE" Heuristic

Step 1: Go straight form 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 $L + S - t_{12} + BW^*$.

Step 4: Wait at point 1 for an amount of time equal to FW*.

Step 5: Visit all the points sequentially from point 1 to point n. The total duration of the trip is FW* + S. The value of the heuristic is $C^T_{max} = \min \{L + S - t_{12} + BW^*, FW^* + S\}$

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" heuristic 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 2.1, we will omit it (the reader may refer to Kim (1985) for more details).

A second class of heuristics is the "MAXEPT" class. For the tour problem we have:

(Tour) "MAXEPT" Heuristic

Step 1 Go straight from point 1 to point m (recall that $r_m = r^* = \max_i r_i$).

Step 2 Visit all the points from m to n.

Step 3 Return directly to point m-1.

Step 4 Visit all the points from m-1 to 1. The total duration of the trip is $C^M_{max} = \max\{r^*, t_{1m}\} + S - t_{m-1,m} + t_{m-1,n}$.

The intuition underlying this procedure is that a schedule that goes directly to the point with the largest release time will visit the rest of the points after their release time, and hence incur no additional waiting time.

A proof similar to that in Theorem 3.1 establishes the following result.

Theorem 3.3 The worst-case performance ratio of the "MAXEPT" heuristic is 2 and its complexity is O(n).

For the path problem, this heuristic can be described as follows:

(Path) "MAXEPT" Heuristic

Step 1 Go straight from point 1 to point m.

Step 2 Visit all the points from m to n.

Step 3 Return directly to point m-1.

Step 4 Visit all the points from m-1 to 2. The total duration of this trip is $\max\{r^*, t_{1m}\} + S - t_{12} - t_{m-1,m} + t_{m-1,n}$

Step 5 Go to point 1.

Step 6 Go directly from point 1 to m.

Step 7 Visit all the points from m to 2.

Step 8 Go directly to point m + 1.

Step 9 Visit all the points from m+1 to n. The total duration of this trip is $\max\{r^*, t_{1m}\} + S - t_{12} - t_{m,m+1} + t_{2,m+1}$.

The value of this heuristic is $C^M_{max} = \min\{(\max\{r^*, t_{1m}\}+S-t_{12}-t_{m-1,m}+t_{m-1,n}), (\max\{r^*, t_{1m}\} +S-t_{12}-t_{m,m+1}+t_{2,m+1})\}$

Methods similar to the ones presented previously, establish the following result:

Theorem 3.4 The "MAXEPT" heuristic solves the path SLP in O(n) time and its worst-case performance ratio is (L + S)/L.

From these results it is obvious that the worst-case performance of the "TRAVERSE" heuristic is better than that of the "MAXEPT" heuristic. A strong-

er result is that the former heuristic always performs better than the latter as shown below.

Theorem 3.5 The "TRAVERSE" heuristic always performs better than the "MAXEPT" heuristic.

Proof: For the tSLP, if BW* > 0, then $C^T_{max} = r_b + v_{1b}$ and

$$C^M_{max} - C^T_{max} \geq \max(r^*, t_{1m}) - r_b + v_{bn} + t_{m-1,n} - t_{m-1,m} \geq 0.$$

For the pSLP, if $C^T_{max} = FW^* + S = r_s - v_{1s} + S$,

$$C^M_{max} - C^T_{max} \geq \max(r^*, t_{1m}) - r_s + v_{1s} - t_{12} \geq 0.$$

The proofs for the other cases are similar so they will be ommitted.

So far, all 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).

To complete our worst-case analysis, we have to investigate the worst possible value of the ratio S/L itself. For instance, one might wonder whether this ratio can be made arbitrarily large, or whether it is bounded by a constant. Thus far our analysis has shown that the answer to this question depends on whether or not the distance matrix is restricted to be Euclidean. Specifically, the following observations are in order:

(1) If the distance matrix is not restricted to be Euclidean (but still satisfies the shoreline requirements), S/L can grow as fast as n. In fact, for a distance matrix in which all interpoint distances are the same, S/L equals n-1.

(2) If the distance matrix is Euclidean, we conjecture that $S/L \leq 2\pi/3 (\doteq 2.094)$ and that this upper bound is tight.

The investigation of the maximum possible value of the S/L ratio in the Euclidean case is rather complicated. The limiting case $S/L = 2\pi/3$ corresponds to the "half football" continuous curve shown in Figure 3.1 (this curve consists of two 60^{0} circular arcs AO and BO, each of radius L, centered at endpoints B and A respectively). The complicating factor in the Euclidean shoreline case is due to the fact that whereas a discrete set of n points may satisfy the shoreline conditions, the continuous piecewise linear curve that joins these points may not.

If our conjecture for the Euclidean shoreline case is correct, that is, if the ratio S/L is bounded from above by a constant, then each worst-case performance ratio we have derived is also bounded by a constant.

3.2 Computational Performance of Heuristics

In this section we conduct a computational study of several heuristics for the "path" version of the problem. The heuristics examined include the "TRAVERSE" and "MAXEPT" heuristics described earlier, plus another three algorithms. The first approach we considered is:

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.

Another approach we examined is:

The "GREEDY" Heuristic

In this nearest neighbor heuristic, the vehicle always moves from its current location to visit the "nearest" point. The metric we are using tries

to account for both the geographical and temporal closeness of the points. This measure is defined as $\max\{r_j - a_i, t_{ij}\}$, where a_i is the departure time from the current point i, and j is yet unvisited.

Since the candidates for the next visit are, at worst n points, the overall time bound will be $O(n^2)$.

Finally, we have developed:

The "ADJACENT POINT INTERCHANGE" (API) Heuristic

This algorithm can be described as follows:

Given the vehicle's current location, determine the three adjacent points closest to the origin and as yet unvisited. The heuristic then tries to interchange the first and second points (in order of increasing distance from point 1). If the interchange is feasible and it results in a smaller arrival time at the third point, the second point becomes the next point to be visited; otherwise, the first point becomes the next point to be visited. This algorithm will take $O(n^2)$ time.

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, namely, the East Coast of North America (from Boston, to Cristobal, the entrance to the Panama Canal). 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 (note that the last ratio exceeds the maximum ratio for a Euclidean case). Then, for each port on the shoreline we have generated random release times which 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 R_{max} equal to 7, 14, 21, 28, and 35 days. In all runs, the ship is initially located in Boston.

The computational experiments were carried out on an IBM PC and on a SANYO MBC - 555 which are personal computers with 256 K RAM. We summarize the results in Table 3.1, where each value represents the average ratio, over the 10 problem instances, of the objective value obtained by the heuristics divided by either the optimal value or a lower bound. Due to the limit of static memory allocation of the Microsoft Pascal Compiler, n=8 was the largest problem on which we were able to run the exact $O(n^2 2^n)$ Dynamic Programming algorithm of Psaraftis (1983) and obtain the optimal solution. In the other cases, we used a lower bound on the optimal value given by max {S, r*}. This bound is, in many cases, very loose.

The computational performance of the "ZOOM" heuristic seems to be remarkably good for any range of release times and any problem size. For n=8, the algorithm never deviated more than 10% from the optimal value. 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 ratios are larger, but this result might be attributable to the looseness of the lower bound rather than to the performance of the algorithm.

Not surprisingly, the performance of the "GREEDY" heuristic was very good for small problem sizes and large release times ranges. This performance may be due to the dominance of the release times; the traveling times become insignificant, thus the routing problem becomes a "release time-game."

The performance of the "API" heuristic was very good, on all problem sizes, for low release times ranges. It seems that this behavior, as opposed to that of the "GREEDY", can be explained by realizing that in a low release

times range the spatial rather than the temporal component is most significant.

In conclusion, the "ZOOM" heuristic consistently performed very well even though the underlying routing structure was far from the straight line. Finally, the performance of the "MAXEPT" and "TRAVERSE" heuristics were inferior to that of the "ZOOM" heuristic. (Further details on these runs be found in Kim (1985)).

4. OTHER PROBLEM VARIANTS

So far we have focused on the problem of minimizing C_{max} subject only to release time constraints. This problem, defined either on the straight line network or on the more general "shoreline" network, is but one member of a broader family of problems, each having a different objective function and/or different time constraints. The purpose of this section is to introduce these other problem variants and report our (limited) knowledge about their status. We continue assuming that the vehicle is at point 1 at time t = 0 and that we do not require the vehicle to return to point 1 at the end of its schedule.

Let us classify each problem in this class by the triplet [R|T|Z], with parameters R, T and Z representing the distance metric, the time constraint configuration and the objective function respectively. By convention, R = ST for the straight line case and R = SH for the shoreline case.

Parameter T may correspond to one of the following cases:

(a) T=r, the case with release times (only), that is, the case examined in the previous two sections in which we require that $C_i \geq r_i$ for all i.

(b) T=d, the case with "hard" deadlines (only). In this case, each point i must be visited no later than a specified

deadline d_i, that is, $C_i \leq d_i$ for all i. In this instance, the overall schedule is infeasible if $C_i > d_i$ for some i.

(c) T=sd, the case with "soft" deadlines (or, due dates). In this case each point i again has a deadline d_i, but we no longer require the deadline to be a hard constraint. The potential violation of deadlines obviously makes sense if we penalize such violations by an appropriate choice of Z (see items (1)-(4) to follow).

(d) T = r+d, the "time window" case with "hard" deadlines, a combination of cases (a) and (b), and

(e) T = r+sd, the "time window" case with "soft" deadlines, a combination of cases (a) and (c).

Finally, parameter Z may correspond to one of the following cases:

(1) $Z = C_{max}$, as discussed already in Sections 2 and 3.

(2) $Z = \Sigma C_i$, the case in which we wish to minimize the sum of completion times $\sum_{i=1}^{n} C_i$ (or, equivalently, the average completion time $(1/n)\sum_{i=1}^{n} C_i$)

(3) $Z = L_{max}$, the case in which we wish to minimize the maximum lateness $L_{max} = \max_i L_i$, and the lateness of point i is defined as $L_i = C_i - d_i$ for all i.

(4) $Z = \Sigma T_i$, the case in which we wish to minimize the total tardiness $\sum_{i=1}^{n} T_i$ and the tardiness of point i is defined as $T_i = \max(L_i, 0)$ for all i,

(5) $Z = \Sigma U_i$, the case in which we wish to minimize the number of late visits $\sum_{i=1}^{n} U_i$ and $U_i = 1$ if $C_i > d_i$ and is zero otherwise; and finally,

(6) $Z = \emptyset$, by which we denote the "feasibility case" for which no optimization is involved. In this instance, we simply wish to find whether the problem has a feasible schedule, that is, one satisfying the constraints imposed by T.

Several remarks are in order:

(i) Although it is possible to define a few additional variants with respect to Z, we shall not examine these in this paper. For instance, we shall not examine the cases $Z=\Sigma w_i C_i$, $Z=\Sigma w_i T_i$, or $Z=\Sigma w_i U_i$, in which each point i "weighed" by a prescribed weight $w_i > 0$. Similarly, we shall not examine the cases $Z = \Sigma L_i$ or $Z = T_{max}$, because the former is equivalent to $Z = \Sigma C_i$ ($\Sigma C_i - \Sigma L_i = \Sigma d_i = \text{const}$) and the latter is minimized whenever L_{max} is minimized (because $\min T_{max} = \max(\min L_{max}, 0)$ - the opposite is not necessarily true).

(ii) For a specific R, not all combinations of T and Z will make sense. Obviously, the objective $Z = \emptyset$ (the feasibility case) is meaningful only if T = d or T = r+d, for in all other cases the problem always has a feasible schedule. In addition, if T = r, the objectives $Z = L_{max}$, $Z = \Sigma T_i$, and $Z = \Sigma U_i$ become irrelevant. Also, if T=sd or T=r+sd (soft deadlines), the objectives $Z = C_{max}$ and $Z=\Sigma C_i$ become irrelevant too, for they do not account for the deadlines that could be violated anyway without penalty. Finally, we shall not examine the variants [R|d|Z] and [R|r+d|Z] with R = any and with $Z = L_{max}$, ΣT_i, or ΣU_i. These cases either have optimal solutions identical with those of the corresponding soft deadline variants [R|sd|Z] and [R|r+sd|Z] respectively (this is true if $\min Z \leq 0$ for $Z = L_{max}$, and if $\min Z = 0$ for $Z = \Sigma T_i$ or ΣU_i), or, are infeasible (otherwise).

(iii) Our prior choices reduce the number of relevant variants from 30 to 14 for a specific R. Thus, for hard deadlines or no deadlines (T = d,r,r+d), Z can be C_{max} or ΣC_i. For soft deadlines (T = sd, r+sd), Z can be L_{max} or ΣU_i. Finally, we have the two feasibility cases $Z=\emptyset$ for T = d, r+d.

Relationships Among Variants

It is obvious that these variants are related to one another. First, all variants of the form [ST|T|Z] are special cases of (or "reduce" to)

variants of the form [SH|T|Z] for all relevant pairs of T and Z. Second, all variants of the form [R|r|Z] and [R|d|Z] reduce to the more general case [R|r+d|Z] for any R and relevant Z. Similarly, all variants of the form [R|r|Z] and [R|sd|Z] reduce to the more general case [R|r+sd|Z] for any R and relevant Z.

There are also some other more interesting and less obvious relationships among these problems. Figure 4.1 gives a schematic representation of these relationships; in this figure, an arc from problem P' to problem P means that the former reduces to the latter. Problems that are linked by bidirectional arcs are "equivalent", that is, reduce to one another.

Reductions corresponding to unnumbered arcs in Figure 4.1 are obvious. In the following discussion we highlight proofs for the reductions corresponding to the eleven numbered arcs in the figure. We note that although many of the reductions to be proved are completely analogous to similar reductions that have been reported in the literature for variants of the single-machine scheduling problem (see for instance Lenstra et al. (1977)), there are also some other reductions that have no analogous to (or have not been reported in) the single-machine scheduling context (to our knowledge).

Let P' -> P denote that problem P' reduces to problem P and let P' <-> P denote that P' and P are equivalent. To investigate reductions, we first convert each one of these problems (cast, except for $Z=\emptyset$, as optimization problems) to "recognition" (or "decision") problems. For R=ST, these recognition versions are defined as follows:

Instance of P: (n, x, r, d, y) with $n \varepsilon Z^+$, $x_i \varepsilon Z_0^+$, $r_i \varepsilon Z_0^+$, $d_i \varepsilon Z_0^+$ for all i, and $y \varepsilon Z$.

Question: Is there a schedule (C_i) with $C_i \varepsilon Z_0^+$ for all i that satisfies all of the requirements listed below? (as appropriate).

(a) $|C_i - C_j| \geq |x_i - x_j|$ for all (i,j).

(b) For T = r, r+d, r+sd , $C_i \geq r_i$ for all i.

(c) For T = d, r+d , $C_i \leq d_i$ for all i.

(d) For Z = C_{max} , $C_i \leq y$ for all i.

(e) For Z = L_{max}, $C_i - d_i \leq y$ for all i.

(f) For Z = ΣC_i , $\sum_{i=1}^{n} C_i \leq y$.

(g) For Z = ΣT_i , $\sum_{i=1}^{n} \max(C_i - d_i, 0) \leq y$.

(h) For Z = ΣU_i , $|\{i: C_i > d_i\}| \leq y$.

(The definition for R = SH is identical with this definition above except that the set of distances $t_{ij} \varepsilon Z^+$ replaces the set of x_i's in the problem instance, and constraint (a) is replaced with $|C_i - C_j| \geq t_{ij}$ for all (i, j)).

P' -> P if for every instance (n',x',r',d',y') of P' it is possible to construct in polynomial time an instance (n,x,r,d,y) of P so that the answer to P is yes if and only if the answer to P' is also yes. We now prove the reductions (corresponding to the numbered arcs of Figure 4.1):

(1) $[R|r+d|\emptyset]$ <-> $[R|r+d|C_{max}]$.

Proof: (1a) Prove that P' = $[R|r+d|\emptyset]$ -> P = $[R|r+d|C_{max}]$. Take n=n', $r_i = r_i'$, $d_i = d_i'$ for all i and $y = \max_i d_i$. Then if P' has a solution C_i' with $C_i' \leq d_i'$, P has a solution $C_i = C_i'$ for all i with $C_i \leq d_i$ and $C_i \leq y$ for all i. And if P has a solution C_i with $C_i \leq d_i$ and $C_i \leq y$ for all i, P' has a solution $C_i' = C_i$ with $C_i' \leq d_i'$ for all i.

(1b) Prove that P' = $[R|r+d|C_{max}]$ -> P = $[R|r+d|\emptyset]$. Take n=n', $r_i = r_i'$, and $d_i = \min(d_i', y')$ for all i. Then if P' has a solution C_i' with $C_i' \leq d_i'$ and $C_i' \leq y'$ for all i, P has a solution $C_i = C_i'$ for all i with $C_i \leq d_i$. And if P has a solution C_i with $C_i \leq d_i$ for all i, P' has a solution $C_i' = C_i$ for

all i with $C_i' \leq d_i'$ and $C_i' \leq y'$ for all i. (Note: This result essentially states that in terms of computational complexity there is no difference between the feasibility problem and the-seemingly more difficult-optimization problem whose objective is to minimize C_{max})

(2) $[R|d|\emptyset] <-> [R|d|C_{max}]$.

Proof: Follows immediately from (1) (take $r_i = r_i' = 0$ for all i).

(3) $[R|r+d|\emptyset] -> [R|r+d|\Sigma C_i]$.

Proof: Take $n = n'$, $r = r_i'$, $d_i = d_i'$ for all i and $y = \sum_{i=1}^{n} d_i$. Then if P' has a solution C_i' with $C_i' \leq d_i'$, P has a solution $C_i = C_i'$ for all i with $C_i \leq d_i$ for all i and $\sum_{i=1}^{n} C_i \leq y$. And if P has a solution C_i with $C_i \leq d_i$ for all i (and with $\sum_{i=1}^{n} C_i \leq y$), P' has a solution $C_i' = C_i$ with $C_i' \leq d_i'$ for all i.

(Note: unfortunately, this reduction does not go the other way. Therefore, the ΣC_i optimization problem is at least as hard as-and may in fact be harder than-the feasibility problem in terms of complexity.)

(4) $[R|d|\emptyset] -> [R|d|\Sigma C_i]$.

Proof: Follows immediately from (3) (take $r_i = r_i' = 0$ for all i).

(5) $[R|r+d|C_{max}] <-> [R|r+sd|L_{max}]$.

Proof: (5a) Prove the $P' = [R|r+d|C_{max}] -> P = [R|r+sd|L_{max}]$. Take $n = n'$, $r_i = r_i'$, $d_i = \min(d_i', y')$ for all i and $y = 0$. Then if P' has a solution C_i' with $C_i' \leq d_i'$ and $C_i' \leq y$ for all i, P has a solution $C_i = C_i'$ for all i with $C_i - d_i \leq 0$ for all i. And if P has a solution C_i with $C_i - d_i \leq 0$ for all i, P' has a solution $C_i' = C_i$ for all i with $C_i' \leq d_i$ and $C_i' \leq y$ for all i.

(5b) Prove that $P' = [R|r+sd|L_{max}] -> P = [R|r+d|C_{max}]$. Take $n = n'$, $r_i = r_i'$, $d_i = d_i' + y'$ for all i and $y = y' + \max_i d_i$. Then if P' has a solution C_i' with $C_i' - d_i' \leq y'$ for all i, P has a solution $C_i = C_i'$ for all i with $C_i \leq d_i$ for all i and $C_i \leq y$ for all i. And if P has a solution C_i with $C_i \leq d_i$ for all i and $C_i \leq y$ for all i, then P_i' has a solution $C_i' = C_i$ for all i with $C_i' - d_i' \leq y'$ for all i.

(6) $[R|d|C_{max}] <-> [R|sd|L_{max}]$.

Proof: Follows immediately from (5) (take $r_i = r'_i = 0$ for all i).

(7) $[R|r+sd|L_{max}] -> [R|r+sd|\Sigma T_i]$.

Proof: Take $n = n'$, $r_i = r'_i$, $d_i = d'_i + y'$ for all i and $y = 0$. Then if P' has a solution C'_i with $C'_i - d'_i \leq y'$ for all i, P has a solution $C_i = C'_i$ for all i with $\sum_{i=1}^{n} \max(C_i - d_i, 0) \leq y$, because both the left hand side and the right hand side are zero. And if P has a solution C_i with $\sum_{i=1}^{n} \max(C_i - d_i, 0) \leq 0$ then $C_i \leq d_i$ for all i and thus P' has a solution $C'_i = C_i$ for all i with $C'_i - d'_i \leq y'$ for all i.

(8) $[R|sd|L_{max}] -> [R|sd|\Sigma T_i]$.

Proof: Follows immediately from (7) (take $r_i = r'_i = 0$ for all i).

(9) $[R|r+sd|L_{max}] -> [R|r+sd|\Sigma U_i]$.

Proof: Same as proof for (7), except substitute $|\{i: C_i > d_i\}|$ for $\sum_{i=1}^{n} \max(C_i - d_i, 0)$.

(10) $[R|sd|L_{max}] -> [R|sd|\Sigma U_i]$.

Proof: Follows immediately from (8) (take $r_i = r'_i = 0$ for all i).

(11) $[R|r|\Sigma C_i] -> [R|r+sd|\Sigma T_i]$.

Proof: Take $n = n'$, $r_i = r'_i$, $d_i = 0$ for all i and $y' = y$. The reduction is then obvious.

Note that in Figure 4.1 we have omitted arrows that represent reductions that can be proven by transitivity (for instance, $[R|r|C_{max}] -> [R|r+sd|L_{max}]$ because $[R|r|C_{max}] -> [R|r+d|C_{max}]$ and $[R|r+d|C_{max}] -> [R|r+sd|L_{max}]$, etc.). We also note that to our knowledge, the equivalences $[R|r+d|C_{max}] <-> [R|r+sd|L_{max}]$ and $[R|d|C_{max}] <-> [R|sd|L_{max}]$ have been reported in the single-machine scheduling context only in one direction ($C_{max} -> L_{max}$ and not vice versa). Section 5 further describes the possible relation of these problems with the machine scheduling class.

Complexity Status

The following discussion summarizes our current state of knowledge regarding the computational complexity of this class of problems.

(1) For the straight line case (R = ST), the following variants are trivially solvable and thus belong to P. $[ST|d|\emptyset]$, $[ST|d|C_{max}]$, $[ST|d|\Sigma C_i]$, $[ST|sd|L_{max}]$, $[ST|sd|\Sigma T_i]$ and $[ST|sd|\Sigma U_i]$. The solution in all of these cases is a one-way traversal of the vehicle from point 1 to point n. If any of the (hard) deadlines in the first three problems is violated during the traversal, then the problem is infeasible. All cases can be solved in O(n) time.

(2) As described in Section 2, problem $[ST|r|C_{max}]$ can be solved in $O(n^2)$ time and also belongs to P.

(3) Problem $[ST|r+d|\Sigma C_i]$ is NP-complete. The credit for this result must be attributed to a recent paper by Afrati et al. (1985), who, working independently, showed that the so-called Traveling Repairman Problem (TRP) is NP-complete. In the TRP, a repairman must repair m+n machines so as to minimize the sum of completion times. Repairman and machines are all located on a straight line, with m machines on the left of the repairman and n on his right. Repair times are zero. Afrati et al. (1985) showed that in the absence of deadlines this problem can be solved in O(mn) time, and that if "hard" deadlines are imposed for each machine, the TRP is NP-complete. They proved NP-completeness for the TRP by a rather involved transformation from 0-1 KNAPSACK. Kim (1985) easily showed that TRP->$[ST|r+d|\Sigma C_i]$ and hence the latter problem is NP-complete as well.

(4) The complexity status of all other problems on the straight line is thus far open. We have focused our efforts on two "key" problems, $[ST|r+d|C_{max}]$ and $[ST|r|\Sigma C_i]$, so far without success. We conjecture that both these

problems (and, as a result, their "descendants" $[ST|r+d|\emptyset]$, $[ST|r+sd|L_{max}]$, $[ST|r+sd|\Sigma T_i]$ and $[ST|r+sd|\Sigma U_i]$) are NP-complete.

(5) With the exception of $[SH|r+d|\Sigma C_i]$, which, as a generalization of $[ST|r+d|\Sigma C_i]$ is also NP-complete, all other problems on the shoreline (R = SH) are open. This is true even for those variants whose straight line counterparts are trivial (that is, the $[SH|d|\emptyset]$, $[SH|d|C_{max}]$, $[SH|d|\Sigma C_i]$, $[SH|sd|C_{max}]$, $[SH|sd|\Sigma T_i]$ and $[SH|sd|\Sigma U_i]$ cases), and for the $[SH|r|C_{max}]$ case.

5. 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.

Concerning the computational performance of the heuristics, the dynamic programming algorithm, while not optimal for the shoreline structure, seems to consistently perform very well.

The analysis in Section 4 has shown that this class of problems is on the borderline between problems in P and NP-complete problems. As with machine scheduling problems, the resolution of exactly which of the problems

left open in Section 4 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 that many more extensions such as the multi-vehicle case can 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 the routing and scheduling of vehicles with time windows (in itself an already rapidly growing area).

On several occasions in this paper we drew an analogy between this class of problems and the class of single-machine scheduling problems (identify the machine with the vehicle and the jobs with the pickup points). With this identification, the shoreline problem lies some where between (i) a general machine scheduling problem with arbitrary sequence-dependent processing times t_{ij} and (ii) a specialized, but important version with sequence-independent processing times (with t_{ij} independent of j). The analogy between single-machine scheduling and shoreline routing so far has been restricted to the classification structure of these problems in terms of their objective function and constraints, and we have seen that indeed there are many resemblances. A more substantive question is whether there is any direct connection between these two classes of problems (particularly machine scheduling with sequence-independent processing times) in terms of computational complexity. This connection (if it exists) could conceivably be very useful in increasing our knowledge about the complexity status of our problems. We have tried to discover such connections, mainly with the better-understood class of straight-line problems, but without success. It is inter-

esting to note, however, that for all straight-line problems that are either NP-complete or open, their single-machine, sequence-independent processing time scheduling counterparts are NP-complete, and for all single-machine sequence-independent processing time scheduling cases in P, their straight-line counterparts are in P as well. These facts seem to indicate that the straight-line class is probably at least as hard as single-machine scheduling problems with sequence-independent processing times. There is, however, one case that does not obey this "rule". Whereas the computational complexity of the problem of minimizing total job tardiness on a single machine is still open, (Lenstra et al. 1977, Garey and Johnson 1979), the corresponding straight-line problem $[ST|sd|\Sigma T_i]$ can be solved in $O(n)$ time.

Even tougher seems to be the resolution of the complexity status of the more general "shoreline" case. In particular, the $[SH|r|C_{max}]$ case lies "between" two problems, one of which is in P (the $[ST|r|C_{max}]$ case) and the other NP-complete (this is the $[Euclidean|r|C_{max}]$ case, which is NP-complete even if $r_i = 0$ for all i (Papadimitriou 1977)). The "shoreline" restriction (which amounts to the extra requirement that $[t_{ij}]$ satisfies $t_{ik} \leq t_{ij}$ and $t_{kj} \leq t_{ij}$ for all $1 \leq i < k < j \leq n$) is strong enough to "soften" a problem from NP-complete to trivial in the absence of release times. Whether the problems remain "easy" when release times (or other time constraints) are imposed, remains to be seen.

ACKNOWLEDGMENTS

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 Uiversity's Research and Scholarship Development Fund, and by Grant # ECS-83-16224 of the National Science Foundation.

		ZOOM	GREEDY	API	TRAVERSE	MAXEPT
n=8*	R_{max} = 7	1.02	1.04	<u>1.01</u>	1.02	1.21
	14	<u>1.02</u>	1.13	1.19	1.04	1.34
	21	1.07	<u>1.06</u>	1.17	1.09	1.39
	28	1.06	<u>1.02</u>	1.24	1.17	1.41
	35	1.04	<u>1.01</u>	1.15	1.07	1.32
n=10**	R_{max} = 7	1.18	1.33	<u>1.08</u>	1.23	1.54
	14	<u>1.44</u>	1.74	1.85	1.54	2.00
	21	<u>1.19</u>	1.32	1.41	1.25	1.67
	28	<u>1.09</u>	1.14	1.38	1.25	1.51
	35	<u>1.07</u>	1.11	1.23	1.15	1.40
n=20**	R_{max} = 7	1.36	1.52	<u>1.22</u>	1.32	1.73
	14	<u>1.54</u>	1.83	1.78	1.63	2.00
	21	<u>1.24</u>	1.50	1.53	1.30	1.76
	28	<u>1.21</u>	1.33	1.41	1.28	1.58
	35	<u>1.09</u>	1.13	1.26	1.17	1.48
n=30**	R_{max} = 7	1.26	1.50	<u>1.17</u>	1.26	1.58
	14	<u>1.46</u>	1.74	1.66	1.54	2.03
	21	<u>1.42</u>	1.58	1.67	1.48	1.86
	28	<u>1.29</u>	1.38	1.46	1.34	1.70
	35	<u>1.22</u>	1.28	1.33	1.29	1.56

*Each number represents the average ratio, over 10 random problem instances, of the objective value obtained by the heuristics divided by the optimal value.

**Each number represents the average ratio, over 10 random problem instances, of the objective value obtained by the heuristics divided by the lower bound max {S, r*}.

Underlined numbers designate the minimum entry in each row.

Table 3.1

Average Performance of Heuristics

REFERENCES

AFRATI, F., S. COSMADAKIS, C. PAPADIMITRIOU, G. PAPAGEORGIOU, and N. PAPAKOSTANTINOU. 1985. The Complexity of the Travelling Repairman Problem. Working Paper, National Technical University of Athens.

BAKER, E. 1983. An Exact Algorithm for the Time-Constrained Traveling Salesman Problem. Operations Research 31, 938-945.

BAKER, E. and J. SCHAFFER. 1984. Solution Improvement Heuristics for the Vehicle and Scheduling Problem with Time Window Constraints. Working Paper, University of Miami.

BODIN, L., B. GOLDEN, A. ASSAD, and M. BALL. 1983. Routing and Scheduling of Vehicles and Crews: The State of the Art. Computers and Operations Research 10, 62-212.

CHRISTOFIDES, N., A. MINGOZZI, and P. TOTH. 1981. State-Space Relaxation Procedures for the Computation of Bounds to Routing Problems, Networks, Vol. 11, No. 2.

DESROSIERS, J., F. SOUMIS, and M. DESROCHERS. 1983. Routing with Time Windows: Synthesis. Working Paper G-83-05, HEC, Montreal.

DESROSIERS, J., Y. DUMAS, and F. SOUMIS. 1985. A Dynamic Programming Method for the Large-Scale Single Vehicle Dial-A-Ride Problem with Time Windows. Working Paper 84-12, HEC, Montreal

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

GAREY, M., and D. JOHNSON. 1975. Complexity Results for Multiprocessor Scheduling Under Resource Constraints. SIAM Journal of Computing, 4, 397-411.

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

GRAHAM, R., E. LAWLER, J. LENSTRA, and A. RINNOOY KAN, 1979. Optimization and Approximation in Deterministic Sequencing and Scheduling: A Survey. Discrete Optimization II, Annals of Discrete Mathematics 5, 287-326. North-Holand Publishing Company.

JAW, J., A. ODONI, H. PSARAFTIS, and N. WILSON. 1984. A Heuristic Algorithm for the Multi-Vehicle Advance-Request Dial-A-Ride Problem with Time Windows. Transportation Research, forthcoming.

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

LENSTRA, J., A. RINNOOY KAN, and P. BUCKER. 1977. Complexity of Machine Scheduling Problems Studies in Integer Programming, Annals of Discrete Mathematics, 1, 343-362, North-Holand Publishing Company.

PSARAFTIS, H. 1983. An Exact Algorithm for the Single Vehicle Dial-A-Ride Problem with Time Windows. Transportation Science 17, 351-357.

PSARAFTIS, H., J. ORLIN, B. BIENSTOCK, and, P. THOMPSON. 1985. Analysis and Solution Algorithms of Sealift Routing and Scheduling Problems: Final Report. Working Paper, No. 1700-85, Sloan School of Management, M.I.T.

RONEN, D. 1983. A Review of Cargo Ships Routing and Scheduling Models, Euro. Journal of Oper. Res., 12.

SEXTON, T. and L. BODIN. 1983A. Optimizing Single Vehicle Many-to-Many Operations with Desired Delivery Times: I, Scheduling, Transportation Science (forthcoming).

SEXTON, T. and L. BODIN. 1983B. Optimizing Single Vehicle Many-to-Many Operations with Desired Delivery Times: II, Routing, Transportation Science, forthcoming.

SOLOMON, M., 1983. Algorithms for the Vehicle Routing and Scheduling Problem with Time Window Constraints. Operations Research, forthcoming.

SOLOMON, M. 1984a. On the Worst-Case Performance of Some Heuristics for the Vehicle Routing and Scheduling Problem with Time Window Constraints. Networks, forthcoming.

SOLOMON, M. 1984b. The Minimum Spanning Tree Problem with Time Window Constraints. American Journal of Mathematical and Management Sciences, forthcoming.

SOLOMON, M., E. BAKER, and J. SCHAFFER. 1986. Accelerated Branch Exchange Procedures for the Vehicle Routing and Scheduling Problem with Time Windows, Working Paper, Northeastern University.

LIST OF FIGURE CAPTIONS

Figure	Caption
1.1	Typical shoreline instances.
2.1	Straight line case: a typical schedule.
2.2	Definition of BW_i in "TRAVERSE" heuristic.
2.3	Sets S_1, S_2 and S_3 for the straight-line case (tour version).
2.4	The dashed line schedule can replace the solid line schedules with no additional delay.
3.1	The Euclidean shoreline configuration with the conjectured maximum ratio S/L.
4.1	Relationships among problem variants in this class. An arc from problem P' to problem P means that P' reduces to P. Bidirectional arcs denote equivalence. Numbers on certain arcs refer to reduction proofs in the text. For the R=ST case (only), the symbols *(in P), !(NP-complete), and ?(open) display the complexity status of the problem.

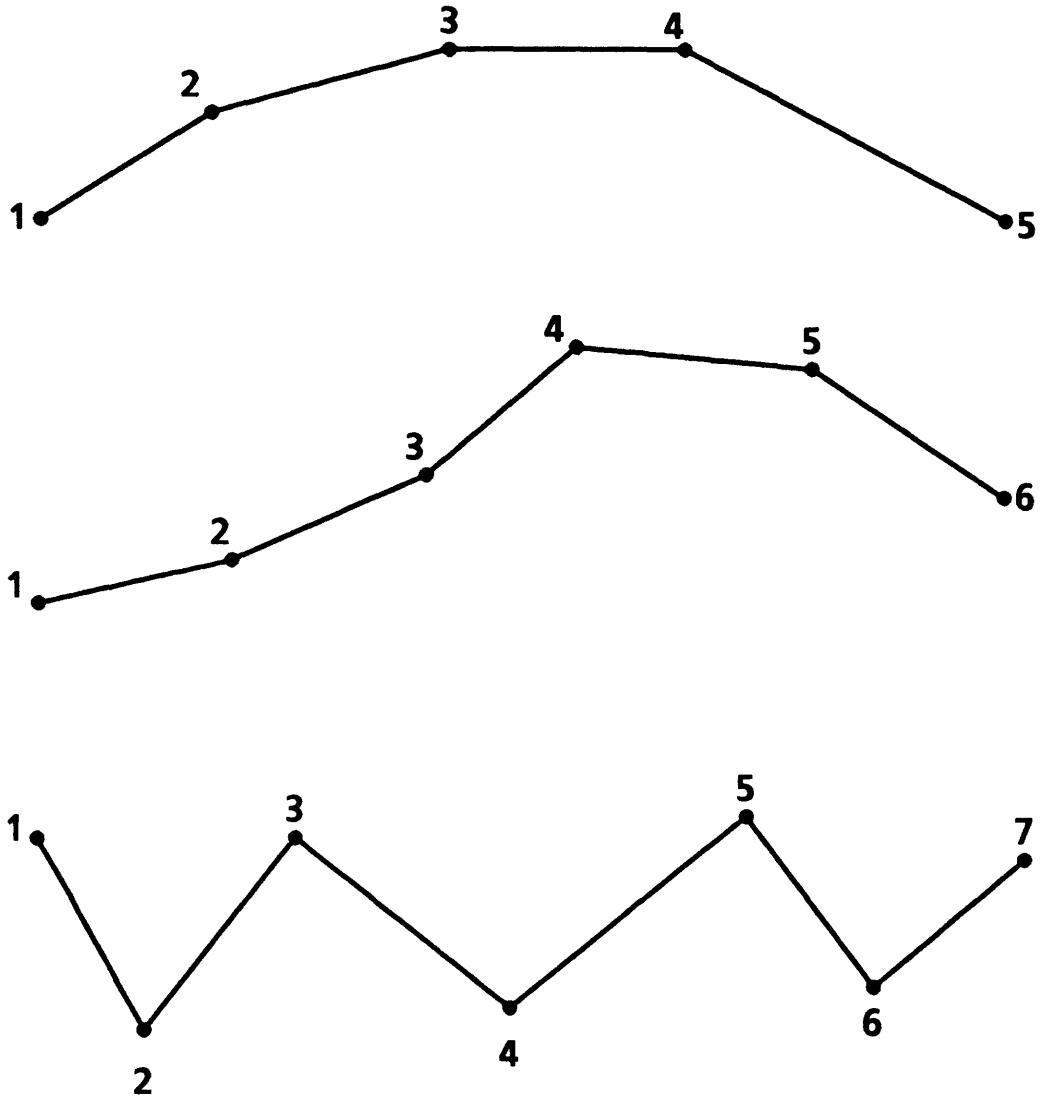


Figure 1.1. Typical Shoreline Instances

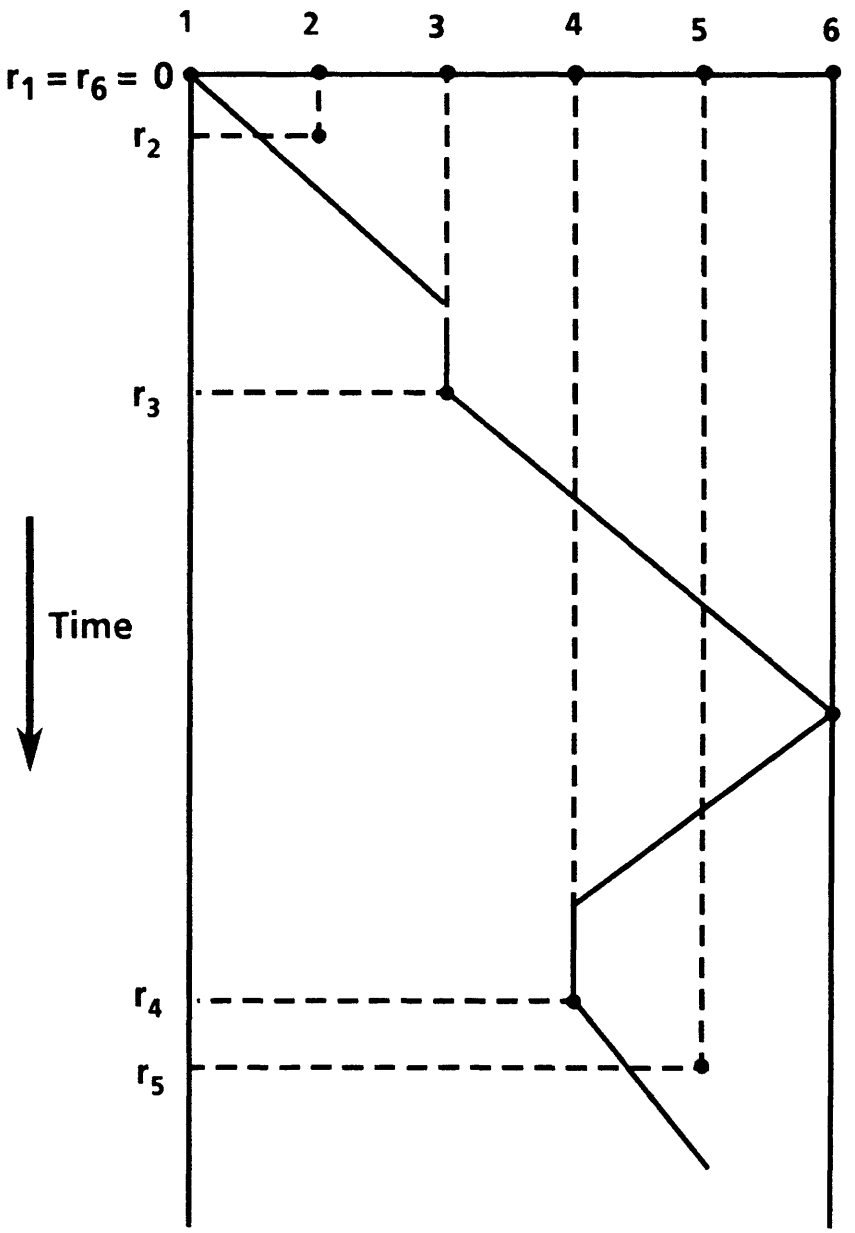


Figure 2.1. Straight Line Case: A Typical Schedule

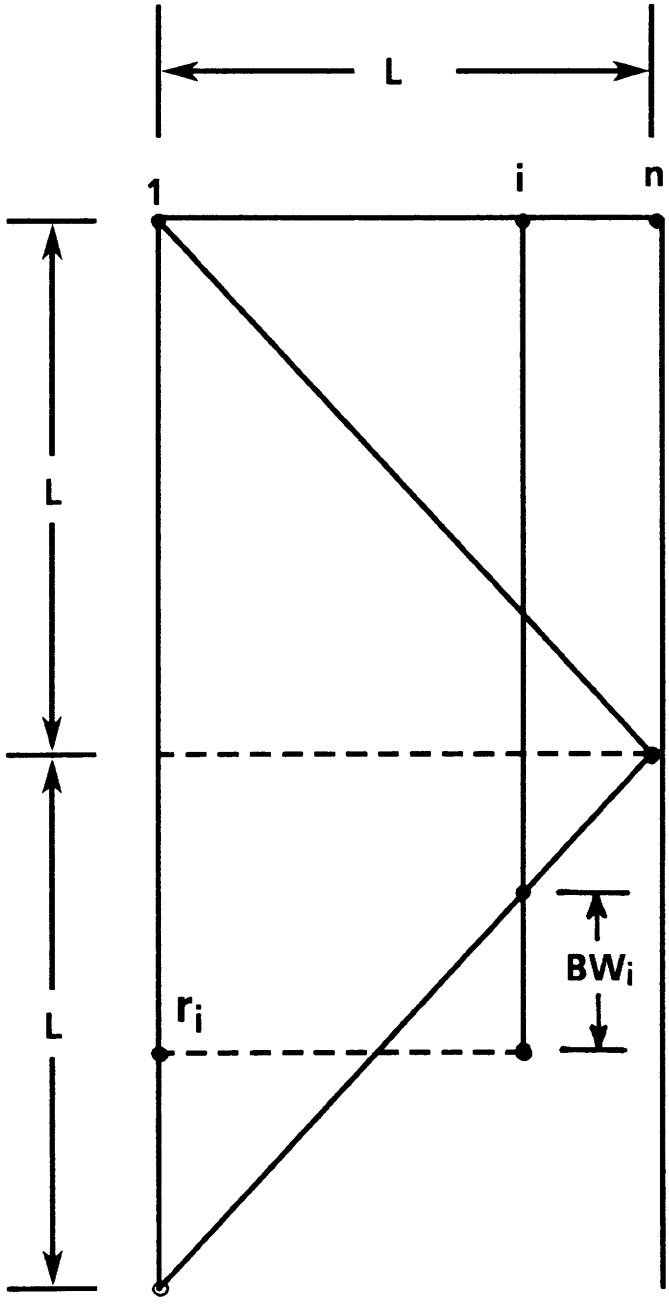


Figure 2.2. Definition of BW_i in "TRAVERSE" Heuristic

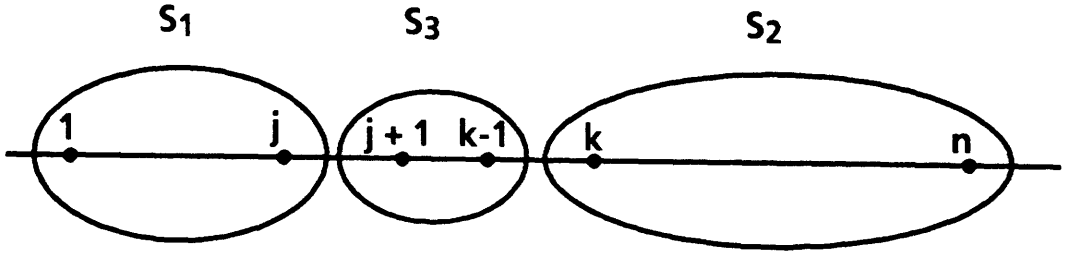


Figure 2.3. Sets S_1, S_2 and S_3 for the Straight-line Case (tour version)

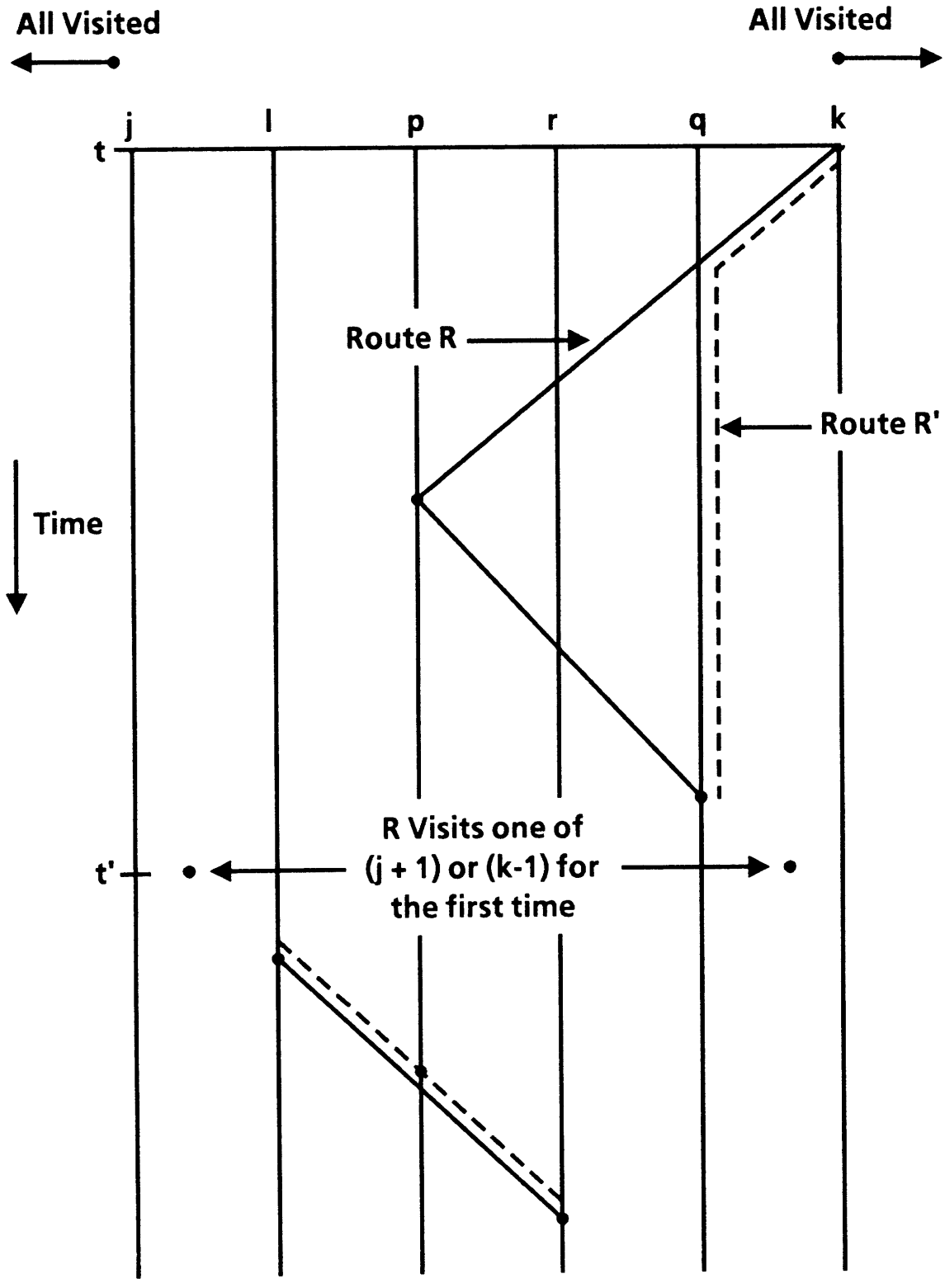


Figure 2.4. The Dashed Line Schedule Can Replace the Solid Line Schedule with No Additional Delay

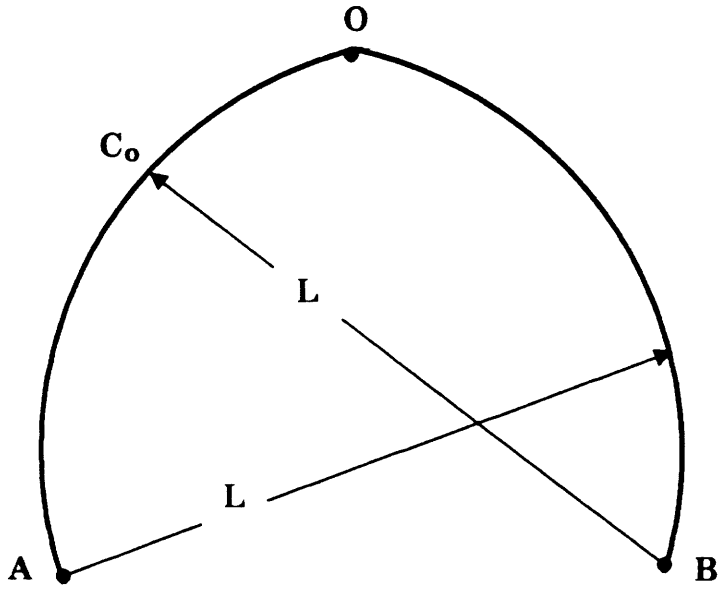


Figure 3.1. The Euclidean Shoreline Configuration with the Maximum Ratio S/L

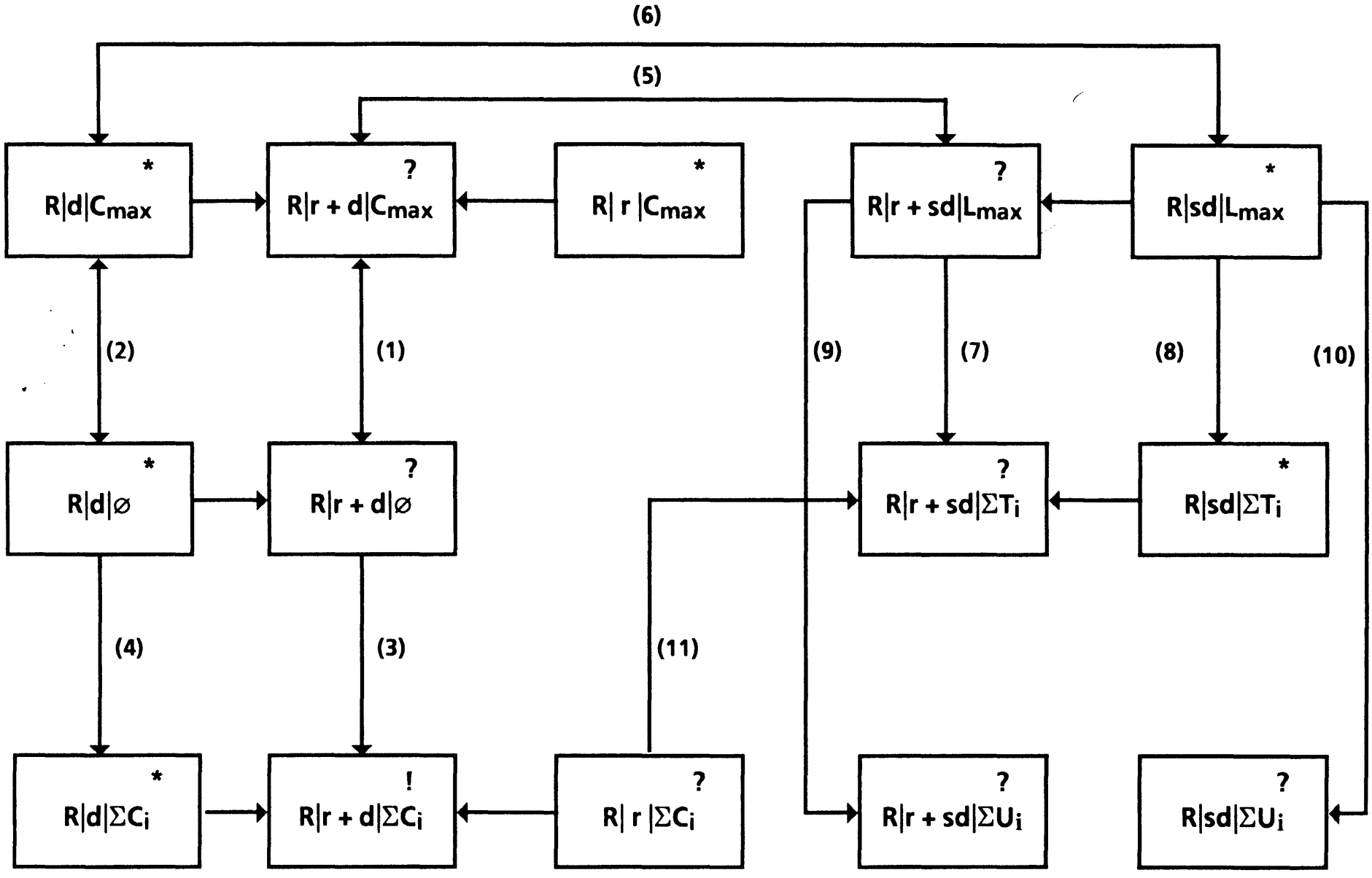


Figure 4.1. Relationships among problem variants in this class. An arc from problem P' to problem P means that P' reduces to P. Bidirectional arcs denote equivalence. Numbers on arcs refer to reduction proofs in the text. For the R = ST case (only), the sumbols *(in P), !(NP-complete), and ?(open) specify the complexity status of the problem.