J. Opl Res. Soc. Vol. 43, No. 6, pp. 629-635
Printed in Great Britain.

0160-5682/92 $3.50+0.00

The Orienteering Problem with Time Windows

MARISA G. KANTOR[1] and MOSHE B. ROSENWEIN[2]
[1]Princeton University, USA and [2]AT&T Bell Laboratories, Holmdel, NJ, USA

The orienteering problem with time windows, denoted by OPTW, belongs to a class of routeing and scheduling problems that arise in physical distribution. It may be modelled as a problem on a graph. It considers a set of nodes (customers), each with an associated profit and service duration (time window), and a set of arcs, each with an associated travel time. The objective of the problem is to construct an acyclic path beginning at a specified origin and ending at a specified destination that maximizes the total profit while observing time window constraints on all nodes and not exceeding a designated time limit. The problem is classified as *NP*-hard and, thus, an exact algorithm that executes in reasonable computational time is unlikely to exist. Since the problem is highly-constrained, we were able to develop a heuristic (referred to as the 'tree' heuristic) based upon an exhaustive search of the feasible solution space. The tree heuristic systematically generates a list of feasible paths and then selects the most profitable path from the list. In comparison with an insertion heuristic, the tree heuristic was found to produce improved values of total profit for heavily-constrained, modest-sized problems with reasonable computational effort.

Key words: distribution, heuristics, travelling salesman, orienteering problem

INTRODUCTION

The orienteering problem with time windows, denoted by OPTW, belongs to a class of routeing and scheduling problems that arise in physical distribution. OPTW may be modelled as a problem on a graph consisting of a set of nodes and a set of arcs. A non-negative score and a time window are associated with each node. A positive weight corresponding to travel time is associated with each arc. The objective of the OPTW is to construct a path with maximum total score beginning at a specific origin and ending at a specific destination such that (1) no node is included in the path more than once; (2) the time duration, or 'length', of the path does not exceed a specific upper bound; and (3) time window constraints on the nodes are observed, i.e. a node may only be 'visited' during specified time intervals.

The unconstrained orienteering problem (OP) was introduced by Tsiligrides[1] and represents an extension of the well-known travelling salesman problem (TSP). Leifer and Rosenwein[2] provide an extensive review of the OP research. The objective of the TSP is to construct a minimum-cost path that includes all nodes in a particular graph. In contrast, OP seeks a path with maximum total score subject to a constraint on the duration of the path. All nodes are not necessarily contained in the path. The significance of the OP to physical distribution was demonstrated by Golden *et al*[3]. They considered a firm which must service a set of customers, each with an associated profit. Limited resources (e.g. vehicles) however, restrict the number of customers that can be serviced and force the firm to choose a subset of customers (ideally, the most profitable) to visit.

Time-window constrained routeing (see a survey by Solomon and Desrosiers[4]) arises in bank and postal deliveries, industrial refuse collection, dial-a-ride services, and school bus routeing. In these problems, each customer must be serviced between a specific earliest time and a specific latest time. The duration of time that a customer may be visited is called a time window.

OPTW combines OP and time windows. OPTW may arise in a distribution firm which must select the most profitable subset of customers to service subject to restrictions on the maximum amount of time that delivery resources are available. In addition, the firm must schedule its routes to accommodate customer time windows. OPTW is a complex combinatorial problem, for three types of decisions are involved: (1) allocation of customers to be serviced; (2) routeing or sequencing of customers that are to be serviced; and (3) timing or scheduling of customer deliveries. This paper

Correspondence: M Rosenwein, Room 3K-328, AT&T Laboratories, Crawfords Corner Road, Holmdel, New Jersey 07733, USA

represents the initial investigation of the OPTW. It develops and reports on several solution algorithms for the problem.

OPTW falls into a category of problems described as *NP*-hard. (See Garey and Johnson[5] for a rigorous treatment of computational complexity of combinatorial problems). A polynomial-time algorithm that obtains optimal solutions is unlikely to exist. Researchers have therefore resorted to heuristics to solve such problems. These heuristics, typically, only consider a small fraction of the feasible solution space. The large number of constraints of the OPTW imply that the feasible solution space is smaller than many other NP-hard problems. Thus, a heuristic, based on an exhaustive search of the solution space, seemed practical for the OPTW. Because an exhaustive search is generally implemented with a tree, the proposed heuristic for the OPTW is referred to as the tree heuristic.

We compared the performance of the tree heuristic with an insertion heuristic. The insertion heuristic developed for the OPTW is an extension of the insertion heuristic developed by Laporte and Martello[6] for OP. An insertion heuristic constructs a path by iteratively inserting a previously unsequenced node into a route in the 'best' slot. Insertion heuristics are popular for solving routeing and scheduling problems, e.g. see Solomon[7] on the vehicle routeing problem with time windows. In comparison with a tree search-based algorithm, an insertion procedure considers a much smaller portion of the feasible solution space. Thus, insertion heuristics tend to have shorter computational running times in comparison with tree searches but generate solutions that are, typically, inferior. Experimentation showed that the tree heuristic obtains improved values of total score for the OPTW, over the insertion heuristic. The tree heuristic's computational performance was acceptable for heavily-constrained, modest-sized problems.

PROBLEM DESCRIPTION

The following notation is needed to state formally the problem.

Notation

$N = \{1, 2, \ldots, n\}$ represents a set of nodes.

$A = \{(i,j) \mid i,j \in N\}$ represents a set of arcs.

$T_{\max}$ = time allotted to complete the path.

e_j = earliest available time of node j.

d_j = latest available time of node j.

t_{ij} = time needed to travel from i to j, representing the length of arc (i,j).

s_j = score associated with node j.

$$y_j = \begin{cases} 1, & \text{if node } j \text{ is visited} \\ 0, & \text{otherwise.} \end{cases}$$

$$x_{ij} = \begin{cases} 1, & \text{if arc } (i,j) \text{ is traversed} \\ 0, & \text{otherwise.} \end{cases}$$

The OP involves the construction of a path $P \subseteq N$ beginning at a specified origin, node 1, and ending at a specified destination, node n, such that $\Sigma_{j \in N} s_j y_j$ is maximized and $\Sigma_{(i,j) \in A} t_{ij} x_{ij} \leqslant T_{\max}$. In OPTW, node j may only be visited during the interval $[e_j, d_j]$, its specified time window. (We implicitly assume that at most one time window is associated with each node. In practice, multiple time windows may be associated with a node. The heuristics, described below, may be easily extended to the more general case.) The time windows e_1 and e_n are set to 0, and d_1 and d_n are set to $T_{\max}$. We will use the notation $p(k)$ to refer to the kth element in P, e.g. $p(1) = 1$, $p(|P|) = n$. The notation $|\cdot|$ denotes the cardinality of set $\cdot$.

If i and j are consecutive nodes in P, i.e. $i = p(k), j = p(k+1)$, for $k = 1, 2, \ldots, |P| - 1$, then the time, b_j, at which node j is serviced is given by $b_j = \max\{b_i + t_{ij}, e_j\}$. Furthermore, the waiting time w_j is given by $w_j = \max\{e_j - b_i - t_{ij}, 0\}$.

STATEMENT OF ALGORITHMS

Although a mathematical program may be stated to model OPTW, an exact solution for meaningful values of n is unlikely to be obtained in reasonable computational time. Thus, we resort to heuristics to solve OPTW.

Insertion algorithm: description

The insertion algorithm developed for the OPTW constructs a path P from 1 to n by iteratively inserting a node $j \in N - P$ into the best available slot in the route. The node j to be inserted between consecutive nodes u and v is selected by computing ρ_{juv}, the ratio of j's score, s_j, to the additional travel time incurred by j's inclusion in the path. This criterion for selecting a node j to insert is identical to the criterion used by Laporte and Martello[6] in their version of an insertion heuristic for the OP. The insertion of node j between nodes u and v must satisfy the condition that all nodes that are visited after v can be serviced before their latest times.

Insertion algorithm: formal statement

Initialize $P = \{1, n\}, e_1 = e_n = 0, d_1 = d_n = T_{\max}$.
Do Until ($N-P$ is empty)
Compute $\rho_{juv} = s_j/(t_{uj} + t_{jv} - t_{uv})$, for every $j \notin P$, for every pair of nodes, $u = p(k)$, $v = p(k+1)$, and for every $k = 1, 2, \ldots, |P| - 1$.
Let $(\bar{j}, \bar{u}, \bar{v})$ correspond to max $\{\rho_{juv}\}$ subject to:

(1) $\sum_{(i,j)\in P} t_{ij} + t_{uj} + t_{jv} - t_{uv} \leqslant T_{\max}$,

(2) $b_u + t_{uj} \leqslant d_j$,
(3) the insertion of j does not cause any constraint $b_i \leqslant d_i$ to become violated, for every $i \in N$ corresponding to positions $k+1, k+2, \ldots, |P|$ in P.

If no such triple $(\bar{j}, \bar{u}, \bar{v})$ exists, then STOP. Else, insert $\bar{j}$ between $\bar{u}$ and $\bar{v}$ in P.

Tree heuristic: description

If the problem constraints,

$$\sum_{(i,j)\in A} t_{ij}x_{ij} \leqslant T_{\max}$$

and

$$e_j \leqslant b_j \leqslant d_j, \qquad \text{for every } j \in N,$$

are sufficiently tight, and if $|N|$ is not large, then the feasible solution space of the OPTW is greatly reduced. Thus, we developed a heuristic that is based upon generating a reasonably large number of feasible routes and then choosing the route with the greatest total score. In order to generate formally these routes, we developed a tree-search procedure. A tree search systematically constructs partial paths beginning with node 1. A subset of nodes $j \in N - \bar{P}$ is added to a partial path $\bar{P}$ until either (i) $\bar{P}$ is abandoned; or (ii) node n is added to $\bar{P}$ to form a complete path P. The route generation process abandons partial routes that are infeasible or that are unlikely to yield the best total score.

A series of tests are performed prior to augmenting a partial path. Some additional notation is

required. Let $\bar{j} \in N - \bar{P}$ denote a candidate node that may be added to $\bar{P}$, $\bar{j} \neq n$, and let $\bar{i}$ denote the last node added to $\bar{P}$. The algorithm requires the following parameters. Let $W_{\max}$ denote the maximum waiting time allowed at any orienteering node, and let $V_{\max}$ denote the maximum allowable length of any orienteering arc. At any iteration, let S^* denote the best known total score to date in the search. Also, let $0 < \alpha, \beta < 1$.

In order to augment the partial path $\bar{P}$ with node $\bar{j}$, the following rules must be satisfied.

R1: $\bar{j} \notin \bar{P}$ (a node may exist only once in a path).

R2: $b_{\bar{j}} \leqslant T_{\max}$ (path duration constraint).

R3: $b_{\bar{j}} \leqslant d_{\bar{j}}$ (time window constraint).

R4: $t_{\bar{i}\bar{j}} \leqslant V_{\max}$ (maximum allowable travel time between two nodes).

R5: $w_{\bar{j}} \leqslant W_{\max}$ (maximum allowable waiting time at a node).

R6: If $b_{\bar{j}} > \alpha T_{\max}$, then $\sum_{j \in \bar{p}} s_j y_j$ must be $\geqslant \beta S^*$. We require that $\bar{P}$ shows sufficient 'promise' of yielding an improved total score over the 'incumbent' best-known total score, otherwise, it is abandoned.

Observe that rules R1–R3 are based upon the stated constraints for the OPTW while rules R4–R6 are heuristics that tend to eliminate seemingly unattractive partial paths.

The tree heuristic is based upon a depth-first search which begins with a partial path $\bar{P} = \{1\}$ and adds nodes to $\bar{P}$ until either a complete path P is generated or until rules R1–R6 suggest that the 'current' partial path be abandoned. The best score is updated if a complete path P is generated, and its total score exceeds the best-known score, achieved previously.

Some notation is used in the formal statement of the algorithm. Let k denote the depth of the tree, and let $S(\bar{P}) = \Sigma_{j \in \bar{P}} s_j$, i.e. the total score of the partial path $\bar{P}$.

Tree heuristic: formal statement

Step 0: Initialize $\bar{P} = \{1\}$, $\bar{i} = 1$, $k = 1$, and $S(\bar{P}) = s_1 = S^*$.
Step 1: Compose a list $\bar{J}_k \subseteq N - n$ of all nodes that satisfy rules R1–R5, i.e. nodes that may be added to $\bar{P}$. All nodes $\bar{j} \in \bar{J}_k$ are initially unmarked.
Step 2: Get the next, unmarked element $\bar{j} \in \bar{J}_k$. If none exists, go to Step 4.
Step 3: $\bar{P} \leftarrow (\bar{P}, \bar{j})$. Mark $\bar{j}$. Update the total score of $\bar{P}$, i.e. $S(\bar{P}) = S(\bar{P}) + s_{\bar{j}}$, update the 'last' node added to $\bar{P}$, i.e. $\bar{i} = \bar{j}$, and update the depth of the tree, i.e. $k = k + 1$. If R6 is satisfied, go to Step 1. Else, go to Step 5.
Step 4: $\bar{P}$ may not be augmented with a node $\bar{j} \in \bar{J}_k$. If $\bar{P}$ may be augmented with node n and if $S(\bar{P}) + s_n > S^*$, update $S^* = S(\bar{P}) + s_n$, i.e. a new path is found that yields an improved total score.
Step 5: Backtrack to the previous level of the tree. Update the tree-depth index, i.e. $k = k - 1$, adjust the total score, i.e. $S(\bar{P}) - S(\bar{P}) - s_{\bar{j}}$, and update the index to the 'last' node in $\bar{P}$, i.e. $\bar{i} = p(k)$.
Step 6: Termination test. If $k = 0$, STOP. Else, go to Step 2.

COMPUTATIONAL EXPERIENCE

Both the insertion and tree heuristics were programmed in Turbo Pascal on a Compaq Portable II computer. Geographical data, i.e. node coordinates, and profits for the OPTW were obtained from Tsiligrides'[1] data for the OP. Travel times were equated with the geometric distances between nodes. The time windows were generated as follows. For every $j \in N - \{1, n\}$, e_j is a uniformly distributed random number over the interval $[0, T_{\max}/2]$, and d_j is a (truncated) uniformly distributed random number over the interval $[e_j, \min(e_j + 2/3T_{\max}, T_{\max})]$. In our

computational results, T_{max} is varied, as in previous empirical research on the OP. Since the time windows are functions of T_{max}, they, too, are varied for each run.

In addition to varying T_{max}, the tree heuristic was tested with various values for the parameters W_{max}, V_{max}, α, and β, employed in Rules R4–R6. The parameters V_{max} and W_{max} were set as follows. The t_{ij} were sorted in non-decreasing order, and a 'cut-off' value t_q was specified. Arcs with $t_{ij} > t_q$ were not considered in augmenting a partial path $\bar{P}$. Similarly, all arcs that would cause the waiting time at a node to exceed t_q were not considered. The value of t_q was chosen by selecting a critical percentile, q, of the sorted, non-decreasing list of t_{ij}. Thus, V_{max} and W_{max} are functions of q.

Tables 1 and 2 present our computational results for the Tsiligrides' data with $n = 21$. As illustrated in Figure 1, this problem is of interest as nodes close to the origin and destination (nodes 1 and 21, respectively) have low values of s_j while nodes far from nodes 1 and 21 have large values of s_j. Although many values for the tree heuristic's parameters were tested, we selected three sets of parameters to present.

TABLE 1. *Total Scores*

T_{max}	Insertion	Tree I	Tree II	Tree III
15	60	90	90	90
20	95	120	140	120
23	140	150	155	150
25	110	150	150	150
27	140	200	200	200
30	180	200	210	200
32	215	215	215	215
35	150	170	210	200
38	170	220	260	230
40	165	220	245	235
45	165	230	280	230

TABLE 2. *CPU Times (hh:mm:ss)*

T_{max}	Insertion	Tree I	Tree II	Tree III
15	1:46	0:23	0:41	0:25
20	1:40	0:33	2:21	0:44
23	1:17	0:39	2:52	1:06
25	1:15	1:11	6:06	1:37
27	1:12	2:17	49:41	4:16
30	1:08	7:10	47:15	2:12
32	1:03	19:39	1:11:01	45:45
35	1:13	2:39	46:57	10:45
38	1:88	24:26	5:38:25	1:44:58
40	1:11	13:46	1:51:59	51:44
45	1:08	14:17	3:48:08	58:32

$$\text{Tree I:} \quad \alpha = \beta = 0.25, \quad q = 0.35.$$

$$\text{Tree II:} \quad \alpha = \beta = 0.50, \quad q = 0.35.$$

$$\text{Tree III:} \quad \alpha = \beta = 0.25, \quad q = 0.40.$$

As α, β, and q, increase, the number of partial paths that are considered increases, e.g. Rule R6 does not apply for a particular $\bar{j}$ until $b_{\bar{j}} > \alpha T_{max}$.

In our testing, we ran the insertion heuristic prior to running the tree heuristic. Thus, we were able to initialize S^* with the value of total score obtained by the insertion heuristic. In Table 1, we compare the total score obtained by the insertion heuristic with the total score obtained by various runs with the tree heuristic. On average, Tree I improves by 24.4% on the total score obtained by Insertion, Tree II improves by 35.4% on the total score obtained by Insertion, and Tree III improves by 29.2% on the total score obtained by Insertion. Total score does not monotonically increase as T_{max} increases, for time windows are present. As illustrated in Table 2, the improve-

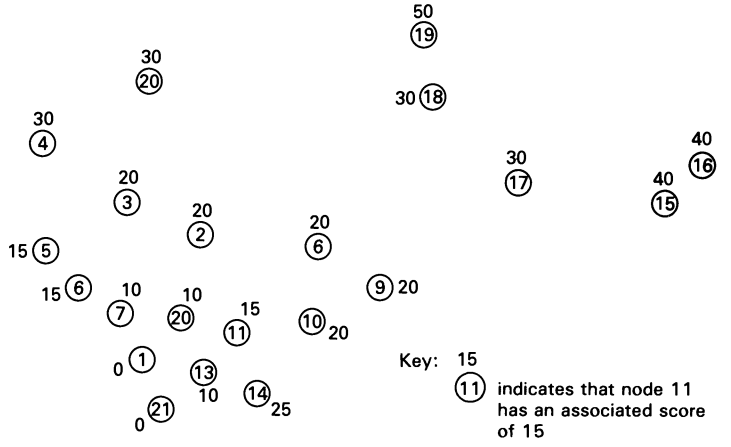

FIG. 1. *Graphical distributon of nodes and associated scores*

ment in total score obtained by the tree heuristic is coupled with an increase in CPU running times. In the tree heuristic, as in similar search procedures, e.g. see results for branch-and-bound in Rosenwein,[8] the timing of a discovery of a strong incumbent solution can greatly affect the duration of the search. Thus, we cannot expect a monotonic behaviour of CPU times with respect to T_{max} in Table 2.

Tree II obtains the best value of total score since it considers a larger set of partial paths. However, it may not be amenable to a personal computing environment, particularly for large values of T_{max}. Thus, for a personal computing environment, the parameter values used in Tree I or Tree III should be selected, as these scenarios obtained significant improvements in total score with reasonable computational effort. If the tree algorithm was executed on a more powerful personal computer or a large main-frame computer, significant reductions in running times would result, and the parameter values used in Tree II could be selected.

The tree heuristic proved to be effective for modest-sized, constrained problems. As T_{max} increases and/or as the duration of the time windows increases, the tree heuristic encounters difficulties. We discovered that for Tsiligrides' 33 node problem, for $T_{max} > 45$, the tree heuristic could not uncover a solution with reasonable computational effort, e.g. less than two hours. Thus, the tree heuristic is not effective for larger problems if executed on a small personal computer. Although the tree heuristic was only successful in solving problems with $n = 21$ on a small personal computer, this problem class is, nevertheless, significant. For example, in the vehicle routeing work of Fisher and Jaikumar[9], the average number of customers assigned to a vehicle was between 10–15. Thus, an ability to solve the OPTW with 21 customers may be meaningful in practice.

FUTURE RESEARCH

Future research, conducted with more powerful hardware, may illustrate that the tree search solution approach is reasonable for problems with larger value of n. Also, fewer arcs and partial paths would need to be excluded in such an environment, thus potentially increasing the solution quality, i.e. increased values of total score, of the tree heuristic for modest-size n, e.g. $n = 21$. Future research may also experiment with different insertion-based heuristics. Since the tree heuristic initializes S^* according to the value obtained by the insertion heuristic and since Rule R6 is dependent on S^*, an improved initial value of S^* will enhance the performance of the tree heuristic.

REFERENCES

1. T. TSILIGRIDES (1984) Heuristic methods applied to orienteering. *J. Opl Res. Soc.* **35**, 797–809.
2. A. C. LEIFER and M. B. ROSENWEIN (1991) Strong linear programming relaxations for the orienteering problem. AT&T Technical Report, Holmdel, NJ.

3. B. L. GOLDEN, L. LEVY and R. VOHRA (1987) The orienteering problem. *Naval Res. Logist.* **34**, 307–318.
4. M. M. SOLOMON and J. DESROSIERS (1988) Time window constrained routing and scheduling problems. *Transportation Sci.* **22**, 1–22.
5. M. R. GAREY and D. S. JOHNSON (1979) *Computers and Intractibility; A Guide to the Theory of NP-Completeness.* Freeman, San Francisco, CA.
6. G. LAPORTE and S. MARTELLO (1990) The selective travelling salesman problem. *Discrete Appl. Math.* **26**, 193–207.
7. M. M. SOLOMON (1987) Algorithms for the vehicle routing and scheduling problem with time window constraints. *Opns Res.* **35**, 254–265.
8. M. B. ROSENWEIN (1991) An improved bounding procedure for the constrained assignment problem. *Computers and Opns Res.* **18**, 531–535.
9. M. L. FISHER and R. JAIKUMAR (1981) A generalized assignment heuristic for vehicle routing. *Networks* **11**, 109–124.