The m-Traveling Salesman Problem with Minmax Objective

Article *in* Transportation Science · August 1995

DOI: 10.1287/trsc.29.3.267

4 authors, including:

Paulo M. França
São Paulo State University
103 PUBLICATIONS **1,662** CITATIONS

SEE PROFILE

Michel Gendreau
Polytechnique Montréal
477 PUBLICATIONS **23,428** CITATIONS

SEE PROFILE

Felipe M Müller
Universidade Federal de Santa Maria
59 PUBLICATIONS **351** CITATIONS

SEE PROFILE

The m-Traveling Salesman Problem with Minmax Objective[1]

PAULO M. FRANÇA

Faculdade de Engenharia Elétrica, Universidade Estadual de Campinas, 13081 Campinas-SP, Brazil

MICHEL GENDREAU

Centre de recherche sur les transports, Université de Montréal, Montréal, Canada H3C 3J7

GILBERT LAPORTE

Centre de recherche sur les transports, Université de Montréal, Montréal, Canada H3C 3J7

FELIPE M. MÜLLER

Departamento de Eletrônica e Computação, Universidade Federal de Santa Maria, Santa Maria-RS, Brazil

This article proposes algorithms for the Minmax version of the m-Traveling Salesman Problem in which the objective is to minimize the length of the longest route. A tabu search heuristic and two exact search schemes are developed. Problems involving up to 50 vertices are solved to optimality.

The *m-Traveling Salesman Problem* (m-TSP) arises naturally in a number of distribution contexts. The problem is defined on a graph $G = (V, A)$, where $V = \{v_0, \ldots, v_n\}$ is the vertex set, v_0 denotes a *depot*, and $A = \{(v_i, v_j): v_i, v_j \in V, i \neq j\}$ is the arc set. A matrix of non-negative *costs* or *distances* $C = (c_{ij})$ is associated with A. This matrix is symmetrical if and only if $c_{ij} = c_{ji}$ for all $(v_i, v_j) \in A$; it satisfies the triangle inequality if and only if $c_{ij} \leq c_{ik} + c_{kj}$ for all $(v_i, v_j), (v_i, v_k), (v_k, v_j) \in A$. There are m identical vehicles based at the depot. The m-TSP consists of determining m vehicle routes of least total cost, starting and ending at the depot, and such that every vertex $v \in V \setminus \{v_0\}$ is visited exactly once by one vehicle. The m-TSP can be readily transformed into a 1-TSP (or TSP) by introduction of artificial vertices as shown, for example, in LENSTRA and RINNOOY KAN,[17] but the resulting TSP is highly degenerate. Computational results obtained by LAPORTE and NOBERT[15] indicate that it is preferable to solve the m-TSP directly, without first transforming it into a TSP.

In the m-TSP, the aim is to minimize an *efficiency* or *Minsum* criterion, i.e., the total solution cost. There are, however, contexts where an *equity* or *Minmax* criterion is more appropriate, i.e., the cost of the most expensive route should be minimized. This occurs in situations where the m salesmen should be assigned delivery routes of similar lengths. GIUST[8] describes an example related with the distribution of gas in the province of Namur, Belgium. This is done by a small trucking firm that operates four vehicles. In this case, it is important to achieve an equitable distribution of workloads. Another application unrelated to transportation is the problem of scheduling n jobs on m identical parallel machines in order to minimize the makespan when processing times are sequence dependent (i.e., c_{ij} is equal to the changeover cost from job i to job j, plus the actual processing time of job j). Note that in scheduling problems the cost matrix is typically asymmetrical. The mathematical struc-

[1]Accepted by Mark S. Daskin.

Transportation Science
Vol. 29, No 3, August 1995

0041-1655/95/2903-0267 $01 25

ture of the sequence independent minimum makespan problem is really different. For recent papers on this problem, see DELL'AMICO and MARTELLO,[3] as well as FRANÇA et al.[4]

The Minmax m-TSP has received considerably less attention than its Minsum counterpart. It is strongly NP-hard as it includes the 1-TSP as a special case. FREDERICKSON, HECHT and KIM[5] provide approximate algorithms for the Minmax m-TSP, derived from corresponding TSP heuristics described in ROSENKRANTZ, STEARNS and LEWIS.[20] The first constructs m routes simultaneously by using a least cost insertion criterion or a nearest neighbour criterion. The authors show that when C satisfies the triangle inequality, the worst-case performance ratio of this heuristic is equal to 2 for the least cost insertion, and to $m + (m/2)\log n$ for the nearest neighbour. Another heuristic described in [5] consists of first obtaining a good TSP solution, and of splitting the tour into m subtours of more or less equal costs. The worst-case performance ratio of this heuristic is equal to $b + 1 - 1/m$, where b is a worst-case bound for the single TSP algorithm. Thus, if C is symmetric and satisfies the triangle inequality and CHRISTOFIDES'[2] heuristic is used to construct the TSP tour, b takes the value $3/2$ and the worst-case performance ratio for the Minmax m-TSP heuristic is therefore equal to $5/2 - 1/m$. There are no known worst-case results for cases where C is asymmetric or does not satisfy the triangle inequality, and no computational experience is reported on the empirical performance of these heuristics.

The purpose of this article is to describe a tabu search heuristic for the Minmax m-TSP, as well as two exact algorithms. These algorithms apply equally well to symmetric and asymmetric instances but, as pointed out in Section 2, we have tested the asymmetric case only. These are described in Sections 1 and 2, respectively. Computational results are reported in Section 3, and the conclusion follows in Section 4.

1. A TABU SEARCH HEURISTIC

TABU SEARCH is a global optimization metaheuristic first proposed by GLOVER.[9] Briefly, tabu search explores the solution space by repeatedly moving from a solution to the best of its neighbours. To escape local optima, the objective is allowed to deteriorate; to prevent cycling, solutions that were recently examined are temporarily forbidden and inserted in a constantly updated *tabu list*. Tabu search methods are open-ended and the key to efficient implementation is that they should be tailored to the characteristics of the problem at hand. In recent years, a number of tabu search heuristics have been successfully applied to a variety of *Vehicle Routing Problems* (VRPs) (see, e.g., OSMAN,[18] PUREZA and FRANÇA,[19] SEMET and TAILLARD,[21] TAILLARD,[23] GENDREAU, HERTZ and LAPORTE[7]). For general expository articles on tabu search, see GLOVER,[10,11] GLOVER and LAGUNA,[12] and GLOVER, TAILLARD and DE WERRA.[13]

The tabu search algorithm we propose for the Minmax m-TSP embeds the GENI insertion heuristic and the US post-optimization procedure developed by GENDREAU, HERTZ and LAPORTE for the TSP.[6] An initial solution is first constructed by means of GENI. Neighbour solutions are then explored by repeatedly removing a vertex from its current route, and inserting it using GENI into another route containing one of its closest neighbours. Finally, the US post-optimization procedure is applied to each route separately.

As shown in [6], GENI is less myopic and more powerful than standard insertion procedures in that every vertex may only be included in a route considering one of its p nearest neighbours, and every insertion is executed simultaneously with a local reoptimization of the current tour. Here is a short description of GENI, valid for symmetrical and asymmetrical problems. For a full description, the reader is referred to [6]. Consider the partially constructed tour $(v_0, v_1, \ldots, v_t, v_0)$ and define the two p-neighbourhoods of vertex v as follows. If v is not on the tour and $p \leqslant t$ or if v is on the tour and $p \leqslant t - 1$, then $N_p^-(v)$ is the set of the p closest predecessors of v already on the tour, and $N_p^+(v)$ is the set of the p closest successors of v already on the tour. If v is not on the tour and $p \geqslant t + 1$, then $N_p^-(v) = N_p^+(v) = \{v_0, v_1, \ldots, v_t\}$; if v is on the tour and $p \geqslant t$, then $N_p^-(v) = N_p^+(v) = \{v_0, v_1, \ldots, v_t\} \setminus \{v\}$. In symmetrical problems, one needs only define a single p-neighbourhood $N_p(v)$ for v since in this case $N_p^-(v) = N_p^+(v)$. Also denote by P_{ij} the set of vertices on the path from v_i to v_j for a given orientation of the tour. To define a generalized insertion, arbitrarily select a free vertex v as the next candidate for insertion and consider two types of insertion.

Type I: Select $v_r \in N_p^-(v)$, $v_s \in N_p^+(v)$ and $v_k \in N_p^+(v_{r+1}) \cap P_{sr} \setminus \{v_r, v_s\}$. Delete arcs (v, v_{r+1}), (v_s, v_{s+1}) and (v_k, v_{k+1}); insert arcs (v_r, v), (v, v_s), (v_{r+1}, v_k) and (v_{s+1}, v_{k+1}). Paths $(v_{r+1}, \ldots, v_s)$ and $(v_{s+1}, \ldots, v_k)$ are reversed.

Type II: Select $v_r \in N_p^-(v)$, $v_s \in N_p^+(v)$, $v_k \in N_p^+(v_{r+1}) \cap P_{sr} \setminus \{v_s, v_{s+1}\}$ and $v_l \in N_p^-(v_{s+1}) \cap P_{rs} \setminus \{v_r, v_{r+1}\}$. Delete arcs (v_r, v_{r+1}), (v_{l-1}, v_l),

(v_s, v_{s+1}) and (v_{k-1}, v_k); insert arcs (v_r, v), (v, v_s), (v_l, v_{s+1}), (v_{k-1}, v_{l-1}) and (v_{r+1}, v_k). Paths $(v_{r+1}, \ldots, v_{l-1})$ and $(v_l, \ldots, v_s)$ are reversed.

To determine the best move, it is necessary to compute the cost of the tour corresponding to each type of insertion, to each orientation of the tour (in the case of symmetrical problems), and to each possible choice of v_r, v_s, v_k, v_l. For an n-vertex TSP, GENI can be executed in $O(np^4 + n^2)$ operations. Note that the order in which vertices are selected for insertion in the partial tour does not seem to have any impact on the performance of GENI.

In the US post-optimization procedure, each vertex is in turn removed from the tour using the reverse GENI operation, and the vertex is then reinserted in the tour using GENI. The procedure ends when no further improvement can be obtained by removing and reinserting any vertex. Again, US has proved highly successful on the TSP.

We now describe how to adapt GENI for the Minmax m-TSP. This adaptation uses the concepts of *global neighbours* and *local neighbours*. Global neighbours are the q closest predecessors and the q closest successors of a vertex v, among all vertices of $V \setminus \{v\}$, were q is an input parameter. Each vertex also has $2m$ local neighbourhoods, two for each route. The local neighbourhood of a vertex $v \in V$ relative to route k is selected as the set of the p closest predecessors and the p closest successors among all the vertices already in that route, where p is an input parameter. The sets of local neighbours of a given vertex must be dynamically updated during the course of the algorithm. Figure 1 illustrates the global and local neighbourhoods of a vertex in a situation where all distances are symmetrical, in which case, the sets of closest predecessors and closest successors coincide. In the tabu search algorithm, the global neighbours of vertex determine the set of routes into which it may be inserted. For any of these routes, local neighbours are used to define the GENI insertions.

We now provide a step by step description of the heuristic. Some comments on the choice of parameters flow.

Phase 1: Initial Solution

Step 1. Determine the q global predecessors and the q global successors of each vertex v_i.

Step 2. Randomly select m vertices, and construct for each a return route between the vertex and the depot. Initialize the local neighbourhoods of each vertex with respect to each route.

Step 3. Consider in turn each unrouted vertex and using GENI with neighbourhood size p, insert it in a route that produces the least increase of the Minmax objective, while updating the lists of the local neighbours.

Phase 2: Neighbourhood Search

Step 1. Set the iteration count $t := 0$, set $\bar{z}$ equal to the length of the longest route and set $z^* := \bar{z}$. Record the current solution as the incumbent. Initially, no move is tabu.

Step 2. For each vertex v_i belonging to the longest route k, determine the set $R(v_i)$ of candidate routes

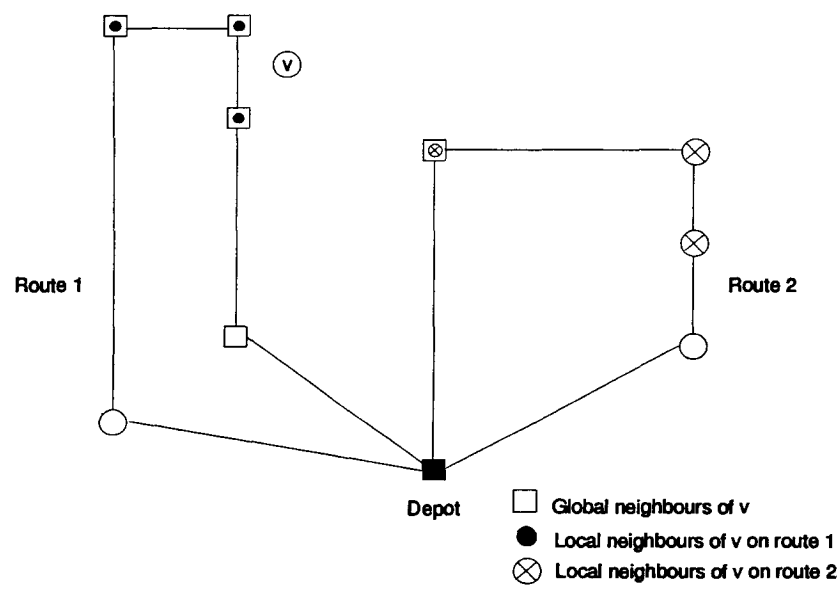

Fig. 1. Global and local neighbourhoods of vertex v for $q = 5$ and $p = 3$.

consisting of all non-tabu routes different from k that include at least one global successor or one global predecessor of v_i. For each route $l \in R(v_i)$, determine z_{il}, the length of the longest route, among k and l, that would result if v_i was moved from route k to route l.

Step 3. Let $S(k)$ be the vertex set of route k. Let v_{i^*} be a vertex of $S(k)$ and let l^* be a route of $R(v_{i^*})$ yielding $z_{i^*l^*} = \min_{v_i \in S(k)} \min_{l \in R(v_i)} \{z_{il}\}$. Move v_{i^*} from route k to route l^* using GENI with neighbourhood size p. If $z_{i^*l^*} < \bar{z}$, identify the longest route and update $\bar{z}$; if $\bar{z} < z^*$, set $z^* := \bar{z}$ and update the incumbent. If $z_{i^*l^*} < \bar{z}$, set $\bar{z} := z_{i^*l^*}$. Declare the move of v_{i^*} from route l^* to route k tabu until iteration $t + \theta$, update the lists of the local predecessors and successors affected by the changes in routes k and l^*. Set $t := t + 1$. If T, the maximum allowed number of iterations without improvement has not been reached, go to Step 2.

Phase 3: Post-optimization

Step 1. Consider the best known solution and let k be index of the longest route.

Step 2. Apply the US post-optimization procedure to route k. Let l be index of the longest route. If $l = k$, stop. Otherwise, set $k := l$ and repeat this step. ■

In our implementation, the value of q is proportional to $\sqrt{n}$. With this rule, the number of global neighbours increases with problem size but does not become unmanageable when n increases. A similar rule is used in TABUROUTE[7] to determine the number of initial solutions. In the same spirit, the value of p is proportional to the square root of the average number of vertices per route, but is never less than 3 since $p = 2$ produces poor results.[6] More specifically, the values of q and p are taken as $q = \lceil \alpha\sqrt{n} \rceil$ and $p = \max\{3, \lceil \sqrt{n/m} \rceil\}$, where α is an input parameter. Tests have shown that setting $\alpha = 0.6$ yields good quality results for reasonable computational effort. We have tested a more extensive scheme consisting of repeatedly running the algorithm with four values of α, but the extra computational effort was not justified by the improvement in the solution quality and this option was dropped.

As in other tabu search applications (see, e.g., [7]), it is preferable to initiate the search from a good solution rather than from a randomly generated solution. This is accomplished by the use of GENI in Phase 1. It should be noted, however, that each individual route constructed by means of GENI is initiated from a random seed as extensive testing has shown that GENI is virtually unaffected by the starting solution.[6] As in TAILLARD[22,23] and GENDREAU, HERTZ and LAPORTE,[7] the number of iterations θ for which a move is declared tabu is not a constant. In our implementation, θ is a random variable uniformly distributed in some interval $[\underline{\theta}, \bar{\theta}]$. We have tested ten choices of these parameters in the range [3, 20] and the conclusion was that selecting $\underline{\theta} = 3$ and $\bar{\theta} = 7$ was the best choice. This is consistent with the range of intervals proposed by GLOVER and LAGUNA[12] for "simple dynamic tabu term rules." Note that in order to reduce the amount of bookkeeping required, no tabu list is actually maintained in our implementation. Instead, we simply use a "tabu tag" θ to indicate for how many iterations a particular move is declared tabu. Finally, the number T of iterations without improvements is proportional to problem size. Tests have shown that setting $T := 10n$ is a good compromise.

Our last comment on this algorithm concerns the relative contribution of the post-optimization phase. Tests indicate that the impact of Phase 3 is to improve solution costs by about 1%, for a computational effort equal to approximately 1% of the total time. The improvement is more significant in instances for which the longest route contains a large number of vertices.

2. TWO EXACT SEARCH SCHEMES

THE TWO EXACT algorithms we propose for the Minmax m-TSP are based on the solution of a closely related problem, the *Distance Constrained VRP* (DVRP). This problem is similar to the Minsum m-TSP except that now every route is constrained not to exceed a preset constant D. The DVRP is a hard combinatorial problem, much more difficult in practice than a Minsum m-TSP of the same size, the reason being that maximum route length constraints are not naturally satisfied when relaxed. Exact constraint relaxation algorithms indicate that DVRPs involving up to 100 vertices can be solved to optimality (see LAPORTE, DESROCHES and NOBERT[14] in the symmetrical case, and LAPORTE, NOBERT and TAILLEFER[16] in the asymmetrical case). By contrast, using similar approaches, Minsum m-TSPs can be solved optimally for several hundreds of vertices. To our knowledge, this study is the first documented attempt to solve the Minmax m-TSP exactly.

Contrary to problems with Minsum objective, Minmax problems are not easily amenable to solution by means of standard branch-and-bound algorithms since solutions obtained at intermediate nodes of the search tree do not as a rule contain sufficient information on the value of the objective to allow fathoming. Instead, it is common in such

contexts to resort to dichotomous search. For example, DELL'AMICO and MARTELLO[3] solve the sequence independent minimum makespan problem by embedding bin packing problems within a dichotomous search scheme.

Our approach is based upon the following considerations. Let z^* denote the optimal solution value of the Minmax m-TSP. If the associated DVRP is feasible, then clearly D is an upper bound on z^*; if S is the sum of the costs of all arcs used in the DVRP solution, then $\lceil S/m \rceil$ is a lower bound on z^* since the mean length of a route does not exceed the maximum route length; if the DVRP is infeasible, there is no solution in which all routes have a length not exceeding D and therefore $D + 1$ is a lower bound on z^*. These observations suggest the following search schemes.

Dichotomous Search:

Step 1. Using a heuristic, obtain an upper bound $\bar{z}$ on the optimal solution value of the Minmax m-TSP.

Step 2. Solve the associated DVRP with $D := \bar{z}$. Let U be the length of the longest route in the solution, and set $L := \lceil S/m \rceil$.

Step 3. If $L = U$, an optimal Minmax m-TSP solution of value $z^* = U$ has been obtained: stop.

Step 4. Solve the associated DVRP with $D := L + \lfloor (U - L)/2 \rfloor$. If this problem is feasible, let U be the length of the longest route in the solution, and set $L := \lceil S/m \rceil$. If the problem is infeasible, set $L := D + 1$. Go to Step 3. ■

This approach requires solving $O(\log(U - L + 1))$ DVRPs, where U and L are defined as in Step 2. A more parsimonious downward search scheme can sometimes be obtained if the initial upper bound on $\bar{z}$ in Step 2 is close to z^*.

Downward Search:

Step 1. Using a heuristic, obtain an upper bound $\bar{z}$ on the optimal value of the Minmax m-TSP. Set $D := \bar{z}$.

Step 2. Solve the associated DVRP with $D := D - 1$.

Step 3. If the problem is infeasible, the previous feasible solution was optimal for the Minmax m-TSP: stop. If the problem is feasible, set D equal to the length of the longest route in the solution and go to Step 2. ■

The advantage of the downward search scheme over the dichotomous approach is that the number of DVRPs to be solved is likely to be lower if a good heuristic starting solution is used. The main disadvantage, however, is that one of the DVRP's will always be infeasible, meaning more often than not a very large search tree.

In either scheme, any exact algorithm for the DVRP can be used. We have chosen to implement the LAPORTE, NOBERT and TAILLEFER[16] algorithm for asymmetrical DVRPs which was already available. Here, the DVRP is first formulated as an assignment problem (AP) with subtour elimination constraints and without maximum route length constraints. The problem is then solved by means of an enumerative scheme which computes at each active node of the search tree a lower bound on the AP solution value, or the full AP solution. Undominated subproblem solutions are examined for possible violations of the relaxed constraints, and these are introduced by applying the CARPANETO and TOTH[1] branching rules. This strategy also works in principle for symmetrical problems, but the number of illegal subtours generated is then likely to be prohibitive. Instead, it is preferable to use in this case an integer linear programming based approach that uses only one decision variable for each pair $\{v_i, v_j\}$, thus producing fewer subtours.[14]

3. COMPUTATIONAL RESULTS

WE HAVE IMPLEMENTED the nearest neighbour (NN) algorithm of FREDERICKSON, HECHT and KIM,[5] and the tabu search heuristic (TS) described in Section 1. These heuristics were tested in conjunction with the dichotomous (DICH) and the downward (DOWN) exact search schemes. As expected, the NN/DOWN combination did not perform well because of the low quality of the first feasible solution and was soon discarded. Our computational results cover the remaining three cases NN/DICH, TS/DICH and TS/DOWN.

Two types of asymmetrical problems were tested. 1) In *structured problems*, $2(n + 1)$ points $P_0, P_1, \ldots, P_n, Q_0, Q_1, \ldots, Q_n$ were first generated in the $[0, 100]^2$ square according to a continuous uniform distribution. For $i < j$, c_{ij} was then defined as the truncated Euclidean distance between P_i and P_j; for $i > j$, c_{ij} was set equal to the truncated Euclidean distance between Q_i and Q_j. Using these values, each coefficient c_{ij} was then replaced by the length of a shortest path from i to j, so that the triangle inequality was satisfied in the resulting (c_{ij}) matrix. This generation procedure is the same as in [16]. An alternative scheme could have been to first create a matrix of integer c_{ij}'s generated according to a discrete uniform distribution, and then replace these by shortest path costs, but this method tends to produce a highly degenerate and unstructured matrix, with several costs in the same

small interval. The costs obtained with our method tend to have a larger variance. 2) *Unstructured problems* were simply obtained by generating each c_{ij} according to a discrete uniform distribution in $[1, 100]$. In each case, ten different problems were generated for each combination of $n = 20, 30, 40, 50$ and $m = 2, 3, 4, 5$. All problems were solved on a Sun SPARC2 workstation and a maximum of 1000 seconds were allowed for the solution of any DVRP. Instances that could not be solved within that time specification were deemed unsuccessful. In Tables I and II, we report *average* results over the number of successful instances when this number is at least five. When not all instances can be solved, the number of successful cases is indicated by the number of parentheses in the third column.

The various headings of Tables I and II are defined as follows:

HEURISTIC/EXACT: NN: nearest neighbour heuristic;
TS: tabu search;
DICH: dichotomous search;
DOWN: downward search;

n: number of vertices excluding the depot;
m: number of vehicles;
$\bar{z}/z^*$: heuristic solution value divided by optimal solution value;
$\underline{z}/z^*$: lower bound value provided by the first DVRP solution divided by optimal solution value;
#DVRP: number of DVRPs solved to optimality;

TIME: HEUR: time required to run the heuristic;
EXACT: time required to run the remaining part of the exact algorithm;
TOTAL: total solution time.

Computational results indicate that the nearest neighbour heuristic requires insignificant amounts of machine time, but produces highly suboptimal solutions. The tabu search heuristic generates much better solutions at the expense of more computational effort. Good approximate solution values are obtained in the case of structured problems, but unstructured problems remain difficult, particularly for large values of n. We can offer the following explanation for this behaviour. Tabu search is an iterative search method that explores the solution space by examining sequences of neighbouring

TABLE I

Computational Results for Structured Problems

HEURISTIC/EXACT	n	m	$\bar{z}/z^*$	$\underline{z}/z^*$	#DVRP	Time (seconds) HEUR	EXACT	TOTAL
NN/DICH	20	2	1.53	0.95	3.8	0.001	35.781	35.782
		3	1.42	0.93	3.9	0.004	729.475	729.479
		4(7)	1.60	0.90	3.9	0.001	1494.058	1494.059
		5(8)	1.57	0.88	4.5	0.005	3386.915	3386.920
	30	2	1.42	0.98	2.7	0.001	30.227	30.228*
		3	1.54	0.95	4.7	0.007	2784.769	2784.776
	40	2	1.42	0.99	3.4	0.006	294.682	294.688
		3(8)	1.61	0.96	4.5	0.011	2388.599	2338.610
	50	2	1.47	0.98	4.4	0.006	762.982	762.988*
		3(6)	1.50	0.97	4.2	0.006	2937.325	2937.331*
TS/DICH	20	2	1.03	0.98	2.7	7.252	35.523	42.775
		3	1.02	0.97	2.8	6.022	248.720	254.742
		4(9)	1.04	0.95	2.7	4.498	1188.387	1192.885
		5(8)	1.03	0.95	3.1	4.723	2655.643	2660.366
	30	2	1.05	0.99	2.1	24.777	26.074	50.851
		3(9)	1.06	0.97	3.3	22.632	2626.976	2649.608
	40	2	1.08	0.99	2.5	54.228	221.774	276.002*
		3(8)	1.10	0.97	3.0	34.079	2342.698	2376.777
	50	2	1.12	0.98	3.4	90.280	751.451	841.731
		3(6)	1.14	0.98	3.2	69.555	2900.326	2969.881
TS/DOWN	20	2	1.03	0.98	2.4	7.252	12.521	19.773*
		3	1.02	0.97	2.1	6.022	77.983	84.005*
		4(9)	1.04	0.95	2.3	4.498	709.799	714.297*
		5(8)	1.03	0.95	2.6	4.723	1782.522	1787.245*
	30	2	1.05	0.99	2.0	24.777	24.475	49.252
		3	1.06	0.97	2.9	24.665	652.896	677.561*
	40	2	1.08	0.99	2.8	54.228	223.666	277.894
		3(9)	1.10	0.97	4.7	32.959	1720.938	1753.897*
	50	2	1.12	0.98	3.7	90.280	897.859	988.139

TABLE II

Computational Results for Structured Problems

HEURISTIC/EXACT	n	m	$\bar{z}/z^*$	$\underline{z}/z^*$	#DVRP	Time (seconds) HEUR	EXACT	TOTAL
NN/DICH	20	2	1.75	0.94	4.0	0.003	18.620	18.623*
		3	2.00	0.87	4.1	0.003	204.507	204.510
		4	1.67	0.86	4.8	0.001	251.697	251.698
		5	1.75	0.82	4.7	0.003	472.884	472.887
	30	2	2.02	0.96	3.9	0.003	79.807	79.810
		3	1.85	0.93	4.0	0.006	173.905	173.911*
		4(7)	1.97	0.91	4.3	0.001	1508.280	1508.281
	40	2	2.14	0.98	3.7	0.006	46.722	46.728*
		3(9)	2.13	0.94	4.8	0.005	1248.020	1248.025
	50	2	2.31	0.98	3.7	0.006	204.393	204.399
		3(8)	2.32	0.96	4.4	0.011	1611.085	1611.096*
TS/DICH	20	2	1.10	0.96	3.1	8.655	26.183	34 838
		3	1.12	0.91	3.1	7.173	179.782	186.955
		4	1.12	0.89	3.6	5.018	776.466	781.464
		5	1.09	0.87	3.5	5.133	131.129	136.262
	30	2	1.27	0.96	3.5	21.532	98.826	120.358
		3	1.26	0.94	3.1	20.885	193.536	214.421
		4(7)	1.22	0.91	3.6	13.850	1858.554	1872.404
	40	2	1.39	0.98	3.2	45.958	91.298	137.256
		3(9)	1.41	0.95	4.3	36.142	1279.559	1315.292
	50	2	1.52	0.98	3.5	86.213	292.339	378.552
		3(8)	1.48	0.96	4.0	67.229	1671.998	1739.227
TS/DOWN	20	2	1.10	0.96	2.7	8.655	15.299	23.954
		3	1.12	0.91	2.7	7.173	69.587	76.760*
		4	1.12	0.89	3 5	5.018	106.869	111.887*
		5	1.09	0.87	3.2	5.133	78.284	83.417*
	30	2	1.27	0.96	4.2	21.532	39.495	64.027*
		3	1.26	0.94	3.8	20.885	208.602	229.487
		4(9)	1.24	0.91	4.9	13.454	1628.827	1642.281*
		5(5)	1.27	0.90	5.2	18.267	2149.038	2167.305*
	40	2	1.39	0.98	4.3	45.958	64.241	110.199
		3(9)	1.41	0.95	6.4	36.142	843.762	879.904*
		4(5)	1.39	0.94	4.6	29.092	1623.018	1652.110*
	50	2	1.52	0.98	3.9	86.213	112 488	198.701*
		3(7)	1.49	0.97	4.9	68.990	2327.904	2396.894

solutions, the underlying assumption being that the solution space is somewhat structured. In other words, the algorithm exploits the fact that any two successive solutions examined should not differ too much from each other. When the solution space is highly unstructured the search becomes more random and less efficient. For example, tabu search would perform very poorly on the problem of minimizing a function f that assigns an arbitrary value to each vector of $\{0, 1\}^n$.

Our results also throw some light on the quality of the two exact algorithms. When examining these results, one must bear in mind that the DVRP is a very hard problem. We solved it by means of a constraint relaxation algorithm whose performance diminishes when D becomes smaller.[16] In our problem, D becomes very small indeed as we attempt to bring it down to its lowest possible value. As a result, the success rate of the exact algorithm is sometimes relatively low and there is very little likelihood it would improve substantially if we were, for example, to double the time limit as the difficulty of the problem increases exponentially with n.

We have compared the various versions of the exact algorithm. The best option is indicated by an asterisk in Tables I and II. As expected, downward search works better than dichotomous search if the initial upper bound is close to the optimal solution since in this case fewer DVRPs need to be solved. In some cases, the initial upper bound U is only a few units above the optimal solution value z^*, which limits the number of DVRPs that have to be solved using the downward search algorithm. Furthermore, it is not always necessary to consider all values D between U and z^* as some may be skipped when new feasible solutions are encountered. This should occur more often in "coarse grain" problems, i.e., those that exhibit large gaps between succes-

sive solution values. This behaviour is more likely to be observed if the c_{ij}'s are generated over a small internal (e.g., [1, 100]), instead of a larger one (e.g., [1, 1000]). Whether dichotomous or downward search is used, applying TS as opposed to NN means that fewer DVRPs must be solved, but this does not always compensate for the extra time required to run the heuristic. In fact, the downward search algorithm only works well if a good heuristic solution is used as a starting point. The NN/DOWN combination was by far the worst and the TS/DOWN combination appears to be the best in most cases.

Contrary to tabu search, the exact DVRP algorithm works better on unstructured problems. This is explained by the fact that in these problems, the costs of the feasible solutions have a larger variance than in the case of structured problems: this results in more dominance, earlier fathoming of subproblems, and more limited search trees. Finally note that one interesting feature of our approach is that the number of DVRPs solved is relatively low and any algorithmic improvement for this problem will translate directly into shorter computing times for the Minmax m-TSP.

4. CONCLUSION

WE HAVE PROPOSED new approximate and exact algorithms for the Minmax m-TSP, a version of the m-TSP in which the objective is to minimize the length of the longest route. Problems involving up to 50 vertices were solved to optimality. To the authors' knowledge, no computational results have previously been reported for this difficult problem and no exact algorithm has ever been proposed. Tabu search produces significantly better results than the nearest neighbour heuristic, at the expense of extra computational effort. However, when a dichotomous search scheme is used, it does not always pay to spend too much time on the heuristic. This explains why NN/DICH is often the best option in this case. Overall, the best combination is to use tabu search to determine an initial solution, followed by downward search.

ACKNOWLEDGMENTS

THE WORK OF Paulo M. França was supported in part by Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP), and by Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), Brazil. Michel Gendreau and Gilbert Laporte were funded by the Canadian natural Sciences and Engineering Research Council under grants OGP0038816 and OGP0039682, and by the Quebec government under grant FCAR 93ER0289. Felipe M. Müller was partially supported by Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), Brazil. Thanks are also due to the associate editor and to three referees for their valuable comments.

REFERENCES

1. G. CARPANETO AND P. TOTH, "Some New Branching and Bounding Criteria for the Asymmetric Travelling Salesman Problem," *Management Sci.* **26**, 736–743 (1980).
2. N. CHRISTOFIDES, "Worst-Case Analysis of a New Heuristic for the Traveling Salesman Problem." Report 388, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, PA, 1976.
3. M. DELL'AMICO AND S. MARTELLO, "Optimal Scheduling of Tasks on Identical Parallel Processors," *ORSA Journal on Computing* **7**, 191–200 (1995).
4. P. M. FRANCA, M. GENDREAU, G. LAPORTE AND F. M. MÜLLER, "A Composite Heuristic for the Identical Parallel Machine Scheduling Problem with Minimum Makespan Objective," *Comput. & Oper. Res.* **21**, 205–210 (1994).
5. G. N. FREDERICKSON, M. S. HECHT and C. E. KIM, "Approximation Algorithms for Some Routing Problems," *SIAM Journal on Computing* **7**, 178–193 (1978).
6. M. GENDREAU, A. HERTZ AND G. LAPORTE, "New Insertion and Post-Optimization Procedures for the Traveling Salesman Problem," *Oper. Res.* **40**, 1986–1094 (1992).
7. M. GENDREAU, A. HERTZ AND G. LAPORTE, "A Tabu Search Heuristic for the Vehicle Routing Problem," *Management Sci.* **40**, 1276–1290 (1994).
8. É. GIUST, "Optimisation de tournées de véhicules. Application à la distribution de gaz," M.Sc. Dissertation, Facultés Universitaires Notre-Dame de la Paix, Namur, Belgium, 1992.
9. F. GLOVER, "Heuristic for Integer Programming Using Surrogate Constraints," *Decision Sci.* **8**, 156–166 (1977).
10. F. GLOVER, "Tabu Search, Part F," *ORSA Journal of Computing* **1**, 190– 219 (1990).
11. F. GLOVER, "Tabu Search, Part II," *ORSA Journal of Computing* **2**, 4–32 (1991).
12. F. GLOVER and M. LAGUNA, "Tabu Search," in *Modern Heuristic Techniques for Combinatorial Problems*, C. Reeves (ed.), Blackwell Scientific Publications, Oxford, U.K., 1993, pp. 70–150.
13. F. GLOVER, É. TAILLARD and D. DE WERRA, "A User's Guide to Tabu Search," *Annals of Operations Research* **41**, 3–28 (1993).
14. G. LAPORTE, M. DESROCHERS AND Y. NOBERT, "Two Exact Algorithms for the Distance-Constrained Vehicle Routing Problem", *Networks* **14**, 161–172 (1984).

15. G. LAPORTE AND Y. NOBERT, "A Cutting Planes Algorithm for the m-Salesmen Problem", *Journal of Operational Research Society* **31**, 1017–1023 (1980).
16. G. LAPORTE, Y. NOBERT AND S. TAILLEFER, "A Branch-and-Bound Algorithm for the Asymmetrical Distance-Constrained Vehicle Routing Problem", *Mathematical Modelling* **9**, 857–868 (1987).
17. J. K. LENSTRA AND A. H. G. RINNOOY KAN, "Some Simple Applications of the Traveling Salesman Problem", *Operational Research Quarterly* **26**, 717–734 (1975).
18. I. H. OSMAN, "Metastrategy Simulated Annealing and Tabu Search Algorithms for the Vehicle Routing Problem", *Ann. Oper. Res.* **41**, 421–451 (1993).
19. V. M. PUREZA AND P. M. FRANÇA, "Vehicle Routing Problem via Tabu Search Metaheuristic," Publication #747, Centre de recherche sur les transports, Montreal (1991).
20. D. J. ROSENKRANTZ, R. E. STEARNS AND P. M. LEWIS II, "An analysis of Several Heuristics for the Traveling Salesman Problem," *SIAM Journal on Computing* **6**, 563–581 (1977).
21. F. SEMET AND É. TAILLARD, "Solving Real-Life Vehicle Routing Problems Efficiently Using Taboo Search," *Ann. Oper. Res.* **41**, 469–488 (1993).
22. É. TAILLARD, "Robust Taboo Search for the Quadratic Assignment Problem," *Parallel Computing* **17**, 433–455 (1991).
23. É. TAILLARD, "Parallel Iterative Search Methods for Vehicle Routing Problems," *Networks* **23**, 661–673 (1993).

(Received, December 1992; revisions received: December 1993, August 1994; accepted: September 1994)