Solving the Asymmetric Traveling Salesman Problem with Periodic Constraints

Giuseppe Paletta
Dipartimento di Economia e Statistica, Università della Calabria, 87036 Rende (CS), Italy

Chefi Triki
Dipartimento di Matematica, Università di Lecce, via Arnesano, 73100 Lecce, Italy

In this article we describe a heuristic algorithm to solve the asymmetrical traveling salesman problem with periodic constraints over a given *m*-day planning horizon. Each city *i* must be visited r_i times within this time horizon, and these visit days are assigned to *i* by selecting one of the feasible combinations of r_i visit days with the objective of minimizing the total distance traveled by the salesman. The proposed algorithm is a heuristic that starts by designing feasible tours, one for each day of the *m*-day planning horizon, and then employs an improvement procedure that modifies the assigned combination to each of the cities, to improve the objective function. Our heuristic has been tested on a set of test problems purposely generated by slightly modifying known test problems taken from the literature. Computational comparisons on special instances indicate encouraging results. © 2004 Wiley Periodicals, Inc. NETWORKS, Vol. 44(1), 31–37 2004

Keywords: asymmetric traveling salesman problem; periodic constraints; construction algorithm; improvement procedure

1. INTRODUCTION

The necessity of formulating periodic constraints within routing problems arises in many real-life applications. Many distribution problems are, indeed, characterized by the fact that the cities should not be visited at each day of an *m*-day planning horizon, but rather a specified number of times. This is the case, for example, of some mail collection/delivery problems, snow removal, refuse collection, grocery distribution, and fuel oil delivery.

The periodicity aspect in the routing problems has attracted the interest of many researchers over the last 2 decades. A wide variety of well-known routing problems has been covered in this direction. Christofides and Beasley [3] have formulated the period vehicle routing problem (PVRP) and have solved the period traveling salesman problem (PTSP) by using heuristic algorithms. Other heuristics to solve the PTSP have been also proposed by Paletta [7], Chao et al. [1], and recently again by Paletta [8]. The PTSP problem has also been studied by Cordeau et al. [4], who have proposed a tabu search metaheuristic algorithm. They have also employed their tabu search approach to solve the PVRP and the multidepot vehicle routing problem. Gaudioso and Paletta [6] have solved a variant of the PVRP by using an algorithm based on a combination of a city-route assignment heuristic and a bin-packing algorithm for the route-vehicle assignment. The PVRP has also been studied by Chao et al. [2], who have proposed an efficient heuristic that first assigns the visit combinations by solving an integer linear programming model and solves a VRP for each day and then uses local improvement and reinitialization techniques to improve the quality of the solution.

This article represents a further step towards the completion of the above-mentioned work. Indeed, we cover here another routing problem, namely the Periodic Asymmetric Traveling Salesman Problem (PATSP).

In the next section, we will define the asymmetric traveling salesman problem, and describe the different approaches for formulating periodicity constraints. Section 3 will be devoted to the development of a heuristic framework to solve the PATSP. The computational performance of the heuristic will be discussed in Section 4, and some concluding remarks will be made in Section 5.

2. PROBLEM DESCRIPTION

The PATSP represents a natural extension of the asymmetric traveling salesman problem to cover an *m*-day planning horizon. Within this time horizon, each city *i* must be visited r_i times, with at most one visit per day. These visits are assigned to *i* by selecting one of a given set of feasible combinations of r_i visit days with the objective of minimiz-

Received April 2002; accepted January 2004
Correspondence to: G. Paletta; e-mail: g.paletta@unical.it
DOI 10.1002/net.20011
Published online in Wiley InterScience (www.interscience.wiley.com).

ing the total distance traveled by the salesman. The definition of the PATSP will be done through the description of the classical ATSP and the introduction of the different approaches proposed in the literature to represent the periodicity constraints.

2.1. Asymmetric Traveling Salesman Problem

The asymmetric traveling salesman problem (ATSP) may be formally defined in terms of the graph theory on a directed network $G = (I, A)$. The set of vertices $I = \{1, \ldots, i, \ldots, n\}$ includes all the cities to be served and, in addition, we denote as vertex 0 the home city of the salesman. With $A = \{(i, j): \forall i \text{ and } j \in I \cup \{0\}\}$ we denote the set of all directed arcs that connect the vertices of the network. To each arc $(i, j) \in A$ is associated a traversal cost $c_{i,j}$ that generally represents its length. The network is assumed asymmetric and, thus, $c_{i,j}$ and $c_{j,i}$ may be different. The ATSP problem consists in finding a minimum cost circuit passing through each vertex in the network exactly once. Such a circuit is known as a Hamiltonian circuit or simply a tour.

2.2. Periodicity Constraints

Periodicity within routing problems can be formulated by using different alternative representations. All of these lead to the definition, for each city $i \in I$, of a set of possible combinations of r_i visit days that are feasible for that city. They mainly differ in the level of restriction in the definition of allowable alternatives and, consequently, in the number of feasible combinations that can be generated. In what follows we will describe three different representations that have been proposed in the literature.

Predetermined Sequence. This widely used representation has been adopted by Christofides and Beasley [3]. Each city i specifies explicitly all the allowable alternatives that define the set of combinations, denoted by $V(i)$, and the salesman should select only one of these combinations. Although the set $V(i)$, in this case, is given directly by the city, in the successive cases a preprocessing phase is needed to form an explicit definition of the combinations.

Periodic Sequence. In this formulation, used by Chao et al. [1] and Cordeau et al. [4], each city specifies the number of service visits r_i during the m-day planning horizon, and each city must be visited every m/r_i days. If, for example, $r_i = 3$ and $m = 6$, then the city i must be periodically visited every $m/r_i = 2$ days up to the end of the planning horizon. In this case, visiting the city i in the first day of the 6-day planning horizon means that i must be also visited on the third and fifth day [i.e., combination $\{1, 3, 5\}$]. If, however, the first visit to city i is postponed to the second day, then the subsequent visits should be done on the fourth and sixth day [i.e., combination $\{2, 4, 6\}$]. The ratio m/r_i, i.e., the number of alternatives, defines the cardinality of $V(i)$, the set of feasible combinations for city i.

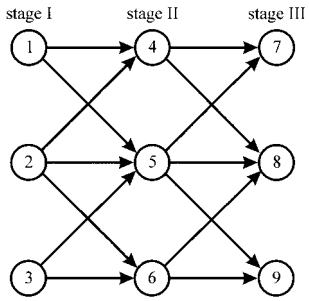

FIG. 1. Multistage network structure.

Multistage Network Sequence. In this representation proposed by Paletta [7], through the m-day planning horizon, a city i must be visited once during each time interval of T_i days, so $r_i = m/T_i$. Furthermore, additional constraints impose that at least l_i days and at most u_i days must elapse between two successive visits. Paletta has appropriately represented this formulation as an acyclic multistage network corresponding to each city $i \in I$. The nodes of stage k represent the allowable alternative days to execute the kth visit to the city i, whereas each arc represents 2 possible successive visit days. The set of feasible combinations $V(i)$ is thus defined by all the paths between the nodes of the first and the last stages of the directed network. An example of a network structure with $m = 9$, $T_i = 3$, $l_i = 2$, and $u_i = 4$ is represented in Figure 1.

In this article we will suppose that the set $V(i)$ for each city $i \in I$ is defined on the basis of the periodic sequence representation. The use of other representations is straightforward.

3. HEURISTIC ALGORITHM

The algorithm that we propose is a tour construction heuristic followed by an improvement procedure. The development of the overall algorithm is based on the definition of five modules that, once integrated, produce a suboptimal solution for the PATSP. Besides the concatenated modules regarding the initialization, the construction and the improvement of the solution, we develop two additional modules to define the insertion/removal criteria.

For the description of the different modules we need to introduce some additional notations. We indicate by $P = \{1, \ldots, t, \ldots m\}$ the m-day planning horizon during which each city $i \in I$ must be visited r_i times. Moreover, H_t will represent the tour currently designed for day $t \in P$.

3.1. Module I: Starting Tour

The algorithm selects a city $i \in I$, assigns to it one of the combinations belonging to the set $V(i)$, and forms the starting tour $H_t = \{0 - i - 0\}$ for each day t of the chosen

combination. This approach is expected to be more efficient than the trivial initialization with all empty tours.

3.2. Module II: Tour Construction Procedure

The construction procedure selects, at each iteration, an as-yet unvisited city and assigns to it the feasible combination that minimizes the total insertion cost, as defined in module IV.

Finally, it inserts the selected city into the tour developed for each day of the assigned combination. The procedure terminates when all the cities are visited. For a detailed description of the construction procedure we need to define the set Z that includes all the as yet unvisited cities, and the set W including the already visited cities (clearly $I = Z \cup W$). The steps of the procedure can be summarized as follows:

Repeat

Select a city s from Z such that $s = arg\ max_{i \in Z} \{r_i \cdot min_{j \in W}\{c_{i,j}\}\}$;

For each combination $k \in V(s)$

—calculate, for each day t, the insertion cost $ic(s, t)$, as defined in Module IV;

—calculate the total insertion cost $ic_k(s)$ for each $k \in V(s)$, as defined in Module IV;

end for k

Assign to s the combination with the lowest total insertion cost $ic(s) = min_{k \in V(s)}\{ic_k(s)\}$;

For each day t of the assigned combination, insert s in the tour H_t;

Set $Z = Z \backslash \{s\}$ and $W = W \cup \{s\}$;

Until Z is empty.

At the end of the construction phase, the algorithm provides a reasonably good solution but that is not, in some cases, close enough to the optimal solution. This gap is due to the fact that the algorithm is based on an optimization process that uses only the local information that is made available at each step of the iterative procedure. The employment of an improvement procedure may represent a necessary step towards the globalization of the optimization process in order to produce a better solution.

3.3. Module III: Improvement Procedure

Once the construction phase is achieved, an improvement procedure is used with the objective of reducing the total distance. The procedure is based on the modification of the assigned combination to some of the cities of set I. Let $T(i)$ be the set of all the cities that are either predecessor or successor to city i in at least one of the tours inside which i is inserted. For each city i, the procedure first removes all the cities belonging to the set $T(i) \cup \{i\}$ and then reinserts them in the tours corresponding to the newly assigned combination. The solution obtained is accepted if its quality is improved; otherwise, it is rejected. A detailed scheme of the procedure can be represented as follows:

For each $i \in I$

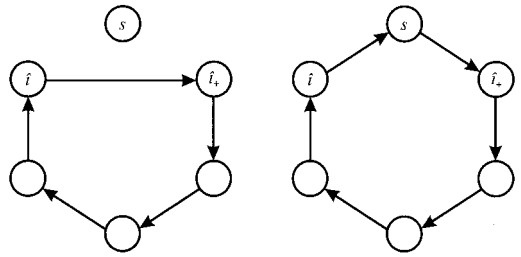

FIG. 2. Insertion Rule 1.

Define the set $T(i)$;

For each $s \in T(i) \cup \{i\}$

Remove s from the tours inside which it is inserted, as defined in Module V;

End for s

For each $s \in T(i) \cup \{i\}$

—Calculate the total insertion cost $ic_k(s)$ for each $k \in V(s)$, as defined in Module IV;

—Assign to s the combination that corresponds to the lowest total insertion cost $ic(s) = min_{k \in V(s)}\{ic_k(s)\}$;

—Insert s in the tours designed for the days of the assigned combination;

End for s

If the total length of the resulting tours is reduced, then make the exchange; otherwise, the solution remains unaltered;

End for i.

It is worthwhile noting that in both the construction and the improvement modules we have not imposed constraints related to empty tours. Indeed, it is possible that the heuristic provides a solution with empty tours for 1 or more days of the planning horizon.

3.4. Module IV: Insertion Criterion

The insertion criterion involves a twofold task in the case of the PATSP: the insertion of a city in a tour, and the assignment of one of the feasible combinations. For the first task, common to all ATSP algorithms, we have evaluated several alternatives, considering the insertion cost as a selection criterion. The insertion rule corresponding to the lowest insertion cost is selected. In the second task, specific to the periodic problems, the heuristic selects the combination that minimizes the total insertion cost over the planning horizon.

More specifically, we propose new rules that have been employed within our algorithm as criterion to insert a city s in a tour H_t.

Rule 1. This rule, which corresponds to the standard cheapest-cost insertion rule [9], inserts the vertex s by removing from H_t the arc $(\hat{\imath}, \hat{\imath}_+)$ corresponding to the following minimal cost:

$$ic_1(s, t) = \min_{i \in H_t}\{c_{i,s} + c_{s,i_+} - c_{i,i_+}\}$$

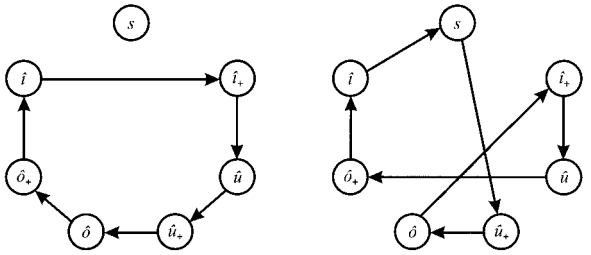

FIG. 3. Insertion Rule 2.

and by replacing it with the arcs $(\hat{\imath}, s)$ and $(s, \hat{\imath}_+)$ (Fig. 2).

Rule 2. The vertex s is inserted in H_t by removing the arcs $(\hat{\imath}, \hat{\imath}_+)$, $(\hat{u}, \hat{u}_+)$, and $(\hat{o}, \hat{o}_+)$ that correspond to the following minimal cost:

$$ic_2(s, t) = \min_{i,u,o \in H_t, i \neq u \neq o}\{c_{i,s} + c_{s,u_+} + c_{o,i_+} + c_{u,o_+} - c_{i,i_+} - c_{u,u_+} - c_{o,o_+}\}$$

and replacing them with the arcs $(\hat{\imath}, s)$, $(s, \hat{u}_+)$, $(\hat{o}, \hat{\imath}_+)$, and $(\hat{u}, \hat{o}_+)$ for $i > u > o$, where $i > u > o$ indicates that nodes i, u, and o appear in this order when the tour is followed in its direction (Fig. 3).

Rule 3. The vertex s is inserted in H_t by removing the arcs $(\hat{\imath}, \hat{\imath}_+)$, $(\hat{u}, \hat{u}_+)$, and $(\hat{o}, \hat{o}_+)$ that correspond to the following minimal cost:

$$ic_3(s, t) = \min_{i,u,o \in H_t, i \neq u \neq o}\{c_{o,s} + c_{s,i_+} + c_{u,o_+} + c_{i,u_+} - c_{i,i_+} - c_{u,u_+} - c_{o,o_+}\}$$

and replacing them with the arcs $(\hat{o}, s)$, $(s, \hat{\imath}_+)$, $(\hat{u}, \hat{o}_+)$ and $(\hat{\imath}, \hat{u}_+)$ for $i > u > o$ in the direction of the tour (Fig. 4).

Rule 4. The vertex s is inserted in H_t by removing the arcs $(\hat{\imath}, \hat{\imath}_+)$, $(\hat{u}, \hat{u}_+)$, and $(\hat{o}, \hat{o}_+)$ that correspond to the following minimal cost:

$$ic_4(s, t) = \min_{i,u,o \in H_t, i \neq u \neq o}\{c_{u,s} + c_{s,o_+} + c_{i,u_+} + c_{o,i_+} - c_{i,i_+} - c_{u,u_+} - c_{o,o_+}\}$$

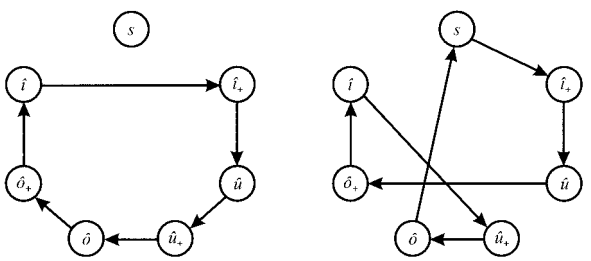

FIG. 4. Insertion Rule 3.

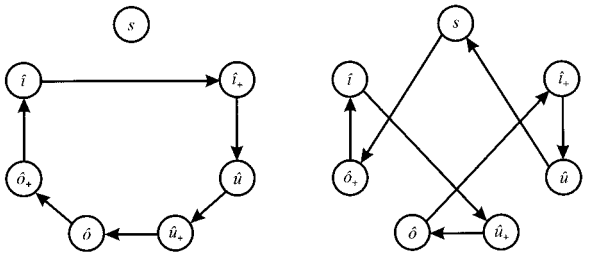

FIG. 5. Insertion Rule 4.

and replacing them with the arcs $(\hat{u}, s)$, $(s, \hat{o}_+)$, $(\hat{\imath}, \hat{u}_+)$, and $(\hat{o}, \hat{\imath}_+)$ for $i > u > o$ in the direction of the tour (Fig. 5).

The insertion rules 2, 3, and 4 correspond to deleting the links beginning at $\hat{\imath}$, $\hat{u}$, and $\hat{o}$, and inserting vertex s after $\hat{\imath}$, $\hat{o}$, or $\hat{u}$, respectively, preserving the orientation of the tour.

On the basis of these rules the lowest insertion cost is determined as:

$$ic(s, t) = \min\{ic_1(s, t), ic_2(s, t), ic_3(s, t), ic_4(s, t)\}.$$

We proceed, therefore, by calculating, for each combination $k \in V(s)$, the total insertion cost $ic_k(s)$ defined as the sum of the insertion costs of s into the tours corresponding to the days of the combination k. The lowest total insertion cost among all the combinations of $V(s)$, i.e.,

$$ic(s) = \min_{k \in V(s)}\{ic_k(s)\},$$

will define a solution to the combination assignment problem.

3.5. Module V: Removal Rule

In the algorithm we propose the following rules to delete the vertex r from the tour H_t.

Rule 1. The vertex r is deleted from H_t by removing its two incident arcs $(r_-, r) \in H_t$ and $(r, r_+) \in H_t$ and replacing them with the arc (r_-, r_+). The cost is (Fig. 6):

$$rc_1(r, t) = c_{r_-,r_+} - c_{r_-,r} - c_{r,r_+}$$

Rule 2. The vertex r is deleted from H_t by removing its two incident arcs $(r_-, r) \in H_t$ and $(r, r_+) \in H_t$ and the

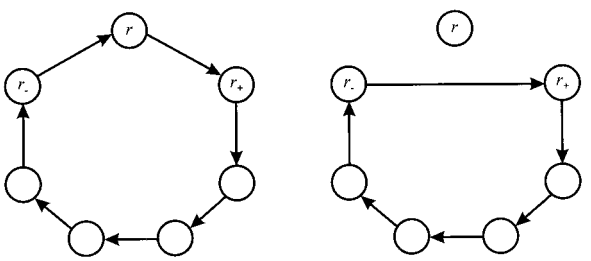

FIG. 6. Removal Rule 1.

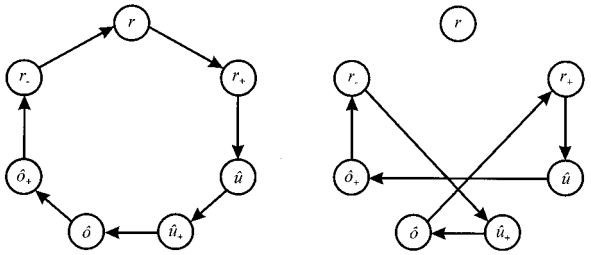

FIG. 7. Removal Rule 2.

arcs $(\hat{u}, \hat{u}_+)$, and $(\hat{o}, \hat{o}_+)$ such that:

$$rc_2(r, t) = \min_{u,o \in H_t, u \neq o}\{c_{r_-,u_+} + c_{o,r_+} + c_{u,o_+} - c_{r_-,r} - c_{r,r_+} - c_{u,u_+} - c_{o,o_+}\},$$

and replacing them with the arcs $(r_-, \hat{u}_+)$, $(\hat{o}, r_+)$, and $(\hat{u}, \hat{o}_+)$ for $r > u > o$ in the direction of the tour. This removal rule amounts to first deleting vertex r, followed by performing the only orientation preserving 3-opt move with respect to the links beginning at r_-, $\hat{u}$, and $\hat{o}$ (Fig. 7).

We define then the removal cost of a vertex r from H_t as the minimum between the two alternative costs $rc_1(r, t)$ and $rc_2(r, t)$ corresponding to the removal rules described above:

$$rc(r, t) = \min\{rc_1(r, t), rc_2(r, t)\}.$$

The total removal cost of vertex r is thus the sum of the costs of removing the city r from the tours corresponding to the days of the combination assigned to the city r.

4. COMPUTATIONAL RESULTS

This section is dedicated to the application of our heuristic to solve some test problems and to the discussion of its performance. To our knowledge, there are no test problems available for the PATSP, because we believe that it is the first time that this problem has been tackled. For this reason, we have slightly modified well-known ATSP test problems (19 of which are available in TSPLIB [10] and eight additional problems proposed by Fischetti and Toth [5] to cover an m-day extended horizon and to generate, consequently, new instances of the PATSP. More specifically, we have fixed the horizon to 1 week with 6 working days, and assigned to each city an integer value r_i representing the number of its service visits within that 6-day planning horizon. For each test problem, the values of r_i have been assigned arbitrarily as follows:

- $r_i = 6$ (i.e., one visit in every day of the planning horizon) for $i = 1, \ldots, \lceil n/4 \rceil$; in this case the only allowable combination of visit days is {1, 2, 3, 4, 5, 6};
- $r_i = 3$ (i.e., three visits in the planning horizon) for $i = \lceil n/4 \rceil + 1, \ldots, \lceil n/2 \rceil$; in this case the allowable combinations of visit days are {1, 3, 5} and {2, 4, 6};
- $r_i = 2$ (i.e., two visits in the planning horizon) for $i = \lceil n/2 \rceil + 1, \ldots, \lceil 3n/4 \rceil$; in this case, the allowable combinations of visit days are {1, 4}, {2, 5}, and {3, 6};
- $r_i = 1$ (i.e., only one visit in the planning horizon) for $i = \lceil 3n/4 \rceil + 1, \ldots, n$; in this case, the allowable combinations of visit days are {1}, {2}, {3}, {4}, {5}, and {6};

where $\lceil x \rceil$ represents the smallest integer not less than the number x.

TABLE 1. Computational results for a planning horizon $m = 6$.

Problem	n	Mean_val	Best_val	Mean_err	Max_err	N_best	T_mean
Pft53	52	22690.92	22557.00	0.59	2.27	1	0.03
Pft70	69	115047.10	114387.00	0.58	1.39	1	0.10
Pbr17	16	229.50	228.00	0.66	2.63	12	0.00
Pry48p	47	69105.41	68368.00	1.08	1.42	2	0.04
Pkro124p	99	135479.20	133699.00	1.33	2.56	1	0.29
Pp43	42	6273.69	6267.00	0.11	0.18	4	0.00
Pftv33	33	4541.21	4528.00	0.29	1.50	2	0.00
Pftv35	35	4808.14	4769.00	0.82	3.77	12	0.00
Pftv38	38	4993.79	4938.00	1.13	3.40	12	0.00
Pftv44	44	5433.30	5356.00	1.44	3.16	1	0.00
Pftv47	47	6514.28	6422.00	1.44	2.82	4	0.00
Pftv55	55	5992.49	5924.00	1.16	2.36	5	0.01
Pftv64	64	7364.05	7266.00	1.35	3.59	1	0.10
Pftv70	70	6638.87	6511.00	1.96	6.60	3	0.10
Pftv90	90	5788.31	5721.00	1.18	2.60	1	0.27
Pftv100	100	6229.29	6154.00	1.22	3.54	1	0.22
Pftv110	110	6656.65	6580.00	1.16	4.98	9	0.35
Pftv120	120	7271.53	7177.00	1.32	3.07	2	0.40
Pftv130	130	7698.99	7563.00	1.80	3.82	1	0.69
Pftv140	140	8312.76	8109.00	2.51	6.44	2	0.98
Pftv150	150	8679.15	8525.00	1.81	4.20	5	1.32
Pftv160	160	9097.17	8907.00	2.14	4.17	2	1.61
Pftv170	170	9373.11	9239.00	1.45	3.83	1	2.13
Prbg403	402	7905.22	7901.00	0.05	0.28	27	43.59
Prbg323	322	4060.63	4045.00	0.39	0.99	2	14.68
Prbg443	442	7838.90	7835.00	0.05	0.20	5	61.52
Prbg358	357	5211.36	5204.00	0.14	0.40	1	22.01

Each test problem has then been solved as many times as the number of cities n. In each run, we consider the home city 0 and a different city $i \in I$ to form the starting tour, as described in Module I. The computational results, collected in Table 1, indicate:

1. Problem: denominated as the original name preceded by "P" (for periodic);
2. n: number of cities of the set I;
3. Mean_val: mean objective value over n runs;
4. Best_val: best objective value obtained over n runs;
5. Mean_err: mean percentage relative error with respect to Best_val over n runs;
6. Max_err: maximum percentage relative error with respect to Best_val over n runs;
7. N_best: number of times in which the best solution has been obtained over n runs;
8. T_mean: mean time required for the solution of an instance of the test problem (in minutes on a Pentium III, 933 MHz, 256 MB RAM).

It is clear that, for these periodic test problems, the optimal solution is not known, and there is no exact algo-

TABLE 2. Computational results for a planning horizon $m = 1$.

Problem	n	Mean_val	Opt_val	Mean_err	Max_err	N_opt	T_mean
ft53	52	6935.69	6905.00	0.44	4.81	32	0.00
ft70	69	38773.19	38673.00	0.26	1.25	1	0.11
br17	16	39.00	39.00	0.00	0.00	16	0.00
ry48p	47	14593.81	14422.00	1.19	3.74	0	0.00
kro124p	99	36809.39	36230.00	1.60	5.96	0	0.42
p43	42	5621.40	5620.00	0.02	0.09	17	0.00
ftv33	33	1307.18	1286.00	1.65	8.09	21	0.00
ftv35	35	1481.26	1473.00	0.56	4.68	15	0.00
ftv38	38	1544.82	1530.00	0.97	4.51	17	0.00
ftv44	44	1636.70	1613.00	1.47	4.40	13	0.00
ftv47	47	1784.00	1776.00	0.45	4.28	8	0.00
ftv55	55	1638.84	1608.00	1.92	4.04	6	0.09
ftv64	64	1879.95	1839.00	2.23	5.38	3	0.10
ftv70	70	1982.74	1950.00	1.68	5.59	1	0.12
ftv90	90	1591.91	1579.00	0.82	10.64	21	0.36
ftv100	100	1812.06	1788.00	1.35	12.08	18	0.41
ftv110	110	2003.15	1958.00	2.31	7.41	3	0.58
ftv120	120	2200.09	2166.00	1.57	5.26	1	0.71
ftv130	130	2354.57	2307.00	2.06	6.68	0	0.69
ftv140	140	2483.59	2420.00	2.63	9.01	0	0.87
ftv150	150	2669.39	2611.00	2.24	8.77	2	1.18
ftv160	160	2742.37	2683.00	2.21	7.94	4	1.53
ftv170	170	2814.24	2755.00	2.15	8.78	0	2.06
rbg403	402	2465.00	2465.00	0.00	0.00	402	42.23
rbg323	322	1326.01	1326.00	0.00	0.08	318	14.71
rbg443	442	2720.00	2720.00	0.00	0.00	442	65.62
rbg358	357	1163.00	1163.00	0.00	0.00	357	23.89

TABLE 3. Computational results for the ptsp instances.

Problem	n	Mean_val	Best_val	Mean_err	Max_err	Min_err	N_best	T_mean
P1	51	436.17	432.10	0.94	2.56	0.00	3	0.00
P2	51	1113.48	1105.81	0.69	2.57	0.00	3	0.00
P3	51	472.15	466.71	1.17	4.26	0.00	3	0.07
P4	76	558.57	549.05	1.73	3.26	0.00	1	0.14
P5	76	1396.51	1382.61	1.01	1.86	0.34	0	0.10
P6	76	654.75	643.50	1.75	4.23	0.13	0	0.33
P7	101	655.83	629.96	4.11	5.90	2.20	0	0.40
P8	101	1638.38	1599.32	2.44	3.64	1.61	0	0.20
P9	101	735.00	715.70	2.70	4.04	1.49	0	0.99
P10	101	1260.46	1222.71	3.09	6.53	1.90	0	0.30
P11	66	490.97	490.97	0.00	0.00	0.00	65	0.00
P12	88	664.10	664.10	0.00	0.00	0.00	87	0.06
P13	110	830.80	830.80	0.00	0.00	0.00	109	0.10
P14	132	994.60	994.60	0.00	0.00	0.00	131	0.23
P15	154	1157.07	1157.07	0.00	0.02	0.00	152	0.40
P16	49	660.70	660.12	0.09	2.20	0.00	38	0.00
P17	67	776.84	776.43	0.05	0.48	0.00	57	0.10
P18	85	875.53	873.73	0.21	0.39	0.00	21	0.20
P19	103	958.67	958.51	0.02	0.29	0.00	76	0.40
P20	121	1034.08	1033.58	0.05	0.25	0.00	57	1.09
P21	78	1375.09	1375.08	0.00	0.12	0.00	76	0.00
P22	155	4423.79	4312.31	2.59	5.22	0.98	0	0.40
P23	232	8543.75	8387.26	1.87	5.88	−0.26	4	2.20
P24	49	2073.37	2064.84	0.41	1.58	0.00	4	0.01
P25	97	3244.70	3207.44	1.16	3.37	0.47	0	0.04
P26	145	4126.66	4054.12	1.79	4.00	0.65	0	0.20
P27	193	4632.30	4591.05	0.90	2.50	−0.14	2	0.64
P28	241	4738.77	4678.19	1.30	2.61	0.07	0	1.63
P29	289	5665.28	5586.23	1.42	3.23	−0.04	1	3.61
P30	73	4495.25	4443.30	1.17	2.41	0.27	0	0.00
P31	145	5477.04	5393.04	1.56	3.12	−0.02	1	0.30
P32	217	7432.74	7339.88	1.27	2.62	0.03	0	1.76
P33	289	8423.54	8328.09	1.15	2.70	−0.03	1	6.35

rithm in the literature to solve these self-generated PATSP test problems. For this reason, we have selected the minimum value among all the n obtained solutions that we have used as reference to compute the mean and maximum percentage relative errors.

From the results shown in Table 1, we can note that the values of the obtained solutions depend on the choice of the starting tour. The best value has been obtained, indeed, only once for 10 test problems, and only a few times for the other test problems. However, the mean and maximum percentage relative errors remain relatively small for most of the test problems. Moreover, the CPU time remains relatively short across all the test problems, and starts to increase (still reasonably) only as the number of cities increases markedly.

The question remains about the quality of the solution produced by our heuristic: how far is the best obtained solution with respect to the optimal one? Because it is not possible to attack this question directly, we split the answer into two parts: one related to the asymmetric aspect of the problem and the other related to the periodic aspect. In the first case, we have reduced the planning horizon to a single day, i.e., $m = 1$, and we have solved the same test problems under the constraint that each city should be visited once within that day. The problem is thus reduced to the classical ATSP for which we know the optimal objective values. The computational results are summarized in Table 2 (here the optimal solution "Opt_val" substitutes "Best_val", and "N_opt" indicates how often the optimal solution is found).

The results of Table 2 show that our heuristic finds the optimal solution for most of the test problems. The heuristic fails, indeed, in only five occasions to reach the optimum and finds only suboptimal solutions. Moreover, the mean percentage relative error values remain relatively small for the whole set of the test problems.

It is also worthwhile noting that, even though the solution quality continues to depend on the starting tour for most of the test problems, this does not happen with test problems br17, rbg403, rbg443, and rbg358. In these cases, our algorithm finds the optimal solution in all the n executed runs independently from the choice of the starting tour.

The last computational experiments carried out have the objective of checking the quality of the solution with respect to the periodicity aspect. To do this we have used the heuristic to solve, as special cases of the PATSP, PTSP test problems that are known in the literature. Test problems P1–P10 have been proposed in [3], test problems P11–P23 have been proposed in [1], and test problems P24–P33 have been proposed in [4]. To have comparable results with the best-known solution value ("Best_val") for these test problems, it is necessary to add to our model the nonempty tour constraints that we have not included in the original formulation. These constraints, which are generally considered in the solution of the PTSP, impose that the salesman should not have any empty tour in all the days of the planning horizon. The collected results are reported in Table 3. Here, "Min_err" stands for the minimum percentage relative error with respect to "Best_val" over n runs, and assumes a negative value whenever a better value than "Best_val" is

reached. With this respect, "N_best" will indicate the number of times in which we obtain "Best_val" or a lower value.

The results of Table 3 show the good performance of the heuristic even in the solution of the PTSP test problems. For most of the problems, indeed, the best-known solution has been found, and in some cases we have improved this value and found a better objective value (P23, P27, P29, P31, and P33). In all the other cases the error values stay under an acceptable threshold as well as the mean computational time.

5. CONCLUDING REMARKS

In this article, we have presented and discussed the asymmetric traveling salesman problem with periodicity constraints. We have proposed a heuristic algorithm based on a tour construction procedure followed by an improvement phase. The effectiveness of the proposed heuristic has been tested on a set of self-generated periodic problems. Moreover, we have proved its efficiency by using a special class of periodic test problems for which an optimal or suboptimal solution is known *a priori.* In all the cases, the mean and maximum percentage relative errors remain relatively small, and the execution time is still within reasonable limits.

Acknowledgments

We wish to thank both the editor and the referees for their constructive comments and recommendations, which have significantly improved the presentation of this article.

REFERENCES

[1] I.-M. Chao, B.L. Golden, and E. Wasil, A new heuristic for the period traveling salesman problem, Comput Ops Res 22 (1995), 553–565.

[2] I.-M. Chao, B.L. Golden, and E. Wasil, An improved heuristic for the period vehicle routing problem, Networks 26 (1995), 25–44.

[3] N. Christofides and J.E. Beasley, The period routing problem, Networks 14 (1984), 237–256.

[4] J.-F. Cordeau, M. Gendreau, and G. Laporte, A tabu search heuristic for periodic and multi-depot vehicle routing problems, Networks 30 (1997), 105–119.

[5] M. Fischetti and P. Toth, A polyhedral approach to the asymmetric traveling salesman problem, Manage Sci 43 (1997), 1520–1536.

[6] M. Gaudioso and G. Paletta, A heuristic for the periodic vehicle routing problem, Transport Sci 26 (1992), 86–92.

[7] G. Paletta, A multiperiod traveling salesman problem: heuristic algorithms, Comput Ops Res 19 (1992), 789–595.

[8] G. Paletta, The period traveling salesman problem: a new heuristic algorithm, Comput Ops Res 29 (2002), 1343–1352.

[9] D.J. Rosenkrantz, R.R. Stearns, and P.M. Lewis, An analysis of several heuristics for the traveling salesman problem, SIAM J Comput 6 (1977), 563–581.

[10] TSPLIB, http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95.