The 2-period balanced traveling salesman problem

Article · February 2007

Source: RePEc

CITATIONS
2

READS
593

2 authors, including:

Francesco Mason
Università Ca' Foscari Venezia
22 PUBLICATIONS **102** CITATIONS

SEE PROFILE

Department of Applied Mathematics, University of Venice

WORKING PAPER SERIES

Tatiana Bassetto, Francesco Mason

The 2-period Balanced Traveling Salesman Problem

Working Paper n. 154/2007
October 2007

ISSN: 1828-6887

The 2- period Balanced Traveling Salesman Problem

Tatiana Bassetto
tbassetto@unive.it
Dept. of Applied Mathematics
University of Venice

Francesco Mason
fmason@unive.it
Dept. of Applied Mathematics
University of Venice

Abstract. In the 2-period Balanced Traveling Salesman Problem (2B-TSP), the customers must be visited over a period of two days: some must be visited daily, and the others on alternate days (even or odd days); moreover, the number of customers visited in every tour must be 'balanced', i.e. it must be the same or, alternatively, the difference between the maximum and the minimum number of visited customers must be less than a given threshold. The salesman's objective is to minimize the total distance travelled over the two tours. Although this problem may be viewed as a particular case of the Period Traveling Salesman Problem, in the 2-period Balanced TSP the assumptions allow for emphasizing on routing aspects, more than on the assignment of the customers to the various days of the period. The paper proposes two heuristic algorithms particularly suited for the case of Euclidean distances between the customers. Computational experiences and a comparison between the two algorithms are also given.

Keywords: period routing problem, period traveling salesman problem, logistic, heuristic algorithms.

JEL Classification Numbers: C61.

MathSci Classification Numbers: 90B06, 90C59.

0 Introduction.

In the organization of picking up orders for commercial firms, it often happens that an agent (or vehicle) has to visit his/her customers (or cities) periodically but at different times on the grounds of the number and the frequency of the orders. The simplest, but frequent, case is when the set V of customers can be divided into two sets over a period of two days: customers visited daily (set D) and customers visited only once in the period (set S). This way, two tours must be constructed in such a way that both contain all the elements in D, while the customer in S are partitioned between them.

Moreover, we wish the agent to work every day almost the same time amount: this way, we also impose some balancing constraints on the two tours.

This can be achieved in different ways: we can require that the difference between the number of customers, which are visited in each day, must be below a given threshold or, in a multi-objective setting, it must be minimised.

In this paper we particularly consider the case in which the two tours contain the same number of customers or they can differ by one.

The objective is to minimise the total distance travelled in the two days.

The (non-balanced) 2-period TSP may be viewed as a particular case of the Period Traveling Salesman Problem (PTSP) [5]. This last problem generalizes the TSP by extending the planning period to *p* days. In PTSP every customer must be visited a specified number of times: in many cases for each customer a set of feasible (allowable) combinations of visit days is also given. The aim is to build *p* routes (one route for every day) in order to minimize the total covered distance: obviously, PTSP and 2-period TSP are NP-hard problems.

The literature on periodic routing problems is not very extensive.

The earliest work can be considered the paper [6], by Christofides and Beasley (1984). In this paper, the authors propose solution approaches for the routing problem over more than one day, although the purpose is to study the PVRP (Period Vehicle Routing Problem), i.e. a problem with weighted nodes and a capacity constraint on the vehicles. They propose two heuristic algorithms which make use of the solution of other NP-hard problems, i.e. the Period Median Problem and the well known Travelling Salesman Problem.

Other heuristic approaches were subsequently proposed in 1992 by Paletta [15], and Chao *et al.* [5]; then, in 1997, Cordeau, Gendreau and Laporte [8] present a tabu search technique; in 2002 Paletta [16] presents a new heuristic algorithm for the PTSP, improved in Bertazzi, Paletta and Speranza [3] in 2004. Other works are involved with PVRP [9] and with asymmetric PTSP [17].

In PTSP there are two interrelated problems: an assignment problem, because for each customer a feasible combination of visit days must be chosen, and then a routing problem, in order to find the best tour that visits the customers of each particular day of the period. These problems are often solved subsequently: then improvement-exchange procedures are used to get a better solution.

Butler, Williams and Yarrow, in 1997 [4], introduce the 2-Period TSP. The authors solve exactly a particular case study applied to milk collection in Ireland. Their work appears to be independent from the paper of Christofides and Beasley [6] (which they do not quote). After introducing the problem, they give an integer programming formulation. The procedure they suggest is an exact approach to the problem by a combination of cuts and Branch and Bound. However, in their development, as the authors say, decisions ‘on line’ about the introduction of constraints must be made.

Balancing constraints are never explicitly considered (both in PTSP and 2-period TSP): in [5], in the assignment phase, at the beginning of the

algorithm, a uniform distribution of customers to the visit days is searched for. But in the improvement steps, the solution can became not balanced. In [6], a constraint on the maximum length of each route is given. In [4] the solution to the particular case gives two tours, having respectively 24 and 31 nodes (including the depot: the nodes to be visited daily are 12).

The 2-period balanced TSP is the particular case of the PTSP in which the period consists of two days (p = 2) and, in this way, two tours T^1* and T^2* must be built: differently from the (more general) PTSP, in the 2-period balanced TSP there is not a set of different combinations of visit days.

In this paper we propose two heuristics particularly suited for the case in which the underlying graph satisfies the triangular property.

The paper is divided into two sections: in the first one, after giving some notations and definitions, the 2-period balanced TSP is formulated as an integer programming problem; in the second one, the two heuristic algorithms are described. Last, conclusions are traced.

1 An integer programming model for the 2BTSP.

Let $G = (V, E)$ a complete graph of n nodes ($n > 1$) without loops. Let c_{ij} be the weight of the edge (i, j). In the 2-period balanced TSP the set V can be partitioned into two (disjoint) subsets:

- the set of single-nodes $S = \{s_1, s_2, \dots, s_k\}$, i.e. the ones to visit once over two days;
- the set of double-nodes $D = \{d_1, d_2, \dots, d_h\}$, i.e. the ones to visit every day.

Obviously, $h + k = n$.

In [4] a depot is also defined: here, for sake of simplicity, and without lack of generality, we include it in the set D of double-nodes. This way, $D \neq \varnothing$.

In what follows, a node belonging to the set or tour X, will be called X-node. Besides this, given a set A, we shall denote with $|A|$ its cardinality and with $\lfloor x \rfloor$, x being a real number, the greatest integer $\leq x$.

We want to build two tours, T^1 and T^2 (one for every day), which satisfy a balance constraint, in order to minimize the total travelled distance. Both tours visit all the D-nodes, while every single-node, i.e. every S-node, can be inserted only in one of the two tours, T^1 or T^2.

This way, in every feasible solution, S is partitioned into two subsets, S^1 and S^2 (with $S = S^1 \cup S^2$ and $S^1 \cap S^2 = \varnothing$), the first one made up of nodes visited on the first (or odd) day and the second one on the second (or even) day.

Let k_1 be the cardinality of S^1 and k_2 the cardinality of S^2 , so that $k_1 + k_2 = k$. T^1-nodes constitute the set $D \cup S^1$, while T^2-nodes are $D \cup S^2$.

Balance constraints can be formulated in several way: we use a parameter g^* which represents the maximum allowed difference between the number of customers to be visited on the first and on the second day. Thus, we must have:

$$/k_1 - k_2 / \leq g^*.$$

Letting $g^* = 1$, then in a feasible solution the number of visited customers must be equal in the two days (if k is even), or it can differ by one unit (when k is odd).

The problem can be formulated as an integer linear programming one introducing the following Boolean variables:

- $x_{ijq} = 1$ if and only if the customer j is visited immediately after i on the q-day and 0 otherwise, ($q = 1$ or 2);
- $y_{iq} = 1$ iff the customer i is visited on the q-day and 0 otherwise (for every node $s_i \in S$ and $q = 1$ or 2).

The 2-period balanced TSP formulation is:

$$Min \sum_{(i,j)\in E} \sum_{q=1}^{2} c_{ij} x_{ijq} \qquad (1)$$

s.t.

$$\sum_{j\in V} x_{ijq} = 1 \qquad \forall i \in D; \forall q \qquad (2)$$

$$\sum_{j\in V} x_{jiq} = 1 \qquad \forall i \in D; \forall q \qquad (3)$$

$$y_{i1} + y_{i2} = 1 \qquad \forall i \in S \qquad (4)$$

$$\sum_{j\in V} x_{ij1} = \sum_{j\in V} x_{ji1} = y_{i1} \qquad \forall i \in S \qquad (5)$$

$$\sum_{j\in V} x_{ij2} = \sum_{j\in V} x_{ji2} = y_{i2} \qquad \forall i \in S \qquad (6)$$

$$\sum_{i,j\in Z} x_{ijq} \leq |Z| - 1 \qquad Z \subseteq S \cup D; \forall q \qquad (7)$$

$$\left| \sum_{i\in S} y_{i1} - \sum_{i\in S} y_{i2} \right| \leq g^* \qquad (8)$$

$$y_{iq} \in \{0,1\} \qquad \forall i \in S; \forall q \qquad (9)$$

$$x_{ijq} \in \{0,1\} \qquad \forall (i,j) \in E; \forall q \qquad (10)$$

The objective function minimizes the total costs.

Constraints (2) and (3) impose that all double nodes $i \in D$ are visited every day, both for odd ($q = 1$) and for even ($q = 2$) days.

Constraints (4) guarantee that every single-node $i \in S$ is visited only once, either on even or on odd days; next constraints (5) and (6) compel the existence of only one outside edge and one inside edge for any node $i \in$ S.

Constraints (7) are classical sub-tour elimination constraints of the TSP, Z being a subset of nodes of the graph G (see [12] and [13]).

Inequalities (8) represent the 'balance constraints': they impose an upper bound to the difference between the number of nodes (customers) that can be visited by a vehicle in each day.

There are two extreme cases: when D contains only the depot, the problem becomes a VRP in which customers have unit demand and there are two vehicles, each of them associated to a day, with capacity $\lfloor (k+g^*)/2 \rfloor$; on the other hand, if $S = \varnothing$ (so that every node has to be visited every day) the problem becomes a TSP.

Obviously, the 2-period balanced TSP is a NP-hard problem (see [6], [8], [13]).

2 Two heuristic techniques for 2BTSP.

The core of the 2-period balanced travelling salesman problem is how to partition optimally the customers in S into the two sub-sets S^1 and S^2, to be attributed, respectively, to the two days of the period. This appears the crucial point, because softwares now available allow to solve the subsequent Travelling Salesman Problems, in $D \cup S^1$ and $D \cup S^2$, in an exact way, at least for instances with some hundreds of nodes (a good example is constituted by Concorde). This encourages the use of exact subroutines which solve TSP as a step in the achievement of an approximate solution of the 2-balanced period TSP.

Even the non balanced version of the period TSP appears quite difficult to solve in an exact way: as we pointed out above, Butler, Williams and Arrows, in [4], propose a solution for a particular problem of 42 nodes, but they do not give a fully automatised procedure. In our experience, branch and cut takes a too long time also for moderate size instances. So approximate algorithms are useful.

We propose two heuristics, that we shall call A1 and A2, respectively, for the case in which, in the balancing constraint, $g^*=1$. The two procedures, particularly A2, work if G is a geometric graph or if it satisfies the triangular property.

A1 is a very simple technique which quickly provides a feasible good solution.

The second algorithm, A2, taking into account the possibility of 'edge crossing' (i.e., "two different edges cross in a graph drawing if their geometric representations intersect" [1]), gives a feasible solution, often better than the one obtained by A1 but at a greater computational cost.

A1 and A2 both require, as a prerequisite, a hamiltonian cycle GT over all the nodes in V (General Tour): in practice, GT can be obtained by well known softwares, once more, for instance, Concorde.

In this section first we describe the algorithm A1, then we motivate it; finally, we introduce the algorithm A2 on the basis of some observations concerning A1.

A1 algorithm

Step 1. Choose a visit direction on the circuit GT. Choose also a single node *s*. Put $S^1 = \{s\}$. If *k* is even, go to step 2; otherwise go to step 3.

Step 2. Add to S^1 the *(k/2) – 1* subsequent single nodes following *s* in the chosen order of visit in GT. Put in S^2 the *k/2* (following) remaining single nodes. Go to step 4.

Step 3. Add to S^1 the *[(k-1)/2]* single nodes following *s* in the chosen order of visit in GT. Put in S^2 the remaining *(k-1)/2* single nodes. Go to step 4.

Step 4. Solve the TSP both in $S^1 \cup D$ and in $S^2 \cup D$. If there are other single nodes not yet considered as first node, choose one and go back to step 1. Otherwise go to step 5.

Step 5. Choose the best solution between the ones given from the procedure. STOP.

A1 is motivated by the following considerations.

Given a region R of area *R*, in which *N* customers are uniformly distributed, provided *N* is sufficiently large, i.e., at least a few dozens, a well known formula (see for example [7]) gives a good approximation for the total distance covered *L* in a TSP. This value is the square root of the product *RN*, multiplied by a constant *c*, which depends on the metrics we use:

$$L = c\sqrt{RN}$$

To be true, in this formula *(N+1)* should be used, instead of *N*, but for large *N* this can be disregarded.

In our problem, due to the balance constraints, we must visit every day all the double nodes and (with an approximation, at most, of one unit!) one half of the total number of single nodes.

Let us divide R into two compact sub-regions, R_1 and R_2 having the same area. We must decide how many single nodes are to be visited on the first day from each sub-region in such a way that the total number of single nodes chosen is k_1. Of course, the k_2 single nodes, which are not yet visited on the first day, must be included in the second tour.

Introducing a parameter α, which represents the fraction of nodes visited in region R_1, the distance travelled during the first day can be approximated by the following formula:

$$L = c\sqrt{(R/2)((h/2)+\alpha k_1)} + c\sqrt{(R/2)((h/2)+(1-\alpha)k_1)}$$

Obviously, the same formula is valid for the distance travelled during the second day, replacing k_1 with k_2.

Due to the concavity of the square root function, it is easy to see that the minimum of L is attained when $\alpha = 0$ or $\alpha = 1$. This suggests to visit all the single nodes which belong to a sub-region on a day, and to visit the remaining ones (in the other sub-region) on the other day.

From an operative point of view, we must give a rule to define the two sub-regions.

To this aim, experience shows that sub-paths of GT are very often used in the optimal tours for each day. This, in turn, suggests to divide S in two subsets putting in the same set $\lfloor (k+1)/2 \rfloor$ single points which are consecutively visited in GT: these points are roughly contained in a sub-region having, more or less, half the area of the whole region containing all the points.

This way, we have the possibility to construct more different solutions, depending on which is the first single node of the $k/2$ consecutive ones to be visited on a particular day: it should be noted that, when k is even, the number of different solutions is exactly $k/2$; but when k is odd, then the number of different solutions (due to the difference between the numbers of customers visited each day) grows up to k.

Finally, once we have determined these two sub-sets S^1 and S^2, we solve two TSPs, one for the first day and the other for the second one.

A2 algorithm

The algorithm A2, taking into account the triangular inequality, looks for a feasible solution often better than the one obtained by A1. Consider, for example, the case of the graph in the following figure:

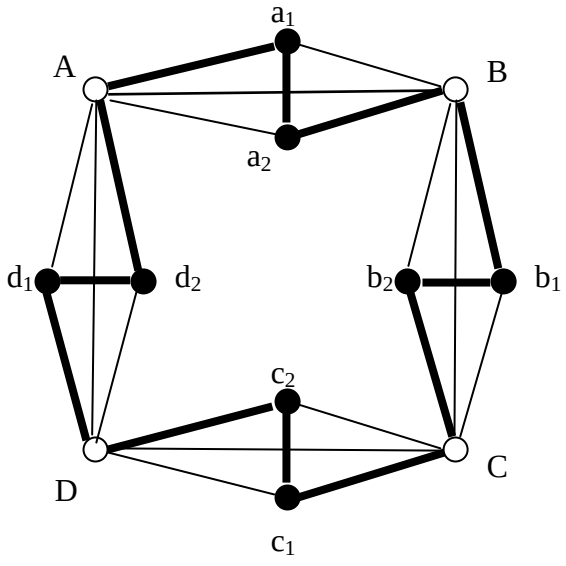

Fig. 1 A graph and a shortest Hamiltonian Circuit.

Black nodes are single ones and white nodes are double ones. One (minimum cost) General Tour is clearly the one in heavy lines. A typical solution given by the algorithm A1 is: a first tour which visits all the nodes from A to C (in the same order as in the GT), then it visits D and comes back to A; a second tour, which is symmetric with respect to the first one (see fig.2: continuous and dotted lines).

However, this solution is not optimal, because the triangular property is not adequately taken into account. Focusing the attention on paths from A to B, it is clear (see once more fig. 1) that the length of the path {A, a_1, a_2, B} on the first day plus the length of the edge (A, B) on the second day is greater than the length of the path {A, a_1, B} (to be run on the first day) plus the length of {A, a_2, B} (to be attributed to the second day).

This way, in an optimal solution, the two single nodes, a_1 and a_2, between A and B must be visited on different days (the same is true for every couple of adjacent single nodes in GT; see fig.3).

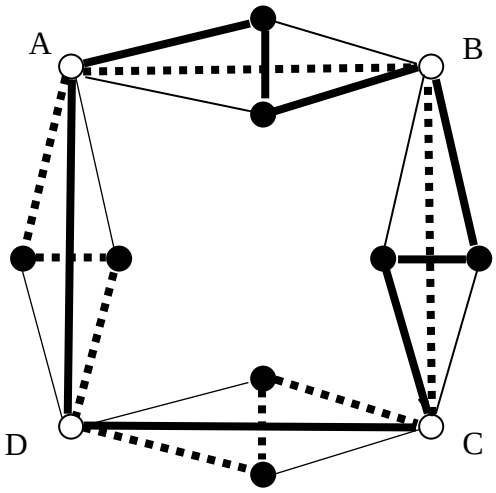

Fig 2.The two tours obtained with A1.

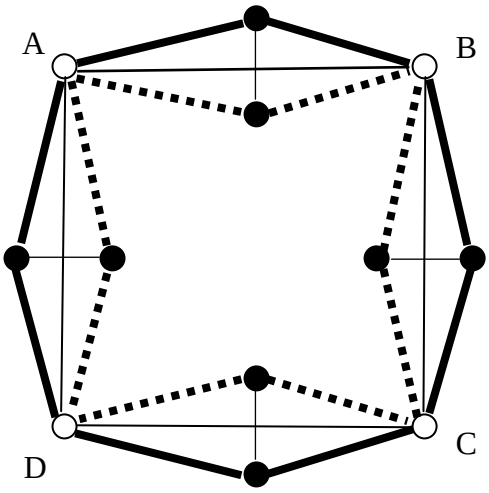

Fig 3. The two tours in an optimal solution.

More generally speaking, here the crucial point is the following: if a path in GT linking two double nodes, say u and v, and containing some single nodes, crosses the chord (u, v), the single nodes on one side of this chord must be visited in a different day with respect to the ones on the other side. This is the most important point in A2.

Before giving A2 in detail, some more definitions are useful.

Let "single-node path" $P(u, v)$, SNP for short, be any path in GT having node set $\{u, s_1, s_2, \ldots s_{r-1}, v\}$, and $r \geq 2$ edges, in which the two endpoints, u and v, are double-nodes, while nodes, $s_1, s_2, \ldots s_{r-1}$ are single.

Single node paths are univocally defined whenever G contains more than two double nodes. In the case in which there are only two double nodes, (including the depot) we have two single node paths: they are distinguishable introducing a visit order in GT.

Let the *cardinality* of a SNP be the number of nodes in the path and indicate it by $|P(u, v)|$.

As well as in the case of A1, in A2 also we must choose a visit direction for GT. In our computational experiences, we introduced the visit order in which the nodes of the convex hull of G are met clockwise.

The algorithm A2 consists of two phases. In the first one, every single node path which crosses its respective chord is analysed and its nodes are divided into two subsets; then, a partition problem is solved, step by step, inserting single nodes into two graphs**,** G^1 and G^2, i.e. the graphs containing the nodes to be visited respectively in the first and the second day, in such a way that the balancing constraint is fulfilled. In this phase some heuristics are required in order to solve this partition problem. In the second phase the TSP is solved both in G^1 and G^2 in order to find the best tours T^1* and T^2*.

A2 algorithm

Step 0. Choose a visit direction in GT. Let $G^1 = G^2 = (D, \varnothing)$. List all the SNP's in GT as $P_1(u_1, v_1), P_2(u_2, v_2), \ldots, P_t(u_t, v_t)$. Go to step 1.

Step 1. For $i = 1, \ldots t$, consider the SNP $P_i(u_i, v_i) \subseteq$ GT. Test if any edge $(s_1^i, s_2^i) \in P_i(u_i, v_i)$ crosses the chord (u_i, v_i) by the crossing-edge test: if yes, put $P_i(u_i, v_i)$ in a set W of 'crossing SNP's'. Otherwise, put $P_i(u_i, v_i)$ in a set Z of 'non-crossing SNP's'. Re-number SNP's, both in W and in Z. Let $w = |W|$ and $z = |Z|$. (We will suppose, without lack of generality, $W \neq \varnothing \neq Z$. In case this would not be true, the modifications to the algorithm are quite evident). Go to step 2.

Step 2. For $i = 1, \ldots w$, consider the SNP $P_i(u_i, v_i) \subseteq W$. Delete in $P_i(u_i, v_i)$ every edge (s_1^i, s_2^i) which crosses the chord (u_i, v_i).
Let $P_i^1, P_i^2, \ldots P_i^m$ be the disjoint sub-paths $\subset P_i(u_i, v_i)$ (possibly consisting of only one node) in the sequence in which they are visited in $P_i(u_i, v_i)$.

Insert single nodes which belong to sub-paths P_i with odd apex into a set W_{iL} and the ones with even apex into a set W_{iR}. Let $w_{iR} = /W_{iR}/$, $w_{iL} = /W_{iL}/$. Go to step 3.

Step 3. For $i = 1,\ldots z$, consider the SNP $P_i(u_i, v_i) \subseteq Z$. Insert single nodes of this path in a set Z_i. Let $z_i = /Z_i/$. Go to step 4.

Step 4. Solve the following partition problem

$$
\begin{array}{lll}
(P) & Min \quad \Sigma_{i=1}^{w} (x_{iL}w_{iL} + x_{iR}w_{iR}) + \Sigma_{i=1}^{z} y_i z_i & \\
\text{s.t.} & \Sigma_{i=1}^{w} (x_{iL}w_{iL} + x_{iR}w_{iR}) + \Sigma_{i=1}^{z} y_i z_i \geq k/2 & (*) \\
 & x_{iL} = 1 - x_{iR} & (**) \\
 & x_{iL}, \; x_{iR}, \; y_i \in \{0, 1\}. &
\end{array}
$$

Insert into G^1 the single nodes of the sets W_{iL}, W_{iR}, Z_i for which the corresponding variable, in the optimal solution, is equal to 1. Insert into G^2 the remaining single nodes. Insert both into G^1 and G^2 all the double nodes. Go to step 5.

Step 5. Solve (twice) the TSP both in G^1 and G^2: call, respectively, T^1* and T^2* the two optimal tour. Go to step 6.

Step 6. If the (absolute) difference between $|G^1|$ and $|G^2|$ is $\leq g*$, i.e. if the balancing constraint is satisfied, STOP. If the balancing constraint is not fulfilled, go to step 7.

Step 7. Let w.l.o.g. $|G^1| > |G^2|$. For every single-node s in G^1, consider its two neighbourhood nodes in T^1*: let them be called $a(s)$ and $b(s)$. Compute the transfer-cost $r(s)$ given by

$$r(s) = c_{a(s),b(s)} - c_{s,a(s)} - c_{s,b(s)} + min_{(i,j)} \; c_{is} + c_{sj} - c_{ij},$$

where the minimum has to be computed with respect to all the edges $(i, j) \in T^2*$. Go to step 8.

Step 8. Find the single-node $s*$ with the minimum transfer-cost. Transfer $s*$ from G^1 into G^2. Go to step 5.

A2 requires the resolution of the partition problem *(P)*. In the formulation, constraints (**) guarantees that, for every crossing SNP P(u, v), the single nodes on opposite sides with respect to the chord linking u and v will be visited in different days.

(P) can be reduced to the following subset sum problem:

$$(P') \qquad \text{Min} \quad \Sigma_{i=1}^{w} x_i \, \overline{w}_i + \Sigma_{i=1}^{z} y_i z_i$$

$$\text{s.t.} \quad \Sigma_{i=1}^{w} x_i \, \overline{w}_i + \Sigma_{i=1}^{z} y_i z_i \geq \overline{k} / 2$$

$$x_i, \; y_i \in \{0, 1\},$$

with the substitution:

$$\overline{w}_i = |w_{iL} - w_{iR}| \qquad \overline{k} = k - \Sigma_{i=1}^{w} \min (w_{iL}, w_{iR}).$$

(P') can be solved by different techniques (see [14] for an exhaustive analysis).
A feasible solution to *(P)* can be built from the one to *(P')* as follows. If in the optimal solution $x_i = 1$, put in G^1 the nodes of the set W_{iL} iff $|W_{iL}| > |W_{iR}|$; otherwise put in G^1 the nodes in W_{iR}. (Note that if $|W_{iL}| = |W_{iR}|$ then $\overline{w}_i = 0$ and it is indifferent to put the nodes of W_{iL} in G^1 or in G^2 but, anyway, not in the same set where nodes of W_{iR} are inserted!).

The algorithm A2, in order to satisfy the balancing constraints (8), firstly partitions all the single-nodes paths in two sets, W and Z. Then, it partitions the nodes of the SNP's in the subsets W_{iL}, W_{iR} and Z_i. Note that, while the elements of W and Z are paths, the elements in the sets W_{iL}, W_{iR} and Z_i are (single) nodes. The aim is to build the two sets of nodes, G^1 and G^2 in such a way that the difference of cardinality between G^1 and G^2 is as close as possible to zero. In step 2, the sets W_{iL} and W_{iR} contain, respectively, the single nodes that lie by the same side with respect to the path from u_i to v_i.

The complexity of the algorithm A2 depends first of all on the resolution of TSP. Excluding from consideration the TSP, the complexity depends on [9]:

- the complexity of "crossing-edge test" (step 1), i.e. $O(\mathbf{n^2})$;
- the complexity of the insertion cost of a node (steps 9 and 10), i.e. $O(\mathbf{n^2})$.

Observation. From a practical point of view, the best solution cost given by the first algorithm is less than twice the cost of GT.

3 Computational experiences.

In order to compare the two heuristics, we considered 10 random instances, each of them consisting in a graph of 48 nodes, of which 16 were double. So, every day, 32 nodes must be visited.

We number these instances as I_1, I_2, …, I_{10}. The instances were drawn using Concorde.

In each instance, and both for A1 and A2, we distinguished the case in which the obtained tours T^1* and T^2* 'crossed each other' or not, i.e., there exists (at least) one couple of edges, one in T^1* and the other in T^2*, which cross each other.

In the 'crossing' situation, we performed a post-analysis, in order to improve he solution, taking account of the triangular property. This was done by an exchange technique, moving sections of single node paths from a tour to the other one. This operation gave a better solution in many cases and changed the performance of one algorithm with respect to the other: it must be said, however, that the (happily limited) dimensions of the instances allowed a visual analysis and consequent choice of the nodes to be exchanged. Moreover, the subset sum problems to be solved were easily handled: in most cases they have multiple solutions (and the choice between them was a 'problem inside the problem)! Unfortunately, it was not possible to compare the obtained solutions with the true optima.

The results we obtained are reported in the following table: the value of the best solution is in heavy line. Stars denote the best solution which we obtained without using improvement techniques.

	A1			**A2**		
Istances	**Value**	**crossing**	**improved sol.**	**value**	**crossing**	**Improved sol.**
I_1	338.04*	yes	**331.51**	340.29	no	-
I_2	**309.29***	no	-	311.05	no	-
I_3	305.96*	yes	305.49	311.98	yes	**303.14**
I_4	290.85	yes	286.86	287.13*	yes	**285.58**
I_5	273.23*	no	-	274.25	yes	**272.07**
I_6	330.65	yes	329.58	**329.15***	no	-
I_7	334.89	yes	**325.61**	331.44*	no	-
I_8	311.13	yes	**307.51**	309.74*	no	-
I_9	352.58*	yes	**346.97**	353.35	yes	350.76
I_{10}	316.28*	yes	**313.95**	317.99	no	-

The table shows that it is not possible to say which of the two proposed algorithms is to be preferred. But we can note that the algorithm A2 behaves better, almost always, only after the elimination of 'edge crossing'.

Finally, we do not give solution times: generally speaking, they obviously depend on the number of TSPs to be solved in both algorithms. In the studied instances, few seconds were required: A2 is a little bit faster in that it requires a lesser number of these steps.

4 Concluding remarks.

In this paper we have introduced, formulated and solved the Single-Double Balanced TSP. Two heuristic algorithms are proposed for its solution.

The performance of the heuristics depends above all on the quality of the initial solution and on the chosen improving methods. Different improving methods lead to algorithms more or less effective according to objectives; this way the computational cost can change very deeply.

An interesting generalization of the problem, that can have also practical application, is concerned with the utilization of more vehicles and of different cost functions linked to the visit of nodes. This can be matter of future research.

References

[1] M.J. Atallah (1999) *Algorithms and Theory of Computation Handbook*, CRC.

[2] S. Baptista, R.C. Oliviera, E. Zùquete (2002) "A period vehicle routing case study". *European Journal of Operational Research* **139**, 220-229.

[3] L. Bertazzi, G. Paletta, M.G. Speranza (2004), "An improved heuristic for the period traveling salesman problem". *Computers and Operations Research* **31**, 1215-1222.

[4] M. Butler, H.P. Williams, L-A.Yarrow (1997) "The two-period travelling salesman problem applied to milk collection in Ireland". *Computational Optimization and Applications* **7** n° 3, 291 – 306.

[5] I.M. Chao, B.L. Golden, E.A. Wasil (1995) "A new heuristic for the period traveling salesman problem". *Computers and Operations Research* **22**, 553-565.

[6] N. Christofides, J.E. Beasley (1984) "The period routing problem". *Networks* **14**, 237-256.

[7] N. Christofides, S. Eilon (1969) "Expected distances in distribution problems". *Operational Research Quarterly* **20**, n°4, 437-443.

[8] J.F. Cordeau, M. Gendreau, G. Laporte (1997) "A tabu search heuristic for periodic and multi-depot vehicle routing problems". *Networks* **30**, 105-109.

[9] M. Gaudioso, G. Paletta (1992) "A Heuristic for the Periodic Vehicle Routing Problem". *Transportation Science* **26** n° 2, 86 – 92.

[10] G. Ghiani, R. Musmanno, G. Paletta, C. Triki (2005) "A heuristic for the periodic rural postman problem". *Computers and Operations Research* **32**, 219-228.

[11] B. Korte, J. Vygen (2000). *Combinatorial optimization. Theory and algorithms*. Springer.

[12] G. Laporte (1992) "The traveling salesman problem: an overview of exact and approximate algorithms". *European Journal of Operational Research* **59**, 231-247.

[13] E.L. Lawler, J.K. Lenstra, H.G. Rinnooy Kan, D.B. Shmoys (1985) *The traveling salesman problem: a guided tour of combinatorial optimization.* Wiley, New York.

[14] S. Martello, P. Toth (1990) *Knapsack problems.* Wiley, Chichester.

[15] G. Paletta (1992) "A multiperiod traveling salesman problem: heuristic algorithms". *Computers and Operations Research* **19**, 789-795.

[16] G. Paletta (2002) "The period traveling salesman problem: a new heuristic algorithm". *Computers and Operations Research* **29**,1343-1352.

[17] G. Paletta, C. Triki (2004) "Solving the asimmetric traveling salesman problem with periodic constraints". *Networks* **44**, n° 1, 31-37.

[18] C.H. Papadimitriou, K. Steiglitz (1998) *Combinatorial optimization. Algorithms and complexity.* Dover Publications, New York.