The multi-vehicle covering tour problem: building routes for urban patrolling

W. A. Oliveira\textsuperscript{a}\textsuperscript{*}, M. P. Mello\textsuperscript{b}, A. C. Moretti\textsuperscript{c} and E. F. Reis\textsuperscript{a}

\textsuperscript{a}Department of Mathematics, Federal Technological University of Paraná, PR, Ponta Grossa, Brazil
\textsuperscript{b}Department of Applied Mathematics, Institute of Mathematics, Statistics and Scientific Computing, University of Campinas, Campinas, SP, Brazil
\textsuperscript{c}School of Applied Science, University of Campinas, Limeira, SP, Brazil

E-mail: woliveira@utfpr.edu.br [Oliveira]; margarid@ime.unicamp.br [Mello];
moretti@ime.unicamp.br [Moretti]; edneif@utfpr.edu.br [Reis]

Abstract

In this work we study a particular aspect of the urban community policing: routine patrol route planning. We seek routes that guarantee visibility, as this has a sizable impact on the community perceived safety, allowing quick emergency responses and providing surveillance of selected sites (e.g., hospitals, schools). The planning is restricted to the availability of vehicles and strives to achieve balanced and short routes. We study an adaptation of the model for the multi-vehicle covering tour problem, in which a set of locations must be visited, whereas another subset must be close enough to the planned routes. It constitutes an NP-hard integer programming problem. Suboptimal solutions are obtained with several heuristics, some adapted from literature and others developed by us. The heuristics aim to construct short routes so that one could make several rounds during a work shift. We solve randomly generated problems and a problem with real data.

Keywords: Integer programming problems, vehicle routing, covering, heuristics

1 Introduction

Urban police patrolling aims to serve several objectives: crime prevention, quick response to emergencies, surveillance of public buildings, etc. The two most common ways of organizing patrol car operations are the allocation of a car to a certain geographical area and the allocation of a car to a certain route. The second is the method usually chosen by the municipal guards and military police in the state of São Paulo, Brazil. The planning of these routes rarely adopts a scientific approach, empirical rules are used instead.

\textsuperscript{*}Corresponding author
Using a fleet of known size, we consider how to efficiently construct routes in order to adequately patrol a given geographical area. The routes should satisfy the following criteria: a certain set of sites must be included in the routes; a second set may or may not belong to the routes; a third set is made up of sites that must be covered, in the sense that these sites are not visited, but they must be close enough to at least one visited site; the number of routes must equal the number of available vehicles; the routes must be balanced, that is, the number of visited sites for each route is approximately the same; all routes must start and finish at the same point, denoted by base of operations.

The problem is modeled as an integer programming problem whose objective is to minimize the overall length of all routes. The model is a modification of the multi-vehicle Covering Tour Problem (m-CTP) (Hachicha et al. 2000). The resulting combinatorial optimization problem is NP-hard, which justifies the use of heuristics developed in this work to obtain suboptimal solutions of acceptable quality in reasonable time.

The heuristics were implemented and compared using several synthetic instances in MATLAB®. Randomly generated data and real data from the city of Vinhedo, São Paulo, Brazil, were obtained and used to validate the code.

2 m-CTP for patrol route planning

In the problem considered in the previous section, a set of geographical points that need to be visited during a routine patrol was determined by the police force, which might include schools, hospitals, public buildings, etc. Additionally, there is another set that must be at convenient distance from the route, for instance, public parks, community centers, bank agencies, etc. The design of routes is of crucial importance, due to the limited resources (the size of the fleet) available to cover usually large geographical areas.

The situation here is slightly different from the model described by Hachicha et al. (2000). There, the length and number of sites per tour are limited, as well as the number of routes. In our case, the number of routes is fixed (we do not want cars and personnel idling at the base). This implies some adaptations to their model, and more importantly, to the heuristics to obtain approximate solutions.

The model involves a graph $G = (V \cup W, E)$, whose nodes correspond to strategic geographical points, e.g., intersections, certain locations, etc., that are either important on its own or serve to establish reference locations for the routes. The node set is partitioned into two subsets: $V = \{0, \ldots, n\}$ is the set of nodes that may belong to routes and $W = \{n + 1, \ldots, n + \ell\}$ is the set of nodes that must be covered, but not visited. The set $V \setminus \{0\}$ is denoted by $V^*$. The set $V$ contains a subset $T$ of nodes that must be visited. Node 0 ($\in T$) corresponds to the base. The symbol $T^*$ denotes the set $T \setminus \{0\}$. The set $E$ contains all possible undirected arcs between nodes of $V$. The entry $c_{ij}$, of the distance matrix $C = (c_{ij})$, contains the distance between nodes $i, j = 0, 1, \ldots, n + \ell$. Ideally these distances (or travel times) would be calculated separately from real data, taking into account possible paths between nodes, average speeds, etc. This would imply nonsymmetrical distances. The undirectedness assumption is a simplification of the problem. Furthermore, in the real instance considered, involving the city of Vinhedo, the true path lengths (corresponding to routes using the city streets) were not known, so Euclidean distances were used instead as an approximation. This decision was made after a proper survey of the features of the problem conducted with Vinhedo patrol professionals.

This assumption implies that $C$ is symmetric with zero diagonal. In this case, we choose to
represent the set (undirected arc) \{i, j\} by the ordered pair \((i^*, j^*)\), where \(i^* = \min\{i, j\}\) and \(j^* = \max\{i, j\}\). The size of the fleet is denoted by \(m\) and the admissible distance from a node in the route to a node that must be covered is denoted by \(c\). We need one last parameter, to express our tolerance regarding the lack of balance between different routes. We compare routes by means of the number of nodes each route contains. The number \(r\) denote the maximum difference allowed for the total visited nodes in any two routes.

We want to construct \(m\) routes satisfying the following conditions:

- Each route is a closed path in \(G\) containing node 0.
- Each node in \(V^*\) must belong to at most one route.
- Each node in \(T^*\) must belong to exactly one route.
- There must be at least a visited node at a distance of at most \(c\) from each node in \(W\).
- The number of nodes in any two routes differ by at most \(r\).

The graph in Figure 1 illustrates an instance of the problem. Nodes in \(T\) are represented by stars, circles denote the nodes in \(W\) and bullets represent the nodes in \(V \setminus T\). The base node is the one indicated by an arrow. There are 11 nodes in \(T\), 24 nodes in \(W\) and 12 in \(V \setminus T\). The arcs were not drawn, to avoid obscuring the nodes, but in Figure 3 one may see the arcs used in a near-optimal solution to this instance.

Finally, we need a criterion to choose between feasible sets of routes. We have chosen the set that has the smallest overall length. The reason is that smaller routes could possibly be traveled many times during a work shift, affording a better surveillance of the area.

Variants of this model have been used in many contexts. Labbé & Laporte (1986) discuss how to simultaneously decide on the location of mailboxes and the planning of routes involving candidate sites, such that all users are close enough to some mailbox. The planning of routes for medical mobile
units (Brown & Fintor 1995, Foord 1995, Hodgson et al. 1998, Oppong & Hodgson 1994, Swaddiwudhipong et al. 1995) bears many similarities with the planning of urban patrol routes. In places where medical services for small villages are rendered by mobile units, their routes must be planned in such a way that, in addition to visiting specific locations, the villages not included in the tour must be within walking distance from some other village included in the tour. The model can also fit situations in business sectors, see Simms (1989) for an application in dairy practice. The special case of $m$-CTP where $V = T$ reduces to a vehicle routing problem (Laporte 1992) with unit demand. Baldacci et al. (2005) present three Scatter Search heuristic algorithms for the 1-CTP. Motta et al. (2001) use a GRASP approach to solve a variant of the 1-CTP, the Generalized Covering Tour Problem, whose minimum length tour may pass through a subset of $W$. In another point of view, Jozefowiez et al. (2004, 2007) present a multi-objective covering tour problem, a generalization of the 1-CTP where the parameter $c$ is omitted and replaced by an objective. They propose a hybrid strategy approach that combines a multi-objective evolutionary algorithm with a branch-and-cut algorithm to determine the Optimal Pareto sets.

3 Mathematical formulation of the $m$-CTP

The mathematical model below closely resembles to the model in Hachicha et al. (2000). The two models share the objective function and the first four sets of constraints. After that, the different assumptions made here imply different sets of constraints.

In order to facilitate the modeling of the covering restrictions, we define the set of nodes in $V$ within the allowed prescribed distance $c$ from each node $j \in W$: $S_j = \{i \in V \mid c_{ij} \leq c\}$. We may suppose without loss of generality that $|S_j| \geq 2$, since if there is only one node $i \in V$ close enough to some $j$, then we may as well include $i \in T$ and eliminate $j$ from $W$. Similarly, we may assume that $S_j \cap T = \emptyset$, for all $j$, since we do not need to worry about covering nodes that close enough to some node in $T$. With these assumptions, $S_j = \{i \in V \setminus T \mid c_{ij} \leq c\}$ and its cardinality is at least two, for all $j \in W$.

The model contains two sets of binary variables. The variable $y_{ijk}$ is 1 if node $i$ is visited by vehicle $k$, and 0 otherwise, for $i \in V$, $k = 1, \ldots, m$. The variable $x_{ijk}$ is 1 if vehicle $k$ uses arc $(i, j)$ in its route, and 0 otherwise.

Using these variables, the “cost” of a solution, given in (1), is the cumulative length of all routes, which we wish to minimize. The constraints may be modeled as follows. The covering of node $j \in W$ is guaranteed by (2). Constraint (3) makes sure that each node in $V \setminus T$ belongs to at most one route. (4) implies that, if node $h$ belongs to route $k$, then it has two neighbors in the route. Constraint (5) avoids subtours, by forcing that, if node $h \in S \subseteq V^*$ belongs to route $k$, then the cut-set $(S, V \setminus S)$ must contain at least two arcs of route $k$. (6) guarantees that each route has two arcs incident to the base. The maximum difference between the number of nodes of different routes is enforced by (7). The fact that every node in $T$ must be visited by some tour is expressed in (8), whereas (9) forces that the base belongs to every route. The last set of constraints, (10), simply specifies the allowed values for the variables.

$$\begin{align*}
\text{Min} & \quad \sum_{k=1}^{m} \sum_{i=0}^{n-1} \sum_{j=i+1}^{n} c_{ij} x_{ijk}, \\
\text{subject to:} & \quad 2 & \geq |S_j|, \\
& \quad y_{ijk} \leq x_{ijk}, & \quad \forall i \in V, \quad \forall k \in \{1, \ldots, m\}, \\
& \quad \sum_{i \in V} y_{ijk} = 1, & \quad \forall k \in \{1, \ldots, m\}, \\
& \quad \sum_{i \in V} x_{ijk} = 2, & \quad \forall k \in \{1, \ldots, m\}, \\
& \quad \sum_{i \in V} c_{ij} x_{ijk} \leq c_{ij}, & \quad \forall i, j, \quad \forall k \in \{1, \ldots, m\}, \\
& \quad y_{ijk} \leq \sum_{l \in S} x_{ljk}, & \quad \forall i, j, \quad \forall k \in \{1, \ldots, m\}, \\
& \quad y_{ijk} \leq \sum_{l \in V \setminus S} x_{ljk}, & \quad \forall i, j, \quad \forall k \in \{1, \ldots, m\}, \\
& \quad y_{ijk} \leq 1, & \quad \forall i, j, \quad \forall k \in \{1, \ldots, m\}, \\
& \quad y_{ijk} \geq 0, & \quad \forall i, j, \quad \forall k \in \{1, \ldots, m\}.
\end{align*}$$

(1)
4 Heuristics

The heuristics developed for the urban patrolling problem modeled in the last section are divided into three phases. In Phase 1, subsets $V_k$ and $W_k$, $k = 1, \ldots, m$, are selected, where $V_k$ and $W_k$ are the nodes that may be visited and that should be covered by route $k$, respectively. Phase 2 deals with $m$ 1-CTP problems, defined on the subgraph induced by $V_k \cup W_k$. At this point, $m$ closely related problems are considered separately. The last phase tries to improve the solution by taking this interrelation into account. The three-phase sequence is repeated according to a criterion specific to the routine employed in Phase 1. The best solution in the whole loop of three-phase (or outer iterations) is selected.

Table 1 below summarizes the routines employed in each phase of the various heuristics. The routine employed in Phase 1 will lend its name to the heuristic. Note that they share Phase 2 in common. The second and third heuristics in Phase 1 were proposed in Hachicha et al. (2000). The routine in Phase 2 and the Balanced 2-opt routine in Phase 3 are variants of the H-1-CTP routine and the 2-opt* routine from Hachicha et al. (2000), respectively. The remaining routines were developed by us. Note that the first three heuristics use Balanced 2-opt in Phase 3, while the Sector Partition uses Multicover Elimination.

The Sweep Routine corresponds to steps 1 and 2 of the sweeping algorithm of Hachicha et al. (2000). Similarly, Route-first/Cluster-second is formed by steps 1 and 2 of the algorithm of same name in Hachicha et al. (2000). In these routines, some modifications were made to achieve the balanced routes.
The **Greedy Selection** routine gradually selects sites using a criterion that selects the nearest site to the one previously selected, forming a circular ordered list. The nodes in this list are then partitioned into \( m \) subsets of approximately equal size, keeping the order of the original list and starting with the first node in the list. In subsequent iterations, the selection step is not repeated. Instead, we simply shift by one the order of the nodes in the list and redo the partitioning. This is repeated approximately \( t/m \) times, where \( t \) is the cardinality of the list.

In the **Sector Partition** routine, we use a geographical approach to divide \( V, T, \) and \( W \), into \( m \) subsets, corresponding to circular sectors. Each outer iteration corresponds to a counterclockwise shift of the sectors. This simple geographical division is used to reduce the computational time in Phase 1.

The **Modified 1-CTP Routine** is a modification of the heuristic described in Gendreau et al. (1997) for the covering tour problem.

The **Balanced 2-opt** routine contemplates improvements via arc swapping, and is adapted from the 2-opt* heuristic of Hachicha et al. (2000).

The **Multicover Elimination** checks whether some node in \( W \) is overcovered, i.e., it is covered by more than one node included in a route. If this is the case, there may be room for improvement, by removing one of the superfluous nodes.

In the described routines, it is often necessary to know the subset of nodes in \( W \) covered by a particular node in \( V \). The subset covered by node \( i \) is denoted by \( C_i = \{ j \in W \mid i \in S_j \} \). Recall that \( S_j = \{ \ell \in V \setminus T \mid c_{\ell j} \leq c \} \).

### 4.1 Greedy Selection

Initially we form a single route \( R = (h_0, h_1, \ldots, h_k) \) that contains all nodes in \( T \) and covers all nodes in \( W \) as follows. The routine starts with \( h_0 = 0 \), \( R = (h_0) \) and \( L = T^* \cup W \). The set \( L \) is gradually emptied using the criterion that selects the nearest site to the one previously selected in \( L \). If the chosen node \( h \) belongs to \( T \), it is simply appended to \( R \), and \( \{ h \} \cup C_h \) is removed from \( L \). If \( h \) belongs to \( W \), then one selects from \( S_h \) the node that covers the greatest number of yet uncovered nodes, say node \( \ell \), and appends it to \( R \). Then \( C_\ell \) is removed from \( L \).
Here, and in the next two routines, once \( R \) is constructed we consider the sequence \( R^* = R \setminus \{0\} \) as a circular list. In order to apply Phase 2, we need to divide \( V, T, \) and \( W \) into \( m \) subsets that are induced by a partition of \( R^* \) as follows. Let \( p = \lfloor z/m \rfloor, q = z - mp \). Starting from the first node in \( R^* \), we select \( q \) subsets of sequential nodes of size \( p + 1 \), and \( m - q \) subsets of size \( p \). The nodes in \( V \) (resp., \( T \)) in the \( k \)th subset union \( \{0\} \) are denoted by \( V_k \) (resp., \( T_k \)). The \( k \)th subset of \( W \) is \( W_k = \bigcup \{C_i \mid i \in V_k\} \).

In the next round, the partition of \( R^* \) will start at the second node, and then third, and so on, in a total of \( p \) or \( p + 1 \) outer iterations, depending on whether \( z \) is a multiple of \( m \) or not.

### 4.2 Sweep Routine

The difference between this routine and the previous one is the strategy that we build the single route \( R = (h_0, h_1, \ldots, h_z) \). The sweeping process is applied to the vertices \( T \cup W \) and several solutions are generated. The routine starts with \( h_0 = 0, R = (h_0) \) and \( L = T^* \cup W \). Choose an arbitrary node \( \tilde{h} \in L \) and consider a half line from \( h_0 \) passing through \( \tilde{h} \). The set \( L \) is gradually emptied using the criterion that sweeps the nodes \( h \in L \) according to the ascending order of the angles \( \theta_h = \angle \tilde{h}h_0h \). If the chosen node \( h \) belongs to \( T \), it is simply appended to \( R \), and \( \{h\} \cup C_h \) is removed from \( L \). If \( h \) belongs to \( W \), then one selects from \( S_h \) the node that covers the greatest number of yet uncovered nodes, say node \( \ell \), and appends it to \( R \). Then \( C_\ell \) is removed from \( L \). Once \( R \) is constructed, the process continues as in the Greedy Selection routine.

### 4.3 Route-first/Cluster-second

Again, the difference between this routine and Greedy Selection routine is the strategy that we build the single route \( R = (h_0, h_1, \ldots, h_z) \). Here, a feasible 1-CTP solution for \( V, T, \) and \( W \) is determined by means of the Modified 1-CTP Routine, say route \( R = (h_0, h_1, \ldots, h_z), h_0 = 0 \), that contains all nodes in \( T \) and covers all nodes in \( W \). This tour is then divided into smaller feasible routes as in the Greedy Selection routine.

### 4.4 Sector Partition

This routine is applicable only in cases where there is a geographical model of the problem. The site associated with the base node is taken as the geographical center of a circular disk containing all locations (nodes) under consideration. This disk is partitioned into \( m \) circular sectors of same central angle. Nodes corresponding to sites in the \( k \)th partition form the sets \( V_k, T_k, \) and \( W_k \). The first partition is arbitrary. In the next iteration, the sectors are shifted counterclockwise by \( 360^\circ/t \), and this is repeated until we return to the original partition, a total of \( t \) outer iterations. In the computer experiments, the value \( t = 10 \) was used. Despite of its simplicity, if the distribution of nodes is non-uniform, this way of choosing the partitions \( V_k, T_k, \) and \( W_k \) does not guarantee the equilibrium among the number of nodes in each partition, possibly producing unbalanced routes.
4.5 Modified 1-CTP Routine

Gendreau et al. (1997) developed a heuristic for the covering tour problem, or 1-CTP, using elements of the GENIUS heuristic for the traveling salesman problem (TSP) of Gendreau et al. (1992) and PRIMAL1 set covering heuristic of Balas & Ho (1980).

This heuristic uses the fact that, if the set of nodes that should be visited (that is, the support of the optimal $y$) is known, then the covering tour problem reduces to a traveling salesman problem. This suggests the construction of a covering problem by considering the “covering” aspect separately, namely the combinatorial optimization problem (11) below. Notice that, since $m = 1$, there is no need for the tour index $k$.

\[
\begin{align*}
\min & \quad \sum_{j \in V} c_j y_j, \\
\text{s.t.} & \quad \sum_{j \in S_i} y_j \geq 1, \quad i \in W, \\
& \quad y_j = 1, \quad j \in T, \\
& \quad y_j \in \{0,1\}, \quad j \in V.
\end{align*}
\]  

(11)

In PRIMAL1, variables $y_j$ are gradually included in the solution according to a greedy rule, using one of three merit functions: (i) $f(c_j, b_j) = c_j / (\log_2 b_j)$; (ii) $f(c_j, b_j) = c_j / b_j$; (iii) $f(c_j, b_j) = c_j$; where $b_j$ is the number of uncovered (in the current partial solution) nodes in $W$ that are covered by $j$.

On the other hand, TSP routes are also constructed incrementally in GENIUS. During the construction phase, a tentative tour is built starting with three arbitrarily selected nodes and using a general insertion procedure (GENI), with rules for selecting candidates and rules for evaluating the inclusion in the tour. Once a complete tour is obtained, one seeks to improve it by removing and reinserting each node of the tour, in a postoptimization procedure called US (for Unstringing and Stringing).

These heuristics are combined to produce a covering tour heuristic as follows. Nodes are gradually selected and added to the set of nodes that should be visited, and a new approximate solution to the TSP involving each insertion of these nodes is obtained (using GENIUS), this procedure stops when all nodes in $W$ have been covered. The selection of the each node is made by considering the merit functions in PRIMAL1 heuristic using the coefficient $c_j$ (cost to insert this node in the approximate solution to the TSP, calculated by GENI selection rules). At the end, postoptimization is applied by removing superfluous nodes. The whole process is done for each merit function, and two sequences of the merit functions are considered.

Numerical experiments with our code for the various heuristics showed that the GENI part was fast and produced acceptable results, whereas the postoptimization US routine was quite costly. After several trials, we arrived at a modified routine of the covering tour heuristic, in order to avoid the repeated construction of unnecessary approximate solutions of the TSP at the insertion of each node. This is done by inserting the new node using GENI rules and removing superfluous nodes using US rules without discarding the previous TSP solution.

More details about the experiments with the different heuristics tried out, which showed a better computational performance with these modifications, can be found in Oliveira (2008). In the modified routine listed below, the merit functions are used in the sequence (i)-(ii)-(iii).
STEP 1. Let $H \leftarrow T$, $\bar{z} \leftarrow \infty$. Let $f$ be the merit function specified in (i). Let $z$ be the value of the tour obtained applying GENIUS to the TSP involving the nodes in $H$. If all nodes in $W$ are covered, go to Step 2. Otherwise, go to Step 3.

STEP 2. If $z \leq \bar{z}$, let $\bar{z} \leftarrow z$, $\hat{H} \leftarrow H$. If the definition used for the merit function is the last, stop with the best solution so far, with value $\bar{z}$ and set of visited nodes $\hat{H}$. Otherwise, remove from $H \setminus T$ nodes associated with multiple covered nodes in $W$ using US. Go to Step 3 with the next definition of the merit function $f$.

STEP 3. Select within $V \setminus H$ node $h^*$ using PRIMAL1, with cost coefficients $c_j$ calculated using criteria defined in GENI. Insert $h^*$ in the route employing GENI, and let $H \leftarrow H \cup \{h^*\}$. Repeat the process until all nodes in $W$ are covered. Let $z$ be the value of the tour obtained. Go to Step 2.

When this routine is used in Phase 2, it is run $m$ times with the sets $V_k$, $T_k$, and $W_k$, for $k = 1, \ldots, m$, constructed in Phase 1.

4.6 Balanced 2-opt

This is a modification of the 2-opt* heuristic described in Hachicha et al. (2000). The changes introduced aim to keep the balance, in terms of number of nodes, among the routes. There, the number of routes is fixed, since we do not want cars and personnel idling at the base. Steps 1 and 2 are adapted from Hachicha et al. (2000), since care must be taken not to destroy the balance between routes. Step 3 is new. It considers swapping of nodes belonging to different routes. Notice that the exchanges considered in this last step do not alter the number of nodes in each route.

At this point, the initial set of $m$ routes constructed in Phase 1 has already undergone improvement in Phase 2, and the following procedure is executed.

STEP 1. Let $\rho$ be the smallest number of nodes per route, excluding the base node. Transform the set of $m$ routes into a single route by making $m$ copies of the base node, as described in Lenstra & Rinnooy Kan (1975). Make a list of all possible pairs of distinct arcs in this single route.

STEP 2. In this step, the arc-pairs in the list are considered sequentially, until either a better solution is reached and return to Step 1 or there is no pair left, in which case go on to Step 3. Let $\{\{r,s\},\{t,u\}\}$ be the current pair. Consider the two possibilities of replacement: (i) $\{\{r,t\},\{s,u\}\}$, (ii) $\{\{r,u\},\{s,t\}\}$. The replacement may split the single route into two routes. If this happens, the new routes correspond to a feasible solution only if each of them contains at least one copy of the base node. Furthermore, the number of nodes in the route(s) between each pair of copies of the base node must be at least $\rho$ (in case one of the subroutes created by the replacement contains only one copy of the base node, the total number of nodes in this subroute must be at least $\rho + 1$). For each feasible alternative solution, calculate its objective value. If there is an improvement, recover $m$ routes from the best improved single (or pair of) route(s) and go back to Step 1. If the end of the list is reached with no (feasible) improvement, recover $m$ routes from the best improved single (or pair of) route(s) and go back to Step 3.


**STEP 3.** Make a list of pairs of nodes, each distinct from the base node, and belonging to different routes. Consider the pairs sequentially. For each pair, calculate the value of the alternative solution obtained by swapping the nodes. Keep the solution with best objective function value.

### 4.7 Multicover Elimination

If a node in $W$ is covered by more than one node, we may improve the solution by eliminating one of the superfluous nodes. At this point, the initial set of $m$ routes constructed in Sector Partition has already undergone improvement in Phase 2. Note that only nodes in $V \setminus T$ (the support of $y$ that are not in $T$) may be removed from the solution. If a node $i$ may be removed, we consider the alternate solution obtained by deleting the arcs incident to $i$ and adding the arc linking its neighbors. Observe that, if the distances between nodes satisfy the triangle inequality, the removal of a node always implies in the decrease of the objective function.

The possibility of removal of node $i$ is checked in the brute force way: we delete it from the route and check whether this results in some node in $W$ being uncovered. This is a two-step procedure. In the first step we build a list of candidates for removal, and in the second step we examine the list sequentially, trying to remove as many nodes as possible.

**STEP 1.** Examine all nodes in the route that belong to $V \setminus T$ and build a list of superfluous nodes in descending order with respect to the amount of decrease in the objective function implied by their removal.

**STEP 2.** Consider the nodes in the list sequentially and remove the node under consideration if its removal does not destroy the required coverage.

Note that this procedure is simpler than Balanced 2-opt routine and it was chosen as the postoptimization for Sector Partition heuristic in order to reduce the computational time in Phase 3.

### 5 Numerical experiments

The heuristics were tested on a set of randomly generated instances and on an instance constructed from official data of the city of Vinhedo, São Paulo. They were implemented in MATLAB® and tests were run on an Intel Pentium 3.40 GHz, with 2GB of RAM and Linux operating system. The characteristics of the randomly generated instances are described next, followed by comments about the collection of data from the city of Vinhedo.

We close this section with statistics summarizing the test results and comments on the performances of the different heuristics.

#### 5.1 Randomly Generated Instances

The full characterization of an instance involves the following parameters: the cardinalities of the sets $V$, $T$, and $W$; the distance $c$ calculated from the sets $S_j$, $j \in W$; the number $m$; and the parameter $r$ (the maximum difference between the number of nodes in different routes).
In order to observe the behavior of the heuristics with respect to scale, we have designed five classes of problems, regarding the total of nodes in $V \cup W$: 100, 150, 200, 300, and 400. The first, third and fifth classes have equal number of nodes in both sets. The second class has $|V| = 50$, $|W| = 100$, and the fourth has $|V| = 100$, $|W| = 200$. Each class is subdivided in three subclasses by changing the cardinality of $T$. The subclass 1 has $|T| = |V|/8$, the subclass 2 has $|T| = |V|/4$ and the last one has $|T| = |V|/2$. We generated 20 instances per subclass, totaling 300 instances overall.

Each instance is initially characterized by a collection of pairs $\{(x_i, y_i) \mid i = 1, \ldots, |V| + |W|\}$, where both $x_i$ and $y_i$ are random variables uniformly distributed in the interval $[0, 100]$. These pairs are interpreted as coordinates of points in a $100 \times 100$ plane region. The base coordinates are randomly picked from the rectangle $[35, 65] \times [35, 65]$. The sequence of pairs is renumbered, so that the basis corresponds to the first pair, but the relative order of the remaining pairs is not changed. Pairs corresponding to the remaining nodes in $T^*$ are chosen sequentially in this list, starting from the second pair, and are followed by the pairs corresponding to the nodes in $V \setminus T$. All other pairs correspond to nodes in $W$.

Next we choose the constant $c$ used in the definition of the neighborhood $S_j$ of node $j \in W$. That is, the nodes in $V$ which are close enough to $j$ in the sense that $j$ will be considered covered if the route contains a node in $S_j$. The following considerations guide our choice. We want $|S_j| \geq 2$, for all $j$, so $c$ must be greater than the maximum of the distances $c_{jh(j)}$, for all $j \in W$, where $h(j)$ is the index of the node in $V$ that is the second closest node to $j$. Furthermore, we want that every node in $V \setminus T$ cover some node in $W$, otherwise this node could be eliminated from consideration in a pre-processing run. Thus, we also want $c$ to be greater than $m_h$, the distance from node $h$ to $W$, for all $h \in V \setminus T$. The value of $c$ is thus the greatest of the two lower bounds.

The number of subroutes $m$ was fixed at three for all instances. Classes 100, 150 and 200 have $r = 2$, class 300 has $r = 3$ and the last class has $r = 4$.

### 5.2 City of Vinhedo Instance

This city is located approximately 80 km east of the city of São Paulo, Brazil. Its economy is mostly based on agriculture, in particular the growing of grape and production of related goods, specially wine. It occupies an area measuring roughly 81,742 km², and, as of 2009, had about 60,000 inhabitants. It is part of a group of cities known as the fruit circuit, which promotes three to four festivals per year, the fig festival, the strawberry festival, and so on. The per capita income is high and the criminality rate is low.

Municipal Guard of the city of Vinhedo provided two lists of addresses. The first list, with 101 addresses, corresponded to sites that needed to be visited during rounds, and the second, with 133 addresses, contained the addresses of sites that should be covered. The geographical coordinates of these sites were calculated, as well as the coordinates of all street intersections. The geographical coordinates of the Municipal Guard base gave us the base node location. This real instance thus had 102 nodes in $T$, 133 nodes in $W$ and 2496 nodes in $V \setminus T$ (the intersections). Figure 2 shows the distribution of points within the city region.

Empirical experience in security acquired by the Municipal Guard leads to the conclusion that a patrolling officer can watch each site up to a distance of 150 m, so we chose $c = 150$ m. This allowed the elimination of 1563 nodes in $V \setminus T$, whose distance to $W$ was greater than $c$. We verify this by comparing the Figure 2 and Figure 6. The latter illustrates a set of routes obtained in the tests. Note
that the distribution of nodes in this instance is not uniform. Our real-data instance had $|T| = 102$, $|V \setminus T| = 933$ and $|W| = 133$. The values considered for $m$ were five, six and seven, which are related to the likely number of cars available for patrol duties. The parameter $r$ was chosen as six in the first case ($m = 5$) and as eight in the remaining cases. According to the Municipal Guard this choice does not overload the patrolling officers.

### 5.3 Summary of Results for Random Instances

Figure 3 shows a typical output of the heuristics, in this case for the instance shown in Figure 1. Notice that the three routes contain all nodes in $T$ (the star shaped nodes), but some nodes in $V \setminus T$ (bullets) were not picked. The grey disks are the neighborhoods of the nodes in $W$ (circles). One can verify that the solution is feasible with respect to coverage, as well as each grey disk contains at least one visited node. Furthermore, the routes are balanced, differing by at most one node.

Tables 2 and 3 summarize the relevant data collected in the numerical experiments with the four heuristics listed in Table 1. Space constraints imposed the splitting of the results in two tables, but they follow the same format. The first column indicates the subclass and the next two groups of columns refer to two different heuristics. For each heuristic we report the time (in seconds) spent on average over the 20 instances of each subclass, the average cost (total length of all tours) and the quality index (Q.I.) of the heuristic. This last number is the average cost divided by the minimum average cost over all four heuristics, showing how good the performance of that heuristic was, as compared to the one with the best average cost. Therefore, the Q.I. of the routine with the best cost is 1, and the table cell containing this entry is shaded for emphasis.

The Route-first/Cluster-second heuristic has the biggest number of the Q.I. equals to 1, but the advantage is not striking. The worst Q.I. is 1.112 (11.2% worse than the best result), achieved by
Figure 3: Routes obtained for instance shown in Figure 1

| Subclass | Greedy Selection | Sector Partition |
|----------|------------------|------------------|
|          | Q.I.  | Cost  | Time (sec) | Q.I.  | Cost  | Time (sec) |
| 100-1    | 1.043 | 396.8 | 7.4        | 1.001 | 380.9 | 8.3        |
| 100-2    | 1.070 | 477.4 | 8.2        | 1.051 | 469.0 | 15.4       |
| 100-3    | 1.059 | 555.3 | 59.6       | 1.012 | 530.8 | 147.8      |
| 150-1    | 1.048 | 434.3 | 17.0       | 1.007 | 417.3 | 13.1       |
| 150-2    | 1.059 | 491.7 | 15.1       | 1.017 | 472.2 | 22.5       |
| 150-3    | 1.073 | 563.5 | 63.4       | 1.029 | 540.3 | 138.9      |
| 200-1    | 1.080 | 542.2 | 135.8      | 1.000 | 502.0 | 65.2       |
| 200-2    | 1.074 | 583.6 | 306.5      | 1.019 | 553.7 | 307.3      |
| 200-3    | 1.063 | 721.3 | 2505       | 1.005 | 681.7 | 2011       |
| 300-1    | 1.112 | 576.5 | 539.3      | 1  | 518.3 | 140.2      |
| 300-2    | 1.084 | 589.7 | 519.9      | 1.031 | 560.5 | 259.5      |
| 300-3    | 1.095 | 727.3 | 4377       | 1.027 | 682.2 | 1953       |
| 400-1    | 1.100 | 679.3 | 7785       | 1  | 617.3 | 1384       |
| 400-2    | 1.112 | 775.3 | 9376       | 1.009 | 703.6 | 2345       |
| 400-3    | 1.106 | 942.8 | 13176      | 1.021 | 869.9 | 6654       |

We observed discrepancies in the computational effort (measured by the time spent) of the heuristics.
Table 3: Average results per subclass for Sweep Routine and Route-first/Cluster-second

| Subclass | Sweep Routine | Route 1st/Cluster 2nd |
|----------|---------------|-----------------------|
|          | Q.I. | Cost (km) | Time (sec) | Q.I. | Cost (km) | Time (sec) |
| 100-1    | 1.039 | 395.3     | 5.3        | 1  | 380.1 | 6.6 |
| 100-2    | 1.035 | 461.9     | 7.2        | 1  | 446.0 | 13.7 |
| 100-3    | 1.009 | 528.9     | 59.2       | 1  | 524.1 | 108.7 |
| 150-1    | 1.023 | 424.0     | 12.4       | 1  | 414.4 | 11.5 |
| 150-2    | 1.016 | 471.7     | 13.3       | 1  | 464.1 | 20.1 |
| 150-3    | 1.015 | 533.0     | 60.1       | 1  | 525.0 | 105.2 |
| 200-1    | 1.027 | 515.4     | 108.4      | 1  | 501.7 | 97.2 |
| 200-2    | 1.009 | 548.5     | 257.6      | 1  | 543.1 | 328.4 |
| 200-3    | 1.000 | 678.5     | 2887       | 1  | 678.3 | 3041 |
| 300-1    | 1.015 | 526.2     | 319.7      | 1.003 | 520.1 | 259.4 |
| 300-2    | 1.025 | 557.6     | 492.5      | 1  | 543.5 | 395.5 |
| 300-3    | 1.004 | 666.6     | 3739       | 1  | 663.7 | 4377 |
| 400-1    | 1.015 | 626.7     | 4760       | 1.004 | 620.2 | 3689 |
| 400-2    | 1.008 | 702.5     | 6204       | 1  | 696.8 | 6630 |
| 400-3    | 1     | 851.7     | 12362      | 1.009 | 859.7 | 13796 |

tics, as illustrated by the charts in Figures 4 and 5. The average time of an heuristic in a given subclass may double when compared with another heuristic in the same subclass. The disparity of ranges led to the construction of two distinct charts. Figure 4 contains data relative to subclasses 100-1 to 150-3, the data of the remaining subclasses are in Figure 5.

In the first set of subclasses, the Sweep heuristics is the fastest, except for the subclass 150-1, where Route-first/Cluster-second does slightly better. However, as the size increases, it is replaced by the Sector Partition heuristics, whose times are smaller than the others, with the exception of class 200-2, where its time is only 19% bigger than the minimum time.

If we are looking for a balance between speed and solution quality, and the problem is large scale, these random test examples point to the Sector Partition heuristic as the best choice. This shows that the Sector Partition routine is well-adapted to random instances whose nodes are uniformly distributed. On the other hand, we may safely say that the Greedy Selection heuristic presented the worst results, with the highest Q.I. in every subclass and times increasing with the instance size.

5.4 Vinhedo Instance: Comparison of Heuristics

Table 4 reports the cost (in kilometers), quality index (Q.I.), time (in seconds), the value chosen for the parameter $r$, the maximum difference between the cardinalities of the different routes obtained, the size of the fleet, and the heuristic considered. The numerical experiments data are collected in three cases: fleet of size five, six or seven.
Table 4: Results from the Vinhedo instance

| Cost (Km) | Q.I.  | Time (sec) | r | Maximum difference | Vehicles | Heuristic                      |
|-----------|-------|------------|---|---------------------|----------|--------------------------------|
| 106.271   | 1.12099 | 79745      | 6 | 3                   | 5        | Greedy Selection               |
| 94.801    | 1     | 77831      | 6 | 2                   | 5        | Sweep Routine                  |
| 99.486    | 1.04942 | 68459      | 6 | 3                   | 5        | Route 1st/Cluster 2nd          |
| 114.022   | 1.08799 | 64813      | 8 | 4                   | 6        | Greedy Selection               |
| 104.800   | 1     | 63351      | 8 | 3                   | 6        | Sweep Routine                  |
| 104.806   | 1.00006 | 51102      | 8 | 1                   | 6        | Route 1st/Cluster 2nd          |
| 121.494   | 1.10100 | 52112      | 8 | 4                   | 7        | Greedy Selection               |
| 110.349   | 1     | 51985      | 8 | 4                   | 7        | Sweep Routine                  |
| 111.234   | 1.00802 | 41490      | 8 | 2                   | 7        | Route 1st/Cluster 2nd          |

The Sector Partition heuristic is omitted from Table 4 because it did not obtain balanced feasible solutions, due to the extremely non-central location of the base of operations. Recall that in the randomly generated instances, the base node was chosen inside a square of side length 30, whose center coincided with the center of the region, a square of side length 100. Thus, in this case, the partition produced by the heuristic obtained a set of balanced routes.

Unfortunately, this did not happen in the Vinhedo instance. The geographical sites considered fit
in a rectangular area of aspect ratio 1.5, that is, the width to height ratio (see Figure 6, which shows the routes built by the Sweep Routine heuristic in the 5-vehicle case). If we were to consider the “base region” adopted for the synthetic instances, the base of operations should be located in a rectangle of sides 30% of the bigger rectangle, with same center. In the Vinhedo instance the base lies below and outside this ideal rectangle, missing the region by 6% of the vertical dimension.

Furthermore, the geographical distribution of sites that need to be visited or covered is highly non-uniform. If we divide the shaded rectangle of Figure 6 in four rectangles by drawing vertical and horizontal lines intersecting at the base, we will find 75.3% of all nodes concentrated in the northwest rectangle, 15.2% in the northeast, 6.8% in the southwest and only 2.7% in the southeast rectangle. As a result, the Sector Partition heuristic produced quite unbalanced routes, with the maximum difference between two routes exceeding the value of the parameter $r$.

Finally, the speed exhibited by the Sector Partition heuristic suggests that it is worthwhile to further investigate its possibilities. One direction for future research is that of a non-uniform division in sectors, choosing the central angles in such a way that the number of nodes in each sector is approximately equal.

The bad performance of the Greedy Selection heuristic was confirmed in the (very large scale) Vinhedo instance, with the highest Q.I. in every test considered.

In discordance with the previous tests, the Sweep Routine scored better objective function values, although not by much, than the Route-first/Cluster-second heuristic, in all cases considered, as indicated by the shaded cells in Table 4. However, Sweep Routine exhibited 13% to 25% larger times than Route-first/Cluster-second.

Figure 5: Average times for subclasses 200-1 to 400-3
If we divide the cost by the number of cars, we get an estimate of the size of a typical route. For example, it turns out that the routes were less than 20 km long in the case of five vehicles. This length is quite acceptable, as a patrol car may cover this distance several times during a tour of duty. This contemplates very well the objective of providing a good coverage with the available resources. Finally, the size of the problem indicates how poor trial and error solutions usually prove to be, making an automated method such as this more desirable.

6 Conclusions and future work

We developed, implemented and tested various heuristics for the construction of routes in the context of urban patrolling. They were compared among themselves in numerical tests involving a set of randomly generated problems, and variants of a problem defined with real data from the city of Vinhedo, Brazil.

We developed a modified Sweep Routine, which did very well for the randomly generated instances with sizes in the lower end of the spectrum, up to 150 nodes overall. For larger instances, the Sector Partition heuristic, developed by us, was usually the fastest (see the charts in Figures 4 and 5). The quality of the solutions produced by all heuristics was quite uniform in the batch of synthetic problems.

The heuristics were tested using the Vinhedo instance. The Greedy Selection heuristic, when compared with the other heuristics, produced solutions with lower quality. The solutions of our modifications of the Sweep Routine and Route-first/Cluster-second Routine had similar quality, though the former was always better. On the other hand, the latter was consistently faster.

The solutions obtained in the Vinhedo instance show the advantage of using this kind of modeling and heuristics, since the scale of the problem makes it unlikely arriving at such a scheme via a trial
and error approach.

We observed that the Sector Partition heuristic needs improvement in order to be applied in real instances having highly non-uniform distribution of nodes, turning it into a competitive and robust routine. This will be the subject of future research.

Acknowledgements

The first author was partially support by CAPES and FAEPEX (grant 534/09). The second author was partially supported by CNPq (grant 307907/2007-4) and FAPESP (grant 2009/05470-0). The third author was partially supported by CNPq-PRONEX Optimization and FAPESP (grant 2006/53768-0). The authors thank Osmir Cruz, Commander of the Municipal Guard of Vinhedo, who graciously supplied the list of addresses that ought to be visited or covered during rounds. They further thank the municipal government of Vinhedo and the company SSR-Tecnologia for translating the list of addresses into a list of geographical coordinates.

References

Balas, E. & Ho, A. (1980), Set covering algorithms using cutting planes, heuristics, and subgradient optimization: A computational study, in M. Padberg, ed., ‘Combinatorial Optimization’, Vol. 12 of *Mathematical Programming Studies*, Springer Berlin Heidelberg, pp. 37–60.

Baldacci, R., Boschetti, M., Maniezzo, V. & Zamboni, M. (2005), Scatter search methods for the covering tour problem, in R. Sharda, S. Vo, C. Rego & B. Alidaee, eds, ‘Metaheuristic Optimization via Memory and Evolution’, Vol. 30 of *Operations Research/Computer Science Interfaces Series*, Springer US, pp. 59–91.

Brown, M. & Fintor, L. (1995), ‘US screening mammography services with mobile units: results from the national survey of mammography facilities’, *Radiology* 195(2), 529–532.

Foord, F. (1995), ‘Gambia: evaluation of the mobile health care service in West Kiang district’, *World health statistics quarterly. Rapport trimestriel de statistiques sanitaires mondiales* 48(1), 18–22.

Gendreau, M., Hertz, A. & Laporte, G. (1992), ‘New insertion and postoptimization procedures for the traveling salesman problem’, *Operations Research* 40(6), 1086–1094.

Gendreau, M., Laporte, G. & Semet, F. (1997), ‘The covering tour problem’, *Operations Research* 45(4), 568–576.

Hachicha, M., Hodgson, M. J., Laporte, G. & Semet, F. (2000), ‘Heuristics for the multi-vehicle covering tour problem’, *Computers and Operations research* 27(1), 29–42.

Hodgson, M. J., Laporte, G. & Semet, F. (1998), ‘A covering tour model for planning mobile health care facilities in Suhum District, Ghama’, *Journal of Regional Science* 38(4), 621–638.
Jozefowiez, N., Semet, F. & Talbi, E. (2004), ‘A multi-objective evolutionary algorithm for the covering tour problem’, *Applications of multi-objective evolutionary algorithms*, CA Coello and GB Lamont, eds., World Scientific pp. 247–267.

Jozefowiez, N., Semet, F. & Talbi, E. (2007), ‘The bi-objective covering tour problem’, *Computers & operations research* **34**(7), 1929–1942.

Labbé, M. & Laporte, G. (1986), ‘Maximizing user convenience and postal service efficiency in post box location’, *Belgian Journal of Operations Research Statistics and Computer Science* **26**, 21–35.

Laporte, G. (1992), ‘The vehicle routing problem: An overview of exact and approximate algorithms’, *European Journal of Operational Research* **59**(3), 345 – 358.

Lenstra, J. K. & Rinnooy Kan, A. H. G. (1975), ‘Some simple applications of the travelling salesman problem’, *Operational Research Quarterly* **26**(4), 717–733.

Motta, L., Ochi, L. & Martinhon, C. (2001), Grasp metaheuristics for the generalized covering tour problem, in ‘Proceedings of IV metaheuristic international conference, Porto, Portugal’, Vol. 1, Citeseer, pp. 387–93.

Oliveira, W. A. (2008), Construção de rotas para patrulhamento urbano preventivo, Master’s Thesis, Institute of Mathematics, Statistics and Scientific Computing, Department of Applied Mathematics, State University of Campinas, Campinas, Brazil.

Oppong, J. R. & Hodgson, M. J. (1994), ‘Spatial accessibility to health care facilities in Suhum District, Ghana’, *The Professional Geographer* **46**(2), 199–209.

Simms, J. C. (1989), ‘Fixed and mobile facilities in dairy practice’, *The Veterinary Clinics of North America - Food Animal Practice* **5**(3), 591–601.

Swaddiwudhipong, W., Chaovakiratipong, C., Nguntra, P., Mahasakpan, P., Lerdlukanavonge, P. & Koonchote, S. (1995), ‘Effect of a mobile unit on changes in knowledge and use of cervical cancer screening among rural Thai women’, *International Journal of Epidemiology* **24**(3), 493–498.