A Method for Optimizing Waste Collection Using Mathematical Programming: A Buenos Aires Case Study

Flavia Bonomo[1,2], Guillermo Durán[2,3,4], Federico Larumbe[1] and Javier Marenco[1,5]

[1] Departamento de Computación, FCEyN, Universidad de Buenos Aires, Argentina

[2] CONICET, Argentina

[3] Departamento de Matemática, FCEyN, Universidad de Buenos Aires, Argentina

[4] Departamento de Ingeniería Industrial, FCFM, Universidad de Chile, Chile

[5] Instituto de Ciencias, Universidad Nacional de General Sarmiento, Argentina

Corresponding author:

Prof. Guillermo Durán

Departamento de Matemática, FCEyN, Universidad de Buenos Aires

Ciudad Universitaria - pab. I

(1428) Buenos Aires - ARGENTINA

e-mail: gduran@dm.uba.ar

Abstract. A method is proposed that uses operations research techniques to optimize the routes of waste collection vehicles servicing dumpster or skip-type containers. The waste collection problem is reduced to the classic travelling salesman problem, which is then solved using the Concorde solver program. A case study applying the method to the collection system in the southern zone of Buenos Aires is also presented. In addition to the typical minimum distance criterion, the optimization problem incorporates the objective of reducing vehicle wear and tear as measured by the physics concept of mechanical work. The solution approach, employing graph theory and mathematical programming tools, is fully described and the data correction process is also discussed. The application of the proposed method minimized the distance travelled by each collection vehicle in the areas studied, with actual reductions ranging from 10% to 40% of the existing routes. The shortened distances led in turn to substantial decreases in work done and therefore in vehicle wear and tear. Extrapolation of the results to the entire southern zone of Buenos Aires indicates potential savings for the civic authorities of more than US$200,000 per year in addition to the qualitative impacts of less traffic disruption, less vehicle driver fatigue and less pollution.

Keywords: Environment, Mathematical programming, Waste collection, Vehicle routing, TSP.

1. Introduction.

Designing efficient urban waste collection systems has become a priority for local governments of major cities around the world due to concerns regarding pollution, public health and the environment as well as the budgetary impacts of the systems' transport, operating and labour costs. In this article we propose a method that uses operations research techniques to optimize the routes of waste collection vehicles servicing dumpster or skip-type containers. The waste collection problem is reduced to the classic travelling salesman problem, which is then solved using the Concorde solver program. A case study applying the method to the collection system in the southern zone of Buenos Aires (Argentina) is also presented.

There are three principal types of waste collection: residential, commercial and industrial (Golden, Assad and Wasil, 2002). Residential collection serves private homes; a single garbage truck is capable of servicing from 100 to 1,000 such residences per day. Though frequencies will vary from city to city, daily pickup is not unusual. Commercial collection, on the other hand, provides garbage removal for customers such as shopping centres, restaurants and office buildings, which may be assigned a pickup time window. Each commercial route can service between 60 and 400 customers daily with two or three visits to a transfer station or disposal site. Finally, industrial collection services factories, building sites and other major construction projects. What distinguishes it from commercial collection is that industrial waste containers tend to be four or five times bigger and often only one can be emptied per pickup. This poses a vehicle routing problem that is very different from the situation facing commercial collection services. The objectives of waste collection routing problems vary considerably, typical examples being to minimize the number of vehicles, minimize the distance covered, identify compact routes or minimize vehicle wear and tear.

A considerable literature exists on waste collection optimization. Liebman (1975) covers the entire range of problems on the subject, including simulation and optimization models for the installation of transfer, treatment and disposal facilities and for collection vehicle routing and human resource planning. Male and Liebman (1978) set out one of the first algorithms designed to simultaneously generate the zoning and vehicle routing for a garbage collection system.

A survey of works on waste collection vehicle routing may be found in (Kim, Kim and Sahoo, 2006). Eisenstein and Iyer (1997) present a “dynamic scheduling algorithm” that generates dynamic schedules for maximizing the service level in Chicago’s collection system. Chang, Lu and Wei (1997) propose an integer programming method integrated with a geographical information system for the city of Kaohsiung (Taiwan). Given the complexity inherent in garbage collection vehicle routing, some approaches have resorted to heuristic procedures, examples of which are Mourao and Almeida (2000) for Lisbon (Portugal) and Yeomans, Huang and Yoogalingam (2003) for the Canadian city of Hamilton. Finally, Arribas, Blazquez and Lamas (2010) develop a method combining heuristics with integer linear programming techniques that was successfully applied in Santiago, Chile.

A new residential waste collection system currently being implemented in Buenos Aires will require local residents to deposit their garbage into dumpster or skip-type containers located throughout the city. Each container has a capacity of 1,000 litres. The idea is to replace the old system of individual household waste receptacles with one that affords greater environmental protection and collection efficiency.

For organizational purposes Buenos Aires is divided into 6 waste collection zones as shown in Figure 1. The shaded areas are the parts of each zone where the new system is already in operation. Responsibility for managing collection in Zones 1 through 4 and Zone 6 has been assigned to individual private operators while in Zone 5, covering the city's southern district, the same duties are handled by the city government's urban sanitation authority known as the “Ente de Higiene Urbano” (EHU).

(Figure 1 here)

The precise locations of the containers are specified in advance. Those serviced by the EHU, which will be the exclusive focus of the present article, are grouped into 4 subzones. Each of the 4 is assigned a single collection vehicle, thus excluding zone definition from the scope of our investigation. The trucks make two identical trips daily, one in the morning and another in the evening. A trip begins when a vehicle starts out from the EHU in the direction of the first container in its subzone. After servicing all of its assigned containers it then heads to the transfer station to dump the collected waste before returning to the depot. Emptying a container and cleaning up in the immediate vicinity takes about 3 minutes. On the morning trip the trucks depart the EHU at 7 am and return at approximately 3 pm while on the evening trip they set out at 6 pm. The 4 subzones have 47, 133, 134 and 161 containers respectively, for a total of 375. The new system is slated for expansion in the short term to 20 subzones with 2,000 to 2,500 containers.

Each collection vehicle can hold 15,000 kg of waste. In the morning round each vehicle collects between 10,000 kg and 15,000 kg, whereas in the afternoon trip each one picks up from 2,500 kg to 5,000 kg. Every 10 days a truck follows behind the collection vehicles and cleans the containers as they are emptied. At the transfer station the garbage is compacted before being transported by larger vehicles to the final disposal site located in the greater urban area.

The purpose of this study is to improve the routes currently followed by the collection vehicles while maintaining the above-described general pattern in which each truck starts out from the EHU, empties all the containers in its assigned subzone, dumps the waste at the transfer station and returns to the EHU. The two objectives of this improvement are, in order of importance, the minimization of distance travelled and the reduction of vehicle wear. Since carrying large loads puts considerable strain on various parts of the vehicles, reducing their distance travelled while heavily loaded would bring substantial benefits to the system operator.

As regards labour costs, this route optimization has no effect, given that the waste collection workers are paid a fixed amount per shift. Shorter distances would presumably lessen driver fatigue, however, as well as diminishing vehicle wear, and would also mean less pollution and disruption of city traffic. The quantifiable economic savings are due to the reduction in fuel consumption, lower vehicle maintenance costs and longer vehicle service lives.

It should be noted that with the data currently available, travel times along the streets covered by the vehicles cannot be accurately estimated. For this reason, total distance travelled rather than travel time was chosen as the principal objective.

To quantify vehicle wear the concept of mechanical work is borrowed from physics. It is measured for a given route segment as the product of the distance travelled and the force exerted. The unit of this metric in the International System of Units is the Joule (J), formally defined as the force of one Newton acting to move an object through a distance of one metre. Thus, the work done by a vehicle over any segment between two consecutive containers is the distance separating them multiplied by the load carried, and the total work for the entire route is just the sum of the work done on each segment. To the best of our knowledge, no existing studies in the literature on the optimization of waste collection include the reduction of work done by the collection vehicles as a criterion.

This paper describes how computational tools were applied to effectively solve the optimization problem of the waste collection vehicles in the four EHU subzones using integer linear programming techniques. The computational tools and the correction and interpretation of the data are also discussed. As will be seen in Section 4, the proposed optimization process generated routes that were highly efficient in terms of the objectives adopted, with reductions of distance travelled and work done by the trucks of 10% to 45% compared to the existing routes.

The remainder of this article is organized as follows. Section 2 defines a natural representation of the map of the southern zone of Buenos Aires City with graph theory techniques. Section 3 shows how the waste collection problem can be reduced to the classic travelling salesman problem and briefly outlines the implementation of a software tool that carries out map processing tasks, calculates optimal routes and displays the results. Section 4 then presents these results, Section 5 sets out a sensitivity analysis of the impact of adding new waste collection vehicles to each city subzone, and finally, the article closes with Section 6 containing our conclusions and some indications for extensions of the work reported here.

2. A graph-theoretic representation of the Buenos Aires southern zone map

This section describes the data that was available to us and defines in detail how the map of the southern zone of Buenos Aires was represented by a graph. The representation is the heart of the entire implementation, laying the basis for the tasks of calculating vehicle routes and distances, validating the processed information and eliminating incorrect data.

Note first of all that the basic unit of the map is the city block, used here in the sense of a section of street between two consecutive cross streets. As in many cities, address numbers within a single block in Buenos Aires generally increase over an interval of 100, with odd numbers on one side and even on the other. Since the streets of the city tend to follow a grid pattern, adjacent blocks in parallel streets have the same address interval. This enables the location of any block along the length of a street to be identified in terms of its “100-block” number, hereafter denoted simply as the block number.

The map used was provided by the Buenos Aires city government in the form of a shapefile, one of the standard computer file formats for representing maps and geographical information. The file is a database that for every city block stores the locations of its corners, the name of the street, the initial and final block addresses, the traffic direction (one way or two way), and the positions of any traffic signals. Notation for this and the following section is included in Table 1.

(Table 1 here)

The traffic signal data contain a point indicating the approximate location of each traffic light. If two streets intersect at point *p* and the intersection thus formed has a traffic signal, there will also be a point in the database near *p* representing it. Since the presence of traffic signals is an important factor in determining possible routes, the optimization process must be able to find these locations. The process does this for a given intersection by searching for a point in the database whose distance from the intersection is less than a certain value *D*. After testing with various values it was found that traffic lights were correctly detected by setting *D*=3.5 metres.

2.1 Construction of the graph and calculation of travel distances

In what follows, a natural representation of the city map is described as a graph that allows vehicle distances travelled to be calculated. Since the calculations must take into account the city blocks' traffic directions, a digraph (a graph in which the arcs are directed) is employed. The nodes in the digraph represent the ends of each block and the container locations, which for our purposes are the significant vehicle positions.

If a vehicle is at an intersection, the two possible blocks it may be in (in one or the other of the two streets) are identified on the map as different positions. In a two-way street the position will depend on which side of the street it is on. Thus, vehicle position is defined in terms of the street and block number, the traffic direction and the exact position within the block. For example, a position might be specified as 700-block Rivadavia, even number side, 20 metres from the start or end of the block. Two nodes representing two positions have a directed arc joining them if a vehicle can travel directly from one node to the other. This means that a path in the graph represents a valid travel segment for a vehicle. The weight of an arc is the distance in metres between the positions representing its node ends.

An intersection of two two-way streets will have 8 positions, one for each direction of each the four blocks touching the intersection. If there are no prohibited turns, these points will connect and a vehicle can turn in either direction. However, Buenos Aires traffic regulations forbid left turns at intersections with a traffic light. This case is illustrated in Figure 2, indicating the permitted turns for a vehicle in Rivadavia Av. arriving at Boyacá St. Note that the 8 points in the figure are the 8 positions of the intersection; what varies is the block and/or traffic direction.

(Figure 2 here)

We now formally define a digraph $G = (V, A)$ that models the valid travel segments. Let $C = \{c_1=(s_1, p_1),..., c_n=(s_n, p_n)\}$ be the set of city blocks where $s_i \in$ {increasing, decreasing, two-way} and $p_i = (q(i_1),..., q(i_{t(i)}))$ is the vector of geographic points of block c_i for $i = 1,..., n$. Points $q(i_1)$ and $q(i_{t(i)})$ are the intersection corners and points $q(i_2),...,q(i_{t(i)-1})$ indicate the container locations in the block. The points for each block are ordered in the direction of increasing block numbers. If the direction is increasing a vehicle can travel from $q(i_1)$ to $q(i_2)$, from $q(i_2)$ to $q(i_3)$ and so on to $q(i_{t(i)})$. In the decreasing direction it can travel from $q(i_{t(i)})$ to $q(i_{t(i)-1})$, from $q(i_{t(i)-1})$ to $q(i_{t(i)-2})$ and so on to $q(i_1)$. On a two-way street the vehicle can travel in either direction.

Now let $Q_i = \{q(i_1),..., q(i_{t(i)})\}$ be the set of points in block c_i (i.e., the vector p_i considered as a set) and define M_i is the permitted traffic direction for block c_i as follows: $M_i = \{\textit{increasing}\}$ if $s_i = \textit{increasing}$; $M_i = \{\textit{decreasing}\}$ if $s_i = \textit{decreasing}$; $M_i = \{\textit{increasing}, \textit{decreasing}\}$ if $s_i = \textit{two-way}$.

For each block c_i the set of graph nodes is $V_i = \{c_i\} \times Q_i \times M_i$. The nodes are given by the various significant vehicle positions in the block as a function of the geographical points and the block traffic directions. In G the set of nodes is $V = \cup_{i=1}^{n} V_i$.

Two nodes $(c_i, q(i_k), m)$ and $(c_j, q(j_l), m')$ are said to be *consecutive* if and only if $i = j$, $m = m'$; $m = \textit{increasing} \Rightarrow l = k+1$ and $m = \textit{decreasing} \Rightarrow l = k-1$. Also, a node $(c_i, q(i_k), m) \in V$ is said to be an *exit end* of block $c_i = (s_i, p_i)$ with $p_i = (q(i_1),..., q(i_{t(i)}))$ if and only if $m =$

increasing $\Rightarrow k$ = t(i) and m = *decreasing* $\Rightarrow k$ = 1. Finally, a node (c_i, $q(i_k)$, m) $\in V$ is an *entry point* of block c_i if it is a corner and not an exit end.

Given the nodes $v_1 = (c_i, q(i_k), m) \in V$ and $v_2 = (c_j, q(j_l), m') \in V$, a *turn* is said to exist from v_1 to v_2 if $q(i_k) = q(j_l)$, v_1 is an exit end of c_i and v_2 is an entry point to c_j. To define the angle of the turn, we begin by letting $v_1' = (c_i, q(i_x), m) \in V$ and $v_2' = (c_j, q(j_y), m') \in V$ such that v_1' and v_1 are consecutive and v_2 and v_2' are also consecutive. The *turn angle* is then the angle between the half-line with origin $q(i_k)$ and direction $q(i_x)$ to $q(i_k)$ and the half-line with origin $q(i_k)$ passing through $q(j_y)$. The angle is considered to range over the interval $(-\pi,\pi]$ so that left turns are positive angles and right turns are negative ones.

To constrain arcs representing prohibited turns, we first define R as the set of exit points of blocks with traffic signals. Given two nodes $v_1 = (c_i, q(i_k), m) \in V$ and $v_2 = (c_j, q(j_l), m') \in V$, there is a *prohibited turn* from v_1 to v_2 if a turn from v_1 to v_2 is physically possible, $q(i_k) \in R$ and the turn angle is greater than $\pi/4$. Given two nodes $v_1 \in V$ and $v_2 \in V$, there is a *permitted turn* from v_1 to v_2 if a turn from v_1 to v_2 is physically possible and there is no prohibited turn from v_1 to v_2.

The arcs are now defined as the paths between two nodes such that either the second node is consecutive to the first one within the same block, or the turn from the first node to the second one is permitted, that is, $A = \{(v_1, v_2) \in V \times \mathrm{V}$: v_1 and v_2 are consecutive, or a turn is permitted from v_1 to $v_2\}$

Since the arcs indicate all permitted movements by a vehicle, applying a shortest path algorithm to the graph will generate a shortest valid trip that can be taken from one position to another within the city.

2.2 Location of containers

Included in the databases supplied by the city were four lists of current container locations, one for each subzone. The locations are listed in the order in which they are currently serviced by the collection vehicles and are indicated in Figure 3.

With this information, the length of the current routes can be calculated in order to compare them with the results obtained from our optimization method. Even if we do not know whether the path taken between any given consecutive pair of containers is the shortest possible one, we assume it is and then calculate lower bounds for the corresponding values of distance and work. Analogously, it can be also assumed that the paths between the EHU and the first container, the last container and the transfer station, and the transfer station and the EHU are also the shortest possible ones.

(Figure 3 here)

The locations indicated on the container lists were not fully standardized. Various locations were specified in terms of some nearby institution instead of a map reference while in other cases multiple variations of the same street name were used. These defects had to be rectified manually in the database before the information could be processed.

The lists were thus corrected so that all the container locations were given either by street name and block number or by intersecting streets. Once this task was completed, each location was translated into a map position (*street and block number, traffic direction, point*) in order to generate nodes representing the containers in the graph of the city.

In the case of a location denoted by street name and block number, the map position was defined by identifying the street and the block whose address interval [*initial block address*, *final block address*] contained that location. If the block was two way, the location was assigned either the traffic direction of increasing block numbers or the other direction, depending on whether the block address numbers were odd or even.

In the case of locations specified in terms of intersecting streets, the blocks at the intersection point were first identified. In the simplest instance there would be four such blocks, two in each street, of which one had to be chosen to define the container position. The criterion employed on the location lists is that the first street named is the one in which the container is actually found. Its precise position would then be in the block before the intersecting street, considering the vehicle circulation direction. If the first-named street is two way, the side of the street in which the container is located was defined arbitrarily.

As with any database of real-world phenomena, some of the map data were incorrect. The traffic directions for many of the streets in the four collection zones were verified, particularly those figuring in the optimal routes generated by the optimization process. This was done simply by comparing the data with other digital maps and aerial photos so that the database directions could be validated. These manual corrections helped ensure the map information actually used was as accurate as possible.

3. Solution strategy

Since the first objective of our optimization process was to minimize the distance travelled by each vehicle, the most natural option was to transform the vehicle route design problem into a travelling salesman problem (TSP). The TSP consists in finding the shortest Hamiltonian circuit (a trail that visits each node of a graph or digraph exactly once and returns to the starting node). In terms of computational complexity, the problem is NP-hard (Garey and Johnson, 1979), that is, there is no known polynomial-time algorithm to solve it.

Our second objective was to reduce the total work done by the vehicles as an approximation of wear and tear. Since the type of graph used here generally has multiple optimal solutions for the TSP, the strategy adopted was to find a set of optimal solutions – about 200 for each instance proved to be a suitable number as regards run times and solution quality – and then select the one with the lowest work value. For this approach to succeed, the implementation had to be able to solve the TSP to optimality and in a non-deterministic way. The *Concorde* solver program (Applegate et al., 2006) was chosen for the purpose.

The remainder of this section will describe the generation of the graph and the TSP solution. Section 3.1 discusses the construction of an instance of the shortest Hamiltonian path problem between two given nodes that models our problem; Section 3.2 derives the shortest vehicle paths between each pair of nodes in the digraph containing the containers, the EHU and the transfer station; and Section 3.3 explains the use of the *Concorde* software for solving the TSP on non-directed graphs. The problem must first be transformed into a symmetric TSP, however. This process is the subject of Section 3.4 and Section 3.5. Finally, Section 3.6 briefly outlines the implementation of a software tool that carries out map-processing tasks, calculates optimal routes and displays the results. For the notation used in this section, see Table 1.

3.1 Construction of Hamiltonian path instance

To model the waste vehicle problem we construct a weighted digraph in which the containers are represented by nodes. The weight of an arc from node A to node B is defined as the distance of the shortest vehicle path from container A to container B (see Section 3.2). The digraph also contains the EHU and the transfer station. An arc is added from the EHU to each container and from each container to the transfer station. The weights of these arcs are also defined as the distance of the shortest path from one element to another. The digraph is denoted $G_1 = (V_1, A_1)$ and $w_1 : A_1 \rightarrow \mathbf{R}$ is the distance function that associates each arc with its corresponding path distance. An example of such a digraph is depicted in Figure 4.

(Figure 4 here)

Now it is attempted to find a shortest Hamiltonian path in the digraph that starts from the node representing the EHU, visits all the container nodes and ends at the transfer station node.

3.2 Shortest path algorithm

Once the digraph associated with the city map has been defined, A^* (Hart, Nilsson and Raphael, 1968), a variant of Dijkstra's Algorithm (Dijkstra, 1959), is applied to obtain a shortest vehicle path between each pair of nodes in the digraph containing the containers, the EHU and the transfer station. The problem with the original Dijkstra's Algorithm is that during execution, the set of nodes waiting to be analyzed expands into a shape similar to a rhombus with the origin point as its centre. This means that the greater is the distance between the nodes to be analyzed, the less efficient will be the algorithm. For example, to calculate a shortest path between two points that turns out to be 10 km, Dijkstra's Algorithm will calculate the shortest paths to all points located at a distance less than 10 km from the origin. For our city digraph, executing this calculation in a SmallTalk implementation on a machine with an Intel Dual Core 1.60 GHz processor took more than an hour.

The A^* variant sidesteps this problem by using a different method of choosing the next node to be analyzed. Instead of just considering the distance from the origin, the heuristic also takes into account the distance to the destination. To ensure the method functions properly it exploits the property that the path distance is always greater than or equal to the Euclidean distance between the two points. Thus, at an intermediate point on the path to which the shortest path has already been calculated, the algorithm deduces that the distance of the shortest path from the origin to the destination is greater than or equal to the sum of the path distance from the origin to the intermediate point plus the Euclidean distance.

The suitability of A^* for the present application is clearly demonstrated by the fact that when this variant was used to calculate the 10 km example given above on the same computer, execution time fell drastically from more than 60 minutes to a mere 280 milliseconds.

3.3 Application of the Concorde solver

Concorde is a computer program written by David Applegate, Robert Bixby, Vašek Chvátal and William Cook at the Georgia Institute of Technology in the United States. It generates exact solutions for instances of the TSP using integer linear programming in conjunction with an LP solver such as *QSOpt*, an open source package developed by the same authors. According to its developers, Concorde can solve instances of up to 1,000 elements efficiently (Applegate et al., 2006). One instance of our problem with 98 nodes was solved by Concorde in two-tenths of a second while another containing 326 nodes was solved in 2 seconds. The program was run on a computer with an Intel Dual Core 1.60 GHz processor.

The program can also solve the TSP using the Chained Lin-Kernighan (Martin, Otto and Felten, 1996) heuristic, which is very efficient and provides optimal or near-optimal results for small instances. For our 98-node example it produced an optimal circuit in less than one second, an execution time similar to that for the exact algorithm. Finally, since

Concorde relies on randomness, the optimal solutions it generates will be different each time it is run.

3.4 Transformation of a Hamiltonian directed path into a Hamiltonian non-directed circuit

In Section 3.1 our problem was modelled as a shortest directed Hamiltonian path problem. *Concorde*, however, solves the TSP, which consists in finding a shortest non-directed Hamiltonian circuit, and takes as input the distance matrix of a complete graph. To solve the problem, the directed Hamiltonian path model is translated into a non-directed Hamiltonian circuit model. This will be done in two steps: first, the directed path is transformed into a directed circuit, and second, the directed circuit is transformed into a non-directed one.

We begin by calculating a shortest directed Hamiltonian path as a function of a shorter directed Hamiltonian circuit. In that circuit the EHU node must be located after the transfer station node. In other words, starting from the EHU node the circuit has to pass through all of the container nodes and then the transfer station node before returning to the EHU node. Since the digraph must be complete, $G_2=(V_2, A_2)$ is built from G_1 setting $V_2 = V_1$ and $A_2 = V_2 \times V_2$. A weight function $w_2 : A_2 \rightarrow \mathbf{R} \cup \{+\infty\}$ is defined as follows: $w_2((v_1, v_2)) = w_1((v_1, v_2))$ if $(v_1, v_2) \in A_1$; $w_2((v_1, v_2)) = +\infty$ if $(v_1, v_2) \notin A_1$, unless v_1=Transfer Station and v_2=EHU. In that last case, $w_2((v_1, v_2)) = 0$.

Thus, if there is no arc between any two nodes in G_1, an arc of infinite or zero weight is added to G_2 thereby making G_2 complete. Every Hamiltonian path in G_1 that starts at the

EHU node and ends at the transfer station node will therefore generate a Hamiltonian circuit in G_2 of finite distance, and vice versa.

3.5 Transformation of a digraph into a non-directed graph

The second step is to reduce the problem of finding a shortest Hamiltonian circuit in a digraph to one of identifying a shortest Hamiltonian circuit in a non-directed graph (Kumar and Li, 1996). The graph $G_3=(V_3, A_3)$ is defined with a *fictitious* node and a real node for each node in G_2. In formal terms, we assume that $V_2 = \{v_1,...,v_p\}$ and let $F = \{f_1,...,f_p\}$ so that node f_i is the fictitious node associated with v_i. We then define $V_3 = V_2 \cup F$ and $A_3 = V_3 \times V_3$. Let M be a sufficiently large number and let the weight function associated with the edges be $w_3 : A_3 \rightarrow \mathbf{R} \cup \{+\infty\}$, as follows: $w_3((x, y)) = +\infty$ if $x \in V$ and $y \in V$; $w_3((x, y)) = +\infty$ if $x \in F$ and $y \in F$; $w_3((x, y)) = -M$ if $x = v_i$ and $y = f_i$, for some i; $w_3((x, y)) = w_2(v_i, v_j)$ if $x = v_i$ and $y = f_j$ and $i \neq j$, for some i.

Figure 5 presents an example of the definition of a non-directed graph G_3 based on a digraph G_2 with M=10,000. G_3 has 3 real nodes and 3 fictitious nodes. The edges between the real nodes have a value of $+\infty$, which implies they will not be used by the shortest paths. The same holds for the edges between the fictitious nodes. A shortest path in this graph will always alternate between fictitious and real nodes. Moreover, since the edges between a real node and its corresponding fictitious one have a value of $-M$, the shortest paths will always use them. Since a shortest path will not contain arcs with infinite weight but will contain those with a negative weight, a real node will always be followed in a shortest path by its fictitious one.

(Figure 5)

The weight of the edges between a real node v_i and a fictitious one f_j with $i \neq j$ is the weight of the arc between v_i and v_j in G_2. This implies that the set of edges incident to the real node v_i in G_3 represents the exit arcs incident to v_i in G_2. Reciprocally, the set of edges incident to the fictitious node f_j in G_3 represents the entry arcs incident to v_j in G_2. Thus, a path (f_{i1}, v_{i1}, f_{i2}, v_{i2},..., f_{ik}, v_{ik}) in G_3 is interpreted as a path (v_{i1}, v_{i2},..., v_{ik}) in G_2. An example of the foregoing is given in Figure 6.

(Figure 6)

The procedures described in this section have taken us from a digraph G_1 for which we set out to calculate a shortest Hamiltonian path, to a complete digraph G_2 whose shortest Hamiltonian circuit enabled us to calculate the shortest Hamiltonian path in G_1. Then, by calculating the shortest Hamiltonian circuit of a complete non-directed graph G_3, the shortest Hamiltonian circuit in G_2 is easily obtained. In this manner the problem is reduced to the travelling salesman problem in a complete non-directed graph and the *Concorde* package can now be applied to efficiently obtain a solution.

3.6 Description of the implementation

A program was designed in the *SmallTalk* programming language to execute the map processing tasks, calculate optimal routes and display the results. The *VisualWorks NonCommercial, 7.4.1* implementation, available for the *Microsoft Windows XP/Vista* and *GNU-Linux* operating systems, was employed for the job.

The *VisualWorks* system contains a set of objects, classes and methods that were used to model the city map, the digraph of the containers and the algorithms for calculating the shortest paths and routes. It also has interfaces with the map database in *PostgreSQL*, the *Concorde* solver and the *Takenoko* graphical display package. The graphical interface of the computer application is shown in Figure 7.

(Figure 7)

To process the information, calculate the shortest routes and display them the system performs the following tasks:

1) Read the database of the city map.
2) Store the locations of the containers on each current route.
3) Execute the following steps for each of the 4 subzones:
   a) Calculate the shortest vehicle path between each pair of elements: EHU, containers and transfer station.
   b) Calculate the distance and work done for each current route.
   c) Construct digraph G_1 using the distances between the elements.

   d) Construct digraph G_2 based on G_1 as defined in Section 3.4.
   e) Construct complete graph G_3 based on G_2 as defined in Section 3.5.
   f) Execute *Concorde* with the G_3 distance matrix.
   g) Interpret the results to construct a shortest route.
4) Also, for each route the system can do the following:
   a) Calculate the distance.
   b) Calculate the work done.
   c) Generate a list indicating the order of the containers.
   d) Generate a list of the blocks travelled by the vehicle.
   e) Display an image of the map showing the route path.
   f) Display an animation of the vehicle route on the map.

4. Results and discussion

In this section the results are reported for the four instances of the problem corresponding to the four subzones, which as noted have 47, 133, 134 and 161 containers, respectively. The distances and work done for the routes currently used by the EHU are then compared to those of the shortest routes generated by our optimizations. For calculation purposes it is assumed that each container has an average weight of 100 kg, as suggested by EHU personnel.

Multiple runs of *Concorde* were executed for each instance, generating multiple shortest paths due to the program's use of randomness. The solutions with the lowest work values were retained as the definitive ones. The actual results are given in Table 2, showing for each subzone the distance (in meters) and work (in Joules) for the current EHU route and the corresponding shortest route solution (only the least work results are displayed). As can be seen, the shortest routes improve significantly on the existing ones in both distance and work.

(Table 2 here)

More specifically, the results show a decrease in total distance per day of about 120 km (60 km per trip), which together with the lower amount of work done by the vehicles translates into an annual saving in fuel consumption of some US$20,000. Given the city authorities' short-term goal of increasing the number of containers sixfold, an annual fuel cost reduction of approximately US$120,000 can be projected for the near future. The shorter distances also impact positively on vehicle maintenance and service lives. Based on the percentage decrease in work done, the drop in maintenance expense per vehicle can be estimated at US$3,000 while the prorated annual cost reduction due to extended service lives would be about US$1,500 per vehicle. Given that the number of vehicles should increase to 20 once all of the planned containers have been added, the estimated global annual savings due to the decrease in work expended is US$90,000. Together with the cut in fuel consumption, the City thus stands to save a total of more than US$200,000 per year.

Also evident from Table 2 is a very large gap between the degrees of improvement in subzones 1 and 4 owing to their different levels of complexity. The containers in Subzone 1 number less than one-third of those in Subzone 4 and are distributed among only 7 different streets. In such conditions, identifying a route approximating the shortest one is relatively intuitive. By contrast, Subzone 4 includes a densely packed area of containers, rendering the intuitive approach quite impossible in practical terms. Not surprisingly, then, the shortest route found through optimization resulted in a large improvement of 33.64%.

(Table 3 here)

To find optimal alternatives for each route, Concorde's TSP algorithm, which uses randomness, was run 200 times. For the simplest subzone, 4 different optima were found with values for work of 5.50×10^8, 5.39×10^8, 5.38×10^8 and 5.51×10^8 Joules respectively. For the other subzones, however, many optimal alternatives were generated. As an example, for Subzone 2 there were 190 alternatives, meaning that only 10 solutions were repeats. Though all of the solutions cover the same distance, the varying order in which the containers are serviced means the amount of work done will differ. Thus, among the various minimum distance routes we chose the one involving the least work. Table 3 summarizes a number of characteristics for the optimal solutions of each subzone, indicating the minimum, maximum and average work as well as the standard deviation (which gives an idea of the variation between iterations) and the execution times.

In a given solution run, the greatest amount of time was devoted to the calculation of the shortest path between each pair of elements. For example, in Subzone 4, the one with the most containers, the first TSP solution required 28 minutes, but once the shortest paths between the element pairs were computed, the remaining time to solution averaged just 2.49 seconds. If a container is added or moved, only 162 new paths would have to be calculated, which takes approximately 24 seconds; the exact time depends on the new container position. The time needed to calculate the shortest paths does not depend solely on the number of paths but also on their length. The more dispersed are the containers, the longer are the paths and, therefore, also the solution times.

These results justify the conclusion that the proposed optimization method for the waste collection problem was successful given the significant reductions obtained in both total distance and work done. If information on the traffic speeds in the streets were incorporated into the method, a similar improvement in route times could presumably be achieved.

5. Sensitivity analysis

In this section the impact of using more than one waste collection vehicle (presumably a smaller model) in each of the four original subzones is studied. To this end a new heuristic is implemented that divides these areas into a smaller subzones and assigns a single vehicle to each one. For example, to divide Subzone 2, which has 133 containers, into two new subzones of 67 and 66 containers, respectively, the heuristic would decide how to partition the containers between the subzones and then determine the corresponding values of distance and work.

In general terms, the heuristic assigns N vehicles to C containers via the following procedure. A TSP solution is first generated for all C containers. Then, based on the order specified by this solution, the container list is divided into two: the first C/N containers and the remaining ones. The first C/N containers constitute a new subzone and the original algorithm is executed on it to find the route of minimum work and distance. This procedure is repeated recursively with the remaining containers and N–1 vehicles, thus creating additional new subzones. The base case for each iteration is the newly created subzone in which a single vehicle is used for the corresponding container list. The subzone's optimal route is then calculated using the original algorithm.

This subzone partition heuristic was applied to each of the 4 original subzones. Since the shortest paths between all the pairs of containers for these subzones were already calculated the execution times were relatively short, ranging between 1 second and 7 minutes. To assign 3 vehicles to the original Subzone 2 (133 containers), for example, the optimal route is first found to determine the ordered container list. The first third of the list is then used to define a new subzone and calculate its route of minimum distance and work. This procedure is repeated with the remaining containers and 2 vehicles to generate the second and third new subzones, and the best routes are calculated for each.

(Table 4 here)

The effects of adding vehicles in this manner to the original subzones are shown in Table 4. In the case of Subzone 1, as well as the original case of 1 vehicle serving all 47 containers we obtained values for a partition into 2 subzones with 24 and 23 containers each. The table indicates the distance travelled and work expended in the new subzones and also gives the combined totals. As can be seen, the combined distance increased by 12 km over the single-vehicle solution. This was due to the fact that in addition to the distances covered to service the containers, this total includes the distances from the EHU to the respective subzones, from the subzones to the transfer station and from there back to the EHU. For each vehicle added these trip segments increase the total distance. The work done decreased considerably, however, from 5.385×10^8 to 4.489×10^8 joules. The reason for this drop is that each container emptied adds to the total load carried, and therefore the work done, over the rest of the route. Thus, if the priority is to reduce the amount of work rather than total distance travelled, more vehicles should be added.

The same behaviour is found for Subzone 2, where using 1, 2, 3, 30 and 133 vehicles was tested. The last of these cases implies a single vehicle for each container and gives the optimal result in terms of work done but with, of course, an extremely large increase in distance. The impacts using different numbers of smaller vehicles in subzones 3 and 4 are also set out in the table.

Note that since total work for an original subzone falls as vehicles are added, there will be less wear and tear on the vehicles. But the addition of these smaller trucks also involves acquisition costs and impacts on city traffic. Since the EHU authorities requested that we optimize the distance travelled, the original approach (a single vehicle for each of the 4 subzones) generated the best solutions. The heuristic just described could nevertheless be employed to analyze the use of smaller vehicles for related services such as special waste collection or container cleaning. It may also prove useful for defining new zones once the City's plans for additional containers have been implemented.

6. Conclusions and future research

The application of our optimization method resulted in significant improvement of the current waste vehicle routes. Route distances were reduced by up to 39% and work done, though not the variable being optimized, was cut by as much as 43%. This latter result was due in part to the fact that distance is one of the constituent factors of work, but also to the analysis of the various TSP solutions. Extrapolation of these outcomes to the entire southern zone of Buenos Aires indicates potential savings of more than US$200,000 annually in addition to less traffic disruption, less vehicle driver fatigue and less air pollution.

Much of this study consisted in modelling the graph and implementing the shortest path algorithm, taking into account all of the relevant details for producing routes on the city map. The graphical interface for creating images and animations was a key part of the implementation given that it enabled the information to be visualized in various forms, an essential capability when working with large amounts of information. Both the graphical interface and the data layer were abstracted into objects in the system so that if other technologies are used, the parts that have to be modified will be very localized.

A possible extension of this study would be to develop a program that calculates routes in real time. The program could then be used to recalculate the routes in response to eventualities such as street closures, traffic congestion or demonstrations. The efficiency of the proposed algorithms allows for such an implementation. Also, a GPS module could be included that would input the waste vehicle locations at any given moment to the program. Drivers equipped with mobile devices could then be notified of the next container location to be serviced. These additions would give the system the scalability required to accommodate new containers and vehicles.

To reduce vehicle wear even further, the optimization of total work done along a route could be formulated as the main objective of the problem. One approach to this would be to design heuristics that search for alternative routes using the optimal TSP solution as a starting point. Another possibility would be to employ column-generating algorithms to solve the work optimization problem (Lavigne et al., 1997).

The container waste collection project launched by the Buenos Aires City Government will add containers gradually over a period of years until every block in the city has at least one. This means that new zones and routes will have to be added to the model, a task that will raise the interesting problem of how to define the zone assigned to each vehicle. The solution might be to use the sum of the distances of the shortest routes as a way of validating the zone definition.

If information on the average traffic speeds in each block were added to the city map database, the time required to travel the length of a block could be determined. The program could then be used to calculate a shortest time route. The implementation of the A^* algorithm would have to be modified to use a different lower bound function such as the Euclidean distance multiplied by the average traffic speed.

Finally, above and beyond its specific results this study has demonstrated that operations research can be highly useful in implementing a system for the efficient organization of the waste collection service of this city.

Acknowledgements: We are grateful to Bill Cook for his many suggestions on the use of *Concorde*. Many thanks are also due to Daniel Iglesias, Jorge Takashashi and Julián Dunayevich of the Buenos Aires City Government for providing the necessary data, to Florencia Fernández Slezak and Kenneth Rivkin for their valuable collaboration, and to the anonymous referees for helpful suggestions that significantly improved the final version of this article. Partial support for this work was provided by ANPCyT Grant PICT-2007-00518, CONICET Grant PIP 112-200901-00160 and UBACyT Grant X069, Argentina. The first author was partially supported by ANPCyT Grant PICT-2007-00533 and UBACyT Grant

X606, Argentina. The second author was partially supported by FONDECyT Grant 1110797 and Millennium Science Institute “Complex Engineering Systems”, Chile.

References

D. L. Applegate, R. E. Bixby, V. Chvatal, and W. J. Cook (2006), *The Traveling Salesman Problem: A Computational Study*, Princeton University Press, Princeton, New Jersey.

C. Arribas, C. Blazquez, and A. Lamas (2010), Urban solid waste collection systems using mathematical modelling and tools of geographic information systems, *Waste Management & Research* 28 (4), 355–363.

N. Chang, H. Lu, and L. Wei (1997), GIS technology for vehicle routing and scheduling in solid waste collection systems, *Journal of Environmental Engineering* 123, 901–933.

E. W. Dijkstra (1959), A note on two problems in connexion with graphs, *Numerische Mathematik* 1 (1), 269–271.

D. Eisenstein and A. Iyer (1997), Garbage collection in Chicago: a dynamic scheduling model, *Management Science* 43, 922–933.

M. Garey and D. Johnson (1979), *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman and Company, San Francisco.

B. Golden, A. Assad, and E. Wasil (2002), Routing vehicles in the real world: applications in the solid waste, beverage, food, dairy, and newspaper industries, In: P. Toth and D. Vigo, editors, *The vehicle routing problem*, SIAM, Philadelphia, PA, 245–286.

P. E. Hart, N. J. Nilsson, and B. Raphael (1968), A formal basis for the heuristic determination of minimum cost paths, *IEEE Transactions on Systems Science and Cybernetics* 4 (2), 100–107.

B. Kim, S. Kim, and S. Sahoo (2006), Waste collection vehicle routing problem with time windows, *Computers and Operations Research* 33, 3624–3642.

R. Kumar and H. Li (1996), *On a Symmetric TSP: Transformation to Symmetric TSP and Performance Bound*, Manuscript.

J. Lavigne, G. Desaulniers, J. Desrosiers, and F. Soumis (1997), Basic Modeling with GENCOL, GERAD, GC-REF-002, 22 pages.

J. Liebman (1975), Models in Solid Waste Management. In: S. Gass and R. Sisson, editors, *A Guide to Models in Governmental Planning and Operations*, Sauger Books, Potomac, Md.

J. Male and J. Liebman (1978), Districting and Routing for Solid Waste Collection, *Journal of the Environmental Engineering Division* 104-1, 1–14.

O. Martin, S. W. Otto, and E. W. Felten (1996), Large-step Markov Chains for the Traveling Salesman Problem, *Complex Systems* 5, 299–326.

M. Mourao and M. Almeida (2000), Lower-bounding and heuristic methods for a refuse collection vehicle routing problem, *European Journal of Operational Research* 121, 420–434.

J. Yeomans, G. Huang and R. Yoogalingam (2003), Combining Simulation with Evolutionary Algorithms for Optimal Planning Under Uncertainty: An Application to Municipal Solid Waste Management Planning in the Regional Municipality of Hamilton-Wentworth, *Journal of Environmental Informatics* 2(1), 11–30.

Figures

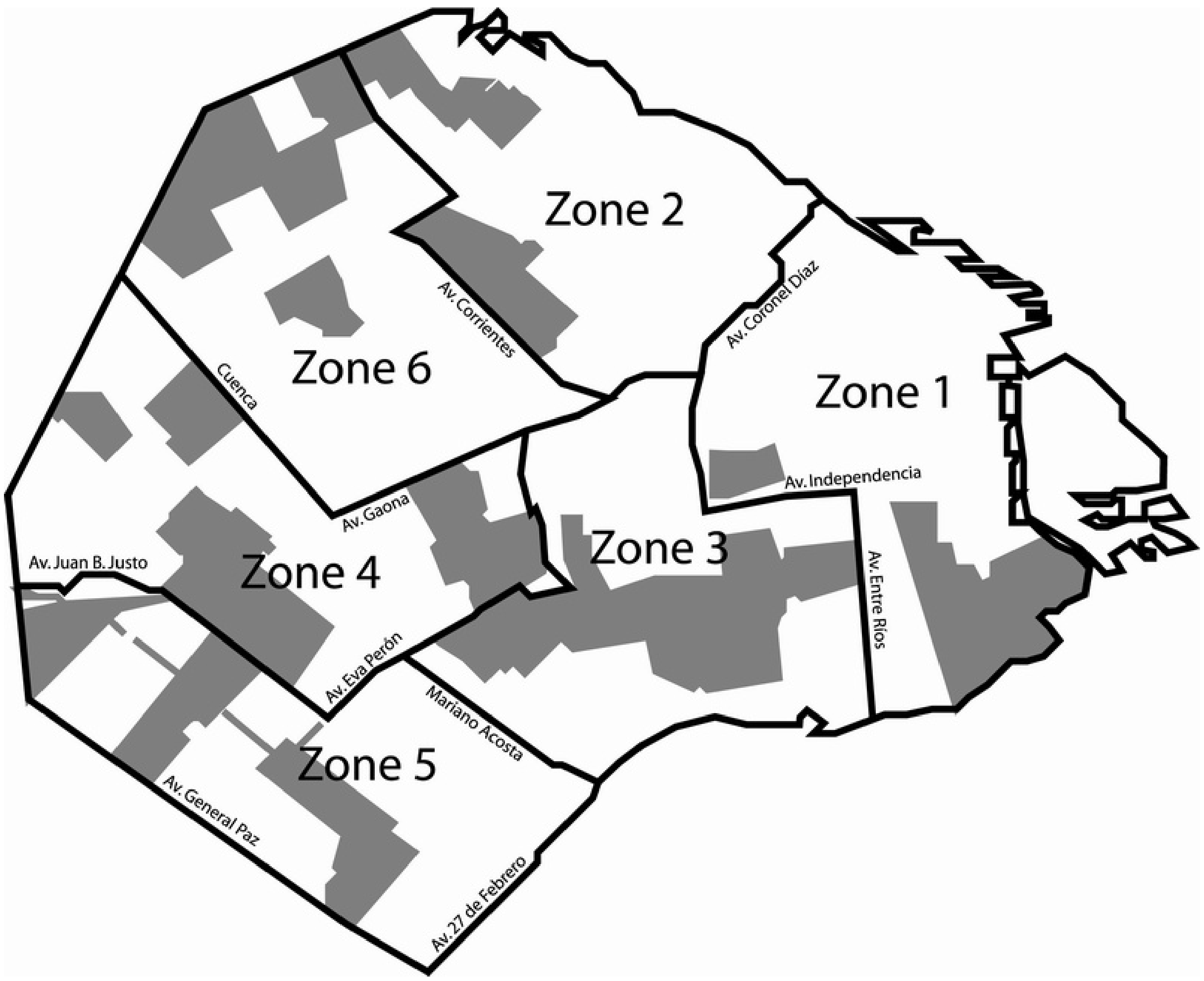

Figure 1. Waste collection zones in Buenos Aires. Scale 1:120,000 (1cm = 1.2km).

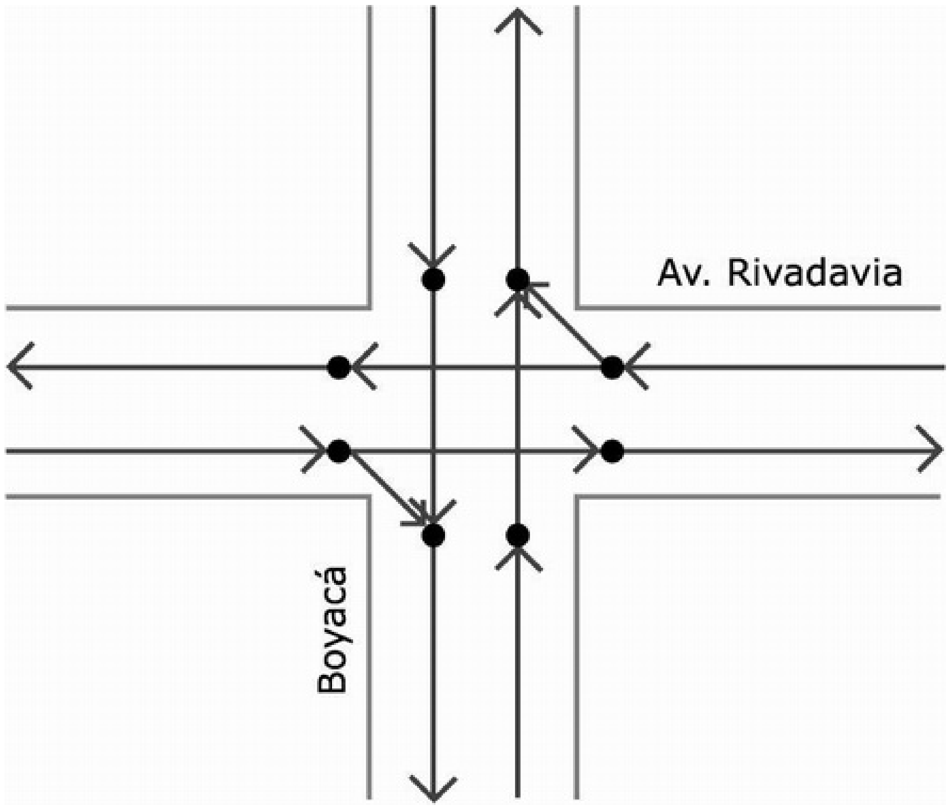

Figure 2. Turn arcs, Rivadavia Av. at Boyacá St. with traffic signal.

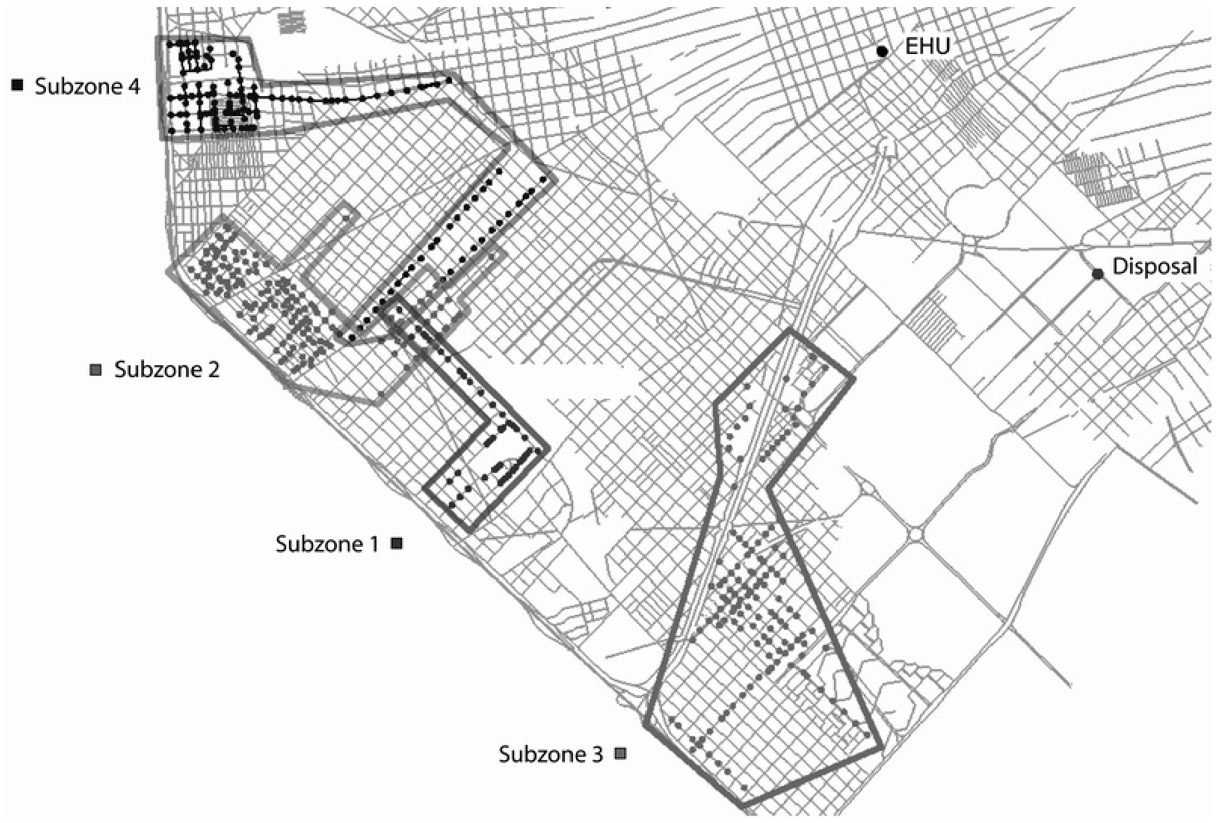

Figure 3. Current container locations by subzone. Scale 1:40,000 (1cm = 400m).

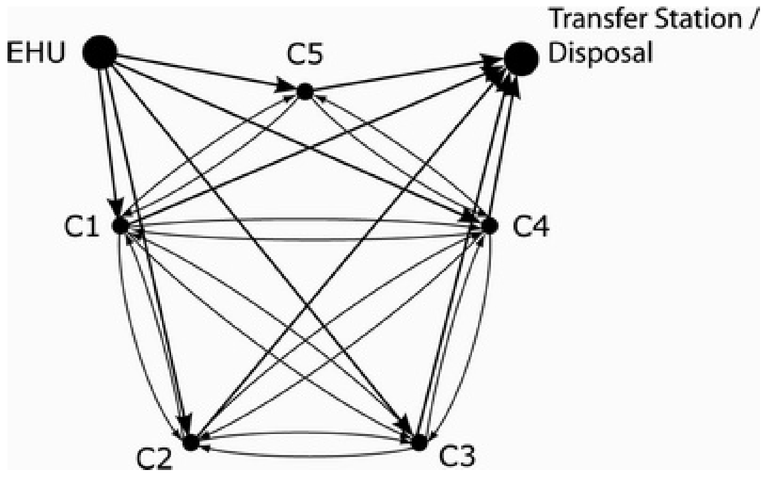

Figure 4. G_1. Example of a graph including the *Ente de Higiene Urbano* (EHU), five containers C1, ... , C5 and the transfer station.

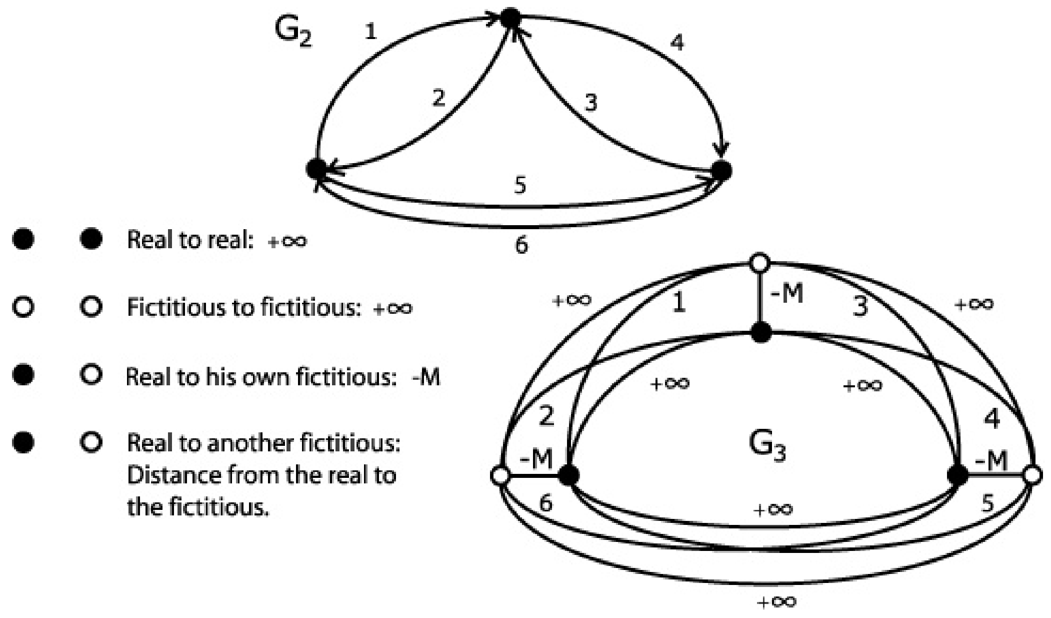
G2
1
2
3
4
5
6
● ● Real to real: +∞
○ ○ Fictitious to fictitious: +∞
● ○ Real to his own fictitious: -M
● ○ Real to another fictitious: Distance from the real to the fictitious.
G3
-M
+∞

Figure 5. Model digraph G_2 with a non-directed graph G_3. Example of the transformation defined in Section 3.4.

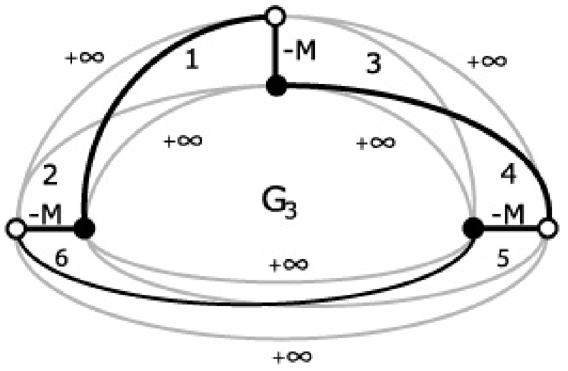

Figure 6. Cycle in the non-directed graph G_3 given by the transformation defined in Section 3.4. Length of the black path with fictitious arcs (arcs with one endpoint white and the other black): -3*M* + 11; length of the black path without fictitious arcs: 11.

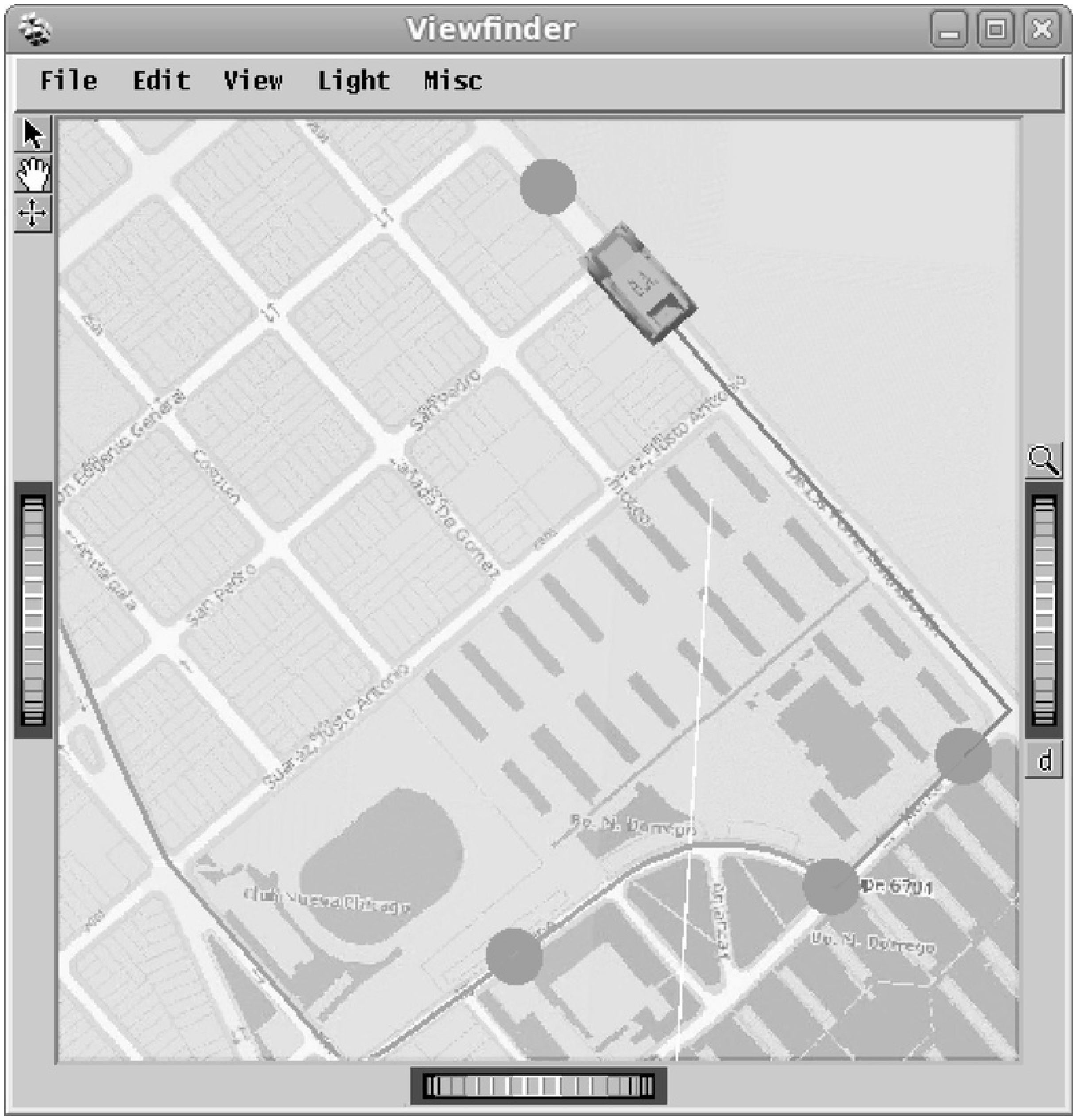

Figure 7. Graphical interface of the computer application. Scale 1:5,000 (1cm = 50m).

Symbol	Meaning
p	point in the map
D	threshold distance to associate traffic signals to street intersections (in meters)
$G = (V,A)$	digraph modelling the city
$C = \{c_1=(s_1, p_1),..., c_n=(s_n, p_n)\}$	city blocks
$s_i \in$ {increasing, decreasing, two-way}	possible street directions

$p_i = (q(i_1),\ldots, q(i_{t(i)}))$	vector of geographic points of block c_i for $i = 1,\ldots, n$
$q(i_1)$ and $q(i_{t(i)})$	intersection corners
$q(i_2),\ldots,q(i_{t(i)-1})$	container locations in the block
$Q_i = \{q(i_1),\ldots, q(i_{t(i)})\}$	set of points in block c_i (the vector p_i considered as a set)
M_i	set of permitted traffic directions for the block c_i
$V_i = \{c_i\} \times Q_i \times M_i$	set of graph nodes for the block c_i
$V = \cup_{i=1}^{n} V_i$	set of nodes of graph G
R	set of exit points of blocks with traffic signals
$A = \{(v_1, v_2) \in V \times V$: v_1 and v_2 are consecutive, or a turn is permitted from v_1 to $v_2\}$	set of arcs of graph G
$G_1 = (V_1, A_1)$	weighted digraph representing the city, the EHU and the transfer station
$w_1 : A_1 \to \mathbf{R}$	distance function associated to the arcs of G_1 (in meters)
$G_2=(V_2, A_2)$, $V_2 = V_1$, $A_2 = V_2 \times V_2$	complete weighted digraph defining an instance of Hamiltonian circuit
$w_2 : A_2 \to \mathbf{R} \cup \{+\infty\}$	weight function for G_2
$G_3=(V_3, A_3)$, $V_3 = V_2 \cup F$ and $A_3 = V_3 \times V_3$	complete weighted non-directed graph defining an instance of Hamiltonian circuit
$V_2 = \{v_1,\ldots,v_p\}$, $F = \{f_1,\ldots,f_p\}$	node f_i is the fictitious node associated with v_i
M	large number (greater than the sum of all the finite weights in G_2)
$w_3 : A_3 \to \mathbf{R} \cup \{+\infty\}$	weight function for G_3

Table 1. Notation for sections 2 and 3.

		Current route		**Shortest route**		**Percent improvement**	
	Containers	**Distance (Km)**	**Work (J) ($x10^9$)**	**Distance (Km)**	**Work (J) ($x10^9$)**	**Distance**	**Work**
1	47	27.010	0.61	24.126	0.54	10.68%	11.43%
2	133	68.102	4.77	41.752	2.95	38.69%	38.07%
3	134	52.270	4.28	39.762	2.84	23.93%	33.64%

4	161	61.692	5.44	40.841	3.11	33.80%	42.79%

Table 2. Results for the four subzones.

	Subzone	1	2	3	4
	Containers	47	133	134	161
	Optima found	4	190	180	172
Work (J) (x 10^9)	**Actual**	**0.61**	**4.77**	**4.28**	**5.44**
	Minimum	**0.54**	**2.95**	**2.84**	**3.11**
	Maximum	0.55	3.21	3.11	3.37
	Mean	0.54	3.13	2.93	3.21
	Standard Deviation	0.01	0.96	0.86	0.68
	Percent improvement	**11.43%**	**38.07%**	**33.64%**	**42.79%**
Execution time	**Initial iteration (min)**	9.23	28.72	29.14	28.42
	200 iterations (mln)	0.55	5.60	4.98	8.31
	Mean of iterations (sec)	0.16	1.68	1.49	2.49

Table 3. Selected characteristics of the 200 TSP iteration solutions. The execution time for the initial iteration includes the time consumed in calculating the minimum paths between each node pair. The execution time for the 200 iterations does not include the initial iteration time.

Subzone	**#Vehicles**	**#Containers**	**#Tours**	**Distance (Km)**	**Work (J) (x 10^8)**	**Ex.Time (s)**
1	1	47	4	24.13	5.385	36.47
	2	24	2	20.10	2.295	
		23	1	20.33	2.194	
		Total		40.43	4.489	37.88
2	1	133	191	41.75	29.47	383.40
	2	67	19	24.48	8.72	
		66	4	32.26	10.89	
		Total		56.74	19.61	121.65
	3	45	22	20.04	4.81	
		44	1	24.55	6.18	
		44	4	28.07	4.97	
		Total		72.66	15.96	126.33
	30	Total		489.43	9.05	103.32
	133	Total		2067.79	8.33	1.04
3	1	134	178	39.76	28.41	391.47
	2	67	10	29.03	11.9	
		67	1	28.92	8.61	
		Total		57.94	20.51	413.89
	3	45	18	26.47	6.54	
		45	21	27.17	6.5	
		44	1	22.47	4.51	

		Total		76.11	17.55	182.03
4	1	161	163	40.84	31.01	433.67
	2	81	49	29.70	12.22	
		80	9	27.86	10.49	
		Total		57.56	22.71	176.02
	3	54	4	25.00	67.36	
		54	33	26.35	68.89	
		53	3	24.63	61.39	
		Total		75.97	197.64	150.70

Table 4. Sensitivity analysis for the addition of vehicles in each subzone. The execution times refer to the 200 iterations and do not include the time consumed in calculating the minimum paths between node pairs, which are obtained previously.