On a Routing and Scheduling Problem Concerning Multiple Edge Traversals in Graphs

G. W. Groves
Department of Industrial Engineering, University of Stellenbosch, Private Bag X1, Matieland 7602, South Africa

J. le Roux
Department of Quantitative Management, University of South Africa, PO Box 392, UNISA, 0003, South Africa

J. H. van Vuuren
Department of Applied Mathematics, University of Stellenbosch, Private Bag X1, Matieland 7602, South Africa

Practical vehicle routing problems generally have both routing and scheduling aspects to consider. However, few heuristic methods exist that address both these complicated aspects simultaneously. We present heuristics to determine an efficient circular traversal of a weighted graph that requires a subset of its edges to be traversed, each a specified (potentially different) number of times. Consecutive time instances at which the same edge has to be traversed should additionally be spaced through a scheduling time window as evenly as possible, thus introducing a scheduling consideration to the problem. We present a route construction heuristic for the problem, based on well–known graph theoretic algorithms, as well as a route improvement heuristic, that accepts the solution generated by the construction heuristic as input and attempts to improve it in an iterative fashion. We apply the heuristics to various randomly generated problem instances, and interpret these test results. © 2005 Wiley Periodicals, Inc. NETWORKS, Vol. 46(2), 69–81 2005

Keywords: arc routing problems; network routing; chinese postman problem; rural postman problem

1. INTRODUCTION

Consider a weighted graph $\mathcal{G} = (V, E)$, with vertex set $V = \{v_1, \ldots, v_p\}$, edge set E, and edge weights denoted $c(i,j) \in \mathbb{R}^+$ for all $v_iv_j \in E$. The well–known Chinese Postman Problem (CPP) [28] is the problem of determining a minimum–weight circuit traversing each edge $v_iv_j \in E$ at least once. The CPP is tractable and may be solved in $O(|V|^3)$ time [13]. The Rural Postman Problem (RPP) is a generalization of the CPP in which a minimum–weight circuit is sought, traversing each edge in a subset $R \subseteq E$ at least once. The RPP is NP–Hard [31], except when $R = E$, in which case the problem reduces to the CPP. The problem definitions of the CPP and RPP have been generalized extensively, and procedures catering for directed and mixed graphs, for example, have been introduced. See Dror [12], Eiselt et al. [16, 17], and Ball et al. [2] for an overview.

Problems, such as those mentioned above, where visiting requirements are placed on the edges, or arcs, of a graph are known as arc routing problems (ARPs). They stand in contrast to vertex routing problems, where visiting requirements are placed on the vertices of a graph. Practical applications of ARPs include bus routing [1, 4, 5, 8, 10], meter reading [38, 40], control of plotting and drilling machines [24], optimization of laser–plotter beam movements [22], mail delivery [32, 37], garbage collection [3, 6, 21], street sweeping [7, 15], and snow gritting [14].

In this article we consider a problem that is a natural generalization of the RPP in the sense that we still seek to minimize the total circuit weight. However, (1) we no longer require each edge in the circuit to be traversed at least once, but rather require that each edge should be traversed at least a prespecified (potentially different) number of times and that (2) the time instances or positions within the circuit at which consecutive traversals of the same edge occur, should be spread over the whole circuit as evenly as possible, for all edges. This problem therefore has both a circuit cost and a spread objective, and these are typically conflicting, in the sense

Received January 2003; accepted April 2005
Correspondence to: J.H. van Vuuren; e-mail: vuuren@san.ac.za
Contract grant sponsor: South African National Research Foundation; Contract grant number: GUN2053755
Contract grant sponsor: Research Subcommittee B (university of stellen bosch)
DOI 10.1002/net.20073
Published online in Wiley InterScience (www.interscience.wiley.com).

that one might intuitively attempt to traverse an edge in the graph a sufficient number of times while one finds oneself in the vicinity of that edge, in a bid to minimize the total circuit weight. However, this would result in a solution with a bad spread. Conversely, separated consecutive traversals of the same edge over the circuit would typically increase the total circuit weight, due to an increased amount of free–running (which we shall call passive traversals) to reach edges that have to be traversed (actively) at an appropriate time or position within the circuit, thereby resulting in a bad circuit routing (cost). This problem has many potential applications, including:

1. Servicing of transportation networks, such as railway systems (consisting of metro rails, freight links, long distance lines, etc.) or road networks (consisting of suburban roads, dirt roads, highways, etc.). Maintenance of such networks typically requires routine servicing of each of their links. However, the service frequency of links in the network might vary with the link type. For example, it may be necessary to service dirt roads four times annually, while highways might only require servicing once a year. These service frequencies may be prescribed by law or by service company policy.
2. Snow ploughing or garbage removal in cities where certain locations need more frequent service than others. For example, it might be necessary to collect garbage more regularly from industrial sites than from residential ones, or busy main roads may need more frequent snow sweeping than quiet rural ones. These service frequencies are usually dictated by practical considerations.

The above-mentioned scheduling and routing problem is described generically in section 2, after which a nonlinear binary programming formulation is introduced, for solving a specific version of the problem, in section 3. However, the time complexities of standard solution techniques for binary programming problems may prohibit us from solving large problem graphs. A more focussed problem formulation, with a definition for spread more suited to practical problems, is introduced in section 4, and a simple construction heuristic is introduced for that problem in section 5. A local search heuristic is then presented in section 6, which may be used to improve the solution found by the heuristic in section 5. We present a number of test results in section 7 on randomly generated graphs within different graph structure classes. The graph instances that were used to obtain these test results are also available as benchmarks on the internet. Finally, we conclude the article, in section 8, by commenting on the efficiency of our solution procedure and by reflecting upon possible improvements.

2. PROBLEM DESCRIPTION

The problem under consideration may be described as an RPP with additional *spread* requirements, in the following manner:

Consider a weighted, order p graph $\mathcal{G}$, with vertex set $V(\mathcal{G}) = \{v_1, \ldots, v_p\}$ and edge set $E(\mathcal{G})$. Denote the weights associated with the edge v_iv_j of $\mathcal{G}$ by the tuple $(c(i,j), f(i,j))$, where $c(i,j)$ is referred to as the *cost weight* and $f(i,j)$ the *frequency weight*. We seek a closed route $\mathcal{R}$ that traverses each edge of $\mathcal{G}$ at least $f(i,j)$ times (each of these minimum number of traversals are called *active traversals*, while other traversals are called *passive traversals*). The sum of the cost weights of the edges in $\mathcal{R}$ must be as small as possible, while simultaneously ensuring that the active traversals of the same edges are separated from each other in $\mathcal{R}$ (this separation is called *spread*) in accordance to some defined criterion, for all edges in the graph.

It is clear from this description that, practically, the problem under consideration may be defined in many forms, depending on the criterion used for the spread objective and the manner in which the goal of optimizing the two objectives is treated. In the next section, a mathematical programming formulation is introduced, in which spread is measured in terms of the number of active traversals separating traversals of the same edge. This measure of spread is considered to be inadequate in practical situations, and therefore, a new version of the problem is defined in section 4, measuring spread in terms of temporal deviation within a specified time window.

3. MATHEMATICAL PROGRAMMING FORMULATION

Denote a solution to our generalisation of the RPP as a sequence $\mathcal{S} = \langle (v_{s_1}, v_{t_1}), (v_{s_2}, v_{t_2}), \ldots, (v_{s_n}, v_{t_n}) \rangle$ of actively traversed edges in the order in which they are traversed. Passive traversals are omitted from the sequence and are assumed to take place along routes corresponding to shortest distances between the edges of the sequence. Define the decision variables

$$x_{ij}^k = \begin{cases} 1 & \text{if } v_iv_j \text{ is the } k\text{th entry of } \mathcal{S}, \\ & \quad i.e. \text{ if } v_i = v_{s_k} \text{ and } v_j = v_{t_k} \\ 0 & \text{otherwise} \end{cases}$$

for all $k = 1, \ldots, n$ and all $i,j = 1, \ldots, p$, where

$$n = \sum_{v_iv_j \in E(\mathcal{G})} f(i,j)$$

denotes the length of the sequence $\mathcal{S}$. Then we attempt to construct a sequence $\mathcal{S}$, which minimizes the routing cost objective

$$\mathcal{R}(\mathcal{S}) = \sum_{k=1}^{n} \sum_{i,j=1}^{p} x_{ij}^k c(i,j) + \sum_{k=1}^{n-1} \sum_{i,j,l,m=1}^{p} x_{ij}^k x_{lm}^{k+1} d(j,l) + \sum_{i,j=1}^{p} x_{ij}^n d(j, s_1), \quad (3.1)$$

where $d(j,l)$ denotes the passive traversal cost of a shortest path between v_j and v_l in $\mathcal{G}$; hence, $d(j,l)$ is calculated by adding the weights $c(\cdot,\cdot)$ in any shortest path (as determined by a method such as Dijkstra's algorithm [11]) from v_j to v_l. To ensure that every edge of $\mathcal{G}$ has been actively traversed at least the required number of times, we require that the constraints

$$\sum_{k=1}^{n} \left(x_{ij}^{k} + x_{ji}^{k} \right) \geq f(i,j) \tag{3.2}$$

are satisfied for all $i,j = 1, \ldots, p$. The constraints

$$\left(\frac{n}{f(i,j)} - \epsilon_{ij} \right) x_{ij}^{k} x_{ij}^{\ell} \leq |\ell - k| \tag{3.3}$$

for all $k, \ell = 1, \ldots, n$ and $i,j = 1, \ldots, p$ ensure a spread between consecutive active traversals of the same edge, where $\varepsilon_{ij} \geq 0$ represents a tolerance within which solutions are deemed acceptable in terms of scheduling spread. If the above binary program has a feasible solution in the special case where $\varepsilon_{ij} = 0$ for all $i,j = 1, \ldots, p$ then the spread between all pairs of consecutive traversals of the same edge in $\mathcal{G}$ is ideal, for all edges $v_i v_j \in E(\mathcal{G})$. However, it may often not be the case that there exists a feasible solution for the program where $\varepsilon_{ij} = 0$. In such cases some of the tolerances will have to be positive. These tolerances may be user–specified constants, or may be incorporated as variables that are to be minimized, by altering the objective funcion (3.1) appropriately. Note that the nonlinear constraints (3.3) may be replaced by a milder set of linear constraints of the form

$$\sum_{k=r}^{r+b} \left(x_{ij}^{k} + x_{ji}^{k} \right) \leq 1, \tag{3.4}$$

which dictate that at least b edges are to be traversed between consecutive active traversals of the edge $v_i v_j \in E(\mathcal{G})$.

The problem may therefore be solved via an integer programming approach. However, a large number of branches may occur when solving this problem with a traditional technique, such as the branch–and–bound method, potentially rendering an integer programming approach impractical.

Furthermore, the method of measuring spread between consecutive active traversals of the same edge, as defined in (3.3), may not be appropriate in many practical applications. In practical applications, a separation in terms of the times at which an edge is serviced within some scheduling window is frequently desired. In these cases a count of the number of active traversals of other edges found between consecutive active traversals of the same edge may render an inadequate measure of temporal spread. We therefore reformulate the notion of spread in the next section, and define a version of the problem more suited to practical vehicle routing applications.

4. PROBLEM DEFINITION OF THE SMTPP

A more focussed version of the general problem described in section 2, thought to be relevant to problems involving the determination of vehicle routes in networks, is presented in this section. This incarnation of the problem is referred

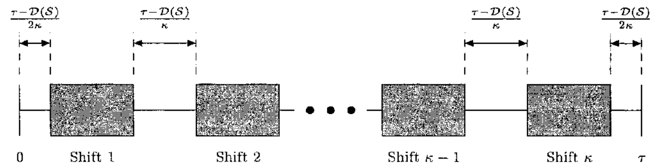

FIG. 1. The route schedule for a solution sequence $\mathcal{S}$, with total travelling time $\mathcal{D}(\mathcal{S})$. The schedule has κ shifts and a duration of τ time units.

to as the SMTPP (Scheduled Multiply Traversed Postman Problem), and defines spread in temporal terms. Given the weighted graph defined in section 2, the definition of the SMTPP is developed as follows.

The total routing cost of the route is still given by

$$\mathcal{C}(\mathcal{S}) = \sum_{i=1}^{n} c(s_i, t_i) + \sum_{j=1}^{n-1} d(t_j, s_{j+1}) + d(t_n, s_1), \tag{4.1}$$

where $d(k,l)$ denotes the cost of the shortest path between any two vertices v_k and v_l in $\mathcal{G}$, and $c(i,j)$ is the cost weight of the edge $v_i v_j$, as before. In the SMTPP, however, the notion of spread is defined in terms of the degree of separation of the time instances at which active traversals of edges commence, within the total scheduling window. Denote the scheduling window length by τ. The time taken to traverse an edge, denoted $p(i,j)$, is specified as an input parameter for the problem, and may, for example, be calculated from the speed of the vehicle. The total time spent traveling, denoted by $\mathcal{D}(\mathcal{S})$, is smaller than τ in a feasible solution. In practical cases the value of $\mathcal{D}(\mathcal{S})$ is often expected to be substantially smaller than τ. Particularly in these cases it becomes necessary to separate the route into portions, leaving periods of vehicle inactivity between the portions. Practically, these periods of inactivity correspond to times at which the vehicle may not work (e.g., after hours or public holidays) or simply to idle time. In the SMTPP, it is assumed that the route is separated into κ segments of equal duration (called *shifts*) that are evenly placed throughout the interval $[0, \tau]$ on the real line (where κ is a user–specified parameter), as shown graphically in Figure 1. The value of κ chosen by the user will reflect the scheduling needs in the application being modeled. If daily routes are sought in a weekly time window, then a logical choice of parameters is $\kappa = 7$, and $\tau = 7 \times 24 \times 60$ minutes.

Under the scheme depicted in Figure 1, the time instants at which the traversals of edges commence may be determined in $O(n)$ time, where n is the length of the solution sequence $\mathcal{S}$. Given the traversal commencement times of each edge, the spread of a solution is captured by the function

$$\mathcal{T}(\mathcal{S}, \tau, \kappa) = \left(\sum_{\substack{v_i v_j \in E(\mathcal{G}) \\ f(i,j) > 1}} \frac{\sum_{l=2}^{f(i,j)} \left(\frac{u_l^{(v_i,v_j)}(\mathcal{S},\tau,\kappa) - u_{l-1}^{(v_i,v_j)}(\mathcal{S},\tau,\kappa)}{\tau / f(i,j)} - 1 \right)^2}{f(i,j) - 1} \right)^{\frac{1}{2}}, \tag{4.2}$$

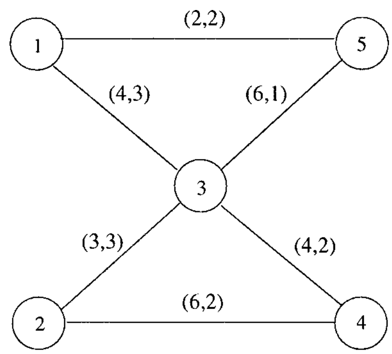

FIG. 2. Graphical representation of a small problem graph, $\mathcal{G}^*$.

where $u_l^{(v_i,v_j)}(\mathcal{S},\tau,\kappa)$ denotes the time instant at which active traversal of the l^{th} occurrence of edge v_iv_j in $\mathcal{S}$ commences. Edges that need to be traversed actively just once may be traversed at any time within the scheduling window, without influencing the temporal spread of the route, and consequently, edges with frequency $f(i,j) = 1$ are omitted from the objective $\mathcal{T}(\mathcal{S},\tau,\kappa)$. The objective takes its minimum value of 0 if and only if $u_l^{(v_i,v_j)}(\mathcal{S},\tau,\kappa) - u_{l-1}^{(v_i,v_j)}(\mathcal{S},\tau,\kappa) = \tau/f(i,j)$ for all edges $v_iv_j \in E(\mathcal{G})$, that is, when the temporal spread between consecutive active traversals of all edges are ideal (equal and maximal). Otherwise the value $\mathcal{T}(\mathcal{S},\tau,\kappa)$ is positive. The expression in (4.2) is the square root of the sum of the squares of the percentage deviations of all pairs of closest active traversals (of the same edge) from their ideal values. The function is similar to a standard deviation calculation, but differs in that it calculates the (linearised) second moment of the percentage temporal deviation values with respect to zero, and not with respect to their average values.

The functions (4.1) and (4.2) are the objectives of the SMTPP. The optimization approach taken in the SMTPP is to view the spread function (4.2) as a constraint and to attempt to minimise the distance function (4.1).

> The SMTPP is therefore the problem of determining a route $\mathcal{R}$, with corresponding solution sequence $\mathcal{S}$, of minimum total distance $C(\mathcal{S})$, for which $\mathcal{T}(\mathcal{S},\tau,\kappa) \leq \hat{\mathcal{T}}$.

Here $\hat{\mathcal{T}}$ denotes the threshold value for spread, a value above which a solution is considered to have an unacceptably high spread deviation. The choice of a value for $\hat{\mathcal{T}}$ is influenced by the user in the heuristics presented in this article. Specifically, the value of $\hat{\mathcal{T}}$ is set equal to a user–specified fraction, denoted c, of the $\mathcal{T}(\mathcal{S},\tau,\kappa)$ value of the solution obtained by the construction heuristic (described in the next section). The local search heuristic, described in section 6, is then used in an attempt to find a good solution for which $\mathcal{T}(\mathcal{S},\tau,\kappa) \leq \hat{\mathcal{T}}$. However, the construction heuristic is expected to yield a reasonable spread value in practice, and hence, not much user experimentation with values for c is expected.

Consider the small example graph $\mathcal{G}^*$ in Figure 2. Assume that the cost weights represent distances and are expressed in kilometres.

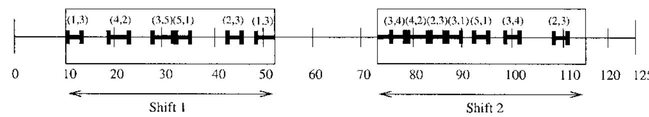

FIG. 3. Graphical representation of the schedule of the solution sequence $\mathcal{S}^*$.

The traversal durations of the edges are calculated by assuming an average vehicle speed of 60 km/h. The speed of 60 km/h is chosen to simplify the discussion, because it implies that the cost weights equal the traveling time (in minutes) of the edges [and hence, $\mathcal{C}(\mathcal{S}^*) = \mathcal{D}(\mathcal{S}^*)$]. Assume also that the scheduling window has length $\tau = 125$ minutes and is to be divided into two shifts.

A candidate solution to the SMTPP on the problem graph $\mathcal{G}^*$ is given by

$$\mathcal{S}^* = \langle(1,3),(4,2),(3,5),(5,1),(2,3),(1,3),(3,4),(4,2),(2,3),(3,1),(5,1),(3,4),(2,3)\rangle.$$

This sequence represents the full closed route **(1,3)**, (3,4), **(4,2)**, (2,3), **(3,5)**, **(5,1)**, (1,3), (3,2), **(2,3)**, (3,1), **(1,3)**, **(3,4)**, **(4,2)**, **(2,3)**, **(3,1)**, (1,5), **(5,1)**, (1,3), **(3,4)**, (4,2), **(2,3)**, (3,1). Here, bold-faced edges denote active traversals (present in $\mathcal{S}^*$), while passive traversals, typeset in normal font, are found by calculating shortest distance routes between nonadjacent active traversals in $\mathcal{S}^*$.

A graphical representation of the schedule corresponding to the route is shown in Figure 3. For this solution $\mathcal{C}(\mathcal{S}^*) = 85$ and $\mathcal{T}(\mathcal{S}^*, 125, 2) = 0.2939$. The traversal commencement times of all of the edges in $\mathcal{S}^*$ are displayed in Table 1.

5. CONSTRUCTION HEURISTIC FOR THE SMTPP

We introduce a simple construction heuristic solution procedure for the SMTPP, that operates by linking circuit segments through several copies of the graph $\mathcal{G}$. The method is first described in algorithmic fashion, followed by a more detailed explanation of each step.

TABLE 1. Traversal commencement times of the edges in $\mathcal{S}^*$, for the example SMTPP graph instance $\mathcal{G}^*$.

Position in $\mathcal{S}^*$	Edge	Traversal commencement time
1	(1,3)	10
2	(4,2)	18
3	(3,5)	27
4	(5,1)	33
5	(2,3)	42
6	(1,3)	49
7	(3,4)	73
8	(4,2)	77
9	(2,3)	83
10	(3,1)	86
11	(5,1)	92
12	(3,4)	98
13	(2,3)	108

5.1. Procedure: Construction Heuristic for the SMTPP

Inputs. (1) A weighted graph $\mathcal{G}$ of order p and with vertex set $V(\mathcal{G}) = \{v_1, \dots, v_p\}$, edge set $E(\mathcal{G})$ and edge weights $c(i,j)$ (cost weights), $f(i,j)$ (frequency weights), and $p(i,j)$ (traversal durations), for all edges $v_iv_j \in E(\mathcal{G})$. (2) Scheduling window length, τ. (3) Number of shifts used, κ.

Outputs. A closed route traversing each edge $v_iv_j \in E(\mathcal{G})$ at least $f(i,j)$ times and in which an attempt is made to spread out the traversal commencement times of the active traversals of each edge v_iv_j in the interval $[0, \tau]$.

1. Construct $N = 2f_{\max}$ copies of the graph $\mathcal{G}$, where
$$f_{\max} = \max_{v_iv_j \in \mathcal{G}} \{f(i,j)\}. \tag{5.1}$$
Assign to each copy of the graph a unique index from 1 through N.
2. Assign to each edge of $\mathcal{G}_k$ the status *passive*, $k = 1, \dots, N$.
3. Repeat the following until all edges of the original graph $\mathcal{G}$ have been selected: (a) From the edges of the problem graph $\mathcal{G}$ that have not yet been selected, select an edge v_iv_j randomly. (b) Choose $f(i,j)$ indices between 1 and N (inclusive) according to a predetermined selection rule (discussed later). (c) For each of these $f(i,j)$ indices chosen, assign to the edge v_iv_j the status *active* in the corresponding indexed copies of the problem graph.
4. Identify connecting vertices between the indexed copies of the problem graph (chosen according to a method described later). These vertices define the starting and ending points of subroutes to be found in each indexed graph during the next step. Any vertex incident to an active edge in $\mathcal{G}_1$ is selected to be both the starting vertex of $\mathcal{G}_1$ and the ending vertex of $\mathcal{G}_N$.
5. Find, in $\mathcal{G}_k$, a subroute that traverses the edges marked *active* at least once and that starts and ends at the vertices identified in the previous step, for all $k = 1, \dots, N$. A modified version of Frederickson's heuristic is used to find this subroute.
6. Construct a route $\mathcal{S} = \langle (v_{s_1}, v_{t_1}), (v_{s_2}, v_{t_2}), \dots, (v_{s_n}, v_{t_n}) \rangle$ by linking the consecutively numbered subroutes through their ending and starting vertices.
7. Determine the traversal commencement times of the edges in $\mathcal{S}$ according to the scheme depicted in Figure 1. ■

The method uses $N = 2f_{\max}$ copies of the problem graph in step 1 to ensure that the same edge is never marked *active* in consecutively indexed copies of the problem graph. The selection rule in step 3(b) chooses the set of $f(i,j)$ indexed graphs that has the smallest total sum of $p(i,j)$ values for the active edges already assigned to them, while ensuring a gap of at least $\lfloor N/f(i,j) - 1 \rfloor$ or alternatively $\lceil N/f(i,j) - 1 \rceil$ between the $f(i,j)$ graph copy indices. The decision of whether to use $\lfloor N/f(i,j) - 1 \rfloor$ or $\lceil N/f(i,j) - 1 \rceil$ gaps between closest indices is made as follows: the value $N/f(i,j) - 1$ is rounded to the nearest integer, and that number of gaps is used (starting from the smaller index and proceeding) until the sum of the rounding errors (i.e., the difference between the number of gaps used and $N/f(i,j) - 1$) exceeds the rounding error of rounding $N/f(i,j) - 1$ in the opposite direction. From that point onward the other rounded value is used once and the process repeats itself. Note that a computer implementation of the heuristic would not wastefully store the N indexed copies of the problem graph but would instead store only the information about which edges are active in each graph. In step 4 the starting and ending vertices identified for consecutively indexed graph copies are those pairs of vertices (one in each graph copy), considering only vertices incident to *active* edges, that have the cheapest traversal cost between them. Because a closed route is sought, the same vertex is chosen to be the starting vertex of the first graph and ending vertex of the last graph. Step 5 implements a version of Frederickson's heuristic for the RPP [20], that is modified to find a route that starts and ends at specified (potentially different) vertices.

The construction heuristic is applied to the graph of Figure 2 to illustrate its operation. The same input parameters as those used in section 4 are used, and therefore the scheduling window length, τ, equals 125 and the number of shifts used, κ, equals 2. The edge cost weights are assumed to be expressed in kilometres, and a constant vehicle speed of 60 km/h is used.

The largest number of active traversals required for any edge in the graph is 3, and therefore, step 1 of the algorithm constructs six copies of the graph, indexed $\mathcal{G}_1^*, \dots, \mathcal{G}_6^*$. In each of these graphs, some of the edges are designated *active*, according to the selection rule [step 3(b)] described earlier. The active edges for each of these graph copies are depicted by means of bold faced lines in Figure 4. The vertices at

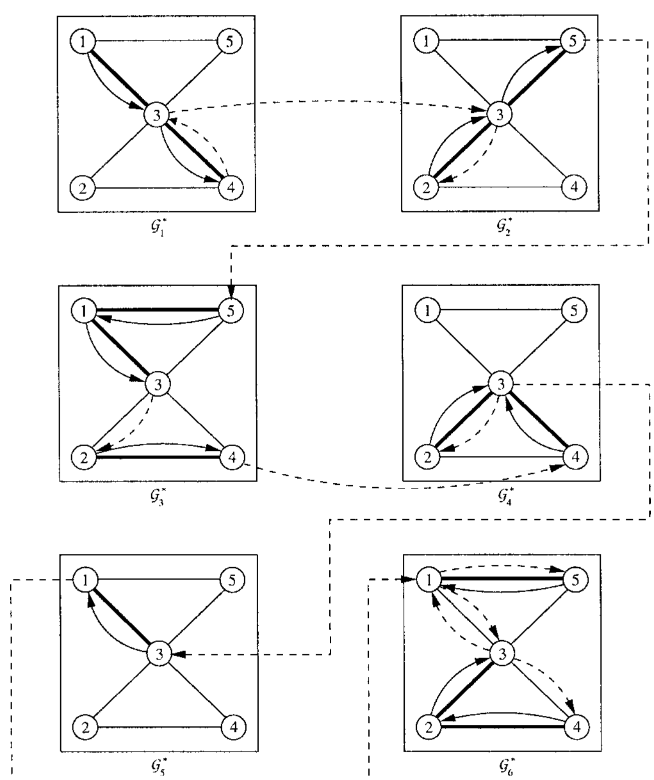

FIG. 4. Heuristic solution (represented by the path followed by the arrows). Edges that need to be actively traversed are shown as boldfaced edges in each of the indexed graphs $\mathcal{G}_k^*$, $k = 1, \dots, 6$.

TABLE 2. Routes within the indexed graphs $\mathcal{G}_i^*$, $i = 1, \ldots, 6$.

Graph	Starting vertex	Ending vertex	Route
$\mathcal{G}_1^*$	1	3	$\langle(\mathbf{1,3}), (\mathbf{3,4}), (4,3)\rangle$
$\mathcal{G}_2^*$	3	5	$\langle(3,2), (\mathbf{2,3}), (\mathbf{3,5})\rangle$
$\mathcal{G}_3^*$	5	4	$\langle(\mathbf{5,1}), (\mathbf{1,3}), (3,2), (\mathbf{2,4})\rangle$
$\mathcal{G}_4^*$	4	3	$\langle(\mathbf{4,3}), (3,2), (\mathbf{2,3})\rangle$
$\mathcal{G}_5^*$	3	1	$\langle(\mathbf{3,1})\rangle$
$\mathcal{G}_6^*$	1	1	$\langle(1,5), (\mathbf{5,1}), (1,3), (3,4), (\mathbf{4,2}), (\mathbf{2,3}), (3,1)\rangle$

Edges that are actively traversed are shown in bold face.

which the subroutes in each of the indexed graph copies start and end are identified next (step 4). These vertices are listed in Table 2.

A route is now found in each of these indexed graphs according to the modified version of Frederickson's heuristic. The resultant route in each graph is also shown in Table 2. The heuristic solution is found, starting from vertex 1, by traversing each of these subroutes and returning to vertex 1. This closed route is given in coded form by

$$\mathcal{S}_0^* = \langle(1,3), (3,4), (2,3), (3,5), (5,1), (1,3), (2,4), (4,3), (2,3), (3,1), (5,1), (4,2), (2,3)\rangle.$$

The full solution is depicted by the traversal in Figure 4. A schedule of the route is shown in Figure 5. The traversal commencement times for the edges in $\mathcal{S}_0^*$ are shown in Table 3. The total weight of this solution is 77 cost units, which is better than that of the feasible solution presented in section 4. The temporal spread of the solution equals 0.3098.

6. LOCAL SEARCH HEURISTIC FOR THE SMTPP

The class of *local search methods* is a family of heuristics that operate by iteratively making transformations (referred to as *moves*) to a candidate solution in a way that tends to improve the solution as the search progresses. Typically, a local search heuristic operates by considering a number of candidate moves during an iteration, and selects the best one to perform on the solution. During the next iteration, the process is repeated on the transformed solution.

The broad procedure according to which moves are made, for the SMTPP, is described in section 6.1. A procedure that forms part of the process of performing a move is described next, in section 6.2. This is followed, in section 6.3, by a pseudocode listing of the local search procedure for the SMTPP. Finally, an example illustrating the application of the heuristic to a small SMTPP instance, is presented in section 6.4.

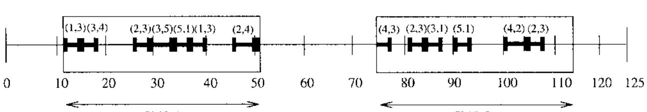

FIG. 5. Graphical representation of the traversal commencement times of $\mathcal{S}_0^*$, the solution obtained by the construction heuristic for the example SMTPP instance.

TABLE 3. Traversal commencement times of the edges in $\mathcal{S}_0^*$, obtained by the construction heuristic, for the example SMTPP graph instance $\mathcal{G}^*$.

Position in $\mathcal{S}^*$	Edge	Traversal commencement time
1	(1,3)	12
2	(3,4)	16
3	(2,3)	26
4	(3,5)	29
5	(5,1)	35
6	(1,3)	37
7	(2,4)	44
8	(4,3)	50
9	(2,3)	81
10	(3,1)	84
11	(5,1)	90
12	(4,2)	100
13	(2,3)	106

6.1. *Performing Local Search Moves*

The method according to which moves are performed allows move types to be used that directly specify the order in which required edges are traversed in the changed solution, for some general ARP. An example of such a move type is one that simply exchanges the order in which two required edges are traversed. Given, for example, the following route

$$\mathcal{S}^1 = \langle(3,4), (4,1), (5,6), (5,4), (6,8)\rangle,$$

edges $(3,4)$ and $(5,4)$ might be exchanged, to yield

$$\mathcal{S}^2 = \langle(5,4), (4,1), (5,6), (3,4), (6,8)\rangle.$$

Typically, such a move type would consider all pairs of these exchanges and then perform the one that yields the route of minimum overall cost. During the previous exchange, the traversal directions of the required edges are not altered, and it may be better to traverse the edge $(3,4)$ (for example) in the direction $(4,3)$ instead of in the direction $(3,4)$. Consequently, it is necessary to determine the optimal traversal directions after the exchange. Applying a move therefore involves altering the order of the required edges in the route, and then determining their direction of traversal. The method described in the next section illustrates how the traversal directions may be determined.

The local search heuristic heuristic for the SMTPP makes use of a move type analogous to the Two–Opt move type [9, 18], for the *traveling salesman problem*. The basic operation of the move type, for some general ARP, is shown in Figure 6. The vertex 0 represents the "domicile vertex" of a closed route, and may correspond to a vehicle depot in a practical application. It may be omitted in problems where a particular vertex is not specified.

6.2. *Determining Optimal Traversal Directions*

The algorithm described in this section computes the optimal traversal directions for the edges in a solution sequence $\mathcal{S}$, given a fixed order of the edges in the sequence.

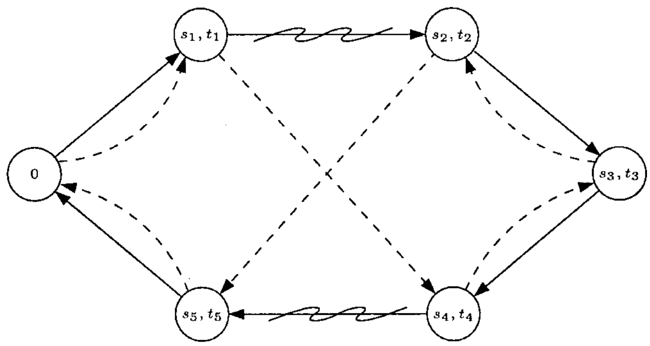

FIG. 6. The mechanism behind the Two-Opt move. The edges (s_1t_1, s_2t_2) and (s_4t_4, s_5t_5) may, for example, be removed from the cycle $(0, s_1t_1)$, (s_1t_1, s_2t_2), (s_2t_2, s_3t_3), (s_3t_3, s_4t_4), (s_4t_4, s_5t_5), $(s_5t_5, 0)$, and the traversal reconnected to form the alternative cycle $(0, s_1t_1)$, (s_1t_1, s_4t_4), (s_4t_4, s_3t_3), (s_3t_3, s_2t_2), (s_2t_2, s_5t_5), $(s_5t_5, 0)$.

The algorithm may be applied each time after a local search move has been made to yield the shortest distance for the new ordering of entries within $\mathcal{S}$.

Consider a routing sequence $\mathcal{S} = \langle(v_{s_1}, v_{t_1}), (v_{s_2}, v_{t_2}), \ldots, (v_{s_n}, v_{t_n})\rangle$ for a general ARP. From the solution sequence, construct a directed, layered auxilliary graph $\mathcal{L}$ with vertex set $V(\mathcal{L}) = \{b, s_1t_1, t_1s_1, s_2t_2, t_2s_2, \ldots, s_nt_n, t_ns_n, e\}$. The first and last layer of the auxilliary graph consist of a single vertex, and the other layers consist of two vertices. Each layer of the graph represents the two possible active traversal directions (v_{s_i}, v_{t_i}) and (v_{t_i}, v_{s_i}), $1 \leq i \leq n$ of an edge in $\mathcal{S}$. The vertices b and e in $\mathcal{L}$ represent the vertices in $\mathcal{G}$ at which the route is to begin and end. For a closed route $v_b = v_e$, and in the RPP these vertices are adjacent to a required edge.

For each i $(0 < i < n)$ construct edges in $\mathcal{L}$ directed from s_it_i to $s_{i+1}t_{i+1}$ and $t_{i+1}s_{i+1}$, and from t_is_i to $s_{i+1}t_{i+1}$ and $t_{i+1}s_{i+1}$. Also, add edges directed from b to s_1t_1 and t_1s_1 and from s_nt_n and t_ns_n to e. Assign a weight of $d(t_i, s_{i+1})$ $[d(s_i, t_{i+1})$, respectively$]$ to the edge in $\mathcal{L}$ between s_it_i and $s_{i+1}t_{i+1}$ $[t_is_i$ and $t_{i+1}s_{i+1}$, respectively$]$ for every $0 < i < n$, where $d(i,j)$ denotes the weight of a shortest path from i to j. Similarly assign a weight of $d(t_i, t_{i+1})$ $[d(s_i, s_{i+1})$, respectively$]$ to the edge in $\mathcal{L}$ between s_it_i and $t_{i+1}s_{i+1}$ $[t_is_i$ and $s_{i+1}t_{i+1}$, respectively$]$ for every $0 < i < n$. Finally, assign a weight of $d(b, s_1)$ $[d(b, t_1)$, respectively$]$ to the edge in $\mathcal{L}$ between b and s_1t_1 $[t_1s_1$, respectively$]$ and a weight $d(t_n, e)$ $[d(s_n, e)$, respectively$]$ to the edge between s_nt_n $[t_ns_n$, respectively$]$ and e. This construction is shown graphically in Figure 7.

Each route from b to e in $\mathcal{L}$ represents one way of arranging the active traversal directions of edges within $\mathcal{S}$, and a shortest path from b to e represents a set of optimal directions by which to traverse the edges of $\mathcal{S}$. For example, if vertex t_2s_2 is on the calculated shortest path, then the second edge of $\mathcal{S}$ should be actively traversed from v_{t_2} to v_{s_2}, and hence, coded as (v_{t_2}, s_{t_2}) in $\mathcal{S}$. Note that the total weight of the route may be found by adding the sum of the weights of the required edges to the weight of the shortest path. This method for performing local search moves for ARPs has been proposed independently by Groves et al. [25, 26], and by Lacomme et al. [29, 36].

The computational complexity of finding a shortest path in a directed, acyclic graph, such as $\mathcal{L}$, is $O(|E(\mathcal{L})| + |V(\mathcal{L})|)$ (see, e.g., Gondran and Minoux [23]), because no updating of information, such as occurs in Dijkstra's [11] or Floyd's [19] methods, is necessary during the algorithm execution. Here, $E(\mathcal{L})$ is the edge set of $\mathcal{L}$ and $V(\mathcal{L})$ the vertex set, as before. However, because no earlier layer of $\mathcal{L}$ can be reached from a later layer, and because each layer consists of a predetermined number of vertices, and is connected to the other layers in the particular manner shown, this complexity can be reduced to $O(|V(\mathcal{L})|)$.

If a shortest path through $\mathcal{L}$ is calculated each time that a move is evaluated, the computational complexity of the Two–Opt move type is $O(|\mathcal{S}|^3)$ per iteration. Note, however, that the length of this shortest path can be determined in a constant amount of time if the shortest distances from each vertex to all vertices in later layers in $\mathcal{L}$ is known for the untransformed solution (see Groves and van Vuuren [27] for details). This allows, for example, Two–Opt moves with $O(|\mathcal{S}|^2)$ time complexity per iteration, to be used in heuristics for the RPP (or to optimize an individual route in a constrained ARP). In the SMTPP, however, the mentioned time-saving method cannot be used, due to the requirements of the spread calculation. Nevertheless, other methods that exploit the fact that some distance labels remain unchanged between iterations can be used to speed up execution (see Groves [25] for details).

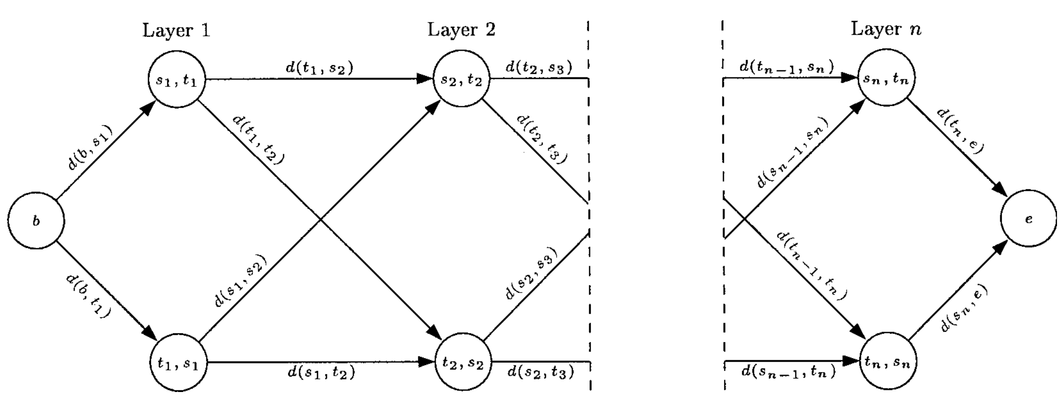

FIG. 7. The layered graph, $\mathcal{L}$, corresponding to the solution sequence $\mathcal{S} = \langle(v_{s_1}, v_{t_1}), (v_{s_2}, v_{t_2}), \ldots, (v_{s_n}, v_{t_n})\rangle$.

6.3. Description of Local Search Heuristic

The heuristic described in this section uses the solution generated by the construction heuristic, presented in section 5, as a starting solution and attempts to improve it in an iterative fashion using Two–Opt transformations.

The Two–Opt heuristic differs from a standard implementation in that solutions are not compared purely on their cost objective functions. A solution is considered to be better than the current best encountered solution in one of two cases, depending on whether the spread of the best encountered solution is more than $\hat{T}$ or not. If the spread of the best encountered solution is infeasible, then the solution is better if its spread value is better, regardless of the value of its cost objective function. However, if the spread value of the best encountered solution is feasible, then a solution is better if its cost objective function value is less (provided its spread value is feasible) or if the cost objective functions are equal but the solution has a better spread value.

When selecting the best move to make, the heuristic does not take into account the case where the traveling time [i.e., $D(\mathcal{S})$] of solutions exceed the scheduling window length τ. The heuristic measures the spread as it would be measured if τ equalled $D(\mathcal{S})$. If the traveling time of the final solution exceeds τ, it indicates that the heuristic is unable to find a feasible solution. In this case, the user must choose a less restrictive value for c, or reduce the number of required traversals. The rationale for not taking τ into account during the search is to allow the traveling time of the solution to gradually decrease as the heuristic reduces the cost objective function.

In practice, it was found that the heuristic makes many moves in which the cost objective function value remains unchanged towards the end of the execution run, and that the corresponding improvements in spread between these iterations are very slight. It is preferable to allow these moves to take place, because they frequently allow solutions with a better cost objective function to be uncovered later. Nevertheless, in the computer implementations described in this article, a maximum of 10 consecutive iterations that yielded no improvement in the cost objective were allowed, before the procedure was terminated.

It is worth noting that the local search heuristic may be used to approximate a so-called "efficient frontier," as done in investment portfolio optimization problems [34, 35]. Such a curve, representing the trade–off between the two objective functions may be a useful aid for practitioners.

The local search heuristic described above was applied to the small problem instance of Figure 2 as an example. The same problem parameters as those in Section 5 were used, and it was assumed that a user specified that $\hat{T}$ should equal 90% of the spread value of that obtained by the construction heuristic (i.e., $\hat{T} = 0.2788$). The best solution found by the improvement heuristic under these conditions was

$$\mathcal{S}^*_{\text{best}} = \langle (1,3), (3,2), (2,4), (1,5), (1,3), (2,3), (3,4), (4,2), (2,3), (3,1), (1,5), (5,3), (3,4) \rangle,$$

with a cost weight of 72 km and a spread objective function value of 0.2227. The traversal commencement times of each edge in this solution is given in Table 4, and shown graphically in Figure 8.

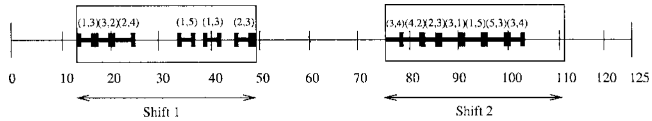

FIG. 8. Graphical representation of the traversal commencement times of the edges of $\mathcal{S}^*_{\text{best}}$, the final solution obtained by the local search heuristic for the SMTPP instance in Figure 2.

TABLE 4. Traversal commencement times of the edges of $\mathcal{S}^*_{\text{best}}$, the final solution obtained by the local search heuristic on the SMTPP instance in Figure 2.

Position in $\mathcal{S}^*_{\text{best}}$	Edge	Traversal commencement time
1	(1,3)	13.25
2	(3,2)	17.25
3	(2,4)	20.25
4	(1,5)	34.25
5	(1,3)	38.25
6	(2,3)	45.25
7	(3,4)	48.25
8	(4,2)	78.75
9	(2,3)	84.75
10	(3,1)	87.75
11	(1,5)	91.75
12	(5,3)	93.75
13	(3,4)	99.75

7. TEST RESULTS

The results of applying the construction and local search heuristics, for the SMTPP, on randomly generated test problem instances are presented in this section. Each of these graph instances belongs to one of following six different structures:

1. **Random trees.** Connected acyclic graphs. [Encoded, using the four–letter acronym "`TREE`"]
2. **Trees with multiple star–like structures.** Trees consisting of a number of connected focal vertices to which leaves are attached. [Encoded, using the acronym "`STAR`"]
3. **General connected graphs.** [Encoded, using the acronym "`GNRL`"]
4. **Grid–like graphs.** Graphs with a rectangular mesh–like structure. [Encoded, using the acronym "`GRID`"]
5. **Circulant–like graphs.** Graphs whose adjacency matrices are near circulant matrices. [Encoded, using the acronym "`CIRC`"]
6. **Near–complete graphs.** Graphs with a very high edge density. [Encoded, using the acronym "`COMP`"]

Graphs of different sizes (numbers of edges) were considered within each of the above structure classes, according to the following categories. The following categories of graphs were used.

1. **Small graphs.** Graphs of size at least 30 and at most 100 [Encoded, using the letter "`S`"]
2. **Medium graphs.** Graphs of size at least 100 and at most 200 [Encoded, using the letter "`M`"]
3. **Large graphs.** Graphs of size at least 200 and at most 300 [Encoded, using the letter "`L`"]

The orders of the trees, grid–like, near–complete, and star–like graphs are determined by the structure of the graph, but the orders of the general connected graphs and the circulant–like graphs need to be specified. The order of each circulant–like graph was set to half its size, and the order of each general connected graph was taken between 15 and 50 (for small graphs), 50 and 100 (medium graphs), and 100 and 150 (large graphs).

Ten instances of triply weighted graphs within each size class and within each structure class were generated randomly, giving a total of 180 test graphs altogether. The graphs are coded as follows: the sixth instance of a medium grid–like graph generated is, for example, labeled "`M-GRID-6`." The weights of the graph edges in all of the above classes were generated by randomly placing vertices within the unit square and then assigning cost weights equal to 1000 times the Euclidean distances between the vertices. The time taken to traverse each edge was set equal to the cost weight of the edge. This is equivalent to assuming that the cost weights are expressed as distances (in km), and the vehicle for which a route is sought travels at a constant speed of 60 km/h. The frequency weight of an edge was set equal to 0 according to a 50% probability, otherwise its value was chosen uniformly between 1 and 4. The scheduling window length, τ, for each instance was set equal to twice the $\mathcal{C}(\mathcal{S})$ value obtained by applying the construction heuristic, and $\kappa = 10$ throughout. The value of $\hat{\mathcal{T}}$ used for each instance was set equal to the spread value of the solution obtained by applying the construction heuristic to that instance.

Each of these graph instances may be found on the internet [30, 39]. The results obtained by applying the local search procedure of section 6 are given in Tables 5–7. The columns labeled "LB" contain lower bound values for each of the instances, calculated by determining a minimum weight maximum cardinality matching on the odd–degree vertices of the complete subgraph induced by the required edges [where $f(i,j) - 1$ additional required edges are created between vertices i and j for each required edge v_iv_j for which $f(i,j) > 1$]. The edge weights of an edge v_iv_j of the complete graph on which the matching above is calculated equals $d(i,j)$, the shortest distance from vertex i to j in $\mathcal{G}$. The execution times, listed in the columns labeled "Time," are expressed in seconds, and measure the total execution time of the heuristics (including preprocessing). All results were obtained using a Pentium IV (2.8 GHz) personal computer with 512 MB of RAM.

The difference between the $\mathcal{C}(\mathcal{S})$ and "LB," shown in Tables 5–7, tends to be large. The lower bounding procedure is not expected to yield good lower bounds, in general, and therefore, no judgements on the performance of the heuristics is made. However, in the case of near–complete graphs, the average percentage gap over the lower bound values, for the solutions obtained by the improvement procedure, equal 8, 3, and 2% for the small, medium, and large graphs, respectively. In these cases the lower bounding procedure, and the construction heuristic, yield good results because the subgraphs on which the matching phases of these procedures are determined tend to be connected.

The improvement heuristic yields an average improvement of 13% on the solution generated by the construction heuristic, on the problem instances considered. The improvement quality attained seems to be roughly uniform across the graph classes, with the exception of the near–complete graphs, where average improvements of 9, 5, and 3% are obtained for the small, medium, and large graphs, respectively. These inferior improvement qualities reflect the fact that the construction heuristic already obtains solutions that are closer to the optimal values, in the case of near–complete graphs. The average computational time expended equals approximately 30 minutes for the instances of the large data set, 6.5 minutes for the instances of the medium data set, and 27 seconds for the instances of the small data set.

8. CONCLUSION

In this article heuristics are introduced to solve the newly defined problem of finding an efficient closed route through a weighted graph, traversing each edge a pre–specified number of times, with the additional constraint that consecutive traversals of the same edge should be as evenly spread through the route as possible. A mathematical program is introduced, using a simplistic definition of this spread requirement. However, this simplistic view of spread is considered to be inadequate in many practical cases, and hence, a more focussed version of the problem (referred to as the SMTPP in this article), relevant specifically to vehicle routing problems, is defined. A graph theoretic solution construction heuristic is introduced for the SMTPP, with a computational complexity of $O(p^3)$, where p denotes the order of the input graph. A local search improvement heuristic is introduced for the SMTPP, which operates by attempting to improve a solution generated by the construction heuristic. The local search heuristic has a computational complexity of $O(q^3)$ per iteration, where q denotes the total number of required traversals in the problem. The heuristics were tested on random test instances, generated according to different classes of problem graph structure. The local search heuristic improves on the solution generated by the construction heuristic by an average of 13% for the test problems considered. The test data are available on the internet as benchmark problems for further work on the SMTPP.

Acknowledgments

The authors are endebted to Werner Gründlingh for producing the graphics in this paper. Work towards this paper was supported by the South African National Research Foundation under grant number GUN 2053755 and Research Sub–Committee B at the University of Stellenbosch. Any opinions, findings, and conclusions or recommendations

TABLE 5. Test results obtained for the small SMTPP benchmark data set.

Instance	Size	$\|\mathcal{S}\|$	τ	LB	Constr. Heur. $\mathcal{C}(\mathcal{S})$	Constr. Heur. $\mathcal{T}$	Impr. Heur. $\mathcal{C}(\mathcal{S})$	Impr. Heur. $\mathcal{T}$	Time (s)
S-COMP-1	51	45	42,264	17,168	21,132	0.1775	19,443	0.1629	1
S-COMP-2	61	79	101,078	44,108	50,539	0.2235	45,620	0.1650	12
S-COMP-3	40	46	58,802	23,862	29,401	0.2531	25,501	0.1996	1
S-COMP-4	40	56	63,894	26,116	31,947	0.1509	30,089	0.1375	2
S-COMP-5	88	111	121,906	54,377	60,953	0.2119	55,972	0.1956	68
S-COMP-6	53	75	76,066	32,019	38,033	0.1948	35,060	0.1733	10
S-COMP-7	98	124	148,352	66,717	74,176	0.1646	70,346	0.1478	84
S-COMP-8	89	124	178,970	80,325	89,485	0.2175	84,289	0.1573	76
S-COMP-9	70	73	92,014	35,885	46,007	0.2255	40,568	0.1932	10
S-COMP-10	76	74	91,542	38,851	45,771	0.2204	40,320	0.1764	9
S-CIRC-1	34	46	80,832	26,311	40,416	0.2265	31,740	0.2251	2
S-CIRC-2	62	108	180,926	61,063	90,463	0.2106	85,091	0.2045	16
S-CIRC-3	20	18	28,950	9,711	14,475	0.2328	12,496	0.1764	0
S-CIRC-4	82	122	289,134	76,028	14,4567	0.2610	114,668	0.2518	86
S-CIRC-5	76	100	191,886	46,742	95,943	0.1941	83,786	0.1858	32
S-CIRC-6	57	63	141,334	45,638	70,667	0.2510	62,368	0.2426	4
S-CIRC-7	64	85	154,944	44,824	77,472	0.2367	61,478	0.2208	16
S-CIRC-8	82	94	184,016	54,059	92,008	0.2142	80,401	0.1851	20
S-CIRC-9	77	106	206,392	62,168	103,196	0.1775	91,451	0.1754	29
S-CIRC-10	39	53	106,718	31,031	53,359	0.2392	41,692	0.2255	3
S-GNRL-1	40	59	189,816	31,918	94,908	0.1770	83,107	0.1752	3
S-GNRL-2	50	75	99,552	33,205	49,776	0.1681	44,523	0.1618	12
S-GNRL-3	91	113	160,242	57,995	80,121	0.2128	70,606	0.2003	44
S-GNRL-4	63	76	109,696	44,493	54,848	0.2449	48,482	0.2124	11
S-GNRL-5	83	122	203,052	73,293	101,526	0.2110	87,776	0.1943	63
S-GNRL-6	77	95	139,940	54,253	69,970	0.2080	59,550	0.1793	32
S-GNRL-7	50	51	114,248	37,074	57,124	0.2165	49,321	0.2058	2
S-GNRL-8	92	126	180,948	72,859	90,474	0.2179	81,277	0.1869	87
S-GNRL-9	70	89	122,390	49,118	61,195	0.2369	55,194	0.2049	30
S-GNRL-10	99	130	212,714	70,937	106,357	0.2470	85,506	0.2186	147
S-GRID-1	91	85	292,126	44,831	146,063	0.2171	116,642	0.2159	16
S-GRID-2	80	97	235,932	55,495	117,966	0.2026	100,195	0.1967	19
S-GRID-3	100	165	342,510	91,238	171,255	0.2121	149,226	0.2018	108
S-GRID-4	89	97	352,948	59,667	176,474	0.1830	142,551	0.1781	34
S-GRID-5	44	44	152,514	26,686	76,257	0.2012	65,648	0.1367	1
S-GRID-6	83	122	286,642	80,943	143,321	0.1962	132,394	0.1950	33
S-GRID-7	83	108	238,080	58,893	119,040	0.1861	96,784	0.1823	45
S-GRID-8	77	83	207,682	47,749	103,841	0.2408	89,775	0.2404	10
S-GRID-9	84	123	277,914	83,063	138,957	0.2115	115,810	0.2003	55
S-GRID-10	82	116	264,610	72,078	132,305	0.1994	116,662	0.1951	40
S-STAR-1	91	128	416,556	69,924	208,278	0.1875	171,286	0.1638	46
S-STAR-2	86	72	206,428	39,454	103,214	0.2040	84,018	0.1764	9
S-STAR-3	47	31	106,292	25,348	53,146	0.2485	44,502	0.1778	0
S-STAR-4	68	105	294,148	59,932	147,074	0.1926	122,924	0.1341	27
S-STAR-5	53	72	250,708	51,650	125,354	0.1769	108,358	0.1656	9
S-STAR-6	68	66	190,644	41,094	95,322	0.2307	86,946	0.1417	4
S-STAR-7	67	76	219,148	45,752	109,574	0.2414	90,726	0.2043	10
S-STAR-8	46	66	134,068	29,942	67,034	0.2361	60,894	0.2072	3
S-STAR-9	48	63	200,924	41,008	100,462	0.1938	87,664	0.1613	4
S-STAR-10	37	46	101,464	24,600	50,732	0.2080	47,242	0.1689	2
S-TREE-1	57	71	243,300	38,738	121,650	0.1610	93,084	0.1265	9
S-TREE-2	37	46	154,964	24,466	77,482	0.2836	61,268	0.2608	1
S-TREE-3	72	107	502,016	69,572	251,008	0.2178	187,992	0.1947	45
S-TREE-4	93	109	396,940	64,198	198,470	0.1845	169,880	0.1711	38
S-TREE-5	42	42	217,520	24,900	108,760	0.2585	75,956	0.2285	1
S-TREE-6	97	115	577,176	87,782	288,588	0.1852	250,290	0.1640	46
S-TREE-7	31	56	231,964	41,388	115,982	0.2436	85,468	0.2275	5
S-TREE-8	59	98	328,900	51,922	164,450	0.2029	138,948	0.1686	31
S-TREE-9	90	120	495,052	80,014	247,526	0.1658	212,372	0.1482	54
S-TREE-10	40	71	221,396	42,062	110,698	0.2091	88,356	0.1812	7

The execution time expended on each instance is listed in the column labeled "Time." A lower bound value for each instance is shown in the column labeled "LB."

TABLE 6. Test results obtained for the medium SMTPP benchmark data set.

Instance	Size	$\|\mathcal{S}\|$	τ	LB	Constr. Heur. $\mathcal{C}(\mathcal{S})$	Constr. Heur. $\mathcal{T}$	Impr. Heur. $\mathcal{C}(\mathcal{S})$	Impr. Heur. $\mathcal{T}$	Time (s)
M-COMP-1	130	153	196,310	88,638	98,155	0.2081	92,173	0.1663	261
M-COMP-2	181	255	259,482	123,001	129,741	0.1934	126,310	0.1731	1435
M-COMP-3	191	230	230,612	105,128	115,306	0.1912	109,680	0.1627	677
M-COMP-4	117	132	126,638	56,843	63,319	0.2270	58,139	0.1699	138
M-COMP-5	178	194	236,786	110,182	118,393	0.2131	114,233	0.1573	341
M-COMP-6	115	111	132,338	57,461	66,169	0.2532	59,455	0.1865	120
M-COMP-7	187	246	270,522	126,776	135,261	0.1902	131,344	0.1694	436
M-COMP-8	137	181	222,852	102,984	111,426	0.1811	106,217	0.1483	175
M-COMP-9	182	217	223,768	105,234	111,884	0.2346	107,892	0.1852	278
M-COMP-10	140	180	203,394	92,452	101,697	0.2066	96,707	0.1672	134
M-CIRC-1	197	223	482,052	131,063	241,026	0.2274	199,468	0.2251	930
M-CIRC-2	180	252	505,764	154,353	252,882	0.2272	221,310	0.2218	643
M-CIRC-3	187	239	435,652	128,355	217,826	0.2179	189,576	0.2119	589
M-CIRC-4	178	249	531,714	159,144	265,857	0.2395	231,363	0.2341	650
M-CIRC-5	190	252	498,186	142,349	249,093	0.1966	230,678	0.1948	451
M-CIRC-6	174	234	477,052	138,539	238,526	0.2044	211,409	0.2022	538
M-CIRC-7	175	240	466,700	138,184	233,350	0.1938	210,594	0.1916	578
M-CIRC-8	118	157	324,196	94,229	162,098	0.2409	127,177	0.2390	229
M-CIRC-9	196	220	453,528	124,505	226,764	0.2086	191,656	0.2060	466
M-CIRC-10	166	227	448,062	136,976	224,031	0.2205	197,510	0.2197	547
M-GNRL-1	158	174	281,936	101,431	140,968	0.2155	122,036	0.1867	332
M-GNRL-2	171	213	277,538	108,573	138,769	0.2013	125,415	0.1857	450
M-GNRL-3	152	204	394,472	123,192	197,236	0.2041	174,299	0.1945	398
M-GNRL-4	126	158	284,466	91,815	142,233	0.2299	123,127	0.2230	155
M-GNRL-5	169	201	431,502	110,430	215,751	0.2044	174,337	0.1987	480
M-GNRL-6	152	187	273,668	99,480	136,834	0.1668	130,117	0.1557	147
M-GNRL-7	197	246	324,340	128,952	162,170	0.2514	145,755	0.2275	729
M-GNRL-8	151	195	380,980	106,399	190,490	0.2279	154,516	0.2205	380
M-GNRL-9	151	183	337,204	106,357	168,602	0.2260	148,887	0.2237	194
M-GNRL-10	107	120	263,916	74,529	131,958	0.1793	118,496	0.1767	46
M-GRID-1	156	193	500,634	122,076	250,317	0.2455	204,180	0.2431	297
M-GRID-2	175	221	504,452	130,901	252,226	0.2203	206,493	0.2163	602
M-GRID-3	131	154	414,366	89,613	207,183	0.2244	165,169	0.2213	167
M-GRID-4	122	148	402,256	98,555	201,128	0.1860	178,992	0.1838	92
M-GRID-5	138	136	333,836	79,572	166,918	0.2349	126,652	0.2306	113
M-GRID-6	170	225	473,118	128,040	236,559	0.2336	184,335	0.2296	737
M-GRID-7	146	178	489,368	109,255	244,684	0.2075	223,619	0.2071	173
M-GRID-8	122	154	415,160	88,869	207,580	0.2013	161,215	0.1971	165
M-GRID-9	167	182	499,068	121,283	249,534	0.2022	218,407	0.2005	219
M-GRID-10	107	141	310,912	87,421	155,456	0.1999	131,879	0.1906	115
M-STAR-1	198	244	815,940	141,924	407,970	0.2241	357,160	0.2202	500
M-STAR-2	158	224	714,556	138,596	357,278	0.1826	301,140	0.1701	332
M-STAR-3	196	267	978,836	167,292	489,418	0.1977	395,170	0.1805	609
M-STAR-4	108	150	553,696	103,072	276,848	0.1803	253,828	0.1716	60
M-STAR-5	107	161	531,188	103,192	265,594	0.1689	228,778	0.1605	151
M-STAR-6	108	140	458,548	87,618	229,274	0.1833	202,556	0.1706	87
M-STAR-7	154	207	763,932	133,490	381,966	0.2260	306,994	0.2004	251
M-STAR-8	132	147	554,692	92,076	277,346	0.2259	225,562	0.1969	114
M-STAR-9	105	142	360,084	77,974	180,042	0.1714	171,520	0.1613	40
M-STAR-10	102	121	403,860	65,550	201,930	0.2217	157,536	0.2004	46
M-TREE-1	173	201	805,400	123,018	402,700	0.2294	311,850	0.2091	317
M-TREE-2	107	118	468,524	73,838	234,262	0.2299	197,118	0.2144	57
M-TREE-3	178	209	892,388	139,030	446,194	0.2278	385,110	0.2231	223
M-TREE-4	180	210	952,036	123,862	476,018	0.2279	408,000	0.2186	230
M-TREE-5	187	235	1,058,596	143,706	529,298	0.2268	453,388	0.2231	559
M-TREE-6	196	247	1,081,432	157,152	540,716	0.1918	484,816	0.1879	507
M-TREE-7	113	140	653,156	107,346	326,578	0.2025	271,046	0.1945	100
M-TREE-8	161	225	1,039,372	151,554	519,686	0.2101	409,562	0.1995	415
M-TREE-9	176	246	1,208,984	157,506	604,492	0.2341	492,638	0.2257	495
M-TREE-10	145	163	708,016	118,394	354,008	0.1933	297,484	0.1729	141

The computation time expended on each instance is listed in the column labeled "Time." A lower bound value for each instance is shown in the column labeled "LB."

TABLE 7. Test results obtained for the large SMTPP benchmark data set.

Instance	Size	$\|\mathcal{S}\|$	τ	LB	Constr. Heur. $\mathcal{C}(\mathcal{S})$	Constr. Heur. $\mathcal{T}$	Impr. Heur. $\mathcal{C}(\mathcal{S})$	Impr. Heur. $\mathcal{T}$	Time (s)
L-COMP-1	251	299	317,956	150,275	158,978	0.2239	152,192	0.1758	1206
L-COMP-2	221	268	301,498	142,300	150,749	0.2003	145,726	0.1580	653
L-COMP-3	227	290	340,766	162,100	170,383	0.2086	166,799	0.1802	570
L-COMP-4	241	300	309,120	144,193	154,560	0.2045	148,348	0.1813	1000
L-COMP-5	205	277	309,698	146,008	154,849	0.2235	149,306	0.1707	771
L-COMP-6	250	319	364,884	174,522	182,442	0.2167	176,054	0.1490	1283
L-COMP-7	291	379	408,338	196,309	204,169	0.2262	200,131	0.1878	1656
L-COMP-8	239	316	350,690	164,408	175,345	0.2035	169,185	0.1722	1337
L-COMP-9	232	318	333,174	156,967	166,587	0.1946	160,824	0.1564	2691
L-COMP-10	230	246	281,112	130,888	140,556	0.2276	134,048	0.1854	1259
L-CIRC-1	247	309	599,236	173,252	299,618	0.2302	266,659	0.2289	1078
L-CIRC-2	268	350	647,530	189,790	323,765	0.2554	277,386	0.2532	1808
L-CIRC-3	261	346	631,294	179,996	315,647	0.2278	277,602	0.2268	2610
L-CIRC-4	210	274	510,898	149,996	255,449	0.2279	222,980	0.2245	783
L-CIRC-5	267	319	689,280	142,229	244,640	0.2190	214,090	0.2176	877
L-CIRC-6	218	283	544,226	165,668	272,113	0.2251	230,645	0.2222	1229
L-CIRC-7	245	303	582,098	170,198	291,049	0.2074	256,801	0.2062	1370
L-CIRC-8	274	334	640,668	181,974	320,334	0.2398	271,759	0.2382	2011
L-CIRC-9	267	344	629,696	182,056	314,848	0.2233	282,322	0.2228	1548
L-CIRC-10	276	380	740,456	219,582	370,228	0.2000	345,257	0.1975	2561
L-GNRL-1	215	266	533,522	152,411	266,761	0.2404	222,167	0.2351	955
L-GNRL-2	238	322	643,992	187,908	321,996	0.2257	279,705	0.2195	1522
L-GNRL-3	238	274	629,244	170,195	314,622	0.1911	270,550	0.1868	1142
L-GNRL-4	231	344	589,774	174,184	294,887	0.2154	261,204	0.2104	2379
L-GNRL-5	221	312	612,060	176,439	306,030	0.1984	266,473	0.1935	1982
L-GNRL-6	246	263	489,280	142,229	244,640	0.2190	214,090	0.2176	877
L-GNRL-7	215	314	682,760	173,846	341,380	0.2199	298,953	0.2166	1562
L-GNRL-8	219	294	646,776	170,193	323,388	0.2342	278,936	0.2310	1745
L-GNRL-9	292	348	673,412	214,975	336,706	0.2054	298,080	0.2033	3138
L-GNRL-10	295	358	635,404	192,342	317,702	0.2140	285,150	0.2115	2839
L-GRID-1	290	369	860,292	222,351	430,146	0.2077	397,531	0.2061	1752
L-GRID-2	221	267	748,294	166,040	374,147	0.2104	339,438	0.2052	468
L-GRID-3	298	352	791,266	201,581	395,633	0.2028	348,937	0.2021	2100
L-GRID-4	294	346	714,494	176,628	357,247	0.2384	308,964	0.2370	1662
L-GRID-5	274	296	727,826	166,472	363,913	0.2498	313,546	0.2494	918
L-GRID-6	240	304	737,244	180,482	368,622	0.2141	307,636	0.2114	1577
L-GRID-7	231	291	718,814	166,957	359,407	0.2038	298,136	0.2026	1190
L-GRID-8	227	266	677,696	159,304	338,848	0.1953	293,593	0.1943	1015
L-GRID-9	295	369	830,388	222,091	415,194	0.2191	364,848	0.2164	2502
L-GRID-10	270	328	798,338	185,230	399,169	0.2032	358,701	0.2009	1601
L-STAR-1	261	330	991,236	175,792	495,618	0.2357	421,638	0.2297	2840
L-STAR-2	227	301	1,032,028	171,096	516,014	0.1952	452,620	0.1908	1730
L-STAR-3	224	285	909,504	174,920	454,752	0.2400	393,152	0.2319	1782
L-STAR-4	259	303	1,015,912	197,234	507,956	0.1973	447,106	0.1913	1904
L-STAR-5	285	326	1,076,640	184,430	538,320	0.2069	442,370	0.1880	2513
L-STAR-6	224	289	932,976	182,848	466,488	0.2137	406,718	0.1979	1263
L-STAR-7	269	353	1,087,500	217,462	543,750	0.2140	478,934	0.2115	2860
L-STAR-8	262	313	981,512	199,838	490,756	0.1694	459,566	0.1583	2089
L-STAR-9	226	317	1,136,208	200,636	568,104	0.2149	509,734	0.2067	1730
L-STAR-10	279	328	1,321,708	208,072	660,854	0.1989	595,182	0.1905	1964
L-TREE-1	275	331	1,773,488	230,454	886,744	0.2214	677,688	0.2137	4165
L-TREE-2	223	276	1,309,656	171,442	654,828	0.2047	568,314	0.1998	1239
L-TREE-3	244	295	1,331,460	191,154	665,730	0.2151	518,782	0.2029	2519
L-TREE-4	225	281	1,350,248	176,960	675,124	0.2174	572,128	0.2137	1407
L-TREE-5	285	344	1,696,228	222,454	848,114	0.1958	749,256	0.1912	4241
L-TREE-6	250	271	1,274,724	165,398	637,362	0.2144	587,284	0.2075	841
L-TREE-7	298	332	1,472,300	202,148	736,150	0.2353	564,632	0.2167	4434
L-TREE-8	235	302	1,421,340	180,168	710,670	0.2273	532,228	0.2152	4060
L-TREE-9	208	272	1,470,648	181,506	735,324	0.2092	648,244	0.2038	1565
L-TREE-10	207	225	980,432	138,004	490,216	0.2291	402,364	0.2065	896

The computation time expended on each instance is listed, in seconds, in the column labeled "Time." A lower bound value for each instance is shown in the column labeled "LB."

expressed in this article are those of the authors and do not necessarily reflect the views of the South African National Research Foundation.

REFERENCES

[1] R.D. Angel, W.L. Caulde, R. Noonan, and A. Whinston, Computer-assisted school bus scheduling, Manag Sci B18 (1972), 279–288.

[2] M.O. Ball, T.L. Magnanti, C.L. Monma, and G.L. Nemhauser (Editors), Network routing, North-Holland, Amsterdam, 1995.

[3] E.L. Beltrami and L.D. Bodin, Networks and vehicle routing for municipal waste collection, Networks 4 (1974), 65–94.

[4] B. Bennett and D. Gazis, School bus routing by computer, Transport Res 6 (1972), 317.

[5] L.D. Bodin and O. Berman, Routing and scheduling of school buses by computer, Transport Sci 13 (1979), 113–129.

[6] L.D. Bodin, G. Fagin, R. Welebny, and J. Greenberg, The design of a computerized sanitation vehicle routing and scheduling system for the town of Oyster Bay, New York, Comput Oper Res 16 (1989), 45–54.

[7] L.D. Bodin and S.J. Kursh, A detailed description of a computer system for the routing and scheduling of street sweepers, Comput Oper Res 6 (1979), 181–198.

[8] J. Braca, J. Bramel, B. Posner, and D. Simchi-Levi, A computerized approach to the New York City school bus routing project, Working paper, Columbia University, New York, 1993.

[9] G.A. Croes, A method for solving traveling salesmen problems, Oper Res 6 (1958), 791–812.

[10] J. Desrosiers, J.A. Ferland, J. Rousseau, G. Lapalme, and L. Chapleau, TRANSCOL: A multi-period school bus routing and scheduling system, TIMS Stud Manage Sci 22 (1986), 47–71.

[11] E.W. Dijkstra, A note on two problems in connection with graphs, Num Math 1 (1959), 267–271.

[12] M. Dror (Editor), Arc routing: Theory, solutions and applications, Kluwer Academic Publishers, Dordrecht, 2000.

[13] J. Edmonds and E.L. Johnson, Matching, Euler tours and the chinese postman problem, Math Program, 5 (1973), 88–124.

[14] R.W. Eglese, Routing winter gritting vehicles, Discrete App Math 48 (1994), 231–244.

[15] R.W. Eglese and H. Murdock, Routing road sweepers in a rural area, Oper Res Soc, 42 (1991), 281–288.

[16] H.A. Eiselt, M. Gendreau, and G. Laporte, Arc routing problems, part I: The Chinese postman problem, Oper Res 43 (1995), 231–242.

[17] H.A. Eiselt, M. Gendreau, and G. Laporte, Arc routing problems, part II: The rural postman problem, Oper Res 43 (1995), 399–414.

[18] M.M. Flood, The traveling–salesman problem, Oper Res 4 (1956), 61–75.

[19] R.W. Floyd, Algorithm 97: Shortest path, Commun ACM 5, (1962), 345.

[20] G.N. Frederickson, Approximation algorithms for some routing problems, J Assoc Comput Machinery 26 (1979), 538–554.

[21] L.F. Gelders and D.G. Cattrysse, Public waste collection: A case study, Belgian J Oper Res Stat Comput Sci 31 (1991), 3–15.

[22] G. Ghiani and G. Improta, Optimizing laser–plotter beam movement, Technical Report, Università di Napoli "Federico II," Napoli, Italy.

[23] M. Gondran and M. Minoux, Graphs and algorithms, Wiley–Interscience, 1984.

[24] M. Grötschel, M. Jünger, and G. Reinelt, Optimal control of plotting and drilling machines: A case study, Oper Res 35 (1991), 61–84.

[25] G.W. Groves, Scheduling evenly spaced routes in networks, PhD dissertation, University of Stellenbosch, South Africa, 2004.

[26] G.W. Groves, J. le Roux, and J.H. van Vuuren, Network routing and scheduling, Paper presented at the International Conference on Operations Research in Development, Kruger National Park, 2001.

[27] G.W. Groves and J.H. van Vuuren, Efficient heuristics for the rural postman problem, ORiON 21(1) (2005), 33–51.

[28] M. Guan, Graphic programming using odd and even cycles, Chin Math 1 (1962), 237–277.

[29] P. Lacomme, C. Prins, and W. Ramdane–Chérif, Fast algorithms for general arc routing problems, Paper presented at the Sixteenth Triennial Conference of the International Federation of Operations Research Societies, Edinburgh, 2002.

[30] J le Roux, Scheduled multiple traversal postman problem (SMTPP), [online], [cited 2004, April 29], Available from: `http://jleroux.navors.net/repositories/smtpp.htm`.

[31] J.K. Lenstra and A.H.G. Rinnooy Kan, On general routing problems, Networks 6 (1976), 273–280.

[32] L. Levy and L.D. Bodin, "Scheduling the postal carriers for the United States Postal Service: An application of arc partioning and routing." In Vehicle routing: Methods and studies, B.L. Golden and A.A. Assad (Editors), North-Holland, Amsterdam, 1988, 359–394.

[33] K. Mehlhorn and S. Näher, LEDA: A platform for combinatorial and geometric computing, Cambridge University Press, Cambridge, 1999.

[34] H.M. Markowitz, Portfolio selection, J Finance 7 (1952), 77—91.

[35] H.M. Markowitz, Portfolio selection — Efficient diversification of investments, John Wiley and Sons Inc., New York, (1959).

[36] W. Ramdane–Chérif, Problèmes de tournées sur arcs, PhD Dissertation, University of Troyes, France, 2002 (in French).

[37] S. Roy and J. Rousseau, The capacitated Canadian postman problem, INFOR 27 (1989), 58–73.

[38] H. Stern and M. Dror, Routing electric meter readers, Comput Oper Res 6 (1979), 209–223.

[39] J.H. van Vuuren Scheduled multiple traversal postman problem (SMTPP), [online], [cited 2004, April 29], Available from: `http://dip.sun.ac.za/~vuuren/repositories/smtpp.htm`.

[40] J. Wunderlich, M. Collette, L. Levy, and L.D. Bodin, Scheduling meter readers for southern California gas company, Interfaces 22 (1992), 22–30.