



Network service scheduling and routing

G. Groves^a, J. le Roux^b and J. H. van Vuuren^c

^a*Department of Industrial Engineering, Stellenbosch University, Private Bag X1, Matieland, 7602, Republic of South Africa,*

^b*Department of Quantitative Management, University of South Africa, PO Box 392, Pretoria, 0003, Republic of South Africa,*

^c*Department of Applied Mathematics, Stellenbosch University, Private Bag X1, Matieland, 7602, Republic of South Africa
E-mail: gw_groves@mweb.co.za [Groves]; lrouxj@unisa.ac.za [le Roux]; vuuren@sun.ac.za [van Vuuren]*

Received December 2001; received in revised form June 2003; accepted March 2004

Abstract

Real-life vehicle routing problems generally have both routing and scheduling aspects to consider. Although this fact is well acknowledged, few heuristic methods exist that address both these complicated aspects simultaneously. We present a graph theoretic heuristic to determine an efficient service route for a single service vehicle through a transportation network that requires a subset of its edges to be serviced, each a specified (potentially different) number of times. The times at which each of these edges are to be serviced should additionally be as evenly spaced over the scheduling time window as possible, thus introducing a scheduling consideration to the problem. Our heuristic is based on the tabu search method, used in conjunction with various well-known graph theoretic algorithms, such as those of Floyd (for determining shortest routes) and Frederickson (for solving the rural postman problem). This heuristic forms the backbone of a decision support system that prompts the user for certain parameters from the physical situation (such as the service frequencies and travel times for each network link as well as bounds in terms of acceptability of results) after which a service routing schedule is suggested as output. The decision support system is applied to a special case study, where a service routing schedule is sought for the South African national railway system by SPOORNET (the semi-privatised South African national railways authority and service provider) as part of their rationalisation effort, in order to remain a lucrative company.

Keywords: Chinese [rural, capacitated] postman problem; tabu search; vehicle routing and scheduling

1. Introduction

Consider a transportation system comprising a number of destinations, linked by a network of connecting travel routes from a number of different classes or types, such as a railway system (consisting of metro rails, freight links, long distance lines, etc.) or a road network (consisting of suburban roads, dirt roads, highways, etc.). Maintenance of such a network typically



Fig. 1. The IM2000 testing vehicle used by SPOORNET to service rail tracks.

requires routine servicing of each of the links. However, the service frequencies 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 in some applications, or may be service company policy in other applications. Suppose a single service vehicle is used to service the entire transportation network and that the service vehicle is not required to return to a depot periodically. Then the problem faced by service inspectors is therefore to determine a closed service route for the servicing vehicle through the network that services some subset of the network links, visiting each link in the subset at least a pre-specified (potentially different) number of times, while simultaneously minimising the total service cost (which typically depends on the total travel distance and/or time through the network, when actually servicing or when free running between required link services) and achieving an acceptable amount of temporal spread with respect to consecutive services of the same link, for all network links, over the scheduling time window.

This routing problem came to the attention of the authors when SPOORNET (the semi-privatised South African national railways authority and service provider) approached them in search of an algorithm or set of rules by which to service the South African national rail grid in an efficient manner. The railways authority leases only a single railways servicing vehicle, called the IM2000, as shown in Fig. 1. The IM2000 travels at speeds between 40 and 100 km/h while performing a number of sophisticated rail track measurements, including angle measurements of tracks and sleepers in their stone beds and a variety of fatigue measurements. The South African rail grid is shown in Fig. 2, and consists of four types of tracks: main lines, secondary lines, branch lines and coal/ore lines. These track types are classified according to a complicated set of rules involving maximum axle loads, and the speed limits of trains on each of these track types differ; hence these different track types have different annual service frequency requirements. SPOORNET required a routing and annual scheduling specification for the IM2000 over the entire rail grid, which is:

- efficient in terms of the total distance travelled (due to the high operating cost of the vehicle), and

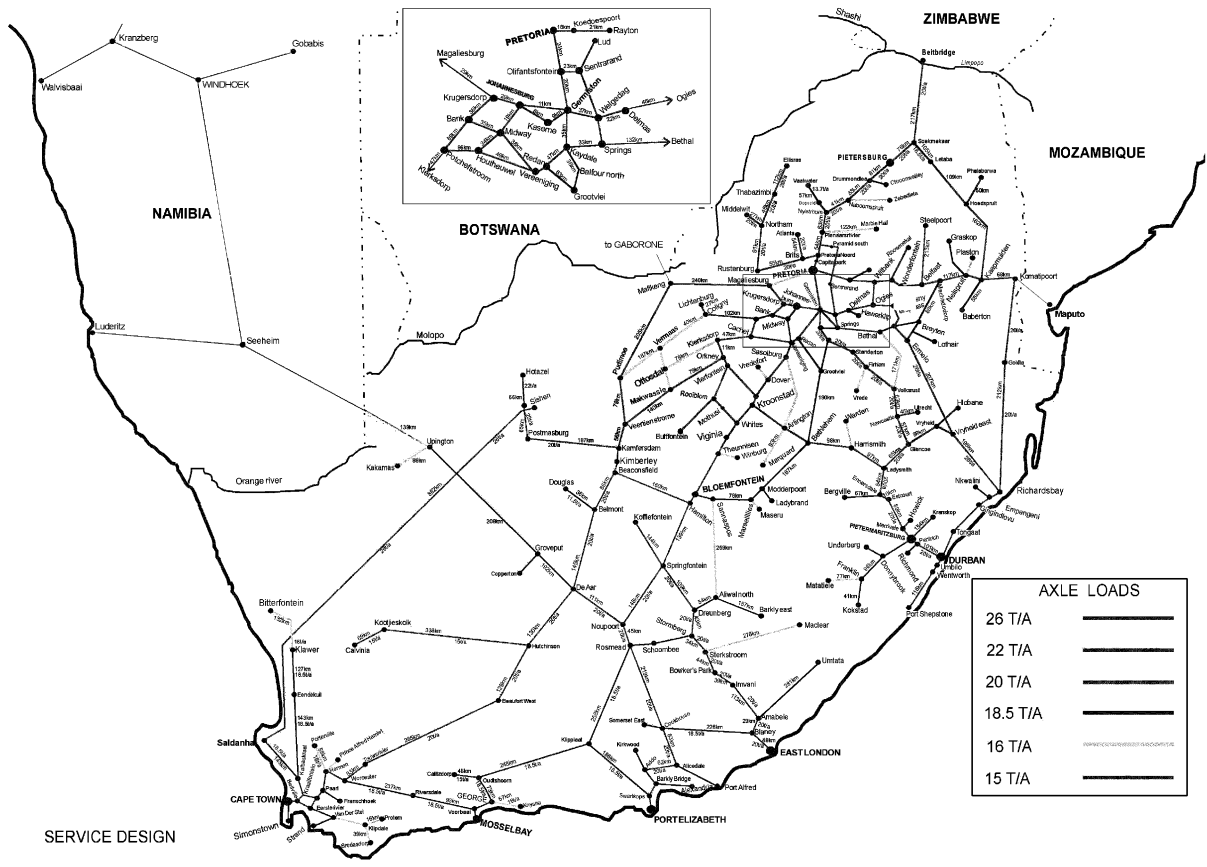


Fig. 2. The South African national rail grid, containing 195 main stations and 228 rail track segments.

- efficient in terms of the temporal spread of consecutive services of the same track segment, for all segments.

Finally, it should be noted that the IM2000 is the only rail track service vehicle employed by SPOORNET; hence its jurisdiction is the entire national rail grid, and it is not required to return to a depot after each working shift: it functions according to a predetermined, long-term schedule and is able to remain idle on a side line anywhere in the country at any time.

The problem described above seems to be a new contribution to the considerable body of existing literature on vehicle routing problems. However, the routing aspect of this problem strongly resembles that in variations of the well-known Chinese postman problem (see, for example, Dror and Stern, 1987; Eiselt et al. 1995a; Gendreau et al., 1997; Ghiani and Importa, 2000; Lin and Zhao, 1988; Nobert and Picard, 1996; Yan and Thompson, 1998), the rural postman problem (see, for example, Eiselt et al., 1995b; Hertz et al., 1999; Pearn and Wu, 1995; Stern and Dror, 1979) and the capacitated postman problem (see, for example, Amberg et al., 2000; Golden et al., 1983; Malandraki and Daskin, 1993; Pearn, 1989, 1991; Roy and Rousseau, 1989; Wu and Manber, 1992). A feature of our problem, which is not present in the above-

mentioned well-known routing problems, is the requirement that consecutive services of each of the connecting links within the network should be spaced as evenly over the scheduling time window as possible, thus introducing a scheduling consideration to the problem.

Moreover, the objectives of the routing and scheduling aspects of the SPOORNET problem are conflicting in the sense that one might intuitively attempt to service a link in the network the required number of times while one finds oneself in the vicinity of the link, and then proceed to other links in order to minimise the routing cost objective (which may be measured in time or distance units), but this would result in a badly spaced service time schedule. Conversely, a temporal spread between consecutive services of a specific link would typically increase the total routing cost of the service schedule, due to an increased amount of free running in order to reach links that require servicing.

The above-mentioned scheduling and routing problem is formulated more precisely (using graph theoretic notation) in Section 2. The basic approach towards the solution of this problem will be to develop a heuristic (tabu) search procedure for determining routing schedules. This will be explained in detail (and will be illustrated by means of example) in Section 4. However, the search procedure requires a starting solution, which it then attempts to improve iteratively. We describe, in Section 3, how a good initial solution may be generated. In Section 5, we report on some preliminary test results when applying our method to randomly generated graphs, after which the architecture of a decision support system, using our tabu search heuristic, to generate good service schedules and routings as a decision aid for network service/inspection managers is discussed in Section 6. Finally, this decision support system is applied in Section 7 to the realistic special case study where a railway service routing schedule is sought for the South African railway system by SPOORNET. We conclude the paper in Sections 8 and 9 by commenting on the efficiency of our solution procedure and by reflecting upon possible improvements.

2. Model formulation

We model the transportation network that has to be serviced by a triply weighted graph \mathcal{G} of order p , with vertex set $V(\mathcal{G}) = \{v_1, \dots, v_p\}$ and edge set $E(\mathcal{G})$. Denote the weights associated with the edge $v_i v_j \in E(\mathcal{G})$ by the triple $(c(i, j), d(i, j), f(i, j))$, where $c(i, j) \geq 0$ [$d(i, j) \geq c(i, j)$] represents the cost (time) to free run [service the edge, respectively] between v_i and v_j , and where $f(i, j) \in \mathbb{N}$ represents the required service frequency (number of services) of the edge $v_i v_j$ during the time window for which a service schedule is sought. Let v_1 denote the domicile vertex from which the service vehicle is required to start its tour and to which it must return at the end of the tour. We encode a service routing as a sequence $\mathcal{S} = \langle (v_{s_1}, v_{t_1}), (v_{s_2}, v_{t_2}), \dots, (v_{s_n}, v_{t_n}) \rangle$ of the order in which the edges are to be serviced. Note that only edges actually serviced are present in the coding, and that free-running edges are not accommodated. The free-running paths followed between two edge services in such a sequence are assumed to be shortest travelling time paths and are omitted for the sake of convenience. We shall attempt to construct a service sequence that minimises the routing objective

$$\mathcal{R}(\mathcal{S}) = \sum_{i=1}^n d(s_i, t_i) + e(1, s_1) + \sum_{j=1}^{n-1} e(t_j, s_{j+1}) + e(t_n, 1), \quad (2.1)$$

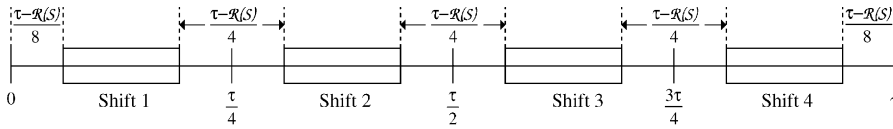


Fig. 3. A schedule $\mathcal{W}(\mathcal{S}, \tau, 4)$ corresponding to the service sequence \mathcal{S} , of length τ and consisting of four shifts, of combined length $\mathcal{R}(\mathcal{S})$.

where $e(k, l)$ denotes the free-running cost (time) between any two vertices $v_k, v_l \in V(\mathcal{G})$; hence $e(k, l)$ is calculated by adding the weights $c(i, j)$ in the problem graph \mathcal{G} that correspond to a shortest path from v_k to v_l in \mathcal{G} .

Note that, for a feasible service schedule, the schedule window will always be longer than or equal to the total time spent travelling. It is with respect to this schedule window, and not the travelling time that consecutive services of the same edge need to exhibit temporal spread. The difference between the schedule window and the travelling time needs to be absorbed somewhere in the schedule. It would, for example, be possible to absorb this time in even amounts after each time an edge is serviced. This would spread the services out throughout the service window, but it would mean that the service vehicle must stop and wait every time it finishes servicing an edge. For real-life applications it would be more desirable to work in shifts, servicing several edges consecutively. Real-life schedules would typically also have other complicating constraints on the structure of the schedule. An example of such a constraint might be periods during which some edges may not be serviced. For the purposes of our study, however, we assume that the schedule is divided into shifts of equal or more or less equal length and that the slack time is absorbed in even amounts before and after each shift. Figure 3 depicts a schedule of length τ consisting of four shifts.

A service sequence \mathcal{S} is converted to a schedule $\mathcal{W}(\mathcal{S}, \tau, \kappa)$ of length τ and consisting of κ shifts according to the following rules:

- Using the slack time $\tau - \mathcal{R}(\mathcal{S})$, identify the planned starting and ending times of each shift, so as to absorb slack time in equal amounts between shifts (see Fig. 3 for an example).
- Divide the service tour into κ segments of equal length (the tour may be terminated anywhere along an edge or on a vertex).
- Starting from the beginning of the tour, copy each segment obtained in the previous step to consecutive shifts.
- For each shift (starting from the first shift, and proceeding consecutively to following shifts) do the following: If the route segment in that shift ends while an edge is being serviced, attempt to extend the shift by extending both the starting and ending times of the shift in equal amounts so as to absorb slack time. If this is not possible, the edge is allowed to overlap shifts.

The above conversion rules remain fixed throughout the implementation of our solution procedure. For a given schedule $\mathcal{W}(\mathcal{S}, \tau, \kappa)$, we shall adopt the temporal spread objective

$$\mathcal{T}(\mathcal{W}(\mathcal{S}, \tau, \kappa)) = \sum_{\substack{v_i v_j \in E(\mathcal{G}) \\ f(i,j) > 1}} \frac{\sum_{u=2}^{f(i,j)} \left| \frac{s^u(i,j) - s^{u-1}(i,j)}{\tau / f(i,j)} - 1 \right|}{f(i,j) - 1}, \tag{2.2}$$

where $0 \leq s^u(i, j) \leq \tau$ denotes the time instant at which the u th service of edge $v_i v_j \in E(\mathcal{G})$ is commenced. Since edges that have to be serviced only once may be serviced at any time during the service schedule, without influencing the time spread of the service route, edges with frequency $f(i, j) = 1$ are omitted from the temporal spread objective (2.2). The objective $\mathcal{F}(\mathcal{W}(\mathcal{S}, \tau, \kappa))$ takes the value 0 if and only if $s^u(i, j) - s^{u-1}(i, j) = \tau/f(i, j)$ for all edges $v_i v_j \in E(\mathcal{G})$, i.e. when the time spread between consecutive services of all edges are ideal (equal and maximal). Otherwise the value $\mathcal{F}(\mathcal{W}(\mathcal{S}, \tau, \kappa))$ is positive.

We illustrate the concepts, as well as routing and scheduling efficiencies, of a service sequence \mathcal{S} and its corresponding schedule $\mathcal{W}(\mathcal{S}, \kappa, \tau)$ by means of a small example.

Example 2.1. Consider the small network in Fig. 4. Suppose we require a service route starting from and returning to vertex 1 with a corresponding schedule $\mathcal{W}(\mathcal{S}^*, 130, 2)$ of total length 130 time units and that we wish to employ two work shifts during the schedule. Therefore the domicile vertex is v_1 . The coded service sequence

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

represents the (by no means optimal) complete network traversal **(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 actual services. The routing objective for this service sequence is $\mathcal{R}(\mathcal{S}^*) = 98$ cost units. According to the above-mentioned rules, \mathcal{S}^* may be converted into the schedule $\mathcal{W}(\mathcal{S}^*, 130, 2)$ depicted in Fig. 5, for which the scheduling objective is $\mathcal{F}(\mathcal{W}(\mathcal{S}^*, 130, 2)) = 17.0\%$.

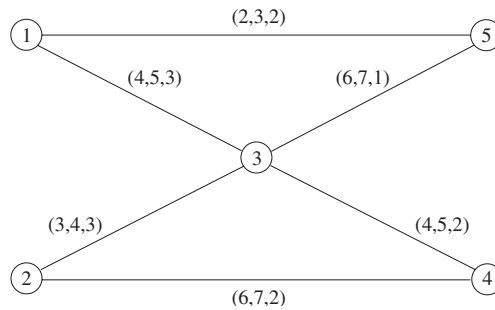


Fig. 4. Graphical representation of a small transportation network, \mathcal{G}^* .

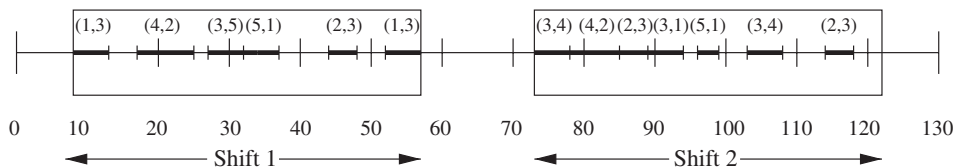


Fig. 5. Graphical representation of the schedule $\mathcal{W}(\mathcal{S}^*, 130, 2)$ in Example 2.1.

3. Finding an initial solution

We generate a starting solution by linking routes through several copies of our problem graph \mathcal{G} in a space–time diagram, forcing temporal spread between the service times of copies of the same edge, for all edges. The method is first described in algorithmic fashion, followed by a more detailed explanation of each step:

1. Construct $B = 2f_{\max}$ copies of the problem graph \mathcal{G} , where f_{\max} equals the highest service frequency in the problem graph. Assign to each graph a unique index from 1 through B .
2. Assign to each edge of \mathcal{G}_k the status *non-required*, $k = 1, \dots, B$.
3. Repeat the following until all edges of the original problem 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 $v_i v_j$ at random.
 - (b) Choose $f(i, j)$ indices between 1 and B (inclusive) according to a predetermined selection rule (described in more detail hereafter).
 - (c) For each of these $f(i, j)$ indices chosen, assign to edge $v_i v_j$ the status *required* in the correspondingly indexed graph \mathcal{G}_k .
4. Identify connecting vertices between the indexed graphs \mathcal{G}_k using shortest path information.
5. Find, in \mathcal{G}_k , a route that traverses the edges marked *required* at least once and that starts and ends at the vertices identified in the previous step according to a slight modification of Frederickson’s algorithm for the rural postman problem (described in more detail hereafter), for all $k = 1, \dots, B$.
6. Construct a service tour by linking the consecutively numbered paths through their ending and starting vertices and by linking the domicile vertex to the starting vertex of \mathcal{G}_1 and to the ending vertex of \mathcal{G}_B .
7. Convert the service tour to a sequence of the form $\mathcal{S} = \langle (v_{s_1}, v_{t_1}), (v_{s_2}, v_{t_2}), \dots, (v_{s_n}, v_{t_n}) \rangle$ and construct a corresponding schedule according to the conversion rules in Section 2.

We use $B = 2f_{\max}$ graphs in step 1 of the algorithm to ensure that no two edges in consecutively indexed graphs are marked *required*. The selection rule in step 3(b) chooses the set of $f(i, j)$ indexed graphs that has the minimum total service time for required edges already assigned to them, while ensuring a gap of at least $\lfloor B/f(i, j) - 1 \rfloor$ or $\lceil B/f(i, j) - 1 \rceil$ between the $f(i, j)$ indices. In step 4, the starting and ending vertices identified for consecutively indexed graphs are those pairs of vertices (one in each graph), considering only vertices incident to *required* edges that have the shortest travelling time between them. The starting vertex of the first graph and the ending vertex of the last graph are the domicile vertex. Step 5 implements a slight modification of Frederickson’s algorithm for the rural postman problem to find a path in each graph. The rural postman problem is known to be NP-Hard (Lenstra and Rinnooy-Kan, 1976). The interested reader may consult Frederickson (1979) and Eiselt et al. (1995b) for more information on the algorithm, but it essentially involves the computation of a minimum spanning tree and a minimum cost matching to find a tour. Our modification allows us to find a path beginning at the starting vertex and finishing at the ending vertex of the graph, as opposed to a tour that starts and ends at the same vertex. In step 6, shortest travelling paths are used to link the graphs. The tabu search algorithm (which will be used iteratively to improve upon the quality of the solution found here) and step 5

of the above algorithm also use this shortest path information. In our implementation, the shortest travelling times between all pairs of vertices are calculated at the beginning of the run, using Floyd's algorithm (Floyd, 1962). This algorithm has $O(p^3)$ worst case running time, where p is, as before, the order of the problem graph. The matching algorithm in step 5 also has $O(p^3)$ running time, though the best known implementations have slightly better worst case performance. The entire procedure for generating a starting solution therefore has a worst case running time of $O(p^3)$. The steps of the method as defined above are a convenient way of describing the functionality of the method, but the actual implementation differs somewhat. For example, our implementation does not wastefully store B copies of the problem graph. We now illustrate our method for constructing initial solutions by means of an example.

Example 3.1. Consider again the graph in Fig. 4. Suppose we desire a schedule of length 130 time units, consisting of two shifts and with a spread objective value less than or equal to 15%. Furthermore, suppose we wish to start and end the route at vertex number 1. The highest service frequency in the graph is 3, and therefore six auxiliary copies of the graph are made in the space–time diagram, shown in Fig. 6. In each of these graphs, some of the edges are designated as being *required* according to the rules described earlier. Figure 6 shows the required edges for each of these graph copies as bold-faced edges. The vertices at which the paths in each of the indexed graphs start and end are identified next. These vertices are shown in Table 1. A route is now found for each of these graphs according to our modified rural postman method. Table 2 shows the resultant path in each graph. The starting solution for the tabu search procedure is the schedule calculated from the route, starting from vertex 1, traversing each of these graphs and returning to vertex 1. This tour 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 starting solution is depicted by the arrows in Fig. 6. Dotted arrows denote free running, while solid arrows denote active servicing. As described earlier, the difference between the schedule time available and the total travelling time of the tour is allocated in equal amounts before the start and after the end of each shift. The tour is then separated and placed into the resulting shifts to form the schedule $\mathcal{W}(\mathcal{S}_0, 130, 2)$. The routing objective value of this solution is $\mathcal{R}(\mathcal{S}_0) = 90$ time units. Figure 7 depicts schedule $\mathcal{W}(\mathcal{S}_0, 130, 2)$, for which the scheduling objective value is $\mathcal{T}(\mathcal{W}(\mathcal{S}_0, 130, 2)) = 19.5\%$ (this is worse than the minimally acceptable value of 15% – it is even

Table 1
Vertices at which the paths in each of the indexed graphs \mathcal{G}_i^* start and end,
 $i = 1, \dots, 6$

Graph	Starting vertex	Ending vertex
\mathcal{G}_1^*	1	3
\mathcal{G}_2^*	3	5
\mathcal{G}_3^*	5	4
\mathcal{G}_4^*	4	3
\mathcal{G}_5^*	3	1
\mathcal{G}_6^*	1	1

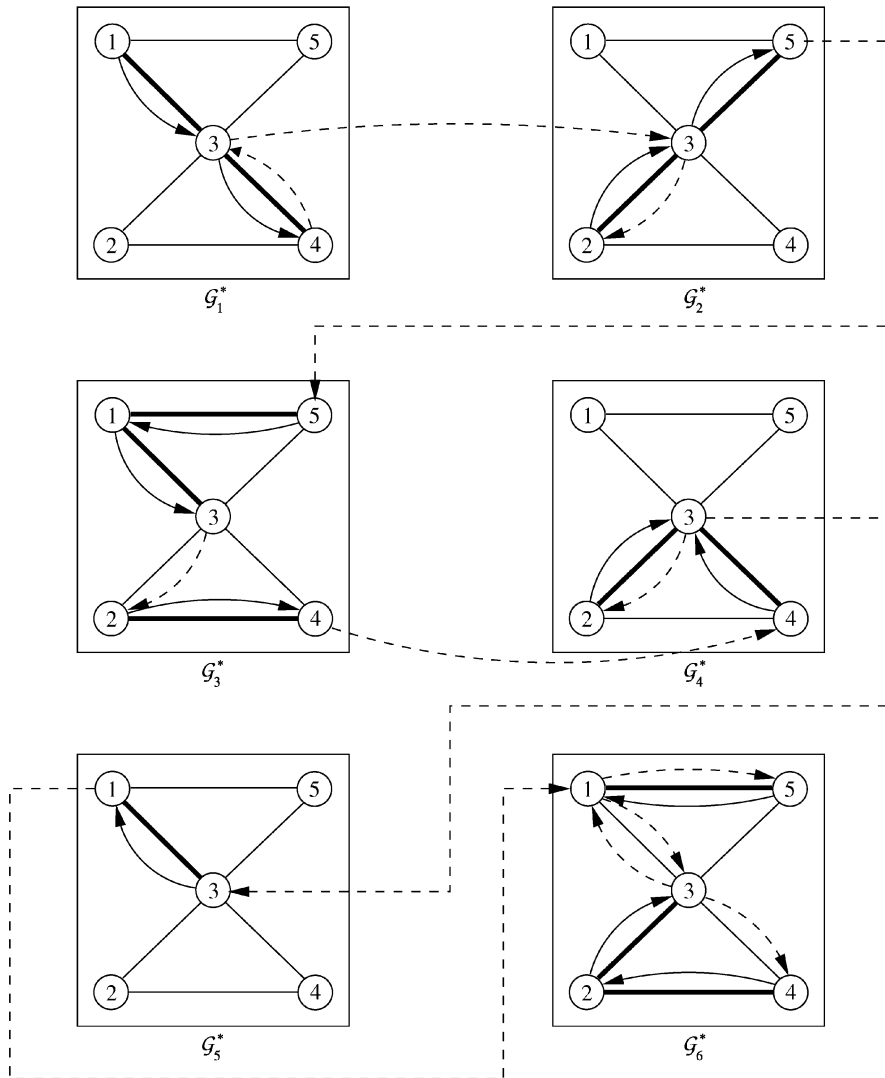


Fig. 6. Space–time representation of initial service routing and scheduling (represented by the dotted and solid arrows). Required edges for each of the indexed graphs \mathcal{G}_k^* , $k = 1, \dots, 6$ are shown as thick edges.

worse than the 17% scheduling objective value associated with the arbitrary service route \mathcal{S}^* presented in Example 2.1, but the routing objective is much better than that of Example 2.1).

4. Optimisation procedure

Local search methods, such as the tabu search method, which we employ to improve upon the quality of the initial solution found in Section 3, operate by iteratively modifying the structure of some candidate solution in coded form to produce a new coded candidate solution each time. This

Table 2
Service routes within the indexed graphs \mathcal{G}_i^* , $i = 1, \dots, 6$.

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

Edges that are serviced actively are shown in bold face.

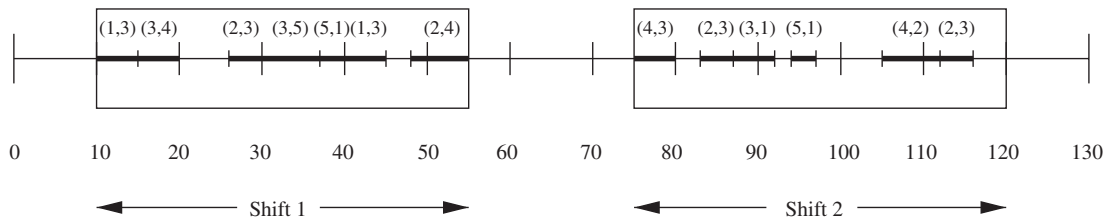


Fig. 7. Service schedule $\mathcal{W}(\mathcal{S}_0, 130, 2)$ for Example 3.1.

process is controlled in a way that will hopefully improve the candidate solution over time. Tabu search procedures use memory structures to record attributes of these modifications or *moves*. Moves that would result in the reversal or possible repetition of any or combinations of these attributes are designated as *tabu* for a certain number of iterations of the procedure. During each iteration of the procedure, several potential moves are evaluated and the best non-tabu move is chosen to produce the next solution. In most implementations, as in ours, it is accepted that the best tabu move is chosen instead if it is better than any solution encountered up to that point or if no non-tabu moves can be made. The motivation for using the tabu system is that it prevents the procedure from cycling between the same set of solutions, since non-improving moves are allowed, and because it forces the procedure to modify portions of the solution encoding that have not recently been modified (i.e. it enforces some *diversification*).

As indicated earlier, a solution is encoded by a service sequence \mathcal{S} . The quality of a solution depends not only on the order of the edges within \mathcal{S} , but also on the direction of their traversal during services. For example, the service sequence \mathcal{S}_0 in Example 3.1 renders a routing objective value of $\mathcal{R}(\mathcal{S}_0) = 90$ units, while the alternative service sequence

$$\mathcal{S}'_0 = \langle (1, 3), \underline{(4, 3)}, (2, 3), (3, 5), (5, 1), (1, 3), (2, 4), (4, 3), (2, 3), (3, 1), (5, 1), (4, 2), (2, 3) \rangle,$$

in which the service direction of the second entry of \mathcal{S}_0 has been swapped, renders the routing objective value $\mathcal{R}(\mathcal{S}'_0) = 98$ units.

It is conceivable, if we were to use a longer path than the shortest travelling time path (by which to travel between two edges in \mathcal{S} , that it might be possible to obtain a better scheduling objective value $\mathcal{T}(\mathcal{W}(\mathcal{S}, \tau, \kappa))$. However, this is not practically desirable, because it would be wasteful for the service vehicle to drive the extra distance when it might as well be idle. In our model the idle

times take place between shifts, so that the number of shifts used is an important parameter affecting the solution quality.

The task of our (heuristic) optimisation procedure will therefore be to find a good ordering of the entries of \mathcal{S} and to find a good direction in which to service the edges in this ordering. Given the user-defined schedule information and assuming that shortest paths between all pairs of vertices have been calculated, this is the only information that is necessary to construct a schedule.

4.1. Service directions in the solution sequence

Before describing the rest of the tabu search procedure we discuss the problem of determining the direction of the traversal of the edges of the routing sequence \mathcal{S} that will result in the shortest total travelling time, given a fixed ordering of the edges within \mathcal{S} .

Consider a routing sequence $\mathcal{S} = \langle (v_{s_1}, v_{t_1}), (v_{s_2}, v_{t_2}), \dots, (v_{s_n}, v_{t_n}) \rangle$. We wish to rearrange the directions of edges within \mathcal{S} without altering their ordering to obtain the shortest possible travelling time. From the solution sequence, a directed, layered auxiliary graph \mathcal{L} with vertex set $\mathcal{V}(\mathcal{L}) = \{v_s, v_{s_1}v_{t_1}, v_{t_1}v_{s_1}, v_{s_2}v_{t_2}, v_{t_2}v_{s_2}, \dots, v_{s_n}v_{t_n}, v_{t_n}v_{s_n}, v_t\}$ is constructed. Each layer of the graph consists of two service directions $v_{s_i}v_{t_i}$ and $v_{t_i}v_{s_i}$, ($0 < i \leq n$). The vertices v_s and v_t both represent the domicile vertex, at which the vehicle starts and ends its journey. Figure 8 depicts this layered graph.

For each i ($0 < i < n$), construct edges in \mathcal{L} directed from $v_{s_i}v_{t_i}$ to $v_{s_{i+1}}v_{t_{i+1}}$ and $v_{t_{i+1}}v_{s_{i+1}}$, and from $v_{t_i}v_{s_i}$ to $v_{s_{i+1}}v_{t_{i+1}}$ and $v_{t_{i+1}}v_{s_{i+1}}$. We also add edges directed from v_s to $v_{s_1}v_{t_1}$ and $v_{t_1}v_{s_1}$ and from $v_{s_n}v_{t_n}$ and $v_{t_n}v_{s_n}$ to v_t . A weight of $d(s_i, t_i) + e(t_i, s_{i+1})$ [$d(t_i, s_i) + e(s_i, t_{i+1})$, respectively] is assigned to the edge in \mathcal{L} between $v_{s_i}v_{t_i}$ and $v_{s_{i+1}}v_{t_{i+1}}$ [$v_{t_i}v_{s_i}$ and $v_{t_{i+1}}v_{s_{i+1}}$, respectively] for every i ($0 < i < n$). Similarly, a weight of $d(s_i, t_i) + e(t_i, t_{i+1})$ [$d(t_i, s_i) + e(s_i, s_{i+1})$, respectively] is assigned to the edge in \mathcal{L} between $v_{s_i}v_{t_i}$ and $v_{t_{i+1}}v_{s_{i+1}}$ [$v_{t_i}v_{s_i}$ and $v_{s_{i+1}}v_{t_{i+1}}$, respectively] for every $0 < i < n$.

Each path from v_s to v_t in \mathcal{L} represents a unique way to arrange the service directions of edges within \mathcal{S} . Our objective is to find a path from v_s to v_t that will represent the least possible routing cost objective value. This shortest path represents the optimal directions by which to service the edges of \mathcal{G} according to the ordering within \mathcal{S} . The length of this shortest path equals the total routing cost $\mathcal{R}(\mathcal{S})$ of the tour induced by the routing sequence \mathcal{S} . This shortest path sub-problem is applied to \mathcal{S} every time a change is made to its ordering. Due to the fact that \mathcal{L} is acyclic, a shortest path algorithm that is linear with respect to the order of \mathcal{L} may be used (Melhorn and Nüher, 1999) to determine the optimal service directions for edges in \mathcal{S} (no updating of information, such as occurs in Dijkstra's or Floyd's methods, is necessary during the algorithm execution). Further execution time is saved by taking advantage of the fact that \mathcal{L} is already ordered, so that nodes in later layers cannot reach nodes in preceding layers, meaning that the shortest path algorithm does not need to compute this ordering.

4.2. Tabu search moves

The moves discussed in this section are responsible for determining the ordering of edges within \mathcal{S} (disregarding the service directions of these edges). The shortest path sub-problem in Section 4.1

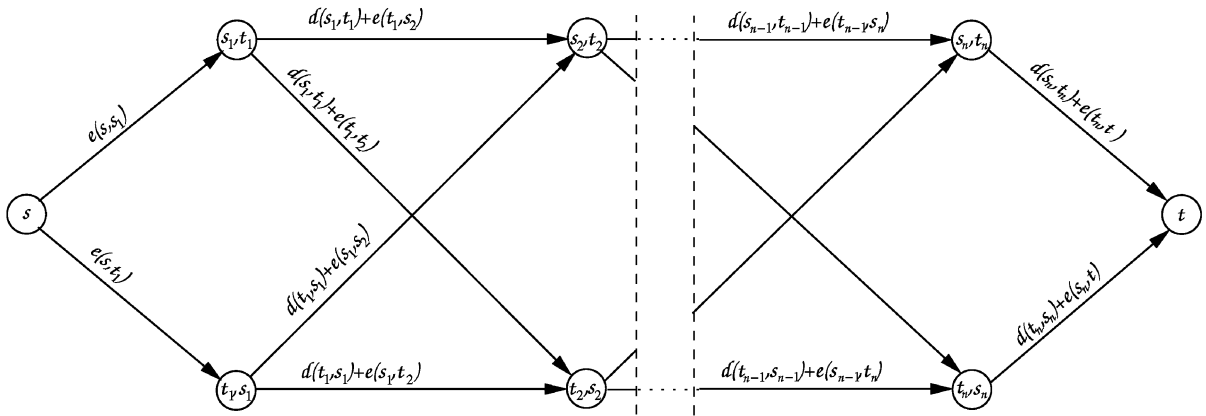


Fig. 8. The layered graph \mathcal{L} for a routing sequence $\mathcal{S} = \langle (v_{s_1} v_{t_1}), (v_{s_2} v_{t_2}), \dots, (v_{s_n} v_{t_n}) \rangle$.

may be seen as a component of each of these move types, because it is executed whenever any of these move procedures are executed. Only one type of move is executed per iteration of the tabu search algorithm. The move types are listed below.

(A) *Swap move*. This move procedure swaps the positions of two elements in the routing sequence \mathcal{S} . For example, swapping the first and the third elements in the sequence \mathcal{S}_0 (as found in Example 3.1), the alternative sequence

$$\mathcal{S}_{0,\text{before}}^{\text{swap}} = \langle \underline{(2, 3)}, (3, 4), \underline{(1, 3)}, (3, 5), (5, 1), (1, 3), (2, 4), (4, 3), (2, 3), (3, 1), (5, 1), (4, 2), (2, 3) \rangle$$

is found (*before* finding the optimal service directions). After finding the optimal service directions, the routing becomes

$$\mathcal{S}_{0,\text{after}}^{\text{swap}} = \langle \underline{(3, 2)}, (4, 3), \underline{(1, 3)}, (3, 5), (5, 1), (1, 3), (2, 4), (4, 3), (2, 3), (3, 1), (5, 1), (4, 2), (2, 3) \rangle,$$

for which the routing objective value is $\mathcal{R}(\mathcal{S}_{0,\text{after}}^{\text{swap}}) = 98$ units instead of the value $\mathcal{R}(\mathcal{S}_0) = 90$.

(B) *Insert move*. This procedure moves an edge in \mathcal{S} from one position to another. The edges between the two positions move accordingly to fill the empty position. For example, moving the first element of \mathcal{S}_0 (as found in Example 3.1) to the fourth position yields the alternative sequence

$$\mathcal{S}_{0,\text{before}}^{\text{insert}} = \langle (3, 4), (2, 3), (3, 5), \underline{(1, 3)}, (5, 1), (1, 3), (2, 4), (4, 3), (2, 3), (3, 1), (5, 1), (4, 2), (2, 3) \rangle$$

(*before* finding the optimal service directions). The routing remains unchanged after finding the optimal service directions, for which the routing objective value is $\mathcal{R}(\mathcal{S}_{0,\text{before}}^{\text{insert}}) = \mathcal{R}(\mathcal{S}_{0,\text{after}}^{\text{insert}}) = 102$ units instead of the original value of $\mathcal{R}(\mathcal{S}_0) = 90$.

(C) *Slide move*. The edges in \mathcal{G} with a service frequency greater than one will have more than one copy in \mathcal{S} . The *slide* move takes all copies of the same edge and moves each of them the same number of positions to the left or right in \mathcal{S} . For example, sliding all copies of (1, 3) in \mathcal{S}_0 three positions to the right, and sliding all copies of (3, 4) in \mathcal{S}_0 one position to the left yields the

alternative sequences

$$\mathcal{S}_{0,\text{before}}^{\text{right}} = \langle (3, 4), (2, 3), (3, 5), \underline{(1, 3)}, (5, 1), (2, 4), (4, 3), (2, 3), \underline{(1, 3)}, (5, 1), (4, 2), (2, 3), \underline{(3, 1)} \rangle$$

$$\mathcal{S}_{0,\text{before}}^{\text{left}} = \langle \underline{(3, 4)}, (1, 3), (2, 3), (3, 5), (5, 1), (1, 3), \underline{(4, 3)}, (2, 4), (2, 3), (3, 1), (5, 1), (4, 2), (2, 3) \rangle$$

(before finding the optimal service directions). After finding the optimal service directions, the routings become

$$\mathcal{S}_{0,\text{after}}^{\text{right}} = \langle (3, 4), (2, 3), (3, 5), \underline{(1, 3)}, (5, 1), (2, 4), (4, 3), (2, 3), \underline{(3, 1)}, (5, 1), (4, 2), (2, 3), \underline{(3, 1)} \rangle$$

$$\mathcal{S}_{0,\text{after}}^{\text{left}} = \langle \underline{(4, 3)}, (1, 3), (2, 3), (3, 5), (5, 1), (1, 3), \underline{(3, 4)}, (4, 2), (2, 3), (3, 1), (5, 1), (4, 2), (2, 3) \rangle,$$

for which the routing objective values are $\mathcal{R}(\mathcal{S}_{0,\text{after}}^{\text{left}}) = 102$ and $\mathcal{R}(\mathcal{S}_{0,\text{after}}^{\text{right}}) = 93$ units.

(D) *Expand move*. The *expand* move type increases the number of positions separating copies of the same edge in \mathcal{S} by the same amount. For example, expanding the interval between repetitions of the edge (1, 5) within \mathcal{S}_0 (as found in Example 3.1) by one position, yields the alternative sequence

$$\mathcal{S}_{0,\text{before}}^{\text{expand}} = \langle (1, 3), (3, 4), (2, 3), \underline{(5, 1)}, (3, 5), (1, 3), (2, 4), (4, 3), (2, 3), (3, 1), \underline{(5, 1)}, (4, 2), (2, 3) \rangle$$

(before finding the optimal service directions). After finding the optimal service directions, the routing becomes

$$\mathcal{S}_{0,\text{after}}^{\text{expand}} = \langle (1, 3), (3, 4), (2, 3), \underline{(1, 5)}, (5, 3), (1, 3), (2, 4), (4, 3), (2, 3), (1, 3), \underline{(5, 1)}, (4, 2), (2, 3) \rangle,$$

for which the routing objective value is $\mathcal{R}(\mathcal{S}_{0,\text{after}}^{\text{expand}}) = 98$ units.

(E) *Contract move*. The *contract* move type is similar to the *expand* move except that it reduces the number of positions between all copies of the same edge by the same amount.

(F) *Two-opt move*. This move type is similar to its counterpart for the travelling salesman problem (Croes, 1958), except for the fact that the direction of traversal of the edges in \mathcal{S} still has to be calculated after its application. Two-opt works by removing two shortest paths in a tour and then reconnecting the tour in a different way. Figure 9 depicts the mechanism behind the method. The method essentially chooses two points in \mathcal{S} and reverses the order in which the elements between the two points are serviced. For example, performing an instance of two-opt on the edges between positions 2 and 5 within \mathcal{S}_0 (as found in Example 3.1) yields the alternative sequence

$$\mathcal{S}_{0,\text{before}}^{2\text{Opt}} = \langle (1, 3), \underline{(5, 1)}, \underline{(3, 5)}, (2, 3), \underline{(3, 4)}, (1, 3), (2, 4), (4, 3), (2, 3), (3, 1), (5, 1), (4, 2), (2, 3) \rangle$$

(before finding the optimal service directions). After finding the optimal service directions, the routing becomes

$$\mathcal{S}_{0,\text{after}}^{2\text{Opt}} = \langle (1, 3), \underline{(1, 5)}, \underline{(5, 3)}, (3, 2), \underline{(4, 3)}, (1, 3), (2, 4), (4, 3), (2, 3), (3, 1), (5, 1), (4, 2), (2, 3) \rangle,$$

for which the routing objective value is $\mathcal{R}(\mathcal{S}_{0,\text{after}}^{2\text{Opt}}) = 98$ units.

If a slide or two-opt move is made during an iteration, then all possible combinations of that move type are considered. The number of combinations evaluated during an expand or contract move depends on how close all the copies of the same edges are to each other in \mathcal{S} . In any event, the running time of determining a neighbour move is $O(n^3)$ for all of the move types, where n is the

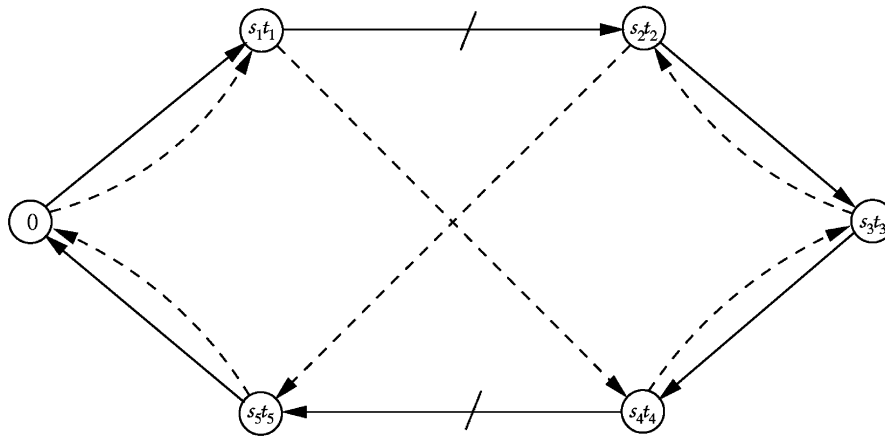


Fig. 9. The mechanism behind the two-opt move. The edges (s_1t_1, s_2t_2) and (s_4t_4, s_5t_5) may, for example, be removed and the traversal reconnected to form the path $0, s_1t_1, s_4t_4, s_3t_3, s_2t_2, s_5t_5, 0$.

number of services in \mathcal{S} . Our algorithm steps through a fixed sequence of these move types. This sequence is arranged so that moves that potentially make major changes to the structure of \mathcal{S} (two-opt, slide, contract and expand) are followed by swap or insert moves that perform a fine-tuning function.

4.3. Tabu lists

The tabu lists of a tabu search algorithm store the information relating to which attributes of moves are tabu. In our case a move is designated as tabu, if any of the edges in \mathcal{S} are assigned a new neighbour to which it was recently adjacent on the same side. This implies that every time an edge changes neighbours, it becomes tabu with respect to its old neighbour on the side from which it was moved. If this edge is now moved adjacent to and on the same side as this old neighbour or if the old neighbour is moved adjacent to and on the same side as the edge, the status of the move will be tabu. These tabu attributes remain for a fixed duration.

4.4. Controlling the search

Standard descriptions of heuristic search techniques assume that the algorithm is optimising a single objective. However, our problem has two objectives: a routing objective and a scheduling objective. These objectives tend to conflict with each other, which creates a problem when comparing candidate neighbourhood moves with respect to their efficiencies. To remedy this situation, a user-defined threshold value, denoted $\hat{\mathcal{T}}$, is defined for the scheduling objective, $\mathcal{T}(\mathcal{W}(\mathcal{S}, \tau, \kappa))$: this is an amount above which a candidate solution is considered unacceptable. The search method therefore seeks a shortest route for which

$$\mathcal{T}(\mathcal{W}(\mathcal{S}, \tau, \kappa)) \leq \hat{\mathcal{T}}. \quad (4.1)$$

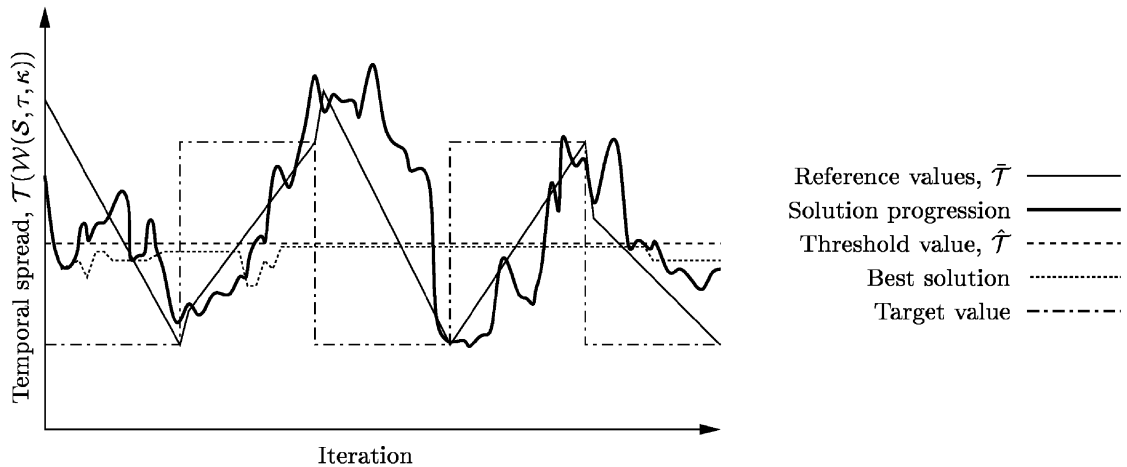


Fig. 10. Strategic oscillation approach towards controlling the search.

For the purposes of this discussion a solution, \mathcal{S} , will be called acceptable if (4.1) is satisfied, and if $\mathcal{R}(\mathcal{S}) \leq \tau$. A score value,

$$\mathcal{L}(\mathcal{S}, \tau, \kappa, i) = c(i)\mathcal{R}(\mathcal{S}) + [1 - c(i)]\mathcal{T}(\mathcal{W}(\mathcal{S}, \tau, \kappa)) \tag{4.2}$$

is assigned to any candidate solution during iteration i of the tabu search, where $0 \leq c(i) \leq 1$ is a parameter whose value is adjusted in alternating phases so as to control the emphasis given to each search objective. The goal of adjusting $c(i)$ according to these alternating phases is to drive $\mathcal{T}(\mathcal{W}(\mathcal{S}, \tau, \kappa))$ to either an upper or lower target value during each phase. These target values are set equal to a fixed percentage above and below the best value of $\mathcal{T}(\mathcal{W}(\mathcal{S}, \tau, \kappa))$ encountered up to that point during the search. If no solution satisfying (4.1) has been encountered, the lower target value is set to some fixed percentage below the threshold value, and the upper target value is not used. This procedure attempts to reach the target value within a certain number of iterations, I (say), by adjusting the value of $c(i)$ at the start of each iteration according to the difference between $\mathcal{T}(\mathcal{W}(\mathcal{S}, \tau, \kappa))$ and a reference value, denoted by $\bar{\mathcal{T}}(i)$. The search phase changes only when the search has been in a single phase for I or more iterations and when the spread tolerance (4.1) is violated. This implies that the procedure resembles more closely a single objective search if it struggles to find a solution with a good temporal spread objective value, or if either a very high tolerance $\hat{\mathcal{T}}$ is specified, or if there are no edges that require more than one traversal. For large problems and relatively low threshold values $\hat{\mathcal{T}}$, the best solutions tend to be very close to $\hat{\mathcal{T}}$, causing the procedure to oscillate about the threshold value. This oscillation is similar to the notion of strategic oscillation of Glover (1995), in which oscillation takes place about the feasibility level of some constraint. The motivation for spending time searching the unacceptable regions of the solution space is that this may allow the procedure to uncover combinations of edges that yield good route segments, which may still be reconciled with the spread objective requirements as shown in Fig. 10.

The reference value, $\bar{\mathcal{T}}(i)$, is linearly increased from the value of $\mathcal{T}(\mathcal{W}(\mathcal{S}, \tau, \kappa))$ at the start of the search phase, so as to reach the (upper or lower) target value within I iterations. The value of

$c(i)$ is adjusted according to one of the values

$$c(i+1)^+ = \min \left\{ 1, c(i) \left[1 + k \left(\frac{\bar{\mathcal{T}}}{\mathcal{T}} - 1 \right) \right] \right\}$$

or

$$c(i+1)^- = c(i) \left[1 - k \left(1 - \frac{\bar{\mathcal{T}}}{\mathcal{T}} \right) \right]$$

during the next iteration, where $0 < k \leq 1$, depending on whether \mathcal{T} is smaller or larger (respectively) than $\bar{\mathcal{T}}$. The value of $c(1)$ is set equal to that value for which both objective functions of the starting solution would contribute in equal amounts to the solution score in (4.2), so as to provide some stability during early stages of the search.

At the end of iteration i , the solution, \mathcal{S}_i , with the best score value in (4.2) is compared with the best-encountered solution, \mathcal{S}^* (say), at that time. \mathcal{S}_i becomes the new best solution if (and only if) it fulfils any of the following conditions:

- \mathcal{S}_i is acceptable, while \mathcal{S}^* is not.
- \mathcal{S}_i is acceptable, and $\mathcal{R}(\mathcal{S}_i) < \mathcal{R}(\mathcal{S}^*)$.
- \mathcal{S}_i is not acceptable, but $\mathcal{R}(\mathcal{S}_i) < \mathcal{R}(\mathcal{S}^*)$, provided that $\mathcal{R}(\mathcal{S}^*) > \tau$.
- \mathcal{S}_i is not acceptable and $\mathcal{R}(\mathcal{S}_i) < \tau$, but $\mathcal{T}(\mathcal{W}(\mathcal{S}_i, \tau, \kappa)) < \mathcal{T}(\mathcal{W}(\mathcal{S}^*, \tau, \kappa))$.

The search procedure terminates when both the average percentage improvement per iteration drops below a user-defined value and a certain minimum number of iterations have taken place, or if the procedure has been running for longer than a user-defined timeout value.

This method of guiding the search is thought to be effective, because the emphasis given at any stage of the search to each of the objective functions is not explicitly controlled, but rather allowed to adjust according to the progression of the search and the quality of solutions found. The above method of controlling the search assumes that the two objective functions are negatively correlated, which was found to be true for test problems investigated. Suitable values for the parameters used in this method of oscillation were found experimentally, by considering a large number of test problems. For example, the choice of $k = 0.75$ (as taken throughout this paper) yielded good solutions. In these test problems, the spread objective function values seemed to follow $\bar{\mathcal{T}}(i)$ closely for all but very small problems ($|\mathcal{S}| < 20$).

Let us illustrate our search method by means of an example.

Example 4.1. Consider again the graph in Fig. 4 for which an initial solution was determined in Example 3.1. We assume a spread threshold value of $\hat{\mathcal{T}} = 15\%$ and that $\tau = 130$ and $\kappa = 2$. The algorithm proceeds as follows from the initial solution,

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

for which $\mathcal{R}(\mathcal{S}_0) = 90$ and $\mathcal{T}(\mathcal{W}(\mathcal{S}_0, 130, 2)) = 19.5\%$. Suppose that the designated order in which to execute moves is two-opt, swap, slide, insert, expand, two-opt, swap, two-opt, insert, swap, contract, swap, two-opt, ... Table 3 shows the objective functions, $\bar{\mathcal{T}}$ and c values for each of the iterations considered in this example.

Table 3

Tabu search results for Example 4.1 if the search procedure is left to run to completion

Solution	$\mathcal{R}(\mathcal{S})$	$\mathcal{T}(W(\mathcal{S}, 130, 2))$ (%)	$\bar{\mathcal{T}}$ (%)	c
\mathcal{S}_0 (initial solution)	90	19.5	–	–
\mathcal{S}_1 (after first iteration)	91	12.7	19.375	0.00216
\mathcal{S}_2 (after second iteration)	85	14.1	19.250	0.00301
\mathcal{S}_3 (after third iteration)	85	22.1	19.125	0.00383
\vdots	\vdots	\vdots		
\mathcal{S}_{64} (best solution)	77	14.4		

Iteration 1: The spread objective function value of the initial solution is unacceptable and the search starts in a phase that attempts to decrease the objective function value to 20% below that of the threshold value. The first move to be performed is two-opt. The best positions in the initial (heuristic) route at which to perform this move are determined to be at positions 3 and 13 in \mathcal{S}_0 . The route now becomes

$$\mathcal{S}_1 = \langle (1, 3), (4, 3), (\mathbf{3, 2}), (2, 4), (5, 1), (1, 3), (2, 3), (3, 4), (4, 2), (3, 1), (1, 5), (5, 3), (\mathbf{2, 3}) \rangle$$

(after finding optimal traversal directions). This solution is acceptable and becomes the best encountered.

Iteration 2: The value of the reference spread value $\bar{\mathcal{T}}$ and c , are adjusted according to the new solution (see Table 3). The next move to be made is a swap move. The best move is to exchange positions 8 and 9 of \mathcal{S}_1 . The route now becomes

$$\mathcal{S}_2 = \langle (1, 3), (4, 3), (3, 2), (2, 4), (5, 1), (1, 3), (3, 2), (\mathbf{2, 4}), (\mathbf{4, 3}), (3, 1), (1, 5), (5, 3), (2, 3) \rangle$$

(after finding optimal traversal directions). This move also yields a new best solution.

Iteration 3: The value of c is updated and the next move (a slide) is performed. The best slide is to move all copies of edge (2, 4) in \mathcal{S}_2 one position to the left. The result is

$$\mathcal{S}_3 = \langle (1, 3), (3, 4), (\mathbf{4, 2}), (2, 3), (5, 1), (1, 3), (\mathbf{4, 2}), (2, 3), (4, 3), (3, 1), (1, 5), (5, 3), (2, 3) \rangle$$

(after finding optimal traversal directions). For this solution the scheduling objective value is again worse than 15%.

The tabu status of the edges after performing iteration 3 is shown in Table 4. Table 3 shows the results when the search is left to run to completion. The best solution is found after 64 iterations and the search terminates after 770 iterations, when the average improvement per iteration completed drops below an improvement of 0.02% (as measured with respect to the first feasible solution obtained). The best solution is coded as

$$\mathcal{S}_{64} = \langle (2, 3), (5, 1), (1, 3), (3, 4), (4, 2), (2, 3), (3, 1), (1, 5), (5, 3), (3, 2), (2, 4), (4, 3), (3, 1) \rangle$$

(after finding optimal traversal directions). The schedule corresponding to the final solution is given in Fig. 11. The full routing corresponding to the solution is

$$(1, 3), (3, 2), (\mathbf{2, 3}), (3, 5), (\mathbf{5, 1}), (\mathbf{1, 3}), (\mathbf{3, 4}), (\mathbf{4, 2}), (\mathbf{2, 3}), (\mathbf{3, 1}), (\mathbf{1, 5}), (\mathbf{5, 3}), (\mathbf{3, 2}), (\mathbf{2, 4}), (\mathbf{4, 3}), (\mathbf{3, 1}).$$

Table 4

The tabu status of \mathcal{S} after three iterations

	$(1, 3)_1$		$(1, 3)_2$		$(1, 3)_3$		$(1, 5)_1$		$(1, 5)_2$		$(2, 3)_1$		$(2, 3)_2$		$(2, 3)_3$		$(2, 4)_1$		$(2, 4)_2$		$(3, 4)_1$		$(3, 4)_2$		$(1, 3)$	
	B	A	B	A	B	A	B	A	B	A	B	A	B	A	B	A	B	A	B	A	B	A	B	A	B	A
$(1, 3)_1$																										
$(1, 3)_2$																12										
$(1, 3)_3$																										
$(1, 5)_1$																										
$(1, 5)_2$																				13						
$(2, 3)_1$																										
$(2, 3)_2$						13																				
$(2, 3)_3$																13										12
$(2, 4)_1$																										
$(2, 4)_2$																										
$(3, 4)_1$																										
$(3, 4)_2$																										
$(1, 3)$																										

The table assumes that an edge remains tabu for 10 iterations. The table shows the last iteration at which moving the listed edges to either before (indicated by label B) or after (indicated by label A) each of the other edges is considered tabu. For example, entry $(1, 3)_2$ – $(2, 4)_1$ has a value of 12 in the ‘A’ column, indicating that any move which places the second copy of $(1, 3)$ (each copy in the table is numbered according to its position in the initial solution, \mathcal{S}_0) adjacent to and after the first copy of $(2, 4)$ is tabu until the algorithm has run for more than 12 iterations in total.

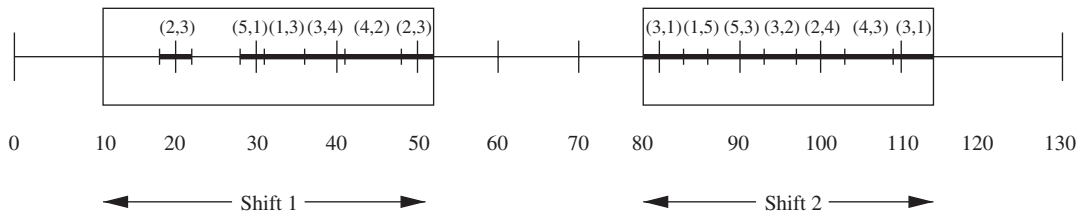


Fig. 11. Service schedule $\mathcal{W}(\mathcal{S}_{64}, 130, 2)$ for the best solution \mathcal{S}_{64} found to the network in Example 4.1.

5. Test results

The tabu search procedure, as described in Section 4, was implemented on a number of randomly generated graphs in an attempt to provide benchmark solutions for other researchers to use as a gauge of efficiency for future solution techniques that may be developed for our routing and scheduling problem. In total, 180 benchmark problems were created. Ten small instances ($20 \leq \text{size} \leq 35$ and $10 \leq \text{order} \leq 20$), 10 medium instances ($35 \leq \text{size} \leq 50$ and $20 \leq \text{order} \leq 30$) and 10 large instances ($50 \leq \text{size} \leq 65$ and $30 \leq \text{order} \leq 40$) of problem graphs were generated randomly in each of the following graph structure classes: (i) trees, (ii) trees with multiple star-like structures, (iii) general connected graphs, (iv) grid-like graphs, (v) circulant-like graphs and (vi) near-complete graphs. The exact method of random graph generation within each structure class as well as the full solutions obtained to each problem are fully documented in Groves et al. (2002, 2003).

In the above solutions, the fraction

$$L = \frac{\mathcal{R}(\mathcal{S})}{\sum d(i,j)f(i,j)} \quad (5.1)$$

was adopted as a measure of the routing efficiency found by the tabu search approach. The denominator in (5.1) represents an absolute (and in most cases unattainable) lower bound for $\mathcal{R}(\mathcal{S})$. Hence $L \geq 1$, and the smaller $L - 1$, the better the corresponding solution in terms of the routing cost objective function. However, even an optimal solution cannot reasonably be expected to yield an efficiency of $L = 1$, due to possibly forced back tracking. It was found, as expected, that graphs with larger edge densities yielded more efficient solutions than sparse graphs, in terms of the efficiency measure (5.1). For example, average values of (1.061, 1.035, 1.019) for L were obtained for, respectively, the classes of (small, medium, large) near-complete test problem graphs, while average values of (2.080, 2.210, 2.300) for L were obtained for, respectively, the classes of (small, medium, large) trees. It is, perhaps, surprising that this efficiency measure does not seem to increase significantly with the problem graph order, as was expected to happen.

6. A decision support system

The tabu search procedure described in Section 4 forms the backbone of a decision support system, called ROUTE OPTIMISER, which was developed as a possible aid to service inspectors of transportation networks. This system was developed on the LEDA platform for combinatorial and geometric computing (Melhorn and Nüher, 1999). Figure 12 shows a screen shot of ROUTE

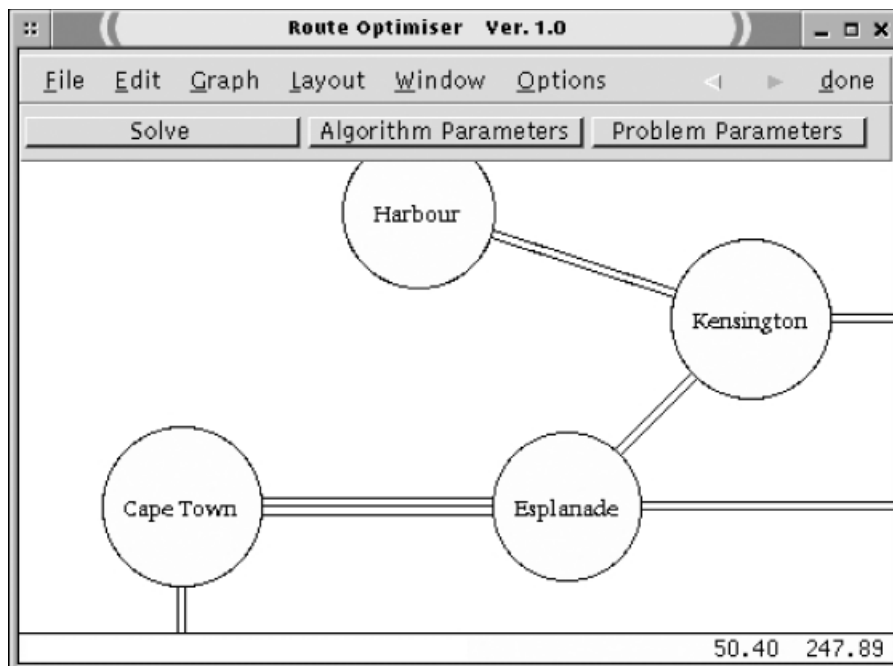


Fig. 12. Main screen shot of the decision support system ROUTE OPTIMISER.

OPTIMISER, version 1.0. The implementation requires the user to input the transportation network in the form of a triply weighted graph, as described in Section 2. Previously entered networks may also be loaded from or saved to disc. Once the network has been provided, the user is prompted for solution parameters such as the schedule length, number of shifts required and the minimally acceptable scheduling threshold value in terms of deviations from a perfectly spread schedule. Algorithm parameters (such as the stopping criterion or move types used) may also be specified by a specialist user; otherwise default values are used. The final solution output is given in the form of a table, listing edge traversals (distinguishing between free running and actual servicing) per shift. The system is also able to provide real-time data on the progress made by the tabu search method.

7. South African railways application

As mentioned in the introduction, the routing and scheduling problem considered in this paper came to the attention of the authors via SPOORNET, the (semi-privatised) main line railway authority and service provider in South Africa. During the late 1990s, SPOORNET engaged in various programmes to streamline its operations. One of these cost saving initiatives involved searching for an algorithm or suitable set of rules according to which the South African rail grid could be serviced efficiently on an annual basis. The authors subsequently developed the heuristic method presented in the main body of the paper and, after testing and benchmarking the efficiency of the algorithm on various random graph structures (as described in Section 5), the decision support system, ROUTE OPTIMISER (described in Section 6), was designed to solve problems such as those of SPOORNET. In order to test the practical applicability of ROUTE OPTIMISER, the system was applied to the South African national rail grid as a case study. The rail network, shown in Fig. 2, may be modelled by means of a graph (where vertices represent stations and edges represent rail tracks) with characteristics shown in Table 5. The main South African stations in this network are enumerated in Table A1, in the appendix.

There are four essentially different kinds of rail tracks in South Africa, and different speed limits for trains are enforced on these track types, as indicated below:

1. branch lines, 60 km/h;
2. secondary and metro lines, 80 km/h;
3. main lines, 100 km/h;
4. ore and coal lines, 40 km/h.

These lines have to be serviced with different frequencies according to an intricate set of rules, involving factors such as the speed limit and average axle load of trains on the particular lines,

Table 5
Characteristics of the SPOORNET graph

Problem dimension	Value
Graph order, p (number of stations)	195
Graph size, q (number of railtrack segments)	228
Length of solutions sequence, $ \mathcal{S} $	443

maximal gradients of the lines and quality demands made by large customers of SPOORNET. The resulting service frequency for each rail track segment in Fig. 2 is given in Table A2, in the appendix.

As mentioned before, SPOORNET leases only one (sophisticated) service vehicle, called the IM2000, shown in Fig. 1. The IM2000 is able to test tracks while travelling over them. It considers the comfort level of travel over the track, measures the angles of sleepers in their stone beds beneath the track and detects wear and tear of the tracks themselves. The operating cost of the IM2000 is approximately US\$11 per kilometre, irrespective of whether it is testing or free running. The IM2000 usually travels at speeds of between 40 and 100 km/h (depending on the condition of the track), and can travel a distance of at most 1500 km, before it has to stop and itself be serviced.

A schedule for a 1-year period (365 days, each comprising an 8-h work shift) was devised in accordance with the requirements of SPOORNET. A basic time unit of half an hour was adopted, giving rise to a schedule of length of $365 \times 8 \times 2$ time units (i.e. $\tau = 5840$ and $\kappa = 365$). The routing cost of each rail track segment in the South African rail grid was also measured in half-hour time units, and were calculated from the known distances spanned by each track and the corresponding speed limit on each track. These costs are the same for active servicing and for passive free running of tracks, and are listed for all track segments in Table A2, in the appendix. SPOORNET indicated that they would be willing to tolerate deviations from a perfect temporal spread of up to 15% in the service schedule.

The quality of the initial solution for the SPOORNET problem, as found by the graph theoretic procedure described in Section 3, is summarised in Table 6. For this problem the absolute (but unattainable) lower bound on the routing objective cost was $\sum d(i,j)f(i,j) = 1868$ half-hour units. This means that the level of efficiency in (5.1) achieved by the graph theoretic initial solution, was $L = 1.546$. The properties of the final solution, as determined by the tabu search procedure described in Section 4, are also shown in Table 6. The full final solution is given in Table A3, in the appendix. The stopping criterion for the tabu search method was activated if an improvement of less than 0.02% in the score in (4.2) of the best candidate solution (as measured with respect to the first feasible solution obtained) was observed, or if a timeout constraint of 48 h was reached. After 48 h of computing time on an Intel Pentium II 400 MHz processor with 64 MB of random access memory, the tabu search procedure yielded a 17% improvement on the initial routing cost objective (which translates to a measurable cost saving of US\$54,140) and a simultaneous improvement of 3.9% on the initial temporal spread objective, as found by the

Table 6

Results obtained for the SPOORNET graph on a Pentium II personal computer with an Intel 400 MHz processor and 64 MB of random access memory

Result	Value
Routing cost of initial solution	2888 half-hour units
Temporal spread objective value for initial solution	18.8%
Time required to find initial solution	1 s
Routing cost of best tabu search solution	2393 half-hour units
Temporal spread objective value for best tabu search solution	14.9%
Time required for tabu search	48 h

already efficient graph theoretic heuristic. The level of efficiency in (5.1) achieved by the final tabu search solution was $L = 1.281$.

SPOORNET received the routing schedule enthusiastically. Unfortunately, no data was available on the efficiency of previous schedules for the IM2000, but the engineers felt that no manual routing schedule could come close to the efficiency achieved by even the graph theoretic heuristic in our ROUTE OPTIMISER decision support system.

8. Evaluation of solution within the context of development

As mentioned in Section 6, the solution procedure described in this paper was implemented as a computerised decision support system, called ROUTE OPTIMISER, and was used to devise an efficient service routing schedule for the South African national rail grid, as described in Section 7. This service routing schedule was received enthusiastically by SPOORNET, since it seems to improve significantly upon current scheduling practice and therefore seems capable of bringing about significant cost savings.

Following the country's transformation to democracy in 1994, the South African government has intensified its efforts at the privatisation of state-owned enterprises. The privatisation of SPOORNET brought with it the threat of closure of many unprofitable railway lines in rural areas. However, the South African government was opposed to the closure of these lines, because it argued that the lines rendered a basic service to rural communities (which together constitute a large proportion of the South African population) and hence that the proposed closures were irreconcilable with the general spirit of *development* in South Africa, which the government had clearly set as one of its main priorities. In an effort to become profitable once more, improve their level of service and hopefully overturn their proposed rationalisation decisions, SPOORNET subsequently undertook several cost-cutting initiatives in response to the government's sentiments, of which the research project described in this paper, is one. It is within this context that the current project may be seen as an operations research project in *development*.

9. Conclusion

In this paper, a new graph heuristic was developed to solve the problem of finding an efficient (in terms of travel time) service routing through a transportation network that services each transportation link at least a specified (potentially different) number of times and which simultaneously attempts to spread out as much as possible consecutive services of the same network links over the total schedule window, for all links. This is a difficult problem, of which special case sub-problems are known to be NP-hard. Our solution approach involved use of the well-known tabu search heuristic in conjunction with various standard graph theoretic algorithms and has a total worst case scenario order of complexity of $O(q^3)$ per iteration, where q denotes the number of links in the network. Although the routing and scheduling problem was originally formulated and solved in the context of railway track service schedules, the method of course has wider application possibilities (including road network servicing, garbage collection (where it might be necessary to collect more frequently from industrial areas than from residential areas),

street sweeping (where it might be necessary to sweep more regularly in leafy or rural areas than in urban ones)).

Acknowledgements

The authors are indebted to Stephen Berjak and Werner Gründlingh for producing the graphics in this paper and to Stephen Benecke for his assistance in the library. This paper is based upon work 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 expressed in this paper are those of the authors and do not necessarily reflect the views of the National Research Foundation, the University of Stellenbosch or the University of South Africa.

References

- Amberg, A., Momshke, W., Voss, S., 2000. Multiple center capacitated arc routing problems: a tabu search algorithm using capacitated trees. *European Journal of Operational Research*, 124, 2, 360–376.
- Ball, M.O., Magnanti, T.L., Monma, C.L., Nemhauser, G.L. (eds) 1995. *Network Routing*. Elsevier, Amsterdam. (Volume in the series Nemhauser, G.L., Kan, A.H.G.R. (eds) *Handbooks in Operations Research and Management Science*).
- Croes, G.A., 1958. A method for solving travelling-salesman problems. *Operations Research*, 6, 791–812.
- Dror, M., Stern, H., 1987. Postman tour on a graph with precedence relation on arcs. *Networks*, 17, 283–294.
- Eiselt, H.A., Gendreau, M., Laporte, G., 1995a. Arc routing problems, Part I: the Chinese postman problem. *Operations Research*, 43, 2, 231–242.
- Eiselt, H.A., Gendreau, M., Laporte, G., 1995b. Arc routing problems, Part II: the rural postman problem. *Operations Research*, 43, 3, 399–414.
- Floyd, R.W., 1962. Algorithm 97: shortest path. *Commun. Ass. Comput. Mach.*, 5, 345.
- Frederickson, G.N., 1979. Approximation algorithms for some routing problems. *SIAM Journal on Computing*, 7, 178–193.
- Gendreau, M., Laporte, G., Semet, F., 1997. The covering tour problem. *Operations Research*, 45, 4, 569–576.
- Ghiani, G., Improta, G., 2000. An algorithm for the hierarchical Chinese postman problem. *Operations Research Letters*, 26, 1, 27–32.
- Glover, F., 1995. *Tabu Search and Uses*. Graduate School of Business, University of Colorado, Colorado.
- Golden, B.L., De Armon, J.S., Baker, E.K., 1983. Computational experiments with algorithms for a class of routing problems. *Computers and Operations Research*, 10, 1, 47–60.
- Groves, G.W., Le Roux, J., Van Vuuren, J.H., 2002. Scheduled Multiple Traversal Postman Problem. [Online], [Cited 2003, May 13], Available from <http://dip.sun.ac.za/~vuuren/repositories/smtpp.htm> or <http://jleroux.navors.net/repositories/smtpp.htm>
- Groves, G.W., Le Roux, J., Van Vuuren, J.H., 2003. On a routing & scheduling problem concerning multiple edge traversals in graphs, *Networks*, to appear.
- Hertz, A., Laporte, G., Hugo, P.N., 1999. Improvement procedures for the rural postman problem. *INFORMS Journal on Computing*, 11, 1, 53–62.
- Lenstra, J.K., Rinnooy-Kan, A.H.G., 1976. On general routing problems. *Networks*, 6, 273–280.
- Lin, Y., Zhao, Y., 1988. A new algorithm for the directed Chinese postman problem. *Computers and Operations Research*, 15, 6, 577–584.
- Malandraki, C., Daskin, M.S., 1993. The maximum benefit Chinese postman problem and the maximum benefit travelling salesman problem. *European Journal of Operational Research*, 65, 2, 218–234.

- Melhorn, K., Nüher, S., 1999. *LEDA: A Platform for Combinatorial and Geometric Computing*. Cambridge University Press, Cambridge, UK.
- Nobert, Y., Picard, J.-C., 1996. An optimal algorithm for the mixed Chinese postman problem. *Networks*, 27, 95–108.
- Pearn, W.L., 1989. Approximate solutions for the capacitated arc routing problem. *Computers and Operations Research*, 16, 6, 589–600.
- Pearn, W.L., 1991. Augment-insert algorithms for the capacitated arc routing problem. *Computers and Operations Research*, 18, 2, 189–198.
- Pearn, W.L., Wu, T.A., 1995. Algorithms for the rural postman problem. *Computers and Operations Research*, 22, 8, 819–828.
- Roy, S., Rousseau, J.-M., 1989. The capacitated Canadian postman problem. *INFOR*, 27, 1, 58–73.
- Stern, H.I., Dror, M., 1979. Routing electric meter readers. *Computers and Operations Research*, 6, 209–223.
- Wu, S., Manber, E., 1992. An algorithm for min-cost edge-disjoint cycles and its applications. *Operations Research Letters*, 12, 3, 173–178.
- Yan, H., Thompson, G.L., 1998. Finding postal carrier walk paths in mixed graphs. *Computational Optimization and Applications*, 9, 229–247.

Appendix

This appendix contains additional data in tabular form for the SPOORNET problem, which was deemed too bulky to include in the main body of the paper. The enumeration of stations shown on the map in Fig. 2 is shown in Table A1. This enumeration is utilised in the remaining two tables of the appendix. Table A2 contains the required service frequencies (in the form of services per annum) and the cost of (active or passive) traversals (measured in half-hour time units) of the IM2000 for all rail tracks in the South African national rail grid. Finally, Table A3 contains the final solution to the SPOORNET problem, as found by the tabu search procedure. These tables show only active edge traversals (i.e. tracks that are to be serviced) on a daily basis. It is assumed that shortest distances between consecutive tracks that have to be serviced may be determined via standard algorithms, such as those of Floyd or Dijkstra. The solution quality of the final solution shown in Table A3 is summarised in Table 6 in the main body of the paper.

Table A1
Station names versus vertex numbers in the SPOORNET graph

No.	Station	No.	Station	No.	Station
0	Cape Town	1	Simonstown	2	Bellville
3	Kraaifontein	4	Eersterivier	5	Stellenbosch
6	Kalbaskraal	7	Van Der Stel	8	Paarl
9	Saldanha	10	Eendekuil	11	Strand
12	Klipdale	13	Franschoek	14	Hemron
15	Sishen	16	Klawer	17	Protém
18	Bredasdorp	19	Porterville	20	Prince Alfred Hamlet
21	Worcester	22	Bitterfontein	23	Touwsrivier
24	Riversdale	25	Beaufort West	26	Voorbaai
27	Hutchinson	28	Mossel Bay	29	George
30	Kootjieskolk	31	De Aar	32	Knysna
33	Oudtshoorn	34	Calvinia	35	Noupoort
36	Groveput	37	Belmont	38	Calitzdorp
39	Klipplaat	40	Rosmead	41	Springfontein
42	Copperton	43	Upington	44	Douglas
45	Beaconsfield	46	Swartkops	47	Schoombiee
48	Koffiefontein	49	Dreunberg	50	Hamilton
51	Kakemas	52	Kimberley	53	Port Elizabeth
54	Barkly Bridge	55	Stomberg	56	Aliwal North
57	Bloemfontein	58	Kamfersdam	59	Alexandria
60	Addo	61	Sterkstroom	62	Barksly East
63	Sannaspos	64	Theunnisen	65	Postmasburg
66	Veertien Strome	67	Addo	68	Alicedale
69	Maclea	70	Bowker's park	71	Marseillies
72	Winburg	73	Virginia	74	Hotazel
75	Pudimoe	76	Makwassie	77	Cookhouse
78	Port Alfred	79	Imvani	80	Maseru
81	Modderpoort	82	Whites	83	Vermaas
84	Mafikeng	85	Ottosdal	86	Orkney
87	Somerset East	88	Amabele	89	Ladybrand
90	Bethlehem	91	Kroonstad	92	Coligny
93	Magaliesburg	94	Klerksdorp	95	Rooibloem
96	Blaney	97	Umtata	98	Harrismith
99	Arlington	100	Grootvlei	101	Dover
102	Lichtenburg	103	Bank	104	Krugersdorp
105	Pretoria	106	Cachet	107	Bultfontein
108	Mothusi	109	East London	110	Warden
111	Ladysmith	112	Marquard	113	Sasolburg
114	Kaydale	115	Vredefort	116	Midway
117	Johannesburg	118	Germiston	119	Witbank
120	Pretoria North	121	Vereeniging	122	Glencoe
123	Ennersdale	124	Springs	125	Ogies
126	Roosenekai	127	Wonderfontein	128	Brits
129	Pienaarsrivier	130	Newcastle	131	Vryheid
132	Bergville	133	Estcourt	134	Welgedag
135	Bethal	136	Delmas	137	Broodsnyersplaas
138	Belfast	139	Atlanta	140	Rustenburg

Table A1. (Contd.)

No.	Station	No.	Station	No.	Station
141	Pyramid South	142	Marble Hall	143	Nylstroom
144	Utrecht	145	Volksrust	146	Vryheids East
147	Hlobane	148	Merrivale	149	Hawerklip
150	Ermelo	151	Steelpoort	152	Machadodorp
153	Northam	154	Sentrastrand	155	Vaalwater
156	Naboomspruit	157	Firham	158	Richards Bay
159	Howick	160	Pietermaritzburg	161	Breyton
162	Lothair	163	Nelspruit	164	Middelwit
165	Thabazimbi	166	Drummondlea	167	Standerton
168	Empangeni	169	Kranskop	170	Donnybrook
171	Graskop	172	Plaston	173	Kaapmuiden
174	Ellisras	175	Chroomvalley	176	Pietersburg
177	Nkwalini	178	Gingindlovu	179	Durban
180	Richmond	181	Underberg	182	Franklin
183	Hoedspruit	184	Komatipoort	185	Soekmekaar
186	Tongaat	187	Umbilo	188	Matatiele
189	Kokstad	190	Phalaborwa	191	Letaba
192	Golela	193	Beitbridge	194	Wentworth

Table A2

Track weights for SPOORNET railway lines

Track	Time	Frequency	Track	Time	Frequency	Track	Time	Frequency
(0, 1)	2	1	(02)	1	3	(2, 3)	1	3
(2, 4)	1	2	(3, 5)	1	3	(3, 6)	1	1
(4, 5)	1	2	(4, 7)	1	1	(5, 8)	1	3
(6, 10)	6	1	(6, 9)	6	1	(7, 11)	1	1
(7, 12)	2	1	(8, 13)	1	1	(8, 14)	1	3
(9, 15)	43	4	(10, 16)	4	1	(12, 17)	1	1
(12, 18)	1	2	(14, 21)	1	3	(14, 19)	2	1
(14, 20)	1	1	(16, 22)	4	1	(21, 23)	2	3
(21, 24)	8	1	(23, 25)	6	3	(24, 26)	3	1
(25, 27)	3	3	(26, 28)	1	1	(26, 29)	2	1
(27, 30)	11	1	(27, 31)	3	3	(29, 32)	2	1
(29, 33)	2	1	(30, 34)	2	1	(31, 35)	3	2
(31, 37)	3	3	(31, 36)	6	1	(33, 38)	2	1
(33, 39)	8	1	(35, 40)	1	3	(35, 41)	3	3
(36, 42)	2	1	(36, 43)	7	1	(37, 44)	3	1
(37, 45)	2	3	(39, 40)	9	1	(39, 46)	6	1
(40, 47)	1	1	(41, 48)	5	1	(41, 49)	3	2
(41, 50)	3	3	(43, 51)	3	1	(45, 52)	1	3
(45, 50)	4	2	(46, 53)	1	3	(46, 54)	1	3
(47, 55)	1	1	(49, 56)	2	1	(50, 57)	1	3
(52, 58)	1	3	(54, 59)	1	1	(54, 60)	1	3
(49, 55)	1	2	(55, 61)	1	2	(56, 62)	5	1
(56, 63)	9	1	(57, 63)	1	1	(57, 64)	3	3

Table A2. (Contd.)

Track	Time	Frequency	Track	Time	Frequency	Track	Time	Frequency
(58, 65)	5	2	(58, 66)	1	3	(60, 67)	2	1
(60, 68)	1	3	(61, 69)	9	1	(61, 70)	1	2
(63, 71)	3	1	(64, 72)	1	1	(64, 73)	1	3
(65, 74)	3	2	(66, 75)	2	2	(66, 76)	3	3
(68, 77)	2	3	(68, 78)	5	1	(70, 79)	1	2
(71, 80)	1	1	(71, 81)	1	1	(73, 82)	1	3
(75, 83)	6	1	(75, 84)	7	2	(76, 85)	1	1
(76, 86)	2	3	(77, 87)	1	1	(40, 77)	2	3
(79, 88)	2	2	(81, 89)	1	1	(81, 90)	2	1
(82, 91)	1	3	(83, 92)	2	1	(83, 85)	1	1
(84, 93)	3	2	(85, 94)	2	2	(86, 94)	1	3
(86, 91)	2	2	(86, 95)	2	2	(88, 96)	1	1
(88, 97)	3	1	(90, 98)	1	3	(90, 99)	1	3
(90, 100)	2	1	(91, 99)	1	3	(91, 101)	1	3
(92, 102)	1	2	(92, 103)	3	2	(93, 104)	1	2
(93, 105)	1	1	(94, 106)	1	3	(95, 107)	2	2
(95, 108)	1	2	(77, 96)	8	1	(96, 109)	1	2
(98, 110)	2	1	(98, 111)	2	3	(99, 113)	6	1
(99, 112)	3	1	(100, 114)	2	1	(100, 121)	3	1
(101, 113)	1	3	(101, 115)	1	1	(103, 104)	1	3
(103, 116)	1	2	(103, 106)	1	3	(104, 117)	1	3
(105, 118)	1	3	(105, 119)	4	2	(105, 120)	1	3
(106, 121)	4	2	(82, 108)	1	2	(111, 122)	2	2
(111, 123)	1	3	(113, 121)	1	3	(114, 121)	1	3
(114, 118)	1	3	(114, 124)	1	2	(116, 121)	1	2
(116, 117)	1	2	(117, 118)	1	3	(119, 125)	1	2
(119, 126)	2	2	(119, 127)	2	2	(120, 128)	1	2
(120, 129)	1	3	(122, 130)	1	2	(122, 131)	2	2
(123, 132)	2	1	(123, 133)	1	3	(124, 134)	1	2
(124, 135)	3	2	(125, 136)	1	2	(125, 137)	1	2
(127, 138)	1	2	(127, 137)	1	2	(128, 139)	1	2
(128, 140)	1	2	(129, 141)	1	2	(129, 142)	4	1
(129, 143)	1	3	(130, 144)	1	2	(130, 145)	1	2
(131, 146)	1	2	(131, 147)	2	2	(133, 148)	2	3
(118, 134)	1	2	(134, 136)	1	2	(135, 145)	6	1
(136, 149)	1	2	(137, 150)	4	4	(138, 151)	5	2
(138, 152)	1	2	(140, 153)	2	2	(141, 154)	3	3
(143, 155)	2	2	(143, 156)	1	3	(145, 157)	2	2
(146, 150)	10	4	(146, 158)	10	4	(148, 159)	1	2
(148, 160)	1	3	(150, 161)	1	1	(150, 162)	1	1
(152, 163)	3	2	(152, 161)	3	1	(153, 164)	1	2
(153, 165)	1	2	(134, 154)	1	2	(156, 166)	1	3
(157, 167)	1	2	(158, 168)	1	2	(160, 169)	4	2
(160, 170)	5	1	(160, 179)	2	3	(160, 180)	1	1
(135, 161)	3	1	(163, 171)	5	2	(163, 172)	1	1
(163, 173)	1	2	(165, 174)	3	2	(166, 175)	2	1
(166, 176)	2	3	(114, 167)	2	2	(168, 177)	1	1
(168, 178)	1	2	(170, 181)	1	1	(170, 182)	3	1

Table A2. (Contd.)

Track	Time	Frequency	Track	Time	Frequency	Track	Time	Frequency
(173, 183)	5	2	(173, 184)	2	2	(176, 185)	2	2
(178, 186)	2	2	(179, 187)	1	2	(179, 186)	2	2
(182, 188)	2	1	(182, 189)	1	1	(183, 190)	1	2
(183, 191)	3	2	(184, 192)	4	2	(185, 193)	5	2
(187, 194)	1	2	(185, 191)	4	1	(158, 192)	5	2

The traversal and service time is expressed in half-hour units and is valid for both active and passive traversals by IM2000.

Table A3

Algorithmic solution to SPOORNET's routing problem

Day	Sequence of tracks serviced
2	(3, 2), (3, 6), (6, 10)
3	(9, 15)
12	(9, 6)
14	(8, 14)
16	(23, 25)
18	(25, 27)
20	(31, 27), (31, 36)
21	(31, 35)
23	(68, 77), (54, 60), (46, 54), (46, 53)
24	(87, 77), (35, 41)
26	(41, 48)
27	(64, 72), (73, 82)
29	(107, 95)
30	(113, 121), (114, 124), (161, 135)
33	(150, 137), (119, 126)
34	(119, 105), (93, 84)
36	(140, 153)
37	(156, 143), (156, 166), (176, 166), (193, 185)
40	(173, 183), (163, 173)
42	(152, 163), (173, 184)
44	(192, 158), (178, 186)
45	(169, 160)
46	(160, 170)
48	(179, 186)
50	(158, 146), (131, 147)
51	(111, 98)
53	(91, 99), (91, 86), (106, 94), (66, 76)
54	(37, 44)
56	(31, 37), (40, 35), (77, 40)
57	(68, 78)
59	(68, 60), (39, 46)
62	(29, 33), (38, 33)
64	(40, 39)
65	(61, 55), (70, 79), (79, 88), (96, 109)
68	(61, 70), (41, 49), (50, 41), (50, 57), (57, 64)
69	(82, 91)
70	(99, 112)

Table A3. (Contd.)

Day	Sequence of tracks serviced
73	(90, 99), (71, 81), (98, 90), (90, 100)
74	(121, 114)
75	(113, 101)
76	(91, 101), (85, 94), (75, 83), (75, 66)
78	(76, 86)
79	(86, 94), (103, 106), (103, 104), (104, 117), (104, 93), (139, 128), (140, 128), (153, 164)
80	(165, 174)
82	(143, 155), (129, 143), (120, 129), (120, 105)
83	(118, 105), (114, 118), (135, 124)
85	(136, 134), (136, 125), (137, 125), (171, 163)
88	(152, 161), (150, 146)
91	(122, 131), (160, 148), (187, 194), (179, 187)
94	(168, 158), (178, 168), (179, 160), (133, 148), (123, 133), (111, 123), (111, 122)
95	(122, 130), (157, 167)
97	(114, 167), (75, 84)
98	(66, 58), (52, 58), (45, 52), (37, 45)
101	(43, 51)
104	(31, 27), (27, 30)
107	(27, 25), (23, 21)
109	(14, 21), (3, 5), (2, 3), (0, 2)
110	(15, 9)
117	(5, 8)
120	(8, 13)
121	(8, 14)
122	(29, 32)
124	(33, 39)
125	(53, 46), (54, 46), (54, 59), (60, 54)
127	(68, 77)
128	(61, 55), (49, 55), (56, 49), (62, 56)
130	(56, 63)
131	(73, 64), (82, 73), (108, 82), (95, 108), (95, 86)
133	(106, 94), (121, 106)
134	(113, 121), (121, 116), (103, 92), (103, 116), (117, 116), (118, 117), (118, 134)
135	(154, 134), (141, 154)
136	(141, 129)
137	(120, 128), (153, 165), (156, 166), (176, 166)
140	(183, 191), (183, 190), (184, 192)
143	(158, 146), (146, 131), (150, 146)
146	(150, 162), (137, 150), (137, 127), (127, 138), (138, 152), (138, 151)
149	(119, 127), (119, 125), (136, 149), (124, 134), (135, 145)
150	(145, 157)
152	(130, 145), (130, 144)
153	(170, 182)
155	(160, 148), (148, 159)
156	(111, 98), (91, 99)
158	(66, 76), (65, 58), (65, 74)
161	(37, 45), (50, 45)
162	(35, 41), (35, 40)
164	(47, 55), (61, 69)
167	(97, 88), (96, 88), (79, 88)

Table A3. (Contd.)

Day	Sequence of tracks serviced
169	(79, 70), (77, 40)
171	(60, 68), (60, 67), (54, 60), (53, 46)
174	(26, 24), (24, 21)
177	(8, 14), (4, 5), (12, 18), (2, 4)
178	(2, 0), (2, 3)
180	(9, 15)
187	(5, 3)
189	(21, 14)
191	(21, 23)
192	(23, 25)
196	(31, 35), (50, 41), (64, 57)
198	(64, 73), (95, 107)
199	(82, 91), (101, 91), (101, 115), (101, 113), (121, 114), (114, 100)
201	(90, 98), (81, 90), (90, 99)
203	(86, 91), (94, 86), (76, 86), (85, 83), (85, 94)
205	(103, 106), (104, 103), (104, 117), (117, 118)
206	(140, 153), (141, 154)
208	(124, 114), (114, 118), (105, 118), (93, 105)
209	(84, 93), (105, 120), (120, 129), (129, 143), (143, 156), (176, 185)
211	(185, 193)
213	(183, 173)
215	(184, 173), (173, 163)
216	(152, 163), (137, 150), (126, 119)
219	(105, 119)
220	(92, 102)
222	(66, 75), (58, 66)
223	(52, 58), (45, 52), (31, 37), (36, 43)
228	(57, 50), (63, 57)
229	(81, 89), (98, 110), (98, 111), (122, 111)
233	(111, 123)
233	(133, 123), (148, 133), (182, 188)
236	(180, 160), (160, 169), (160, 179)
237	(187, 194)
239	(186, 179), (178, 186), (192, 158)
240	(146, 158)
242	(147, 131), (150, 146)
243	(161, 150), (135, 124)
246	(121, 100), (91, 99), (73, 82)
247	(121, 113), (136, 134), (125, 136)
249	(125, 137), (163, 171)
250	(172, 163)
252	(185, 191)
253	(166, 176), (175, 166), (156, 166)
255	(156, 143), (143, 155), (142, 129)
256	(165, 174)
259	(153, 164), (128, 140), (128, 139)
260	(93, 104), (94, 106), (85, 76)
262	(66, 76), (37, 45), (37, 31)
263	(42, 36)
265	(27, 31)

Table A3. (Contd.)

Day	Sequence of tracks serviced
266	(25, 27), (23, 25)
268	(5, 8)
269	(10, 16), (22, 16)
272	(15, 9)
280	(7, 11)
282	(12, 17)
285	(4, 7)
288	(0, 2)
289	(3, 5)
291	(14, 21), (26, 28)
292	(29, 26)
295	(46, 54), (68, 60)
296	(77, 68)
297	(77, 96)
298	(96, 109), (61, 70), (55, 49), (41, 49)
299	(41, 35)
300	(40, 35)
301	(47, 40), (40, 77), (41, 50), (63, 71)
302	(80, 71)
305	(113, 99), (114, 167), (167, 157), (157, 145)
307	(130, 145), (130, 144)
309	(122, 130), (131, 122), (158, 146), (158, 168), (177, 168)
312	(179, 187), (148, 160), (170, 181)
313	(182, 189)
315	(179, 160), (168, 178)
318	(184, 192), (183, 190), (191, 183)
319	(176, 185)
321	(129, 143), (165, 153)
322	(120, 128)
325	(75, 84), (45, 52), (50, 45)
326	(50, 57), (64, 57)
328	(64, 73), (108, 82), (108, 95), (86, 95), (76, 86), (83, 92), (102, 92)
329	(103, 92), (116, 103)
331	(106, 121), (121, 114), (117, 116), (121, 116), (113, 101), (91, 101)
332	(91, 82), (99, 90), (98, 90), (123, 132)
334	(123, 133), (148, 159), (133, 148), (123, 111)
335	(146, 131), (146, 150)
337	(150, 137), (127, 137), (138, 151)
338	(152, 138), (127, 138), (127, 119)
340	(119, 125), (149, 136), (134, 124), (114, 118), (105, 118), (105, 120)
341	(120, 129), (129, 141), (154, 141), (134, 154), (118, 134), (117, 118), (104, 117), (103, 104), (103, 106)
343	(86, 94), (58, 66)
344	(65, 74), (65, 58)
346	(52, 58)
349	(30, 34)
352	(21, 23)
353	(14, 20), (19, 14), (5, 8), (4, 5), (12, 18)
355	(7, 12), (2, 4), (0, 1)

Only active traversals by the IM2000 are shown.