LOCAL SEARCH IN ROUTING PROBLEMS WITH TIME WINDOWS

M.W.P. SAVELSBERGH

Department of Operations Research and System Theory, Centre for Mathematics and Computer Science, Kruislaan 413, NL-1098 SJ Amsterdam, The Netherlands

Abstract

We develop local search algorithms for routing problems with time windows. The presented algorithms are based on the k-interchange concept. The presence of time windows introduces feasibility constraints, the checking of which normally requires $O(N)$ time. Our method reduces this checking effort to $O(1)$ time. We also consider the problem of finding initial solutions. A complexity result is given and an insertion heuristic is described.

Keywords and phrases

Local search, vehicle routing problem, traveling salesman problem, k-interchange, NP-completeness, computational complexity, time windows, heuristics.

1. Introduction

Physical distribution is becoming more and more the subject of mathematical research. Among the great variety of distribution problems, two prevail: the routing of (capacitated) vehicles through a collection of points to pick up or deliver goods, the *vehicle routing problem* (VRP), and the scheduling of vehicles to meet time or precedence constraints imposed upon their routes, the *vehicle scheduling problem* (VSP). We will consider here a problem where the spatial aspect of routing is blended with the temporal aspect of scheduling, because time window constraints must be respected. Time window constraints form one of the extensions of the basic problem which arise in practical applications. Very little research has been done on this subject. Solomon [12,13] modifies existing heuristics for the VRP to handle time windows, and Christofides et al. [1] use state space relaxations to obtain bounds to be used in enumerative methods.

We will restrict ourselves to *local search* procedures for routing problems with time windows. About twenty years ago, Croes [2] and Lin [6] introduced the notion of k-interchanges to improve solutions of the *traveling salesman problem* (TSP) (Croes for $k = 2$ and Lin for $k = 3$). Several other papers have since been written that examine issues related to the application of this method. Lin and Kernighan [7] generalized it, and Papadimitriou and Steiglitz [9] reported results on its worst-case behaviour. The method has also been applied with considerable success to other classes of problems. Kanellakis and Papadimitriou [4] adapted the method for the use in asymmetric TSPs and Psaraftis [10] examined the use of k-interchanges in precedence constrained routing problems.

In this paper, a local search method based on the k-interchange concept will be presented which takes time windows into account. For the description of the method we will restrict ourselves to the TSP. However, the techniques presented are of a more general nature and can be applied in other types of routing problems as well. The presence of time windows in a TSP implies that we are not just looking for a cycle, but for a route starting and finishing at specific points in time. Therefore, we will consider the TSP *with time windows* (TSPTW) in which a vertex will be specified (the depot in vehicle routing problems) at which the salesman starts and finishes. This has the advantage that the presented method can easily be incorporated in cluster first-route second approaches to the VRP with time windows.

The approach we will follow draws from the k-interchange procedure by Lin [6] for the TSP. As in the TSP, a k-interchange is a substitution of k links of a tour with k other links. In contrast to the (standard) TSP where the processing of a single k-interchange takes $O(1)$ time, testing in a straightfoward way whether a single TSPTW interchange does not violate the time window constraints requires $O(N)$ time, where N indicates the size of the problem (in this case the number of vertices). We will develop a method that performs this test in $O(1)$ time. In the second part of the paper, we will take a closer look at the question of finding an initial feasible tour. It turns out that the problem of deciding whether there exists a feasible tour at all is NP-complete in the strong sense. This result justifies the use of a heuristic procedure for the construction of an initial tour.

2. Local search for the standard TSP

In the TSP we are given a finite set V of vertices and a distance $t_{i,j}$ for each pair of vertices $i, j \in V$. The problem is to find a tour with minimal total length, where a tour is a closed path that visits each vertex exactly once (for further information on the TSP, we refer to [5]). A TSP tour of $N = |V|$ vertices can be described by a sequence $(1, 2, \ldots, i, \ldots, N)$, where i represents the ith vertex of the tour. It is clear that a TSP tour has N links. We make the additional assumption that the distance matrix is symmetric and satisfies the triangle inequality.

Recall that a k-interchange is a substitution of k links of a tour with k other links. A tour is said to be *k-optimal* (k-opt) if it is impossible to obtain a tour of shorter length by replacing k of its links by another set of k links. Since the computational effort rises rapidly with k, we only consider the cases $k = 2$ and $k = 3$.

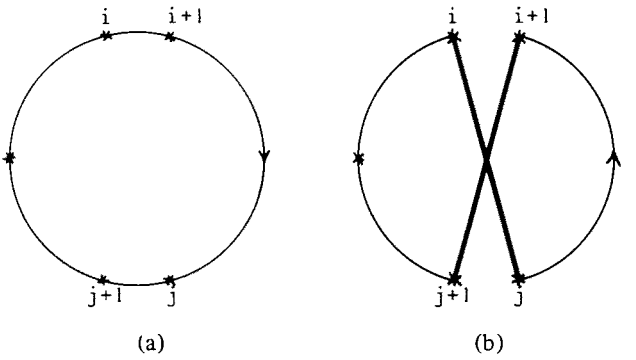

Fig. 1. A 2-interchange.

We start with the case $k = 2$. Performing a single 2-interchange on a TSP tour involves the substitution of two of its links, say $(i, i + 1)$ and $(j, j + 1)$ in fig. 1(a), with two other links, in this case (i, j) and $(i + 1, j + 1)$ in fig. 1(b).

Such an interchange results in a local tour improvement if and only if the following condition holds:

$$t_{i,i+1} + t_{j,j+1} > t_{i,j} + t_{i+1,j+1} .$$

Notice that the orientation of the path $(i + 1, \ldots, j)$ is reversed in the new tour. (In the sequel, when we are referring to a 2-interchange, we will always mean the deletion of the links $(i, i + 1)$ and $(j, j + 1)$ from the current tour and their replacement by the links (i, j) and $(i + 1, j + 1)$.) The total number of possible 2-interchanges equals the number of subsets of two links that can be formed from the set of N links that make up the tour. This number is equal to $\binom{N}{2}$, which implies a time complexity of $O(N^2)$ for the verification of 2-optimality.

In contrast to the case $k = 2$, where the two links $(i, i + 1)$ and $(j, j + 1)$ that will be deleted uniquely identify the two links (i, j) and $(i + 1, j + 1)$ that will replace them, in the case $k = 3$ there are eight ways of substituting any given triplet of links with a triplet of other links. Figures 2(b) and 2(c) show two of the eight possible 3-interchanges that can be performed by deleting the links $(i, i + 1)$, $(j, j + 1)$ and $(k, k + 1)$ of an initial TSP tour [fig. 2(a)].

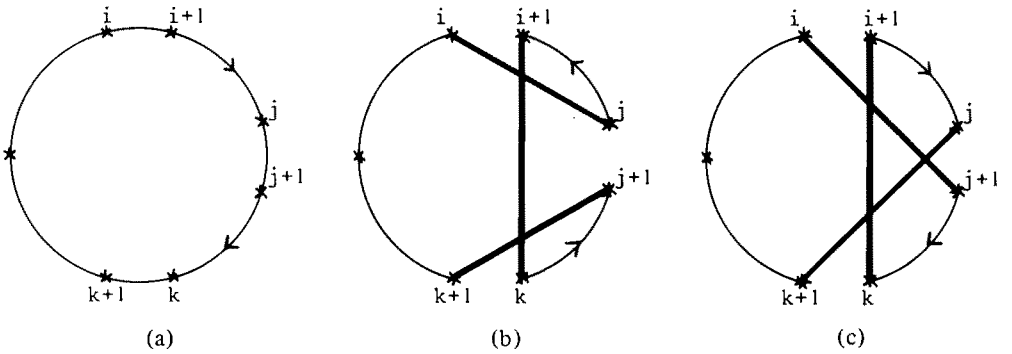

Fig. 2. Two ways to perform a 3-interchange.

For all cases, conditions similar to the one given for the case $k = 2$ can be given to obtain local tour improvement. There is one important difference between the two 3-interchanges shown in fig. 2, namely the fact that in the latter the orientation of the paths $(i + 1, \ldots, j)$ and $(j + 1, \ldots, k)$ is preserved, whereas in the former this orientation is reversed. The total number of possible 3-interchanges is proportional to the number of subsets of three links that can be formed from the set of N links that make up the tour. This number is equal to $\binom{N}{3}$, which implies a time complexity of $O(N^3)$ for the verification of 3-optimality.

Because the computational effort to verify 3-optimality becomes considerable if the number of vertices increases, proposals have been put forward to take only a subset of all possible 3-interchanges into account. We will consider the proposal by Or [8]. His procedure considers only those 3-interchanges that would result in a string of one, two, or three consecutive vertices being inserted between two other vertices. To see how the Or-opt procedure works, the reader is referred to fig. 3. In this tour, the string of three consecutive vertices i, $i + 1$ and $i + 2$ is relocated between j and $j + 1$. The time complexity to verify Or-optimality is $O(N^2)$.

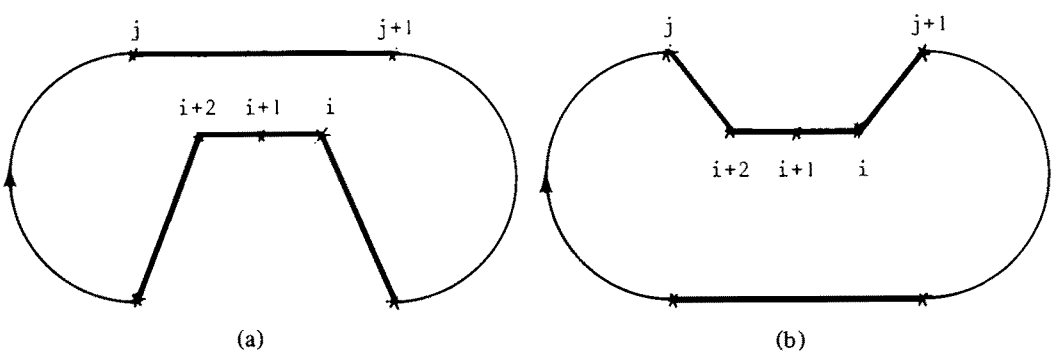

Fig. 3. An Or-interchange.

3. Local search for the TSP with time windows

In the TSPTW we are given in addition to the travel time $t_{i,j}$ for each pair of vertices $i, j \in V$, for each vertex i a *time window*, denoted by $[e_i, l_i]$, where e_i specifies the earliest service time and l_i the latest service time. The latter bound is strict in the sense that departing later than l_i is not allowed and causes the tour to become infeasible, whereas arriving earlier than e_i does not lead to infeasibility but merely introduces waiting time at vertex i. Throughout the paper, we will assume that there is no actual service time at any vertex. This means that we can (and will) depart from a vertex as soon as possible. The following quantities, given a feasible tour $(1, \ldots, N)$, will be very helpful for the description of the algorithm:

A_i, the arrival time at i;

$D_i := \max(A_i, e_i)$, the departure time at i;

$W_i := D_i - A_i$, the waiting time at i.

We make the following observations:

- $A_i < D_i \Leftrightarrow A_i < e_i \Leftrightarrow W_i > 0$;
- $A_i = D_i \Leftrightarrow e_i \leqslant A_i \Leftrightarrow W_i = 0$.

The conditions for local tour improvement in theTSPTW strongly depend on the chosen objective. We consider two objectives below. As to our notation, a quantity with superscript *new* indicates that the value is taken to be the one which would result if the interchange were carried out; the subscript still refers to the ordering of the current tour.

(1) Minimize the time spent on actual traveling:

$$\min\left\{\sum_{i=1}^{N-1} t_{i,i+1} + t_{N,1}\right\}.$$

With this objective, a 2-interchange is both feasible and profitable if and only if the following conditions are satisfied:

- the actual travel time is reduced:

$$t_{i,j} + t_{i+1,j+1} < t_{i,i+1} + t_{j,j+1};$$

– the new tour is feasible:

$$i < k \leqslant j: D_k^{\text{new}} = D_i + t_{i,j} + \sum_{p=k+1}^{j} (W_p^{\text{new}} + t_{p-1,p}) \leqslant l_k;$$

$$j < k \leqslant N: D_k^{\text{new}} = D_i + t_{i,j} + \sum_{p=i+2}^{j} (W_p^{\text{new}} + t_{p-1,p})$$

$$+ W_{i+1}^{\text{new}} + t_{i+1,j+1} + \sum_{p=j+1}^{k-1} (W_p^{\text{new}} + t_{p,p+1}) \leqslant l_k.$$

(2) Minimize the completion time of the tour:

$$\min D_N + t_{N,1}.$$

With this objective, a 2-interchange is both feaible and profitable if and only if the following conditions are satisfied:

– the arrival time at $j+1$ is decreased:

$$A_{j+1}^{\text{new}} < A_{j+1};$$

– and part of the gain can be carried through to the vertex where the salesman finishes:

$$j+1 \leqslant k \leqslant N: D_k > e_k;$$

– the reversed part of the tour is feasible:

$$i < k \leqslant j: D_i + t_{i,j} + \sum_{p=k+1}^{j} (W_p^{\text{new}} + t_{p-1,p}) \leqslant l_k.$$

The second condition needs some further consideration. If this condition is violated, the interchange will not alter the completion time of the tour. It will only reduce the completion time of the path from 1 to $k-1$, for the smallest k for which violation occurs. The question arises whether it is wise to carry out an interchange if only part of the route is completed earlier. We have adopted the following criterion:

A 2-interchange that reduces the completion time of an initial part of the tour but not of the complete tour is carried out if and only if it also reduces the actual total travel time.

The main problem with the use of k-interchange procedures in the TSPTW is checking the feasibility of an interchange. A 2-interchange will reverse the path $(i + 1, \ldots, j)$. But this means one has to check the feasibility of all the vertices on the new path. In a straightforward implementation, this requires $O(N)$ time for each 2-interchange. This will result in a time complexity of $O(N^3)$ for the verification of 2-optimality. By employing an efficient search strategy, we can reduce the checking effort to $O(1)$ time for each 2-interchange.

The description of the algorithm which follows is based on the second objective extended with the rule specified above. Other objectives require only minor adjustments.

We employ the following *lexicographic search strategy*. We choose the links $(i, i + 1)$ in the order in which they appear in the current tour starting with $(1, 2)$. After fixing a link $(i, i + 1)$ we choose the links $(j, j + 1)$ to be equal to $(i + 2, i + 3)$, $(i + 3, i + 4), \ldots, (N - 1, N)$, in that specific order (see fig. 4). Now consider all

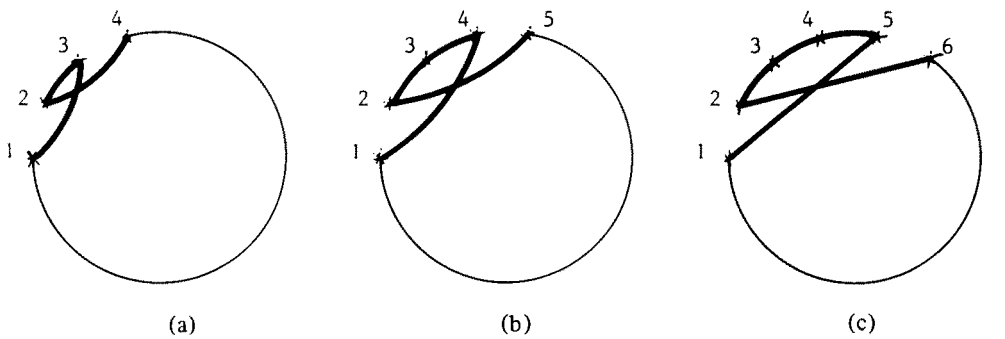

Fig. 4. The lexicographic search strategy.

possible interchanges for a fixed link $(i, i + 1)$. Using the ordering of the 2-interchanges given above implies that in each newly examined 2-interchange, the path $(i + 1, \ldots, j - 1)$ of the previously considered 2-interchange is expanded by the link $(j - 1, j)$. Therefore it is possible, using the information available from the previously considered 2-interchange, to compute the length and to check the feasibility of the path from $(i + 1, \ldots, j)$ in constant time. To accomplish this, we define the following quantities (here and in the sequel, the link appearing as a superscript determines on which interchange the information is based):

$\mathrm{PFS}^{(j,j+1)}$: possible forward shift in time of the departure time at j causing no violation of the time-window constraints along the path $(j, \ldots, i+1)$;

$$\mathrm{PFS}^{(j,j+1)} = \min_{i+1 \leqslant k \leqslant j} \left\{ l_k - (D_j^{(j,j+1)} + \sum_{p=k}^{j-1} t_{p,p+1}) \right\} ;$$

(The theorem below justifies this definition.)

$\mathrm{TWT}^{(j,j+1)}$: total waiting time on the path $(j, \ldots, i+1)$ (excluding possible waiting time at j, including possible waiting time at $i+1$);

$$\mathrm{TWT}^{(j,j+1)} = \sum_{k=i+1}^{j-1} W_k^{(j,j+1)} ;$$

$\mathrm{TTT}^{(j,j+1)}$: total travel time, excluding the periods of waiting, of the path $(j, \ldots, i+1)$;

$$\mathrm{TTT}^{(j,j+1)} = \sum_{k=i+1}^{j-1} t_{k,k+1} ;$$

If we are currently examining the interchange determined by the link $(j, j+1)$, the path $(j-1, \ldots, i)$ of the previously considered interchange is expanded by the link $(j, j-1)$. This usually results in a change of the departure time at $j-1$ [and thus in the change of the departure time of possibly all the other vertices on the path $(j-1, \ldots, i+1)$]. If we define

$$\mathrm{SHIFT}^{(j,j+1)} := D_j^{(j,j+1)} + t_{j,j-1} - D_{j-1}^{(j-1,j)} ,$$

then the following result holds.

THEOREM

Expanding the path $(j-1, \ldots, i+1)$ with the link $(j-1, j)$ is feasible if and only if

$$\mathrm{SHIFT}^{(j,j+1)} \leqslant \mathrm{PFS}^{(j-1,j)} .$$

Proof

[→]

$$\text{feasible} \rightarrow D_j^{(i,j+1)} + \sum_{p=k}^{j-1} t_{p,p+1} \leqslant l_k \qquad i+1 \leqslant k \leqslant j$$

$$\rightarrow D_j^{(i,j+1)} + t_{j,j-1} \leqslant l_k - \sum_{p=k}^{j-2} t_{p,p+1} \qquad i+1 \leqslant k \leqslant j$$

$$\rightarrow \text{SHIFT}^{(i,j+1)} \leqslant \text{PFS}^{(j-1,j)} .$$

[←]

Note that $D_k^{(i,j+1)} \geqslant D_k^{(j-1,j)}$ for $i+1 \leqslant k \leqslant j-1$. The only vertices for which infeasibility can occur are those for which $D_k^{(i,j+1)} \neq D_k^{(j-1,j)}$. A necessary condition for this to occur is that there is no waiting time on the path $(j, \ldots, k)$ after the interchange is carried out. We have that

$$\text{SHIFT}^{(i,j+1)} \leqslant \text{PFS}^{(j-1,j)}$$

$$\rightarrow D_j^{(i,j+1)} + t_{j,j-1} - D_{j-1}^{(j-1,j)} \leqslant l_k - \left(D_{j-1}^{(j-1,j)} + \sum_{p=k}^{j-2} t_{p,p+1} \right)$$

$$\rightarrow D_j^{(i,j+1)} + \sum_{p=k}^{j-1} t_{p,p+1} \leqslant l_k$$

$$\rightarrow D_j^{(i,j+1)} + \sum_{p=k}^{j-1} t_{p,p+1} + \sum_{p=k}^{j-1} W_k^{(i,j+1)} \leqslant l_k$$

$$\rightarrow A_k^{(i,j+1)} \leqslant l_k . \qquad \square$$

Because the triangle inequality holds, traveling directly from i to j takes less time than through $i+1, i+2, \ldots, j-1$, so we do not have to worry about feasibility at j. To test if part of the gain can be carried through to the vertex where the salesman finishes is a trivial matter if we know the vertex with highest index for which the departure time coincides with the earliest service time. This vertex can be determined

in advance. To check the feasibility at $j + 1$ and test for local improvement is also easy because it only requires the exact departure time at vertex $i + 1$ plus local distances, and it is not difficult to see that

$$D_{i+1}^{(j,j+1)} = D_j^{(j,j+1)} + \mathrm{TTT}^{(j,j+1)} + \mathrm{TWT}^{(j,j+1)}.$$

If we take a closer look at the definition of $\mathrm{SHIFT}^{(j,j+1)}$ given above, we see that it covers two different cases (fig. 5):

$\mathrm{SHIFT}^{(j,j+1)} < 0$: Because the triangle inequality guarantees that the new arrival at $j - 1$ is never earlier than the old arrival, it must have been the case that the old arrival and old departure did not coincide. This means that the old departure was equal to the opening of the time window. But then $|\mathrm{SHIFT}^{(j,j+1)}|$ is exactly the waiting time at $j - 1$.

$\mathrm{SHIFT}^{(j,j+1)} \geqslant 0$: Now $\mathrm{SHIFT}^{(j,j+1)}$ is exactly the difference between the new arrival time and the old arrival time at $j - 1$, that is, the forward shift in time.

Updating of the quantities involved takes a constant amount of time. We present the updating formulae below:

$$\mathrm{TTT}^{(j,j+1)} = \mathrm{TTT}^{(j-1,j)} + t_{j,j-1};$$

$$\mathrm{TWT}^{(j,j+1)} = \max(\mathrm{TWT}^{(j-1,j)} - \mathrm{SHIFT}^{(j,j+1)}, 0);$$

$$\mathrm{PFS}^{(j,j+1)} = \min(\mathrm{PFS}^{(j-1,j)} - \mathrm{SHIFT}^{(j,j+1)}, l_j - D_j^{(j,j+1)});$$

It is easily verified that the transformations for $\mathrm{TWT}^{(j,j+1)}$ and $\mathrm{TTT}^{(j,j+1)}$ are correct. The correctness of the transformation for $\mathrm{PFS}^{(j,j+1)}$ can be proved as follows.

Define

$mfs_k^{(j,j+1)}$: maximal forward shift in time of the departure time at j causing no violation of the time window constraints at k;

$$mfs_k^{(j,j+1)} := l_k - \left(D_j^{(j,j+1)} + \sum_{p=k}^{j-1} t_{p,p+1} \right).$$

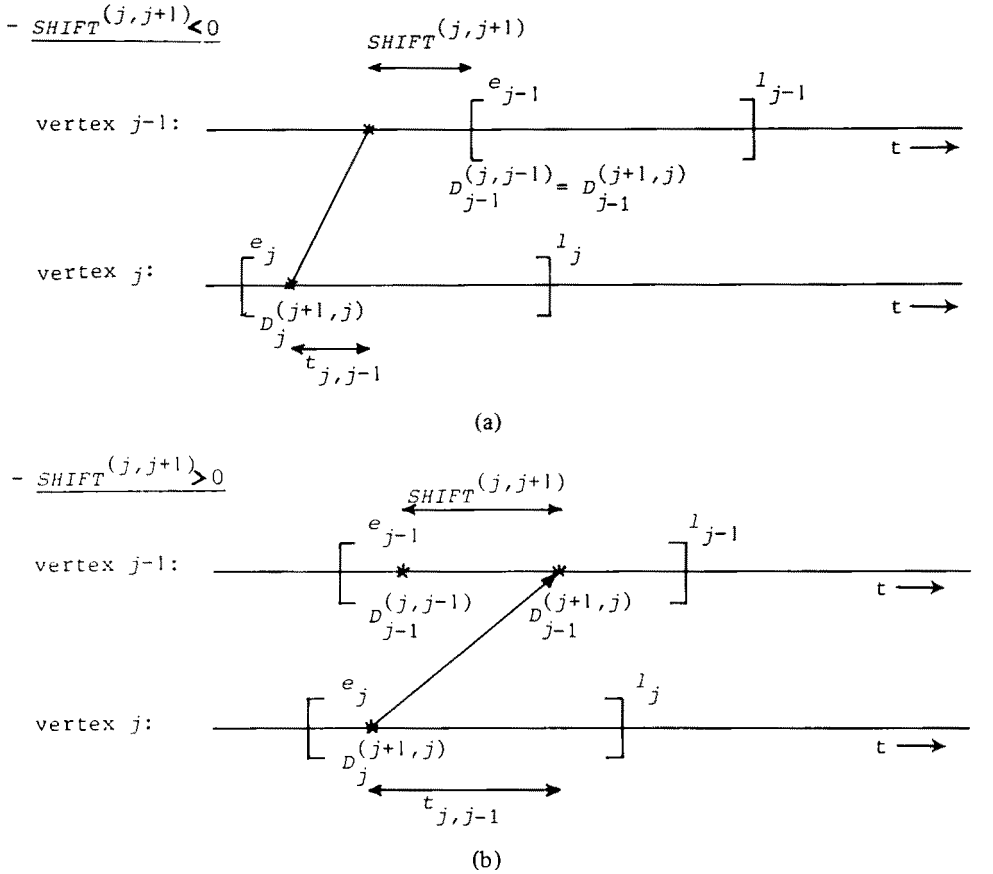

Fig. 5. Schematic presentation of the possible shifts.

We have that

$$mfs_k^{(j,j+1)} = l_k - \left(D_{j-1}^{(j,j+1)} + \sum_{p=k}^{j-1} t_{p,p+1} \right)$$

$$= l_k - \left(D_j^{(j-1,j)} + \sum_{p=k}^{j-2} t_{p,p+1} \right) - D_j^{(j,j+1)} - t_{j,j-1} + D_{j-1}^{(j-1,j)}$$

$$= mfs_k^{(j-1,j)} - \text{SHIFT}^{(j,j+1)}.$$

But that means that

$$\begin{aligned}
\mathrm{PFS}^{(j,j+1)} &= \min_{i+1 \leq k \leq j} \{mfs_k^{(j,j+1)}\} \\
&= \min(l_j - D_j^{(j,j+1)};\ \min_{i+1 \leq k \leq j-1} \{mfs_k^{(j,j+1)}\}) \\
&= \min(l_j - D_j^{(j,j+1)};\ \min_{i+1 \leq k \leq j-1} \{mfs_k^{(j-1,j)} - \mathrm{SHIFT}^{(j,j+1)}\}) \\
&= \min(l_j - D_j^{(j,j+1)};\ \mathrm{PFS}^{(j,j-1)} - \mathrm{SHIFT}^{(j,j+1)}).
\end{aligned}$$

It is easy to see that the time complexity for each individual 2-interchange is reduced to O(1) because the necessary feasibility checks and tests for local improvement plus the updating of all quantities involved require O(1) time. This gives an overall time complexity of $O(N^2)$ for the verification of 2-optimality.

Next, we will consider the Or-interchanges and we start with the case where only one vertex is moved. Because the concepts presented in this part only slightly differ from those described for the 2-interchanges, we will take a more intuitive and informal approach. It is also left to the reader to adjust the formulae for the objectives given for the 2-interchanges in order to make them applicable to the Or-interchanges. It we look at fig. 6, we see that the orientation of the path $(j+1, \ldots, i-1)$ is preserved. This makes it easy to handle the feasibility checks. We also see that there are

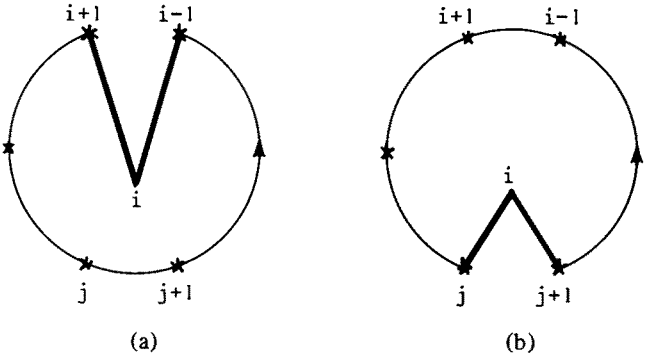

Fig. 6. An Or-interchange where only one vertex is moved.

two possibilities for relocating the vertex i. We can relocate i earlier (backward relocation) or later (forward relocation) in the current tour. Therefore, the search splits into two separate parts, namely a *backward* search and a *forward* search. We order the successively tested Or-interchanges in the same way as we did with the 2-interchanges. An Or-interchange is fully determined by the following quantities:

PROFIT : the gain when going directly from $i-1$ to $i+1$

$$A_{i+1} - (D_{i-1} + t_{i-1,i+1});$$

LOSS : the loss if we use i as an intermediate when going from j to $j+1$

$$\max(D_j + t_{j,i}, e_i) + t_{i,j+1} - A_{j+1};$$

plus for the backward search:

$PFS^{(j,j+1)}$: possible forward shift in time of the departure time at $j+1$ causing no violation of the time window constraints on the path $(j+1, \ldots, i-1)$;

$$PFS^{(j,j+1)} = \min_{j+1 \leq k \leq i-1} \left\{ l_k - \left(D_{j+1} + \sum_{p=j+1}^{k-1} t_{p,p+1} \right) \right\}$$

$TWT^{(j,j+1)}$: total waiting time on the path $(j+1, \ldots, i-1)$;

$$TWT^{(j,j+1)} = \sum_{k=j+1}^{i-1} W_k^{(j,j+1)},$$

plus for the forward search:

$PBS^{(j,j+1)}$: possible backward shift in time of the departure time at $i+1$ causing no extra waiting time on the path from $(i+1, \ldots, j)$;

$$PBS^{(j,j+1)} = \min_{i+1 \leq k \leq j-1} \{ D_k - e_k \}.$$

Note that after fixing the vertex i, we can compute in advance all the values of $PFS^{(j,j+1)}$ by walking backward through the current tour starting at $i-1$ and the values of $PBS^{(j,j+1)}$ by walking forward through the current tour starting at $i+1$. In each step, updating is performed according to the following rules:

$$PFS^{(j,j+1)} = W_{j+1} + \min(l_{j+1} - D_{j+1};\ PFS^{(j+1,j+2)});$$

$$PBS^{(j,j+1)} = \min(D_j - e_j;\ PBS^{(j-1,j)}).$$

The other quantities (PROFIT, LOSS, $TWT^{(j,j+1)}$) are also easy to compute and it is simple to include the necessary feasible check for LOSS during these computations. Now such an Or-interchange during the backward search is feasible if and only if

$$LOSS \leqslant PFS^{(j,j+1)},$$

and profitable (with respect to the partial completion time) if and only if

$$PROFIT > LOSS - TWT^{(j,j+1)},$$

and during the forward search it is feasible and profitable (with respect to the partial completion time) if and only if

$$\min(PBS^{(j,j+1)}, PROFIT) > LOSS.$$

The Or-interchanges where the string of vertices being moved consists of more than one vertex can be treated similarly. Only the computation of LOSS requires some more work. Furthermore, if our objective is just to minimize the time the vehicle is away from the depot, we can decrease the number of tested interchanges. Let i^* be the vertex with highest index for which the earliest service time and departure time coincide. We only have to consider strings of consecutive vertices with higher indices than i^*. A very attractive feature of the Or-interchanges implemented as described above is the fact that checking Or-optimality (moving only strings of a fixed length k) requires $O(N^2)$ time.

The techniques described above for the 2-interchanges and Or-interchanges can also be used to implement the verification of k-optimality (for all possible k) subject to time windows in time $O(N^k)$.

k-interchange procedures are very sensitive for the number and tightness of the time windows. If there are many tight time windows, then the number of tested k-interchanges is greatly reduced because of early detection of infeasibility. Therefore, the number of tight time windows can be used as a decision parameter which invokes a 3-interchange procedure in case this number is large.

4. The initial solution

The local search methods described in the previous section require an initial feasible tour. Finding such a tour is a nontrivial problem since (in contrast to the standard TSP) we have the following theorem.

THEOREM

The problem of finding a feasible tour for the TSPTW is NP-complete in the strong sense, even in the case where the distance matrix is symmetric and satisfies the triangle inequality.

Proof

We will start from the following problem, that is known to be NP-complete in the strong sense [3]:

3-PARTITION

Instance

A finite set $A = \{a_1, a_2, \ldots, a_{3m}\}$ of $3m$ elements, a bound $B \in Z^+$ and a 'size' $s(a_j) \in Z^+$ for each $a_j \in A$, with $B/4 < s(a_j) < B/2$ and

$$\sum_{a_j \in A} s(a_j) = mB .$$

Question

Can A be partitioned into m mutaully disjoint sets $S_1, S_2, \ldots, S_m$ such that, for $1 \leqslant i \leqslant m$

$$\sum_{a_j \in S_i} s(a_j) = B \, ?$$

(Notice that the above constraints on the item size imply that every S_i must contain exactly three elements from A).

Given an instance of 3-PARTITION, we construct the following instance of TSPTW. There are $4m - 1$ vertices, the 'partition' vertices $1, \ldots, 3m$ and the 'splitting' vertices $3m + 1, \ldots, 4m - 1$. Each partition vertex j has a window $[0, mB]$ and a weight $w_j = s(a_j)$ $(j = 1, \ldots, 3m)$. Each splitting vertex j has a window $[(j - 3m)B, (j - 3m)B]$ and a weight $w_j = 0 (j = 3m + 1, \ldots, 4m - 1)$. The distance matrix $T = (t_{ij})$ is now defined in terms of the weights by $t_{ij} = (w_i + w_j)/2$.

Note that the length of any tour is equal to mB and that the windows are defined in such a way that the partition vertices can be visited at any time, while the splitting vertices must be visited at specific points in time that lie B apart. Since the length of a path between two splitting vertices is equal to the sum of the weights associated with the vertices on that path, in any feasible tour the total weight mB is split up in parts of weight B each. This implies that there exists a feasible tour if and only if 3-PARTITION has a solution.

The theorem claims NP-completeness for the special case of the TSPTW in which the distance matrix is symmetric and satisfies the triangle inequality. The matrix T as defined above is symmetric, and if it does not satisfy the triangle inequality, then we add a suitably large number to each vertex weight to ensure that it holds, modify the windows accordingly, and observe that the proof carries through. □

This result justifies the use of a heuristic approach in finding an initial feasible solution. Solomon [12] described and analyzed several heuristics for the VRP with time windows and found that the insertion heuristics performed best. With this in mind, we developed a sequential insertion heuristic based on criteria which include both the spatial and temporal aspects of the problem in searching for an initial feasible solution. The insertion heuristics for the TSP [4] use two criteria, $c_1(i,u,j)$ and $c_2(i,u,j)$ to determine which new customer to insert into the route under construction, where (i, j) is a link of the route and u is yet unrouted. Let $(i_1, i_2, \ldots, i_m)$ be the current route. For each unrouted vertex, we first compute its best insertion place in the emerging route, using the first criterion, as follows:

$$c_1(i_u, u, i_{u+1}) = \min_{p=1,\ldots,m-1} \{c_1(i_p, u, i_{p+1})\}$$

Next, the best vertex u^* to be inserted in the route is selected, using the second criterion, as the one for which:

$$c_2(i_{u^*}, u^*, i_{u^*+1}) = \min_{u \text{ unrouted}} \{c_2(i_u, u, i_{u+1})\}$$

Vertex u^* is then inserted in the route between i_{u^*} and i_{u^*+1}. This procedure continues until all the vertices are routed.

Certain difficulties arise when one wants to apply this heuristic to the TSPTW. Inserting u between i_p and i_{p+1} could affect all the arrival times at vertices i_{p+1}, $i_{p+2}, \ldots, i_m$, which may result in an infeasible tour. This means that we need the quantity

$$\text{PFS} := l_N - \left(D_k + \sum_{p=k}^{N-1} t_{p,p+1}\right),$$

which is similar to the one used in the previous section to check the feasibility of an insertion.

Which criteria to use strongly depends on the tightness of the time windows involved. If a time window is relatively wide, the spatial aspect is more important,

but if a time window is quite tight, then the temporal aspect becomes dominant. Therefore, we introduce two phases: first the vertices with tight time windows are routed and next the vertices with large time windows. (The definition of tight and large can be set according to the user's preference.)

Contrary to Solomon [12] our aim is not, primarily, to minimize a certain objective function, but just to find a feasible solution. This will be apparent from the criteria stated below, where our main concern is to keep the route under construction as flexible as possible.

Note that it is the first criterion used by an insertion heuristic that determines the place where a vertex will be inserted in the emerging route. The second criterion only serves as a guideline to choose between the vertices available for insertion. Therefore, in trying to achieve our goal, creating a feasible tour, we have to rely primarily on the first criterion.

Let us define the 'extra mileage' cost of city u with respect to the link (i, j) by:

$$\mathrm{em}(i,u,j) := \max(D_i + t_{i,u}; e_u) + t_{u,j} - A_j.$$

In phase 1, where the vertices with tight time windows are routed, the temporal aspect is dominant. The criteria $c_1(i_u, u, i_{u+1})$ and $c_2(i_{u^\star}, u^\star, i_{u^\star+1})$ to be used are:

$$c_1(i_u, u, i_{u+1}) = \max_{p=1,\ldots,m-1 \text{ feasible}} \{\min(l_u - \max(D_{i_p} + t_{i_p,u}; e_u);$$

$$\mathrm{PFS}_{i_{p+1}} - \mathrm{em}(i_p, u, i_{p+1})\}$$

$$c_2(i_{u^\star}, u^\star, i_{u^\star+1}) = \min_{u \text{ unrouted}} \{\mathrm{em}(i_u, u, i_{u+1})\}.$$

The first criterion is guided by the remaining flexibility of the route under construction with respect to the time windows, whereas the second criterion searches for the vertex whose inclusion will lead to the smallest increase in length of the tour. In phase 2, where the vertices with large time windows are routed, feasibility problems play a minor role and we can concentrate on the spatial aspect. Therefore, we interchange the criteria $c_1(i_u, u, i_{u+1})$ and $c_2(i_{u^\star}, u^\star, i_{u^\star+1})$ and obtain:

$$c_2(i_u, u, i_{u+1}) = \min_{p = 1, \ldots, m-1 \text{ feasible}} \{\text{em}(i_p, u, i_{p+1})\}$$

$$c_1(i_{u^*}, u^*, i_{u^*+1}) = \max_{u \text{ unrouted}} \{\min(l_u - \max(D_u + t_{i_u, u}; e_u);$$

$$\text{PFS}_{i_u} - \text{em}(i_u, u, i_{u+1})\} \ .$$

5. Computational results

In order to test the computational performance of the described algorithms, we need a set of test problems for the TSPTW. Because no such set is available in the literature, we constructed one ourselves. The problems of this test are based on a well-known TSP instance, introduced by Smith and Thompson [10]. This instance has 48 vertices and coordinates in the interval [1, 2000]. To obtain TSPTW instances, we randomly generated time windows for a subset of customers. The construction is guided by four parameters, namely the percentage of customers which receives a time window (p), the maximum route time (mrt) allowed for the vehicle, and two parameters (α and β) which bound the width of a time window. The test problems are now generated by the following procedure.

(1) Randomly identify customers who will receive a time window until the desired percentage is reached.

(2) For each of these customers j the center of the time window is drawn from a uniform distribution over the interval $[t_{1j}, \text{mrt} - t_{1j}]$, thereby ensuring that the center is reachable from the depot; randomly generate the width of the interval where the width is bounded from below by $\alpha \times$ mrt and from above by $\beta \times$ mrt.

We have embedded the described methods for the 2-interchange and Or-interchanges in three different test programs, the difference between the programs being the acceptance of a feasible interchange. The first is guided primarily by the spatial aspects (objective 1), whereas the second is guided primarily by the temporal aspects of the problem (objective 2). The third is the one we proposed in sect. 3 (objective 2 plus extension). All programs consist of three calls of the Or-interchange procedure (trying to move vertex strings of length three, two, and one, respectively) followed by one call of the 2-interchange procedure.

The programs were written in C and run on the PDP-11/70 computer at the Academic Computer Centre in Amsterdam (SARA). The run times listed in the tables are those provided by the UNIX profiling facility.

Table 1

Average results for the initial tour and the three test programs

	Total time	Waiting time	Travel time
Initial tour	16288.6	865.4	15423.2
Program I	15428.2	1658.7	13771.3
Program II	14488.0	220.2	14267.8
Program III	14531.7	558.6	13937.1

Table 2

Average number of tested and performed interchanges for the Or-interchange algorithm with strings of three, two, and one consecutive vertices being relocated, respectively

	Or-opt (3)		Or-opt (2)		Or-opt (1)		Time Or-opt (ms)
	No. test	No. perf	No. test	No. perf	No. test	No. perf	
Program I	1960.3	1.2	2891.0	3.2	2636.2	2.7	327.26
Program II	3670.0	4.5	7136.0	7.2	10405.1	13.3	720.88
Program III	3370.7	4.0	5228.0	5.2	7842.4	9.8	612.86

Table 3

Average number of tested and performed interchanges for the 2-interchange algorithm

	2-opt		Time 2-opt (ms)
	No. test	No. perf	
Program I	1068.8	4.9	106.68
Program II	926.5	2.7	80.03
Program III	863.4	2.9	76.67

The results given in tables 1, 2 and 3 are based on ten randomly generated problems with parameters mrt, α, β and p set to 15, 000, 0.025, 0.200 and 25, respectively. Travel times between vertices were taken to be equal to the corresponding Euclidean distance. Table 1 clearly illustrates the influence of the chosen objective on the interchange algortihms. Table 2 and 3 provide some additional information on the performance of the algorithm. We see that the choice of objective has some impact on the computational effort to reach and verify Or-optimality and that the number of actually performed interchanges is small compared to the number of tested interchanges. Very attractive are the computational times, which are less than one second for a combination of one call of the 2-interchange procedure and three calls of the Or-interchange procedure.

6. Conclusions

We have described two techniques that enable us to modify the k-interchange concept for local improvement in routing problems in such a way that time windows can be handled without increasing the time complexity. We also proved that the problem of deciding whether or not there exists a feasible solution for the TSP with time windows is NP-complete in the strong sense and with this in mind, constructed an insertion heuristic which tries to combine the spatial and temporal aspects of the problem. The computational results obtained when the described techniques were implemented together with the initial tour heuristic are very satisfactory. The growing importance of time windows in practical distribution problems is an encouragement for fundamental research in this area. Hopefully, this paper will contribute to its exploitation.

Acknowledgements

The author would like to express his thanks to Ben Lageweg and Jan Karel Lenstra for their stimulating comments. This research was supported by the Netherlands Foundation for the Technical Sciences (STW) through Grant No. CWI-22.0348.

References

[1] N. Christofides, A. Mingozzi and P. Toth, Space state relaxation procedures for the computation of bounds to routing problems, Networks 11(1981)145.

[2] A. Croes, A method for solving traveling salesman problems, Oper. Res. 5(1958)791.

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

[4] P.C. Kanellakis and C.H. Papadimitriou, Local search for the asymmetric traveling salesman problem, Oper. Res. 28(1980)1086.

[5] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan and D.B. Shmoys (eds.), *The Traveling Salesman Problem* (Wiley, Chichester, 1985).

[6] S. Lin, Computer solutions to the traveling salesman problem, Bell System Tech. J. 44(1965) 2245.

[7] S. Lin and B.W. Kernighan, An effective heuristic algorithm for the traveling salesman problem, Oper. Res. 21(1973)498.

[8] I. Or, *Traveling Salesman-type Combinatorial Problems and Their Relation to the Logistics of Blood Banking*, Ph.D. Thesis, Dept. of Industrial Engineering and Management Sciences, Northwestern University (1976).

[9] C.H. Papadimitriou and K. Steiglitz, Some examples of difficult traveling salesman problems, Oper. Res. 26(1978)434.

[10] H.N. Psaraftis, *k*-interchange procedures for local search in a precedence-constrained routing problem, Eur. J. Oper. Res. 13(1983)391.

[11] T.H.C. Smith and G.L. Thompson, A LIFO implicit enumeration search algorithm for the symmetric traveling salesman problem using Held and Karp's 1-tree relaxarion, Ann. Discrete Math. 1(1977)479.

[12] M.M. Solomon, Vehicle routing and scheduling with time window constraints: Models and algorithms, Working Paper 83-02-01, Dept. of Decision Sciences, University of Pennsylvania (1983).

[13] M.M. Solomon, On the worst-case performance of some heuristics for the vehicle routing and scheduling problem with time window constraints, Working Paper 83-05-03, Dept. of Decision Sciences, University of Pennsylvania (1983).