Centrum voor Wiskunde en Informatica
Centre for Mathematics and Computer Science

M.W.P. Savelsbergh

Local search for constrained routing problems

Department of Operations Research and System Theory Report OS-R8711 July

The Centre for Mathematics and Computer Science is a research institute of the Stichting Mathematisch Centrum, which was founded on February 11, 1946, as a nonprofit institution aiming at the promotion of mathematics, computer science, and their applications. It is sponsored by the Dutch Government through the Netherlands Organization for the Advancement of Pure Research (Z.W.O.).

Local Search for Constrained Routing Problems

Martin W.P. Savelsbergh
Department of Operations Research and System Theory,
Centre for Mathematics and Computer Science, Amsterdam

We develop local search algorithms for routing problems with various side constraints such as time windows on vertices and precedence relations between vertices. The algorithms are based on the k-exchange concept. The presence of side constraints introduces feasibility problems. Checking the feasibility of a given solution in the straight forward way requires time which is linear in the number of vertices. Our method reduces this effort to constant time.

1980 Mathematics Subject Classification: 90B05, 90B35, 90C27.
Key Words & Phrases: traveling salesman problem, vehicle routing problem, local search, iterative improvement.

1. Introduction

Croes [1958] and Lin [1965] were the first to introduce the notion of k-exchanges to improve solutions of the *traveling salesman problem* (TSP): Croes for $k=2$ and Lin for the general case. Several papers have since been written that examine issues related to the application of this method. Lin and Kernighan [1973] generalized it and Papadimitriou and Steiglitz [1978] reported results on its worst case behavior. The method has also been applied with considerable success to other classes of problems. Kanellakis and Papadimitriou [1980] adapted the method for the asymmetric TSP and Psaraftis [1983] examined the use of k-exchanges in precedence constrained routing problems.

In this paper a local search method based on the k-exchange concept will be presented that handles various types of side constraints without increasing the time complexity. 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 to other types of routing problems as well.

The approach we will follow draws from the k-exchange procedure of Lin [1965] for the TSP. A k-exchange is a substitution of k links of a tour with k other links. The introduction of side constraints, such as time windows on vertices and precedence relations between vertices, makes it necessary to check the feasibility of an exchange. In contrast to the standard TSP where the processing of a single k-exchange takes constant time, testing in a straightforward way whether a single exchange does not violate the side constraints requires $O(n)$ time, where n is the number of vertices. We will develop a method that performs this feasibility test in constant time. The key idea of the method is the use of an appropriate search strategy and an appropriate set of global variables.

Our main motivation for the development of k-exchange methods that can handle side constraints efficiently is a quite practical one. We need such algorithms in CAR (Computer Aided Routing), a software package for distribution management [Anthonisse, Lenstra, Savelsbergh 1987]. CAR employs a 'cluster-first route-second' method, the second phase of which asks for the solution of a number of traveling salesman problems. In many real-life situations, routing algorithms must be capable of handling various types of side constraints. Frequently occurring side constraints are:

- time windows of customers;
- both collections and deliveries at customers;

Report OS-R8711
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

- precedence relations between customers;
- fixed paths.

In an earlier paper [Savelsbergh 1985] we described a local search method based on the k-exchange concept that handles time windows of customers efficiently. In this paper we will present extensions that enable us to handle the other side constraints mentioned above as well.

2. Local Search for the standard TSP

In the TSP [Lawler et al 1985], we are given a complete graph on a set V of vertices and a travel time $t_{i,j}$ for each arc $(i,j) \in V \times V$. A solution to the TSP is a tour, i.e., a cycle which visits each vertex exactly once. The duration of a tour is the sum of the travel times of the arcs contained in it. The objective is to find a tour of minimum duration. Let $n = |V|$ indicate the number of vertices. We assume that a given vertex, say vertex 1, will serve as the first and last vertex of any tour (the depot in vehicle routing and scheduling problems) and that the matrix (t_{ij}) is symmetric and satisfies the triangle inequality, i.e.,

$$t_{ij} = t_{ji} \qquad \text{for all } i,j \in V,$$
$$t_{ik} \leq t_{ij} + t_{jk} \qquad \text{for all } i,j,k \in V.$$

Recall that a k-exchange 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 shorter tour by replacing k of its links by another set of k links. Since the computational effort to construct a k-optimal tour rises rapidly with k, we consider only the cases $k=2$ and $k=3$.

We start with the case $k=2$. Performing a single 2-exchange on a tour involves the substitution of two of its links, say $(i,i+1)$ and $(j,j+1)$, with two other links (i,j) and $(i+1,j+1)$ (see Figure 1).

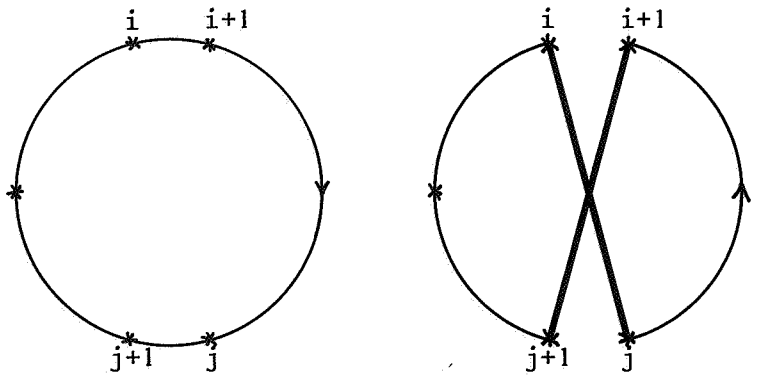

Figure 1: A 2-exchange.

Such an exchange results in a local tour improvement if and only if

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

Notice that the orientation of the path $(i+1,\ldots,j)$ is reversed in the new tour. The total number of possible 2-exchanges is equal to 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 several ways of substituting a given triple of links with another triple of links. Figures 2 shows two possible 3-exchanges that can be performed by deleting the links $(i,i+1)$, $(j,j+1)$ and $(k,k+1)$ of a TSP tour.

For all cases conditions similar to the one given for the case $k=2$ can be given to obtain local tour

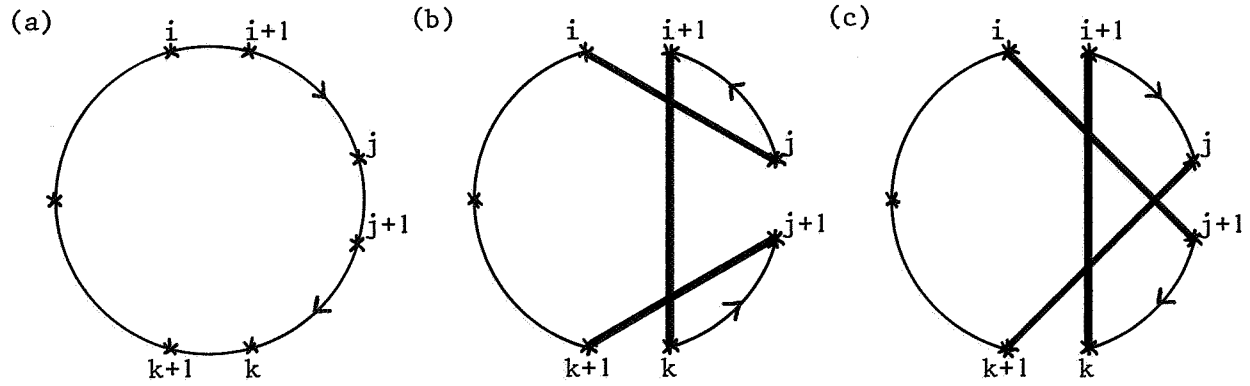

Figure 2: Two ways to perform a 3-exchange.

improvement. There is one important difference between the two 3-exchanges shown above: in the latter the orientation of the paths $(i+1,...,j)$ and $(j+1,...,k)$ is preserved whereas in the former this orientation is reversed. The total number of possible 3-exchanges 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 made to take only a subset of all possible 3-exchanges into account. We will consider the proposal by Or [1976]. His procedure considers only those 3-exchanges 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 Figure 3. In this tour the path $(i_1,...,i_2)$ is relocated between j and $j+1$. Two important observations can be made if we look at Figure 3:

- The orientation of the path $(j+1,...,i_1-1)$ is preserved, which makes it easier to handle the feasibility checks;
- There are two possibilities for relocating the string of vertices; we can either relocate it earlier (backward relocation) or later (forward relocation) in the current tour with respect to the starting point of the tour.

Due to the second observation we split the verification procedure in two separate parts, a *backward* search and a *forward* search. The time complexity to verify Or-optimality is $O(n^2)$.

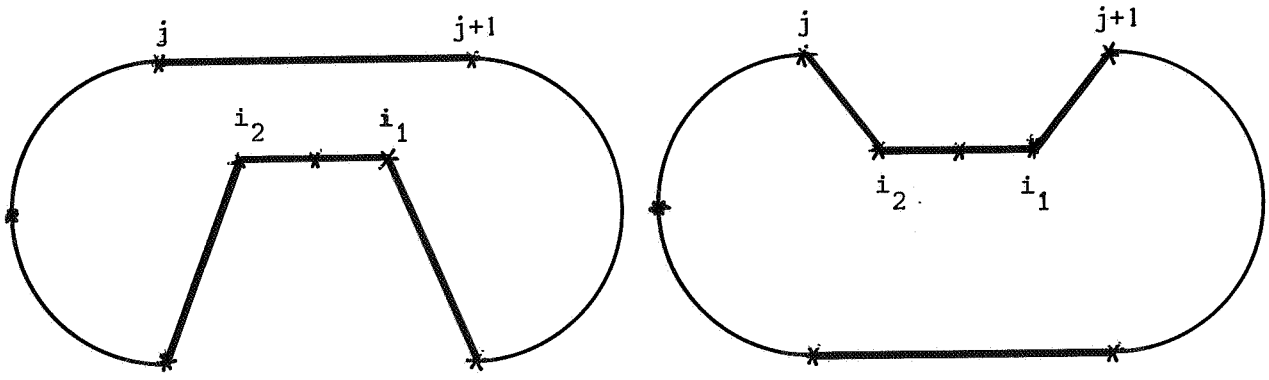

Figure 3: An Or-exchange.

3. A LEXICOGRAPHIC SEARCH STRATEGY

The main problem with the use of k-exchange procedures in the TSP with side constraints is checking the feasibility of an exchange. A 2-exchange, for instance, will reverse the path $(i+1,...,j)$, which means that one has to check the feasibility of all the vertices on the new path with respect to those constraints. In a straightforward implementation this requires $O(n)$ time for each 2-exchange, which results in a time complexity of $O(n^3)$ for the verification of 2-optimality. The basis of the proposed local search methods is the use of a search strategy and a number of global variables such that for each considered exchange, checking its feasibility and updating the global variables require no more than constant time. Because the search strategy is of crucial importance, we present it first.

In the sequel, we will assume that the current tour, for which we want to prove optimality, is given by a sequence $(1,2,...,i,...,n)$, where i represents the ith vertex of the tour, and that we are always examining the exchange that involves the substitution of links $(i,i+1)$ and $(j,j+1)$ with (i,j) and $(i+1,j+1)$ in case of a 2-exchange, and the substitution of (i_1-1,i_1), (i_2,i_2+1) and $(j,j+1)$ with (i_1-1,i_2+1), (j,i_1) and $(i_2,j+1)$ in case of an Or-exchange.

3.1. LEXICOGRAPHIC SEARCH FOR 2-EXCHANGES

We choose the links $(i,i+1)$ in the order in which they appear in the current tour starting with (1,2); this will be referred to as the outer loop. After fixing a link $(i,i+1)$, we choose the link $(j,j+1)$ to be $(i+2,i+3),(i+3,i+4),...,(n-1,n)$ in that order (see Figure 4); this will be referred to as the inner loop.

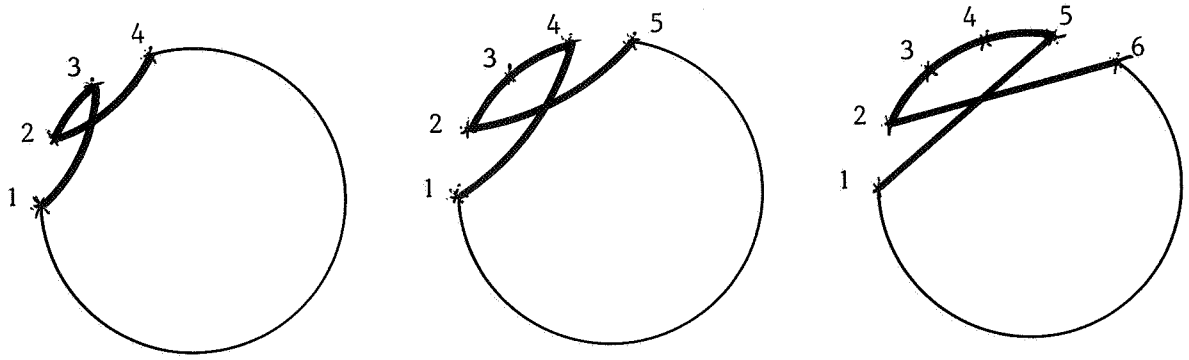

Figure 4: The lexicographic search strategy for 2-exchanges.

Now consider all possible exchanges for a fixed link $(i,i+1)$. The ordering of the 2-exchanges given above implies that in the inner loop in each newly examined 2-exchange the path $(i+1,...,j-1)$ of the previously considered 2-exchange is expanded by the link $(j-1,j)$. This observation, together with an appropriate set of global variables, makes it is possible to maintain information on the feasibility and duration of this path efficiently, i.e., check its feasibility and to update the global variables in constant time.

3.2. LEXICOGRAPHIC SEARCH FOR BACKWARD OR-EXCHANGES

We choose the path $(i_1,...,i_2)$ in the order of the current tour starting with i_1 equal to 3. After the path $(i_1,...,i_2)$ is fixed, we choose the link $(j,j+1)$ to be $(i_1-2,i_1-1),(i_1-3,i_1-2),...,(1,2)$ in that order. That is, the link $(j,j+1)$ 'walks backward' through the tour. Note that in the inner loop in each newly examined exchange the path $(j+2,...,i_1-1)$ of the previously considered exchange is expanded with the link $(j+1,j+2)$.

3.3. Lexicographic search for forward Or-exchanges

We choose the path $(i_1,...,i_2)$ in the order of the current tour starting with i_1 equal to 2. After the path $(i_1,...,i_2)$ is fixed, we choose the link $(j,j+1)$ to be $(i_2+1,i_2+2),(i_2+2,i_2+3),...,(n-1,n)$ in that order. That is, the link $(j,j+1)$ 'walks forward' through the tour. Note that in each newly examined exchange the path $(i_2+1,...,j-1)$ of the previously considered exchange is expanded with the link $(j-1,j)$.

4. The traveling salesman problem with mixed collections and deliveries

Suppose that in addition to the travel time $t_{i,j}$ for each pair of vertices $i,j \in V$ and the specified vertex that serves as first and last vertex of the tour, each vertex has an associated load q_i that has to be either collected or delivered at that vertex. The salesman uses a vehicle with fixed capacity Q and his objective is to serve all vertices while minimizing the duration.

The following quantities will be very helpful for the description of the algorithm. Let Γ be the set of addresses where the salesman has to make a collection and Δ the set of addresses where the salesman has to make a delivery. Given a feasible tour $(1,2,...,n)$, we define

$$C(r,s) := \sum_{k\in\{r,...,s\},k\in\Gamma} q_k, \quad D(r,s) := \sum_{k\in\{r,...,s\},k\in\Delta} q_k.$$

A tour is feasible if and only if

$$0 \leqslant C(1,k)-D(1,k) \leqslant Q \qquad \text{for } k = 1,...,n.$$

An important variant of the TSP with mixed collections and deliveries arises when it is required that all load to be delivered has to be collected at vertex 1, and all load to be collected has to be delivered at vertex 1. In that case a route is feasible if and only if

$$C(1,k)-D(1,k) \leqslant Q \qquad \text{for } k = 1,...,n.$$

2-Exchanges

Consider the 2-exchange where the links $(i,i+1)$ and $(j,j+1)$ are replaced by the links (i,j) and $(i+1,j+1)$. In the following, a quantity with superscript *new* indicates the value after the exchange has been carried out, and arguments always refer to the ordering of the current tour. If the exchange would be carried out the quantities that determine feasibility can be expressed in terms of the quantities of the current tour as follows:

$$\begin{aligned}
C^{new}(1,k) &= C(1,k) && \text{for } 1\leqslant k\leqslant i,\ j+1\leqslant k\leqslant n,\\
D^{new}(1,k) &= D(1,k) && \text{for } 1\leqslant k\leqslant i,\ j+1\leqslant k\leqslant n,\\
C^{new}(1,k) &= C(1,i) + C(k,j) && \text{for } i+1\leqslant k\leqslant j,\\
D^{new}(1,k) &= D(1,i) + D(k,j) && \text{for } i+1\leqslant k\leqslant j.
\end{aligned}$$

The exchange is feasible if and only if

$$0 \leqslant C(1,i) - D(1,i) + \min_{k\in\{i+1,...,j\}}\{C(k,j) - D(k,j)\},$$

$$C(1,i) - D(1,i) + \max_{k\in\{i+1,...,j\}}\{C(k,j) - D(k,j)\} \leqslant Q.$$

If we introduce global variables for $C(1,i) - D(1,i)$, $\min_{k\in\{i+1,...,j\}}\{C(k,j) - D(k,j)\}$ and $\max_{k\in\{i+1,...,j\}}\{C(k,j) - D(k,j)\}$, checking the feasibility of an exchange reduces to two additions and two comparisons, which take constant time. The lexicographic search strategy allows us to maintain the global variables efficiently, i.e., to update them for each new value of j in constant time.

Backward Or-exchange

Consider the backward Or-exchange, where the path $(i_1,...,i_2)$ is relocated backward between j and $j+1$. We find that

$$C^{new}(1,k) = C(1,k) \qquad \text{for } 1\leqslant k\leqslant j,\ i_2+1\leqslant k\leqslant n,$$

$$
\begin{aligned}
D^{new}(1,k) &= D(1,k) && \text{for } 1 \leqslant k \leqslant j,\ i_2+1 \leqslant k \leqslant n,\\
C^{new}(1,k) &= C(1,k) + C(i_1,i_2) && \text{for } j+1 \leqslant k \leqslant i_1-1,\\
D^{new}(1,k) &= D(1,k) - D(i_1,i_2) && \text{for } j+1 \leqslant k \leqslant i_1-1,\\
C^{new}(1,k) &= C(1,k) - C(j+1,i_1-1) && \text{for } i_1 \leqslant k \leqslant i_2,\\
D^{new}(1,k) &= D(1,k) + D(j+1,i_1-1) && \text{for } i_1 \leqslant k \leqslant i_2.
\end{aligned}
$$

The exchange is feasible if and only if

$$
\begin{aligned}
&0 \leqslant \min_{k \in \{j+1,\ldots,i_1-1\}} \{C(1,k) - D(1,k)\} + C(i_1,i_2) - D(i_1,i_2),\\
&\max_{k \in \{j+1,\ldots,i_1-1\}} \{C(1,k) - D(1,k)\} + C(i_1,i_2) - D(i_1,i_2) \leqslant Q,\\
&0 \leqslant \min_{k \in \{i_1,\ldots,i_2\}} \{C(1,k) - D(1,k)\} + C(j+1,i_1-1) - D(j+1,i_1-1),\\
&\max_{k \in \{i_1,\ldots,i_2\}} \{C(1,k) - D(1,k)\} + C(j+1,i_1-1) - D(j+1,i_1-1) \leqslant Q.
\end{aligned}
$$

We rewrite this as

$$
\begin{aligned}
&D(i_1,i_2) - C(i_1,i_2) \leqslant \min_{k \in \{j+1,\ldots,i_1-1\}} \{C(1,k) - D(1,k)\},\\
&C(i_1,i_2) - D(i_1,i_2) \leqslant Q - \max_{k \in \{j+1,\ldots,i_1-1\}} \{C(1,k) - D(1,k)\},\\
&D(j+1,i_1-1) - C(j+1,i_1-1) \leqslant \min_{k \in \{i_1,\ldots,i_2\}} \{C(1,k) - D(1,k)\},\\
&C(j+1,i_1-1) - D(j+1,i_1-1) \leqslant Q - \max_{k \in \{i_1,\ldots,i_2\}} \{C(1,k) - D(1,k)\}.
\end{aligned}
$$

We have now accomplished our goal: we can introduce global variables for $D(i_1,i_2) - C(i_1,i_2)$, $\min_{k \in \{i_1,\ldots,i_2\}} \{C(1,k) - D(1,k)\}$ and $\max_{k \in \{i_1,\ldots,i_2\}} \{C(1,k) - D(1,k)\}$ that can be maintained efficiently in the outer loop and global variables for $D(j+1,i_1-1) - C(j+1,i_1-1)$, $\min_{k \in \{j+1,\ldots,i_1-1\}} \{C(1,k) - D(1,k)\}$ and $\max_{k \in \{j+1,\ldots,i_1-1\}} \{C(1,k) - D(1,k)\}$ that can be maintained efficiently in the inner loop.

Forward Or-exchange

Consider the forward Or-exchange, where the path $(i_1,\ldots,i_2)$ is relocated forward between j and $j+1$. Analogously to the backward Or-exchange, we find that

$$
\begin{aligned}
C^{new}(1,k) &= C(1,k) && \text{for } 1 \leqslant k \leqslant i_1-1,\ j+1 \leqslant k \leqslant n,\\
D^{new}(1,k) &= D(1,k) && \text{for } 1 \leqslant k \leqslant i_1-1,\ j+1 \leqslant k \leqslant n,\\
C^{new}(1,k) &= C(1,k) - C(i_1,i_2) && \text{for } i_2+1 \leqslant k \leqslant j,\\
D^{new}(1,k) &= D(1,k) + D(i_1,i_2) && \text{for } i_2+1 \leqslant k \leqslant j,\\
C^{new}(1,k) &= C(1,k) - C(i_2+1,j) && \text{for } i_1 \leqslant k \leqslant i_2,\\
D^{new}(1,k) &= D(1,k) + D(i_2+1,j) && \text{for } i_1 \leqslant k \leqslant i_2.
\end{aligned}
$$

The exchange is feasible if and only if

$$
\begin{aligned}
&0 \leqslant \min_{k \in \{i_2+1,\ldots,j\}} \{C(1,k) - D(1,k)\} - C(i_1,i_2) + D(i_1,i_2),\\
&\max_{k \in \{i_2+1,\ldots,j\}} \{C(1,k) - D(1,k)\} - C(i_1,i_2) + D(i_1,i_2) \leqslant Q,\\
&0 \leqslant \min_{k \in \{i_1,\ldots,i_2\}} \{C(1,k) - D(1,k)\} + C(i_2+1,j) - D(i_2+1,j),\\
&\max_{k \in \{i_1,\ldots,i_2\}} \{C(1,k) - D(1,k)\} + C(i_2+1,j) - D(i_2+1,j) \leqslant Q.
\end{aligned}
$$

We rewrite this as

$$
\begin{aligned}
&C(i_1,i_2) - D(i_1,i_2) \leqslant \min_{k \in \{i_2+1,\ldots,j\}} \{C(1,k) - D(1,k)\},\\
&D(i_1,i_2) - C(i_1,i_2) \leqslant Q - \max_{k \in \{i_2+1,\ldots,j\}} \{C(1,k) - D(1,k)\},\\
&D(i_2+1,j) - C(i_2+1,j) \leqslant \min_{k \in \{i_1,\ldots,i_2\}} \{C(1,k) - D(1,k)\},
\end{aligned}
$$

$$C(i_2+1,j) - D(i_2+1,j) \leqslant Q - \max_{k\in\{i_1,\ldots,i_2\}}\{C(1,k) - D(1,k)\}.$$

As in the backward case, we can now easily define global variables that can be efficiently maintained in both loops.

Initial solutions

In order to apply the k-exchange improvement algorithms described above, we need to have an initial feasible tour. In the special case where all load to be delivered has to be collected at vertex 1 and all load to be collected has to be delivered at vertex 1, there exits a feasible tour if and only if it is feasible to visit all delivery vertices before all collection vertices. This strategy is known in vehicle routing problems as *back-hauling*. In the general case, the existence problem is more difficult. We will show that the problem of determining whether there exists a feasible tour for the TSP with mixed collections and deliveries is NP-complete in the strong sense.

Our proof starts from the following problem, which is known to be NP-complete in the strong sense [Garey, Johnson 1979]:

3-PARTITION

Instance: A finite set A of $3m$ elements, a bound $B\in Z^+$ and a 'size' $s(a)\in Z^+$ for each $a\in A$, with $B/4<s(a)<B/2$ and

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

Question: Can A be partitioned into m mutually disjoint sets $S_1, S_2, \cdots, S_m$ such that, for $1\leqslant i\leqslant m$,

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

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

Given an instance of 3-PARTITION, we construct the following instance of the TSP with mixed collections and deliveries. There are $3m$ delivery vertices with load equal to $s(a_j)$, m collection vertices with load equal to B and vertex 1 with load equal to 0 which can either be a delivery or collection. The salesman has a vehicle with capacity B. Note that in a feasible solution a collection can only be made if the vehicle is empty. This implies that a feasible solution will consist of m subsequences, each consisting of a collection followed by three deliveries. But such a solution exists if and only if 3-PARTITION has a solution.

In addition, it is easy to see that the problem of determining whether there exists a feasible solution to the TSP with mixed collections and deliveries is a member of NP: a non-deterministic algorithm need only guess an ordering of the vertices and check in polynomial time whether it is feasible.

5. The traveling salesman problem with precedence constraints

Suppose that we are given in addition to the travel time $t_{i,j}$ for each pair of vertices $i,j\in V$ and the specified vertex at which the tour starts and finishes, precedence constraints specifying that some pairs of vertices have to be visited in a prescribed order. We assume that the precedence-related pairs are pairwise disjoint. This situation occurs for instance in the *static single vehicle dial-a-ride problem* where a single vehicle has to pick up and deliver n customers. Each customer has a pick up and delivery location and the pick up must precede the delivery. Psaraftis [1983] shows that the k-exchange improvement methods can be modified to handle these restrictions. By a straightforward choice of the set of global variables, the lexicographic search strategy produces the same result.

To describe the precedence relations, we attach a label to each vertex containing the following information:

$$prec(v) = \begin{cases} u & \text{if vertex } v \text{ must precede vertex } u, \\ -u & \text{if vertex } u \text{ must precede vertex } v, \\ 0 & \text{if vertex } v \text{ has no precedence relation with other vertices.} \end{cases}$$

Feasibility checking can now be accomplished by a simple marking mechanism based on these labels and an appropriate set of global variables. In the following, when we refer to a 'successor' or 'predecessor' of a vertex, we will always mean its uniquely defined precedence-related successor or precedence-related predecessor, and not a successor or predecessor determined by the current ordering of the tour.

2-Exchanges
A 2-exchange is feasible if and only if there is no pair of precedence-related vertices on the path $(i+1,...,j)$:

$$v \in (i+1,...,j) \Rightarrow |prec(v)| \notin (i+1,...,j).$$

We associate a global variable $mark(v)$ with each vertex $v \in V$, as follows:

$$mark(v) = \begin{cases} 1 & \text{if } prec(v)>0 \wedge |prec(v)| \in (i+1,...,j-1), \\ 0 & \text{otherwise.} \end{cases}$$

With these global variables, the feasibility of exchanges can be checked in constant time. Whenever we try to expand the path $(i+1,...,j-1)$ with the link $(j-1,j)$ and $mark(j)=1$, vertex j has a predecessor that is already on the path, which implies that expansion of the path will result in infeasible exchanges. What remains is to show that we can maintain these global variables efficiently. Again, the lexicographic search strategy provides a simple way to accomplish this. In the inner loop, whenever we expand the path $(i+1,...,j-1)$ with the link $(j-1,j)$, we test if vertex j has a successor, and if so we set the variable associated with this successor to 1:
- If $prec(j)>0$, then $mark(prec(j)):=1$.

Now note that if the inner loop is terminated because a marked vertex is encountered, it is very well possible that there are other marked vertices on the path $(j+1,...,n)$. Fortunately, we do not have to reset all marked vertices on the path $(j+1,...,n)$ but just the successor, if any, of vertex i, because this is the only global variable that is no longer valid. This introduces one additional action in the outer loop:
- If $prec(i)>0$, then $mark(prec(i)):=0$.

Backward Or-exchange
A backward Or-exchange is feasible if and only if there is no pair of precedence-related vertices with one of them on the path $(i_1,...,i_2)$ and the other on the path $(j+1,...,i_1-1)$:

$$v \in (i_1,...,i_2) \Rightarrow |prec(v)| \notin (j+1,...,i_1-1).$$

We associate a global variable $mark(v)$ with each vertex $v \in V$:

$$mark(v) = \begin{cases} 1 & \text{if } prec(v)>0 \wedge prec(v) \in (i_1,...,i_2), \\ 0 & \text{otherwise.} \end{cases}$$

Whenever we try to expand the path $(j+1,...,i_1-1)$ with the link $(j,j+1)$ and $mark(j)=1$, its successor is on the path $(i_1,...,i_2)$, thus implying that expansion will only result in infeasible exchanges. For the backward Or-exchanges the actual marking and resetting can both be controlled in the outer loop:
- If $mark(i_1)<0$, then $mark(|prec(i_1)|):=1$.
- If $mark(i_2+1)<0$, then $mark(|prec(i_2+1)|):=0$.

Forward Or-exchange
A forward Or-exchange is feasible if and only if there is no pair of precedence-related vertices with one of them on the path $(i_1,...,i_2)$ and the other on the path $(i_2+1,...,j)$:

$$v \in (i_1,...,i_2) \Rightarrow |prec(v)| \notin (i_2+1,...,j).$$

We associate a global variable $mark(v)$ with each vertex $v \in V$:

$$mark(v) = \begin{cases} 1 & \text{if } prec(v)<0 \wedge |prec(v)| \in (i_1,...,i_2), \\ 0 & \text{otherwise.} \end{cases}$$

Whenever we try to expand the path $(i_2+1,...,j)$ with the link $(j,j+1)$ and vertex $j+1$ is marked, its predecessor is on the path $(i_1,...,i_2)$, thus implying that expansion will only result in infeasible exchanges. The actual marking and resetting of global variables is peformed in the outer loop:
- If $mark(i_2)>0$, then $mark(prec(i_2)):=1$.
- If $mark(i_1-1)>0$, then $mark(prec(i_1-1)):=0$.

6. The traveling salesman problem with fixed paths

In many applications of the k-exchange improvement algorithms in vehicle routing, and especially in interactive planning situations, it is useful to be able to specify parts of the tour that may not be separated. One way of doing this is attaching a label to each vertex in the following way:

$$link(v) = \begin{cases} 1 & \text{if vertex } v \text{ may not be seperated from its immediate predecessor,} \\ 0 & \text{otherwise.} \end{cases}$$

If we try to modify the k-exchange methods to handle the fixed path restrictions, we find ourselves in the unique situation where we do not need global variables at all. Feasibility can be checked as follows.

2-Exchange

A 2-exchange is feasible if and only if

$$link(i+1) = 0 \wedge link(j+1) = 0.$$

Or-exchange

An Or-exchange, backward and forward, is feasible if and only if

$$link(i_1) = 0 \wedge link(i_2+1) = 0 \wedge link(j+1) = 0.$$

7. Conclusion

We have described a search strategy that enables us to modify the 2-exchange and Or-exchange algorithms for local tour improvement in the TSP in such a way that various side constraints can be handled without increasing the time complexity. More generally, our techniques can be used to verify k-optimality subject to these side constraints, for each value of k. The growing importance of side constraints in practical distribution planning is an encouragement for fundamental research in this area. This paper is intended as a contribution in this direction.

8. References

J.M. Anthonisse, J.K. Lenstra and M.W.P. Savelsbergh (1987). *Functionele Beschrijving van CAR, een Systeem for 'Computer Aided Routing'.* Notitie OS-N8701, Centrum voor Wiskunde en Informatica, Amsterdam.

A. Croes (1958). A method for solving traveling salesman problems. *Oper. Res. 5,* 791-812.

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

P.C. Kanellakis and C.H. Papadimitriou (1980). Local search for the asymmetric traveling salesman problem. *Oper. Res. 28,* 1086-1099.

E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan and D.B. Shmoys (eds.) (1985). *The Traveling Salesman Problem: a Guided Tour of Combinatorial Optiization,* Wiley, Chichester.

S. Lin (1965). Computer solutions to the traveling salesman problem. *Bell System Tech. J. 44,* 2245-2269.

S. Lin and B.W. Kernighan (1973). An effective heuristic algorithm for the traveling salesman problem. *Oper. Res. 21,* 498-516.

I. OR (1976). *Traveling Salesman-Type Combinatorial Problems and their Relation to the Logistics of Blood Banking,* Ph.D. thesis, Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, IL.

C.H. PAPADIMITRIOU AND K. STEIGLITZ (1978). Some examples of difficult traveling salesman problems. *Oper. Res. 26,* 434-443.

H.N. PSARAFTIS (1983). *k*-Interchange procedures for local search in a precedence-constrained routing problem. *European J. Oper. Res. 13,* 391-402.

M.W.P. SAVELSBERGH (1985). Local search for routing problems with time windows. *Ann. Oper. Res. 4,* 285-305.