Annals of Operations Research 41(1993)469–488

Solving real-life vehicle routing problems efficiently using tabu search

Frédéric Semet and Eric Taillard
École Polytechnique Fédérale de Lausanne, Département de Mathématiques, Chaire de Recherche Opérationnelle, CH-1015 Lausanne, Switzerland

Abstract

This paper presents a tabu search based method for finding good solutions to a real-life vehicle routing problem. The problem considered deals with some new features beyond those normally associated with the classical problems of the literature: in addition to capacity constraints for vehicles and time windows for deliveries, it takes the heterogeneous character of the fleet into account, in the sense that utilization costs are vehicle-dependent and that some accessibility restrictions have to be fulfilled. It also deals with the use of trailers. In spite of the intricacy of the problem, the proposed tabu search approach is easy to implement and can be easily adapted to many other applications. An emphasis is placed on means that have to be used to speed up the search. In a few minutes of computation on a personal workstation, our approach obtains solutions that are significantly better than those previously developed and implemented in practice.

Keywords: Combinatorial optimization, vehicle routing problem, tabu search.

1. Introduction

Generally speaking, solving real-life problems differs considerably from solving academic ones, in particular because of the amount and variety of restrictions to consider. This occurs for the vehicle routing problem (VRP) at a regional level of one of the major Swiss chain stores. A simplified version of the problem faced by this company is as follows.

The problem involves 45 grocery stores located in the cantons of Vaud and Valais in Switzerland (see fig. 1). As each store may place up to two different orders (one concerning the bakery service and the other concerning other types of goods), the problem deals with 70 to 90 orders. An order of a store can not be split, so it may be entirely loaded on any vehicle, but the sum of the volumes and weights of two orders of a same store may be bigger than the capacities of some vehicles.

To stock its shops, the company owns a heterogeneous fleet of vehicles consisting of 21 trucks and 7 trailers. A truck is characterized by its carrying capacity, by its volume capacity, by its cost of use per kilometer, and by

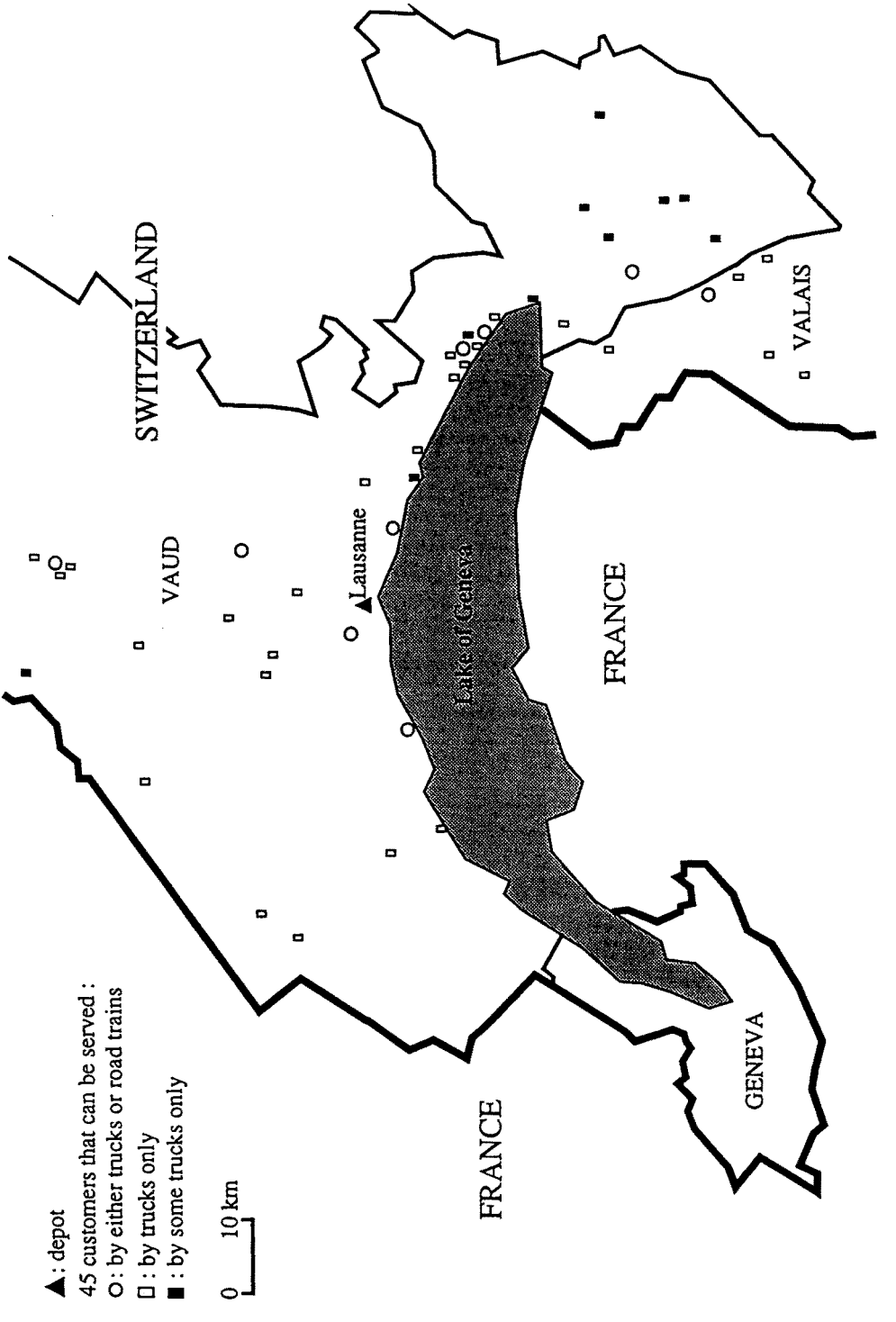

Fig. 1. Distribution network.

the subset (eventually empty) of trailers that it can draw. A trailer is characterized by its carrying capacity, by its volume capacity, by its cost of use per kilometer, and implicitly by the subset of the truck that can draw it. For a trailer, the cost of use per kilometer includes the additional consumption of fuel of the truck drawing it.

When a truck draws a trailer, it composes what we call a *road train.* The volumetric capacity of a road train is the sum of the volumetric capacities of the truck and the trailer. Since the carrying capacity of a road train is limited by Swiss law to a maximal value (28 tons), the carrying capacity of a road train is the minimum of this value and the sum of the carrying capacities of the truck and the trailer.

A store is characterized by a subset of trucks that can reach it, by whether it is equipped for receiving deliveries by road train, by its access time (manœuvring time) by truck and if applicable, by its access time by road train. From now on, we call a store that can receive deliveries by road train a *trailer-store* and call a store that can receive deliveries by truck only a *truck-store.* A trailer-store can receive deliveries either by road train or by any truck alone. Figure 1 shows the situation geographically and the main characteristics of the stores.

An order is characterized by the store that has placed it, by its volume, by its weight, by an unloading time and by a time window during which it must receive deliveries.

Each vehicle may cover one route at most. A route covered by a truck alone is a classical circuit: the truck starts from the depot, delivers every order it has loaded and goes back to the depot. The routes including a trailer have a special character: the road train leaves the depot to reach a trailer-store where the trailer may be uncoupled, so that the truck alone can then serve some truck-store; we will call a route covered by a truck alone that starts from a trailer-store and that comes back to the same trailer-store a *subtour.* While the truck covers a subtour, the trailer is unloaded. Finally, the truck goes back to the trailer-store where the trailer is coupled again, and the road train moves to the next trailer-store, and so on (see fig. 2). Notice that for such routes, the trailer contains the orders of trailer-stores only, but the truck is loaded either with orders for trailer-stores or for truck-stores. In our case, there are 9 trailer-stores (see fig. 1). Obviously, such shops may be served by some routes covered by a truck alone.

Therefore, our problem may be sketched as follows: Determine a transportation schedule using a heterogeneous fleet consisting of trailers and trucks that minimizes the transportation costs and satisfies the following constraints:

- Each order must be delivered during a time window.
- The volume capacity for each vehicle must be respected.
- The carrying capacity for each vehicle must be respected.
- Each store can be reached by a subset of the trucks and trailers.

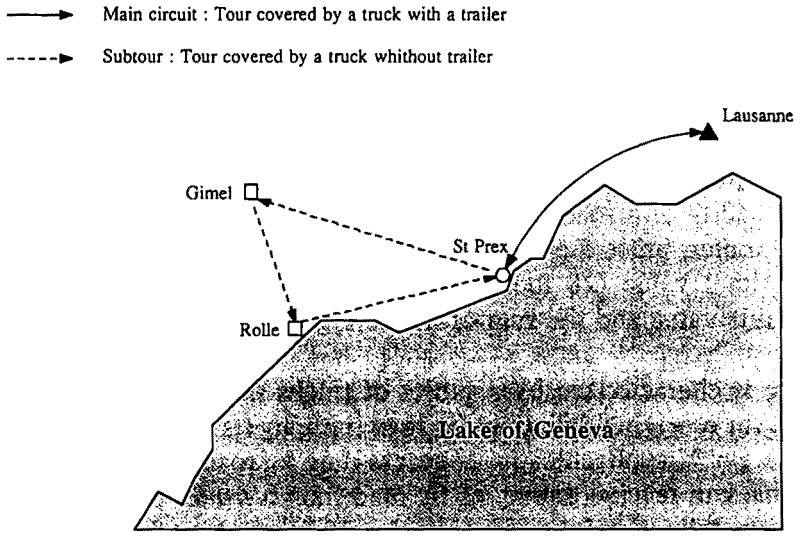

Fig. 2. An example of a road train-route.

In this paper, we present a very flexible approach based on tabu search techniques (TS) to find good solutions to this problem. In section 2, we summarize the first heuristic that we developed to solve the problem and that is now used by the company. Our adaptation of TS is described in section 3. We show how this search can be speeded up in section 4. Numerical results are compared with the solutions used by the company in section 5. Finally, we conclude in section 6 with some general remarks on this work and future research that may be undertaken.

2. First heuristic method based on clustering methods

The problem we are dealing with contains significantly more features than a standard VRP, and hence no classical method is able to handle it. We first sketch the method the company uses for finding good transportation plans. The heuristic method (see [7]) is based on that of Fisher and Jaikumar [3] and on clustering methods. It provides feasible solutions for a VRP that incorporates the same constraints described above except for time window restrictions. It embodies the three following steps:

(1) *Elaboration of trailer-routes*:

Solve the VRP restricted to trailer-stores and to trailers; this provides routes including only trailer-stores from which subtours may be generated in step 2 (see the main circuit in fig. 2).

(2) *Formation of road trains*:

For each trailer-store, build a cluster including the nearest truck-stores. Assign trucks to trailers considering the volumes that may be delivered on subtours.

(3) *Elaboration of truck-routes*:

Solve the VRP on truck-stores not yet delivered, using the remaining trucks.

We now state some important points of these steps. In step 1, the total capacities of trailers (volume and weight) may be insufficient to serve every trailer-store, so the fleet is completed by adding some trucks that are then assigned to trailers. In this phase as well as in step 3, we solve the VRP using the two-phase heuristic method of Fisher and Jaikumar [3]. Indeed, this method may be adapted to deal with two capacity restrictions of a heterogeneous fleet and provides good solutions.

Some key points relative to the implementation of the Fisher and Jaikumar method have to be stated precisely. The first phase consists of assigning orders to vehicles by solving a generalized assignment problem (GAP). The coefficients c_{ik} of the GAP objective function represent the cost of delivering order i with vehicle k. They are computed as follows:

$$c_{ik} = d_{0i} + d_{ii_k} - d_{0i_k},$$

where i_k is the seed order (see [3]) associated with vehicle k and d_{ij} is the distance between the stores where orders i and j have to be delivered. Note that order i does not represent a store since stores may place different orders to the central depot 0. We set c_{ik} to infinity when order i can not be delivered by vehicle k due to the VRP constraints.

To select seed orders, two heuristic criteria are successively used. First, orders for which the volume (respectively, the weight) is greater than the half volumetric capacity (respectively, the carrying capacity) of the larger vehicle are chosen as seed orders. When the number of such orders is less than the number of vehicles, the following criterion is applied. The farthest order from the previously selected seed orders is selected as seed order. This is repeated until the desired number of seed orders is reached.

In step 2 (formation of road trains), clusters are built around trailer-stores, assigning first the nearest truck-stores. Then, some exchanges of truck-stores between clusters are performed based on an index of inter-cluster proximity. This is due to the capacity constraints of the largest truck that can access the trailer-store. The proximity index D_{AB} between two clusters C_A and C_B is defined as follows:

$$D_{AB} = \frac{2}{|C_A||C_B|}\Delta_{AB} - \frac{1}{|C_A|^2}\Delta_{AA} - \frac{1}{|C_B|^2}\Delta_{BB},$$

with

$$\Delta_{AB} = \sum_{i \in C_A} \sum_{j \in C_B} d_{ij}.$$

This type of index between clusters, introduced by Tricot and Donegani [11], can be interpreted as follows. Δ_{AB} measures the separation between clusters A and B, when Δ_{AA} measures the dispersion of cluster A. Thus, D_{AB} is greater as the clusters C_A and C_B are more compact (i.e. Δ_{AA}, Δ_{BB} are smaller) and vice versa. As a particular case, the proximity index between one truck-store α and one cluster C_A is calculated according to the following formula:

$$\frac{1}{|C_A|} \sum_{i \in C_A} d_{\alpha i} - \frac{1}{|C_A|^2} \Delta_{AA}.$$

Given a cluster C_A violating a capacity restriction, the exchanges consist of permuting n truck-stores of C_A for m truck-stores of C_B. Clusters C_B satisfying the capacity constraints are considered according to increasing values of D_{AB}. Different n–m exchanges are tried between C_A and C_B: 1–0, 1–1, 1–2, 2–1, 2–2. Finally, clusters are compacted to keep a set of stores that are close to each other.

At the present time (May 1992), this method is used by the company and has produced solutions significantly superior to those obtained "by hand". The solutions from this method allow us to evaluate the quality of solutions obtained by TS. We specify below our tabu search approach to this problem.

3. Tabu search and VRP

TS first requires us to define the notion of move applicable to a given solution. For the VRP, a move that transfers an order from one route to another is convenient. We denote by r_0 the fictitious route that contains every non-delivered order. Associated with this route, we introduce a cost to ensure $r_0 = \emptyset$ for an admissible solution of the problem. Thus, we use a constraint relaxation technique that allows some orders not to be delivered on certain intermediate steps, but that takes the constraint $r_0 = \emptyset$ into the objective function with a penalty attached to its violation, controlled in a manner that also causes feasible solutions to be generated.

3.1. EVALUATION OF THE COST OF A MOVE

A central aspect of our approach is the implementation of two fast procedures, one for removing an order from a route and one for inserting an order into a route. Both routines have to be able to deal with variants of the TSP with time windows. Since the number of stores included in a route is small in our case (less than 10), we take advantage of heuristic processes that are simple and convenient to implement. For this, we extend fundamental heuristic notions devised for the standard VRP to

allow them to handle the more complex requirements of our formulation. The capacity and accessibility restrictions are not taken into account in this part of the method, but they are considered in the assignment of vehicles to routes (see section 3.3).

The insertion procedure first consists of determining the best place to locate an order on a given route, and then of improving the route using a 2-optimality routine (see [5]). The insertion phase constitutes an extension of the work of Solomon [8]. We briefly summarize the basis for this phase as follows.

Let $(o_{i_0}, o_{i_1} \ldots o_{i_{n+1}})$ be the orders belonging to the route r_i with o_{i_0} and $o_{i_{n+1}}$ constituting fictitious orders that stand for the depot (i.e. $o_{i_0} \equiv o_{i_{n+1}}$). Let o_u denote the order to be inserted. For each feasible insertion of o_u between o_{i_j} and $o_{i_{j+1}}$, we then compute:

- *the increase in the length of the route*:

$$c_1(i_j, u) = d_{i_j u} + d_{u i_{j+1}} - d_{i_j i_{j+1}} \geq 0;$$

- *the delay on arrival at the store that places* $o_{i_{j+1}}$:

$$c_2(i_j, u) = b^u_{i_{j+1}} - b_{i_{j+1}},$$

where $b_{i_{j+1}}$ is the delivering time of $o_{i_{j+1}}$, i.e. the time at which order $o_{i_{j+1}}$ begins being unloaded, and $b^u_{i_{j+1}}$ is the new delivery time if o_u is inserted.

Then the best place to insert o_u in the route, between $o_{i_{j(u)}}$ and $o_{i_{j(u)+1}}$, is determined to minimize the convex combination of the two preceding criteria:

$$\alpha_1 c_1(i, j(u)) + \alpha_2 c_2(i, j(u)) = \min_{p=0 \ldots n} (\alpha_1 c_1(i_p, u) + \alpha_2 c_2(i_p, u)),$$

with $\alpha_1 + \alpha_2 = 1$, $\alpha_1, \alpha_2 \geq 0$. We choose $\alpha_1 = \alpha_2 = 0.5$.

In this way, a good position is determined for o_u on the route in the case o_u is delivered by a truck. In the case of road train-routes, inserting order o_u for a trailer-store needs another computation of c_1 because we enforce the trailer-stores to be served by a road train to deliver a maximum number of truck-stores by truck. The computation of c_2 remains unchanged. We have to consider four cases when o_u is an order for a trailer-store:

Case 1: o_{i_j} *and* $o_{i_{j+1}}$ *are orders for trailer-stores.*

This case is the classical one, so computation of c_1 and c_2 are not modified.

Case 2: o_{i_j} *is an order for a trailer-store,* $o_{i_{j+1}}$ *is an order for a truck-store.*

Let o_{i_k} be the order preceding o_{i_j} on the subtour and o_{i_l} the order following o_{i_j} on the main circuit (see fig. 3); then we compute:

$$c_1(i_j, u) = d_{i_j u} + d_{u i_{j+1}} - d_{i_j i_{j+1}} + (d_{i_k u} - d_{i_k i_j}) + (d_{u i_l} - d_{i_j i_l}),$$

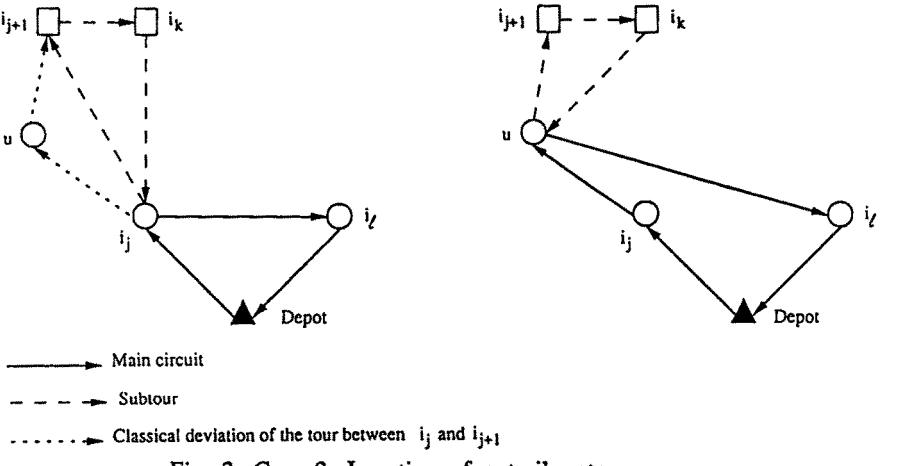

Fig. 3. Case 2: Insertion of a trailer-store between another trailer-store and a truck-store.

which can be written:

$$c_1(i_j, u) = d_{i_j u} + d_{u i_l} - d_{i_j i_l} + (d_{u i_{j+1}} - d_{i_j i_{j+1}}) + (d_{i_k u} - d_{i_k i_j}).$$

This insertion is equivalent to inserting o_u between o_{i_j} and o_{i_l} on the main circuit and to exchanging the origin of the subtour initially issued from o_{i_j}; o_u then becomes the new root of the subtour.

Case 3: o_{i_j} *is an order for a truck-store,* $o_{i_{j+1}}$ *is an order for a trailer-store.*

This case is symmetrical to the previous one. o_u is the predecessor of $o_{i_{j+1}}$ on the main circuit and becomes the new origin of the subtour previously issued from $o_{i_{j+1}}$ (see fig. 4).

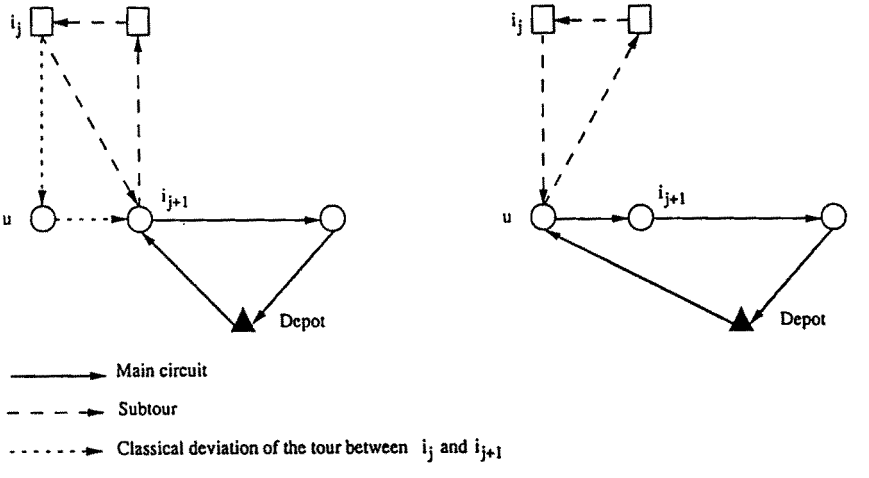

Fig. 4. Case 3: Insertion of a trailer-store between a truck-store and a trailer-store.

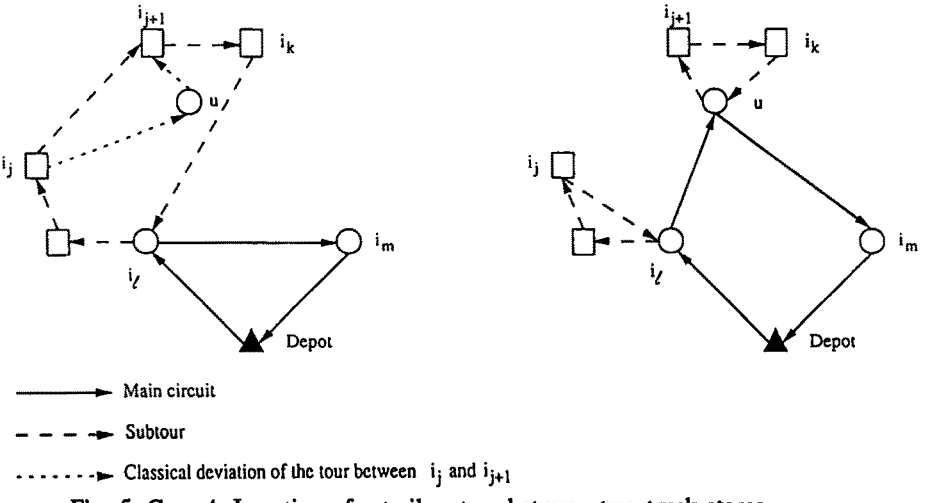

Fig. 5. Case 4: Insertion of a trailer-store between two truck-stores.

Case 4: o_{i_j} *and* $o_{i_{j+1}}$ *are orders for truck-stores.*

Let o_{i_l} be the origin of the subtour including o_{i_j} and $o_{i_{j+1}}$, let o_{i_k} be the predecessor of o_{i_l} on the subtour, and let o_{i_m} be its successor on the main circuit (see fig. 5). Then we compute:

$$c_1(i_j, u) = d_{i_l u} + d_{u i_m} - d_{i_l i_m} + (d_{i_j i_l} + d_{u i_{j+1}} - d_{i_j i_{j+1}}) + (d_{i_k u} - d_{i_k i_l}).$$

This insertion is equivalent to inserting o_u between o_{i_l} and o_{i_m} and to splitting the subtour issued from o_{i_l}. The splitting occurs by deleting the trip from the store where o_{i_j} is delivered to the store that places order $o_{i_{j+1}}$.

The removal procedure, which complements the foregoing insertion procedure, consists of removing the order considered from the route and updating the parameters of the route. The deletion of an order that is the origin of a subtour is the only critical case. The operation in fact makes use of the insertion processes previously described. Every order on the subtour is inserted on the other subtours of the same route using the insertion routine identified above. However, if there is no other trailer-store served by the route, the latter becomes a truck-route.

3.2. TABU LISTS

The next component of our TS method is the definition of the tabu list. Two types of tabu lists were tested: one based on forbidding orders to be assigned to particular routes, and the other based on forbidding stores to receive deliveries by particular routes.

The first tabu list is defined as follows: If order o_j currently belongs to route r_k and is selected to be inserted on another route, the tabu list forbids order o_j to be moved back to route r_k again during a given number s of iterations. This may

be implemented by storing for each order o_j and for each route r_i the number of the iteration after which o_j may again be delivered on route r_i.

The second tabu list type is particularly relevant to our problem where each store places many orders: every order that originates from a given store is forbidden to be assigned back to a route from which any one of these orders was just removed.

Finally, we use the commonly applied aspiration function that releases the tabu state of a move if the move improves the best solution found up to now.

3.3. TABU SEARCH PROCEDURE

Figure 6 illustrates the procedure of performing a move in pseudo-code. The only point that has not been clarified yet is the assignment of vehicles to routes. If there were no trailers, this point would be clear. Then, we would simply compute a square matrix (a_{ij}) that gives the cost of assigning vehicle v_j to route r_i. Clearly,

```
Procedure perform_a_move
  kept_cost := ∞
  for every order o_j do
      remove o_j from route r_k to which it belongs
      for every route r_i ≠ r_k do
        insert o_j on r_i
        find an assignment of vehicles to routes minimizing the total cost
        if (there is an assignment satisfying the restrictions of the problem)
           and (the move is not tabu or is aspirated)
           and (the cost of this move is lower than kept_cost) then
              kept_cost := cost of tried_move
              kept_move := tried_move
        end if
      end for
  end for
  if kept_cost < ∞ then
     perform kept_move;
     update solution, tabu list, assignment ...
  end if
end perform_a_move
```

Fig. 6. Procedure performing a move.

if vehicle v_j can not be assigned to route r_i, either a store can not be served by v_j, or another restriction (volume or carrying capacities) is violated, and a_{ij} has to be set to infinity.

In our problem, a truck can draw only one type of trailer. Assuming that the number of trailers is not limited, we may solve a linear assignment problem. Thus, if the total volume or the total weight exceeds the corresponding capacity of truck

v_j, and if there is a store that can receive deliveries by a trailer on this route, a_{ij} is the delivery cost of route r_i using vehicle v_j plus its associated trailer. However, the optimal assignment may use more trailers than available. To simplify, we reject a priori such solutions even if a finite cost assignment using a feasible number of trailers exists.

The complexity of one step of TS (one execution of the procedure "perform a move") is $O(n^2 + nm^4)$, where m is the number of trucks and n is the number of orders (in our case, $m = 21$ and $70 \leq n \leq 90$). Indeed, inserting an order on a route may be performed in $O(n/m)$ time using a simple insertion procedure (routes are assumed to be balanced) and the assignment of vehicles to routes may be computed in $O(m^3)$ using the Hungarian algorithm (see [6]).

As described above, one step of TS takes a few minutes on a personal workstation that has a computing power of 10 Mips. Since 2000 to 10,000 steps are necessary to find good solutions, such a time is prohibitive. In the next section, we identify a way to reduce computation time, essentially by using data already computed in previous steps.

4. Speeding up TS

CPU time can be dramatically reduced by three strategies: (1) employing an assignment routine designed more appropriately for our problem and invoking it only when it yields the most benefit; (2) recording move evaluations and using a streamlined updating process; (3) employing a screening criterion to reduce the size of the neighbourhood. (1) and (3) imply that an intensification search procedure has to be invoked when the search reaches a solution that satisfies appropriate conditions.

4.1. ASSIGNMENT

A profiler program discloses that over 90% of the CPU time of the unaccelerated version of the method is spent in finding the optimal assignment of vehicles to routes. Therefore, the assignment step requires special attention to devise a more efficient procedure. Upon examining optimal assignments successively obtained, we found that very few changes occur from one assignment to the next. The Hungarian algorithm starts from an empty assignment and builds an optimal one by adding one truck after the other. Therefore, this algorithm is not well suited to our purpose because it does not profit from knowledge of the optimal assignment of the previous step. Thus, we make use of a primal transportation method that progressively improves feasible assignments until reaching one that is optimal (see [1]). Its complexity is $O(m^4)$, but it runs five to seven times faster than the Hungarian algorithm in our application since each initial solution is nearly optimal for the problem of the following stage.

However, the resulting version of TS still spends more than 70% of its CPU time in this procedure, and hence we undertake to reduce the number of invocations

of the assignment procedure. The solution adopted consists simply of computing an optimal assignment only when a move is effectively performed, instead of each time a move is evaluated as a candidate for selection. The quality of the evaluation can be diminished by this change, and we show how to handle this consideration in section 4.3.

4.2. STORING INFORMATION TO ENABLE RESTRICTED UPDATES

By the preceding approach, an assignment solution is generated once instead of $n(m-1)$ times at each iteration, and this change causes the time spent to rebuild the routes during the move trials to become preponderant (consuming 70% of the CPU time). By an analogous approach, we seek to call the insertion and removal procedures as few times as possible, in order to reduce the $O(nm)$ times per iteration that are otherwise invoked. This can be realized by storing the values:

$$l_{ij} = \begin{cases} o_j \notin r_i : \text{length of route } r_i \text{ if order } o_j \text{ is inserted on this route,} \\ o_j \in r_i : \text{length of route } r_i \text{ if order } o_j \text{ is removed from this route.} \end{cases}$$

At each iteration, the only values l_{ij} that must be updated are those for which the composition of route r_i has changed since the previous iteration (there are two such routes). Exploiting this fact, the complexity of one TS step decreases from $O(n^2 + nm^4)$ to $O(m^2n + n^2/m)$, independently from the time improvement gained by altering the assignment solution process.

4.3. INTENSIFICATION OF THE SEARCH

Since the optimal assignment is not computed for each evaluation of a trial move, the information underlying this evaluation may be misleading. Indeed, if the vehicles are very loaded (as typically occurs for good solutions), the evaluation obtained by the accelerated process may fail to disclose that an improving move exists, whereas the assignment solution would directly identify this possibility. For this reason, each time a "good solution" is obtained by the accelerated procedure, we amend the procedure to perform a small number of iterations, during which an optimal assignment is computed at each trial move to determine a true local optimum. We must specify what a "good solution" means:

Let $\hat{s}_k$ be the best solution obtained by the accelerated search up to iteration k and let s_k be the current solution: if the cost of s_k is lower than the cost of $\hat{s}_{k-1}$ (then $\hat{s}_k = s_k$), s_k is a "good solution". Under this circumstance, we search the true local optimum associated with $\hat{s}_k$ by performing a "short" TS (independent of the accelerated one) which computes an optimal assignment for each trial move. We choose to stop this intensive search if more than five iterations are performed without improving the best solution obtained by this search. For this process, we fix the tabu list size equal to 3.

This modification generally yields a best overall solution that is superior to the one obtained by the unmodified form of the accelerated procedure.

4.4. SIZE OF THE NEIGHBOURHOOD

A usual strategy for decreasing computation time is to reduce the size of the neighbourhood, but this often impairs the quality of the solutions obtained. In the present case, we find that a proper choice of the neighbourhood size (with appropriate coordination) does not suffer this defect, but considerably cuts down the time spent to perform an iteration. Rather than inspecting the neighbourhood completely ($n(m-1)$ moves), we inspect approximately a quarter of it ($\lfloor n/4 \rfloor (m-1)$ moves). More precisely, we try to perform a route change for only a quarter of orders, examining at each iteration a different set of orders. At iteration k, the orders considered are:

$$o_{(((k-1)\lfloor n/4 \rfloor) \bmod n)+1}, o_{(((k-1)\lfloor n/4 \rfloor+1) \bmod n)+1} \cdots o_{((k\lfloor n/4 \rfloor-1) \bmod n)+1}.$$

Paradoxically, we sometimes obtained better solutions using this partial neighbourhood examination. This may be explained by the diversity of moves generated by this inspection. Indeed, if no improving move is available for a given solution, neighbourhood restrictions may lead to a move that will destroy the structure associated with the current solution. For the first tabu list type, fig. 7 shows the value of the

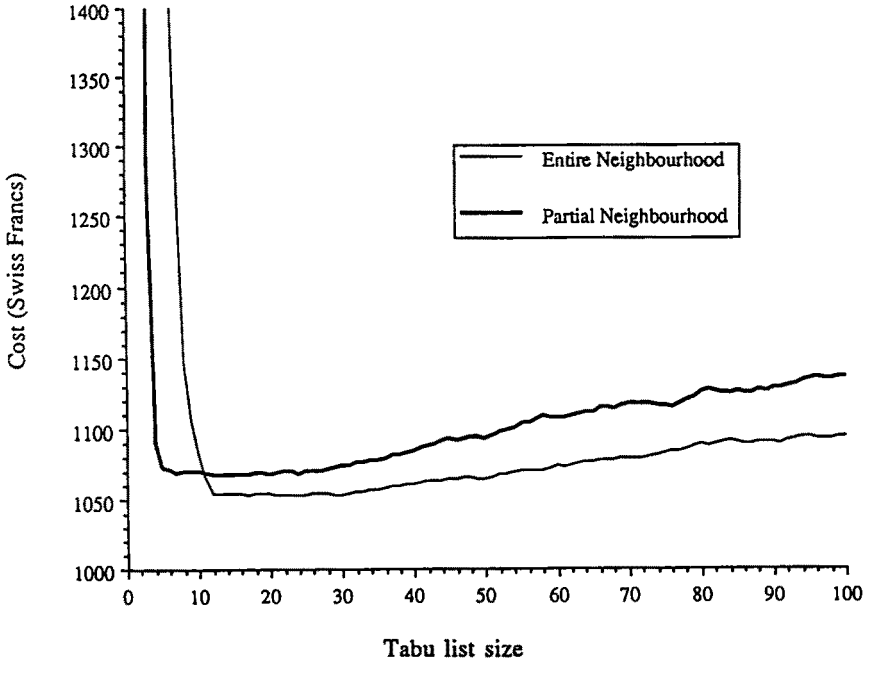

Fig. 7. Partial versus complete examination of the neighbourhood.

objective function for both variants (for a given problem and after 10,000 TS steps) as a function of the tabu list size, because this parameter still has to be determined.

It turns out that a longest tabu list size is needed when the neighbourhood is not reduced. This may be explained by a lack of diversity. Moreover, for an optimal tabu list size, the version with a reduced neighbourhood provides slightly worse solutions (2%) but the CPU time reductions are very important here. From now on, we refer to the version of the method that uses a partial neighbourhood examination.

4.5. AGGREGATION OF ORDERS

An ultimate way to cut down CPU time is to compress the size of the problem: multiple orders for a given store can be aggregated into a unique order when the total volume is lower than the capacity of the smallest truck that may access the store. This aggregation may hinder the discovery of an optimal solution of our problem, but the CPU time and the size of the solution space are considerably reduced, making it possible to obtain good solutions in shorter CPU time.

We tested this by re-solving the previous problem (which was used as a basis for determining parameter settings for all problems subsequently examined), this time aggregating orders. For the first tabu list type, fig. 8 shows the value of the solution obtained as a function of tabu list size for the initial and reduced problem. We make the following observations: first, the optimal tabu list size for the reduced

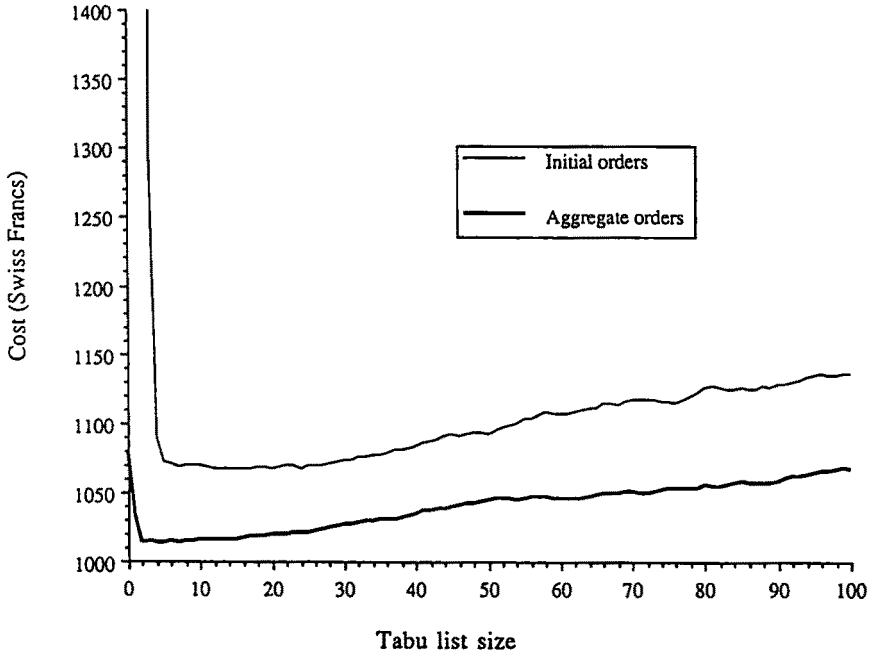

Fig. 8. Effect of the aggregation of the orders.

problem (for which the number of orders is about half the original one) is slightly lower, but not significantly so. Second, for the reduced problem, the optimal objective function values are 5% lower than those of the original problem for optimal tabu list sizes. The incidence of the aggregation of orders is more extensively studied in the next section.

Using all the improvements described above, we were able to perform 1000 iterations of TS in only a few minutes on our personal workstation. By comparison, the initial version of the program required several days to perform this number of iterations!

Our TS procedure can now be summarized as follows:

- Search for the best allowed move over a subset of the neighbourhood that is systematically varied, keeping the assignment of trucks to routes of the preceding iteration.
- Perform this best move.
- Compute the new optimal assignment.
- If a good solution is obtained, find its associated local optimum by performing some steps of TS that compute an optimal assignment at each trial of move.

In the next section, we present more extensive results (for problems occurring each working day) that justify the values of the various parameters selected on the basis of studying the original benchmark problem. The results are also compared with those provided by the heuristic method described earlier that the company has used up to now.

5. Numerical results

First, we study the behaviour of both tabu list types described in the preceding section. On the one hand, the tabu list forbids an order to be reassigned to a route it just left, and on the other hand, the tabu list forbids every order of a store to be reassigned to a route that any one of these orders just left. The two types of lists differ more significantly when the orders are not aggregated. Therefore, we additionally examine a new problem (different from those of figs. 7 and 8) in which the orders are not aggregated. For both tabu list types, we plot (in fig. 9) the solution found as a function of the tabu list size. It turns out that the second type of list provides slightly better and more regular results than the first one.

For the new problem, the optimal tabu list size is greater than the one obtained for the problem treated in section 4. This leads us to examine the choice of tabu list size from the standpoint of developing a general rule for its determination.

Cycling occurs when the size of the list is too small: the same solutions are ultimately visited again and again, thus wastefully reducing the efficiency of the procedure. Conversely, when the tabu list size is too large, some moves that may lead to good solutions will be forbidden. We examine more precisely the behaviour of the

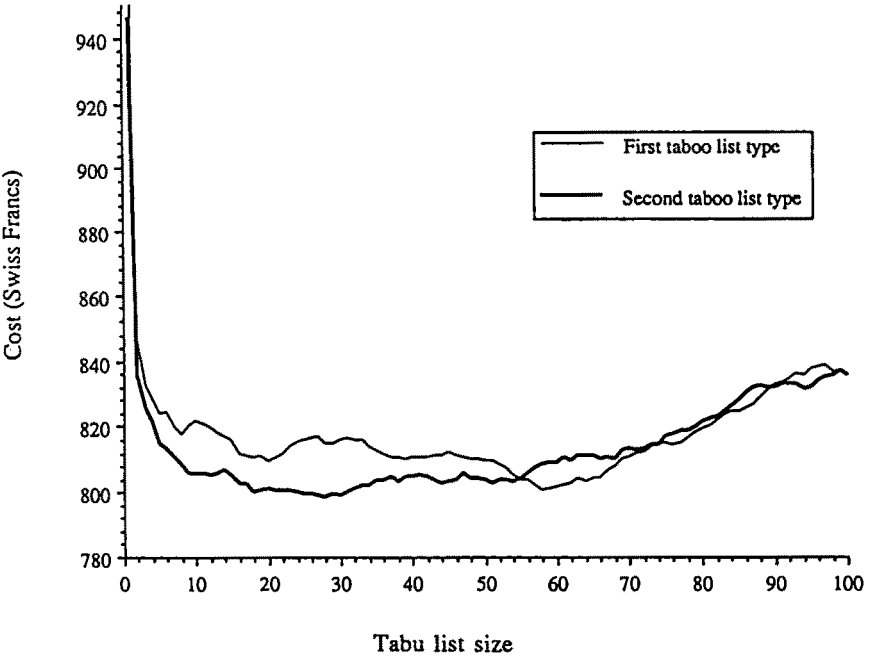

Fig. 9. Influence of the tabu list type.

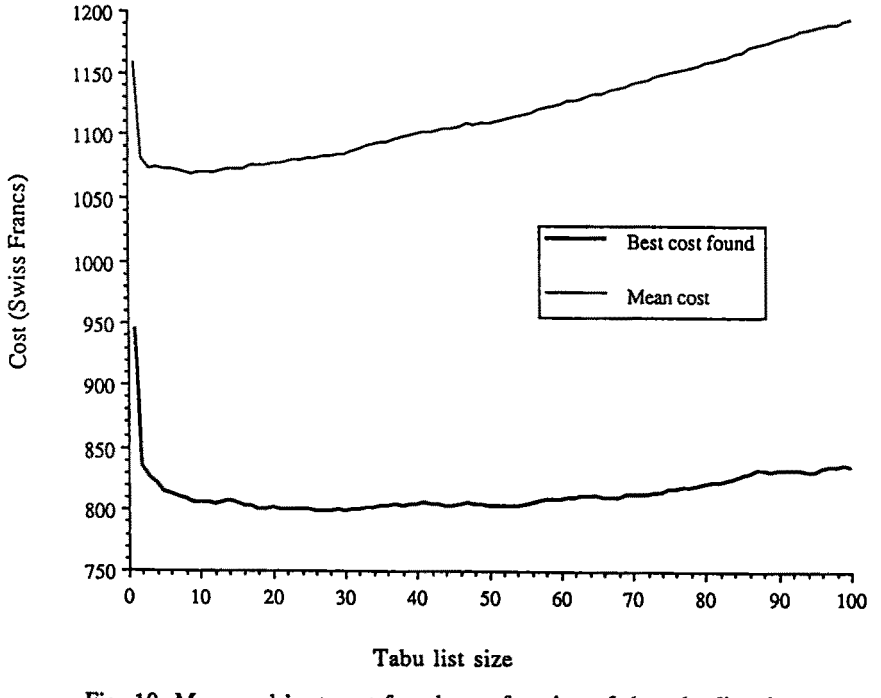

Fig. 10. Mean and best cost found as a function of the tabu list size.

search as a function of the tabu list size by plotting the mean value of the solutions visited in addition to the best value of the solution found. This is illustrated in fig. 10. Best tabu list sizes for this particular problem range between 3 and 10 according to the mean cost criterion, but between 25 and 30 according to the criterion of obtaining the best possible solution $s^*_{10,000}$. Undoubtedly, a tabu list size of 3 is too small to prevent cycling, whereas a size of 30 is too large to uncover the best solutions for some problems (but large tabu lists are often more advisable than small ones).

We therefore decided to generate the tabu list size randomly and uniformly between 6 and 30, determining a new size every 60 iterations. By this means, a region will be visited intensively when the tabu list is small and the search can leave this region when the tabu list is larger, a policy that has also been successfully applied to quadratic assignment and job shop scheduling problems [9,10].

Since all parameters of our TS procedure are now fixed, we examine its behaviour on several different problems whose characteristics are summarized in table 1. To compare the results provided by the first heuristic method with those

Table 1

Characteristics of problems tested.

	Total volume (pallets)	Total weight (kg)	Total number of orders	Lower bound on the number of trucks
Monday	177	71,390	83	10
Tuesday	176	71,330	79	10
Wednesday	196	79,820	78	11
Thursday	190	75,375	77	10
Friday	207	83,675	74	11
Saturday	218	86,840	78	12

obtained by our TS procedure, we point out that the first heuristic method seeks to minimize the total distribution time, and we may calculate the total distribution cost (in Swiss francs) of the solution provided by the first heuristic method by assigning trucks to routes as described above for our TS procedure. Notice that this method does not consider time window restrictions. Consequently, it tends to use a minimum number of very loaded vehicles (see table 2), the routes of which may violate time window constraints.

In table 2, we summarize the mean results obtained for the problems of table 1. The best known solutions of these problems were systematically found by TS. First, we present the results provided by the first heuristic method (derived from the method of Fisher and Jaikumar), which works only on aggregated orders, then those obtained by our TS procedure. Solutions for the TS procedure are presented after 2000 iterations using the second tabu list type, then after 10,000 iterations using

Table 2

Efficiency of the first heuristics and some variants of tabu search.

	Swiss francs (% above best known solution)		Minutes (% above best known solution)		Number of trucks	
	No aggregation	Aggregation	No aggregation	Aggregation	No aggregation	Aggregation
First heuristic method	–	16.0	–	10.0	–	11.00
2000 iterations, second tabu list	9.9	4.8	8.1	4.5	12.55	11.94
10,000 iterations, first tabu list	5.8	2.1	5.4	2.3	12.19	11.90
10,000 iterations, second tabu list	5.2	1.8	4.9	2.1	12.19	11.84
100,000 iterations, second tabu list	2.1	0.5	2.6	1.3	11.95	11.83

the first tabu list type, and finally after 10,000 and 100,000 iterations, again using the second tabu list type. Each problem was solved ten times, mixing orders initially. This generates many independent search paths because of partial neighbourhood examination.

It turns out that the second tabu list type provides solutions whose values are a few tenths of a percent below those obtained with the first tabu list type for 10,000 iterations. The aggregation of orders leads to very good solutions in a very short CPU time, but sometimes we found better solutions without aggregating orders.

Our TS procedure consistently obtains solutions of better quality than those generated by the method currently used by the company. This improvement holds for the total time criterion even if the choice rules of the TS optimization are not based on this criterion.

6. Conclusions

Using a standard tabu search method, using straightforward processes to ensure efficient execution, we achieved very encouraging results, obtaining solutions significantly better than those previously generated for practical application. Indeed, our outcomes indicate that distribution costs can be reduced 10 to 15 percent while continuing to observe all restrictions taken into account by the method based on the approach of [3].

To improve our TS procedure, knowledge about the problem has to be added, for instance by compelling some orders to belong to given routes; this could lead the search to good solutions very quickly.

Another avenue for potential improvement may result by incorporating iterated constructive processes, building the solution one route after the other, assigning orders that are farther from the depot before assigning those that are closer. TS machinery can be used to guide such processes as readily as to guide the processes indicated in the preceding sections; some preliminary experiments in this direction provide very good solutions in less than one CPU minute.

In addition to demonstrating the ability to deal with practical problems having many constraints in an easy way, and to obtain good solutions in moderate CPU time, our work underscores the convenient flexibility of the TS approach. Without important changes, it would be easy to adapt our TS procedure to consider other specifications such as pick-ups and deliveries, intermediate breaks, duration of routes, time windows for trucks, and multiple depots. Each of these can be accommodated by performing another computation of the objective function and by reducing the set of admissible solutions. Additional considerations such as differentiating drivers' territories and allowing multiple routes per vehicle can be handled, respectively, by including a routine to assign drivers to routes and by duplicating trucks to operate within specified time windows.

Acknowledgements

The authors are particularly grateful to Fred Glover, whose multiple suggestions and corrections were very helpful in improving the presentation of this paper. This research was partially supported by the Fond National Suisse de la Recherche Scientifique, Grant No. 20-27926.89.

References

[1] G.H. Bradley, G.G. Brown and G.W. Graves, Design and implementation of large scale primal transshipment algorithms, Manag. Sci. 24(1977)1.

[2] D. Costa, A tabu search algorithm for computing an operational timetable, ORWP 90/91, DMA, École Polytechnique Fédérale de Lausanne (1990), Eur. J. Oper. Res., to appear.

[3] M.L. Fisher and R. Jaikumar, A generalized assigment heuristic for vehicle routing, Networks 11(1981)109.

[4] A. Hertz, Finding a feasible course schedule using tabu search, Publication 703, University of Montreal (1990).

[5] S. Lin, Computer solutions of the traveling salesman problem, Bell Syst. Tech. J. 44(1965)2245.

[6] C.H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization Algorithms and Complexity* (Prentice–Hall, Englewood Cliffs, 1982).

[7] F. Semet, Problèmes de tournées de véhicules sous contraintes d'inaccessibilité, Internal Report, DMA, École Polytechnique Fédérale de Lausanne (1991).

[8] M.M. Solomon, Algorithms for the vehicle routing and scheduling problems with time window constraints, Oper. Res. 35(1987)254.

[9] E. Taillard, Parellel taboo search for the job shop scheduling problem, ORWP 89/11, DMA, École Polytechnique Fédérale de Lausanne (1989), ORSA J. Comput., to appear.

[10] E. Taillard, Robust taboo search for the quadratic assignment problem, Parallel Comput. 17(1991)443.

[11] M.L. Tricot and M. Donegani, Formules de réactualisation pour une famille d'indices de proximité inter-classe en classification hiérarchique, RAIRO Recherche Opérationnelle 23(1989)165.