A Tabu Search Approach for Delivering Pet Food and Flour in Switzerland
Author(s): Y. Rochat and F. Semet
Source: *The Journal of the Operational Research Society*, Vol. 45, No. 11 (Nov., 1994), pp. 1233-1246
Published by: Palgrave Macmillan Journals on behalf of the Operational Research Society
Stable URL: http://www.jstor.org/stable/2583852
Accessed: 05/03/2010 04:16

J. Opl Res. Soc. Vol. 45, No. 11, pp. 1233–1246

0160-5682/94 $9.00+0.00

A Tabu Search Approach for Delivering Pet Food and Flour in Switzerland

Y. ROCHAT[1] and F. SEMET[1,2]

[1]École Polytechnique Fédérale de Lausanne, Switzerland and [2]Université de Montréal, Canada

In this paper, we consider a real-life vehicle routeing problem that occurs in a major Swiss company producing pet food and flour. In contrast with usual hypothetical problems, a large variety of restrictions has to be considered. The main constraints are relative to the accessibility and the time windows at customers, the carrying capacities of vehicles, the total duration of routes and the drivers' breaks. To find good solutions to this problem, we propose two heuristic methods: a fast straightforward insertion procedure and a method based on tabu search techniques. Next, the produced solutions are compared with the routes actually covered by the company. Our outcomes indicate that the total distance travelled can be reduced significantly when such methods are used.

Key words: combinatorial optimization, vehicle routeing, tabu search

INTRODUCTION

This paper deals with a real-life vehicle routeing problem of a major Swiss firm which produces pet food and different types of flour. As each firm in the industry has low production costs, the transportation costs represent a large percentage of the total costs. Since the company wants to minimize these costs, managers are aware of substantial profits that may be obtained with a rationalization of the routes of vehicles. Moreover, the managers encounter some difficulties in developing routes satisfying all constraints. On the one hand, this leads to reductions in the service quality for the customers, on the other hand the drivers can be subject to fines. Indeed, the federal laws in force penalize very severely excess route duration and excess weight.

The basic problem consists of the elaboration of routes covered by the vehicles any day of the week on the basis of the actual demands known the day before. In this paper, we present two heuristic methods which tend to minimize the total distance travelled by the vehicles of the company while satisfying a variety of constraints which are not taken into account in the usual textbook vehicle routeing problems.

This paper is organized into six main sections, which will present respectively: (i) the description of the problem; (ii) the notations; (iii) a fast initialization heuristic based on a method proposed by Solomon[1]; (iv) the tabu search method (TS); (v) our adaptation of TS for the vehicle routeing problem (VRP) and (vi) numerical results which are compared with the routes actually covered by the vehicles of the company. Finally, some conclusions are drawn in the last section.

DESCRIPTION OF THE PROBLEM

Customers

The company sells to its customers pet food and flour in sacks, the weight of which varies between 25 and 50 kilograms and is 42 kilograms on average. The company stocks approximately 5000 customers, located in the French part of Switzerland, which are classified in two groups. The small customers, essentially farmers, which represent 70% to 80% of customers and 45% to 50% of demands, receive deliveries once a month. The large

Correspondence: Y. Rochat, École Polytechnique Fédérale de Lausanne, Département de Mathématiques, Chaire de Recherche Opérationnelle, CH-1015 Lausanne, Switzerland

customers, i.e. mainly wholesalers, retailers and bakers, have a delivery frequency between once a week and once every three weeks.

The great variety of locations of the customers (centre of a city, village, isolated farm, . . .) makes the access of all vehicles to all customers unlikely. For instance, the villages located in the mountains are only accessible with the most powerful trucks. Therefore, for each customer a set of vehicles, that are able to serve it, is considered.

Finally, the deliveries must respect the opening hours or available hours of customers. With each customer we associated, for each day of the week, two time windows during which the delivery must be performed. Indeed, the customers cannot be usually served during the lunch-time break.

Fleet of vehicles

The company serves its customers from one depot located in the north of Lausanne with a heterogeneous fleet consisting of 14 trucks. Each truck is characterized by its volume capacity and its weight capacity. The daily number of orders varies between 100 and 170 orders, which represent 150 tons on average in total weight. Due to the nature of the products, the volume capacity is not considered and only the weight capacity must be taken into account to load the vehicles.

We distinguish three different phases during the daily route of a vehicle.

(1) The travel from one customer to another or the travel from/to the depot according to the transport plan of the route.
(2) The approach operations which consist of finding the way to the customer from the village centre and parking the vehicle. The duration of the access operations for a customer is called the *access time*.
(3) The unloading of the demand. The unloading duration for a customer is called the *service time*.

The duration of a route includes the travel times, the access times, the service times and the possible waiting times arising when the arrival times at customers are not in the defined time windows. The total duration must not be greater than a maximal value set by the company, $D_{\max} = 10\text{h}15$.

The crew requirements

To model precisely this real-life problem, we decided to handle driver breaks during the elaboration of a route. We have based our arguments on the Swiss federal law covering the duration of work and rest of professional drivers. The main points are the following.

(1) After 5.5 hours of uninterrupted work, or 4 hours of uninterrupted driving time, the driver must take at least a one-hour break.
(2) We denoted as uninterrupted work or driving time that which is not interrupted by a 30-minute break.

We decided to consider only two breaks by day, one 30-minute break in the morning and one 60-minute break at lunch time. The reasoning behind this decision is as follows.

Traditionally, the break for lunch occurs between 11:45 am and 1:00 pm. It yields the following advantages. First, the lunch-time break and the usual closing hours of stores coincide and, second, drivers can work until 6:15 pm without taking another break in the afternoon.

At the present time, the drivers leave the depot at 5:00 am every day due to the time windows of the customers. Hence, they must have finished their work at 4:45 pm to respect the maximal duration of a route ($5{:}00 + 1\frac{1}{2}\,\text{hr} + D_{\max}$). Therefore, it is not necessary to consider a break during the afternoon. In order to take the lunch-time break at the indicated hours, drivers must take the morning break between 7:30 am and 10:00 am.

In summary, our optimization problem of a transport plan consists of elaborating a set of routes that minimizes the total travel distance while satisfying the following constraints.

(i) Each customer is visited by only one vehicle.
(ii) Each vehicle route terminates at its departure point, the depot.
(iii) The total duration of each route must not be greater than $D_{\max}$.
(iv) The carrying capacity for each vehicle must be respected.
(v) The vehicles have different carrying capacities.
(vi) Each customer is accessible by a known subset of the vehicles.
(vii) Each order must be served during a time window.
(viii) The driver breaks must be taken into account.

NOTATION

Before describing the proposed heuristic methods, let us introduce first the notation that will be used in this paper.

Let $X = \{x_0, \ldots, x_i, \ldots, x_n\}$ be the set of customers to be served for a given day where x_0 represents the depot. With each customer, we associate an order with a non-negative weight q_i ($q_0 = 0$). Each customer x_i has to be served during either of two time windows defined by their minimal and maximal bounds in minutes (see Figure 1): e_i^j and f_i^j are, respectively, the lower time limit and the upper time limit of the window j, $j = 1, 2$. Sometimes a customer has only a single time window, in which case we have $e_i^2 = e_i^1$ and $f_i^2 = f_i^1$.

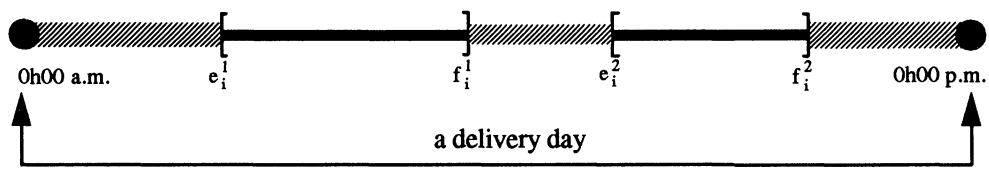

FIG. 1. *Time windows defined for each customer.*

Let $E = \{(x_i, x_j) | i \neq j\}$ be the set of travels between each pair of customers. With each travel (x_i, x_j) we associate two non-negative values d_{ij} and t_{ij}, which represent respectively the distance in kilometres of the shortest path between customers x_i and x_j and its duration in minutes.

For each customer x_i, we define an access time a_i ($a_0 = 0$), a service time s_i ($s_0 = 0$) and a waiting time w_i ($w_0 = 0$). Besides these times, we denote by h_i the arrival time at customer x_i and by b_i the beginning delivery time, i.e. the time at which the order of customer x_i begins being unloaded. If a vehicle travels directly from customer x_i to customer x_j, we have $h_j = b_i + s_i + t_{ij} + a_j$. If the vehicle arrives too early at x_j, the driver must wait before beginning to unload the order. Then, if $h_j \leq f_j^1$ we have $b_j = \max(e_j^1, h_j)$ otherwise $b_j = \max(e_j^2, h_j)$. If a vehicle arrives too late at x_j, a violation on the time windows $\max(0, b_j - f_j^2)$ occurs. This will not be permitted in the final solution.

A heterogeneous fleet $V = \{v_1, \ldots, v_i, \ldots, v_h\}$ of vehicles is based at the depot. For each vehicle v_i, Q_i denotes the carrying capacity of the truck. With each route R_i covered by vehicle v_i we associate a vector $(x_{i_0}, x_{i_1}, \ldots, x_{i_{m_i+1}})$, where x_{i_0} and $x_{i_{m_i+1}}$ represent fictitious customers at the depot and m_i the number of customers served on the route.

In order to model the drivers' breaks efficiently, we consider them as *fictitious customers* $\{x_{-2}, x_{-1}\}$ which satisfy the following conditions.

(1) A break is always taken right after unloading at a customer $x_i \in X$, $i > 0$.
(2) The fictitious customers have specific time windows which are 07:30 am to 10:00 am for the morning break ($e_{-2}^1 = e_{-2}^2 = 450$, $f_{-2}^1 = f_{-2}^2 = 600$) and 11:45 am to 1:00 pm for the lunch break ($e_{-1}^1 = e_{-1}^2 = 705$, $f_{-1}^1 = f_{-1}^2 = 780$).
(3) Access times to these customers are obviously null: $a_{-2} = a_{-1} = 0$.
(4) The service time at each of these customers is set equal to the duration of the break, i.e. $s_{-2} = 30$ and $s_{-1} = 60$.

(5) The length and the duration of the travel between any customer and the fictitious customers are equal to zero. If we suppose that the delivery order of customers in a route R_i contains the sequence (x_i, x_p, x_j) with $i, j \in \{0, \ldots, n\}$ and $p \in \{-2, -1\}$, then we will have:

$$d_{ip} = t_{ip} = 0$$
$$d_{pj} = d_{ij} \quad \text{and} \quad t_{pj} = t_{ij}.$$

Thus, to evaluate a route R_i, the preceding customer of a fictitious customer must always be available.

(6) The weights of the orders associated with fictitious customers are equal to zero: $q_{-2} = q_{-1} = 0$.

In order to consider fictitious customers in our notation, we redefine X as follows: $X = \{x_{-2}, x_{-1}, x_0, \ldots, x_i, \ldots, x_n\}$. Each customer $x_i \in X$, $i \leqslant 0$, is served by all the vehicles of the company.

Before presenting the heuristic methods, note that a number of parameters controlling the initialization heuristic method and the tabu search will be defined. Numerical values will be illustrated in the examples later in the paper.

INITIALIZATION HEURISTIC METHOD

The general scheme, which we used to solve this real-life VRP, embodies the combination of two heuristic methods:

(1) a fast straightforward heuristic procedure,
(2) a method based on tabu search techniques.

Using method (1) allows us to generate an initial solution for the tabu search procedure and increase the flexibility of future software. Indeed, (1) is able to provide an admissible solution for the managers in a very short CPU time.

The simple heuristic procedure we implemented, has been proposed by Solomon[1]. In his paper, Solomon extended some classical methods devised for the standard VRP to allow them to handle time window restrictions. His computational experiments led him to conclude the superiority of his insertion-based algorithm as compared with other procedures. Therefore, we propose an extension of Solomon's insertion method in order to take into account the accessibility restrictions and the drivers' breaks. Let us first summarize the original method.

The routes $R_1, \ldots, R_h$ are developed sequentially. The current route R_i is first initialized according to a criterion such as the insertion of the farthest unserved customer or the insertion of the unserved customer with the earliest lower time limit. Then, R_i is expanded until no new customer could be served on R_i without violating the time window restrictions, the capacity constraint or the total duration constraint. Then, a new route R_{i+1} is considered. The customers are inserted in R_i sequentially according to the following selection procedure.

Let x_u denote an unserved customer. For each feasible insertion of x_u between x_{i_j} and $x_{i_{j+1}}$ in route R_i, we compute:

(i) the increase in the length of the route:

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

(ii) the delay in the start of delivery time for customer $x_{i_{j+1}}$:

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

where $b_{i_{j+1}}$ is the current delivery time of $x_{i_{j+1}}$, and $b^u_{i_{j+1}}$ is the new delivery time if x_u is inserted.

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

$$\alpha_1 c_1(i_{j(u)}, u) + \alpha_2 c_2(i_{j(u)}, u) = \min_{k=0,\ldots,m_i} (\alpha_1 c_1(i_k, u) + \alpha_2 c_2(i_k, u))$$

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

Finally, the best customer x_{u^*} to be inserted in R_i is computed as follows:

$$\lambda d_{0u^*} - (\alpha_1 c_1(i_{j(u^*)}, u^*) + \alpha_2 c_2(i_{j(u^*)}, u^*)) = \max_{x_u \text{ unserved and feasible}} (\lambda d_{0u} - (\alpha_1 c_1(i_{j(u)}, u) + \alpha_2 c_2(i_{j(u)}, u)))$$

with $\lambda \geq 0$.

After each insertion, a 2-optimality routine[2] is used to improve R_i which is updated if and only if the provided circuit is admissible for the time window restrictions.

We now describe how the foregoing heuristic method has to be extended to handle the drivers' breaks and the accessibility constraints. By associating the drivers' breaks with dummy customers, the breaks are taken into account easily. Indeed, instead of assuming that the routes R_i are empty before the insertion of the first customer, i.e. $R_i = (x_{i_0}, x_{i_1})$ with $x_{i_0} = x_{i_1} = x_0$ for all vehicles v_i, we initialize the routes as follows: $R_i = (x_{i_0}, x_{i_1}, x_{i_2}, x_{i_3})$ with $x_{i_0} = x_{i_3} = x_0$ and $x_{i_1} = x_{-2}$, $x_{i_2} = x_{-1}$. Hence, the increase in the length of the route $c_1(i_j, u)$ needs another computation when x_{i_j} or $x_{i_{j+1}}$ are dummy customers. Specifically,

$$\text{if } ((x_{i_j} = x_{-2}) \text{ and } (x_{i_{j+1}} \neq x_{-1})) \text{ or } (x_{i_j} = x_{-1}) \text{ then } c_1(i_j, u) = d_{i_{j-1}u} + d_{ui_{j+1}} - \mu d_{i_{j-1}i_{j+1}}$$

$$\text{if } (x_{i_j} = x_{-2}) \text{ and } (x_{i_{j+1}} = x_{-1}) \text{ then } c_1(i_j, u) = d_{i_{j-1}u} + d_{ui_{j+2}} - \mu d_{i_{j-1}i_{j+2}}.$$

With these modifications the preceding insertion routine can be used in a straightforward fashion.

To handle the accessibility restrictions, a two-phase method is devised. In the first phase we solve a VRP restricted to the accessibility-constrained customers and the trucks that can reach them. As the number of accessibility-constrained customers is low compared with the total number of customers, the consistency of developed routes has to be preserved. Therefore, the objective of the first phase is the construction of compact routes, i.e. routes where the customers are close to each other, even if the total number of tours is not minimized. Indeed, the partially generated routes may be completed during the second phase by inserting some non-accessibility-constrained customers in them.

The first phase can be viewed as a parallel method due to the construction of several routes $R_{i^1}, R_{i^2}, \ldots, R_{i^p}$ simultaneously, $p \leq h$. At each iteration, the alternative possibilities concerning the current customer x_u are the insertion of x_u in one of the existing tours $R_{i^1}, R_{i^2}, \ldots, R_{i^p}$, or the initialization of a new route $R_{i^{p+1}}$ with x_u if $p < h$. The decision is made according to the minimum cost between the initialization cost given by $2d_{0u}$ and the best insertion cost in one of the current routes calculated as

$$\min_{k=1,\ldots,p} (\alpha_1 c_1(i^k_{j(u)}, u) + \alpha_2 c_2(i^k_{j(u)}, u)).$$

To guarantee the compactness of routes, we modify the calculation of $c_1(i_j, u)$, the increase in the length of route R_i, by introducing the notion of the neighbourhood of a customer. The neighbourhood of x_i is defined as the set $N(x_i)$ of the N_n nearest customers from x_i. Note that $N(x_i)$ can include non-accessibility-constrained customers as well as accessibility-constrained customers. Then $c_1(i_j, u)$ is computed as follows:

$$c_1(i_j, u) = \begin{cases} d_{i_j u} + d_{ui_{j+1}} - \mu d_{i_j i_{j+1}} & \text{if } x_u \in N(x_{i_j}) \cup N(x_{i_{j+1}}), \\ \infty & \text{otherwise.} \end{cases}$$

When all the accessibility-constrained customers are served, the second phase of the routine is applied.

The key point of the second phase consists of completing first the routes created during the preceding phase before initializing new routes. The completion step is performed in parallel, i.e. at each iteration we determine what customer is inserted on what tour. Next, the original procedure designed by Solomon is applied to insert the remaining non-accessibility-constrained customers.

By combining these extensions, we develop a new insertion-based method which provides feasible solutions for our real-life VRP in only a few seconds. In the next section a tabu search approach is proposed to improve the solutions obtained by this method.

TABU SEARCH HEURISTIC METHOD

The tabu search method (TS) is a general improvement heuristic procedure which has been very efficient for solving many combinatorial optimization problems such as the quadratic assignment problem[3], the job-shop scheduling problem[4] or the vehicle routeing problem[5-7]. In this section, we sketch some generalities of tabu search. For interested readers, a detailed description of TS is provided in a recent paper by Glover *et al.*[8].

Tabu search, independently suggested by Glover[9] and Hansen[10], is a metaheuristic in which a local search procedure is applied at each step of the general iterative search process.

Let us consider the following optimization problem:

$$\min_{s \in S} f(s).$$

TS starts from an initial solution $s_0 \in S$ which is either obtained with a heuristic procedure or chosen arbitrarily in S. Then, the search explores the set S of solutions by moving from the current solution to another. Specifically, at iteration l, a new solution $s_{l+1} \in S$ is reached by applying a modification m_l to the current solution $s_l \in S$. Such a modification m_l is called a move. Considering all feasible moves, the set of solutions that can be obtained by applying one move to the current solution is defined as the neighbourhood of s_l and is denoted by $N(s_l)$, $N(s_l) \subseteq S$.

$s_{l+1} \in N(s_l)$ is determined as the neighbour of s_l that most improves the objective function f. If f cannot be improved, s_{l+1} can be chosen as the neighbour that least deteriorates the value $f(s_l)$. Thus, TS avoids the local search being trapped in a local minimum.

To avoid performing a move returning to a recently visited region, the reverses of the last Θ moves are forbidden. These moves are then considered as tabu. They are recorded in a list called a tabu list which is updated at each step.

Setting the tabu status to a move is sometimes too restrictive. When a tabu move leads the search to a promising region, then the tabu status has to be overridden. This can be done according to some conditions called aspiration level conditions.

The efficiency of TS can also be improved by using an intensification strategy which consists of closely exploring an attractive region of the solution space S where the solutions satisfy some common properties.

Finally, the tabu search process is stopped when a given criterion is satisfied. A commonly used criterion is a bound on the total number of iterations.

ADAPTATION OF TS FOR VRP

We describe below an adaptation of the tabu search metaheuristic to our problem, which differs from a previously suggested method[6] specifically because of roles played by the relaxation of constraints and the intensification strategy. First we define three basic components on which the tabu search method is based: the solution space, the characterization of a move in this space and the neighbourhood of a solution.

Solution space

The solution space S is defined as the set of routes serving the whole set of customers while satisfying the accessibility constraints. The constraints on the carrying capacities of vehicles, on the duration of routes and on the time windows of customers are relaxed. It is worthwhile noting that obtaining infeasible solutions is convenient to explore efficiently the space of feasible solutions and to avoid staying in a local optimum. Nevertheless, only the feasible solutions will be considered in determining the best solution obtained during the tabu search.

Definition and cost of a move $m(x_{i_k}, R_j)$

A move $m(x_{i_k}, R_j)$ consists of removing a customer x_{i_k}, $i_k > 0$, from a route R_i, the source route, and inserting it in another route R_j, the target route. This move is feasible if and only if customer x_{i_k} is accessible by vehicle v_j (see Figure 2).

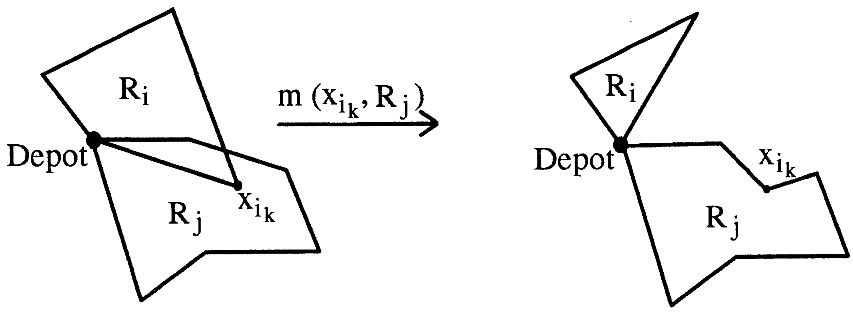

FIG. 2. *Move defined for the TS procedure.*

The evaluation of the cost of a move $m(x_{i_k}, R_j)$ takes into account, on the one hand, the changes in lengths of routes R_i and R_j and, on the other hand, the changes in the violation costs of the relaxed constraints.

The length variation $\Delta(x_{i_k}, R_j)$ of routes R_i and R_j is given by:

$$\min_{p=0,1,\ldots,m_j} (d_{j_p i_k} + d_{i_k j_{p+1}} - d_{j_p j_{p+1}}) - (d_{i_{k-1} i_k} + d_{i_k i_{k+1}} - d_{i_{k-1} i_{k+1}}).$$

For each route R_i, $R_i = (x_{i_0}, x_{i_1}, \ldots, x_{i_{m_i+1}})$, the violation costs relative to the relaxed constraints are computed according to the following formulae.

(i) The excess vehicle capacity C_i:

$$C_i = \max\left(0, \sum_{k=1}^{m_i} q_{i_k} - Q_i\right).$$

(ii) The excess route duration D_i:

$$D_i = \max\left(0, \sum_{k=0}^{m_i} t_{i_k i_{k+1}} + \sum_{k=1}^{m_i} (a_{i_k} + s_{i_k} + w_{i_k}) - D_{\max}\right).$$

(iii) The time window violations TW_i:

$$TW_i = \sum_{k=1}^{m_i} (\max(0, b_{i_k} - f_{i_k}^2))^2,$$

where b_{i_k} and $f_{i_k}^2$ are, respectively, the delivery time and the upper time limit of the second time window at customer x_{i_k}.

Hence, the cost of a move $m(x_{i_k}, R_j)$ is computed as follows:

$$\Delta(x_{i_k}, R_j) + p_1(\delta C_i + \delta C_j) + p_2(\delta D_i + \delta D_j) + p_3(\delta TW_i + \delta TW_j),$$

where δC_i, δD_i, δTW_i and δC_j, δD_j, δTW_j are, respectively, the changes in the violation costs due to the removal of x_{i_k} from R_i and its insertion into R_j, and p_1, p_2 and p_3 are three real positive weights.

Once a move has been performed, the lengths of routes R_i and R_j are improved using a 2-optimality procedure[2]. To avoid creating new time window violations, the routes are updated if and only if the time window restrictions are satisfied.

Neighbourhood of a solution

The neighbourhood of solution s, $N(s)$, is defined as the set of solutions that can be obtained from s by applying one move. As there are h routes and n customers and as each customer can be transferred at most to one of $h - 1$ routes, the cardinality of the neighbourhood is $n(h - 1)$ which is 1000, on average, in our case. Owing to its moderate size, the neighbourhood of a solution will be completely examined. At each iteration, the determination of the best solution in the neighbourhood is performed as explained above.

Objective function

With each solution we associate the following function F_f which represents the sum of route lengths:

$$F_f(s) = \sum_{i=1}^{h} \sum_{k=0}^{m_i} d_{i_k i_{k+1}}$$

Considering the definition of the violation costs, the objective function of our problem is defined as follows:

$$F(s) = F_f(s) + p_1 \sum_{i=1}^{h} C_i + p_2 \sum_{i=1}^{h} D_i + p_3 \sum_{i=1}^{h} TW_i$$

If the solution s is feasible, $F_f(s)$ and $F(s)$ are identical otherwise $F(s)$ contains three penalty terms due to violations of relaxed constraints.

Update of penalty parameters

As it is difficult and tedious to set appropriate values for the penalty parameters p_1, p_2 and p_3, Gendreau *et al.*[4] proposed a method which updates them during the execution of the algorithm. The values of p_1, p_2 and p_3 are initialized randomly and are modified dynamically every L_p iterations, L_p denoting the length of the modification cycle. That leads the search to visit a mix of feasible and infeasible solutions.

Specifically, if the current iteration l is a multiple of L_p, penalty parameters are updated in the following way.

(1) If the L_p previous solutions were feasible with respect to vehicle capacity, set $p_1 = p_1/\gamma$. If they were all infeasible, set $p_1 = \gamma p_1$.
(2) If the L_p previous solutions were feasible with respect to route duration, set $p_2 = p_2/\gamma$. If they were all infeasible, set $p_2 = \gamma p_2$.
(3) If the L_p previous solutions were feasible with respect to time windows, set $p_3 = p_3/\gamma$. If they were all infeasible, set $p_3 = \gamma p_3$.

Here, γ is a real random number between 1.5 and 2.0. Note that if there was a mix of feasible and infeasible solutions relative to one of the relaxed constraints during the last L_p iterations, the value of the associated penalty parameter is unchanged.

Tabu list

The tabu list contains the Θ last moves realized. Hence, if customer x_{i_k} is removed from route R_i at iteration l, inserting x_{i_k} on R_i is forbidden until iteration $l + \Theta + 1$. The length of the tabu list Θ changes dynamically during the execution of our algorithm.

Every L_Θ iterations, Θ is determined randomly between Θ_{min} and Θ_{max} using a distribution that favours the medium values. L_Θ denotes the frequency at which a new value of Θ is generated.

Moreover, the value of the tabu list size Θ is modified at each iteration. Let $F(s_l)$ be the value of the objective function associated with the solution obtained at iteration l and F_f^* the value of the objective function associated with the best feasible solution obtained up to iteration l. The updating procedure is as follows.

If $F(s_l) < F_f^*$
then $\Theta = \Theta_{min}$,
else if $(F(s_l) < F(s_{l-1}))$ and $(\Theta > \Theta_{min})$ then $\Theta = \Theta - 1$,
if $(F(s_l) > F(s_{l-1}))$ and $(\Theta < \Theta_{max})$ then $\Theta = \Theta + 1$.

Such a dynamic list tends to intensify the local search when an improvement has taken place in the current solution and also tends to permit escaping quickly from local optima. Moreover, when we improve the best known solution we explore intensively the region of the local optimum obtained, even if this means returning to a previously encountered solution.

We define the commonly applied aspiration function which releases the tabu state of a move if the move leads to an interesting region. Thus, if we obtain a feasible solution with a value $F_f(s_l) < F_f^*$ or an infeasible solution with a value $F(s_l) < F_f^*$, a tabu move will be performed.

Intensification

The intensification strategy restricts the search to an attractive region of the solution space where the solutions are close in a certain sense. Considering the VRP, we define close solutions as those with some common routes. Our intensification strategy consists of rendering tabu some routes of the best solution s^* encountered so far.

Let L_{int} denote the number of iterations for which tours of s^* are considered as tabu. For the next L_{int} iterations, the TS procedure inspects only moves relative to customers served on non-tabu routes. In that sense, this intensification procedure can also be viewed as a reduction of the neighbourhood of the best solution visited s^*. A similar approach has been used by Taillard[7] for solving classical VRP with parallel iterative search methods.

To determine which routes of s^* are declared tabu, we compute the centre of gravity of each route as the centre of gravity of customers served on the route with weights set equal to one. The coordinates of a centre of gravity are calculated in a polar-coordinate system for which the central depot is the origin. Next, the routes are ordered according to increasing values of the polar coordinate angles of centres of gravity. The search is intensified on a sequence of routes found by selecting a randomly chosen route and its $(h_{int} - 1)$ successors according to the previously determined centre of gravity order. h_{int} denotes the number of consecutive tours currently examined in the intensification process. The remaining tours have a tabu status for the next L_{int} iterations, i.e. no customer can be inserted or be deleted on these tours during L_{int} iterations. Thus, the search intensively examines a subset of routes that are spatially close. Moreover, during the intensification phase the TS method can initiate new routes when some trucks are available.

If the best solution found so far is improved, the intensification process is stopped and the global search restarts. Otherwise, a new sequence of h_{int} routes is examined. When all subsets containing h_{int} consecutive routes have been generated, h_{int} is increased to $h_{int} + 2$. If h_{int} is greater or equal to the total number h^* of routes of s^*, the intensification is stopped and the general search on the whole set of routes restarts. As some natural obstacles, e.g. mountains or lakes, are present in the Swiss topology, sequences for which the centres of gravity of routes are on both sides of such an obstacle are not considered.

Each time a new sequence of routes is selected, the tabu list size Θ and the penalty parameters p_1, p_2, p_3 are set equal to new initial values. The tabu list size Θ has to be lowered when there is a reduction of the neighbourhood due to intensification. Thus, Θ is generated randomly between Θ_{min}^{int}, Θ_{max}^{int} (see above).

Finally, it is worth noting that in our TS algorithm the intensification strategy is used only when the best solution s^* has not been improved during the last L_{gen} iterations. L_{gen} denotes then the number of iterations devoted to the full problem before invoking the intensification procedure.

TS algorithm

Let l^* denote the iteration at which the best solution s^* of value F_f^* has been found, and let L_{max} be the maximum number of iterations allowed for the entire process. The TS algorithm proceeds as follows.

Step 1. Obtain an initial feasible solution s_0 of value $F_f(s_0)$ using the initialization heuristic method.

Step 2. Set $F_f^* = F_f(s_0)$, $s^* = s_0$, $l^* = 0$, $l = 0$, h_{int}, L_Θ, L_{gen}, L_{int}, L_p. Generate penalty parameters p_1, p_2 and p_3 and an initial tabu list size Θ between Θ_{min} and Θ_{max}.

Step 3. $l = l + 1$
If $l \bmod L_\Theta = 0$ then
generate a new tabu list size Θ.
(*Perform a move*)
Determine the best move $m^*(x_{i_k}, R_j)$ which is not tabu or satisfies the aspiration conditions.
Obtain the solution s_l by moving customer x_{i_k} from route R_i to route R_j.
Improve routes R_i and R_j by using a 2-optimality procedure.
Forbid the insertion of x_{i_k} in R_i during the next Θ iterations.
(*Improvement of the best solution found up to now*)
If $(F(s_l) = F_f(s_l))$ and $(F(s_l) < F_f^*)$ then
  $F_f^* = F(s_l)$, $l^* = l$, $s^* = s_l$.
  Set h_{int} equal to the initial value.
(*Update the penalty parameters*)
If $l \bmod L_p = 0$ then
  update the penalty parameters p_1, p_2 and p_3.
(*Intensification*)
If $(l - l^* \geqslant L_{gen})$ and $(((l - (l^* + L_{gen})) \bmod L_{int} = 0)$ and $(h_{int} < h^*)$ then
  if all subsets containing h_{int} consecutive routes have been examined then
    $h_{int} = h_{int} + 2$.
  If $h_{int} < h^*$ then
    generate a new sequence of h_{int} routes,
    generate penalty parameters p_1, p_2 and p_3 and a new tabu list size Θ between Θ_{min}^{int} and Θ_{max}^{int}.

Step 4. Repeat Step 3 until $l = L_{max}$.

NUMERICAL RESULTS

Within the framework of our study, we are interested in the development of routes on the basis of the actual demands recorded during a representative week. Therefore, our analysis is based on a set of customers with the orders they placed each day of this week. For each day the length of routes actually covered by the trucks of the company has been recalculated with the same distance matrix as the one used in our methods. Thus, we can compare the actual tours with the solutions produced by our heuristics.

Observing the data associated with this representative week, we notice that many orders are placed by customers located in the same village. Therefore, to reduce the size of the neighbourhood of a solution and cut down CPU time, we have examined the behaviour of our heuristic methods on problems for which the orders delivered in the same village are aggregated into a single order when the time windows are identical. The characteristics of problems tested are summarized in Table 1.

Numerical experiments performed on a VAXstation 3100M38SPX lead us to use the original data for Monday, Wednesday and Thursday and the aggregated orders for Tuesday and Friday which are the days with the greatest total numbers of orders to deliver. Thus, the

TABLE 1. *Characteristics of the problems tested*

	Total weight (kg)	Total number of orders	Total number of aggregate orders
Monday	121 235	105	65
Tuesday	161 423	129	89
Wednesday	157 853	109	76
Thursday	128 240	118	72
Friday	202 461	170	129

daily number of orders varies between 89 and 129 orders and total weight is about 150 tons on average.

We now give the values of the various parameters selected to obtain the reported results.

Initialization heuristic method

In the first phase of the procedure, we consider only accessibility-constrained customers. For each of these, the neighbourhood includes the $N_n = 15$ nearest customers. For given values of α_1, α_2, λ and μ, two different criteria are used to initialize the routes: the farthest unserved customer and the unserved customer with the earliest lower time limit.

Different sets of parameters are considered:

$$\alpha_1 = 0.0; 0.1; \ldots; 1.0$$

$$\alpha_2 = 1 - \alpha_1,$$

$$(\lambda, \mu) = \begin{cases} (1)\ \lambda = 1.25; 1.50; 1.75; 2.00 \quad \text{and} \quad \mu = \lambda - 1, \\ (2)\ \lambda = 0.0; 0.5; 1.0; 1.5 \quad \text{and} \quad \mu = 1. \end{cases}$$

As 11 values for α_1 and α_2, eight values for λ and μ, and two initialization criteria are used, the reported result is the best of 176 runs for each day of the week. From these results, we have determined the best set of parameters on average, i.e. the set providing the best average on the lengths of the five transport plans computed. We use this to obtain the initial feasible solution for the TS heuristic. The sets of parameters used to obtain the best solution for each day of the week and the best solution on average are summarized in Table 2.

TABLE 2. *Best sets of parameters for the initialization heuristic procedure*

	Initialization criteria	α_1	α_2	λ	μ
Best set for Monday	Earliest	0.4	0.6	0.5	1.0
Best set for Tuesday	Farthest	0.9	0.1	2.0	1.0
Best set for Wednesday	Farthest	1.0	0.0	1.0	1.0
Best set for Thursday	Farthest	0.7	0.3	2.0	1.0
Best set for Friday	Farthest	1.0	0.0	1.5	0.5
Best set on average	Farthest	0.7	0.3	1.5	1.0

The variation in best values can be explained by the fact that the set of customers is different for each day. Moreover, the time windows are more or less restrictive for each problem. In Table 3, the results provided are presented with those obtained using the TS approach.

Tabu search heuristic method

We now fix all parameters of our TS procedure. Every $L_\Theta = 200$ iterations, the tabu list size Θ is generated randomly between $\Theta_{\min} = 10$, $\Theta_{\max} = 40$. The length of the modification cycle L_p for the updating procedure of the penalty parameters is set equal to 10 iterations when orders are not aggregated. Otherwise, L_p is set equal to 20.

When no improvement of the best solution found so far has occurred for $L_{gen} = 1000$ iterations, the intensification is applied. Initially, sequences of $h_{int} = 4$ consecutive routes are examined for $L_{int} = 250$ iterations. Due to the reduction of the neighbourhood, the parameters involved in the generation of the tabu list size are modified and set to the following values: $\Theta_{min}^{int} = 3$, $\Theta_{max}^{int} = 13$. Finally, the maximum number of iterations L_{max} is set equal to 20 000.

The results obtained with our two heuristic methods for each day of the week are presented in Table 3. For each day, we report the variation percentages of the total length of computed routes in comparison with the total length of routes actually covered by the company. First, we present the results provided by the heuristic initialization method: the best determined solution and the solution obtained with the best set on average (see Table 2) which is the initial TS solution. To generate different search paths in the TS approach, each problem was solved 10 times using different initial random seeds. We report in Table 3 the percentage changes of the mean results provided by TS after 5000 iterations, and 20 000 iterations. Finally, we give the percentage for the best result obtained over the 10 runs and the best known solution, where the latter has been provided by the TS procedure during the computational experiments performed to set the different parameters.

We summarize in Table 4 the number of trucks used in the reported results.

TABLE 3. *Percentage change in the total length of routes*

	Initialization method		TS method			
	Best solution	Initial TS solution	Average solution		Best solution	
			5000 iterations	20 000 iterations	over 10 runs	known
Monday	−0.3	7.3	−6.5	−9.1	−10.5	−10.5
Tuesday	8.0	8.6	4.0	2.9	1.7	0.8
Wednesday	9.8	13.8	1.6	0.0	−0.7	−0.9
Thursday	−10.3	−5.6	−15.7	−16.6	−18.7	−20.7
Friday	3.8	10.0	0.2	−2.4	−4.4	−4.4

TABLE 4. *Number of trucks used*

	Actual company solution	Initialization method		TS method			
		Best solution	Initial TS solution	Average solution		Best solution	
				5000 iterations	20 000 iterations	over 10 runs	known
Monday	8	8	8	8.0	8.0	8	8
Tuesday	12	12	12	12.0	12.0	12	12
Wednesday	10	10	10	10.2	10.0	10	10
Thursday	9	8	8	8.0	8.0	8	8
Friday	14	14	14	13.9	13.6	13	13

It is worthwhile to note that most of the routes actually covered by the vehicles of the company violate the different constraints taken into account in our formulation. The violations result from the difficulties encountered by the managers of the company in developing routes satisfying all constraints. Such difficulties motivated our study. The global violations are presented in Table 5 for each day of the representative week.

Our outcomes are globally very encouraging. Indeed, the reported percentages are computed in comparison with total lengths of infeasible routes. Consequently, these lengths may be lower bounds on the optimal solutions for some problems. Moreover, the total distance covered during the week is increased by 7.7% in the initial TS solutions, but is decreased by 3.6% on average using the TS algorithm. The tabu search procedure clearly

TABLE 5. *Violations of the constraints in the actual company solution*

	Excess route duration	Total violation of the time windows	Excess vehicle capacity (kg)
Monday	3h33	2h45	795
Tuesday	6h34	1h41	—
Wednesday	2h28	2h10	1810
Thursday	0h40	0h51	2840
Friday	4h39	3h21	345

outperforms the initialization method, even if the total number of iterations is reduced. Thus, 5000 iterations lead to an average decrease of 1.8% on the total length of routes performed during the week.

Considering the number of vehicles used, the results produced by both procedures are quite similar. Indeed, all solutions require the same number of trucks except for the VRPs occurring on Thursday and Friday. For the Friday problem, TS frequently finds solutions with one vehicle less.

Using the best set of parameters, the initialization routine improves the solution performed by the company only on the VRP occurring on Thursday. For this problem, both methods generate good solutions requiring one vehicle less than the actual solution. It is worthwhile noting that the latter is the only actual solution which nearly satisfies the route duration and time window constraints. For all other problems, the initialization procedure gives feasible solutions which are about 10% longer than the actual ones. However, this method is still of interest as it prescribes routes in only a few seconds of CPU time.

All solutions obtained by the initialization algorithm are significantly improved by the TS method. The results can be classified in two groups. In the group containing the VRPs occurring on Monday, Thursday and Friday, TS is outstanding, producing good solutions with fewer vehicles than the actual solutions. In the group containing the VRPs solved on Tuesday and Wednesday, the performance of TS is moderate. This may be explained by the fact that the violations on the routes actually covered by the trucks are considerable. Moreover, the average weights per order on both days are greater than the average weight on the other days. Under such circumstances the capacity constraints play a central role and may hinder the discovery of solutions better than the actual ones.

Finally, we have to point out that the computation time required by our TS approach is one minute per 110 iterations on our workstation. CPU time could be considerably reduced if the 2-optimality procedure[2] was invoked less frequently. Moreover, we could perform less than 20 000 iterations and maintain good quality in the solution produced.

CONCLUSIONS

In this paper we have presented two heuristic methods for solving a real-life VRP faced by one of the major flour producers in Switzerland. The first procedure is a straightforward insertion-based algorithm whose performance is moderate. However, as it produces a solution in a few seconds, the insertion procedure can be invoked several times with different sets of parameters in order to improve the quality of the solutions proposed.

The second method is an adaptation of the TS metaheuristic. The key points relative to our TS approach are the relaxation of constraints and the intensification strategy. The relaxation of constraints allows us to expand the solution space. Thus, the search is diversified since infeasible solutions are examined as well as feasible ones. The intensification tactic plays a complementary role. Indeed, it leads the search to visit solutions close to the best solution found so far by rendering some routes tabu. The numerical results have led us to conclude that the TS method dominates even if the total number of iterations is small. Thus, we can obtain good solutions in a reasonable amount of CPU time. Moreover, embedding an iterative search method, such as a TS algorithm, in decision support software can be particularly useful. Indeed, as several solutions with around the same length of routes are examined, several transport plans can be proposed to the user.

Acknowledgements—The authors would like to acknowledge Professor E. A. Silver for his multiple and valuable suggestions to improve the presentation of this paper,

REFERENCES

1. M. M. Solomon (1987) Algorithms for the vehicle routing and scheduling problems with time window constraints. *Opns Res.* **35**, 254–265.
2. S. Lin (1965) Computer solutions of the traveling salesman problem. *Bell Syst. Tech. J.* **44**, 2245–2269.
3. E. Taillard (1991) Robust taboo search for the quadratic assignment problem. *Parallel Computing* **17**, 443–455.
4. M. Dell'Amico and M. Trubian (1993) Applying tabu search to the job-shop scheduling problem. *Ann. Opns Res.* **41**, 231–252.
5. M. Gendreau, A. Hertz and G. Laporte (1994) A tabu search heuristic for the vehicle routing problem. To appear in *Mgmt Sci.*
6. F. Semet and E. Taillard (1993) Solving real-life vehicle routing problems efficiently using taboo search. *Ann. Opns Res.* **41**, 469–488.
7. E. Taillard (1993) Parallel iterative search methods for vehicle routing problems. *Networks* **23**, 661–673.
8. F. Glover, E. Taillard and D. de Werra (1993) A user's guide to tabu search. *Ann. Opns Res.* **41**, 3–28.
9. F. Glover (1986) Future paths for integer programming and links to artificial intelligence. *Comps and Opns Res.* **13**, 533–549.
10. P. Hansen (1986) The steepest ascent mildest descent heuristic for combinatorial programming. Presented at the *Congress on Numerical Methods in Combinatorial Optimization*, Capri, Italy.