ELSEVIER

European Journal of Operational Research 86 (1995) 480–490

EUROPEAN JOURNAL OF OPERATIONAL RESEARCH

Theory and Methodology

# Improvement heuristics for the Vehicle Routing Problem based on Simulated Annealing

Alex Van Breedam

*Faculty of Applied Economics, University of Antwerp – RUCA, Middelheimlaan 1, B-2020 Antwerpen, Belgium*

Received February 1993; revised February 1994

**Abstract**

This paper reports on the use of simulated annealing-based improvement methods for the Vehicle Routing Problem. The improvement methods considered are aimed at relocating and/or exchanging stops or strings of stops between different routes, starting from an initial feasible solution. The simulated annealing-based improvement methods are compared against their descent alternatives as well as other metaheuristics implementations on a set of classical test problems. The results are reported.

*Keywords:* Vehicle routing problem; Simulated annealing; Heuristics

## 1. Introduction

The standard Vehicle Routing Problem (VRP) can be described as one of finding a set of routes for a fleet of vehicles which have to service a number of stops from a central depot. It is assumed that every vehicle has the same capacity and the number of vehicles is unlimited. The vehicles depart and arrive at the depot. The demand quantity at each stop is known in advance and is deterministic. No single demand quantity exceeds vehicle capacity.

Besides the vehicle capacity, other side-constraints can be included in the VRP, such as maximum travel distance and/or travel time, time windows, mixed pick-up and delivery.

Various authors have suggested a number of solution methods for the Vehicle Routing Problem. Conceptually, it is possible to divide solution methods for the VRP into four categories: route-building methods, two-phase methods, incomplete optimization algorithms, and improvement methods. Route-construction methods contain savings and insertion heuristics, as proposed, among others, by Clarke and Wright (1964), Gaskell (1967), Mole and Jameson (1976), Savelsbergh (1990) and Altinkemer and Gavish (1991).

Two-phase methods are for instance the Sweep heuristic of Gillet and Miller (1974), the Generalized Assignment heuristic of Fisher and Jaikumar (1981) and the Two-phase heuristic of Christofides et al. (1979).

The incomplete tree-search of Christofides et al. (1979) can be considered as an incomplete optimization algorithm.

Improvement methods are assumed to enhance feasible solutions through a search mechanism. This search mechanism can follow a straight


*SSDI* 0377-2217(94)00064-J

pattern as in descent or hill climbing heuristics. Consequently, this essentially leads to improvements of the objective function towards a local optimum. Moreover, the quality of the final solution largely depends upon the quality of the initial solution.

If, on the contrary, the search rule is not systematic and if a temporary deterioration of the objective function value is accepted, it is possible to offset the above-mentioned limitations of convergence to local optima and the influence of the initial solution, peculiar to local improvement methods. Like a Tabu Search (TS) strategy, Simulated Annealing (SA) can be used for developing such search methods for global optima.

As far as the application of Tabu Search for improvement methods in Vehicle Routing is concerned, Pureza and França (1991), Gendreau et al. (1992), Osman (1991, 1993), Taillard (1992) and Semet and Taillard (1993) all obtained quite satisfactory results. With respect to the application of SA to the VRP, the research of Alfa et al. (1991), Osman (1991, 1993) and Robusté et al. (1990) have to be mentioned. Some of these implementations are used for comparative purposes later in the article.

This article proceeds with a description of the general SA process, followed by a presentation of the improvement methods used. The SA-based improvement methods are compared with their structured search alternatives and with other solution methods for the VRP on fourteen classical test problems, seven of which have restrictions as to the maximum travel distance in addition to vehicle capacity constraints.

Besides capacity and distance constraints, the improvement methods we propose have been conceived to deal with a variety of side-constraints ranging from maximum travel time to time-windows and mixed pick-up and delivery. These restrictions are beyond the scope of this article.

## 2. Description of simulated annealing

The use of SA as a workable method for finding good solutions to optimization problems was first proposed by Kirckpatrick et al. (1983) and independently by Cerny (1985). Research conducted by Metropolis et al. (1953) in the field of Statistical Mechanics served as a basis.

Since then, SA has already been applied to several types of discrete combinatorial optimization problems (Johnson et al. 1989, 1991; Golden and Skiscim, 1986, etc.). Extensive bibliographies of applications can be found in Van Laarhoven and Aarts (1987), Aarts and Korst (1988) and Collins et al. (1988).

The analogy between SA for an optimization problem and the physical annealing process of a solid is obvious. Physical annealing is aimed at obtaining a solid material in its ground state with its atoms arranged into regular patterns. The first step in the process entails melting the material followed by a gradual reduction in temperature. In order to obtain the ground state it is important that cooling should not occur too fast. The ground state of a material refers to the minimum energy configuration of its atoms. The minimum energy configuration of a material corresponds to the minimum value of the objective function of an optimization problem. A feasible solution to the optimization problem can be considered as a configuration of the material. In order to reach a thermal equilibrium at a given temperature, it is necessary for the temperature to remain stable long enough. As a result, the configurations will be distributed according to the Boltzmann distribution. Translated into SA terms, this implies that the number of feasible neighbourhood solutions generated at a certain temperature must be sufficiently high. A neighbourhood solution is a newly generated feasible solution which differs slightly from the current solution, similar to a small displacement of an atom in the material in order to obtain a new configuration. For each feasible solution the value of the objective function is calculated in accordance with the energy of a new configuration of the material. The difference between the objective function value of a newly generated neighbourhood solution and the current solution is defined by $\delta$, which is the resulting change in energy in the case of a physical annealing process. If $\delta < 0$ the neighbourhood solution becomes the current solution, but if $\delta > 0$

the newly generated feasible solution has a probability of $\exp(-\delta/(k_B T))$ of being accepted as the new current solution. $T$ represents the current temperature here and $k_B$ is known as the Boltzmann constant. As mentioned before, the number of neighbourhood solutions generated must be sufficiently high at each temperature to justify the use of the Boltzmann distribution.

With a traditional descent algorithm only those feasible solutions which lower the value of the objective function are considered. SA accepts with certain probability feasible solutions which also increase the value of the objective function. Ideally, this acceptance probability must be close to one at a high temperature at the beginning of the process and is nearly reduced to zero at a temperature close to zero near the end of the process. This prevents the SA algorithm from being trapped in a local minimum as is the case with a descent algorithm.

A major problem inherent in the application of SA is the determination of the annealing or cooling schedule. The cooling schedule consists of fixing the initial temperature, the temperature function with a cooling rate for lowering the temperature, the number of feasible solutions generated at each temperature, and, finally, a stopping rule. The problem of determining a cooling schedule has been treated extensively by Hajek (1988) and a wide range of cooling schedules has been classified by Collins et al. (1988). It is impossible to formulate the 'ideal' annealing schedule for all problems. However, some guiding rules can be considered. The magnitude of the initial temperature $T_{\text{begin}}$ must be a trade-off between the attainability of all feasible solutions and the length of the process. A rule of thumb may be to take the initial temperature equal to a number of times the maximal difference between the cost of the best and the cost of the worst available solution for the problem considered.

The cooling rate should not result in the temperature dropping too quickly, which would preclude the occurrence of the optimal solution. At each temperature the number of iterations must be sufficiently high to reach an equilibrium, but if the number of newly generated feasible solutions is too great for each temperature, the process time increases very rapidly. A stopping rule can consist of a stopping temperature, a fixed number of temperature decreases, a minimal acceptance ratio or a maximum number of total iterations. The acceptance ratio is defined as the number of feasible solutions accepted divided by the total number of feasible solutions generated.

Using the maximum number of total iterations as a stopping rule partially prevents cycling by interrupting the SA-process. Cycling occurs more frequently if the number of feasible neighbourhood moves is very limited. In such situations, repeated selections of inverse moves cannot be excluded.

## 3. Improvement heuristics for the Vehicle Routing Problem

For Vehicle Routing applications, two types of improvement methods can be distinguished: the *within route* and the *between routes* improvement methods. A within route search method tries to lower the route distance and/or time by altering the sequence of stops within a single route. This turns out to be the application of a $k$-opt (Lin, 1965; Lin and Kernighan, 1973) or Or-opt (Or, 1976) improvement to a Travelling Salesman Problem (TSP), because only a single route of the VRP solution is involved.

A *between routes* improvement considers possible exchanges or relocations of stops or strings of stops between every two routes of a feasible solution. Two of the three between routes improvement methods we used are based on concepts proposed by Savelsbergh (1988).

The first method, which we call the *String Relocation* method, tries to insert a stop or a string of stops from one route into another route. Sometimes, this improvement method is symbolically denoted as $(m, 0)$, where $m$ is a nonnegative integer representing the length of the string to be relocated from the origin route to the destination route. Fig. 1 contains an example of a (2, 0) String Relocation.

The *String Exchange* method is the second method under consideration. This method attempts to improve the solution for the VRP by

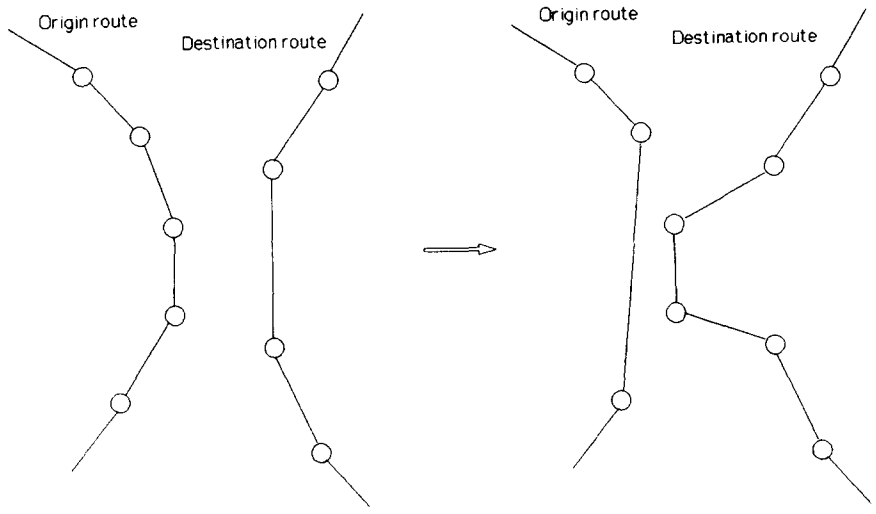


Fig. 1. Example of a String Relocation between two routes. Two nodes are relocated.

exchanging stops or strings of stops between every two routes. Symbolically, it can be represented by $(m, n)$, in which $m$ is the length of the string of the first route and $n$ that of the second route. Fig. 2 illustrates a (2, 1) exchange.

The third method, which we call the *String Mix* method, is a mixture of the previous two methods. The String Mix method tries to relocate or exchange stops or strings of stops, which of both yields the greatest saving.

The three methods proposed have been implemented as descent or as SA-based heuristics. We consider exchanges and/or relocations with a maximum stringlength of 3 stops, i.e. $m \leq 3$ and/or $n \leq 3$.

First the descent implementations of the three methods are considered. This implies that the methods are provided with a systematic search technique. A typical characteristic of the descent implementation is that only moves which result in

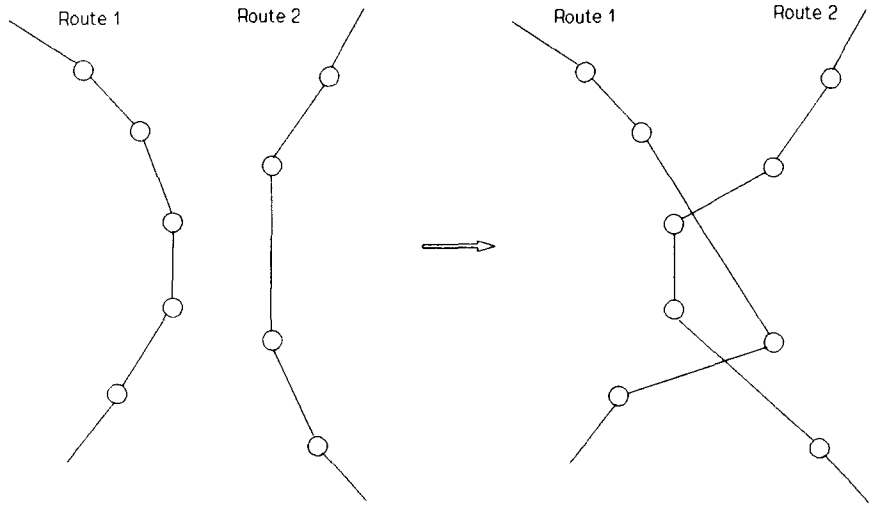


Fig. 2. Example of a String Exchange between two routes. Two nodes of Route 1 are exchanged with one node of Route 2.

an improvement of the current solution are accepted. A move is a transition from a feasible solution to another feasible solution by means of a relocation or exchange.

The general pseudo-code for the improvement methods proposed is presented in Fig. 3. The input required for the descent procedures contains an initial feasible solution, parameter string length SL and parameter $M$, which determines the improvement method to be used. The output of a descent improvement heuristic is a feasible solution which is as least as good as the initial solution.

The pseudo-code clearly illustrates the systematic search rule of the descent implementations. Evaluations of relocations and/or exchanges of strings of stops are always started with the first stop of the first route and the first stop of the second route. The second pair of routes is selected when all possible relocations and/or exchanges between the first pair of routes have been considered, taking the string length (SL) into account. Every time an improving and feasible move results it is performed and the sequences of the routes involved in the move are updated by means of a within route improvement heuristic. Next, the entire evaluation procedure is restarted with the first pair of routes, by returning to the STARTLOOP position.

In the case of the String Relocation all possible relocations of stops between any two routes are considered. The route in which the string of stops i1,...,i2 is selected, is called the origin route (Route1). The string of stops i1,...,i2 is relocated in the destination route (Route2) between the stops j1 − 1 and j1. The for-loop of j2 is not performed in this case.

The String Exchange procedure evaluates the exchange of the string of stops i1,...,i2 of one route (Route1) with the string of stops j1,...,j2 of another route (Route2).

The descent implementation of the String Mix heuristic is a superposition of the String Relocation and the String Exchange procedure. At each iteration the relocation of the string i1,...,i2 and its exchange with string j1,...,j2 is evaluated. If

```
procedure Descent improvement
(* input: an initial set of feasible routes R, SL, M *)
(* output: an improved set of feasible routes R' with cost(R') ≤ cost(R) *)

begin
  STARTLOOP
  for Route1 := 1 to R do
    for Route2 := 1 to R and Route1 <> Route2 do
      for i1 := 1 to #(Route 1) do
        for i2 := i1 to i1 + SL-1 and i2 ≤ #(Route1) do
          for j1 := 1 to #(Route2) do
            for j2 := j1 to j1 + SL − 1 and j2 ≤ #(Route2) do
              begin
                R_new = move(R, M, SL)
                if cost (R_new) < cost(R) and Feasible(R_new) = True
                  begin
                    R = R_new
                    R' = R
                    goto STARTLOOP
                  end
              end
end
```

Fig. 3. General pseudo-code for the improvement methods.

both moves constitute an improvement and are feasible, the move which results in the greater improvement is selected.

The maximum string length parameter SL considerably affects the computation time required. As was mentioned previously, we decided to limit the maximum string length to three stops. These considerations also hold for the SA-based implementations of the three improvement methods proposed.

The three procedures have been embedded in the same SA-metaheuristic whose pseudo-code is shown in Fig. 4. In addition to an initial set of feasible routes with cost $R$, the maximum string length SL and the procedure choice parameter $M$ required for the descent implementation, the input of the SA-metaheuristic is extended with a variety of parameters. Technical parameters proper to a SA-based heuristic are its starting temperature $T_{\text{begin}}$, cooling rate $R_c$, the total number of iterations $N_{\text{tot}}$ and the number of iterations at a temperature $N_{\text{temp}}$. The optional parameters added here are the acceptance ratio and the end temperature $T_{\text{end}}$. Both can be categorized as stopping-rule parameters. The parameter end temperature can be useful if the anneal-

```
procedure Simulated Annealing
(* input:  an initial set of feasible routes R, T_begin, T_end, R_c, N_temp, N_tot, Critical Ratio, M, DL, SL *)
(* output: an improved set of feasible routes R' with cost(R') ≤ cost(R) *)
begin
  T = T_begin
  n_tot = 0
  repeat
    begin
      n_temp = 0
      repeat
        repeat
          n_tot = n_tot + 1
          R_new = neighbour(R, M, DL, SL)
        until (Feasible(R_new) = True) or (n_tot = N_tot)
        if (n_tot < N_tot) then
          begin
            δ = cost(R_new) − cost(R)
            r = random(0, 1)
            if δ < 0 then
              begin
                R = R_new
                R' = R
              end
            else if r < exp(−δ/T) then
              R = R_new
            n_temp = n_temp + 1
          end
      until (n_temp = N_temp) or (n_tot = N_tot)
      T = R_c * T
    end
  until (T ≤ T_end) or (AcceptanceRatio < CriticalRatio) or (n_tot = N_tot)
end
```

Fig. 4. Pseudo-code for the Simulated Annealing metaheuristic.

ing process requires stopping at a given temperature.

The values that have to be assigned to all these technical parameters are for the greater part determined by trial-and-error-like experiments. However, some considerations can be taken into account while determining the annealing schedule, such as these: the number of feasible moves in the neighbourhood, the tightness of the problem constraints, CPU-time and the quality of the improved solution compared to the initial solution.

Additionally, some parameters are required for the SA-implementation of the proposed improvement methods. In order to avoid the generation of too many feasible solutions of bad quality, we introduced a distance limit DL which somewhat restricts the search space for feasible neighbourhood moves. As was mentioned before, parameter string length SL gives the upper limit for the number of stops in a string to be relocated and/or exchanged.

The first step in an SA process, i.e. the initialization, consists of setting the temperature $T$ to a starting temperature $T_{\text{begin}}$ and initializing the counter of the total number of iterations $n_{\text{tot}}$. During the next stage, the generation, a new feasible neighbourhood move is proposed. The way in which a new set of routes is generated depends on the kind of improvement method used, the distance limit and the string length. The three SA-based variants (the String Relocate, the String Exchange and the String Mix method) start with a random selection of two routes from the set of routes included in the current solution.

In the case of the String Relocation variant, a string containing a random number of stops is chosen in the origin route. The length of the string is limited by the maximum string length parameter SL. The string of stops from the origin route is inserted before a stop, chosen randomly from the destination route. The distance between the first stop of the origin route and the stop of the destination route may not exceed the distance limit DL.

When an acceptable string of stops in the origin route and a stop in the destination route have been found, respecting the distance limit, a feasibility test of the relocation in view of the side-constraints is performed for the destination route. If the relocation results in a violation of one or more side-constraints, new neigbourhood moves are generated until feasibility is achieved. After performing a feasible relocation, the sequence of the routes involved is updated by means of a TSP-improvement heuristic.

The String Exchange procedure is analogous to the String Relocation method, except that this time a string of stops, with a given maximum length, needs to be determined at random in two different routes. The first stop of the string in one route must be within the distance limit from the first stop of the string randomly chosen in the other route. A possible exchange of both strings requires a feasibility-check on the side-constraints in both routes concerned. The generation of exchanges continues until one results in a feasible move. If a feasible exchange is performed the sequence of both routes involved is updated.

When the String Mix procedure is used, the routes and stops are chosen as in the String Exchange procedure. When the best between relocation or exchange is chosen, a feasibility check is performed. The generation procedure is repeated until a feasible move is found.

A problem related to SA is cycling. This seems to occur when the number of feasible neighbourhood moves is very limited, which can cause a repeated selection of same, but opposite moves. If this happens, the SA-process is interrupted when the maximum number of iterations $N_{\text{tot}}$ is reached.

The parameter $n_{\text{tot}}$ is increased by one unit every time a move is generated, regardless whether it is feasible or not.

The SA process proceeds by computing the difference $\delta$ between the cost of the newly generated feasible solution $R'$ and the current solution $R$. The cost of a solution is equal to its total route distance. If $\delta < 0$, the newly generated solution $R_{\text{new}}$ is better and thus becomes the current solution $R$. If $\delta > 0$, a random number $r$ in the range $[0, 1[$ is generated. When that random number $r$ is inferior to $\exp(-\delta/T)$, the newly generated solution $R_{\text{new}}$, which is worse than the current solution $R$, is nevertheless accepted and

becomes the current solution. This temporary deterioration of the objective function value, which we may call an uphill move, permits the heuristic to leave a local optimum.

The next step in the SA process consists in the increase of the iterations counter for temperature $T$, $n_{temp}$. While this counter is less than its maximum value, $N_{temp}$, and the maximum number of iterations is not exceeded, the procedure of generating and accepting new feasible solutions is continued. Alternatively, the inner loop is closed and the temperature is decreased according to the temperature function. As long as none of the stopping criteria is met, the outer loop is restarted with the new decreased temperature $T$. If a stopping criterion is met – i.e. if the end temperature is attained or the total number of iterations equals its maximal value – the annealing process is halted.

In order to improve the effectiveness of the heuristic, we decided slightly to adapt the pure SA process by storing the best solution found so far ($R'$). This is done because there is no guarantee that the solution found at the end of the SA heuristic is the best solution found. The annealing process could be in an uphill move at the time of termination. This idea is widely accepted in the literature.

## 4. Comparison of descent and SA-based improvement heuristics

We have used the fourteen test problems fully described in Christofides, Mingozzi and Toth (1979) in order to compare the results of the descent String Relocation, String Exchange and String Mix versus their SA-based variants. All test problems proposed are subjected to capacity constraints. Additionally, problems 6, 7, 8, 9, 10, 13 and 14 have a route length constraint as well.

All tests were performed on a 80386 compatible computer. The heuristics were coded in the C++ programming language. The initial solution for all the problems have been obtained by means of the first insertion heuristic presented in Baker and Schaffer (1986).

A tabular representation of the comparison of descent and SA-based improvement methods is given in Table 1. The best solution is given with its associated string length between brackets. It should be noticed that the CPU-time given is biased because all six improvement heuristics presented are designed to handle a variety of side-constraints, as was mentioned previously. Consequently, CPU-times are overestimated due to unnecessary feasibility-checks performed by our heuristics for this test set.

Table 1 reveals some valuable results. As far as the descent implementations are concerned, dominance of the String Mix heuristic over the String Relocation and Exchange can be seen for 11 of the 14 problems considered. The string length used ranges from 1 to 3. On balance, the SA-based versions of the three improvement methods give considerably better results in comparison with their descent variants. The best overall results were obtained by means of the SA-based String Relocation and String Mix methods. The performance of the String Exchange heuristic is clearly inferior to that of the other two methods, both for the descent and for the SA-based implementations.

The annealing schedule was kept the same for all problems. The begin temperature $T_{begin}$ was determined in function of the average non-improvement value and the acceptance ratio. The cooling rate $R_c$ was set to 0.9, the total number of iterations $N_{tot}$ to 1 000 000, the total number of iterations at a temperature $N_{temp}$ to 1 000, the acceptance ratio to 0.01 and the end temperature $T_{end}$ to 1. Note, however, the huge amount of CPU-time required for the SA-based heuristics compared with their descent variants.

## 5. Comparison with other metaheuristics

In order to evaluate the relative quality of the solutions obtained with our SA-based improvement methods, we compared our best SA-based solution with other SA and Tabu Search (TS) heuristics for the VRP on the same fourteen classical test problems. A brief description of each heuristic is given.

Table 1
A comparison of descent String Relocation, String Exchange and String Mix with SA-based String Relocation, String Exchange and String Mix

| Problem | Initial | SR | SE | SM | SASR | SASE | SASM |
|---|---|---|---|---|---|---|---|
| 1 | 576 | 557 (1) | 569 (1) | 551 (1) | 548 (2) | **521** (2) | **521** (2) |
| | 1.1 | 2.2 | 1.0 | 3.0 | 2089.2 | 4023.0 | 5429.2 |
| 2 | 948 | 948 (1) | 924 (2) | 903 (2) | 856 (3) | 864 (2) | **841** (3) |
| | 1.6 | 2.0 | 3.0 | 4.0 | 2498.0 | 1636.1 | 2898.1 |
| 3 | 1017 | 892 (3) | 981 (1) | 891 (2) | **830** (1) | 878 (2) | 842 (2) |
| | 6.2 | 56.1 | 11.1 | 50.2 | 9838.8 | 8664.1 | 12908.2 |
| 4 | 1298 | 1130 (3) | 1210 (3) | 1122 (3) | **1063** (1) | 1158 (2) | 1064 (2) |
| | 17.9 | 93.1 | 46.0 | 158.1 | 7568.2 | 9704.0 | 13789.0 |
| 5 | 1551 | 1420 (2) | 1456 (3) | 1409 (2) | **1360** (3) | 1525 (2) | 1365 (3) |
| | 25.2 | 83.2 | 201.0 | 247.0 | 6906.1 | 15176.2 | 6576.1 |
| 6 | 646 | 611 (1) | 597 (3) | 567 (2) | **548** (1) | 556 (2) | 554 (2) |
| | 1.1 | 5.2 | 8.2 | 13.1 | 1030.1 | 1512.2 | 7428.3 |
| 7 | 958 | 957 (1) | 954 (1) | 952 (1) | 923 (1) | 930 (2) | **920** (2) |
| | 1.6 | 3.0 | 3.3 | 5.2 | 2687.0 | 1269.2 | 1775.2 |
| 8 | 988 | 943 (1) | 950 (2) | 937 (2) | 873 (1) | 891 (2) | **870** (2) |
| | 7.2 | 28.2 | 43.1 | 97.1 | 9791.1 | 9134.8 | 21261.0 |
| 9 | 1281 | 1226 (1) | 1232 (2) | 1202 (1) | **1197** (1) | 1217 (1) | 1207 (2) |
| | 13.2 | 72.1 | 107.0 | 145.0 | 5546.2 | 4462.0 | 6575.1 |
| 10 | 1558 | 1510 (2) | 1520 (2) | 1486 (2) | 1473 (1) | 1480 (1) | **1462** (1) |
| | 24.2 | 113.2 | 186.0 | 421.0 | 5203.0 | 6255.1 | 7327.2 |
| 11 | 1093 | 1073 (1) | 1093 (–) | 1074 (1) | **1042** (1) | 1075 (1) | 1072 (1) |
| | 16.7 | 32.2 | – | 37.1 | 28297.9 | 9485.2 | 26637.8 |
| 12 | 893 | 844 (3) | 888 (2) | 851 (2) | **821** (1) | 879 (1) | 849 (2) |
| | 6.2 | 24.2 | 15.1 | 56.2 | 4053.1 | 5296.0 | 4738.1 |
| 13 | 1671 | 1671 (1) | 1598 (3) | 1611 (3) | 1592 (1) | 1621 (2) | **1568** (1) |
| | 10.2 | 12.1 | 227.0 | 338.1 | 5368.0 | 6066.2 | 5407.2 |
| 14 | 940 | 894 (1) | 930 (1) | 890 (1) | **867** (2) | 900 (1) | 888 (2) |
| | 6.2 | 15.1 | 9.2 | 33.1 | 2014.1 | 3443.6 | 4582.1 |

The string length used is given between brackets. CPU-time in seconds is reported on the second line of each cell.
*Initial*: initial solution obtained with insertion heuristic of Baker and Schaffer (1986);
*SR*: descent String Relocation; *SE*: descent String Exchange;
*SM*: descent String Mix; *SASR*: SA-based String Relocation;
*SASE*: SA-based String Exchange; *SASM*: SA-based String Mix.
Results are given in rounded distances. The best solutions are printed in bold.

The hybrid SA/TS heuristic of Osman (1991, 1993) (OSA) differs considerably from the more classical SA-implementation, as we propose. Neighbourhood moves are generated by means of a 1-interchange mechanism, which is equivalent to our String Relocation and String Exchange procedures with maximum string length equal to 1, also denoted by (1, 0) and (1, 1). The hybrid SA/TS heuristic works with a non-monotonic cooling schedule and a systematic neighbourhood search technique, which is typical of tabu search implementations. A non-monotonic schedule is realized by resetting the current temperature to a higher initial temperature if no 1-interchange move is accepted during a cycle of systematic neighbourhood search. The only random feature involved in the search process is the generation of the sequence of the pair of routes. The stopping criterion used is based on the number of temperature resets.

The other metaheuristics involved in the comparison are all tabu search algorithms. For a complete overview on TS for combinatorial optimization, see Glover (1989, 1990) and Glover et al. (1992). A Tabu Search strategy, like a SA strategy, is able to search beyond local optima. However, a tabu search implementation is far more deterministic than a traditional SA-imple-

mentation. Tabu Search uses a systematic search technique to determine the best admissible move in the whole neighbourhood. If this move results in the improvement of the current solution, the move is performed. A non-improving move is performed if its opposite move does not occur on a tabu list. In this way, the tabu list, which needs to be updated after each iteration, prevents the cyclic revisiting of previously visited solutions.

The Tabu Search heuristic of Osman (1991, 1993) (OTS) uses the same 1-interchange mechanism to generate moves as that used for the hybrid SA/TS heuristic (OSA). A (1, 1)-interchange procedure is embedded in the parallel tabu search algorithm of Taillard (1992) (TTS). Pureza and França (1991) (PF) proposed a tabu search heuristic with a move-generation procedure, which is equivalent to our String Relocation and String Exchange with maximum string length equal to 1, i.e. (1, 0) and (1, 1). The tabu search procedure of Gendreau et al. (1992) (GHL) uses a (1, 0) search technique which relocates a stop from one route into another, comparable to our String Relocation with maximum string length equal to 1. A special feature of the latter TS metaheuristic is that it temporarily allows solutions which are not feasible with respect to the capacity and/or route distance constraints.

Table 2 reports the results of the metaheuristics on the fourteen test problems.

First, we should like to stress that we used rounded distances. Results for the other metaheuristics are reported in real distances, except for PF where no details are given.

If we compare the results of the other SA heuristic (OSA), a hybrid SA/TS heuristic with our traditional SA implementation, neither is superior. Further, we notice that our SA-heuristic gives a solution which is less good for problems 4, 5, 9, 10 and 13 compared with the best solution. For the nine remaining problems our solution equals or approximates the best solution.

The results of Table 2 confirm, to some extent, the consideration made by some authors (Osman, 1991) who find that TS outperforms SA with regard to solution quality and computation time. Comparison of CPU-times is very difficult, because different computers have been used.

Table 2
Comparison of the SA-based improvement methods with other metaheuristics for the VRP

| Problem | OSA | OTS | TTS | PF | GHL | SA |
|---|---|---|---|---|---|---|
| 1 | 524 | 524 | 524.61 | 536 | 524.61 | 521 |
| 2 | 838 | 844 | 835.26 | 842 | 835.32 | 841 |
| 3 | 830 | 835 | 826.14 | 851 | 826.14 | 830 |
| 4 | 1058 | 1044 | 1028.42 | 1081 | 1031.07 | 1063 |
| 5 | 1378 | 1334 | 1298.79 | – | 1311.35 | 1360 |
| 6 | 555 | 555 | 555.43 | 560 | 555.43 | 548 |
| 7 | 909 | 911 | 909.68 | 929 | 909.68 | 920 |
| 8 | 866 | 866 | 865.94 | 887 | 865.94 | 870 |
| 9 | 1164 | 1184 | 1162.55 | 1227 | 1162.89 | 1197 |
| 10 | 1417 | 1422 | 1397.94 | – | 1404.75 | 1462 |
| 11 | 1176 | 1042 | 1042.11 | 1049 | 1042.11 | 1042 |
| 12 | 826 | 819 | 819.56 | 826 | 819.56 | 821 |
| 13 | 1545 | 1545 | 1541.14 | 1631 | 1545.93 | 1568 |
| 14 | 890 | 866 | 866.37 | 866 | 866.37 | 867 |

Finally, we can compare our SA-based method with the 3-opt based SA-algorithm of Alfa, Heragu and Chen (1991), who only reported solutions for problems 1 and 2. They obtained a cost of 556.0 and 892.3, respectively, for problems 1 and 2, for which we obtained 521 and 841.

## 6. Conclusion

The primary contribution of this paper was to show that even a traditional SA-implementation yields very satisfactory solutions. Compared with their hill-climbing variants, the SA-based improvement methods generally give considerably better solutions. With respect to other metaheuristics for the VRP, our SA-metaheuristics equalled the best solution published for several problems. In general, the excessively high CPU-time remains a major drawback to SA-based methods.

Our results were obtained through the application of the same annealing schedule. However, we are convinced that further experiments with alternative annealing schedules may result in even better solutions.

## Acknowledgement

The author wishes to thank a referee for his valuable comments.

## References


Aarts, E.H.L., and Korst, J.H.M. (1988), *Simulated Annealing and Boltzmann Machines*, Wiley, Chichester.

Alfa, A.S., Heragu, S.S., and Chen, M. (1991), "A 3-opt based simulated annealing algorithm for vehicle routing problems", *Computers and Industrial Engineering* 21, 635–639.

Altinkemer, K., and Gavish, B. (1991), "Parallel savings based heuristic for the delivery problem", *Operations Research* 39, 456–469.

Baker, E.K., and Schaffer, J.R. (1986), "Solution improvement heuristics for the vehicle routing and scheduling problem with time window constraints", *American Journal of Mathematical and Management Sciences* 6, 261–300.

Cerny, V. (1985), "A thermodynamical approach to the travelling salesman problem: An efficient simulation algorithm", *Journal of Optimization Theory and Application* 45, 41–51.

Christofides, N., Mingozzi, A., and Toth, P. (1979), "The vehicle routing problem", in: N. Christofides, A. Mingozzi, P. Toth and C. Sandi (eds.), *Combinatorial Optimization*, Wiley, Chichester.

Clarke, G., and Wright, J.W. (1964), "Scheduling of vehicles from a central depot to a number of delivery points", *Operations Research* 12, 568–581.

Collins, N.E., Eglese, R.W., and Golden B.L. (1988), "Simulated annealing – An annotated bibliography", *American Journal of Mathematical and Management Sciences* 8, 209–307.

Connolly, D.T. (1988), "An improved annealing scheme for the QAP", *European Journal of Operational Research* 46, 93–100.

Eglese, R.W. (1990), "Simulated annealing: A tool for operational research", *European Journal of Operational Research* 46, 271–281.

Fisher, M.L., and Jaikumar, R. (1981), "A generalized assignment heuristic for vehicle routing", *Networks* 11, 109–124.

Gaskell, T. (1967), "Bases for vehicle fleet scheduling", *Operational Research Quarterly* 18, 367–384.

Gendreau, M., Hertz, A., and Laporte, G. (1992), "A tabu search heuristic for the vehicle routing problem", Working Paper ORWP 92/14, Département de Mathématiques, Ecole Polytechnique Fédérale de Lausanne.

Gillet, B., and Miller, L. (1974), "A heuristic algorithm for the vehicle dispatch problem", *Operations Research* 22, 340–349.

Glover, F. (1989), "Tabu search, Part I", *ORSA Journal on Computing* 1, 190–206.

Glover, F. (1990), "Tabu search, Part II", *ORSA Journal on Computing* 2, 4–32.

Glover, F., Taillard, E., and de Werra, D. (1992), "A user's guide to tabu search", Working Paper ORWP 92/101, Département de Mathématiques, Ecole Polytechnique Fédérale de Lausanne.

Golden, B.L., and Skiscim, C.C. (1986), "Using simulated annealing to solve routing and location problems", *Naval Research Logistics Quarterly* 33, 261–279.

Hajek, B. (1988), "Cooling schedules for optimal annealing", *Mathematics of Operations Research* 13, 311–329.

Johnson, D.S., Aragon, C.R., McGeoch, L.A., and Schevon, C. (1989), "Optimization by simulated annealing: An experimental evaluation; Part I, Graph partitioning", *Operations Research* 37, 865–892.

Johnson, D.S., Aragon, C.R., McGeoch, L.A., and Schevon, C. (1991), "Optimization by simulated annealing: An experimental evaluation; Part II, Graph coloring and number partitioning", *Operations Research* 39, 378–406.

Kirckpatrick, S., Gelatt, Jr., C.D., and Vecchi, M.P. (1983), "Optimization by simulated annealing", *Science* 220, 671–680.

Lin, S. (1965), "Computer solutions of the traveling salesman problem", *Bell Systems Computer Journal* 44, 2245–2269.

Lin, S., and Kernighan, B.W. (1973), "An effective heuristic algorithm for the traveling salesman problem", *Operations Research* 21, 498–516.

Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A., and Teller, E. (1953), "Equation of state calculations by fast computing machines", *The Journal of Chemical Physics* 21, 1087–1092.

Mole, R.H., and Jameson, S.R. (1976), "A sequential route-building algorithm employing a generalized savings criterion", *Operational Research Quarterly* 27, 503–511.

Or, I. (1976), "Traveling salesman-type combinatorial optimization problems and their relation to the logistics of regional blood banking", Ph.D. Dissertation, Northwestern University, Evanston, IL.

Osman, I.H. (1991), "Metastrategy simulated annealing and tabu search algorithms for combinatorial optimization problems", Ph.D. Dissertation, The Management School, Imperial College, London.

Osman, I.H. (1993), "Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem", *Annals of Operational Research* 41, 421–451.

Pureza, V.M., and França, P.M. (1991), "Vehicle routing problems via tabu search metaheuristic", Publication CRT-747, Centre de recherche sur les transports, Montréal.

Robusté, F., Daganzo, C.F., and Souleyrette, R.R. (1990), "Implementing vehicle routing models", *Transportation Research* B 24, 263–286.

Savelsbergh, M.W.P. (1988), "Computer aided routing", Ph.D. Dissertation, Erasmus Universiteit Rotterdam.

Savelsbergh, M.W.P. (1990), "A parallel insertion heuristic for vehicle routing with side constraints", *Statistica* Neerlandica 44, 139–148.

Semet, F., and Taillard, E. (1993), "Solving real-life vehicle routing problems efficiently using taboo search", *Annals of Operations Research* 41, 469–488.

Taillard, E. (1992), "Parallel iterative search methods for vehicle routing problems", Working Paper ORWP 92/03, Département de Mathématiques, Ecole Polytechnique Fédérale de Lausanne.

Van Laarhoven, P.J.M., and Aarts, E.H.L. (1987), *Simulated Annealing: Theory and Practice*, Kluwer Academic Publishers, Dordrecht.