J Heuristics (2006) 12: 55–72
DOI 10.1007/s10732-006-4192-1

Path relinking for the vehicle routing problem

Sin C. Ho · Michel Gendreau

Submitted in October 2004 and accepted by David Woodruff in August 2005 after 2 revisions

Abstract This paper describes a tabu search heuristic with path relinking for the vehicle routing problem. Tabu search is a local search method that explores the solution space more thoroughly than other local search based methods by overcoming local optima. Path relinking is a method to integrate intensification and diversification in the search. It explores paths that connect previously found elite solutions. Computational results show that tabu search with path relinking is superior to pure tabu search on the vehicle routing problem.

Keywords Vehicle routing · Tabu search · Path relinking

The *Vehicle Routing Problem* (VRP) is defined on a complete undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{0, 1, \ldots, n\}$ is the set of vertices and $\mathcal{E} = \{(i, j) : i, j \in \mathcal{V}, i < j\}$ is the edge set. It should be mentioned that we use both the notation (i, j) and (j, i) to refer to edge (i, j), where $i < j$. Vertices $1, \ldots, n$ represent customers; with customer i are associated a nonnegative demand d_i and a nonnegative service duration t_i. Vertex 0 is the depot at which is based a fleet of m homogenous vehicles of capacity q. The fleet size is treated as a decision variable. To each edge (i, j) is associated a traveling cost or travel time c_{ij}; travel costs satisfy the triangle inequality. The VRP consists of designing m vehicle routes on $\mathcal{G}$ such that (i) every route begins and ends at the depot; (ii) every customer is visited exactly once; (iii) the total demand of any vehicle route does not exceed q; (iv) the total duration (i.e., the total length of the edges traversed plus the service times of served customers) of any vehicle route

S.C. Ho (✉)
Department of Informatics, University of Bergen, Postboks 7800, N-5020 Bergen, Norway
e-mail: sin@ii.uib.no
tel: +47 55 58 40 94

M. Gendreau
Centre de recherche sur les transports and Département d'informatique et de recherche opérationnelle, Université de Montréal, Canada
e-mail: michelg@crt.umontreal.ca

does not exceed a given limit L; and (v) the total cost of all vehicle routes is minimized. A comprehensive survey of solution techniques for the VRP can be found in the book edited by (Toth and Vigo, 2002).

While many heuristic approaches have been used to successfully tackle the VRP during the past five decades (Alfa, Heragu, and Chen, 1991; Kawamura et al., 1998; Bullnheimer, Hartl, and Strauss, 1998; Bullnheimer, Hartl, and Strauss, 1999; Reimann, Doerner, and Hartl, 2004; Berger and Barkaoui, 2003; Prins, 2004; Tarantilis, 2005), Tabu Search (TS) has been the most widely applied technique to provide good approximate solutions for the problem. This method was originally proposed by Glover (1986) as a local search technique that does not terminate when it encounters local optima, but rather proceeds with moves that degrade the objective function. Cycling is prevented by the use of tabu lists in which the recent history of the search trajectory is recorded. An early TS approach for the VRP was proposed by Osman (1993) who introduced the λ-*interchange* operator, which consists in swapping subsets of at most λ customers between two routes. A rather more involved TS heuristic of the early 90's is *Taburoute* (Gendreau, Hertz, and Laporte, 1994). In Taburoute, the neighborhood operator consists in removing a vertex from its current route, inserting it in a route containing one of its closest neighbors (this could be the same route), and performing a local reoptimization of this route. This method contains some innovative features like the temporary acceptance of infeasible solutions and self-adjusting penalty parameters for infeasible solutions. Taillard (1993) proposed a TS algorithm using a decomposition method that makes it possible to benefit from parallel computing. To this date, this algorithm remains one of the most effective TS heuristic for the VRP. A highly effective method for generating good and diverse solutions is the adaptive memory procedure proposed by Rochat and Taillard (1995). It keeps track of good solutions found during the search and uses them as a basis for the construction of new ones. Xu and Kelly (1996) proposed a TS heuristic based on a network flow model, and defined the neighborhoods by ejection chains. Rego and Roucairol (1996) also used ejection chain based neighborhoods; parallel computing allowed for a more extensive exploration of the search space. Cordeau, Laporte, and Mercier (2001) proposed a simple and flexible TS heuristic— *Unified Tabu Search*. It should be noted that the Unified Tabu Search heuristic is applicable to problems with or without time windows. Its neighborhood structure relies on the *relocate* operator that simply moves one vertex from one route to another. Like Taburoute, it allows intermediate infeasible solutions. The granularity concept proposed by Toth and Vigo (2003) is a candidate list strategy whose purpose is to limit the size of the neighborhood by permanently removing long edges that have a small probability of belonging to an optimal solution. This method is fast and capable of generating very good solutions. For surveys on VRP metaheuristics, the reader is referred to Gendreau, Laporte, and Potvin (2002), Cordeau and Laporte (2004) and Cordeau et al. (2005).

While TS has been widely used in VRP heuristics for several years, Path Relinking is a fairly new approach introduced just a few years ago (Glover and Laguna, 1993), and has not yet been applied to the VRP. However, it has been applied to other combinatorial problems with great success (Aiex, Binato, and Resende, 2003; Aiex et al., 2005; Ghamlouche, Crainic, and Gendreau, 2004; Oliveira, Pardalos, and Resende, 2004; Resende and Ribeiro, 2003; Souza, Duhamel, and Ribeiro, 2003).

In this paper, we present a new TS heuristic for the VRP that makes use of path relinking as an intensification and diversification mechanism. The paper is organized as follows. In Section 1, we present a basic TS heuristic for the VRP. A brief review of path relinking and how it is combined with TS to solve the VRP is given in Section 2. Experimental results

showing the improvements in the performance of the TS heuristic when path relinking is used are presented in Section 3. Finally, Section 4 concludes the paper.

1. The basic tabu search heuristic

Tabu search is a memory-based search strategy that allows the local search process to proceed beyond local optima. This is achieved by allowing the objective function to deteriorate when the current solution is a local optimum, and by keeping track of recent moves or solutions in a so-called tabu list. Whenever the algorithm attempts to move to a solution or to perform a move recorded in the tabu list, the move is banned. This rule prevents cycling and forces other solutions to be explored. However, this feature is not strict, as it can be overridden when some aspiration criterion is satisfied. A commonly used criterion is that the objective function value of a tentative solution be the best ever seen. If this is the case, the search may be allowed to proceed to this tentative solution, since it has obviously never been encountered before, which guarantees that cycling cannot occur.

1.1. Notation of the heuristic

As in Gendreau, Hertz, and Laporte (1994), we allow the search to be conducted in the infeasible part of the solution space. We let $\mathcal{X}$ denote the set of solutions satisfying constraints (i) and (ii). Each solution $x \in \mathcal{X}$ consists of m vehicle routes starting and ending at the depot, such that every customer is visited exactly once. This solution may violate the capacity and duration constraints.

For a solution x, let $c(x)$ denote its travel cost, and let $q(x)$ and $t(x)$ denote the total violation of the load and duration constraints, respectively. The routing cost of a vehicle k corresponds to the sum of the costs c_{ij} associated with the edges (i, j) traversed by this vehicle. The total violation of capacity and duration constraints is computed on a route by route basis with respect to q and t. Each solution x is evaluated by a cost function $z(x) = c(x) + \alpha q(x) + \beta t(x)$, where α and β are self-adjusting positive parameters.

The number of vehicles available is assumed to be unlimited, but as a starting number, $\underline{m}$, it is set to the minimum possible subject to the capacity constraints, i.e. $\underline{m} = \lceil \sum_{i \in \mathcal{V} \setminus \{0\}} d_i / q \rceil$. However, this number may be increased if it will lead to cost savings during the search process.

1.2. Initial solution

A total of I initial solutions are generated by a stochastic insertion heuristic, which works as follows. The routes are initialized by randomly selecting $\underline{m}$ seed customers. Each route only services a single customer. The remaining unrouted customers are then inserted one by one (in a random order) at the location that minimizes the cost of inserting this customer over the current set of routes. The solutions thus obtained are usually not very good ones, therefore they need to be improved before proceeding any further. To do so, a short tabu search with the relocate neighborhood operator is applied to each solution for μ iterations. This tabu search is identical to the one used for the main phase. The best solution among these I solutions is chosen to initiate the main search process.

1.3. Neighborhood structure

The neighborhood structure is based on the relocate operator: the neighborhood of solution x, denoted $\mathcal{N}(x)$, is made up of all solutions that can be reached from x by moving a customer i from its route k to another route l, with $k \neq l$. Such a move is denoted by (k, i, l).

1.4. Recency based memory and tabu tenure

To avoid cycling whenever a move (k, i, l) is performed, any move that transfers i back into route k is declared tabu for θ iterations, where θ is a user defined parameter. A tabu move will still be chosen if it satisfies the aspiration criterion of improving the best known solution.

1.5. Frequency-based memory

To diversify the search and to induce it to explore a wider part of the solution space, frequently made moves are penalized. The frequency associated with move (k, i, l) is the number of times customer i has been moved to vehicle l. For any solution $\bar{x} \in \mathcal{N}(x)$, whenever $z(\bar{x}) \geq z(x)$, a penalty is added to $z(\bar{x})$. Only moves that lead to non-improving solutions are penalized since it makes no sense to penalize improving moves. The penalty $\phi(\bar{x})$ is defined as $\lambda c(\bar{x})\sqrt{nm'}\vartheta_{ik}$, where ϑ_{ik} denotes the number of times customer i has been moved to vehicle k during the search so far, m' denotes the number of non-empty vehicles in solution $\bar{x}$, and λ is an user defined parameter that controls the intensity of diversification.

1.6. Search process

The tabu search heuristic starts off with the initial solution defined in section 1.2. At each iteration, the least cost non-tabu solution $\bar{x}$ is selected from $\mathcal{N}(x)$. Then parameters α and β are modified. Parameter α is adjusted as follows: if there is no violation of the capacity constraints, the value of α is divided by $1 + \delta$, otherwise it is multiplied by $1 + \delta$, where δ is a positive parameter. A similar rule applies also to β with respect to route duration constraints. This process terminates after γ iterations, and is summarized below.

Algorithm 1 Tabu search

If x is feasible, set $x^* = x$ and $c(x^*) = c(x)$; otherwise set $c(x^*) = \infty$.
Set $\alpha = 1$ and $\beta = 1$.
for $i = 1$ to γ **do**
  Select a solution $\overline{x} \in \mathcal{N}(x)$ that minimizes $z(\overline{x}) + \phi(\overline{x})$ and is non-tabu or satisfies the aspiration criterion.
  If $\overline{x}$ is feasible and $c(\overline{x}) < c(x^*)$, set $x^* = \overline{x}$ and $c(x^*) = c(\overline{x})$.
  Set the reverse move tabu for θ iterations.
  Compute $q(\overline{x})$ and $t(\overline{x})$, and update α and β.
  Set $x = \overline{x}$.
return x^*

2. Path relinking

Path relinking was first introduced by Glover and Laguna (1993) in connection with TS as a way of exploring trajectories between elite solutions. The fundamental idea behind this

method is that good solutions to a problem should share some characteristics. By generating paths (i.e., sequences of intermediate solutions) between elite solutions, one could reasonably hope to find better ones. A more thorough description of Path relinking can be found in Glover (1997, 1998) and Glover, Laguna, and Martí (2000).

Path relinking can be interpreted as an evolutionary method where solutions are generated by combining elements from other solutions. Unlike other evolutionary procedures, such as genetic algorithms, where randomness is a key factor in the creation of offsprings from parent solutions, path relinking utilizes systematic, deterministic rules for combining solutions. To generate the desired paths, an *initial solution* and a *guiding solution* are chosen in a so-called *reference set* of elite solutions to represent the starting and the ending points of the path. Attributes from the guiding solution are gradually introduced into the intermediate solutions, so that these solutions contain less characteristics from the initial solution and more from the guiding solution as one moves along the path.

Any path relinking implementation revolves around the following three components that are critical in the design of the algorithm:

- Rules for building the reference set,
- Rules for choosing the initial and guiding solutions,
- A neighborhood structure for moving along paths.

2.1. Building the reference set

The quality (w.r.t. to the objective function of the problem) and the level of diversity of the solutions included in the reference set $\mathcal{R}$ have a major impact on the quality of the generated solutions. In our method, $\mathcal{R}$ is built during TS and enriched during the path relinking phase. We consider five strategies, originally proposed by Ghamlouche, Crainic, and Gendreau (2004), for building $\mathcal{R}$ in the context of network design problems:

Strategy S_1: $\mathcal{R}$ is built with the solutions that at some point during TS become the best overall solution. Here, the idea is to link the overall improving solutions.

Strategy S_2: $\mathcal{R}$ contains the best local minima encountered during the TS phase. This strategy is motivated by the fact that local minimum solutions should share some common characteristics with optimum solutions.

Strategy S_3: This strategy selects $\mathcal{R}$-improving local minima, i.e., local minimum solutions that have a better objective function value than those already in $\mathcal{R}$. The idea here is to introduce the time aspect into the selection process: since usually the better solutions are encountered when the search has been proceeding for some time, this strategy considers less local minima obtained at that stage and thus retains potentially good solutions found early during the search.

Strategy S_4: This strategy accounts both for the attractiveness and the diversity, or *dissimilarity*, of a potential solution when deciding whether or not it should be included in the reference set. Define D_s^b, the level of dissimilarity between solution s and the best solution b, as the number of different edges between the two solutions:

$$D_s^b = \sum_{(i,j)\in\mathcal{E}} h_{ij},$$

where

$$h_{ij} = \begin{cases} 1, & \text{if } (i, j) \text{ is an edge of either solution } s \text{ or solution } b, \text{ but not both;} \\ 0, & \text{otherwise.} \end{cases}$$

This can also be viewed as the Hamming distance between solutions s and b if the two solutions are encoded as binary vectors in which the k-th element is 1 iff the edge number k is used by the solution.
We also define the median position of all solutions $x \in \mathcal{R}$ relatively to the best solution b as:

$$Median = \frac{\sum_{x \in \mathcal{R}}^{x \neq b} D_x^b}{|\mathcal{R}| - 1},$$

where $|\mathcal{R}|$ denotes the number of solutions in the reference set. A solution s is included in $\mathcal{R}$ if the solution value of s is better than the value of the best solution b, or if it is better than the solution value of the worst solution in $\mathcal{R}$ and its level of dissimilarity exceeds the median, $D_s^b > Median$. In both cases, the worst solution in $\mathcal{R}$ is replaced by s.

Strategy S_5: This strategy ensures both the quality and diversity of the solutions when building $\mathcal{R}$. Laguna and Armentano (2005) pointed out that this strategy was one of the important lessons they learned in how a reference set should be updated, and is a standard technique for generating the reference set used in scatter search. Starting with a large set of good solutions $\mathcal{P}$, $\mathcal{R}$ is then partially filled with the best solutions found in $\mathcal{P}$ to ensure the quality of the solutions. $\mathcal{R}$ is then extended with solutions that differ significantly from those already in $\mathcal{R}$. The procedure to implement this strategy can be described as follows:

1. Fill $\mathcal{R}$ with $\min\{|\mathcal{R}|, \mathcal{R}_{\max}\}/2$ solutions satisfying strategy S_1, where $\mathcal{R}_{\max}$ is the maximum number of solutions in $\mathcal{R}$.
2. For each solution $s \in \{\mathcal{P} \backslash \mathcal{R}\}$, calculate the level of diversity $\Delta_s^{\mathcal{R}}$ between solution s and all solutions $x \in \mathcal{R}$ where $\Delta_s^{\mathcal{R}} = \sum_{x \in \mathcal{R}} D_x^s / |\mathcal{R}|$.
3. Extend $\mathcal{R}$ with solutions $s \in \{\mathcal{P} \backslash \mathcal{R}\}$ that maximize $\Delta_s^{\mathcal{R}}$.

2.2. Choosing the initial and guiding solutions

The choice of initial and guiding solutions is as important as the contents of $\mathcal{R}$ for the path relinking phase, since the quality of the new generated solutions, and thus the performance of the method, is highly dependent upon these solutions. We followed again the suggestions of Ghamlouche, Crainic, and Gendreau (2004) and examined the impact of the following five selection criteria:

C_1: The guiding and initial solutions are defined as the *best* and *worst* solutions in $\mathcal{R}$, respectively.
C_2: The guiding solution is chosen to be the *best* solution in $\mathcal{R}$, while the initial solution is the *second best* one.
C_3: The guiding solution is chosen as the *best* solution in $\mathcal{R}$, while the initial solution is defined as the solution with *maximum Hamming distance* from the guiding solution.
C_4: The guiding and initial solutions are chosen *randomly* in $\mathcal{R}$.
C_5: The guiding and initial solutions are chosen as the *most distant* solutions in $\mathcal{R}$.

2.3. Moving along paths

The aim of the path relinking phase is to introduce progressively attributes of the guiding solution into solutions obtained by moving away from the initial solution. Identical parts of the two solutions should remain unchanged during the process. In the context of the VRP, it is not always obvious to identify identical parts of the initial and guiding solutions, because similar solutions may exhibit a different numbering of the routes. For instance, route 1 of the initial solution might correspond to route 2 of the guiding solution, with perhaps a few differences. We must therefore make sure that similarities and differences in the structure of the initial and guiding solutions can be properly identified. To do so, once the initial and guiding solutions have been selected, we perform a matching of their routes. The matching procedure amounts to solving an *Assignment Problem* on an auxiliary complete bipartite graph $\mathcal{G}' = (\mathcal{V}', \mathcal{E}')$, where $\mathcal{V}' = \mathcal{V}'_i \cup \mathcal{V}'_g$ and the vertices of $\mathcal{V}'_i$ and $\mathcal{V}'_g$ correspond respectively to the routes of the initial and of the guiding solutions. To each edge $(k, l) \in \mathcal{E}'$ is associated a weight c_{kl}, which is defined as the number of identical customers in routes k and l of the two solutions. We look for a maximum weight matching in $\mathcal{G}'$, i.e., we want to find a matching of the routes of the two solutions such that the number of identical customers in matched routes is maximized. This problem is solved using a greedy approach that can be described as follows:

Compute the weights c_{kl} for all edges $(k, l) \in \mathcal{E}'$.
Set $\mathcal{W} = \mathcal{E}'$.
Set $\mathcal{J} = \emptyset$.
repeat
  Choose the pair of routes $(k, l) \in \mathcal{W}$ with largest weight c_{kl}.
  Set $\mathcal{J} = \mathcal{J} \cup (k, l)$.
  Delete from $\mathcal{W}$ all edges incident to vertex $k \in \mathcal{V}'_i$ and to vertex $l \in \mathcal{V}'_g$.
until $\mathcal{W} = \emptyset$.

In the path relinking phase, we must make sure that the algorithm is making progress towards the guiding solution. To do so, we use two neighborhoods $\mathcal{N}_1(x)$ and $\mathcal{N}_2(x)$. To simplify the exposition, let us assume that the routes of the guiding solution have been relabeled in accordance with the matching of the routes determined previously, i.e., if $(k, l) \in \mathcal{J}$, we now assign label k to route l of the guiding solution. The first neighborhood, $\mathcal{N}_1(x)$, is made up of all the potential solutions that can be reached from x by moving customers from their current route to another while taking into account the structure of the guiding solution. More precisely, a customer i is eligible to be moved from its current route k if it does not belong to route k in the guiding solution; it will then be relocated into the route l to which it belongs in the guiding solution. $\mathcal{N}_1(x)$ thus contains solutions that are closer to the guiding solution than the current solution x. The second neighborhood, $\mathcal{N}_2(x)$ is defined similarly as the set of all potential solutions that can be reached from x by exchanging two customers i and j between their respective routes while taking into account the structure of the guiding solution. Again, customer i is eligible to be moved from its current route k only if it belongs to a different route l in the guiding solution; it may be swapped with any customer j of route l that does not also belong to route l in the guiding solution. As the neighborhoods of solution x could be fairly restricted if they were limited only to feasible solutions, we allow tunneling through infeasible regions of the solution space. It should be noted that self-adjusting penalties are initialized every time path relinking is started. Algorithm 2 summarizes the path relinking procedure.

Algorithm 2 Path relinking

Require: $\mathcal{R}$, the reference set; x^*, the current best known solution
  Set $\nu = 0$, $\alpha = 1$ and $\beta = 1$.
  repeat
    Select the initial solution, s_i, and the guiding solution, s_g, according to criterion C_*.
    Determine the maximum weight matching of the routes of s_i and s_g.
    Compute $D_{s_g}^{s_i}$.
    Set $x = s_i$.
    repeat
      Select a solution $\overline{x} \in \mathcal{N}_1(x) \cup \mathcal{N}_2(x)$ that minimizes $z(\overline{x})$.
      If $\overline{x}$ is feasible and $c(\overline{x}) < c(x^*)$, set $x^* = \overline{x}$ and $\mathcal{R} = \mathcal{R} \cup \{x^*\}$.
      Compute $q(\overline{x})$ and $t(\overline{x})$, and update α and β.
      Increment ν by 1.
      Set $x = \overline{x}$.
    until $\nu \geq D_{s_g}^{s_i}$ or $x = s_g$.
    Remove s_i from $\mathcal{R}$.
  until $|\mathcal{R}| \leq 1$
  return x^*

2.4. Tabu search with path relinking

In the following, we show how tabu search and path relinking are combined in our implementation. Algorithm 3 describes the resulting procedure.

Algorithm 3 Tabu search with path relinking

  Set $\mathcal{R} = \emptyset$.
  Construct an initial solution. Let x be this solution.
  If x is feasible, set $x^* = x$ and $c(x^*) = c(x)$; otherwise set $c(x^*) = \infty$.
  Set $\alpha = 1$ and $\beta = 1$.
  for $i = 1$ to γ **do**
    Select a solution $\overline{x} \in \mathcal{N}(x)$ that minimizes $z(\overline{x}) + \phi(\overline{x})$ and is non-tabu or satisfies the aspiration criterion.
    If $\overline{x}$ satisfies S_*, set $\mathcal{R} = \mathcal{R} \cup \{\overline{x}\}$.
    If $\overline{x}$ is feasible and $c(\overline{x}) < c(x^*)$, set $x^* = \overline{x}$ and $c(x^*) = c(\overline{x})$.
    Set the reverse move tabu for θ iterations.
    Compute $q(\overline{x})$ and $t(\overline{x})$, and update α and β.
    Set $x = \overline{x}$.
    if $\mod(i, \varphi) = 0$ **then**
      $r =$ `PathRelinking`$(\mathcal{R}, x^*)$.
      If $c(r) < c(x^*)$, set $x^* = r$.
    end if
  return x^*

Initially, the reference set $\mathcal{R}$ is empty, and is extended with solutions according to one of the strategies of Section 2.1. Before path relinking is started, $\mathcal{R}$ is sorted and the $\mathcal{R}_{\max}$ best solutions are chosen to be retained in the set. Path relinking is performed every φ iterations of the main TS loop. One round of path relinking consists of generating several paths with different initial and guiding solutions from $\mathcal{R}$. The initial and guiding solutions are chosen according to one of the criteria listed in Section 2.2. Long paths are favored, since they will have a better chance of producing good solutions. After path relinking is finished, TS

continues with the solution it had before path relinking was triggered. The whole process terminates after γ iterations.

3. Computational experiments

Standard benchmark instances for the VRP were used for experimentation. These include the fourteen classical Euclidean VRP and distance-constrained VRP instances described in Christofides and Eilon (1969) and Christofides, Mingozzi, and Toth (1979) and the twenty large scale instances described in Golden et al. (1998). The characteristics of these instances are given in Tables 1 and 2, respectively. For each problem instance, the table indicates the number of customers (n), the vehicle capacity (q), the route maximum length (L) and the service time (t) for each customer. The table also gives for each problem instance the best known solution so far and the reference to where this solution can be found.

The results given in this paper were obtained with the following parameter settings: $\mathcal{R}_{\max} = 10$, $\delta = 0.5$, $I = 5$, $\mu = 100$, $\theta = [7.5 \log_{10} n]$ (where $[y]$ is the nearest integer to y), $\lambda = 0.015$, $\varphi = 10,000$ and $\gamma = 100,000$. The values chosen for parameters δ, θ and λ were taken from Cordeau, Laporte, and Mercier (2001) for a similar tabu search heuristic. The frequency of path relinking, φ, was determined by running every benchmark instance once for each of the predetermined values of φ and for three different combinations of criteria and strategies. This calibrating process is important because if path relinking is performed too frequently, the search will tend to focus too much on a small portion of the search space. On the opposite, if it is performed very rarely, its impact will be negligible. It is important to find a balance between these two extremes. Table 3 shows the average deviations from the best known solutions for the different values of φ, and it identifies $\varphi = 10,000$ as the value yielding the best results. Thus this value was used in all the experiments reported in the remainder of this section. The other parameters were not tuned prior to the experimentation.

Table 1 Characteristics of the benchmarks instances used for computational experiments

Problem	n	q	L	t	Best known	Ref.
1	50	160	∞	0	524.61	Taillard (1993)
2	75	140	∞	0	835.26	Taillard (1993)
3	100	200	∞	0	826.14	Taillard (1993)
4	150	200	∞	0	1028.42	Taillard (1993)
5	199	200	∞	0	1291.29	Mester and Bräysy (2005)
6	50	160	200	10	555.43	Taillard (1993)
7	75	140	160	10	909.68	Taillard (1993)
8	100	200	230	10	865.94	Taillard (1993)
9	150	200	200	10	1162.55	Taillard (1993)
10	199	200	200	10	1395.85	Rochat and Taillard (1995)
11	120	200	∞	0	1042.11	Taillard (1993)
12	100	200	∞	0	819.56	Taillard (1993)
13	120	200	720	50	1541.14	Taillard (1993)
14	100	200	1040	90	866.37	Taillard (1993)

Table 2 Characteristics of the large scale benchmarks instances used for computational experiments

Problem	n	q	L	t	Best known	Ref.
1	240	550	650	0	5627.54	Mester and Bräysy (2005)
2	320	700	900	0	8447.92	Prins (2004)
3	400	900	1200	0	11036.22	Prins (2004)
4	480	1000	1600	0	13624.52	Prins (2004)
5	200	900	1800	0	6460.98	Prins (2004)
6	280	900	1500	0	8412.80	Prins (2004)
7	360	900	1300	0	10195.56	Mester and Bräysy (2005)
8	440	900	1200	0	11663.55	Mester and Bräysy (2005)
9	255	1000	∞	0	583.39	Mester and Bräysy (2005)
10	323	1000	∞	0	742.03	Mester and Bräysy (2005)
11	399	1000	∞	0	918.45	Mester and Bräysy (2005)
12	483	1000	∞	0	1107.19	Mester and Bräysy (2005)
13	252	1000	∞	0	859.11	Mester and Bräysy (2005)
14	320	1000	∞	0	1081.31	Mester and Bräysy (2005)
15	396	1000	∞	0	1345.23	Mester and Bräysy (2005)
16	480	1000	∞	0	1622.69	Mester and Bräysy (2005)
17	240	200	∞	0	707.79	Mester and Bräysy (2005)
18	300	200	∞	0	998.73	Mester and Bräysy (2005)
19	360	200	∞	0	1366.86	Mester and Bräysy (2005)
20	420	200	∞	0	1821.15	Mester and Bräysy (2005)

Table 3 Average deviation from best for different values of φ

φ	S_2, C_1 (%)	S_3, C_5 (%)	S_4, C_3 (%)
5,000	1.05	1.16	1.08
10,000	0.62	0.54	0.64
20,000	0.97	0.83	1.13
30,000	0.70	0.80	1.05
40,000	0.77	0.79	0.82

The heuristic was coded in C++ and all experiments were performed on a Pentium 4, 2.53 GHz computer.

We want to investigate the benefits of integrating path relinking into tabu search. This is done by performing for each instance a single run of the pure tabu search heuristic, as well as of the different variants of tabu search with path relinking. The main results obtained are reported in Tables 4 and 5. In these tables, the first column gives the name of the respective benchmark instances while the results for pure tabu search runs of 100,000 iterations and 200,000 iterations are displayed in the two next columns, and CPU times (in minutes) for 200,000 iteration TS runs in the following one. The column $TS + PR$ describes the results for the best combination of strategy and criterion, S_3 and C_5, when running tabu search for

Table 4 TS and TS with PR results for classical benchmark instances

Problem	TS 100,000	TS 200,000	Time	TS + PR	Time
1	527.98	527.98	1.22	**524.61**	0.78
2	835.89	835.89	3.74	836.37	1.94
3	831.16	828.98	4.36	828.26	2.41
4	1045.31	1042.01	11.26	1034.08	5.53
5	1318.47	1315.69	21.08	1311.78	10.47
6	555.82	**555.43**	1.13	**555.43**	0.75
7	911.41	910.05	2.58	**909.68**	1.64
8	867.37	**865.94**	4.13	866.71	2.48
9	1189.70	1179.53	10.98	1177.01	6.77
10	1429.23	1424.31	17.90	1420.66	10.55
11	1122.80	1047.01	6.86	1042.97	3.60
12	821.73	821.11	5.29	**819.56**	2.93
13	1579.01	1572.79	9.20	1568.79	5.66
14	869.11	867.13	4.08	**866.37**	2.34
Average deviation from best and time	1.50%	0.76%	7.42	0.54%	4.13

100,000 iterations. The results reported in these tables clearly show that running tabu search for 100,000 iterations with path relinking is much more effective than running pure tabu search for 200,000 iterations. In the case of the combination S_3 and C_5, the average gap was reduced by 0.22% and the average running time was reduced by about 44%. Adding a path relinking phase to TS-100,000 is not costly in terms of CPU time. A TS-100,000 run takes approximately $7.42/2$ minutes on average, versus 4.13 minutes for TS + PR. In the case of large scale instances, the average gap was reduced by 0.54% and the average running time was reduced by about 43%.

The first number of each combination in Tables 6 and 7 describes the average improvement over pure tabu search in the case of 100,000 iterations, whereas the second number gives the average improvement for 200,000 iterations. As shown in these tables, every combination of strategy and criterion is capable of improving the results of TS. The tables identify the combination S_3 and C_5 as the one giving the best results for both classical and large scale instances. This combination considers a set of very good solutions obtained over time, and the focus is on search diversification. It should be noted that our conclusions coincide with those of Ghamlouche, Crainic, and Gendreau (2004) who also obtained their best results with the combination S_3, C_5.

It should be mentioned that the second best combination of strategies is S_3 and C_2 (in the case of classical instances) which differs somewhat from the previous one with a focus on search intensification close to the very best solutions of the reference set. In the case of large scale instances, the second best combination is S_2 and C_4. This combination considers a set of good local minimum solutions with a mix of search intensification and diversification.

One would expect strategy S_2, which selects local minima, to not perform too well. This is not always the case, one explanation may be that some of the solutions in $\mathcal{R}$ are not "real" minima. As we allow infeasible regions to be explored, and infeasible solutions cannot be a

Table 5 TS and TS with PR results for large scale instances

Problem	TS 100,000	TS 200,000	Time	TS + PR	Time
1	5889.88	5832.50	21.27	5850.28	17.09
2	8923.17	8808.37	43.35	8799.33	14.31
3	11861.20	11726.70	147.60	11715.90	70.77
4	14554.00	14515.50	162.92	13973.50	95.39
5	6811.98	6737.69	57.30	6546.19	24.85
6	8942.19	8873.44	141.49	8755.48	63.13
7	10671.50	10608.50	45.19	10716.80	36.72
8	12688.80	12379.80	85.64	12170.90	52.09
9	592.54	591.86	23.76	589.39	13.51
10	751.01	751.01	37.22	749.04	23.08
11	928.91	928.34	53.76	927.99	36.19
12	1125.77	1124.91	72.52	1124.44	51.91
13	880.60	878.85	37.51	875.13	21.99
14	1102.83	1101.98	54.24	1105.49	29.80
15	1376.63	1370.54	76.78	1369.70	44.47
16	1655.30	1649.76	109.85	1648.10	65.08
17	715.17	715.00	31.31	713.68	18.44
18	1024.01	1023.20	45.74	1017.65	26.87
19	1394.09	1392.29	69.82	1397.16	39.45
20	1883.08	1871.37	94.10	1864.01	53.97
Average deviation from best and time	3.66%	3.09%	70.57	2.55%	39.96

part of $\mathcal{R}$, the closest feasible solutions to the actual minimum solutions are chosen. The best performance of strategy S_2 is obtained jointly with criterion C_2 that builds a path from the second best to the best solution in $\mathcal{R}$. This is the third best combination, and like the second one, it focuses on search intensification (in this case, close to the local minima). In the case of large scale instances, the best performance of strategy S_2 is obtained with criterion C_4 which is the second best combination mentioned above.

We have shown that TS with path relinking produces better solutions than pure TS. In the following, we will further assess its performance by comparing it to other methods proposed for solving the VRP.

Table 8 provides a comparison on the results obtained on the fourteen classical instances by some of the metaheuristics for the VRP. These are Taillard (1993), Gendreau, Hertz, and Laporte (1994), Rochat and Taillard (1995), Rego and Roucairol (1996), Rego (1998), Toth and Vigo (2003), Cordeau, Laporte, and Mercier (2001), Tarantilis (2005), Reimann, Doerner, and Hartl (2004), Berger and Barkaoui (2003), Prins (2004) and TS + PR (columns marked T1, GHL, RT, RR, R, TV, CLM, T2, RDH, BB, P and HG). The results are expressed in terms of total distance traveled. For our entry in Table 8, we report the results from combination S_3 and C_5, which is the best one.

Table 6 Average improvement over pure tabu search for classical VRP instances

	C_1 (%)	C_2 (%)	C_3 (%)	C_4 (%)	C_5 (%)
S_1	−0.83	−0.80	−0.82	−0.49	−0.76
	−0.12	−0.09	−0.12	0.22	−0.05
S_2	−0.84	−0.86	−0.78	−0.73	−0.86
	−0.14	−0.16	−0.08	−0.03	−0.16
S_3	−0.85	−0.87	−0.79	−0.76	−0.92
	−0.15	−0.17	−0.09	−0.06	−0.22
S_4	−0.84	−0.81	−0.82	−0.70	−0.83
	−0.14	−0.11	−0.12	0.00	−0.13
S_5	−0.83	−0.80	−0.82	−0.55	−0.76
	−0.12	−0.09	−0.12	0.15	−0.05

Table 7 Average improvement over pure tabu search for large scale VRP instances

	C_1 (%)	C_2 (%)	C_3 (%)	C_4 (%)	C_5 (%)
S_1	−0.68	−0.70	−0.67	−0.71	−0.73
	−0.14	−0.16	−0.14	−0.17	−0.20
S_2	−0.81	−0.81	−0.81	−1.01	−0.83
	−0.28	−0.27	−0.28	−0.47	−0.29
S_3	−0.85	−0.83	−0.65	−0.80	−1.04
	−0.15	−0.29	−0.12	−0.27	−0.51
S_4	−0.80	−0.84	−0.92	−0.89	−0.85
	−0.26	−0.31	−0.38	−0.35	−0.32
S_5	−0.66	−0.68	−0.65	−0.76	−0.79
	−0.12	−0.14	−0.12	−0.22	−0.25

From the table we see that our method improves some of the well-known tabu search heuristics such as *Taburoute* (Gendreau, Hertz, and Laporte, 1994), the ejection chain method (Rego and Roucairol, 1996), the subpath ejection chain method (Rego, 1998), the granular tabu search (Toth and Vigo, 2003) and the adapted *Unified Tabu Search* (Cordeau, Laporte, and Mercier, 2001).

It is not easy to compare running times for the various metaheuristics due to different computers, compilers, data structures, computer languages, parallel implementations, and the programming skills of the developer. However, we attempted to scale the computing times on various computers to equal those on a Pentium 2.53 GHz computer, using the factors determined by Dongarra (2004). Table 9 provides a computational comparison of various metaheuristics. The second column reports the average deviation from best known results, while average computing time in minutes is given in the third column. *Parallel* indicates whether a parallel implementation is used, and *computer* refers to the computer used for running the algorithm. The last column reports the scaled computing time.

Our heuristic is faster than the adapted *Unified Tabu Search*, but it is slower than the rest of the other methods. However, we obtained good results using reasonable computing time.

Table 10 provides a comparison of metaheuristics on large scale instances with Golden et al. (1998), Ergun, Orlin, and Steele-Feldman (2003), Li, Golden, and Wasil (2005), Mester and

Table 8 Results of some of the metaheuristics on the fourteen classic benchmark instances

Pr	T1	GHL	RT	RR	R	TV
1	**524.61**	**524.61**	**524.61**	**524.61**	524.81	**524.61**
2	**835.26**	835.77	**835.26**	835.32	847.00	838.60
3	**826.14**	829.45	**826.14**	827.53	832.04	828.56
4	**1028.42**	1036.16	**1028.42**	1044.35	1047.21	1033.21
5	1298.79	1322.65	1291.45	1334.55	1351.18	1318.25
6	**555.43**	**555.43**	**555.43**	**555.43**	559.25	**555.43**
7	**909.68**	913.23	**909.68**	**909.68**	922.21	920.72
8	**865.94**	**865.94**	**865.94**	866.75	876.97	869.48
9	**1162.55**	1177.76	**1162.55**	1164.12	1191.30	1173.12
10	1397.94	1418.51	**1395.85**	1420.84	1460.83	1435.74
11	**1042.11**	1073.47	**1042.11**	**1042.11**	1052.04	1042.87
12	**819.56**	**819.56**	**819.56**	**819.56**	821.63	**819.56**
13	**1541.14**	1573.81	**1541.14**	1550.17	1558.06	1545.51
14	**866.37**	**866.37**	**866.37**	**866.37**	867.79	**866.37**
ADFB	0.05%	0.86%	0.00%	0.55%	1.54%	0.64%

Pr	CLM[a]	T2	RDH[b]	BB	P	HG
1	**524.61**	524.62	**524.61**	**524.61**	**524.61**	**524.61**
2	835.28	835.28	839.59	**835.26**	**835.26**	836.37
3	**826.14**	**826.14**	828.15	827.39	**826.14**	828.26
4	1032.68	1029.64	1038.16	1036.16	1031.63	1034.08
5	1315.76	1311.48	1308.22	1324.06	1300.23	1311.78
6	**555.43**	**555.43**	**555.43**	**555.43**	**555.43**	**555.43**
7	**909.68**	**909.68**	919.77	**909.68**	912.30	**909.68**
8	865.95	**865.94**	866.31	868.32	**865.94**	866.71
9	1167.85	1163.19	1169.54	1169.15	1164.25	1177.01
10	1416.84	1407.21	1418.03	1418.79	1420.20	1420.66
11	1073.47	**1042.11**	1042.77	1043.11	**1042.11**	1042.97
12	**819.56**	**819.56**	**819.56**	**819.56**	**819.56**	**819.56**
13	1549.25	1544.01	1545.48	1553.12	1542.97	1568.79
14	**866.37**	**866.37**	**866.37**	**866.37**	**866.37**	**866.37**
ADFB	0.56%	0.20%	0.48%	0.49%	0.24%	0.54%

ADFB: Average deviation from best known results.
[a]Computational results obtained from Cordeau et al. (2005).
[b]Computational results obtained from Reimann (2004).

Bräysy (2005) and Tarantilis (2005) denoted by GWKC, EOS, LGW, MB and T, respectively. Table 11 gives a summary of the computational results of the metaheuristics on the large scale instances. When testing these instances we obtained better results than three of the mentioned methods, but using more computing time.

Table 9 Summary of computational comparison of various metaheuristics on the classical benchmark instances

Metaheuristic	ADFB (%)	CPU time	Parallel	Computer	Scaled
T1	0.05	n.a.	Yes	Silicon Graphics 100 MHz	–
GHL	0.86	46.8	No	Silicon Graphics 36 MHz	n.a.
RT	0.00	n.a.	No	Silicon Graphics 100 MHz	–
RR	0.55	24.65	Yes	4 Sun Sparc IPC	n.a.
R	1.54	2.32	No	HP 9000/712	0.06
TV	0.64	3.84	No	Pentium 200 MHz	0.10
BB	0.49	21.25	No	Pentium 400 MHz	1.33
T2	0.20	5.63	No	Pentium 400 MHz	0.35
RDH	0.48	3.63	No	Pentium 900 MHz	0.66
CLM	0.56	24.62	No	Pentium 2 GHz	19.41
P	0.24	5.19	No	Pentium 1 GHz	1.24
HG	0.54	4.13	No	Pentium 2.53 GHz	4.13

n.a.: not available

Table 10 Results of metaheuristics on large scale VRP instances

Pr	GWKC	TV	EOS	LGW	CLM
1	5834.60	5736.15	5741.79	5666.42	5681.97
2	9002.26	8553.03	8917.41	8469.32	8657.36
3	11879.95	11402.75	12106.64	11145.80	11037.40
4	14639.32	14910.62	15316.69	13758.08	13740.60
5	6702.73	6697.53	6570.28	6478.09	6756.44
6	9016.93	8963.32	8836.25	8539.61	8537.17
7	11213.31	10547.44	11116.68	10289.72	10267.40
8	12514.20	12036.24	12634.17	11920.52	11869.50
9	587.09	593.35	587.89	588.25	587.39
10	749.15	751.66	749.85	749.49	752.76
11	934.33	936.04	932.74	925.91	929.07
12	1137.18	1147.14	1134.63	1128.03	1119.52
13	881.04	868.80	870.90	865.20	875.88
14	1103.69	1096.18	1097.11	1097.78	1102.03
15	1364.23	1369.44	1367.15	1361.41	1363.76
16	1657.93	1652.32	1643.00	1635.58	1647.06
17	720.44	711.07	716.46	711.74	710.93
18	1029.21	1016.83	1023.32	1010.32	1014.62
19	1403.05	1400.96	1404.84	1382.59	1383.79
20	1875.17	1915.83	1883.33	1850.92	1854.24
ADFB	3.91%	2.87%	3.76%	1.05%	1.45%

(Continued on next page).

Table 10 (Continued).

Pr	P	RDH	MB[a]	T	HG
1	5646.63	5644.02	**5627.54**	5676.97	5850.28
2	**8447.92**	8449.12	**8447.92**	8459.91	8799.33
3	**11036.22**	**11036.22**	**11036.22**	**11036.22**	11715.90
4	**13624.52**	13699.11	**13624.52**	13637.53	13973.50
5	**6460.98**	**6460.98**	**6460.98**	**6460.98**	6546.19
6	**8412.80**	8412.90	8412.88	8414.28	8755.48
7	10195.59	10195.59	**10195.56**	10216.50	10716.80
8	11828.78	11828.78	**11663.55**	11936.16	12170.90
9	591.54	586.87	**583.39**	585.43	589.39
10	751.41	750.77	**742.03**	746.56	749.04
11	933.04	927.27	**918.45**	923.17	927.99
12	1133.79	1140.87	**1107.19**	1130.40	1124.44
13	875.16	865.07	**859.11**	865.01	875.13
14	1086.24	1093.77	**1081.31**	1086.07	1105.49
15	1367.37	1358.21	**1345.23**	1353.91	1369.70
16	1650.94	1635.16	**1622.69**	1634.74	1648.10
17	710.42	708.76	**707.79**	708.74	713.68
18	1014.80	998.83	**998.73**	1006.90	1017.65
19	1376.49	1367.20	**1366.86**	1371.01	1397.16
20	1846.55	1822.94	**1821.15**	1837.67	1864.01
ADFB	0.91%	0.60%	0.00%	0.60%	2.55%

[a]Computational results obtained from Mester and Bräysy (2005) and Cordeau et al. (2005).

Table 11 Summary of computational comparison of various metaheuristics on the large scale instances

Metaheuristic	ADFB (%)	CPU time	Parallel	Computer	Scaled
GWKC	3.91	37.20	No	Pentium 100 MHz	0.40
TV	2.87	17.55	No	Pentium 200 MHz	0.48
EOS	3.76	137.95	No	Pentium 733 MHz	15.49
LGW	1.05	n.a.	No	n.a.	-
CLM	1.45	56.11	No	Pentium 2 GHz	44.24
P	0.91	66.90	No	Pentium 1 GHz	15.97
RDH	0.60	49.33	No	Pentium 900 MHz	9.02
MB	0.00	72.94	No	Pentium 2 GHz	57.51
T	0.60	45.58	No	Pentium 400 MHz	2.84
HG	2.55	39.96	No	Pentium 2.53 MHz	39.96

Tables 8–12 show that tabu search with path relinking is competitive with other VRP methods with respect to the quality of the results that it produces and also the computing time required.

4. Conclusion

In this paper, we presented a tabu search heuristic with path relinking for the vehicle routing problem. Path relinking uses previously encountered good solutions to obtain diversification and intensification in the search. Using path relinking periodically in the search speeds up the identification of very good solutions. Computational results show that tabu search with path relinking is able to produce better solutions than pure tabu search using much less computing time.

Acknowledgments This work was supported by the Research Council of Norway under grant 127533/432 and by the National Science and Engineering Research Council of Canada. This support is gratefully acknowledged. We wish to thank Lars Magnus Hvattum for spending time to make the code more efficient. Thanks are also due to the two anonymous referees for their valuable comments.

References

Aiex, R.M., S. Binato, and M.G.C. Resende. (2003). "Parallel GRASP with Path-Relinking for Job Shop Scheduling." *Parallel Computing* 29(4), 393–430.

Aiex, R.M., M.G.C. Resende, P.M. Pardalos, and G. Toraldo. (2005). "GRASP with Path-Relinking for the Three-Index Assignment Problem," *INFORMS Journal on Computing* 17(2), 224–247.

Alfa, A.S., S.S. Heragu, and M. Chen. (1991). "A 3-opt Based Simulated Annealing Algorithm for Vehicle Routing Problems." *Computers & Industrial Engineering* 21(1–4), 635–639.

Berger, J. and M. Barkaoui. (2003). "A New Hybrid Genetic Algorithm for the Capacitated Vehicle Routing Problem." *Journal of the Operational Research Society* 54(12), 1254–1262.

Bullnheimer, B., R.F. Hartl, and C. Strauss. (1998). "Applying the Ant System to the Vehicle Routing Problem." In S. Voss, S. Martello, I. H. Osman, and C. Roucairol, (eds.), *Metaheuristics: Advances and Trends in Local Search Paradigms for Optimization*, Boston: Kluwer, pp. 109–120.

Bullnheimer, B., R.F. Hartl, and C. Strauss. (1999). "An Improved Ant System for the Vehicle Routing Problem." *Annals of Operations Research* 89, 319–328.

Christofides, N. and S. Eilon. (1969). "An Algorithm for the Vehicle Dispatching Problem." *Operational Research Quarterly* 20(3), 309–318.

Christofdes, N., A. Mingozzi, and P. Toth. (1979). "The Vehicle Routing Problem." In N. Christofdes, A. Mingozzi, P. Toth, and C. Sandi (eds.) *Combinatorial Optimization*, Wiley, Chichester, pp. 315–338.

Cordeau, J.-F., M. Gendreau, A. Hertz, G. Laporte, and J.-S. Sormany. (2005). New Heuristics for the Vehicle Routing Problem. In *A. Langevin and D. Riopel*, (eds.), *Logistics Systems: Design and Optimization*. Boston: Kluwer.

Cordeau, J.-F. and G. Laporte. (2004). Tabu Search Heuristics for the Vehicle Routing Problem. In C. Rego and B. Alidaee, (eds.), *Metaheuristic Optimization via Memory and Evolution: Tabu Search and Scatter Search*. Boston: Kluwer, pp. 145–163.

Cordeau, J.-F., G. Laporte, and A. Mercier. (2001). "A Unifed Tabu Search Heuristic for Vehicle Routing Problems with Time Windows." *Journal of the Operational Research Society* 52(8), 928–936.

Dongarra, J.J. (2004). Performance of Various Computers using Standard Linear Equations Software, Technical Report CS-89-85, Computer Science Department, University of Tennessee, Knoxville, TN.

Ergun, Ö., J.B. Orlin, and A. Steele-Feldman. (2003). "Creating very Large Scale Neighborhoods out of Smaller ones by Compounding Moves: A Study on the Vehicle Routing Problem." Technical report, Massachusetts Institute of Technology.

Gendreau, M., A. Hertz, and G. Laporte. (1994). "A Tabu Search Heuristic for the Vehicle Routing Problem." *Management Science* 40(10), 1276–1290.

Gendreau, M., G. Laporte, and J.-Y. Potvin. (2002). Metaheuristics for the capacitated VRP, In P. Toth and D. Vigo (eds.), *The Vehicle Routing Problem*, SIAM Society for Industrial and Applied Mathematics, Philadelphia, pp. 129–154.

Ghamlouche, I., T.G. Crainic, and M. Gendreau. (2004). "Path Relinking, Cycle-Based Neighbourhoods and Capacitated Multicommodity Network Design." *Annals of Operations Research* 131(1-4), 109–133.

Glover, F. (1986). "Future Paths for Integer Programming and Links to Artificial Intelligence." *Computers & Operations Research* 13(5), 533–549.

Glover, F. (1997). "Tabu Search and Adaptive Memory Programming—Advances, Applications and Challenges." in R. S. Barr, R. V. Helgason, and J. L. Kennington (eds.), *Interfaces in Computer Science and Operations Research*. Boston: Kluwer, pp. 1–75.

Glover, F. (1998). "A Template for Scatter Search and Path Relinking," in J.-K. Hao, E. Lutton, E. Ronald, M. Schoenauer and D. Snyers (eds), "Artificial Evolution." Vol. 1363 *of Lecture Notes in Computer Science*, Springer, Berlin, pp. 13–54.

Glover, F. and M. Laguna. (1993). Tabu search, In C. R. Reeves (ed.), *Modern Heuristic Techniques for Combinatorial Problems*. Oxford: Blackwell Scientific Publishing, pp. 70–150.

Glover, F., M. Laguna, and R. Martí. (2000). "Fundamentals of Scatter Search and Path Relinking." *Control and Cybernetics* 39(3), 653–684.

Golden, B. L., E.A. Wasil, J.P. Kelly, and I.-M. Chao. (1998). Metaheuristics in Vehicle Routing, In T. G. Crainic and G. Laporte, eds, *Fleet Management and Logistics*. Boston: Kluwer, pp. 33-56.

Kawamura, H., M. Yamamoto, T. Mitamura, K. Suzuki, and A. Ohuchi. (1998). "Cooperative Search on Pheromone Communication for Vehicle Routing problems." *IEEE Transactions on Fundamentals* E81-A, 1089–1096.

Laguna, M. and V.A. Armentano. (2005). Lessons from applying and experimenting with scatter search, In C. Rego and B. Alidaee, (eds.), *Adaptive Memory and Evolution: Tabu Search and Scatter Search*. Kluwer, Norwell, pp. 229–246.

Li, F., B. Golden, and E. Wasil. (2005). "Very Large-scale Vehicle Routing: New test problems, algorithms, and results." *Computers & Operations Research* 32(5), 1165–1179.

Mester, D. and O. Bräysy. (2005). "Active Guided Evolution Strategies for the Large Scale Vehicle Routing Problems with Time Windows." *Computers & Operations Research* 32(6), 1593–1614.

Oliveira, C. A., P.M. Pardalos, and M.G. Resende. (2004). GRASP with path-relinking for the quadratic assignment problem, In C. C. Ribeiro and S. L. Martins (eds.), *Efficient and Experimental Algorithms.* Vol. 3059 *of Lecture Notes in Computer Science,* Springer-Verlag, Berlin, pp. 356–368.

Osman, I. H. (1993). "Metastrategy Simulated Annealing and Tabu Search Algorithms for the Vehicle Routing Problem." *Annals of Operations Research* 41(1-4), 421–151.

Prins, C. (2004). "A Simple and Effective Evolutionary Algorithm for the Vehicle Routing Problem." *Computers & Operations Research* 31(12), 1985–2002.

Rego, C. (1998). "A Subpath Ejection Method for the Vehicle Routing Problem." *Management Science* 44(10), 1447-1459.

Rego, C. and C. Roucairol. (1996), "A Parallel Tabu Search Algorithm using Ejection Chains for the Vehicle Routing Problem," In I. H. Osman and J. P. Kelly (eds), *Meta-Heuristics: Theory and Applications*, Boston: Kluwer, MA, pp. 661–675.

Reimann, M. (2004). "Private Communication".

Reimann, M., K. Doerner, and R.F. Hartl. (2004). "D-Ants: Savings Based Ants Divide and Conquer the Vehicle Routing Problem." *Computers & Operations Research* 31(4), 563–591.

Resende, M.G.C. and C.C. Ribeiro. (2003). "A GRASP with Path-relinking for Private Virtual Circuit Routing." *Networks* 41(2), 104–114.

Rochat, Y. and É.D. Taillard. (1995). "Probabilistic Diversification and Intensification in Local Search for Vehicle Routing." *Journal of Heuristics* 1(1), 147–167.

Souza, M. C., C. Duhamel, and C.C. Ribeiro. (2003). "A GRASP Heuristic for the Capacitated Minimum Spanning tree Problem using a Memory-Based Local Search Strategy," In M. G. C. Resende and J. P. Sousa, (eds.), *Metaheuristics: Computer Decision-Making*. Boston: Kluwer Academic Publishers, pp. 627–658.

Taillard, É.D. (1993). "Parallel Iterative Search Methods for Vehicle Routing Problems." *Networks* 23(8), 661–673.

Tarantilis, C. D. (2005). "Solving the Vehicle Routing Problem with Adaptive Memory Programming Methodology." *Computers & Operations Research* 32(9), 2309–2327.

Toth, P. and D. Vigo. (2003). "The Granular Tabu Search and its Application to the Vehicle-Routing Problem." *Journal on Computing* 15(4), 333–346.

Toth, P. and D. Vigo, (eds.), (2002). *The Vehicle Routing Problem,* SIAM Society for Industrial and Applied Mathematics, Philadelphia.

Xu, J. and J.P. Kelly. (1996). "A Network flow-based Tabu Search Heuristic for the Vehicle Routing Problem." *Transportation Science* 30(4), 379–393.