

**New Heuristics for the Vehicle  
Routing Problem**

J.-F. Cordeau, M. Gendreau  
A. Hertz, G. Laporte  
J.-S. Sormany

G-2004-33

April 2004

Revised: June 2004

Les textes publiés dans la série des rapports de recherche HEC n'engagent que la responsabilité de leurs auteurs. La publication de ces rapports de recherche bénéficie d'une subvention du Fonds québécois de la recherche sur la nature et les technologies.

# New Heuristics for the Vehicle Routing Problem

**Jean-François Cordeau**

*HEC Montréal, Canada*  
cordeau@crt.umontreal.ca

**Michel Gendreau**

*Université de Montréal, Canada*  
michelg@crt.umontreal.ca

**Alain Hertz**

*École Polytechnique de Montréal, Canada*  
alain.hertz@gerad.ca

**Gilbert Laporte**

*HEC Montréal, Canada*  
gilbert@crt.umontreal.ca

**Jean-Sylvain Sormany**

*HEC Montréal, Canada*  
sormjs@crt.umontreal.ca

April, 2004

Revised: June, 2004

*Les Cahiers du GERAD*

G-2004-33

Copyright © 2004 GERAD

### **Abstract**

This article reviews some of the best metaheuristics proposed in recent years for the Vehicle Routing Problem. These are based on local search, on population search and on learning mechanisms. Comparative computational results are provided on a set of 34 benchmark instances.

### **Résumé**

Cet article passe en revue quelques-unes des meilleures métaheuristiques proposées au cours des dernières années pour le problème de tournées de véhicules. Celles-ci font appel à la recherche locale, à la recherche évolutionnaire et à des processus d'apprentissage. On présente des résultats numériques comparatifs sur 34 instances d'essai.

## 1 Introduction

The classical *Vehicle Routing Problem* (VRP) is defined on an undirected graph  $G = (V, E)$  where  $V = \{v_0, v_1, \dots, v_n\}$  is a vertex set and  $E = \{(v_i, v_j) : v_i, v_j \in V, i < j\}$  is an edge set. Vertex  $v_0$  is a depot at which are based  $m$  identical vehicles of capacity  $Q$ , while the remaining vertices represent customers. A non-negative *cost*, *distance* or *travel time* matrix  $C = (c_{ij})$  is defined on  $E$ . Each customer has a non-negative demand  $q_i$  and a non-negative service time  $s_i$ . The VRP consists of designing a set of  $m$  vehicle routes i) of least total cost, ii) each starting and ending at the depot, and such that iii) each customer is visited exactly once by a vehicle, iv) the total demand of any route does not exceed  $Q$ , and v) the total duration of any route does not exceed a preset bound  $D$ .

The VRP is a hard combinatorial problem. Exact algorithms (see, e.g., Naddef and Rinaldi (2002); Baldacci et al. (2004)) can only solve relatively small instances and their computational times are highly variable. To this day, heuristics remain the only reliable approach for the solution of practical instances. In contrast to exact algorithms, heuristics are better suited to the solution of VRP variants involving side constraints such as time windows (Cordeau et al. (2002a)), pickups and deliveries (Desaulniers et al. (2002)), periodic visits (Cordeau et al. (1997)), etc.

In recent years several powerful heuristics have been proposed for the VRP and its variants, based on local search, population search and learning mechanisms principles. Local search includes descent algorithms (Ergun et al. (2003)), simulated annealing (Osman (1993)), deterministic annealing (Golden et al. (1998); Li et al. (2004)), tabu search (Osman (1993); Taillard (1993); Gendreau et al. (1994); Xu and Kelly (1996); Rego and Roucairol (1996); Rego (1998); Barbarosoglu and Ögür (1999); Cordeau et al. (2001)). The two best known types of population search heuristics are evolutionary algorithms (Prins (2004); Berger and Barkaoui (2004); Mester and Bräysy (2004)) and adaptive memory procedures (Rochat and Taillard (1995); Tarantilis and Kiranoudis (2002)). Examples of learning mechanisms are neural networks (Ghaziri (1991, 1996); Matsuyama (1991); Schumann and Retzko (1995)) and ant algorithms (Reimann et al. (2004)).

The field of VRP heuristics is very active, as witnessed by the large number of recent articles listed in the previous paragraph. This chapter summarizes some of the most important new developments in the area of VRP heuristics and presents comparative computational results.

Several surveys have recently been published on VRP heuristics (Laporte and Semet (2002); Gendreau et al. (2002); Cordeau et al. (2002a); Cordeau and Laporte (2004)). This chapter focuses on recent material not covered by these surveys. In the following section we provide a general classification scheme for VRP heuristics. We then provide in Section 3 a description of nine recent heuristics, and computational results in Section 4. The conclusion follows.

## 2 Classification of VRP heuristics

Providing classification schemes in the area of combinatorial optimization can be a daunting task because of the large number of fields and descriptors needed to account for the diversity and intricacy of the concepts involved in the various algorithms — the devil is in the details. By and large broad classification systems that concentrate on the essential ideas can be quite instructive.

At a macro-level, VRP heuristics combine some of the following four components: 1) *construction* of an initial solution; 2) *improvement* procedures; 3) *population* mechanisms; and 4) *learning* mechanisms.

### 2.1 Constructive heuristics

The ideas behind most constructive heuristics are well known and well documented (Laporte and Semet (2002)). These include the Clarke and Wright (1964) savings concept, the sweep mechanism (Gillett and Miller (1974)), and cluster-first route-second methods (Fisher and Jaikumar (1981)), and route-first cluster-second methods (Beasley (1983)).

### 2.2 Improvement heuristics

Most constructive procedures are followed by an improvement phase. In the simplest case, a post-optimization procedure designed for the *Traveling Salesman Problem* (TSP) is applied to individual routes:  $r$ -opt exchanges (Lin (1965)), Or-opt exchanges (Or (1976)), 2-opt\* exchanges (Potvin and Rousseau (1995)), 4-opt\* exchanges (Renaud et al. (1996)), and the more involved unstringing and stringing (US) mechanism (Gendreau et al. (1992)). Exchanges often involve two vehicle routes, such as chain exchanges (Fahrion and Wrede (1990)) and the  $\lambda$ -interchange mechanisms (Osman (1993)), the string cross, string exchange and string relocation schemes of van Breedam (1994). Finally, more complicated operations involve several routes: cyclic exchanges (Thompson and Psaraftis (1993)), edge exchange schemes (Kindervater and Savelsbergh (1997)), ejection chains (Xu and Kelly (1996); Rego and Roucairol (1996); Rego (1998)), and very large neighbourhoods in which a sequence of moves is determined through the solution of an auxiliary network flow optimization problem (Ergun et al. (2003)).

Most classical improvement mechanisms work in a descent mode until a local optimum is reached. In metaheuristics (e.g., simulated annealing, deterministic annealing, tabu search) the same mechanisms are embedded within sophisticated neighbourhood search structures which allow for intermediate deteriorating solutions and even infeasible solutions (e.g., Gendreau et al. (1994)). In variable neighbourhood search (VNS), introduced by Mladenović and Hansen (1997), the neighbourhood structure is allowed to vary during the search; this concept can be coupled with descent methods or with tabu search, for example. Figure 1 depicts a number of ways to design heuristics consisting of a construction phase followed by an improvement phase.

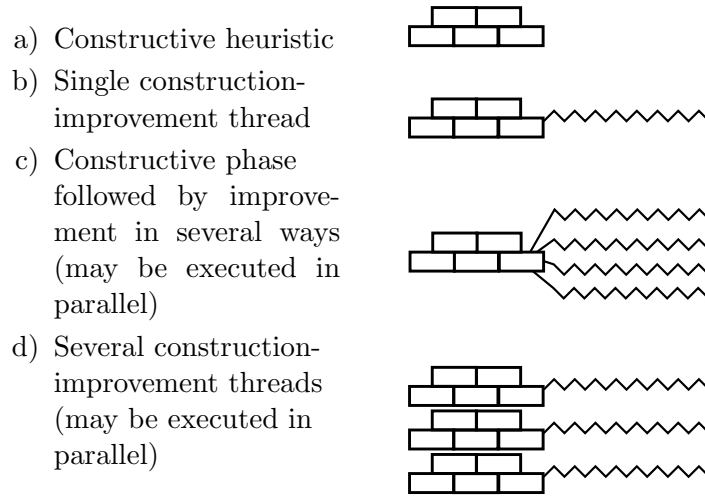




Figure 1: Graphical representation of several construction and improvement heuristics ( : construction; : improvement)

## 2.3 Population mechanisms

Combination of solutions is the basic mechanism of population search which includes a large number of variants known as genetic algorithms (e.g., Reeves (2003)), and memetic algorithms (e.g., Moscato and Cotta (2003)). Classical genetic algorithms operate on a population of encoded solutions called chromosomes. At each iteration (generation) the following operations are applied  $k$  times: select two parent chromosomes; generate two offspring from these parents using a crossover operator; apply a random mutation to each offspring with a small probability; remove the  $2k$  worst elements of the population and replace them with the  $2k$  offspring. Several ways of performing crossovers have been proposed for sequencing problems (e.g., Potvin (1996); Bean (1994); Drezner (2003)).

The idea of combining solutions to generate new ones is central to the adaptive memory procedure put forward by Rochat and Taillard (1995) for the solution of the VRP. These authors extract vehicle routes from several good solutions and use them as a basis for the construction of offspring. A variant, proposed by Tarantilis and Kiranoudis (2002), initiates offspring from chains of vertices extracted from parent solutions. Figure 2 depicts a population mechanism.

## 2.4 Learning mechanisms

Two main learning mechanisms have been used for the design of VRP heuristics. Neural networks operate on a set of deformable templates which are essentially rings that are candidates to become feasible vehicle routes. Rings compete for vertices through a random mechanism in which the probability of assigning a vertex to a ring evolves through a learning process. It is fair to say that neural networks cannot yet compete with most

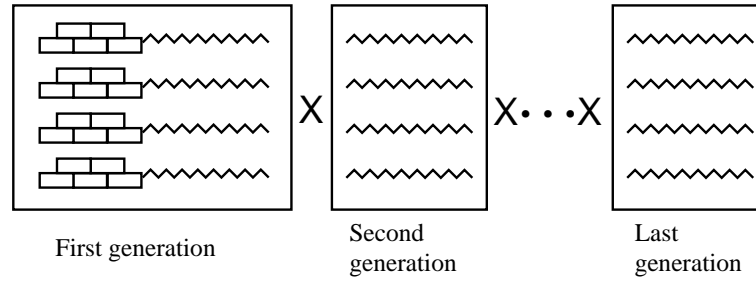


Figure 2: Depiction of a population algorithm obtained by combining ( $X$ ) some elements of a generation to obtain the next generation

other VRP heuristics. Ant algorithms are also derived from a learning paradigm. They are derived from an analogy with ants which lay pheromone on their trail as they forage for food. With time paths leading to the best food sources are more frequented and are marked with a larger amount of pheromone. In construction or improvement heuristics for the VRP elementary moves leading to better solution can be assigned a higher probability of being selected. An algorithm based on such a learning feedback mechanism will be outlined in Section 3. Figure 3 depicts two learning mechanisms. The learning feedback loop enables the process to restart with different rules or parameter settings.

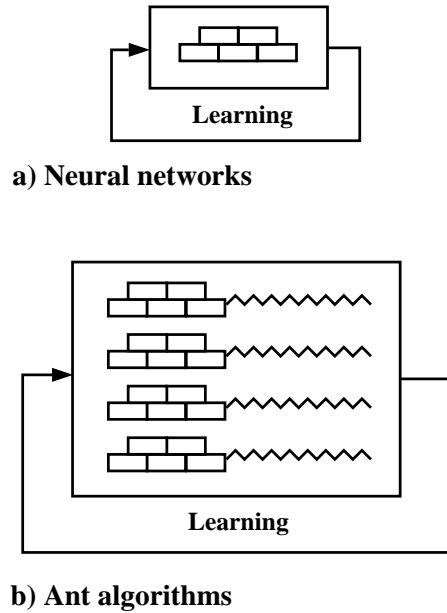


Figure 3: Depiction of two learning mechanisms

### 3 Some recent VRP heuristics

We now summarize nine recent VRP heuristics. The first four are based on local search, the next four are population based, while the last one is an ant algorithm.

#### 3.1 The Toth and Vigo granular tabu search algorithm

The granular tabu search (GTS) algorithm put forward by Toth and Vigo (2003) *a priori* removes from the graph edges that are unlikely to appear in an optimal VRP solution, with the aim of curtailing computation time. The idea was first implemented in conjunction with a tabu search method but the principle is general and could be beneficial to other type of algorithms. Specifically, Toth and Vigo recommend retaining only the edges incident to the depot and all edges whose length does not exceed a given *granularity threshold*  $\nu = \beta \bar{c}$ , where  $\bar{c}$  is the average edge cost in a good feasible solution obtained by a fast heuristic, and  $\beta$  is a sparsification parameter typically chosen in the interval  $[1.0, 2.0]$ . With this choice of  $\beta$ , the percentage of edges remaining in the reduced graph tends to lie between 10% and 20% of the original number. In practice  $\beta$  is dynamically decreased to provide an intensification effect, or increased to diversify the search. Toth and Vigo have implemented GTS in conjunction with some features included in the tabu search algorithms of Taillard (1993) and of Gendreau et al. (1994). Neighbour solutions were obtained by performing intra-route and inter-route edge exchanges.

#### 3.2 The Li, Golden and Wasil heuristic

The search heuristic developed by Li et al. (2004) combines the record-to-record (RTR) principle first put forward by Dueck (1993) with a variable-length neighbour list whose principle is similar to GTS (Toth and Vigo (2003)). The algorithm is called VRTR for variable-length neighbourhood list record-to-record travel. Only a proportion  $\alpha$  of the 40 shortest edges incident to each vertex are retained. The value of  $\alpha$  varies throughout the algorithm.

The RTR search is applied different times from three initial solutions generated with the Clarke and Wright (1964) algorithm with savings  $s_{ij}$  defined as  $c_{i0} + c_{0j} - \lambda c_{ij}$ , where  $\lambda = 0.6, 1.4$  and  $1.6$ . Neighbour solutions are obtained by means of intra-route and inter-route 2-opt moves. During the search, deteriorating solutions are accepted as long as their solution value does not exceed 1.01 times that of the best known solution. When the value of the incumbent has not improved for a number of iterations a final attempt is made to improve the best known solution by means of a perturbation technique. This is done by reinserting some of its vertices in different positions and restarting the search process.

#### 3.3 The unified tabu search algorithm

The unified tabu search algorithm (UTSA) was originally put forward by Cordeau et al. (1997) as a unified tool to solve periodic and multi-depot VRPs. It has been extended to the site dependent VRP (Cordeau et al. (2001)), and to the time-windows version of these problems (Cordeau et al. (2001, 2004)). It possesses some of the features of Taburoute



(Gendreau et al. (1994)), namely the consideration of intermediate infeasible solutions through the use of a generalized objective function containing self-adjusting coefficients, and the use of continuous diversification. Neighbour solutions are obtained by moving a vertex from its route between two of its closest neighbours in another route, by means of a generalized insertion (or GENI) (see, Gendreau et al. (1992)). Contrary to Taburoute, UTSA uses only one initial solution and fixed tabu durations. The tabu mechanism works with an attribute set  $B(x)$  associated with solution  $x$ , defined as  $B(x) = \{(i, k): \text{vertex } v_i \text{ is visited by vehicle } k \text{ in solution } x\}$ . The neighbourhood mechanism removes an attribute  $(i, k)$  from  $B(x)$  and replaces it with  $(i, k')$ , where  $k' \neq k$ ; attribute  $(i, k)$  is then declared tabu. Recently a new diversification phase was introduced into UTSA. Whenever the value of the best known solution has not improved for a number of iterations, the depot is moved to the first vertex of a randomly selected route and temporarily remains in this location. This computational device is a form of data perturbation, a principle put forward by Codenetti et al. (1996). On benchmark test problems the implementation of this simple device has helped reduce the average deviation from the best known solution values from 0.69% to 0.56% without any increase in computing time (see Table 1).

### 3.4 Very large neighbourhood search

Very large neighbourhood search (VLNS) attempts, at every iteration, to identify an improving solution by exploring a neighbourhood whose size is very large with respect to the input data. It was applied to the VRP by Ergun et al. (2003). The heuristic developed by these authors is a descent mechanism that operates on several routes at once, not unlike cyclic transfers (Thompson and Psaraftis (1993)) and ejection chains (Rego and Roucairol (1996)) which act on a set of  $h$  routes  $r_1, \dots, r_h$  by moving vertices from route  $r_\ell$  to route  $r_{\ell+1(\text{mod } h)}$ ,  $\ell = 1, \dots, h$ . Neighbour solutions are defined by means of 2-opt moves, vertex swaps between routes, and vertex insertions in different routes. In order to determine the best sequence of moves at a given iteration, a shortest path problem is solved on an auxiliary graph, called improvement graph. The main advantage of this type of approach is that it allows a broad search to be performed by acting on several moves at once. Its disadvantage lies in the computational effort required at each iteration to determine the best compounded move.

### 3.5 The evolutionary algorithm of Prins

The heuristic put forward by Prins (2004) combines the two main features of evolutionary search: crossover and mutation operations. Improvements are obtained by means of a local search procedure applied to a candidate solution. Hence this algorithm is best described as a memetic algorithm (Moscato and Cotta (2003)). The moves include vertex and edge reinsertions, vertex swaps, combined vertex and edge swaps, and edge swaps. The search procedure ends at the first improving move. This algorithm operates on solutions represented as ordered sequences of customers: all vertices except the depot first appear on a cycle, without trip delimiters, as if a single vehicle traveled all routes in succession, in a cyclic manner, without going through the depot. An optimal partition of this cycle is then

determined by solving a shortest path problem on an auxiliary graph, like what is done in route-first cluster-second algorithms (Beasley (1983)). This procedure is also applied after each mutation. Crossovers are performed as follows. Two cutting locations  $i$  and  $j$  are determined in parent # 1 and the corresponding string is placed in positions  $i, \dots, j$  of offspring # 1, which is then completed by sweeping parent # 2 circularly from position  $j + 1$ . A second offspring is created by reversing the roles of the two parents.

### 3.6 The Bone Route adaptive memory algorithm of Tarantilis and Kiranoudis

Tarantilis and Kiranoudis (2002) have developed a rather effective adaptive memory procedure for the VRP. In a first phase a solution is obtained by means of the Paessens (1988) constructive procedure, which is an enhancement of the Clarke and Wright (1964) algorithm, followed by a tabu search procedure in which neighbours are defined by 2-opt moves, vertex swaps between routes, and vertex reinsertions in the same route or in a different route. The adaptive memory procedure does not initiate new solutions by combining full vehicle routes, as did Rochat and Taillard (1995) but route segments, called *bones*, extracted from good quality routes.

### 3.7 The AGES algorithm of Mester and Bräysy

The active guided evolution strategies (AGES) algorithm of Mester and Bräysy (2004) was initially applied to the VRP with time windows but results have recently been obtained for the classical VRP (Mester (2004)). AGES combines guided local search (Voudouris (1997)) with evolution strategies (Rechenberg (1973)) into an iterative two-stage procedure. Contrary to standard evolutionary search heuristics, AGES uses a deterministic rule for parent selection and the search is driven by a high mutation rate. AGES does not recombine parents, but it creates a single offspring from a single parent through a mutation procedure, and the offspring replaces the parent if it has a better fitness value. Guided local search operates like simple memory based metaheuristics such as simulated annealing and tabu search. It penalizes some solution features that are unlikely to appear in an optimal solution (like long edges) and also uses a frequency weight. This search mechanism therefore combines the basic principle of granular tabu search with continuous diversification. Neighbour solutions are defined by vertex swaps and interchanges, and by 2-opt moves (Potvin and Rousseau (1995)). The search procedure uses very large neighbourhoods (Shaw (1998)). A restart mechanism applied to the best solution encountered is reported to play a significant role in reaching high quality solutions.

### 3.8 The hybrid genetic algorithm of Berger and Barkaoui

The hybrid genetic algorithm of Berger and Barkaoui (2004) combines evolutionary search and local search. It is best described as a memetic algorithm. Its originality lies in the use of two populations. New offspring are created in each population whose size is kept constant by replacing the worst elements by the best ones. A migration operation is

then applied by swapping the best elements of each population. Crossovers are performed by creating one offspring from two parents as follows: a number of routes are extracted from parent # 1, yielding a partial solution which is then completed by inserting in some of the routes vertices of parent # 2 selected according to a proximity criterion (their closeness to the centroid of a route), or by creating new routes. The insertion mechanism I1 (Solomon (1987)) is modified to include a random choice of the cost function parameters. Solutions are improved by performing a large scale neighbourhood search (Shaw (1998)) that combines three insertion mechanisms, and by applying a route improvement scheme. The first insertion mechanism is based on I1: vertices are first ranked according to a function that combines their best reinsertion cost and the number of feasible insertions, and the highest ranked vertex is then reinserted according to a cheapest insertion cost criterion. The second mechanism uses a reject function as in Liu and Shen (1999) that compares for a given vertex the cost of an insertion opportunity to the best insertion cost achievable in the neighbourhood of that vertex. The third mechanism operates on two routes at a time by performing moves or swaps involving up to two vertices (as in the Osman (1993)  $\lambda$ -interchange mechanism), and by implementing the first improving move. Finally, an attempt to improve each route is made by removing in turn each vertex and reinserting it by means of I1.

### 3.9 The $D$ -ants savings based heuristic of Reimann, Doerner and Hartl

The  $D$ -ants heuristic of Reimann et al. (2004) repeatedly applies two phases until a stopping criterion is reached. The first phase iterates between a savings based procedure for generation of a pool of good solutions and an improvement mechanism applied to each of these solutions. A learning mechanism guides the creation of each new generation. Solutions are generated by means of a savings algorithm. Instead of using the classical Clarke and Wright (1964) saving  $s_{ij} = c_{i0} + c_{0j} - c_{ij}$ , the authors use an attractiveness value  $\chi_{ij}$  equal to  $\tau_{ij}^\alpha s_{ij}^\beta$ , where  $\tau_{ij}^\alpha$  contains information on how good combining  $i$  and  $j$  turned out to be in previous iterations, and  $\alpha, \beta$  are user-controlled weights. The combination of vertices  $v_i$  and  $v_j$  occurs with probability  $p_{ij}$  defined as  $\chi_{ij} / (\sum_{(h,\ell) \in \Omega_k} \chi_{h\ell})$ , where  $\Omega_k$  is the

set of the feasible  $(i, j)$  combinations yielding the  $k$  best savings. The authors use 2-opt in the improvement phase. In the second phase, the best solution identified in the first phase is decomposed into several subproblems, each of which is optimized by means of the procedure used in the first phase.

## 4 Comparative computational results

We now present computational results for the various heuristics described in Section 3. Statistics for the 14 Christofides et al. (1979) instances ( $50 \leq n \leq 199$ ) and for the 20 Golden et al. (1998) instances ( $200 \leq n \leq 480$ ) are reported in Tables 1 and 2, respectively. Unless otherwise indicated, solution values correspond to a single run with a given parameter setting. Best solution values are in boldface.

The column headings are as follows:

$n$	: number of customers;
Value	: best solution value produced by the heuristic;
%	: percentage deviation from the best known value;
Minutes	: computation time in minutes;
Best	: best known solution value.

Our first observation relates to the accuracy level reached by these algorithms. On the CMT instances, the average percentage deviation from the best known solution value always lies between 0.03 and 0.64. The worst performers (GTS and UTSA) are two tabu search algorithms, while the best algorithms combine population search and local search (e.g., AGES, Bone Route and the Prins algorithm). This observation is consistent with the results obtained with the first generation of tabu search heuristics (Cordeau and Laporte (2004)) for which the average deviation was typically higher. Results obtained for the larger instances (Table 2) point in the same direction but they must be interpreted with more care because these instances have not been as extensively studied as the first ones.

Computation times are provided for information but, as usual these are hard to interpret because of the different computers employed. The Dongarra (2004) study which is frequently updated throws some light on relative computer speeds. Irrespective of this, it appears that the AGES heuristic of Mester and Bräysy runs rather fast and comes out as the overall winner when both accuracy and computing time are taken into account.

Two additional performance criteria stated by Cordeau et al. (2002b) are simplicity and flexibility. Simplicity relates to ease of understanding and coding of an algorithm. According to this criterion the Li et al. (2004) heuristic is probably the best: it is based on a rather simple mechanism and requires a relatively small amount of coding. The Prins algorithm, UTSA and GTS also possess simple structures which should make them easier to reproduce. At the other extreme, VLNS and AGES are probably the most complicated of all algorithms used in the comparisons. Flexibility measures the capacity of adapting an algorithm to effectively deal with additional constraints. There exists abundant documented evidence that UTSA can be applied to a host of VRP extensions (Cordeau et al. (2001, 2004)). Also, some generic principles like GST and VLNS apply to several contexts and should score high on the flexibility criterion, although they have not to our knowledge yet been applied to VRP extensions such as the VRP with time windows (VRPTW). On the other hand, Prins and Mester and Bräysy report results on the VRPTW.

## 5 Conclusion

In recent years several new metaheuristics have been put forward for the solution of the VRP. These combine a variety of principles including tabu search, population search and learning mechanisms. The best methods combine population search and local search, thus providing at the same time breadth and depth in the solution space exploration. All algorithms described in this study are highly accurate and some are also quite fast. What is now needed is a greater emphasis on simplicity and flexibility.

Table 1: Computational results for the Christofides et al. (1979) instances

			GTS			Li, Golden and		USTA			VLNS			Prins (2004)		
			Toth and Vigo (2003)			Wasil (2004)		Cordeau et al. (2001)			Ergun et al. (2003)					
Instance	<i>n</i>	Type <sup>1</sup>	Value	%	Minutes <sup>2</sup>	Value <sup>3</sup>	%	Value <sup>4</sup>	%	Minutes <sup>5</sup>	Value <sup>6</sup>	%	Minutes <sup>7</sup>	Value	%	Minutes <sup>8</sup>
1	50	C	<b>524.61</b>	0.00	0.81	<b>524.61</b>	0.00	<b>524.61</b>	0.00	2.32	<b>524.61</b>	0.00	23.13	<b>524.61</b>	0.00	0.01
2	75	C	838.60	0.40	2.21	836.18	0.11	835.28	0.00	14.78	835.43	0.02	33.93	<b>835.26</b>	0.00	0.77
3	100	C	828.56	0.29	2.39	827.39	0.15	<b>826.14</b>	0.00	11.67	827.46	0.16	21.30	<b>826.14</b>	0.00	0.46
4	150	C	1033.21	0.47	4.51	1045.36	1.65	1032.68	0.41	26.66	1036.24	0.76	24.45	1031.63	0.31	5.50
5	199	C	1318.25	2.09	7.50	1303.47	0.94	1315.76	1.90	57.68	1307.33	1.24	57.25	1300.23	0.69	19.10
6	50	C, D	<b>555.43</b>	0.00	0.86			<b>555.43</b>	0.00	3.03	<b>555.43</b>	0.00	3.50	<b>555.43</b>	0.00	0.01
7	75	C, D	920.72	1.21	2.75			<b>909.68</b>	0.00	7.41	910.04	0.04	36.53	912.30	0.29	1.42
8	100	C, D	869.48	0.41	2.90			865.95	0.00	10.93	<b>865.94</b>	0.00	12.43	<b>865.94</b>	0.00	0.37
9	150	C, D	1173.12	0.91	5.67			1167.85	0.46	51.66	1164.88	0.20	42.47	1164.25	0.15	7.25
10	199	C, D	1435.74	2.86	9.11			1416.84	1.50	106.28	1404.36	0.61	28.32	1420.20	1.74	26.83
11	120	C	1042.87	0.07	3.18	<b>1042.11</b>	0.00	1073.47	3.01	11.67	<b>1042.11</b>	0.00	69.13	<b>1042.11</b>	0.00	0.30
12	100	C	<b>819.56</b>	0.00	1.10	<b>819.56</b>	0.00	<b>819.56</b>	0.00	9.02	<b>819.56</b>	0.00	5.98	<b>819.56</b>	0.00	0.05
13	120	C, D	1545.51	0.28	9.34			1549.25	0.53	21.00	1544.99	0.25	39.73	1542.97	0.12	10.44
14	100	C, D	<b>866.37</b>	0.00	1.41			<b>866.37</b>	0.00	10.53	<b>866.37</b>	0.00	6.55	<b>866.37</b>	0.00	0.09
Average				0.64	3.84		0.41		0.56	24.62		0.23	28.91		0.24	5.19

1. C: Capacity restrictions; D: Route length restrictions.
2. Pentium (200 MHz).
3. Best variant ( $\alpha = 0.4$ ).
4. Results of recent computational experiments (see Section 3.3); the average % deviation in Cordeau et al. (2001) is 0.69.
5. Pentium IV (2GHz).
6. Best of five runs.
7. Time for reaching the best value for the first time (Pentium III, 733 MHz).
8. GHz PC (75 MFlops).

Table 1: (continued). Computational results for the Christofides et al. (1979) instances

			Bone Route (Tarantitis and Kiranoudis (2002))			AGES best Mester and Bräysy (2004)			AGES fast Mester and Bräysy (2004)			Berger and Barkaoui (2004)			Best
Instance	<i>n</i>	Type <sup>1</sup>	Value	%	Minutes <sup>9</sup>	Value <sup>10</sup>	%	Minutes <sup>11</sup>	Value <sup>10</sup>	%	Minutes <sup>11</sup>	Value	%	Minutes <sup>12</sup>	
1	50	C	<b>524.61</b>	0.00	0.11	<b>524.61</b>	0.00	0.01	<b>524.61</b>	0.00	0.01	<b>524.61</b>	0.00	2.00	<b>524.61</b>
2	75	C	<b>835.26</b>	0.00	4.56	<b>835.26</b>	0.00	0.26	<b>835.26</b>	0.00	0.26	<b>835.26</b>	0.00	14.33	<b>835.26</b>
3	100	C	<b>826.14</b>	0.00	7.66	<b>826.14</b>	0.00	0.05	<b>826.14</b>	0.00	0.05	827.39	0.15	27.90	<b>826.14</b>
4	150	C	1030.88	0.24	9.13	<b>1028.42</b>	0.00	0.47	<b>1028.42</b>	0.00	0.47	1036.16	0.75	48.98	<b>1028.42</b>
5	199	C	1314.11	1.77	16.97	<b>1291.29</b>	0.00	101.93	1294.25	0.23	0.50	1324.06	2.54	55.41	<b>1291.29</b>
6	50	C, D	<b>555.43</b>	0.00	0.10	<b>555.43</b>	0.00	0.02	<b>555.43</b>	0.00	0.02	<b>555.43</b>	0.00	2.33	<b>555.43</b>
7	75	C, D	<b>909.68</b>	0.00	0.92	<b>909.68</b>	0.00	0.43	<b>909.68</b>	0.00	0.43	<b>909.68</b>	0.00	10.50	<b>909.68</b>
8	100	C, D	<b>865.94</b>	0.00	4.28	<b>865.94</b>	0.00	0.44	<b>865.94</b>	0.00	0.44	868.32	0.27	5.05	<b>865.94</b>
9	150	C, D	1163.19	0.06	5.83	<b>1162.55</b>	0.00	1.22	1164.54	0.17	0.50	1169.15	0.57	17.88	<b>1162.55</b>
10	199	C, D	1408.82	0.93	14.32	1401.12	0.41	2.45	1404.67	0.42	0.45	1418.79	1.64	43.86	<b>1395.85</b>
11	120	C	<b>1042.11</b>	0.00	0.21	<b>1042.11</b>	0.00	0.05	<b>1042.11</b>	0.00	0.05	1043.11	0.10	22.43	<b>1042.11</b>
12	100	C	<b>819.56</b>	0.00	0.10	<b>819.56</b>	0.00	0.01	<b>819.56</b>	0.00	0.01	<b>819.56</b>	0.00	7.21	<b>819.56</b>
13	120	C, D	1544.01	0.19	8.75	<b>1541.14</b>	0.00	0.63	1543.26	0.14	0.47	1553.12	0.78	34.91	<b>1541.14</b>
14	100	C, D	<b>866.37</b>	0.00	0.10	<b>866.37</b>	0.00	0.08	<b>866.37</b>	0.00	0.08	<b>866.37</b>	0.00	4.73	<b>866.37</b>
Average				0.23	5.22		0.03	7.72		0.07	0.27		0.49	21.25	

9. Pentium II (400 MHz).

10. For C instances, see Mester and Bräysy (2004). Otherwise, see Mester (2004).

11. Pentium IV (2 GHz).

12. Pentium (400 MHz).

Table 2: Computational results for the Golden et al. (1998) instances

			GTS Toth and Vigo (2003)			Li, Golden and Wasil (2004)		USTA Cordeau et al. (2001)			VLNS Ergun et al. (2003)			Prins (2004)		
Instance	Type <sup>1</sup>		Value	%	Minutes <sup>2</sup>	Value <sup>3</sup>	%	Value <sup>4</sup>	%	Minutes <sup>5</sup>	Value <sup>6</sup>	%	Minutes <sup>7</sup>	Value	%	Minutes <sup>8</sup>
1 240	C		5736.15	1.93	4.98	5666.42	0.69	5681.97	0.97	10.29	5741.79	2.03	134.95	5646.63	0.34	32.42
2 320	C		8553.03	1.24	8.28	8469.32	0.25	8657.36	2.48	35.39	8917.41	5.56	150.83	<b>8447.92</b>	0.00	77.92
3 400	C		11402.75	3.32	12.94	11145.80	0.99	11037.40	0.01	55.39	12106.64	9.70	15.67	<b>11036.22</b>	0.00	120.83
4 480	C		14910.62	9.44	15.13	13758.08	0.98	13740.60	0.85	83.19	15316.69	12.42	106.50	<b>13624.52</b>	0.00	187.60
5 200	C		6697.53	3.66	2.38	6478.09	0.26	6756.44	4.57	5.13	6570.28	1.69	15.50	<b>6460.98</b>	0.00	1.04
6 280	C		8963.32	6.54	4.65	8539.61	1.51	8537.17	1.48	18.64	8836.25	5.03	81.98	<b>8412.80</b>	0.00	9.97
7 360	C		10547.44	3.45	11.66	10289.72	0.92	10267.40	0.70	25.60	11116.68	9.03	85.00	10195.59	0.00	39.05
8 440	C		12036.24	3.20	11.08	11920.52	2.20	11869.50	1.77	71.44	12634.17	8.32	33.95	11828.78	1.42	88.30
9 255	C,D		593.35	1.71	11.67	588.25	0.83	587.39	0.69	37.26	587.89	0.77	49.20	591.54	1.40	14.32
10 323	C,D		751.66	1.30	15.83	749.49	1.01	752.76	1.45	51.11	749.85	1.05	125.05	751.41	1.26	36.58
11 399	C,D		936.04	1.92	33.12	925.91	0.81	929.07	1.16	41.54	932.74	1.56	171.05	933.04	1.59	78.50
12 483	C,D		1147.14	3.61	42.90	1128.03	1.88	1119.52	1.11	157.01	1134.63	2.48	388.62	1133.79	2.40	30.87
13 252	C,D		868.80	1.13	11.43	865.20	0.71	875.88	1.95	34.83	870.90	1.37	235.13	875.16	1.87	15.30
14 320	C,D		1096.18	1.38	14.51	1097.78	1.52	1102.03	1.92	21.56	1097.11	1.46	31.17	1086.24	0.46	34.07
15 396	C,D		1369.44	1.80	18.45	1361.41	1.20	1363.76	1.38	57.64	1367.15	1.63	65.30	1367.37	1.65	110.48
16 480	C,D		1652.32	1.83	23.07	1635.58	0.79	1647.06	1.50	129.50	1643.00	1.25	31.58	1650.94	1.74	130.97
17 240	C,D		711.07	0.46	14.29	711.74	0.56	710.93	0.44	18.03	716.46	1.22	223.62	710.42	0.37	5.86
18 300	C,D		1016.83	1.81	21.45	1010.32	1.16	1014.62	1.59	67.11	1023.32	2.46	299.23	1014.80	1.61	39.33
19 360	C,D		1400.96	2.49	30.06	1382.59	1.15	1383.79	1.24	66.21	1404.84	2.78	393.03	1376.49	0.70	74.25
20 420	C,D		1915.83	5.20	43.05	1850.92	1.63	1854.24	1.82	135.29	1883.33	3.41	121.62	1846.55	1.39	210.42
Average				2.87	17.55		1.05		1.45	56.11		3.76	137.95		0.91	66.90

1. C: Capacity restrictions; D: Route length restrictions.
2. Pentium (200 MHz).
3. Best variant ( $\alpha = 0.01$ )
4. Results of recent computational experiments (see Section 3.3).
5. Pentium IV (2GHz).
6. Best of two runs.
7. Time for reaching the best value for the first time (Pentium III, 733 MHz).
8. GHz PC (75 MFlops).

Table 2: (continued). Computational results for the Golden et al. (1998) instances

			Bone Route (Tarantitis and Kiranoudis (2002))			AGES best Mester and Bräysy (2004)			AGES fast Mester and Bräysy (2004)			D-Ants Reimann et al. (2004)			
Instance	Type		Value	%	Minutes <sup>9</sup>	Value <sup>10</sup>	%	Minutes <sup>11</sup>	Value <sup>10</sup>	%	Minutes <sup>11</sup>	Value <sup>12</sup>	%	Minutes <sup>13</sup>	Best
1 240	C		5676.97	0.88	27.86	<b>5627.54</b>	0.00	8.73	5644.00	0.30	0.70	5644.02	0.29	62.52	<b>5627.54</b>
2 320	C		8512.64	0.77	55.62	<b>8447.92</b>	0.00	46.66	8468.00	0.24	0.20	8449.12	0.01	57.67	<b>8447.92</b>
3 400	C		11199.72	1.48	59.21	<b>11036.22</b>	0.00	40.55	11146.00	0.99	0.70	<b>11036.22</b>	0.00	21.92	<b>11036.22</b>
4 480	C		13637.53	0.10	47.63	<b>13624.52</b>	0.00	470.00	13704.52	0.59	2.50	13699.11	0.55	119.12	<b>13624.52</b>
5 200	C		<b>6460.98</b>	0.00	11.34	<b>6460.98</b>	0.00	0.17	6466.00	0.08	0.50	<b>6460.98</b>	0.00	0.87	<b>6460.98</b>
6 280	C		8429.28	0.20	12.54	<b>8412.88</b>	0.00	75.22	8539.61	1.51	0.10	8412.90	0.00	5.72	<b>8412.80</b>
7 360	C		10216.50	0.21	42.50	<b>10195.56</b>	0.00	2.55	10240.42	0.44	0.85	10195.59	0.00	14.03	<b>10195.56</b>
8 440	C		11936.16	2.34	79.69	<b>11663.55</b>	0.00	34.30	11918.75	2.19	0.27	11828.78	1.42	35.30	<b>11663.55</b>
9 255	C,D					<b>583.39</b>	0.00	8.33	588.25	0.83	0.80	586.87	0.60	21.52	<b>583.39</b>
10 323	C,D					<b>742.03</b>	0.00	6.00	752.92	1.39	0.43	750.77	1.25	17.48	<b>742.03</b>
11 399	C,D					<b>918.45</b>	0.00	110.00	925.94	0.82	1.10	927.27	0.96	96.88	<b>918.45</b>
12 483	C,D					<b>1107.19</b>	0.00	600.00	1128.67	1.94	1.50	1140.87	3.04	61.38	<b>1107.19</b>
13 252	C,D					<b>859.11</b>	0.00	10.25	865.20	0.71	0.18	865.07	0.69	87.20	<b>859.11</b>
14 320	C,D					<b>1081.31</b>	0.00	1.22	1097.68	1.51	0.28	1093.77	1.15	25.85	<b>1081.31</b>
15 396	C,D					<b>1345.23</b>	0.00	7.17	1354.76	0.71	0.26	1358.21	0.96	23.80	<b>1345.23</b>
16 480	C,D					<b>1622.69</b>	0.00	20.00	1634.99	0.76	1.15	1635.16	0.77	39.90	<b>1622.69</b>
17 240	C,D					<b>707.79</b>	0.00	0.75	710.22	0.34	0.16	708.76	0.14	68.50	<b>707.79</b>
18 300	C,D					<b>998.73</b>	0.00	2.50	1009.53	1.08	0.18	998.83	0.01	42.73	<b>998.73</b>
19 360	C,D					<b>1366.86</b>	0.00	6.00	1381.88	1.10	0.25	1367.20	0.02	112.80	<b>1366.86</b>
20 420	C,D					<b>1821.15</b>	0.00	8.40	1840.57	1.03	0.55	1822.94	0.10	71.42	<b>1821.15</b>
Average				0.74	42.05		0.00	72.94		0.93	0.63		0.60	49.33	

9. Pentium II (400 MHz).

10. For C instances, see Mester and Bräysy (2004). Otherwise, see Mester (2004).

11. Pentium IV (2GHz).

12. Best value obtained in several experiments.

13. Pentium (900 MHz).



**Acknowledgments:** This work was partially funded by the Canadian Natural Sciences and Engineering Research Council under grants 227837-00, OGP00338816, 105384-02 and OGP00039682. This support is gratefully acknowledged. Thanks are due to Pierre Sormany for a useful suggestion.

## References

- Baldacci, R., Hadjiconstantinou, E.A., and Mingozzi, A. (2004). An exact algorithm for the capacitated vehicle routing problem based on a two-commodity network flow formulation. *Operations Research*. Forthcoming.
- Barbarosoğlu, G., and Ögür, D. (1999). A tabu search algorithm for the vehicle routing problem. *Computers & Operations Research*, 26:255–270.
- Bean, J.C. (1994). Genetic algorithms and random keys for the sequencing and optimization. *ORSA Journal on Computing*, 6:154–160.
- Beasley, J.E. (1983). Route-first cluster-second methods for vehicle routing. *Omega*, 11:403–408.
- Berger, J., and Barkaoui, M. (2004). A new hybrid genetic algorithm for the capacitated vehicle routing problem. *Journal of the Operational Research Society*, 54:1254–1262.
- Christofides, N., Mingozzi, A., and Toth, P. (1979). The vehicle routing problem. In: Christofides, N., Mingozzi, A., and Toth, P. (eds), *Combinatorial Optimization*, pages 315–338. 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.
- Codenetti, B., Manzini, G., Margara, L., and Resta, G. (1996). Perturbation: An efficient technique for the solution of very large instances of the Euclidean TSP. *INFORMS Journal on Computing*, 8:125–133.
- Cordeau, J.-F., and Laporte, G. (2004). Tabu search heuristics for the vehicle routing problem. In: Rego, C., and Alidaee, B. (eds), *Metaheuristic Optimization via Memory and Evolution: Tabu Search and Scatter Search*, pages 145–163. Kluwer, Boston.
- Cordeau, J.-F., Gendreau, M., and Laporte, G. (1997) A tabu search heuristic for periodic and multi-depot vehicle routing problems. *Networks*, 30:105–119.
- Cordeau, J.-F., Laporte, G., and Mercier, A. (2001). A unified tabu search heuristic for vehicle routing problems with time windows. *Journal of the Operational Research Society*, 52:928–936.
- Cordeau, J.-F., Laporte, G., and Mercier, A. (2004). An improved tabu search algorithm for the handling of route duration constraints in vehicle routing problems with time windows. *Journal of the Operational Research Society*. Forthcoming.
- Cordeau, J.F., Desaulniers, G., Desrosiers, J., Solomon, M.M., Soumis, F. (2002a). VRP with time windows. In: Toth, P., and Vigo, D. (eds), *The Vehicle Routing Problem*, pages 157–193. SIAM Monographs on Discrete Mathematics and Applications, Philadelphia.
- Cordeau, J.-F., Gendreau, M., Laporte, G., Potvin, J.-Y., and Semet, F. (2002b). A guide to vehicle routing heuristics. *Journal of the Operational Research Society*, 53:512–522.

- Desaulniers, G., Desrosiers, J., Erdmann, A., Solomon, M.M., and Soumis, F. (2002). VRP with pickup and delivery. In: Toth, P., and Vigo, D. (eds), *The Vehicle Routing Problem*, pages 225–242. SIAM Monographs on Discrete Mathematics and Applications, Philadelphia.
- Dongarra, J.J. (2004). Performance of various computers using standard linear equations software. Technical Report CS-89-85. Computer Science Department, University of Tennessee.
- Drezner, Z. (2003). A new genetic algorithm for the quadratic assignment problem. *INFORMS Journal on Computing*, 15:320–330.
- Dueck, G. (1993). New optimization heuristics: The great deluge algorithm and the record-to-record travel. *Journal of Computational Physics*, 104:86–92.
- Ergun, Ö., Orlin, J.B., and Steele-Feldman, A. (2003). Creating very large scale neighborhoods out of smaller ones by compounding moves: A study on the vehicle routing problem. Working paper, Massachusetts Institute of Technology.
- Fahrion, R., and Wrede, M. (1990). On a principle of chain-exchange for vehicle-routing problems (1-VRP). *Journal of the Operational Research Society*, 41:821–827.
- Fisher, M.L., and Jaikumar, R. (1981). A generalized assignment heuristic for the vehicle routing problem. *Networks*, 11:109–124.
- Gendreau, M., Hertz, A., and Laporte, G. (1992). New insertion and postoptimization procedures for the traveling salesman problem. *Operations Research*, 40:1086–1094.
- Gendreau, M., Hertz, A., and Laporte, G. (1994). A tabu search heuristic for the vehicle routing problem. *Management Science*, 40:1276–1290.
- Gendreau, M., Laporte, G., and Potvin, J.-Y. (2002). Metaheuristics for the VRP. In: Toth, P., and Vigo, D. (eds), *The Vehicle Routing Problem*, pages 129–154. SIAM Monographs on Discrete Mathematics and Applications, Philadelphia.
- Ghaziri, H. (1991). Solving routing problems by a self-organizing map. In: Kohonen, T., Makisara, K., Simula, O., and Kangas, J. (eds), *Artificial Neural Networks*, pages 829–834. North-Holland, Amsterdam.
- Ghaziri, H. (1996). Supervision in the self-organizing feature map: Application to the vehicle routing problem. In: Osman, I.H., and Kelly, J.P. (eds), *Meta-Heuristics: Theory and Applications*, pages 651–660, Kluwer, Boston.
- Gillett, B.E., and Miller, L.R. (1974). A heuristic algorithm for the vehicle dispatch problem. *Operations Research*, 22:340–349.
- Golden, B.L., Wasil, E.A., Kelly, J.P., and Chao, I-M. (1998). Metaheuristics in vehicle routing. In: Crainic, T.G., and Laporte, G. (eds), *Fleet Management and Logistics*, pages 33–56. Kluwer, Boston.
- Kinderwater, G.A.P., and Savelsbergh, M.W.P. (1997). Vehicle routing: Handling edge exchanges. In: Aarts, E.H.L., and Lenstra, J.K. (eds), *Local Search in Combinatorial Optimization*, pages 337–360. Wiley, Chichester.
- Laporte, G., and Semet, F. (2002). Classical heuristics for the capacitated VRP. In: Toth, P., and Vigo, D. (eds), *The Vehicle Routing Problem*, pages 109–128. SIAM Monographs on Discrete Mathematics and Applications, Philadelphia.

- Li, F., Golden, B.L., and Wasil, E.A. (2004). Very large-scale vehicle routing: New test problems, algorithms, and results. *Computers & Operations Research*. Forthcoming.
- Lin, S. (1965). Computer solution of the traveling salesman problem. *Bell System Technical Journal*, 44:2245–2269.
- Liu, F.-H., and Shen, S.-Y. (1999). A route-neighbourhood-based metaheuristic for vehicle routing problem with time windows. *European Journal of Operational Research*, 118:485–504.
- Matsuyama, Y. (1991). Self-organization via competition, cooperation and categorization applied to extended vehicle routing problems. In: *Proceedings of the International Joint Conference on Neural Networks*, pages 385–390. Seattle, WA.
- Mester, D. (2004). Private communication.
- Mester, D., and Bräysy, O. (2004). Active guided evolution strategies for the large scale vehicle routing problems with time windows. *Computers & Operations Research*. Forthcoming.
- Mladenović, N., and Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research*, 24:1097–1100.
- Moscato, P., and Cotta, C. (2003). A gentle introduction to memetic algorithms. In: Glover, F., and Kochenberger, G.A. (eds), *Handbook of Metaheuristics*, pages 105–144. Kluwer, Boston.
- Naddef, D., and Rinaldi, G. (2002). Branch-and-cut algorithms for the capacitated VRP. In: Toth, P., and Vigo, D. (eds), *The Vehicle Routing Problem*, pages 53–84. SIAM Monographs on Discrete Mathematics and Applications, Philadelphia.
- Or, I. (1976). *Traveling Salesman-Type Combinatorial Problems and their Relation to the Logistics of Regional Blood Banking*. Ph.D. Thesis, Northwestern University, Evanston, IL, 1976.
- Osman, I.H. (1993). Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. *Annals of Operations Research*, 41:421–451.
- Paessens, H. (1988). The savings algorithm for the vehicle routing problem. *European Journal of Operational Research*, 34:336–344.
- Potvin, J.-Y. (1996). Genetic algorithms for the traveling salesman problem. *Annals of Operations Research*, 63:339–370.
- Potvin, J.-Y., and Rousseau, J.-M. (1995). An exchange heuristic for routing problems with time windows. *Journal of the Operational Research Society*, 46:1433–1446.
- Prins, C. (2004). A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers & Operations Research* 31:1985–2002.
- Rechenberg, I. (1973). *Evolutionsstrategie*. Fromman-Holzboog, Stuttgart, Germany.
- Reeves, F. (2003). Genetic algorithms. In: Glover, F., and Kochenberger, G.A. (eds), *Handbook of Metaheuristics*, pages 55–82. Kluwer, Boston.
- Rego, C. (1998). A subpath ejection method for the vehicle routing problem. *Management Science*, 44:1447–1459.

- Rego, C., and Roucairol, C. (1996). A parallel tabu search algorithm using ejection chains for the vehicle routing problem. In: *Meta-Heuristics: Theory and Applications*, Kluwer, Boston, 661–675.
- Reimann, M., Doerner, K., and Hartl, R.F. (2004). D-Ants: Savings based ants divide and conquer the vehicle routing problem. *Computers & Operations Research*. Forthcoming.
- Renaud, J., Boctor, F.F., and Laporte, G. (1996). A fast composite heuristic for the symmetric traveling salesman problem. *INFORMS Journal on Computing*, 8:134–143.
- Rochat, Y., and Taillard, É.D. (1995). Probabilistic diversification and intensification in local search for vehicle routing. *Journal of Heuristics*, 1:147–167.
- Shaw, P. (1998). Using constraint programming and local search methods to solve vehicle routing problems. In: Maher, M., and Puget, J.-F. (eds), *Principles and Practice of Constraint Programming*, pages 417–431. *Lecture Notes in Computer Science*, Springer-Verlag, New York.
- Schumann, M., and Retzko, R. (1995). Self-organizing maps for vehicle routing problems — minimizing an explicit cost function. In: Fogelman-Soulie, F. (ed.), *Proceedings of the International Conference on Artificial Neural Networks*, pages 401–406. Paris.
- Solomon, M.M. (1987). Algorithms for the vehicle routing and scheduling problems with time windows. *Operations Research*, 35:254–265.
- Taillard, É.D. (1993). Parallel iterative search methods for vehicle routing problem. *Networks*, 23:661–673.
- Tarantilis, C.-D., and Kiranoudis, C.T. (2002). Bone Route: An adaptive memory-based method for effective fleet management. *Annals of Operations Research*, 115:227–241.
- Thompson, P.M., and Psaraftis, H.N. (1993). Cyclic transfer algorithms for multi-vehicle routing and scheduling problems. *Operations Research*, 41:935–946.
- Toth, P., and Vigo, D. (2003). The granular tabu search and its application to the vehicle routing problem. *INFORMS Journal on Computing*, 15:333–346.
- van Breedam, A. (1994). *An Analysis of the Behavior of Heuristics for the Vehicle Routing Problem for a Selection of Problems with Vehicle-Related, Customer-Related, and Time-Related Constraints*. Ph.D. Dissertation, University of Antwerp.
- Voudouris, C. (1997). *Guided Local Search for Combinatorial Problems*. Dissertation, University of Essex, United Kingdom.
- Xu, J., and Kelly, J.P. (1996). A network flow-based tabu search heuristic for the vehicle routing problem. *Transportation Science*, 30:379–393.