Metaheuristics for the Vehicle Routing Problem

by

Michel Gendreau[1]
Gilbert Laporte[2]
Jean-Yves Potvin[1]

September, 1998
Revised: August, 1999

Les Cahiers du GERAD

G–98–52

[1] Département d'informatique et recherche opérationnelle, et Centre de recherche sur les transports, Université de Montréal, Case postale 6128, Succursale "Centre-ville", Montréal, Canada H3C 3J7.

[2] GERAD, École des Hautes Études Commerciales, 3000 chemin de la Côte-Sainte-Catherine, Montréal, Canada H3T 2A7.

Abstract

In recent years several *metaheuristics* have been proposed for the *Vehicle Routing Problem*. This article reviews the main metaheuristics for this problem: simulated annealing, deterministic annealing, tabu search, genetic algorithms, ant systems, neural networks. Several comparative computational results are reported.

Key words: Vehicle routing problem, metaheuristics.

Résumé

Au cours des dernières années, plusieurs *métaheuristiques* ont été proposées pour le *problème de tournées de véhicules*. Dans cet article, on passe les principales métaheuristiques en revue: le recuit simulé, le recuit déterministe, la recherche avec tabous, les algorithmes génétiques, les systèmes de fourmis, les réseaux de neurones. On présente plusieurs résultats numériques.

Mots-clefs: Problème de tournées de véhicules, métaheuristiques.

1 Introduction

In recent years several *metaheuristics* have been proposed for the VRP. These are general solution procedures that explore the solution space to identify good solutions and often embed some of the standard route construction and improvement heuristics described in the previous chapter. In a major departure from classical approaches, metaheuristics allow deteriorating and even infeasible intermediary solutions in the course of the search process. The best known metaheuristics developed for the VRP typically identify better local optima than earlier heuristics, but they also tend to be more time consuming.

As far as we are aware, six main types of metaheuristics have been applied to the VRP: 1) Simulated Annealing (SA), 2) Deterministic Annealing (DA), 3) Tabu Search (TS), 4) Genetic Algorithms (GA), 5) Ant Systems (AS), and 6) Neural Networks (NN). The first three algorithms, SA, DA and TS, start from an initial solution x_1, and move at each iteration t from x_t to a solution x_{t+1} in the neighbourhood $N(x_t)$ of x_t, until a stopping condition is satisfied. If $f(x)$ denotes the cost of x, then $f(x_{t+1})$ is not necessarily less than $f(x_t)$. As a result, care must be taken to avoid cycling. GA examines at each step a population of solutions. Each population is derived from the preceding one by combining its best elements and discarding the worst. AS is a constructive approach in which several new solutions are created at each iteration using some of the information gathered at previous iterations. As was pointed out by Taillard *et al.* [66], TS, GA and AS are methods that record, as the search proceeds, information on solutions encountered and use it to obtain improved solutions. NN is a learning mechanism that gradually adjusts a set of weights until an acceptable solution is reached. The rules governing the search differ in each case and these must also be tailored to the shape of the problem at hand. Also, a fair amount of creativity and experimentation is required. Our purpose is to survey some of the most representative applications of local search algorithms to the VRP. For generic articles and textbooks on these metaheuristics, the reader is referred to Rumelhart and McClelland [60], Wasserman [76], van Laarhoven and Aarts [74], Goldberg [33], Davis [17], Pirlot [49], Reeves [55], Dorigo, Maniezzo and Colorni [19], Osman and Kelly [47], Osman and Laporte [48], Aarts and Lenstra [1], and Glover and Laguna [32]. The two articles by Gendreau, Laporte and Potvin [27] and by Golden *et al.* [35] report how various metaheuristic methods have been applied to the VRP and to the VRP with time windows. In the following six sections of this chapter we report on implementations of SA, DA, TS, GA, AS and NN to the solution of the VRP. Some of this material is borrowed or adapted from Gendreau, Laporte and Potvin [27].

2 Simulated Annealing

At iteration t of Simulated Annealing, a solution x is drawn randomly in $N(x_t)$. If $f(x) \leq f(x_t)$, then x_{t+1} is set equal to x; otherwise

$$x_{t+1} := \begin{cases} x & \text{with probability } p_t \\ x_t & \text{with probability } 1 - p_t, \end{cases}$$

where p_t is usually a decreasing function of t and of $f(x) - f(x_t)$. It is common to define p_t as

$$p_t = \exp\Big(-\big[f(x) - f(x_t)\big]/\theta_t\Big), \tag{1}$$

where θ_t denotes the *temperature* at iteration t. The rule employed to define θ_t is called a *cooling schedule*. Typically, θ_t is a decreasing step function of t: initially, θ_t is set equal to a given value $\theta_1 > 0$, and is multiplied by a factor $\alpha(0 < \alpha < 1)$ after every T iterations, so that the probability of accepting a worse solution should decrease with time. Three common stopping criteria are: 1) the value f^* of the incumbent x^* has not decreased by at least $\pi_1\%$ for at least k_1 consecutive cycle of T iterations; 2) the number of accepted moves has been less than $\pi_2\%$ of T for k_2 consecutive cycles of T iterations; 3) k_3 cycles of T iterations have been executed.

2.1 Two early SA algorithms

Two early implementations of SA in the context of the VRP are those of Robusté, Daganzo and Souleyrette [58]. In the first case, the authors define a neighbourhood structure by combining several mechanisms: reversing part of a route, moving part of a route into another part of the same route, trading vertices between two routes. The algorithm was tested on four instances ($n = 80, 100, 120, 500$) but no comparisons with alternative methods are available. In the second implementation, Alfa, Heragu and Chen [2] used a route-first, cluster-second heuristic (Beasley [6]) to construct a first solution, followed by 3-opt (Lin [42]) for the search process. The method was applied to three instances ($n = 30, 50, 75$) and did not produce competitive results.

2.2 Osman's SA algorithms

Osman's [46] implementation of SA is much more involved, and also more successful. It uses a better starting solution, some parameters of the algorithm are adjusted in a trial phase, richer solution neighbourhoods are explored, and the cooling schedule is more sophisticated. The neigbourhood structure of this algorithm uses a λ-*interchange generation mechanism* in which two routes p and q are first selected, together with two subsets of customers S_p and S_q, one from each route, satisfying $|S_p| \leq \lambda$ and $|S_q| \leq \lambda$. The operation swaps the customers of S_p with those of S_q as long as this is feasible. The sets S_p or S_q can be empty and therefore this family of operations includes simply *shifting* customers from one route to another. As the number of combinations of route pairs and choices of S_p and S_q is usually large, this procedure is implemented with $\lambda = 1$ or 2 and in the most efficient versions of the algorithm, the search stops as soon as an improving move is identified (when this does not happen, the whole neighbourhood must be explored). An inferior version of this algorithm consists of examining a whole neighbourhood and of implementing the best move.

The algorithm that was implemented was tested on symmetric VRPs with an unspecified number of vehicles. It operates as follows.

Phase 1. Descent algorithm.

Step 1 (Initial solution). Generate an initial solution by means of the Clarke and Wright algorithm [12].

Step 2 (Descent). Search the solution space using the λ-interchange scheme. Implement an improvement as soon as it is identified. Stop whenever an entire neighbourhood exploration yields no improvement.

Phase 2. SA Search.

Step 1 (Initial solution). Use as a starting solution the incumbent obtained at the end of Phase 1, or a solution produced by the Clarke and Wright algorithm. Perform a complete neighbourhood search using the λ-interchange generation mechanism without, however, implementing any move. Record $\triangle_{\max}$ and $\triangle_{\min}$, the largest and smallest absolute changes in the objective function, and compute β, the number of feasible (potential) exchanges. Set $\theta_1 := \triangle_{\max}$, $\delta := 0$, $k := 1$, $k_3 := 3$ $t := 1$, $t^* := 1$ (this is the iteration at which the best known solution has been identified within the current cycle). Let x_1 be the current solution and $x^* := x_1$.

Step 2 (Next solution). Explore the neighbourhood of x_t using λ-interchanges. When a solution x with $f(x) < f(x_t)$ is encountered, set $x_{t+1} := x$; if $f(x) < f(x^*)$, set $x^* := x$ and $\theta^* := \theta_k$. If a whole exploration yields no better solution than x_t, let x be the best solution encountered in the neighbourhood of x_t and set

$$x_{t+1} := \begin{cases} x & \text{with probability } p_t \\ x_t & \text{with probability } 1 - p_t, \end{cases}$$

where p_t is defined by (1). If $x_{t+1} := x_t$, set $\delta := 1$.

Step 3 (Temperature update). *Occasional increment rule:* if $\delta = 1$, set $\theta_{t+1} := \max\{\theta_t/2, \theta^*\}$, $\delta := 0$ and $k := k+1$. *Normal decrement rule:* If $\delta = 0$, set $\theta_{t+1} := \theta_t / \left[(n\beta + n\sqrt{t})\triangle_{\max} \triangle_{\min}\right]$. Set $t := t + 1$. If $k = k_3$, stop. Otherwise, go to Step 2.

The cooling schedule employed by Osman differs from what is commonly done in SA. The temperature is not decreased continuously, nor as a step function. Instead, it decreases continuously as long as the current solution is modified. Whenever $x_{t+1} = x_t$, the current temperature is either halved, or replaced by the temperature at which the incumbent was identified. It is not clear to what extent this modified cooling schedule is instrumental to the success of the algorithm.

The algorithm was implemented with $\lambda = 1$, using the best Phase 1 solution to initiate Phase 2. In total, 26 instances were tested. We report in Table 1 results obtained on the classical 14 instances proposed by Christofides, Mingozzi and Toth [11], some of which contain a distance restriction, i.e., an upper limit L is imposed on the length of any vehicle route. Bold numbers mean that the algorithm has identified a best known solution.

Table 1 indicates that Osman's SA algorithm generally produces good results but it sometines misses the mark significantly and rarely identifies a best known solution. Computing times tend to be relatively long. Overall, applying SA to the VRP does not yield results that are competitive with those produced by the best TS implementations.

Instance	$f^{*(1)}$ solution value	Best known	Time(2)
E051-05e	528	524.61(1,3,4,5,6,8,9,10,12)	*167.4*
E076-10e	838.62	835.26(3,5)	*6434.3*
E101-08e	829.18	826.14(3,4,5)	*9334.0*
E101-10c	826	819.56(3,4,5,6,9,11,12)	*632.0*
E121-07c	1176	1042.11(1,3,4,5,6)	*315.8*
E151-12c	1058	1028.42(3)	*5012.3*
E200-17c	1378	1291.45(7)	*2318.1*
D051-06c	**555.43**	555.43(1,3,4,5,6,12)	*3410.2*
D076-11c	**909.68**	909.68(1,3,4,6)	*626.5*
D101-09c	866.75	865.94(3,4)	*957.2*
D101-11c	890	866.37(1,3,4,6,12)	*305.2*
D121-11c	1545.98	1541.14(3)	*7622.5*
D151-14c	1164.12	1162.55(3)	*84301.2*
D200-18c	1417.85	1395.85(7)	*5708.0*

(1) Osman [46].
(2) Seconds on a VAX 86000 computer.
(3) Taillard [65].
(4) Gendreau, Hertz and Laporte [26].
(5) Xu and Kelly [79].
(6) Rego and Roucairol [57].
(7) Rochat and Taillard [59].
(8) Bullnheimer, Hartl and Strauss [9].
(9) Bullnheimer, Hartl and Strauss [10].
(10) Optimal solution (see Hadjiconstantinou, Christofides and Mingozzi [36]).
(11) Optimal solution (see Golden *et al.* [35]).
(12) Toth and Vigo [70].

Table 1: Computational results for Osman's algorithm (with first improving moves)

2.3 Van Breedam's experiments

We note in closing this section that Van Breedam [72] has compared and tested several versions of SA using different neighbourhood structures. Tests were conducted on the 14 Christofides, Mingozzi and Toth [11] instances. These experiments are useful in helping to identify best SA strategies, but overall they confirm the superiority of TS based heuristics.

3 Deterministic Annealing

Deterministic Annealing operates in a way that is similar to SA, except that a deterministic rule is used for the acceptance of a move. Two standard implementations of this technique are threshold accepting (Dueck and Scheurer [21]) and record to record travel (Dueck [20]). At iteration t of a threshold accepting algorithm, solution x_{t+1} is accepted if $f(x_{t+1}) <$

$f(x_t)+\theta_1$, where θ_1 is a user controlled parameter. In record-to-record travel a *record* is the best solution x^* encountered during the search. At iteration t, solution x_{t+1} is accepted if $f(x_{t+1}) < \theta_2 f(x_t)$, where θ_2 is a user controlled parameter in general slightly larger than 1. Golden *et al.* [35] have applied a record-to-record travel heuristic to 20 large scale instances of the VRP, eight of which include distance restrictions. Data sets for these instances can be obtained on the web site *http://www-Bus.colorado.edu/Publications/workingpapers/kelly.* Comparisons were made with results obtained by applying the Xu-Kelly TS heuristic (to be described in Section 5) on the same instances. Not only is the record-to-record heuristic much faster than the Xu and Kelly implementation, but it generates a better solution in 11 cases out of 20. Results taken from Golden *et al.* [35] are reported in Table 2. As explained by Toth and Vigo [70], all instances of this series involving a distance restriction contain errors in the Xu and Kelly column and have therefore been omitted.

Instance	**Record-to-record**		**Xu–Kelly's TS**		**Best known solution value**
	f^*	**Time**$^{(1)}$	f^*	**Time**$^{(2)}$	
E241-22k	720.44	*5.69*	**666.84**	*2314.00*	666.84$^{(3)}$
E253-27k	**881.04**	*6.01*	881.07	*1465.77*	881.04$^{(4)}$
E256-14k	**587.09**	*23.01*	589.10	*340.20*	587.09$^{(4)}$
E301-28k	1029.21	*8.15*	**973.60**	*4101.02*	973.60$^{(3)}$
E321-30k	**1103.69**	*21.83*	1118.09	*1577.30*	1103.69$^{(4)}$
E324-16k	749.15	*31.49*	**746.56**	*501.82*	746.56$^{(3)}$
E361-33k	1403.05	*12.42*	**1338.74**	*5718.38*	1338.74$^{(3)}$
E397-34k	**1364.23**	*32.62*	1377.79	*4340.07*	1364.23$^{(4)}$
E400-18k	934.33	*69.19*	**932.68**	*852.72*	932.68$^{(3)}$
E421-41k	1875.17	*31.05*	**1831.62**	*103839.73*	1831.62$^{(3)}$
E481-38k	1657.93	*47.55*	**1656.66**	*8943.45*	1556.66$^{(3)}$
E484-19k	**1137.18**	*101.09*	1140.72	*1151.10*	1137.18$^{(4)}$

(1) Minutes on a 100MHz Pentium-based PC.
(2) Minutes on a DEC ALPHA Workstation.
(3) Xu and Kelly [79].
(4) Golden *et al.* [35].

Table 2: Computational results for the record-to-record algorithm of Golden *et al.*

4 Tabu Search

In Tabu Search, sequences of solutions are examined as in SA, but the next move is made to the best neighbour of the current solution x_t. To avoid cycling, solutions that were recently examined are forbidden, or *tabu*, for a number of iterations. To alleviate time and memory requirements, it is customary to record an attribute of tabu solutions rather than the solutions themselves. The basic TS mechanism can be enhanced by several computational features such as diversification and intensification strategies, as described by Hertz, Taillard and de Werra [37], and by Glover and Laguna [31, 32], for example.

Over the last ten years or so, TS has been applied to the VRP by several authors. Some of the first TS algorithms (Willard [78], Pureza and França [54]) did not yield impressive results, but subsequent implementations were much more successful. These include the work of Osman [46], Gendreau, Hertz and Laporte [26], Taillard [65], Xu and Kelly [79], Rego and Roucairol [57] Rego [56], Barbarosoglu and Ozgur [3]. In addition, Rochat and Taillard [59] have introduced a useful and powerful concept, the *adaptive memory*, which can be used to enhance any TS based algorithm. In the same vein, Toth and Vigo [70] have introduced *granular tabu search* whose principles have a far reaching applicability.

4.1 Two early TS algorithms

One of the first attempts to apply tabu search to the VRP is due to Willard [78]. Here, the solution is first transformed into a giant tour by replication of the depot and neighbourhoods are defined as all feasible solutions that can be reached from the current solution by means of 2-opt or 3-opt exchanges (Lin [42]). The next solution is determined by the best non-tabu move. On three of the Christofides, Mingozzi and Toth [11] benchmark problems, the proposed algorithm does not appear to be competitive with most known approximation algorithms. Pureza and França [54] define the neighbours of a solution by moving a vertex to a different route, or by swapping vertices between two routes while preserving feasibility. As in Willard, the best non-tabu feasible move is selected at each iteration. While better than Willard's algorithm, this implementation did not produce especially good results. Further research has shown that more sophisticated search mechanisms are required to make TS work.

4.2 Osman's TS algorithm

In Osman [46], neighbourhoods are again defined by means of the λ-interchange generation mechanism, with $\lambda = 2$. This includes a combination of 2-opt moves, vertex reassignments to different routes, and vertex interchanges between two routes. In one version of the algorithm called BA (best-admissible), the whole neighbourhood is explored and the best non-tabu feasible move is selected. In the other version, FBA (first-best-admissible), the first admissible improving move is selected if one exists; otherwise the best admissible move is implemented. Results reported in Table 3 indicate that these two TS implementations produce excellent results but these can still be improved in most cases.

4.3 Taburoute

With respect to the previous TS implementations, the Taburoute algorithm of Gendreau, Hertz and Laporte [26] is rather involved and contains several innovative features. The neighbourhood structure is defined by all solutions that can be reached from the current solution by removing a vertex from its current route, and inserting it into another route containing one of its p nearest neighbours using GENI, a *GENeralized Insertion* procedure developed by Gendreau, Hertz and Laporte [25] for the *Traveling Salesman Problem* (TSP). This may result in eliminating an existing route or in creating a new one. A second important feature of Taburoute is that the search process examines solutions that may be infeasible with respect to the capacity or maximum route length constraints. More precisely, the objective function contains two penalty terms, one measuring overcapacity, the other measuring overduration, each weighted by a self-adjusting parameter: every ten iterations, each parameter is divided by 2 if all 10 previous solutions were feasible, or multiplied by 2 if they were all infeasible. This way of proceeding produces a mix of feasible and infeasible solution and lessens the likelihood of being trapped in a local minimum. At various points during the search process, Taburoute reoptimizes the route in which a vertex has just been inserted. This is achieved by using the US (*Unstringing & Stringing*) TSP post-optimization routine also developed by Gendreau, Hertz and Laporte [25].

Taburoute does not actually use a tabu list, but random tabu tags. Whenever a vertex is moved from route r to route s at iteration t, its reinsertion into route r is forbidden until iteration $t + \theta$, where θ is an integer randomly drawn from the interval $[5, 10]$. Yet another feature of Taburoute is the use of a *diversification* strategy which consists of penalizing vertices that have been moved frequently in order to increase the probability of considering slow moving vertices. The objective function is artificially increased by adding to it a term proportional to the absolute frequency of movement of the vertex v currently being considered. Finally, Taburoute uses *false starts*. Initially, several solutions are generated and a limited search is carried out on each of them. The best identified solution is then selected as a starting point for the main search.

We now provide a short description of Taburoute. The readers are referred to the original article [26] for a detailed discussion of the parameter choices. In what follows, W is the set of vertices considered as candidates for reinsertion into another route at each iteration, $q \leq |W|$ is the number of these vertices for which a tentative reinsertion is actually made, and k is the number of consecutive iterations without improvement.

Step 1 (Initialization). Generate $\lceil \sqrt{n}/2 \rceil$ initial solutions and perform TS with $W = V \setminus \{v_0\}$, $q = 5m$ and $k = 50$. This value of q ensures that the probability of selecting one vertex from each route is at least 90%.

Step 2 (Solution improvement). Starting with the best solution observed in Step 1, perform TS with $W = V \setminus \{v_0\}$, $q = 5m$ and $k = 50n$.

Step 3 (Intensification). Starting with the best solution observed in Step 2, perform TS with $k = 50$. Here W is the set of the $\lfloor |V|/2 \rfloor$ vertices that have been most often moved in Steps 1 and 2, and $q = |W|$.

As can be seen from Table 3, Taburoute produces high quality results and often yields a best known solution.

4.4 Taillard's algorithm

The Taillard [65] TS implementation contains some of the features of Taburoute, namely random tabu durations and diversification. It defines neighbourhood using the λ-interchange generation mechanism (Osman [46]). Rather than executing the insertions with GENI, the algorithm uses standard insertions, thus enabling each insertion to be carried out in less time, and feasibility is always maintained. Every so often, individual routes are reoptimized using the optimization algorithm of Volgenant and Jonker [75].

A novel feature of Taillard's algorithm is the decomposition of the main problems into subproblems. In planar problems, these subproblems are obtained by initially partitioning vertices into sectors centered at the depot, and into concentric regions within each sector. Each subproblem can be solved independently, but periodical moves of vertices to adjacent sectors are necessary. This makes sense when the depot is centered and vertices are uniformly distributed in the plane. For non-planar problems, and for planar problems not possessing these properties, the author suggests a different partitioning method based of the computation of shortest spanning arborescences rooted at the depot. This decomposition method is particularly well suited for parallel implementation as subproblems can then be distributed among the various processors. The combination of these strategies yields excellent computational results.

4.5 Xu and Kelly's algorithm

With respect to the previous two TS algorithms, Xu and Kelly [79] use a more sophisticated neighbourhood structure. They consider swaps of vertices between two routes, a global repositioning of some vertices into other routes, and local route improvements. The global repositioning strategy solves a network flow model to optimally relocate given numbers of vertices into different routes. Approximations are developed to compute the ejection and insertion costs, taking vehicle capacity into account. Route reoptimizations are performed by means of 3-opt exchanges (Lin [42]) and a TS improvement routine. The algorithm is governed by several parameters which are dynamically adjusted through the search. A pool of best solutions is memorized and periodically used to reinitiate the search with new parameter values. Overall, this algorithm has produced several best known solutions on benchmark instances, but it is fair to say that it is not as effective as some other TS implementations. It tends to require a substantial computational effort and properly tuning its many parameters can be problematic.

4.6 Rego and Roucairol's algorithms

The main feature of the Rego and Roucairol [57] TS algorithm is the use of ejection chains to move from one solution to the next. An ejection consists of moving a vertex to the position occupied by another vertex, thus creating a chain reaction of ℓ levels. For a given route orientation, denote by v_{i-1} the predecessor of v_i and by v_{i+1} its successor. An ℓ level ejection chain consists of replacing the triplets $(v_{i-1}^k, v_i^k, v_{i+1}^k)$ $(k = 0, \dots, \ell)$ by the triplets $(v_{i-1}^k, v_i^{k-1}, v_{i+1}^k)$ $(k = 1, \dots, \ell)$ and of relocating v_i^ℓ. A *legitimacy condition* is defined to ensure that the resulting solution remains feasible, i.e., that no arc appears more than once. At a general step of the algorithm, a number of vertices are considered as candidate for an ejection, together with their closest neighbours that do not yield an illegetimate ejection chain. As in Taburoute, infeasible intermediate solutions are considered. A parallel implementation of this procedure was developed. Again, this TS implementation yields good quality results, but does not measure up to the best known algorithms. A variant of this algorithm, called the Subpath Ejection Method, was recently introduced by Rego [56]. Unfortunately, it does not seen to improve upon previous TS algorithms.

4.7 Barbarosogly and Ozgur's algorithm

Barbarosogly and Ozgur [3] describe a rather simple TS algorithm containing no diversification strategy and in which only feasible solutions are examined. Neighbour solutions are defined by means of a λ-interchange scheme which favours vertices relatively far from the centroid of their current route and close to the centroid of the new route. Route re-optimizations are performed by applying a 2-opt procedure. The method was applied to the six capacitated instances of the Christofides, Mingozzi and Toth [11] set and yielded interesting results.

4.8 The Adaptive Memory Procedure of Rochat and Taillard

One of the most interesting developments to have occurred in the area of Tabu Search in recent years is the concept of *Adaptive Memory* developed by Rochat and Taillard [59]. It is mostly used in TS, but its applicability is not limited to this type of metaheuristic. An adaptive memory is a pool of good solutions that is dynamically updated throughout the search process. Periodically, some elements of these solutions are extracted form the pool and combined differently to produce new good solution. In the VRP, vehicle routes selected from several solutions will be used as a starting point. The extraction process gives a larger weight to those routes belonging to the best solutions. When selecting these routes, care must be taken to avoid including the same customer twice in a solution. This restriction means that the selection process will often terminate with a partial solution that will have to be completed using a construction heuristic. In the example depicted in Figure 1, extracting routes A,D and H from a memory of two solutions results in a partial solution. Rochat and Taillard have shown that the application of an adaptive memory procedure can enhance a search strategy. This has enabled them to obtain two new best solutions on the 14 standard VRP benchmark instances.

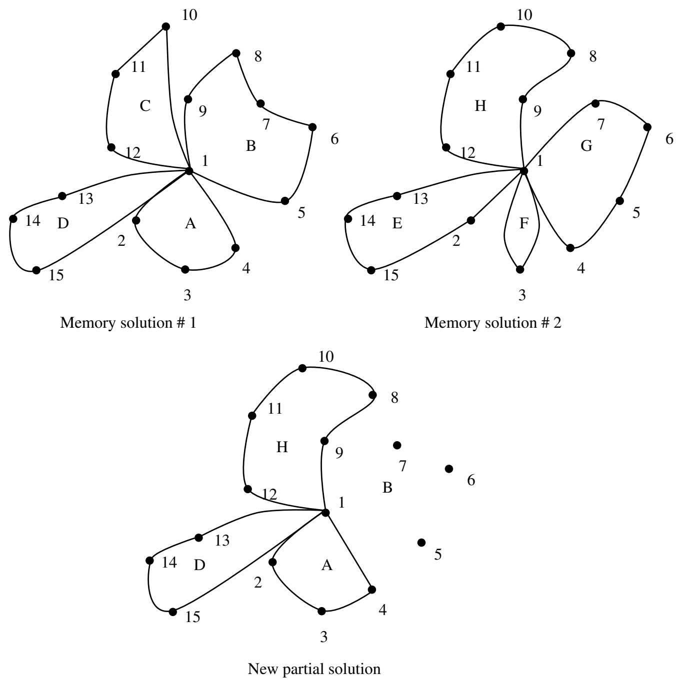

Figure 1: Creating a new partial solution in the adaptive memory procedure.

4.9 The Granular Tabu Search of Toth and Vigo

Granular Tabu Search (GTS) is yet another very promising concept. It was recently introduced by Toth and Vigo [70] and has yielded excellent results on the VRP. The main idea behind GTS stems from the observation that the longer edges of a graph only have a small likelihood if belonging to an optimal solution. Therefore, by eliminating all edges whose length exceeds a *granularity threshold* several unpromising solutions will never be considered by the search process. Toth and Vigo suggest using $\nu = \beta\bar{c}$, where β a *sparsification parameter* typically chosen in the interval $[1.0, 2.0]$, and $\bar{c}$ is the average edge length of a solution obtained by a fast heuristic. If $\beta \in [1.0, 2.0]$, then the percentage of remaining edges in the graph tends to be in the $10\% - 20\%$ range. In practice the value of β is dynamically adjusted whenever the incumbent has not improved for a set number of iterations, and periodically decreased to its initial value. Neighbour solutions are obtained by performing a limited number of edge exchanges within the same route or between two routes. The authors propose a procedure capable of examining all potential exchanges in $O(|E(\nu)|)$ time, where $E(\nu) = \{(i,j) \in E : c_{ij} \leq \nu\} \cup I$, and I is a set of important edges such as those incident to the depot or belonging to high quality solutions. Toth and Vigo have implemented a version of GTS containing several of the features of Taburoute. As results presented in table 3 show, GTS produces excellent solutions within very short computing times.

4.10 Computational comparison

As rightly pointed out by Barr *et al.* [4] and by Golden *et al.* [35], properly testing meta-heuristics is fraught with difficulties:

- These algorithms are usually governed by several user controlled parameters. Parameter setting should be done on a different set of instances from those used for the final tests.
- "Standard" results should be reported for one setting of the parameters. "Best" results corresponding to the best parameter values should be given alongside.
- It is difficult to interpret computing times in the case of parallel implementations.
- Comparable instance values should be used in all comparisons. For example, in the case of the VRP, different distance truncation rules yield vastly different results (see, Gendreau, Hertz and Laporte, [26]).

There is now a consensus among researchers that stricter testing practices must be enforced, but this has not always been the case in the past. The comparative results presented in Table 3 should be read with these remarks in mind. These show that TS generally yields very good solution values on the 14 benchmark instances. Two of the best known solution values are known to be optimal and the remaining ones are probably very close to being optimal. It is now time to move to a new set of larger instances, including those recently developed by Golden *et al.* [35].

On the algorithmic side, time has probably come to concentrate on the development of faster, simpler (with fewer parameters) and more robust algorithms, even if this causes a small loss in solution quality. These attributes are essential if we want to see more of our algorithms implemented in commercial packages.

Instance	Osman (BA)		Taillard	Taburoute			Rochat – Taillard	Xu – Kelly		Rego – Roucairol	Toth – Vigo (GTS)		Best known solution value
				standard		best							
	f^*	**Time** [1]	f^*	f^*	**Time** [2]	f^*	f^*	f^*	**Time** [3]	f^*	f^*	**Time** [6]	
E051-05e	**524.61**	*1.12*	**524.61**	**524.61**	*6.0*	**524.61**		**524.61** [4,5]	*29.22* [4,5]	**524.61**	**524.61**	*0.81*	524.61 [4,7,8,9,11,12,13,14,16]
E076-10e	844	*1.18*	**835.26**	835.77	*53.8*	835.32		**835.26** [4,5]	*48.80* [4,5]	835.32	838.60	*2.21*	835.26 [4,8]
E101-08e	835	*11.25*	**826.14**	829.45	*18.4*	**826.14**		**826.14** [4,5]	*71.93* [4,5]	827.53	828.56	*2.39*	826.14 [4,8,9]
E101-10c	819.59	*6.79*	**819.56**	**819.56**	*16.0*	**819.56**		**819.56** [4,5]	*56.61* [4,5]	**819.56**	**819.56**	*1.10*	819.56 [4,8,9,11,13,15,16]
E121-07c	**1042.11**	*23.31*	**1042.11**	1073.47	*22.2*	**1042.11**		**1042.11** [4,5]	*91.23* [4,5]	**1042.11**	1042.87	*3.18*	1042.11 [4,7,8,9,11]
E151-12c	1052	*51.25*	**1028.42**	1036.16	*58.8*	1031.07		1029.56 [4,5]	*149.90* [4,5]	1044.35	1033.21	*4.51*	1028.42 [8]
E200-17c	1354	*32.88*	1298.79	1322.65	*90.9*	1311.35	**1291.45**	1298.58 [4,5]	*272.52* [4,5]	1334.55	1318.25	*7.50*	1291.45 [10]
D051-06c	**555.44**	*2.34*	**555.43**	**555.43**	*13.5*	**555.43**		**555.43** [5]	*30.67* [5]	**555.43**	**555.43**	*0.86*	555.43 [4,7,8,9,11,16]
D076-11c	913	*3.38*	**909.68**	913.23	*54.6*	**909.68**		965.62 [5]	*102.13* [5]	**909.68**	920.72	*2.75*	909.68 [7,8,9,11]
D101-09c	866.75	*20.00*	**865.94**	**865.94**	*25.6*	**865.94**		881.38 [5]	*98.15* [5]	866.75	869.48	*2.90*	865.94 [8,9]
D101-11c	**866.37**	*92.98*	**866.37**	**866.37**	*65.7*	**866.37**		915.24 [5]	*152.98* [5]	**866.37**	**866.37**	*1.41*	866.37 [7,8,9,11,16]
D121-11c	1547	*22.38*	**1541.14**	1573.81	*59.2*	1545.93		1618.55 [5]	*201.75* [5]	1550.17	1545.51	*9.34*	1541.14 [8]
D151-14c	1422	*40.73*	**1162.55**	1177.76	*71.0*	1162.89		No solution	*168.08* [5]	1164.12	1173.12	*5.67*	1162.55 [8]
D200-18c	1422	*55.17*	1397.94	1418.51	*99.8*	1404.75	**1395.85**	1439.29 [5]	*368.37* [5]	1420.84	1435.74	*9.11*	1395.85 [10]

(1) Minutes on a VAX 8600.
(2) Minutes on a Silicon Graphics Workstation. (36MHz, 5.7 Mflops).
(3) Minutes on a DEC ALPHA Workstation. (DEC OSF/1 v 3.0).
(4) Xu and Kelly [79].
(5) Golden *et al.* [35].
(6) Minutes on a Pentium 200 MHz PC (about three times faster than the Silicone Graphics Workstation used for Taburoute, and twice slower than the DEC ALPHA Workstation used by Xu and Kelly).
(7) Osman [46].
(8) Taillard [65].
(9) Gendreau, Hertz and Laporte [26].
(10) Rochat and Taillard [59].
(11) Rego and Roucairol [57], parallel implementation.
(12) Bullnheimer, Hartl and Strauss [9].
(13) Bullnheimer, Hartl and Strauss [10].
(14) Optimal solution (see Hadjiconstantinou, Christofides and Mingozzi [36]).
(15) Optimal solution (see Golden *et al.* [35]).
(16) Toth and Vigo [70].

Table 3: Computational comparison of TS algorithms.

5 Genetic algorithms

A genetic algorithm (GA) is a randomized global search technique that solves problems by imitating processes observed during natural evolution. This problem solving paradigm was initially proposed by Holland [38], although it took another ten years before it was fully recognized in the research community. A pure GA is a generic problem-solving method that uses little heuristic information about the problem domain. It can thus be applied to a wide range of ill-defined problems that do not lend themselves to specialized methods. Basically, a GA evolves a population of bitstrings, or chromosomes, where each chromosome encodes a solution to a particular instance. This evolution takes place through the application of operators that mimic natural phenomena observed in nature (e.g., reproduction, mutation). A simple GA will be described in the following. We then explain how this paradigm can be applied to a sequencing problem like the VRP.

5.1 A simple genetic algorithm

Starting from some randomly generated initial population of chromosomes $X^1 = \{x_1^1, \ldots, x_N^1\}$, a simple GA may be described as follows: at each of iteration $t = 1, \ldots, T$, apply k times Steps 1 to 3 ($k \leq N/2$), then apply Step 4.

Step 1 (Reproduction). Select two parent chromosomes from X^t.

Step 2 (Recombination). Generate two offspring from the two parent chromosomes using a crossover operator.

Step 3 (Mutation). Apply a random mutation to each offspring (with a small probability).

Step 4 (Generation replacement). Create X^{t+1} from X^t by removing the $2k$ worst solutions in X^t and replace them with the $2k$ new offspring.

In this algorithm, parameter T is the number of generations and k the number of selections per generation. The best solution produced over the T generations is the final result of this algorithm. In Step 1, the selection of the parents is probabilistically biased in favour of the best chromosomes. In Step 2, new offspring are produced through crossover by exchanging bit substrings found on the two parents. Each offspring may then be slightly modified in Step 3 by flipping a bit value from zero to one, or from one to zero, with a small probability at each position. Finally, generation replacement takes place in Step 4. Through this process, it is expected that an initial population of randomly generated chromosomes will improve as parents are replaced by better offspring. Some theoretical results support this claim (Holland [38], Goldberg [33]).

5.2 Application to sequencing problems

The classical approach presented in the previous section is not appropriate for sequencing problems, like the TSP or the VRP. For one thing, the bit string representation of a solution is not natural and is typically replaced by a path representation, namely a string of integers where each integer stands for a particular vertex. The position of each integer on the string denotes its ordering on the route (with the last vertex being implicitly connected to the first one). Secondly, specialized order-based crossover and mutation operators must be developed to produce new offspring sequences. For example, Figure 2 illustrates the application of the classical one-point crossover on two parent routes $0, 1, 2, 3, 4, 5$ and $0, 4, 3, 2, 5, 1$, where vertex 0 stands for the depot. Two offspring are created by exchanging the substring located after a randomly selected cutpoint (after the third position, in this example). Clearly, neither of the offspring is a valid sequence, due to duplication and omission of vertices. A straightforward application of the classical mutation operator would also lead to the same kinds of difficulties.

Parent 1	:	0	1	2	3	4	5
Parent 2	:	**0**	**4**	**3**	**2**	**5**	**1**
Offspring 1	:	0	1	2	**2**	**5**	**1**
Offspring 2	:	**0**	**4**	**3**	3	4	5

Figure 2: One-point crossover.

Specialized crossover and mutation operators have thus been proposed in the literature to produce new offspring sequences from parents (Potvin[51]). Figure 3 represents one of them, called the order crossover (OX) (Oliver, Smith and Holland [45]). First, two cut points are randomly selected, and the substring located between these cut points on parent 1 is assigned to the offspring. In the example, the cut points are selected after the third and fifth position, respectively. The remaining positions are then filled one at a time, starting after the second cut point, by considering the vertices in the order found on the second parent (wrapping around, when the end of the string is reached), while avoiding duplications. A second offspring may be created by inverting the role of the parents. Through OX, the offspring tend to inherit the relative order of the vertices on the parent strings. Other operators tend to preserve the position of the vertices (Goldberg and Lingle [34]) or the edges of the parent solutions (Whitley, Starkweather and Fuquay [77]).

Parent 1	:	0	1	2	3	4	5
Parent 2	:	**0**	**4**	**3**	**2**	**5**	**1**
Offspring 1	:	–	–	–	3	4	–
Offspring 1	:	**0**	**2**	**5**	3	4	**1**

Figure 3: Order crossover.

With respect to mutation, simple remove-and-reinsert (RAR) or swap operators have been devised that move one or two vertices to some other position on the string. In Figure 4, an RAR mutation is shown: vertex 2 is randomly selected and moved from position 3 to position 5. Note that a few vertices must be shifted accordingly. Other, more involved, mutation operators have also been devised, like inversion (Holland [38]).

Offspring 1	:	0	1	$\underline{2}$	3	4	5
Offspring 1	:	0	1	3	4	$\underline{2}$	5

Figure 4: RAR mutation.

Previous experience with GAs for solving combinatorial problems has shown that the classical algorithmic framework, with mutation acting as a secondary operator to slightly perturb solutions, does not yield competitive results. Due to its general applicability, this framework does not exploit enough information about the problem to produce high quality solutions. To be effective, the GA must be hybridized with a local search method, either a local descent (Braun [8], Davis [17], Muhlenbein, Gorges-Schleuter and Kramer [44], Suh and Gucht [64], and Ulder *et al.* [71]) or even a tabu search (Fleurent and Ferland [23]), specifically designed for the problem at hand. In this case, the local search method may be viewed as a powerful mutation operator integrated within the GA, and applied with some probability to each offspring. Alternatively, this approach may be viewed as a multi-start local search method, with starting points obtained through a sampling of the search space provided by the GA.

5.3 Application to the VRP

The literature on the development of GAs for solving the VRP is rather scant. This is to be opposed to applications for the TSP (Potvin [51]) or more complex variants of the VRP with time windows and precedence constraints (Blanton and Wainwright [7], Potvin and Bengio [53], Potvin, Duhamel and Guertin [52], Thangiah [68], Thangiah [67] and Thangiah *et al.* [69]). In the first case, the abundance of research is related to the fact that the TSP is a well-known canonical problem, which provides a useful testbed for experimenting with new ideas. In the second case, the presence of complicating constraints, in particular time windows, has until recently hampered the development of effective problem-solving methods. Thus, there was clearly an opportunity for the GA to provide competitive results, given its relative robustness in the presence of complex constraints. In fact, some very effective implementations have been reported in the literature for the vehicle routing problem with time windows (Potvin and Bengio [53], Thangiah [67]). The work done on the capacitated VRP, including its distance or time constrained variant, was mostly aimed at evaluating the impact of different components or parameters of a GA on the efficiency of the search. Van Breedam [73] compares a GA with previously developed SA and TS heuristics on different types of vehicle routing problems, including the capacitated VRP. He also performs a statistical analysis of the impact of various parameters on solution quality for the GA and SA. Given that a solution to a VRP is made of multiple routes (as opposed to the TSP), the path representation is extended and contains multiple copies of the depot, with each copy acting as a separator between two routes. For example, the string shown in Figure 5 would correspond to a VRP solution made of three routes: the first route contains vertices 1 and 2, the second route contains vertices 3 and 4 and the last route only contains vertex 5.

String	:	0	1	2	0	3	4	0	5

Figure 5: Representation of a VRP solution.

A classical order-based crossover operator, known as PMX (Goldberg and Lingle [34]), and a RAR mutation operator are then adapted for this extended representation. At each iteration, these operators are applied until the required number of feasible offspring solutions is produced (i.e., infeasible offspring are discarded). The author also uses a local descent operator based on four different types of exchange moves, and applies it only to the best solution in the current population. Using his own set of 15 test problems with 100 vertices, among which the first six are capacitated VRPs, the author demonstrates that the local descent operator has a significant positive impact on the performance of the GA. Overall, the best solutions produced by the GA, SA and TS (all developed by the author) are of comparable quality. The GA requires more computation time than the other two methods, but the author points out that "no attention has been paid to the efficiency of the code". No comparison is provided with other metaheuristics for the VRP reported in the literature, as the primary goal of this work was to evaluate the impact of different parameters on solution quality. Another GA application for the time-constrained capacitated VRP may be found in Schmitt [61], [62]. An interesting feature of this work is that a route-first, cluster-second approach is used, thus allowing the classical path representation (without separators) to be used. That is, the strings manipulated by the GA correspond to megaroutes over all vertices. A solution to the VRP is then identified through a sweep procedure starting with the vertex in first position on the string. A route ends when either the capacity or the maximum route time constraint would be exceeded by including the next vertex. The latter vertex is then used to initiate a new route. Using this approach, each string can be decoded into a feasible solution to the problem. The implementation proposed in Schmitt [62] uses the OX crossover operator and a swap mutation operator, where two randomly selected vertices exchange their position. To improve the performance of the GA, this mutation operator is then replaced by a 2-opt local search method (Croes [16], Lin [42]), applied with a probability of 0.15 to each offspring. This GA was tested on the Christofides, Mingozzi and Toth [11] test problems with mixed results: the GA proved better than the Clarke and Wright [12] heuristic, but worse than simple construction heuristics combined with improvement procedures. In all cases, the GA was more computationally expensive than the competing methods. In Bean [5], an encoding scheme based on random keys is proposed to address sequencing problems. In this case, each element of the sequencing problem is tagged with a randomly generated key. Decoding these keys into a solution of the problem is typically accomplished through a sorting procedure. For the VRP, each customer is tagged with a random integer which stands for the vehicle that will sevice the customer, plus a real number takes in the interval $(0, 1)$. By sorting these keys, the sequence of customers on each vehicle route is obtained. This application of random keys for the VRP is only provided for illustrative purposes in Bean [5] and is not explored further. Based on these scarce results, it is fair to say that GAs are not yet competitive on the VRP, particularly in view of some recent TS developments. However, almost all research efforts with GAs have focused on the TSP or time window variants of the VRP. The successes obtained on the latter class of problems tend to indicate that more work on the VRP could lead to competitive implementations.

6 Ant algorithms

Ant Systems (AS) methods are inspired from an analogy with real ant colonies foraging for food. In their search for food, ants mark the paths they travel by laying an aromatic essence called pheromone. The quantity of pheromone laid on a path depends on the length of the path and the quality of the food source. This pheromone provides information to other ants that are attracted to it. With time, paths leading to the more interesting food sources, i.e., close to the nest and with large quantities of food, become more frequented and are marked with larger amounts of pheromone. Overall, this process leads to an efficient procedure for procuring food by ant colonies.

This observation led Colorni, Dorigo and Maniezzo [13] to propose a new class of metaheuristics for solving combinatorial problems based on the following correspondences: artificial ants searching the solution space simulate real ants exploring their environment, objective function values are associated with the quality of food sources and values recorded in an adaptive memory mimic the pheromone trails.

To illustrate the basic principles of the approach, we now briefly describe a simple AS for the TSP, the problem to which Colorni, Dorigo and Maniezzo first applied their method. With each (v_i, v_j) are associated two values: the visibility n_{ij} (the inverse of edge length), which is a static value, and the pheromone trail Γ_{ij}, which is updated dynamically as the algorithm proceeds. At each iteration, artificial ants starting from each vertex of the graph construct n new tours using a probabilistic nearest neighbour heuristic with a modified distance measure. This measure is derived from n_{ij} and Γ_{ij} to favour the selection of cities that are close and connected by edges with a high pheromone value. At the end of each iteration, trail values are updated by first allowing a fraction $(1-\rho)$, where $0 \leq \rho \leq 1$, of the old pheromone to evaporate and by then laying new pheromone on the edges of the tours built during this iteration. If edge (v_i, v_j) was used by ant k and the length of the tour constructed by this ant was L_k, the pheromone trail is increased by $\Delta_{ij}^k = 1/L_k$. The trail value for edge (v_i, v_j) is thus update as follows:

$$\Gamma_{ij} := \rho\Gamma_{ij} + \sum_{k=1}^{N} \Delta_{ij}^k,$$

where N is the number of ants. This process of tour construction and trail update is repeated for a fixed number of iterations. It is important to note the role of the evaporation parameter $(1-\rho)$ that prevents poor solutions obtained in early iterations from conditioning the search too strongly at later stages of the algorithm.

The method was refined by the addition of several features on applications such as the symmetric and asymmetric TSP (Dorigo, Maniezzo and Colorni [19]; Dorigo and Gambardella [18]), job shop scheduling (Colorni *et al.* [14]), graph colouring (Costa and Hertz [15]) and quadratic assignment (Gambardella, Taillard and Dorigo [24]). A general conclusion that can be drawn from these papers is that while the method can sometimes produce excellent results, it cannot usually compete with other metaheuristics or specialized local search heuristics, unless it is hybridized one way or another with a local optimizer.

So far, there have been only three papers dealing with the application of AS to the VRP. In the first one, Kawamura *et al.* [40] propose a complex hybrid variant of AS that involves 2-opt improvement procedures and probabilistic acceptance rules reminescent of simulated annealing. The method was applied to two geometric 30- and 60-customer instances and it identified the optimal solution in both cases. No other tests were performed, which makes it difficult to assess the effectiveness of this procedure. The other two papers are due to Bullnheimer, Hartl and Strauss [9], [10]. In their first paper, the authors develop a hybrid AS in which each vehicle route produced in a given iteration is improved by the 2-opt heuristic before trail update. This algorithm also uses terms related to vehicle capacity and distance savings with respect to the depot when selecting the next vertex to be visited. In the trail update step, they use a number of so-called "elitist" ants to account for the best solution found so far (these ants are assumed to always travel on this best solution). Their computational experiments on the 14 problems of Christofides, Mingozzi and Toth [11] indicate that the addition of a 2-opt step and the use of elitist ants are clearly beneficial. The best results obtained over 30 distinct runs range from 0 to 14.09% above the best known solutions to the problems with an average error of 4.43% (see Table 4).

Instance	**Hybrid AS**		**Improved**		**Best known**
	$f^{*(1)}$	**Time** (2)	f^*	**Time** (2)	**solution value**
E051-05e	**524.61**	*0.6*	**524.61**	*0.1*	524.61 (3,4,5,6,8,9,10,11,12,13)
E076-10e	870.8	*2.4*	844.31	*1.3*	835.26 (3,5)
E101-08e	879.43	*11.3*	832.32	*3.8*	826.14 (3,5,6)
E101-10c	819.96	*10.1*	**819.56**	*5.0*	819.56 (3,5,6,8,10,11,13)
E121-07c	1072.45	*16.2*	1065.21	*9.2*	1042.11 (3,4,5,6,8)
E151-12c	1147.41	*28.5*	1061.55	*18.4*	1028.42 (7)
E200-17c	1473.40	*82.2*	1343.46	*87.6*	1291.45 (7)
D051-06c	562.93	*0.2*	560.24	*0.1*	555.43 (3,4,5,6,8,13)
D076-11c	948.16	*3.5*	916.21	*1.7*	909.68 (5,6,8)
D101-09c	886.17	*7.3*	866.74	*4.8*	865.94 (5,6)
D101-11c	869.86	*3.1*	867.07	*5.8*	866.37 (4,5,6,8,13)
D121-11c	1590.52	*4.3*	1559.92	*11.0*	1541.14 (5)
D151-14c	1202.01	*26.6*	1195.99	*27.5*	1162.55 (5)
D200-18c	1504.79	*57.3*	1451.65	*81.8*	1395.85 (7)

(1) Best value over 30 runs.
(2) Minutes on a Pentium 100.
(3) Xu and Kelly [79].
(4) Osman [46].
(5) Taillard [65].
(6) Gendreau, Hertz and Laporte [26].
(7) Rochat and Taillard [59].
(8) Rego and Roucairol [57], parallel implementation.
(9) Bullnheimer, Hartl and Strauss [9].
(10) Bullnheimer, Hartl and Strauss [10].
(11) Optimal Solution (see Golden *et al.* [35]).
(12) Optimal Solution (see Hadjiconstantinou, Christofides and Mingozzi [36]).
(13) Toth and Vigo [70].

Table 4: Computational results for the AS algorithms.

In their second paper, the authors refine their algorithm in several ways: 1) the capacity term previously used in the vertex selection rule, that was quite expensive to compute, is dropped, and the saving term is incorporated directly in the visibility term in a parametric fashion; 2) only the $\lfloor n/4 \rfloor$ nearest neighbours of any vertex are considered when choosing the next customer to visit; 3) only the five best solutions found in each iteration are used for trail update, and the pheromone quantity laid is further weighted according to the solution's rank. These various changes have led to shorter run times and improved solutions. The computational results obtained on the 14 benchmark problems are quite good with an average error of only 1.51% above the best known solutions (see Table 4) and CPU times that are very reasonable.

Overall, these results are quite encouraging considering the very limited experience with the application of AS to the VRP. If one recalls the improvements obtained by later implementations of other metaheuristics, it seems reasonable to expect future AS implementations to be more competitive.

7 Neural Networks

Neural networks are computational models composed of units that are richly interconnected through weighted connections, like neurons in the human brain: a signal is sent from one unit to another along a connection and is modulated through the associated weight. Although superficially related to their biological counterpart, artificial neural networks exhibit characteristics related with human cognition. In particular, they can learn from experience and induce general concepts from specific examples through an incremental adjustment of their weights. These models were originally designed for tasks associated with human intelligence and where traditional computation has proven inadequate, like artificial vision and speech understanding. More recently, they have been applied to combinatorial problems as well, starting with the pioneering work of Hopfield and Tank [39]. The TSP, in particular, has been the subject of many investigations with the Hopfield-Tank model, the elastic net (EN) (Durbin and Willshaw [22]) and the self-organizing map (SOM) (Kohonen [41]). The EN and SOM models are quite remote from classical neural networks, but they have proven to be more effective on the TSP than the (more classical) Hopfield-Tank model. However, neither of these methods is yet competitive with other metaheuristics (Potvin [50]).

The elastic net and self-organizing maps are deformable templates that adjust themselves to the contour of the vertices to solve a TSP, as illustrated in Figure 6. The white circles are vertices and the small black circles are units of the model. These units are linked to form a ring. Starting from some arbitrary configuration, the location of each unit on the ring is incrementally adjusted (like the connection weights of classical neural network models), until at least one unit becomes sufficiently close to each vertex. At the end, each vertex is assigned to the closest unit. Through this assignment, the ordering of the units along the ring determines an ordering of the vertices on the TSP tour. EN and SOM work similarly, but differ in the mechanisms used to control the migration of units toward the vertices.

Deformable templates can be easily applied to pure geometric problems, like Euclidean TSPs. However, they are not designed to handle additional constraints, like capacity and maximum route time constraints, which often break the geometric nature of the

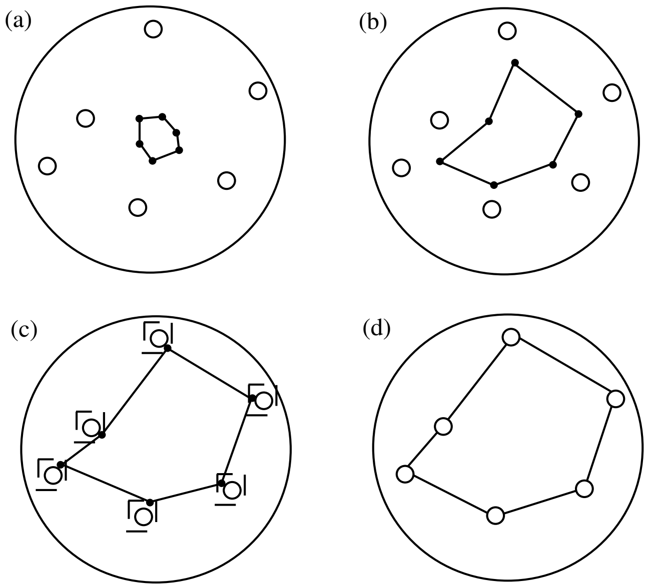

Figure 6: Evolution of a deformable template in (a), (b), (c) and the final solution (d).

problem. Only a few recent efforts have been devoted to the VRP, mostly based on variants of the SOM (Ghaziri [28], Ghaziri [29], Ghaziri [30], Matsuyama [43], and Schumann and Retzko [63]). A generalization of these models from the TSP to the VRP is obtained by using many deformable templates, one for each route. Typically, the models are applied with an increasing number of rings (routes) until a feasible solution is identified. Given that multiple rings are now present, a competition takes place among the rings to get an equal share of vertices. The procedure suggested in (Ghaziri [29]) for the capacitated VRP may be summarized as follows:

Step 1. (Ring competition and migration). Repeat until there is a unit sufficiently close to each vertex:

- **1.1** consider the next vertex (wrap around when the last vertex has been done) and set it as the current vertex;
- **1.2** associate a selection probability with each ring;
- **1.3** select a ring according to the probability distribution defined in Step 1.2;
- **1.4** tentatively assign the current vertex to the closest unit on the selected ring and move this unit (as well as some of its neighbours on the ring) towards the current vertex.

Step 2. (Vertex assignment). Permanently assign each vertex to the closest unit to produce a solution.

The probability associated with each ring is dynamically adjusted as the algorithm unfolds. At the start, the distance between the current vertex and the closest unit on the ring plays the dominant role. Later on, the capacity constraint is taken into account, as rings that cannot accommodate the current vertex without violating this constraint (due to the tentatively assigned vertices) becomes less and less likely to be selected. At the end, only feasible rings have a non negligible probability of being selected. In a later study (Ghaziri [30]) extended this model to address the VRP with maximum route time constraints through a modification of the probability distribution over the rings. Computational results on the Christofides, Mingozzi and Toth [11] test set have shown that these models produce solutions of relatively good quality, but are not competitive with alternative metaheuristics, in particular TS (Gendreau, Hertz and Laporte [26], and Rochat and Taillard [59]).

8 Conclusion

This survey of metaheuristics for the VRP shows that the best of these methods are able to find excellent and sometimes optimal solutions to instances with a few hundreds customers, albeit at a significant cost in computation time. Tabu Search now emerges as the most effective approach. Procedures based on pure Genetic Algorithms and on Neural Networks are clearly outperformed, while those based on Simulated or Deterministic Annealing and on Ant Systems are not quite competitive. Considering the performance improvements obtained with successive implementations of any given approach, it appears, however, that hybrid AS and GA may, in the future, be capable of matching the effectiveness of existing TS heuristics, since these approaches have not been fully exploited. Another observation is that the data sets currently used as benchmarks are made up of instances that are too small to allow to differentiate sharply between the various implementations of some of the metaheuristics, TS in particular. Data sets corresponding to larger instances are thus required. In the same vein, one may wonder how these metaheuristics would perform on the much larger instances often encountered in practical applications. Given their computing requirements, heuristics with such a level of sophistication may be unable to solve satisfactorily these large instances in any reasonable amount of time, especially if real-time applications are contemplated. With respect to the classical heuristics presented in Chapter 5, metaheuristics are rather time consuming but they also provide much better solutions. Typically, classical methods yield solution values between 2% and 10% above the optimum (or the best known solution value), while the corresponding figure for the best metaheuristic implementation is often less than 0.5%. This is illustrated in Figure 7. Time has now come to develop simpler methods capable of quickly providing good quality solutions. This will probably be achieved by speeding up the best available metaheuristics as opposed to investing more effort on classical approaches. The GTS algorithm recently proposed by Toth and Vigo is an important step in the right direction. It draws from the vast expertise accumulated in the field of metaheuristics and exploits some of their best concepts. Yet, by carefully exploiting the problem structure, it manages to avoid most of the unnecessary computations carried out in previous TS algorithms.

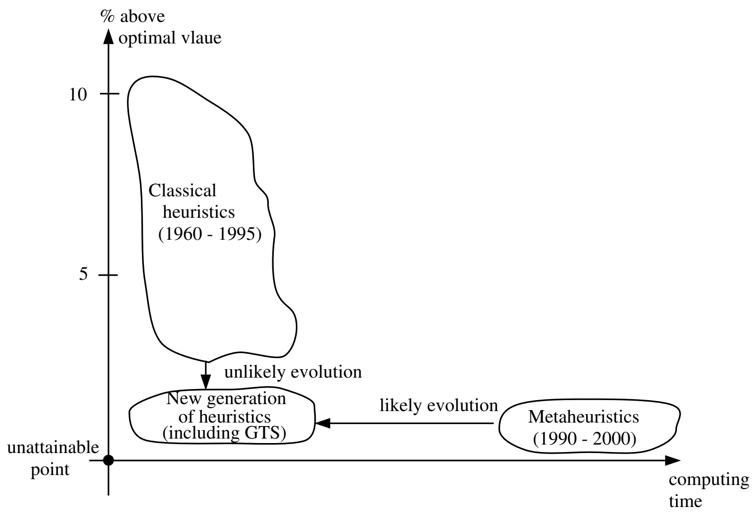

Figure 7: Evolution of heuristics for the vehicle routing problem.

References

[1] E.H.L. Aarts and J.K. Lenstra. *Local Search in Combinatorial Optimization.* Wiley, Chichester, 1997.

[2] A.S. Alfa, S.S. Heragu, and M. Chen. A 3-opt based simulated annealing algorithm for vehicle routing problems. *Computers & Industrial Engineering*, 21:635–639, 1991.

[3] G. Barbarosoglu and D. Ozgur. A tabu search algorithm for the vehicle routing problem. *Computers & Operations Research*, 26:255–270, 1999.

[4] R.S. Barr, B.L. Golden, J.P. Kelly, M.G.C. Resende, and W.R. Stewart Jr. Designing and reporting on computational experiments with heuristic methods. *Journal of Heuristics*, 1:9–32, 1995.

[5] J.C. Bean. Genetic algorithms and random keys for sequencing and optimization. *ORSA Journal on Computing*, 6:154–160, 1994.

[6] J.E. Beasley. Route-first cluster-second methods for vehicle routing. *Omega*, 11:403–408, 1983.

[7] J.L. Blanton and R.L. Wainwright. Multiple vehicle routing with time and capacity constraints using genetic algorithms. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 452–459. Morgan Kaufmann, San Mateo, CA, 1993.

[8] H. Braun. On solving travelling salesman problems by genetic algorithms. In H.P. Schwefel and R. Manner, editors, *Parallel Problem-Solving from Nature*, pages 129–133. Springer-Verlag, Berlin, 1991.

[9] B. Bullnheimer, R.F. Hartl, and C. Strauss. Applying the ant system to the vehicle routing problem. In S. Voß, S. Martello, I.H. Osman, and C. Roucairol, editors, *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*, pages 109–120. Kluwer, Boston, 1998a.

[10] B. Bullnheimer, R.F. Hartl, and C. Strauss. An improved ant system for the vehicle routing problem. *Annals of Operations Research*, 1998b. forthcoming.

[11] N. Christofides, A. Mingozzi, and P. Toth. The vehicle routing problem. In N. Christofides, A. Mingozzi, P. Toth, and C. Sandi, editors, *Combinatorial Optimization*, pages 315–338. Wiley, Chichester, 1979.

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

[13] A. Colorni, M. Dorigo, and V. Maniezzo. Distributed optimization by ant colonies. In F. Varela and P. Bourgine, editors, *Proceedings of the European Conference on Artificial Life*. Elsevier, Amsterdam, 1991.

[14] A. Colorni, M. Dorigo, V. Maniezzo, and M. Trubian. Ant system for job-shop scheduling. *Belgian Journal of Operations Research, Statistics and Computer Science*, 34:39–53, 1994.

[15] D. Costa and A. Hertz. Ants can colour graphs. *Journal of the Operational Research Society*, 48:275–305, 1997.

[16] G. Croes. A method for solving traveling salesman problems. *Operations Research*, 6:791–812, 1958.

[17] L. Davis. *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, New York, 1991.

[18] M. Dorigo and L.M. Gambardella. Ant colony system: A cooperative learning approach for the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1:53–66, 1997.

[19] M. Dorigo, V. Maniezzo, and A. Colorni. Ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man and Cybernetics — Part B*, 26:29–41, 1996.

[20] G. Dueck. New optimization heuristics: The great deluge algorithm and the record-to-record travel. *Journal of Computational Physics*, 104:86–92, 1993.

[21] G. Dueck and T. Scheurer. Threshold accepting: A general purpose optimization algorithm. *Journal of Computational Physics*, 90:161–175, 1990.

[22] R. Durbin and D. Willshaw. An analogue approach to the travelling salesman problem using an elastic net method. *Nature*, 326:689–691, 1987.

[23] C. Fleurent and J.A. Ferland. Genetic and hybrid algorithms for graph colouring. In G. Laporte and I.H. Osman, editors, *Metaheuristics in Combinatorial Optimization*, volume 63 of *Annals of Operations Research*, pages 437–461, 1996.

[24] L.M. Gambardella, É.D. Taillard, and M. Dorigo. Ant colonies for the Quadratic Assignment Problem. Technical Report IDSIA/4–97, IDSIA, Lugano, Switzerland, 1997.

[25] M. Gendreau, A. Hertz, and G. Laporte. New insertion and postoptimization procedures for the traveling salesman problem. *Operations Research*, 40:1086–1094, 1992.

[26] M. Gendreau, A. Hertz, and G. Laporte. A tabu search heuristic for the vehicle routing problem. *Management Science*, 40:1276–1290, 1994.

[27] M. Gendreau, G. Laporte, and J.-Y. Potvin. Vehicle routing: Modern heuristics. In E.H.L. Aarts and J.K. Lenstra, editors, *Local Search in Combinatorial Optimization*, pages 311–336. Wiley, Chichester, 1997.

[28] H. Ghaziri. Solving routing problems by a self-organizing map. In T. Kohonen, K. Makisara, O. Simula, and J. Kangas, editors, *Artificial Neural Networks*, pages 829–834, North-Holland, Amsterdam, 1991.

[29] H. Ghaziri. *Algorithmes connexionnistes pour l'optimisation combinatoire.* Thèse de doctorat, École Polytechnique Fédérale de Lausanne, Switzerland, 1993.

[30] H. Ghaziri. Supervision in the self-organizing feature map: Application to the vehicle routing problem. In I.H. Osman and J.P. Kelly, editors, *Meta-Heuristics: Theory and Applications*, pages 651–660. Kluwer, Boston, 1996.

[31] F. Glover and M. Laguna. Tabu search. In C.R. Reeves, editor, *Modern Heuristic Techniques for Combinatorial Problems*, pages 70–150. Blackwell, Oxford, 1993.

[32] F. Glover and M. Laguna. *Tabu Search.* Kluwer, Boston, 1997.

[33] D.E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley, Reading, MA, 1989.

[34] D.E. Goldberg and R. Lingle. Alleles, loci and the traveling salesman problem. In J.J. Grefenstette, editor, *Proceedings of the First International Conference on Genetic Algorithms*, pages 154–159. Lawrence Erlbaum, Hillsdale, NJ, 1985.

[35] B.L. Golden, E.A. Wasil, J.P. Kelly, and I-M. Chao. Metaheuristics in vehicle routing. In T.G. Crainic and G. Laporte, editors, *Fleet Management and Logistics*, pages 33–56. Kluwer, Boston, 1998.

[36] E. Hadjiconstantinou, N. Christofides, and A. Mingozzi. A new exact algorithm for the vehicle routing problem based on q-paths and k-shortest paths relaxations. In M. Gendreau and G. Laporte, editors, *Freight Transportation*, volume 61 of *Annals of Operations Research*, pages 21–43, 1995.

[37] A. Hertz, É.D. Taillard, and D. de Werra. Tabu search. In E.H.L. Aarts and J.K. Lenstra, editors, *Local Search in Combinatorial Optimization*, pages 121–136. Wiley, Chichester, 1997.

[38] J.H. Holland. *Adaptation in Natural and Artificial Systems.* The University of Michigan Press, Ann Arbor, MI, 1975.

[39] J.J. Hopfield and D.W. Tank. Neural computation of decisions in optimization problems. *Biological Cybernetics*, 52:141–152, 1985.

[40] H. Kawamura, M. Yamamoto, T. Mitamura, K. Suzuki, and A. Ohuchi. Cooperative search on pheromone communication for vehicle routing problems. *IEEE Transactions on Fundamentals*, E81–A:1089–1096, 1998.

[41] T. Kohonen. *Self-Organization and Associative Memory.* Springer, Berlin, 1988.

[42] S. Lin. Computer solutions of the traveling salesman problem. *Bell System Technical Journal*, 44:2245–2269, 1965.

[43] Y. Matsuyama. Self-organization via competition, cooperation and categorization applied to extended vehicle routing problems. In *Proceedings of the International Joint Conference on Neural Networks*, pages I–385–390, Seattle, WA, 1991.

[44] H. Mulhenbein, M. Gorges-Schleuter, and O. Krämer. Evolution algorithms in combinatorial optimization. *Parallel Computing*, 7:65–85, 1988.

[45] I.M. Oliver, D.J. Smith, and J.R.C. Holland. A study of permutation crossover operators on the traveling salesman problem. In J.J. Grefenstette, editor, *Proceedings of the Second International Conference on Genetic Algorithms*, pages 224–230. Lawrence Erlbaum, Hillsdale, NJ, 1987.

[46] I.H. Osman. Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. *Annals of Operations Research*, 41:421–451, 1993.

[47] I.H. Osman and J.P. Kelly. Meta-heuristics: An overview. In I.H. Osman and J.P. Kelly, editors, *Meta-Heuristics: Theory and Applications*, pages 1–21. Kluwer, Boston, 1996.

[48] I.H. Osman and G. Laporte. Metaheuristics: A bibliography. In G. Laporte and I.H. Osman, editors, *Metaheuristics in Combinatorial Optimization*, volume 63 of *Annals of Operations Research*, pages 513–628, 1996.

[49] M. Pirlot. General local search heuristics in combinatorial optimization: A tutorial. *Belgian Journal of Operations Research, Statistics and Computer Science*, 32:8–67, 1992.

[50] J.-Y. Potvin. The traveling salesman problem: A neural network perspective. *ORSA Journal on Computing*, 5:328–348, 1993.

[51] J.-Y. Potvin. Genetic algorithms for the traveling salesman problem. *Annals of Operations Research*, 63:339–370, 1996.

[52] J.-Y. Potvin, C. Duhamel, and F.Guertin. A genetic algorithm for vehicle routing with backhauling. *Applied Intelligence*, 6:345–355, 1996.

[53] J.-Y. Potvin and S.Bengio. The vehicle routing problem with time windows — Part II: Genetic search. *INFORMS Journal on Computing*, 8:165–172, 1996.

[54] V.M. Pureza and P.M. França. Vehicle routing problems via tabu search metaheuristic. Technical Report CRT–347, Centre for research on transportation, Montreal, 1991.

[55] C.R. Reeves, editor. *Modern Heuristic Techniques for Combinatorial Problems.* Blackwell, Oxford, 1993.

[56] C. Rego. A subpath ejection method for the vehicle routing problem. *Management Science*, 44:1447–1459, 1998.

[57] C. Rego and C. Roucairol. A parallel tabu search algorithm using ejection chains for the vehicle routing problem. In I.H. Osman and J.P. Kelly, editors, *Meta-Heuristics: Theory and Applications*, pages 661–675. Kluwer, Boston, 1996.

[58] F. Robusté, C.F. Daganzo, and R. Souleyrette II. Implementing vehicle routing models. *Transportation Research*, 24B:263–286, 1990.

[59] Y. Rochat and É.D. Taillard. Probabilistic diversification and intensification in local search for vehicle routing. *Journal of Heuristics*, 1:147–167, 1995.

[60] D.E. Rumelhart and J.L. McClelland, editors. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition.* MIT Press, Cambridge, MA, 1986.

[61] L.J. Schmitt. *An empirical computational study of genetic algorithms to solve order based problems: An emphasis on TSP and VRPTC.* Ph.D. Dissertation, Fogelman College of Business and Economics, University of Memphis, 1994.

[62] L.J. Schmitt. An evaluation of a genetic algorithmic approach to the vehicle routing problem. Working paper, Department of Information Technology Management, Christian Brothers University, Memphis, 1995.

[63] M. Schumann and R. Retzko. Self-organizing maps for vehicle routing problems — minimizing an explicit cost function. In F. Fogelman-Soulie, editor, *Proceedings of the International Conference on Artificial Neural Networks*, pages II–401–406, Paris, 1995. EC2&Cie.

[64] J.Y. Suh and D.V. Gucht. Incorporating heuristic information into genetic search. In J.J. Grefenstette, editor, *Proceedings of the Second International Conference on Genetic Algorithms*, pages 100–107. Lawrence Erlbaum, Hillsdale, NJ, 1987.

[65] É.D. Taillard. Parallel iterative search methods for vehicle routing problems. *Networks*, 23:661–673, 1993.

[66] É.D. Taillard, L.M. Gambardella, M. Gendreau, and J.-Y. Potvin. Adaptive memory programming: A unified view of metaheuristics. Research Report IDSIA/19–98, IDSIA, Lugano, Switzerland, 1998.

[67] S.R. Thangiah. Vehicle routing with time windows using genetic algorithms. Technical report SRU-CpSc-TR-93-23, Slippery Rock University, Slippery Rock, PA, 1993.

[68] S.R. Thangiah. An adaptive clustering method using a geometric shape for vehicle routing problems with time windows. In L.J. Eshelman, editor, *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 536–543. Morgan Kaufmann, San Mateo, CA, 1995.

[69] S.R. Thangiah, I.H. Osman, R. Vinayagamoorthy, and T. Sun. Algorithms for the vehicle routing problem with time deadlines. *American Journal of Mathematical and Management Sciences*, 13:323–355, 1993.

[70] P. Toth and D. Vigo. The granular tabu search (and its application to the vehicle routing problem). Working paper, DEIS, University of Bologna, 1998.

[71] N.L.J. Ulder, E.H.L. Aarts, H.J. Bandelt, P.J.M. van Laarhoven, and E. Pesch. Genetic local search algorithms for the traveling salesman problem. In H.P. Schwefel and R. Manner, editors, *Parallel Problem-Solving from Nature*, pages 109–116. Springer-Verlag, Berlin, 1991.

[72] A. Van Breedam. Improvement heuristics for the vehicle routing problem based on simulated annealing. *European Journal of Operational Research*, 86:480–490, 1995.

[73] A. Van Breedam. An analysis of the effect of local improvement operators in genetic algorithms and simulated annealing for the vehicle routing problem. RUCA Working Paper 96/14, University of Antwerp, Belgium, 1996.

[74] P.J.M. van Laarhoven and E.H.L. Aarts. *Simulated Annealing: Theory and Applications*. Reidel, Dordrecht, 1987.

[75] A. Volgenant and R. Jonker. The symmetric traveling salesman problem and edge exchange in minimal 1-trees. *European Journal of Operational Research*, 12:394–403, 1983.

[76] P.D. Wasserman. *Neural Computing – Theory and Practice*. Van Nostrand Reinhold, New York, 1989.

[77] D. Whitley, T. Starkweather, and D. Fuquay. Scheduling problems and traveling salesmen: The genetic edge recombination operator. In J.D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 133–140. Morgan Kaufmann, San Mateo, CA, 1989.

[78] J.A.G. Willard. Vehicle Routing Using r-Optimal Tabu Search. M.Sc. Dissertation, The Management School, Imperial College, London, 1989.

[79] J. Xu and J.P. Kelly. A network flow-based tabu search heuristic for the vehicle routing problem. *Transportation Science*, 30:379–393, 1996.