Soft Comput (2008) 12:1125–1141
DOI 10.1007/s00500-008-0281-4

ORIGINAL PAPER

The memetic self-organizing map approach to the vehicle routing problem

Jean-Charles Créput · Abderrafiaâ Koukam

Published online: 13 February 2008

Abstract The paper presents an extension of the self-organizing map (SOM) by embedding it into an evolutionary algorithm to solve the Vehicle Routing Problem (VRP). We call it the memetic SOM. The approach is based on the standard SOM algorithm used as a main operator in a population based search. This operator is combined with other derived operators specifically dedicated for greedy insertion moves, a fitness evaluation and a selection operator. The main operators have a similar structure based on the closest point findings and local moves performed in the plane. They can be interpreted as performing parallels and massive insertions, simulating the behavior of agents which interact continuously, having localized and limited abilities. This self-organizing process is intended to allow adaptation to noisy data as well as to confer robustness according to demand fluctuation. Selection is intended to guide the population based search toward useful solution compromises. We show that the approach performs better, with respect to solution quality and/or computation time, than other neural network applications to the VRP presented in the literature. As well, it substantially reduces the gap to classical Operations Research heuristics, specifically on the large VRP instances with time duration constraint.

Keywords Neural network · Self-organizing map · Evolutionary algorithm · Vehicle routing problem

J.-C. Créput (✉) · A. Koukam
Systems and Transportation Laboratory,
University of Technology of Belfort-Montbeliard,
90010 Belfort, France
e-mail: Jean-Charles.Creput@utbm.fr

A. Koukam
e-mail: Abder.Koukam@utbm.fr

1 Introduction

In this paper we are concerned with the Vehicle Routing Problem (VRP) (Christofides et al. 1979). The VRP is defined on a set $V = \{v_0, v_1, \ldots, v_N\}$ of vertices, where vertex v_0 is a depot at which are based m identical vehicles of capacity Q, while the remaining N vertices represent customers, also called requests or demands. A non-negative cost, or travel time, is defined for each edge $(v_i, v_j) \in V \times V$. 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 of the least total cost, each starting and ending at the depot, such that each customer is visited exactly once by a vehicle, the total demand of any route does not exceed Q, and the total duration of any route does not exceed a preset bound D. As it is the most often done in practice (Cordeau et al. 2005; Laporte et al. 2000), we shall be concerned in this paper with the Euclidean VRP, where each vertex v_i has a location in the plane, and where the travel cost is given by the Euclidean distance $d(v_i, v_j)$ for each edge $(v_i, v_j) \in V \times V$. Then, the main objective of the problem is the total route length.

This problem is one of the most widely studied problems in combinatorial optimization. It has a central place for the determination of efficient routes in distribution management. The problem is NP-hard. Exact methods can only solve low size problems with up to 50 customers efficiently (Toth and Vigo 2001). Then, for large VRP instances, using heuristics is encouraged in that they have statistical or empirical guaranty to find good solutions for possible large scale problems with several hundreds of customers.

Since more than four decades, successive generations of construction heuristics, improvement heuristics, and metaheuristics were developed by the Operations Research (OR) community, to solve the VRP. Construction heuristics, as

the well known (Clarke and Wright 1964) savings heuristic, generate admissible solutions very quickly. Improvement heuristics perform local operations starting from a first constructed initial solution. They are generally based on customer movements and arc exchanges (Or 1976). For more than a decade, researchers have turned to develop general strategies, called metaheuristics, embedding problem oriented local search operators. They generally encapsulate a construction method followed by the application of one or more improvements heuristics (local search). From stage to stage, such heuristics were enriched reusing the past enhancements to build new sophisticated neighborhood search structures.

For example, some of the most powerful metaheuristics for the VRP, referred in the extensive surveys (Cordeau et al. 2005; Laporte et al. 2000; Gendreau et al. 2002; Golden et al. 1998), are the Tabu Search (Cordeau et al. 2001; Rochat and Taillard 1995; Toth and Vigo 2003), simulated annealing, and population based methods, such as evolutionary algorithms (Mester and Bräysy 2005; Prins 2004), adaptive memory (Taillard et al. 2001) and ant algorithms (Reimann et al. 2004). Other methods can hybridize several metaheuristics principles and/or previous heuristics into a common framework. Such examples are the very powerful active guided local search (Mester and Bräysy 2005), which is maybe the overall winner approach considering both quality solution and computation time, or the very recent variable neighborhood search (Kytöjoki et al. 2007), which can address very large instances and which performs very quickly. It is worth noting that during the past decade most of the approaches only solve large instances with up to 500 customers, using the so-called Golden et al. (1998) benchmark problems. Very large instances have been proposed recently by Li et al. (2005), with 560–1200 customers, and at the date of finishing this paper by Kytöjoki et al. (2007) with up to 20000 customers.

Often Operations Research heuristics for the classical VRP operate on graphs and consider a $O(N^2)$ matrix to memorize arc costs. This makes difficult the application to very large instances on standard 32-bit workstations with 4 GB memory. That's why heuristics tend more or less to exploit implicitly the Euclidean triangle inequality (Kytöjoki et al. 2007; Laporte et al. 2000). Almost all benchmark problems for the VRP are Euclidean problems (Cordeau et al. 2005). As stated in Johnson and McGeoch (1997, 2002), 2-Opt or k-Opt heuristics work well on Euclidean problems, whereas the performances rapidly decrease on general graph instances. As well, the good performances on usual Euclidean benchmarks corroborate the fact that geometric problems generally have approximation algorithms (Arora 1998).

Hence, we focus on the Euclidean VRP and propose a Euclidean solving approach. The method presented in this paper takes its origin in neural networks (NN) visual patterns that evolve and distort in the plane according to an underlying data distribution. The neural network considered in this paper is the self-organizing map (SOM) (Kohonen 2001), which is often presented as a non-supervised learning procedure performing a non-parametric regression that reflects topological information of the input data. It can also be seen as a center based clustering algorithm with topologic relationships between cluster centers. The underlying concept, that we call adaptive meshing, lets envisage the application to many spatially distributed problems, as radio-mobile and terrestrial transportation dimensioning, clustering k-median, and combined center-based clustering and routing problems (Créput et al. 2005; Créput and Koukam 2006, 2007a,b; Créput et al. 2000, 2007). For example, Créput and Koukam (2007a,b) deal with an extension and combination of the Euclidean k-median problem and classical VRP. It consists of positioning bus stops, called cluster centers, according to customer locations (k-median problem) and generating vehicle routes among bus-stops (VRP). Bus-stops define clusters where customers are grouped and where they have to walk to take the bus. Furthermore, Créput et al. (2007) is an extension of the VRP with time windows (Solomon 1987) considering some walking distances from customers to bus stops.

Separating the transportation network from the underlying customer demands, rather than modeling routes by an ordering of customers as in classical heuristics, has several positive aspects. By preserving the density and the topology of a data distribution, SOM allows the positioning of facilities in accordance to the demand, respecting the inter-component network architecture. Consequently, the approach has the potential to deal with noisy or incomplete data, and with fluctuating demand. Also, by choosing to dynamically compute Euclidean distances, the spatial complexity of the approach is maintained in $O(N)$, thus allowing application to very large instances. Furthermore, continuous visual feedback during simulations is naturally allowed.

Here, we are interested in solving the VRP by the means of the adaptive meshing concept. To solve the combinatorial optimization problem, we chose to combine self-organization principles in evolution following hybridization of meta-heuristics, as in Gambardella et al. (1999), Moscato (1999), Moscato and Cotta (2003) and Boese et al. (1994). We provide an evolutionary framework which incorporates SOMs as internal operators. The approach is called memetic SOM by reference to memetic algorithms (Moscato 1999; Moscato and Cotta 2003; Buriol et al. 2004; Merz and Freisleben 2001, 2002), which are hybrid evolutionary algorithms (EA) incorporating a local search process. The self-organizing distributed process is intended to facilitate massively parallel insertions in the plane and, theoretically, to confer robustness according to demand fluctuations and network injuries. On another hand, evolution is intended to efficiently guide the population based search toward the useful solu-

tion compromises. This massive and natural parallelism at different levels differentiates the approach from classical Operations Research heuristics which operate on graphs (sequentially) and are often considered complex and difficult to implement. Furthermore, since the communication times at the level of selection are relatively small, the long running times of independent SOM processes favor parallel execution of the method.

In the literature, many applications of neural networks have addressed the traveling salesman problem (TSP). For more information on this subject, we refer the reader to the extensive survey of Cochrane and Beasley (2003). However, extending SOM to the more complex VRP remains a difficult task. Few works were carried out trying to extend SOM, or elastic nets, to the VRP. As far as we know, the most recent approaches are Ghaziri (1996), Gomes and Von Zuben (2002), Matsuyama (1991), Modares et al. (1999), Schumann and Retzko (1995), Schwardt and Dethloff (2005) and Vakhutinsky and Golden (1994). They are generally based on a complex modification of the internal learning law, altered by problem dependant penalties. Here, to apply SOM to the VRP and to improve performances as well, the SOM execution interleaves with other processes, or operators, following the evolutionary method. The standard SOM is a main operator embedded into an EA and combined with other greedy insertion operators, fitness evaluation and selection operators. Then, operators can be designed independently and then combined.

Evaluation of the proposed approach is performed against neural networks and Operations Research heuristics. In the former case, we compare the memetic SOM to the three representative approaches of Ghaziri (1996), Modares et al. (1999) and Schwardt and Dethloff (2005). Only these authors have made a significant use of the publicly available (Christofides et al. 1979) test problems. Other approaches typically used just a few and specific test problems and are hard to evaluate. We will mainly try to show that the memetic SOM yields a substantial gain of accuracy in comparison to the previous SOM based approaches. Considering Operations Research heuristics, we use the Golden et al. (1998) large test problems and compare the approach to the Active Guided Evolution Strategy (AGES) (Mester and Bräysy 2005), Granular Tabu Search (GTS) (Toth and Vigo 2003), Unified Tabu Search Algorithm (UTSA) (Cordeau et al. 2001) and Very Large Neighborhood Search (VLNS) (Ergun et al. 2003), which from our point of view cover the range of heuristic performances. Memetic SOM does certainly not compete with the most powerful heuristics, which benefit from the considerable effort spent over more than thirty years, but we claim that it substantially reduces the gap.

The paper is organized as follows. Section 2 introduces the principle of the method, and Sect. 3 details the memetic SOM approach. Section 4 reports experimental analysis of the algorithm. Section 5 presents evaluations against SOM based approaches and classical heuristics. Finally, the last section is devoted to the conclusion and further research.

2 Method principle

Despite the lack of competitiveness of NN when compared to OR heuristics, and according to Cochrane and Beasley (2003), we believe that the reasons for continuing to study NN for VRP, specifically the SOM, are the followings:

- The incremental development of NN may lead, over time, to more competitive approaches for the VRP.
- SOM may become competitive in the future when massively parallel computer systems will become widely available.
- The insights gained from studying SOM for VRP may help to develop NN approaches for other problems, where heuristics are not as well developed as are the OR heuristics specifically dedicated to the VRP.
- Unlike Hopfield NN models (Smith 1996), which perform poorly on large size problems, SOM can be applied to very large problems.

To illustrate the "philosophy" of the SOM behavior, an example of a TSP tour construction, using a ring network, is illustrated in Fig. 1 on the bier127 instance from TSPLIB (Reinelt 1991) at different steps of a long simulation run. The network dispatches its vertices among customers in a massively parallel manner. At the beginning, the local moves are performed with a great intensity in order to let the ring deploys toward cities starting from scratch (a). Then, the intensity of moves slightly decreases in order to progressively freeze the vertices near cities (b–c). At a final step, customers have just to be assigned to their nearest vertex in the ring in order to generate a final tour ordering.

The aim of the memetic SOM approach is to follow the two metaphors of self-organization and evolution within a single metaheuristic framework. Thus, to extend SOM applicability to more complex problems, such as the VRP, it becomes an operator embedded into an evolutionary algorithm. The SOM is now a long run process applied to a population of solutions. This process is interrupted at each cycle, called a generation, by the application of several evolutionary operators. Interruption of SOM occurs in such a way that approximately only $O(N)$ basic iterations are performed at each generation, for a total of $O(N)$ generations made, N being the problem size. The structure of the algorithm is similar to the memetic algorithm, which is an evolutionary algorithm incorporating a local search (Moscato 1999; Moscato and Cotta 2003; Merz and Freisleben 2001; Krasnogor 2005).

Fig. 1 A TSP tour construction by SOM using the bier127 instance

The main components of the method which are intended for driving the search are:

- a Self-Organizing Map as a low level stochastic search,
- the problem-oriented operators interleaving with SOM,
- random perturbations with a decreasing intensity,
- a fitness evaluation,
- a population based strategy with selection operators,
- the use of intermediate non-admissible solutions,
- a search performed within two phases (construction, improvement).

Here, no recombination, nor crossover, operator is considered. The method can be viewed as a top down approach like the evolution strategies and evolutionary programming (Bäck et al. 1991). In that cases, the recombination operator often plays a secondary role, and can be seen as a large step mutation to help to exit from local minima. On the contrary, the genetic algorithms and genetic programming (Goldberg 1994) are sometimes presented as bottom up construction methods, centered on a crossover, and based on the building blocks hypothesis. The role of the recombination operator is the matter of debates since a long time. However, it is admitted that its effectiveness highly depends on the problem under consideration. Here, recombination is discarded in a first attempt. Another difference of the memetic SOM with the memetic algorithm is that local search runs are not independent runs performed at each cycle to reach local minima, as usual. The SOM is a (long) stochastic gradient descent performed during the many generations allowed. The term "local search" is used by analogy. Also, the memetic SOM is based on a physical metaphor of self-organization in a space. Whereas, the memetic algorithm is based on the metaphor of learning. In Krasnogor (2005), the author presents different applications of the memetic algorithm to either combinatorial optimization problems or to numerical optimization problems. Our approach encompasses combinatorial and spatial aspects as well. It hybridizes a spatial heuristic to solve a Euclidean version of a combinatorial optimization problem.

Different approaches actually focus on the design of "adaptive" memetic algorithms. In these cases, the mechanism generally deals with the adaptive choice of local search operators. Examples are the meta-lamarkian methods, hyperheuristics, or multi-memes approaches (Ong et al. 2006). Such approaches operate at a meta level, independent of the problem. The memetic SOM however put the emphasis on the heuristic level, exploiting the adaptive nature of the localized interactions performed at the level of the micro-components of the problem. The aim is to provide a generic class of metaheuristics suited for a wide range of spatially distributed problems. We applied the concept, called adaptive meshing, to mobile communication systems (Créput et al. 2005; Créput and Koukam 2006). By using a honeycomb adaptive mesh, the approach was able to deal with the positioning of antenna and traffic covering while encompassing frequency allocation constraints through the preservation of an hexagonal topology. The optimization approach is a standard memetic algorithm based on a specific local search operator. Because of its many applications in learning, classification, and pattern recognition, and because of its properties of self-organization, the SOM appeared to be a good candidate for implementing the adaptive meshing concept into the context of terrestrial transportation. We already applied the approach to non-standard problems, combining clustering and vehicle routing in a unified way (Créput and Koukam 2007a,b; Créput et al. 2007).

An evaluation of the performances of a memetic algorithm can be done by carefully measuring the improvement carried out with regard to the embedded local search heuristic. Designing a memetic algorithm which clearly dominates its embedded heuristic, on both solution quality and computation time simultaneously, seems not an easy task to do. How important is the improvement carried out by a memetic approach, allowing lesser computation time than the one usually allowed for the embedded heuristic itself. That is an important question. Memetic algorithms often encapsulate restarts of local search executions and then, are generally time consuming, specifically when dealing with large size instances. This appears clearly in the applications to the TSP (Merz and Freisleben 2002, 2001). Very few memetic approaches really dominate the standard Lin and Kernighan (LK) heuristic, specifically its recent implementation due to Helsgaun (HLK) (Johnson and McGeoch 1997, 2002), on both solution quality and computation time. From our knowing, only the very recent hybrid genetic algorithm of Nguyen et al. (2007) clearly outperforms the HLK heuristic.

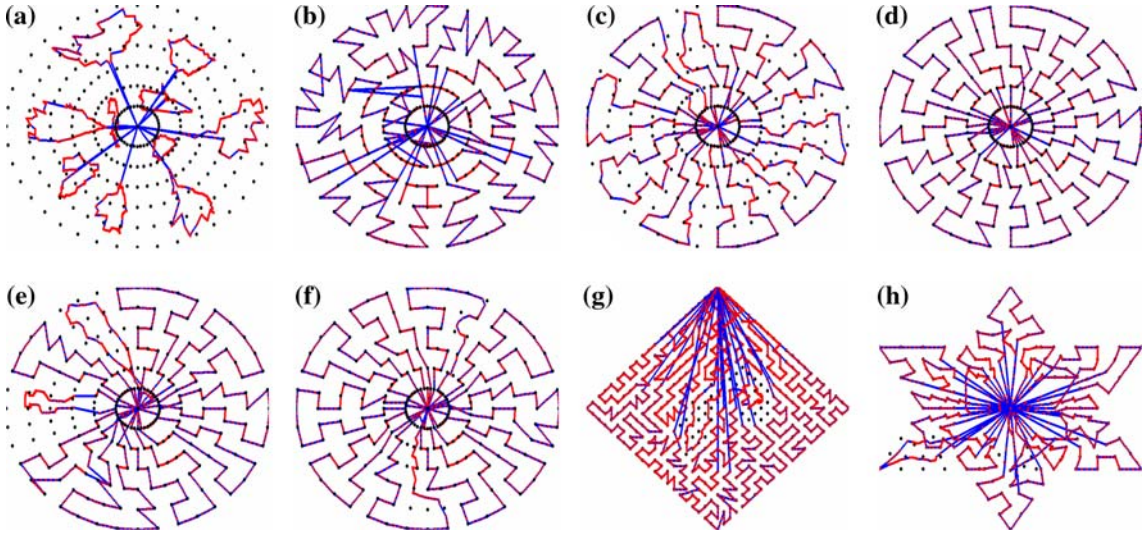

Fig. 2 Two phases algorithm applied on Golden et al. (1998) instances. Construction phase with instance no. 1 in **a**–**d**. Improvement phase with instance no. 1 in **e** and **f**, no. 12 in **g** and no. 17 in **h**

```
Repeat niter times.
   1. Randomly extract a point p from the demand set.
   2. Perform competition to select the winner vertex n* according to p.
   3. Apply learning law to move the neurons within a neighborhood of n*.
   4. Slightly decrease learning rate α and radius σ of neighborhood.
End Repeat.
```

Fig. 3 Self-organizing map algorithm

The approach already embeds the LK heuristic as a main local search and incorporates some of the recent Helsgaun's enhancements as well. Following such an example, the goal is to outperform the SOM based approaches allowing the shortest possible running times, and to indirectly reduce the gap to the OR heuristics.

The optimization process is divided into two phases, that are, a construction phase followed by an improvement phase. Figure 2 illustrates such a behavior on large scale problems of the Golden et al. (1998) benchmark, showing the visual patterns, which represent routes starting and ending at the depot, moving and distorting in the plane. The construction phase is illustrated in Fig. 2a–d. Two consecutive pictures show the network as distorted by the SOM operator, followed by the application of other evolutionary operators which generate an admissible (or near admissible) solution, at a given generation. During the construction phase, the local moves are performed with a great intensity to let the set of routes deploys from scratch. Figure 2a–b presents the network at the beginning of construction and (c–d) several generations later, the intensity of moves vanishing. In Fig. 2e–h, the network is shown at different steps of the improvement phase, illustrating how local perturbations, that are responsible for local improvements, randomly affect some parts of the network at each generation. The next section details the algorithm and explains the role of each operator.

3 The evolutionary algorithm embedding self-organizing maps

3.1 The Kohonen's self-organizing map

The standard SOM (Kohonen 2001) algorithm operates on a non-directed graph $G = (A, E)$, called the network, where each vertex $n \in A$ is a neuron having a location $w_n = (x, y)$ in the plane. The set of neurons A is provided with the d_G induced canonical metric $d_G(n, n') = 1$ if and only if $(n, n') \in E$, and with the usual Euclidean distance $d(n, n')$.

The training procedure structure is summarized in Fig. 3. A given number of iterations *niter* are applied to a graph network, the vertex coordinates being randomly initialized into an area delimiting the data set. Here, the data set is the set of demands, or customers. Each iteration follows four basic steps. At each iteration t, a point $p(t) \in \Re^2$ is randomly extracted from the data set (extraction step). Then, a competition between neurons against the input point $p(t)$ is performed to select the winner neuron n^* (competition step). Usually, it is the nearest neuron to $p(t)$. Then, the following learning law (triggering step):

$$w_n(t+1) = w_n(t) + \alpha(t).h_t(n^*, n).(p - w_n(t)) \quad (1)$$

is applied to n^* and to all neurons within a finite neighborhood of n^* of radius σ_t, in the sense of the topological

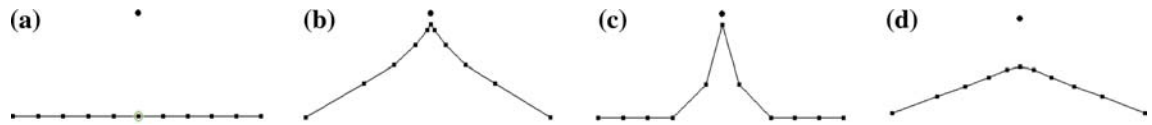

Fig. 4 A single SOM iteration with learning rate α and radius σ. **a** Initial configuration. **b** $\alpha = 0.9, \sigma = 4$. **c** $\alpha = 0.9, \sigma = 1$. **d** $\alpha = 0.5, \sigma = 4$

Fig. 5 The generic memetic loop embedding SOM

```
Initialize population with Pop individuals.
g = 0
While g < Gen
      1. Apply a standard SOM operator (denoted SOM), with a given
      parameter setting, to each individual in population separately. /*
      niter iterations are performed at each generation, for each
      individual. */
      2. Apply mapping operator MAPPING to each individual in population
      to assign each demand to a closest vertex, and then move vertices to
      demand locations.
      3. Apply fitness evaluation operator FITNESS to each individual in
      population.
      4. Save the best individual encountered.
      5. Apply selection operator SELECT.
      6. Apply elitist selection operator SELECT_ELIT.
      7. Apply derived operators from the self-organizing map algorithm
      structure, chosen from the set {SOM, SOMVRP1, SOMVRP2, SOMDVRP},
      separately to each individual in population. /* Such operators
      introduce perturbations or perform greedy insertion moves toward
      residual demands which are not yet already inserted in the network
      because of time duration constraint not satisfied. */
      8. g = g + 1
End while.
Report best individual encountered.
```

distance d_G, using learning rate $\alpha(t)$ and function profile h_t. The function profile is given by the Gaussian:

$$h_t(n^*, n) = \exp\left(-d_G(n^*, n)^2 / \sigma_t^2\right). \quad (2)$$

Finally, the learning rate $\alpha(t)$ and radius σ_t are slightly decreased as geometric functions of time (decreasing step). To perform a decreasing run within t_{max} iterations, at each iteration t coefficients $\alpha(t)$ and σ_t are multiplied by $\exp\left(\ln(x_{\text{final}}/x_{\text{init}})/t_{\max}\right)$, with respectively $x = \alpha$ and $x = \sigma$, x_{init} and x_{final} being respectively the values at starting and final iteration.

Examples of a basic iteration with different learning rates and neighborhood sizes are shown in Fig. 4. In our evolutionary algorithm, a SOM simulation becomes a main operator specified by its running parameters $(\alpha_{\text{init}}, \alpha_{\text{final}}, \sigma_{\text{init}}, \sigma_{\text{final}}, t_{\max})$.

3.2 Evolutionary loop and operators

A construction loop as well as an improvement loop is instantiated based on the generic memetic loop structure presented in Fig. 5. The memetic loop applies a set of operators to a population of *Pop* individuals, at each iteration (called a generation). A loop executes a fixed number of generations *Gen*, proportional to the problem size N. The number of individuals is constant. One individual encapsulates exactly one solution, that is, a network where each vehicle is represented by an independent ring with $5\,N/m$ vertices, m being the number of vehicles, starting and ending at the depot. The number of vertices by vehicle corresponds to the maximum number of customers a vehicle can visit. It has been adjusted empirically to allow a good compromise between number of customers visited, equilibration of route lengths and computation speed.

The construction loop starts its execution with solutions having randomly generated vertex coordinates into a rectangle area containing demands. The improvement loop starts with the best previously constructed solution, which is duplicated in the new population. The main difference between the construction and improvement loops is that the former is responsible for creating an initial ordering from random initialization. It follows that SOM processes embedded in the construction loop have a larger initial neighborhood, proportional to N. The improvement loop, however, is intended for simply performing local improvements using SOM processes with small neighborhoods. Thus, construction and improvement loops play different roles and have a different computation time complexity. They are managed by a master loop possibly controlling restart executions. The master loop is presented in the next subsection and the parameter settings are given in the experimental analysis section.

The main operator is the SOM algorithm. At each generation, a predefined number of SOM basic iterations, proportional to problem size N, are performed letting the long simulation run being interrupted and combined with the

application of other operators. Such operators can be the specialization of a SOM operator to perform request insertions, or introduce perturbations, a mapping/assignment operator for generating admissible orderings, a fitness evaluation, and the selections at the population level. Operators can be seen as interleaving processes, each one influencing the search toward problem goals. Below is a detailed description of these operators:

(1) Self-organizing map operator. It is the standard SOM applied to the graph network. It is denoted by its name and its internal parameters, as $SOM(\alpha_{\text{init}}, \alpha_{\text{final}}, \sigma_{\text{init}}, \sigma_{\text{final}}, t_{\max})$. One or more instances of the operator, with their own parameter values, can be combined. A SOM operator is executed performing *niter* basic iterations by individual, at each generation. Parameter $t_{\max}$ is the number of iterations defining a long decreasing run performed in the stated generation number *Gen*, for each individual. Other parameters define the initial and final intensity and neighborhood for the learning law, as explained in the previous subsection. The operator is used to deploy the network toward customers in construction phase, or to introduce punctual moves to exit from local minima in improvement phase.

(2) SOM derived operators. Two types of operators are derived from the SOM algorithm structure in Fig. 3 for dealing with the VRP. The first type has two versions. The first operator, denoted *SOMVRP*1, is a standard SOM restricted to be applied to a randomly chosen vehicle, using customers already inserted into the vehicle/route. It helps to eliminate the remaining crossing edges in routes. The second operator, denoted *SOMVRP2*, is a variant in order to help to exit from undesirable local states. It is applied as above to a vehicle which, this time, is chosen because it has a vertex closest to a randomly chosen non-inserted customer. The non-inserted customer is furthermore forced to be assigned to a next closest vehicle, by moving its nearest vertex to the customer location. This operator is applied with a small probability at each generation. It has a negligible role except on few large scaled difficult instances.
While capacity constraint is greedily tackled by the mapping/assignment operator below, the second type operator, denoted *SOMDVRP*, deals specifically with the time duration constraint. It performs few greedy insertion moves at each generation. Given a randomly chosen customer that is not yet already assigned to a vehicle, the competitive step selects to be the winner the vehicle vertex for which the route time increase is minimum, the route time duration constraint for that vehicle being satisfied. The evaluation of the route time increase is done moving the vertex to the customer location and including the customer into the route.

(3) Mapping/assignment operator. This operator, denoted *MAPPING*, generates a VRP solution by inserting customers into routes and modifies the shape of the network accordingly, at each generation. The operator first greedily maps customers to the nearest vertex for which the corresponding vehicle capacity constraint is satisfied, and to which no customer has been yet assigned. The capacity constraint is then greedily tackled by the customer assignment. Then, the operator moves the vertices to the location of their assigned customer (if exist) and regularly dispatches (by translation) other vertices along edges formed by two consecutive customers in a route. This results in a vehicle route where assigned vertices alternate with the many more not assigned vertices. At this stage, few customers are possibly not inserted because of capacity constraint violation.

(4) Fitness operator, denoted *FITNESS*. Once the assignment of customers to routes has been performed, this operator evaluates a scalar fitness value for each individual that has to be maximized and which is used by the selection operator. Taking care of time duration constraint, the fitness value is sequentially computed following routes one by one and removing a customer from the route if it leads to a violation of the time duration constraint. The value returned is $fitness = sat - 10^{-5} \times length$, where *sat* is the number of customers that are successfully assigned to routes, and *length* is the length of the routes defined by the ordering of such customers mapped along the rings. The value *sat* is then considered as a first objective and admissible solutions are the ones for which $sat = N$, N being the number of customers.

(5) Selection operators. Based on fitness maximization, the operator denoted *SELECT* replaces *replace* worst individuals, who have the lowest fitness values in the population, by the same number of bests individuals, which have the highest fitness values in the population. An elitist version *SELECT_ELIT* replaces the worst individuals by the single best individual encountered during the current run.

3.3 Master loop

We are now going to describe the main program controlling construction and improvement loops, possibly performing restart executions. The master loop, as depicted in Fig. 6, sequentially executes the two memetic loops to respectively perform the construction and improvement phases. The construction loop starts its execution with randomly generated network patterns. The improvement loop starts with the best previously constructed solution, which is duplicated in the new population. Once an execution ended, the master loop performs population movements. The process of construction, as well as improvement, can be repeated more than

```
While (not NExec executions are performed or sat < N)
  1. Initialize population with Pop randomly generated individuals
  2. Apply construction loop /* GenC generations are performed */
  3. if (sat < N × 99.5/100) go to step 1. (one time max)
  4. Initialize population by duplicating the best constructed individual
  5. fitness' = fitness /* memorize fitness of the best */
  6. Apply improvement loop /* GenI generations are performed */
  7. if (sat < N /*or fitness > fitness'*/) go to step 5. (four times max)
End while
Report best individual encountered
```

Fig. 6 The master loop controlling restart executions

Table 1 Construction loop parameter setting

$Pop = 50$ individuals
$GenC = N$ generations

Operator		Internal SOM parameters					Evolutionary parameters	
		α_{init}	α_{final}	σ_{init}	σ_{final}	$t_{\max}$	*niter*	*replace*
1	*SOM*	0.5	0.5	$2 \times N/m$	4	$GenC \times niter$	$N/4$	–
2	*MAPPING*	–	–	–	–	–	–	–
3	*FITNESS*	–	–	–	–	–	–	–
4	*SELECT*	–	–	–	–	–	–	*Pop*/5
5	*SELECT_ELIT*	–	–	–	–	–	–	*Pop*/10
6	*SOMDVRP*	1.0	1.0	0.5	0.5	–	Min($N \times 5/100$, $N - sat$)	–

one time depending on the number of constraints satisfied (sat $< N$), and possibly the fitness improvement carried out (fitness $>$ fitness$'$). A complete run executes both configurations possibly several times (*NExec*), restarting from random individuals at each time. Here, restarts are used as a usual mechanism to augment accuracy and robustness, eliminating the probability to get non-admissible solutions.

3.4 Algorithm complexity

By the evolutionary dynamics, the goal is to make the closest point assignment coincide to the right assignment, which minimizes objectives and satisfies constraints. The algorithm can be seen as a massive and parallel insertion method to the nearest points. The SOM operators, as well as the mapping operator, are based on the such closest point findings. Then, with a constant population size, a number of generations proportional to N, a SOM neighborhood proportional to N, and N basic iterations performed by generation, N being the problem size, the time complexity of the memetic SOM is $O(N^3)$ in the worst case. The memetic SOM space complexity is $O(N)$, as usual for SOM. It is worth noting that this space complexity allows dealing with very large size instances, on standard computer workstations.

Here, to perform N closest point findings in expected $O(N)$ time for uniform distributions, rather than $O(N.\log(N))$ in the worst case (Preparata and Shamos 1985), we have implemented the spiral search algorithm of Bentley et al. (1980) based on a cell partitioning of the area. Thus, SOM and mapping operators share a similar O(N) expected time complexity for the closest point findings by generation, for a uniform distribution. We claim that this mechanism considerably improves the nearest search performance with regards to usual applications where a complete examination of neurons is performed. The derived version *SOMDVRP* has a $O(N)$ time complexity by iteration, since it systematically inspects all the $O(N)$ vertices. Also, it needs to be applied following the mapping operator and fitness evaluation in order to use the built assignment structures and access the few not yet inserted demands.

4 Experimental analysis

In this section, we present the parameter setting of the algorithm and perform some evaluations and analysis of the main characteristics of the approach. We study the influence of the population size and the role of the main algorithmic components. Statistic analysis is performed based on confidence intervals and t tests.

4.1 Parameter setting

The construction loop parameter setting is detailed in Table 1, and the improvement loop parameter setting, in Table 2. The operators are listed in the tables following their order of

Table 2 Improvement loop parameter setting

$Pop = 50$ individuals
$GenI = N$ generations

Operator		Internal SOM parameters						Evolutionary parameters	
		α_{init}	α_{final}	σ_{init}	σ_{final}	$t_{\max}$	*prob*	*niter*	*replace*
1	*MAPPING*	–	–	–	–	–	–	–	–
2	*FITNESS*	–	–	–	–	–	–	–	–
3	*SELECT*	–	–	–	–	–	–	–	*Pop*/5
4	*SELECT_ELIT*	–	–	–	–	–	–	–	*Pop*/10
5	*SOMDVRP*	–	–	–	–	–	–	$\text{Min}(N \times 5/100, N - sat)$	–
6	*SOM*	0.9	0.5	$2 \times N/m$	$(2 \times N/m)/4$	$GenI \times niter$	–	1	–
7	*SOMVRP1*	0.5	0.5	10	4	$GenI \times niter$	–	N/m	–
8	*SOMVRP2*	1.0	1.0	10	4	$GenI \times niter$	0.1	$(N/m)/4$	–

application in a loop. The parameter *Pop* is the population size; *GenC* and *GenI* are respectively the generation number of the construction and improvement loop. Parameter values for the SOM operators and selections were adjusted after a preliminary round of experiments. Most of the parameter values depend on the main problem parameter, that is, the problem size N, taking care of the number of vehicles m. They are either constant values or set proportional to N, or to the number of vertices by route fixed to $5\ N/m$. Such internal parameter values are quite standard when dealing with SOM for TSP or VRP.

The construction loop (Table 1) uses a SOM process with an initial large neighborhood proportional to N, defined by σ_{init} and σ_{final} parameters, in order to deploy the network ordering from random initialization of vertices. Improvement loop (Table 2) operators have smaller neighborhoods, and apply less iterations by generation, since intended for local improvement. The SOM operator *SOM* in Table 2 performs a single punctual move at each generation to help to exit from local minima. The derived operators *SOMVRP*1 and *SOMVRP*2 work as stated in the previous section, to locally regulate a route shape. The *SOMVRP*2 operator has a low probability ($prob = 0.1$) of application at each generation. Its plays a subsidiary role and was added to deal with few hardest large size instances. The *SOMDVRP* tries to insert few residual and not yet satisfied customers at each generation.

4.2 Performance with regard to population size

Starting with the parameter settings given in the previous subsection, we apply the two loops in sequence, once each, and study the influence of the population size *Pop* on the fitness value. The generation numbers *GenC* and *GenI* are adjusted in order to perform the same number of fitness evaluations, for each loop, and for each run. The tests are done using the c10 instance with 200 customers of the Christofides et al. (1979) benchmark. This test is the larger instance of the test set. It encompasses both the capacity constraint and time duration constraint. For each value of the *Pop* parameter, 10 runs are performed.

Figure 7a presents the progress plots of the fitness according to the number of fitness evaluations carried out for four population sizes from 1 to 100 individuals. The total number of fitness evaluations is fixed. Let N being the number of customers. The generation number of each loop is

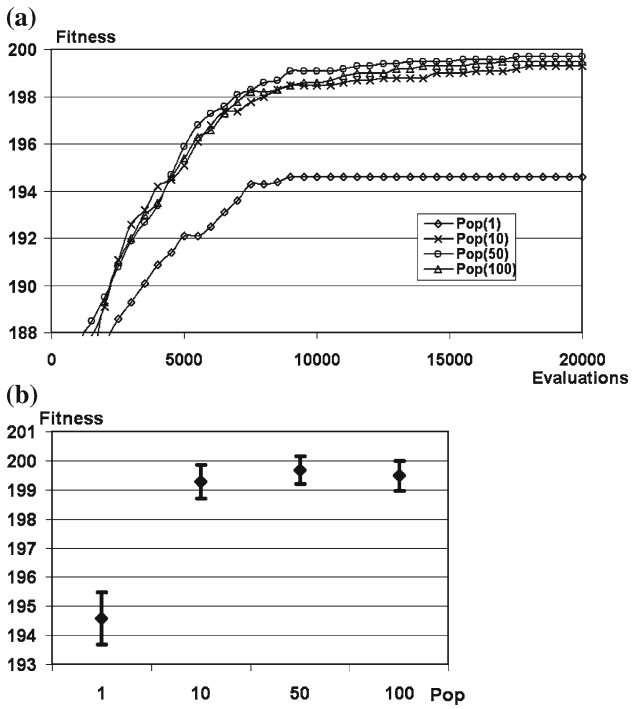

Fig. 7 Performances of memetic SOM according to population size. **a** Achieved fitness against the number of fitness evaluations. **b** Fitness values, in 95% confidence intervals, at the last generation

set to N for case $Pop = 50$, whereas it is set to $N \times 50$ for $Pop = 1$, $N \times 5$ for $Pop = 10$, and to $N/2$ for case $Pop = 100$.

Figure 7b presents the sample mean at the last generation in error bars. Error bars are 95% confidence intervals for the mean. They are computed on the basis of standard deviations over the 10 runs as follows. Assuming that the sample mean $\hat{\mu}$ and standard deviation $\hat{\sigma}$ for $K = 10$ replicates are computed as

$$\hat{\mu} = \frac{1}{K}\sum_{k=1}^{K} y_k,$$

where $y_k, k = 1, \ldots, 10$, are the fitness values at the last generation, and

$$\hat{\sigma} = \sqrt{\frac{1}{K-1}\sum_{k=1}^{K}\left(y_k - \hat{\mu}\right)^2}$$

a confidence interval that would cover the true mean approximately 95% of the time is

$$\hat{\mu} \pm t_{K-1,.025} \cdot \frac{\hat{\sigma}}{\sqrt{K}},$$

where $t_{K-1,.025}$ is the classic Student's t-statistic with $\alpha = .025$ and K-1 degrees of freedom.

The confidence level is $\alpha = .025$ instead of $\alpha = .05$ because the confidence interval is symmetric about $\hat{\mu}$, so the total probability of including the true mean μ is .95, but the probability of missing μ on either side of $\hat{\mu}$ is .025. For $K = 10$ runs, $t_{K-1,.025} = 2.262$. It is worth noting that the procedure for establishing confidence intervals and hypothesis testing are almost identical. Both require the standard error $\frac{\hat{\sigma}}{\sqrt{K}}$ of the sample mean to define the width of a critical region based on the Student distribution. It is a well known result that whenever two confidence intervals do not overlap (if the upper bound of one is below the lower bound of the other) then a two-sample t test will say the means are different, i.e. the null hypothesis will be rejected. In our experiments, this means that a one-sided t test would reject the null hypothesis at the significance level of $\alpha = 2.5\%$, or that a two-sided t test would reject it at the $\alpha = 5\%$ level. But the converse being not true, we may explicitly run a two-sample t test when two confidence intervals overlap.

The experimental results reported in Fig. 7 establish that with the same computational cost in terms of number of evaluations carried out—except when population size is reduced to a single individual—solutions tend to converge toward overlapping target values. Since half of the simulation run concerns each loop, we may note from Fig. 7a that the improvement loop has no effect on the fitness value only when $Pop = 1$. In that case, i.e. with inhibition of selection, the evolution process can be rather assimilated to local search. As confirmed in Fig. 7b, these trials illustrate local search limitation and cast a light on the improvements carried out by the population based search. The case $Pop = 50$ seems to be the best performing case. This could be explained by the number of generations which is more adequate for this case than for the case $Pop = 100$.

4.3 Influence of main algorithmic components

We now turn to an analysis of the main algorithmic components influence, for a fixed population size of 50 individuals. Here again, we use the Christofides, Mingozzi, and Toth (CMT) c10 instance, performing 10 runs by test with a chosen component being removed from the algorithm, and report the mean values within confidence intervals. We use the algorithm configuration with $Pop = 50$ of the previous subsection as a basis for comparison. Figure 8a presents progress plots of the fitness according to the number of generations. Figure 8b presents the mean fitness values obtained at the last generation in 95% confidence intervals. Figure 8c presents the mean route lengths, as well in 95% confidence intervals.

Clearly, selection operators have the greatest influence on the algorithmic performance. Without any selection operator, the algorithm behaves like executing 50 independent runs of local search. In that case, about 196 demands only out of a total of 200 are satisfied on average. This is slightly better than performing a single very long run with $Pop = 1$, as done in the previous subsection. But again, in that case, the improvement phase (half of the run) has no effect on the fitness value.

Also, the *SOMDVRP* operator, responsible for random insertions according to time duration constraint plays an important role. Removing this component yields to solution with less than 200 customers successfully inserted, thus to non-admissible solutions.

Removing the elitist selection, or *SOMVRP*1 operator responsible for single route improvement, yields to fitness values overlapping with the case $Pop = 50$. Then, in order to decide whether these two operators play a significant role, we need to look at the secondary fitness component, which is the total route length, in fact the main VRP objective to minimize. Figure 8c shows that removing the elitist selection has a non-negligible impact on the route length. Since the confidence intervals overlap we performed a one sided two-sample t test with the case $Pop = 50$. It yields a critical value of 18.27 at the significance level $\alpha = 0.025$ whereas the difference of the sample means is 19.42. This indicates that the case $Pop = 50$ performs better. However, removing the *SOMVRP* has a weaker impact on the route length. In that case, a one-sided two-sample t test failed at rejecting the null hypothesis even at the $\alpha = 0.05$ level. Thus, we can not draw a firm conclusion about the utility of the *SOMVRP* operator. This can be explained by the particular

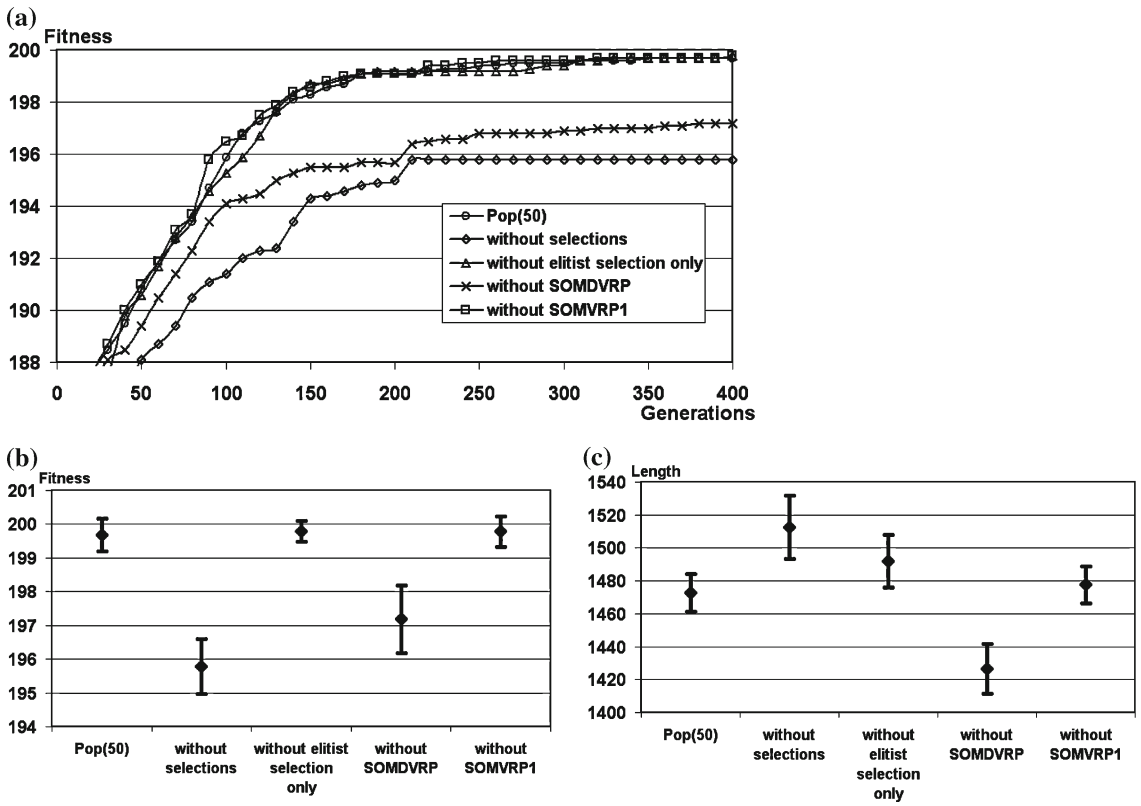

Fig. 8 Performances of memetic SOM removing algorithmic components. **a** Achieved fitness against the number of generations. **b** Fitness values in 95% confidence intervals. **c** Route lengths in 95% confidence intervals

test case used, where the time duration constraint may play a predominant role. Nevertheless, the Pop = 50 configuration case wins in all cases, considering both fitness and length. Finally, it should be noted that the fitness value tends to be stable from around generation 180 to the end of the construction phase, at generation 200, and that the improvement loop starting at generation 200 plays a non-negligible role in all cases, except when there is no selection, to "finalize" the search.

5 Evaluation against NN and OR heuristics

In this section, we perform a comparative evaluation of the memetic SOM against some of the most accurate Neural Network and Operations Research approaches to the VRP of the literature. We first present the context of the evaluations. We explain the choice of the test problems and indicate how computation times are evaluated, according to the different materials used. We also specify the number of runs and define what are the values reported. We evaluate the approach on usual small size instances. Then, we evaluate the performances on the large size instances.

5.1 Description of the experimental context

Here, evaluation of the memetic SOM approach is done against neural network algorithms and Operations Research heuristics. We consider the three SOM based approaches of Ghaziri (1996), Modares et al. (1999), and Schwardt and Dethloff (2005) which provide comparative studies, and report results for the well known Christofides et al. (1979) benchmark problems, denoted CMT problems below. Other neural network versions (Gomes and Von Zuben 2002; Matsuyama 1991; Schumann and Retzko 1995; Vakhutinsky and Golden 1994) are quite algorithmically similar or clearly worse performing. Only Ghaziri (1996) addresses the time-duration version of VRP and solves almost all the corresponding CMT test cases.

We also evaluate the approach against some of the most recent and representative Operations Research heuristics presented in Cordeau et al. (2005). In that case, we will focus on the large size test problems of Golden et al. (1998). The selected approaches have displayed the best performance according to the literature, such as the Active Guided Evolution Strategy (AGES) (Mester and Bräysy 2005) and Granular Tabu Search (GTS) (Toth and Vigo 2003), or are considered more simple and flexible but less performing, such as

Table 3 Estimated performances of computers

Computer	Performance estimated from Dongarra's paper (Mflop/s)	Estimated Dongarra's factor
Sun Sparc 2 workstation	4	1/212
Pentium 200 MHz	38	1/22.3
Pentium III 733 MHz	138	1/6.2
Pentium IV 2000 MHz	781.7	1/1.1
AMD Athlon 2000 MHz	849.2	1

Table 4 Master loop parameter settings for the comparative evaluations

Algorithm configuration type	*GenC*	*GenI*	*NExec*	Criterion for stopping the improvement loop restarting
Fast	$N/2$	$N/2$	1	*sat* = *N*
Medium fast	N	N	1	*sat* = *N and fitness not improved*
Medium long	N	N	5	*sat* = *N and fitness not improved*
Long	N^*5	N^*5	5	*sat* = *N and fitness not improved*

the Unified Tabu Search Algorithm (UTSA) (Cordeau et al. 2001) and Very Large Neighborhood Search VLNS (Ergun et al. 2003).

The proposed memetic SOM was programmed in Java and run on an AMD Athlon 2000 MHz computer. To evaluate the tradeoff between running time and solution quality against other algorithms, one difficulty is that testing is done using different computers. As done in many works, a rough indication of the machine's relative speed may be derived from the Mflop/s measure obtained by Dongarra (2006). Table 3 presents the estimated performances and Dongarra's factors used to convert CPU times of the different materials considered in this paper.

The master loop parameter setting which controls construction and improvement memetic loops is detailed in Table 4. Four configurations of the master loop are considered. They are defined to adjust computation time in accordance with the other compared approaches. The adjusted parameters are the generation number of each loop *GenC* and *GenI*, the minimal number of restart runs *NExec*, and the criterion for stopping the improvement loop restarts. All the tests performed with the memetic SOM are done on a basis of 10 runs per instance. For each test case, is reported the percentage deviation, denoted "%PDM", to the best known optimal route length, of the mean solution value obtained, i.e. %PDM = (mean length − best known) ×100/ best known. Best known values are taken from Cordeau et al. (2005). The average computation time is also reported, it is called "Sec" when given in seconds or "Min" in minutes. Result values for the other approaches, which are taken from the above referenced papers, are given under the same appellations, computation times being normalized to our computer system.

As well, the widths of 95% confidence intervals for the sample means are systematically reported in columns denoted "±%WD". For individual test problems the confidence intervals are computed based on the standard deviation and the 10 replicates as stated in section 4.2. Whereas, when dealing with the sample means formed across different test problems, and as recommended in Rardin and Uzsoy (2001), we estimate the error variance from the mean square for error (MS_E) of a standard analysis of variance, i.e. the "mean squares within". Therefore, an approximate confidence interval is given by

$$\hat{\mu} \pm t_{M\cdot(K-1),.025} \cdot \sqrt{\frac{MS_E}{M \cdot K}},$$

where

$$MS_E = \frac{1}{M}\sum_{i=1}^{M} \hat{\sigma}_i^2,$$

M is the number of problems considered, and $\hat{\sigma}_i$ is the standard deviation based on the $K = 10$ runs for a single test problem.

5.2 Comparison with neural network approaches on the most often used CMT instances

Numerical results on CMT instances are given in Table 5 and illustrated in the graph of Fig. 9. The proposed approach is evaluated against the best performing neural network approaches (Ghaziri 1996; Modares et al. 1999; Schwardt and Dethloff 2005) found in the literature. The fourteen CMT instances of the capacitated VRP are composed of two subsets containing seven instances of each VRP types, with (D subset) or without time duration constraint (C subset). The first column "Name-size-veh" indicates the name, size and number of vehicles of the instance. The second column

Table 5 Numerical results for the CMT instances

Name-size-veh	Best	Memetic SOM (fast)			Memetic SOM (long)			Ghaziri (1996)		Modares et al. (1999)	Schwardt and Dethloff (2005)	Clarke and Wright	UTSA (Cordeau et al. 2001)	
		%PDM	±%WD	Sec[a]	%PDM	±%WD	Sec[a]	%PDM	Sec[b]	%Best	%PDM	%PDM	%PDM	Sec[c]
c1-50-5	524.61	2.37	2.03	1.87	0.00	0.00	131	2.78	0.25	2.36	1.43	11.44	0.00	127
c2-75-10	835.26	5.22	0.97	4.16	1.61	0.20	296			4.88	17.02	7.78	0.00	806
c3-100-8	826.14	1.52	0.42	6.86	0.36	0.21	490	8.14	1.84	4.46	3.78	7.35	0.00	637
c12-100-10	819.56	0.76	0.04	10.04	0.74	0.00	518	0.68	0.48		1.15	1.70	0.00	492
c11-120-7	1042.11	8.41	2.20	12.00	1.62	0.36	1311	5.79	1.19	2.29	7.33	2.78	3.01	637
c4-150-12	1028.42	3.08	0.33	15.38	1.75	0.18	1225	5.47	3.74	5.21	5.67	10.21	0.41	1454
c5-199-17	1291.26	5.14	0.40	26.50	2.94	0.32	2569	8.53	6.57	7.34	6.83	8.09	1.90	3146
Average Ghaziri		3.55	0.45	12.11	1.24	0.08	1040	5.23	2.34		4.37	6.93	0.89	1082
Average Modares		4.29	0.48	11.13	1.38	0.09	1004			4.42	7.01	7.94	0.89	1134
Average C		3.79	0.41	10.97	1.29	0.07	934				6.17	7.05	0.76	1043
c6-50-6	555.43	1.46	0.58	1.80	0.77	0.21	126	1.06	1.22			11.34	0.00	165
c7-75-11	909.68	2.69	0.93	64.51	0.93	0.47	470					7.23	0.00	404
c8-100-9	865.94	1.82	0.40	7.08	0.15	0.28	517	3.28	5.21			12.47	0.00	596
c14-100-11	866.37	0.90	0.86	7.52	0.03	0.05	570	1.55	2.41			1.08	0.00	574
c13-120-11	1541.14	5.20	2.07	18.44	0.89	0.18	1141	4.37	8.86			3.61	0.53	1145
c9-150-14	1162.55	4.03	0.64	30.78	1.82	0.22	1617	8.73	7.70			10.76	0.46	2818
c10-199-18	1395.85	6.22	0.93	58.85	3.16	0.33	3851	13.22	14.83			10.23	1.50	5797
Average Ghaziri		3.27	0.39	20.75	1.14	0.08	1303	5.37	6.70			8.25	0.41	1849
Average D		3.19	0.35	27.00	1.11	0.09	1185					8.10	0.36	1643
Average all		3.49	0.27	18.99	1.20	0.06	1060					7.58	0.56	1343

[a] Time per run in AMD Athlon (2000 MHz) seconds Java program
[b] Time per run in Sun Sparc 2 workstation seconds normalized to AMD Athlon (2000 MHz) seconds
[c] Time per run in Pentium IV (2000 MHZ) seconds normalized to AMD Athlon (2000 MHz) seconds

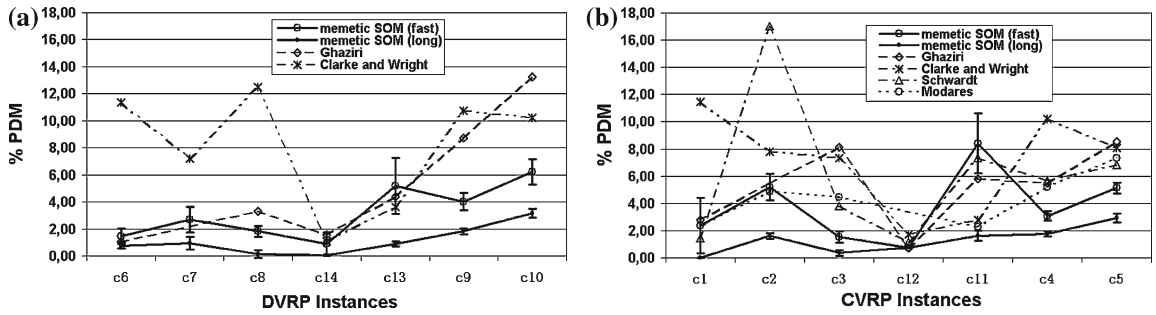

Fig. 9 Comparison with neural network approaches on the CMT instances

indicates the best known values that were obtained initially for a large part by Rochat and Taillard (1995) through a long run. We used the two configurations "fast" and "long" of the algorithm, as specified in previous section, to adjust the computation time.

Approaches are mentioned in Table 5 using the name of the method and/or the names of the authors and the date of publication. Results for a given approach are reported in column "%PDM", "±%WD" and column "Sec". Results for the Modares et al. approach are reported in column "%Best" since they are best values obtained over 10 runs. Results in Table 5, illustrated in Fig. 9, show the improvement carried out by memetic SOM against earlier neural network approaches.

Accuracy is substantially improved since the average deviation to the best known value is reduced from roughly 5 to 3.39% on average for fast runs and to 1.20% for long runs.

Here, all CMT test cases are addressed successfully. Modares et al. (1999) and Schwardt and Dethloff (2005) do not report computation time. Furthermore, they do not address the time duration version of the problem. Only Ghaziri (1996) deals with it, but forgetting to solve the c2 and c7 test cases which are two of the hardest instances. Taking care of the rough approximation performed when comparing computation times, memetic SOM looks like spending more time than Ghaziri (1996) but clearly improves solution quality and solves all the test cases. We can note that the 95% confidence intervals stated in column "±%WD" do not include other approaches means except in the case of the Modares approach. We can note that the average PDM values of the Ghaziri approach clearly fall outside of the intervals.

We also report in Table 5 results obtained by the Clarke and Wright construction heuristic (Clarke and Wright 1964), available at VRP Web http://neo.lcc.uma.es/radi-aeb/WebVRP/. Neural networks perform better than the construction heuristic, for more computation time. Finally, results obtained by the UTSA approach (Cordeau et al. 2001) are given in the two last columns. They illustrate the gap to such a recent Operations Research heuristic. With roughly 27% more computation time, UTSA yields 0.56% of average deviation, whereas our approach yields 1.20%. Since large instances are the ones which justify the use of heuristics, we will now evaluate this gap more closely on the larger instances of Golden et al. (1998).

5.3 Comparison with operations research heuristics on large size instances

The results for the 20 test problems of Golden et al. (1998) are reported in Table 6 and illustrated in Fig. 10. The benchmark contains height problems with time duration constraint (D subset) and twelve problems without (C subset). Comparison is done against some of the recent heuristics presented in the review paper (Cordeau et al. 2005) that cover the global range of metaheuristic performances. They are the AGES (Mester and Bräysy 2005), GTS (Toth and Vigo 2003), UTSA (Cordeau et al. 2001), and VLNS (Ergun et al. 2003) approaches. We use the numerical results reported in Cordeau et al. (2005). We have performed 10 runs per instance using the two configurations of the memetic SOM "medium-fast" and "medium-long", to adjust computation time in accordance to the compared approaches. As above, we report the percent deviation of the mean solution (%PDM) according to the best known value, the 95 % confidence interval for the mean (±%WD), and the average computation time (Min) in minutes.

From what we know, the AGES approach is, at the date of writing, the overall winner considering both solution quality and computation time. Only the very recent approach of Kytöjoki et al. (2007) mentions faster computation times (0.003 min on a AMD Athlon64 3000+) but with lesser solution quality (1.98% deviation), for the same test problems. AGES is however considered complicated (Cordeau et al. 2005). On the contrary, UTSA is recognized as to be simple (easy to understand and implement) and flexible (easy to extend) but more time consuming.

Considering DVRP instances in Fig. 10a, deviation is closest to the one of UTSA for slightly spending more computation time. For such instances, GTS yields worst quality results but computes very quickly. Considering the CVRP instances in Table 6, and Fig. 10b, memetic SOM is less

Table 6 Numerical results for the Golden et al. (1998) instances

Name-size veh	Best	Memetic SOM (medium fast)			Memetic SOM (medium long)			AGES fast (Mester and Bräysy 2004)		AGES long (Mester and Bräysy 2004)		GTS (Toth and Vigo 2003)		VLNS (Schwardt and Dethloff 2005)		UTSA (Cordeau et al. 2001)	
		%PDM	±%WD	Min[a]	%PDM	±%WD	Min[a]	%PDM	Min[b]	%PDM	Min[b]	%PDM	Min[c]	%Best[d]	Min[e]	%PDM	Min[f]
1-240-10	5627.54	4.17	0.04	4.16	4.11	0.08	18.89	0.29	0.64	0.00	7.94	1.93	0.22	2.03	21.77	0.97	9.35
2-320-10	8447.92	0.91	0.38	7.51	0.43	0.08	38.67	0.24	0.18	0.00	42.42	1.24	0.37	5.56	24.33	2.48	32.17
3-400-10	11036.22	3.24	0.45	17.02	2.18	0.21	65.93	0.99	0.64	0.00	36.86	3.32	0.58	9.70	2.53	0.01	50.35
4-480-12	13624.52	3.09	0.90	18.89	3.05	0.62	83.82	0.59	2.27	0.00	427.27	9.44	0.68	12.42	17.18	0.85	75.63
5-200-5	6460.98	4.33	1.87	2.79	0.42	0.03	11.84	0.08	0.45	0.00	0.15	3.66	0.11	1.69	2.50	4.57	4.66
6-280-8	8412.8	1.58	0.98	5.75	0.83	0.07	25.85	1.51	0.096	0.00	68.38	6.54	0.21	5.03	13.22	1.48	16.95
7-360-9	10195.56	2.16	0.62	10.55	1.59	0.36	43.95	0.44	0.77	0.00	2.32	3.45	0.52	9.03	13.71	0.70	23.27
8-440-11	11663.55	54.04	0.62	15.07	3.10	0.10	64.40	2.19	0.25	0.00	31.18	3.20	0.50	8.32	5.48	1.77	64.95
Average D		2.94	0.11	10.22	1.96	0.08	44.17	0.79	0.66	0.00	77.07	4.10	0.40	6.72	12.59	1.60	34.67
9-255-14	583.39	3.43	0.69	3.24	3.18	0.33	14.87	0.83	0.73	0.00	7.57	1.71	0.52	0.77	7.94	0.69	33.87
10-323-16	742.03	3.15	0.33	5.58	2.54	0.16	24.60	1.47	0.39	0.00	5.45	1.30	0.71	1.05	20.17	1.45	46.46
11-399-18	918.45	3.27	0.35	8.86	2.74	0.321	40.58	0.82	1.00	0.00	100.00	1.92	1.49	1.56	27.59	1.16	37.76
12-483-19	1107.19	5.81	0.78	52.80	5.01	0.71	86.03	1.94	1.36	0.00	545.45	3.61	1.92	2.48	62.68	1.11	142.74
13-252-27	859.11	4.01	0.22	2.78	3.65	0.20	13.53	0.71	0.16	0.00	9.32	1.13	0.51	1.37	37.92	1.95	31.66
14-320-30	1081.31	4.47	0.61	5.00	3.49	0.43	27.01	1.51	0.25	0.00	1.11	1.38	0.65	1.46	5.03	1.92	19.60
15-396-34	1345.23	4.02	0.26	9.83	3.27	0.21	40.63	0.71	0.24	0.00	6.52	1.80	0.83	1.63	10.53	1.38	52.40
16-480-38	1622.69	4.22	0.20	13.77	3.79	0.22	59.25	0.76	1.05	0.00	18.18	1.83	1.03	1.25	5.09	1.50	117.73
17-240-22	707.79	2.89	0.52	3.06	2.23	0.24	13.20	0.34	0.15	0.00	0.68	0.46	0.64	1.22	36.07	0.44	16.39
18-300-28	998.73	3.32	0.30	4.82	2.80	0.15	23.08	1.08	0.16	0.00	2.27	1.81	0.96	2.46	48.26	1.59	61.01
19-360-33	1366.86	2.59	0.32	7.12	2.23	0.18	36.40	1.10	0.23	0.00	5.45	2.49	1.35	2.78	63.39	1.24	60.19
20-420-41	1821.15	3.64	0.34	11.13	3.29	0.14	48.73	1.07	0.50	0.00	7.64	5.20	1.93	3.41	19.62	1.82	122.99
Average C		3.73	0.28	10.67	3.18	0.09	35.66	1.03	0.52	0.00	59.14	2.05	1.05	1.79	28.69	1.35	61.90
Average all		3.42	0.13	10.49	2.70	0.06	39.06	0.93	0.58	0.00	66.31	2.87	0.79	3.76	22.25	1.45	51.01

[a] Time per run in AMD 2000 MHz minutes, java program
[b] Time per run in minutes, normalized from Pentium IV 2000 MHz to AMD Athlon 2000 MHz
[c] Time per run in minutes, normalized from Pentium 200 MHz to AMD Athlon 2000 MHz
[d] Best of two runs
[e] Time for reaching the best value for the first time, Pentium III 733 MHz normalized to AMD Athlon (2000 MHz) minutes
[f] Time per run in minutes, normalized from Pentium IV 2000 MHz to AMD Athlon 2000 MHz

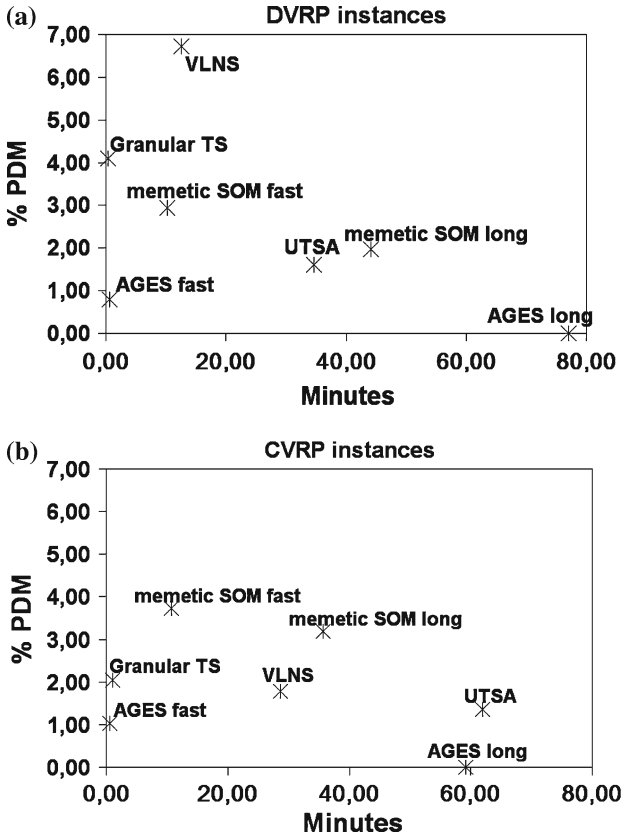

Fig. 10 Comparison with operations research heuristics on Golden et al. (1998) large size instances

performing on accuracy than all approaches, computation times being comparable or lesser than the ones of UTSA and VLNS. On all the tests, the proposed memetic SOM performs better on average than VLNS considering both quality solution and computation time, the difference specifically coming from the DVRP instances. It yields 3.42% of average deviation to best known value in roughly 11 min, whereas VLNS 3.76% in 22 min normalized to the same computer. Also, memetic SOM yields 2.70% deviation in 39 min on average, whereas UTSA yields 1.45% in 51 min. Again, the 95% confidence intervals in Table 6 clearly discriminate the performances of the different approaches.

6 Conclusion

Overall, by incorporating the SOM into an evolutionary algorithm, the approach extends and improves neural networks to the VRP. Operators have a similar structure based on the closest point findings and simple moves performed in the plane in a massively parallel fashion. The evolutionary framework adds another level of parallel computation and improves accuracy as well. This differentiates the approach from classical heuristics which operate generally (and sequentially) on graphs. The approach becomes nearly competitive with some of the classical heuristics for large size problems. Applications to other routing problems within a dynamic and stochastic context are questions to be addressed. Exploiting the natural parallelism of the approach for multi-processor implantations is also a key point to study in further work.

References

Arora S (1998) Polynomial-time approximation schemes for Euclidean TSP and other geometric problems. J ACM 45(5):753–782

Boese KD, Kahng AB, Muddu S (1994) Adaptive multi-start technique for combinatorial global optimization. J Oper Res Lett 16:101–113

Buriol L, França PM, Moscato P (2004) A new memetic algorithm for the asymmetric traveling salesman problem. J Heuristics 10(5):483–506

Bäck T, Hoffmeister F, Schwefel HP (1991) A survey of evolution strategies. In: 4th Int. Conf. on Genetic Algorithms, La Jolla, CA

Bentley JL, Weide BW, Yao AC (1980) Optimal expected time algorithms for closest point problems. ACM Trans Math Softw 6(4):563–580

Christofides N, Mingozzi A, Toth P (1979) The vehicle routing problem. In: Christofides N et al. (eds) Combinatorial optimization. Wiley, New York, pp 315–338

Clarke G, Wright JW (1964) Scheduling of vehicles from a central depot to a number of delivery points. Oper Res 12:568–581

Cordeau JF, Laporte G, Mercier A (2001) A unified tabu search heuristic for vehicle routing problems with time windows. J Oper Res Soc 52:928–936

Cordeau JF, Gendreau M, Hertz A, Laporte G, Sormany JS (2005) New heuristics for the vehicle routing problem. In: Langevin A, Riopel D (eds) Logistics systems: design and optimization. Springer, New York pp 279–297

Créput JC, Lissajoux T, Koukam A (2000) A connexionnist approach to the hexagonal mesh generation. In: 16th IMACS World Congress, Lausanne

Créput JC, Koukam A, Lissajoux T, Caminada A (2005) Automatic mesh generation for mobile network dimensioning using evolutionary approach. IEEE Trans Evol Comput 9(1):18–30

Créput JC, Koukam A (2006) Local search study of honeycomb clustering problem for cellular planning. Int J Mobile Network Des Innov 1(2):153–160

Créput JC, Koukam A (2007) Interactive meshing for the design and optimization of bus transportation networks. J Transp Eng 133(9):529–538

Créput JC, Koukam A (2007) Transport clustering and routing as a visual meshing process. J Inform Optim Sci 28(4):573–601

Créput JC, Koukam A, Hajjam A (2007) Self-organizing maps in evolutionary approach for the vehicle routing problem with time windows. Int J Comput Sci Network Secur 7(1):103–110

Cochrane EM, Beasley JE (2003) The co-adaptive neural network approach to the Euclidean travelling salesman problem. Neural Networks 16:1499–1525

Dongarra JJ (2006) Performance of various computers using standard linear equations software. In: Technical Report CS-89-85, Department of Computer Science, University of Tennessee, USA, available at http://www.netlib.org/benchmark/performance.ps

Ergun Ö, Orlin JB, Steele-Feldman A (2003) Creating very large scale neighborhoods out of smaller ones by compounding moves: a study on the vehicle routing problem. MIT Sloan Working Paper No. 4393–02, USA

Gambardella LM, Taillard E, Agazzi G (1999) MACS-VRPTW: a multiple ant colony system for vehicle routing problems with time windows. In: Corne D, Dorigo M, Glover F (eds) New ideas in optimization. McGraw-Hill, UK pp 63–76

Gendreau M, Laporte G, Potvin J-Y (2002) Metaheuristics for the capacitated VRP. In: Toth P, Vigo D (eds) The vehicle routing problem. SIAM, Philadelphia pp 129–154

Ghaziri H (1996) Supervision in the self-organizing feature map: application to the vehicle routing problem. In: Osman IH, Kelly JP (eds) Meta-heuristics: theory & applications. Kluwer, Boston pp 651–660

Golden BL, Wasil EA, Kelly JP, Chao IM (1998) Metaheuristics in vehicle routing. In: Crainic TG, Laporte G (eds) Fleet management and logistics. Kluwer, Boston pp 33–56

Gomes LCT, Von Zuben FJA (2002) Vehicle Routing Based on Self-Organization with and without Fuzzy Inference. In: Proc. of the IEEE International Conference on Fuzzy Systems, vol. 2, pp 1310–1315

Goldberg DE (1994) Genetic algorithms in search, optimization and machine learning. Addison-Wesley, Reading, MA

Johnson DS, McGeoch LA (1997) The traveling salesman problem: a case study in local optimization. In: Aarts EHL, Lenstra JK (eds) Local search in combinatorial optimization. Wiley, London, pp 215–310

Johnson DS, McGeoch LA (2002) Experimental analysis of heuristics for the STSP. In: Gutin G, Punnen A (eds) Traveling salesman problem and its variations. Kluwer, Dordrecht, pp 369–443

Kohonen T (2001) Self-organization maps and associative memory, 3rd edn. Springer, Berlin

Krasnogor N (2005) A tutorial for competent memetic algorithms: model, taxonomy, and design issues. IEEE Trans Evol Comput 9(5):474–488

Kytöjoki J, Nuortio T, Bräysy O, Gendreau M (2007) An efficient variable neighborhood search heuristic for very large scale vehicle routing problems. Comput Oper Res 34(9):2743–2757

Laporte G, Gendreau M, Potvin JY, Semet F (2000) Classical and modern heuristics for the vehicle routing problem. Int Trans Oper Res 7:285–300

Li F, Golden B, Wasil E (2005) Very large-scale vehicle routing: new test problems, algorithms, and results. Comput Oper Res 32:1165–1179

Matsuyama Y (1991) Self-organization via competition, cooperation and categorization applied to extended vehicle routing problems. In: Proc. of the International Joint Conference on Neural Networks. Seatle, WA, pp 385–390

Merz P, Freisleben B (2001) Memetic algorithms for the traveling salesman problem. Complex Syst 13(4):297–345

Merz P, Freisleben B (2002) Fitness landscape and memetic algorithm design. In: Glover F, Kochenberger G (eds) Handbook of metaheuristics. Kluwer, Norwell, MA, pp 321–353

Mester D, Bräysy O (2005) Active guided evolution strategies for large scale vehicle routing problems with time windows. Comput Oper Res 32:1593–1614

Moscato P (1999) Memetic algorithms: a short introduction. In: Corne D, Dorigo M, Glover F (eds) New ideas in optimization. McGraw-Hill, UK

Moscato P, Cotta C (2003) A gentle introduction to memetic algorithms. In: Glover F, Kochenberger G (eds) Handbook of metaheuristics. Kluwer, Boston, MA, pp 105–144

Modares A, Somhom S, Enkawa T (1999) A self-organizing neural network approach for multiple traveling salesman and vehicle routing problems. Int Trans Oper Res 6:591–606

Mühlenbein H (1991) Evolution in time and space—the parallel genetic algorithm. In: Rawlins G (ed) Foundations of genetic algorithms. Morgan Kaufmann, Los Altos, CA

Nguyen HD, Yoshihara I, Yamamori K, Yasunaga M (2007) Implementation of an effective hybrid GA for large-scale traveling salesman problems. IEEE Trans Syst Man Cybern B 37(1):92–99

Ong YS, Lim MH, Zhu N, Wong KW (2006) Classification of adaptive memetic algorithms: a comparative study. IEEE Trans Syst Man Cybern B 36(1):141–152

Or I (1976) Traveling salesman-type combinatorial problems and their relation to the logistics of blood banking. Ph.D. thesis, Northwestern University, Evanston, USA

Prins C (2004) A simple and effective evolutionary algorithm for the vehicle routing problem. Comput Oper Res 31:1985–2002

Preparata FP, Shamos MI (1985) Computational geometry: an introduction. Springer, New York

Rardin RL, Uzsoy R (2001) Experimental evaluation of heuristic optimization algorithms: a tutorial. J Heuristics 7:261–304

Reimann M, Doerner K, Hartl RF (2004) D-ants: savings based ants divide and conquer the vehicle routing problem. Comput Oper Res 31:563–591

Reinelt G (1991) TSPLIB-A traveling salesman problem library. ORSA J Comput 3:376–384

Rochat Y, Taillard ED (1995) Probabilistic diversification and intensification in local search for vehicle routing. J Heuristics 1:147–167

Schumann M, Retzko R (1995) Self-organizing maps for vehicle routing problems minimizing an explicit cost function. In: Proc. of the International Conference on Artificial Neural Networks, Paris, pp 401–406

Schwardt M, Dethloff J (2005) Solving a continuous location-routing problem by use of a self-organizing map. Int J Phys Distrib Logistics Manage 35(6):390–408

Solomon MM (1987) Algorithms for the vehicle routing, and scheduling problems with time window constrains. Oper Res 35:254–264

Smith K (1996) An argument for abandoning the traveling salesman problem as a neural network benchmark. IEEE Trans Neural Networks 7(6):1542–1544

Taillard ED, Gambardella ML, Gendreau M, Potvin JY (2001) Adaptive memory programming: a unified view of metaheuristics. Eur J Oper Res 135:1–16

Toth P, Vigo D (2001) Branch-and-bound algorithms for the capacitated VRP. In: Toth P, Vigo D (eds) The vehicle routing problem. SIAM, Philadelphia, pp 29–52

Toth P, Vigo D (2003) The granular tabu search and its application to the vehicle routing problem. INFORMS J Comput 15:333–348

Vakhutinsky AI, Golden BL (1994) Solving vehicle routing problems using elastic net. In: IEEE International Conference on Neural Network, pp 4535–4540