Self-Organizing Maps in Evolutionary Approach for the Vehicle Routing Problem with Time Windows

Jean-Charles Créput, Abder Koukam and Amir Hajjam

Systems and Transportation Laboratory University of Technology of Belfort-Montbéliard
90010 Belfort Cedex, France

Summary
The article presents the memetic SOM, an evolutionary algorithm embedding self-organizing maps as operators to address the vehicle routing problem with time windows (VRPTW). We show that it allows to extend the self-organizing map to deal with a version of the vehicle routing problem with time windows where the number of vehicles is an input, and by adding some walking distance from customers to bus stops. Then, we derived solutions for the classical VRPTW with no walking distances. This is illustrated on Solomon's standard test problems.

Key Words :
Evolutionary algorithm, self-organizing map, vehicle routing problem with time-windows

1 Introduction

In the literature, many applications of neural networks have addressed the Euclidean traveling salesman problem (TSP) [5][11]. Mainly, two types of neural networks are applied to the TSP. They are the Hopfield model [20], which is considered performing poorly on large instances, and the self-organizing map (SOM) approach [12], similar to elastic nets [6], that can tackle large instances. When applied in the plane, SOM is a visual pattern that adapts and modifies its shape according to some underlying distribution. Application of SOM to more complex and abstract vehicle routing problems remains a difficult task. For example, SOM has been extended to address the vehicle routing problem (VRP) [8][9][14][15][19][23] but, as far as we know, it has not been applied yet to the vehicle routing problem with time windows (VRPTW) [21].

In practice, hybridization of optimization methods is often compelling. It is a common and promising practice using a population based metaheuristic incorporating a neighborhood search. Examples of such methods are multi-start or restart approaches [1][18], as well as memetic algorithms [16] or genetic local search [17]. Memetic algorithms are special case of evolutionary algorithms (EA), combining advantage of heuristics within population based search. All the SOM applications to TSP and VRP are based on a modification of the SOM internal learning law. Here, to improve quality of results on the TSP and extend SOM to address the VRPTW, that is new, it is the context of the SOM application rather than its internal learning rule which is modified.

The standard and original SOM, as defined in [12][13] since more than two decades, is now a main operator embedded into an EA framework and then combined with other specific greedy operators, fitness evaluation and selection operators. Following the memetic algorithm structure, we present the memetic SOM, an evolutionary algorithm embedding self-organizing maps as internal operators. Furthermore, a specific version of the SOM is derived to tackle time windows. We present the approach and apply it to solve standard VRPTW test problems of Solomon [21], we discuss results by comparison to classical heuristics for the VRPTW.

The paper is organized as follows. Section 2 presents the problem. Objectives and constraints are given. Section 3 presents the standard SOM algorithm. Section 4 describes the memetic SOM approach. Section 5 presents experiments carried out on the VRPTW. The last section is devoted to the conclusion and further research.

2 Problem statement

2.1 Basic concepts

1) Requests. We denote by $V = \{r_1,...,r_n\}$ the finite set of customer demands, called requests. Each request $r_i \in V$ has a location in the Euclidean plane. It has a non-negative demand q_i, a service time s_i and a time window (a_i, b_i). If a vehicle arrives at a location where request r_i is intended to be served by the vehicle, this can not begin the service before a_i. The vehicle has to arrive before b_i. Service is done with service time s_i.

2) Transport routes. Let $B = \{n_1, ..., n_k\}$ being a finite set of cluster centers, also called transport points or bus-stops, localized by their coordinates in the plane. A transport mesh is a set routes. Formally, it is a collection $R = \{R_1, ..., R_m\}$ of m routes, where each route is a sequence $R_i = (n_0^i, ..., n_j^i, ..., n_{k_i}^i)$, $n_j^i \in B$, of k_i+1 successive cluster centers.

3) Request assignment. In our approach, the main difference with classical vehicle routing modeling is that routes are defined by an ordering of cluster centers, rather than by an ordering of customer requests. To build this ordering, each request *r* must be assigned to a single cluster center $n_r \in B$ in one of the *m* routes. A single vehicle is associated to each route. Then, routes are identified with vehicles that follow the routes. C_{R_i} is the capacity of a vehicle associated with route R_i, and L_i its load (sum of request quantities assigned to that vehicles). We denote by $t_{arr}(n_r)$ the time of arrival of the vehicle to point n_r for each request $r \in V$.

2.2 Euclidean Vehicle clustering and routing problem with time windows

The problem is presented as a clustering version of the vehicle routing problem with time windows. It is stated as follows:

Euclidean vehicle clustering and routing problem with time windows (clustering VRPTW). The problem input is given by a set of requests $V = \{r_1, ..., r_n\}$ and a set of interconnected routes $R = \{R_1, ..., R_m\}$. Using notations and definitions of section 2.1, the problem consists of finding cluster center locations, except for some fixed transport points at a depot location, and assignment of requests to cluster centers in routes, in order to minimize the following objectives:

$$length = \sum_{i=1,..,m,\ j=0,...,k_i-1} d\left(n^i_j, n^i_{j+1}\right), \quad (1)$$

$$distortion = \sum_{i=1,...,n} d\left(r_i, n_{r_i}\right), \quad (2)$$

subject to the capacity constraint:

$$L_i \le C_{R_i},\ i \in \{1,...,m\}, \quad (3)$$

and time-window constraint:

$$\min_{r_i \in V}\left(b_i - t_{arr}\left(n_{r_i}\right)\right) \ge 0. \quad (4)$$

The *length* value of objective (1) is the routes total length. The *distortion* value of objective (2) is the sum of distances from request locations to their assigned bus-stops, it is called distortion measure. The problem can be seen as a combination of a classical vehicle routing problem [4] with the well-known Euclidean k-median problem [1], adding time windows. Solutions with *distortion* = 0 and where *length* is minimum are solutions of a classical Euclidean VRPTW where the vehicle number is an input.

3 Kohonen's self-organizing maps

The self-organizing map is a non directed graph $G = (N, E)$ where each vertex $n \in N$ is a neuron having a synaptic weight vector $w_n = (x, y) \in \Re^2$, where $\Re^2$ is the two-dimensional Euclidean space. Synaptic weight vector corresponds to the vertex location in the plane.

```
Randomly generate weight vertices.
Repeat t_max times.
  Randomly extract a point p from the data distribution.
  Perform competition to select the winner neuron n* according to p.
  Apply learning rule to move the neurons of a neighborhood of n*.
  Decrease learning rate α and radius σ of neighborhood.
End Repeat.
```

Fig. 1. Self-organizing map on-line algorithm.

The set of neurons *N* 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 is summarized in Fig.1. A basic iteration follows three 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 closest neuron to $p(t)$. Finally, the following learning rule (triggering step):

$$w_n(t+1) = w_n(t) + \alpha(t).h_t(n^*, n).\left(p(t) - w_n(t)\right), \quad (5)$$

is applied to n^* and to all neurons within a finite neighborhood of n^* of radius σ_t, in the sense of the topological distance d_G, using learning rate $\alpha(t)$ and function profile h_t. The function profile is given by the Gaussian:

$$h_t(n^*, n) = \exp\left(-d_G(n^*, n)^2 \Big/ \sigma_t^2\right). \quad (6)$$

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

Application of SOM to a set of routes

$R = \{R_1, ..., R_m\}$ consists of applying iterations on the undirected graph $G_R = (N, E)$ induced by R. Vertex set N is the set of cluster centers, whereas E is the set of edges composed of any two successive centers from routes. The data set is the request set. SOM has two main properties of topology and density preservation [12]. This explains why it naturally addresses length (1) and distortion (2) minimization. In our evolutionary algorithm, a SOM simulation becomes an operator specified by its running parameters $(\alpha_{init}, \alpha_{final}, \sigma_{init}, \sigma_{final}, t_{\max})$. Furthermore, basing upon algorithm structure of Fig.1. specific greedy operators are derived to address the time window constraint.

4 The generic evolutionary approach

4.1 Generic evolutionary loop and operators

```
Initialize population with Pop individuals.
Do while not Gen generations are performed.
    - Apply one or more standard SOM operators (named SOM).
    - Apply operator for request assignment and fitness evaluation, chosen
from set { FITNESSTW1, FITNESSTW2 }.
    - Memorize best individual.
    - Apply selection operators, chosen from set {SELECT, SELECT_ELIT}, that
replace worst individuals by best ones.
    - Apply SOM derived operators, chosen from set {SOMTW1, SOMTW2}.
End do.
Report best individual encountered.
```

Fig. 2. The generic memetic loop.

The evolutionary loop structure is presented in Fig.2. This structure will be instantiated for the clustering VRPTW. A construction loop as well as an improvement loop are instantiated. A master loop controls execution of the construction phase followed by the improvement phase. Details of instantiations, as the parameter values of operators and their order of application, will be given in experiment section. As usual, one individual represents exactly one solution, that is, a set of routes. The generic memetic loop applies a set of operators to a population of individuals. At each generation, a predefined number of basic iterations are performed letting the SOM decreasing run being interrupted and combined with application of other operators, as fitness evaluation and requests assignment, selection and specific derived versions of SOM. At each generation, each operator is applied with probability *prob*. Three parameters *start*, *max* and *free* are possibly used for synchronization purpose. They mean that an operator starts its execution at generation *start*, and that it is applied alternatively during *max* successive generations followed by *free* successive generations where it is inhibited, and so on indefinitely. Details of operators are the followings:

1) Self-organizing map operator. It is the standard SOM applied to the graph network defined by the routes. It is denoted by its name and its internal parameters, as $SOM(\alpha_{init}, \alpha_{final}, \sigma_{init}, \sigma_{final}, t_{\max})$. One or more instances of the operator can be combined with their own parameter values. A SOM operator is executed performing *niter* basic iterations by individual, at each generation. Parameter t_{max} is the total amount of iterations applied to all individuals. It defines a decreasing run possibly performed in several generations. Once t_{max} basic iterations are performed, the operator resets to its initial parameter values and starts again. The SOM operator naturally addresses a minimization of objectives (1) and (2).

2) SOM derived operators. While the capacity constraint (3) is considered specifically in the fitness/assignment operator below, two problem specific operators are derived from the SOM algorithm structure for dealing with the time window constraint. The operator, denoted $SOMTW_1$, performs a greedy insertion move. Given a request that is not already assigned to some vehicle, the competitive step selects to be the winner the closest vehicle transport point encountered for which the time window constraint is satisfied, letting time window constraints of the other inserted requests in the route also satisfied. A second operator, denoted $SOMTW_2$, selects the route with the minimum increase of travel time as a supplementary condition. The evaluation of the travel time and the verification of the time window constraints are done after moving the transport point on the request location and including the request into the route.

3) Fitness/assignment operator. This operator greedily assigns requests to their closest cluster center, vehicle capacity constraint being satisfied. Here, cluster size is limited to a single request. Thus, the capacity constraint (3) is greedily tackled thru the assignment of requests. Then, the operator evaluates a scalar fitness value that has to be maximized and which is used by the selection operator. During this fitness

evaluation, requests that have their time window constraint violated are removed form the assignment. Objectives of *length* (1) and *distortion* (2) minimization

Table 1. Construction loop $CONFIGTW_1$ for the clustering VRPTW.

$CONFIGTW_1$ (*Size* = 50; *Gen* = 5)									
Operator		Internal SOM parameters				Evolutionary parameters			
		α_{init}	α *final*	σ_{init}	σ *final*	$t_{\max}$	*prob*	*niter*	*replace*
1	*SOM*	1.0	0.5	20.0	0.5	25000	1.0	100	-
2	$FITNESSTW_1$	-	-	-	-	-	1.0	-	-
3	*SELECT*	-	-	-	-	-	1.0	-	1
4	*SEL_ELIT*	-	-	-	-	-	0.1	-	1

Table 2. Improvement loop $CONFIGTW_2$ for the clustering VRPTW.

$CONFIGTW_2$ (*Size* = 50; *Gen* = 50)											
Operator		Internal SOM parameters				Evolutionary parameters					
		α_{init}	α *final*	σ_{init}	σ *final*	$t_{\max}$	*prob*	*niter*	*replace*	*max*	*free*
1	*SOM*	0.9	0.1	6	2	500000	1.0	100	-	5	5
2	$FITNESSTW_2$	-	-	-	-	-	1.0	-	-	-	
3	*SELECT*	-	-	-	-	-	0.5	-	1	-	-
4	*SEL_ELIT*	-	-	-	-	-	0.01	-	1	-	-
5	$SOMTW_1$	1.0	1.0	0.5	0.5	-	1.0	5	-	-	-
6	$SOMTW_2$	1.0	1.0	0.5	0.5	-	1.0	5	-	-	-

and time window constraint (4) are traduced and combined into the scalar fitness value. The fitness/assignment operators are the followings:

- $FITNESSTW_1$: it is a version for the clustering VRPTW specific to the construction phase. The fitness value is *fitness* = *sat*, where *sat* is the number of requests that are successfully assigned to routes, i.e. for which constraints are satisfied. The value *sat* is then considered as a main objective and admissible solutions are the ones for which *sat* = *n*, *n* being the request number.

- $FITNESSTW_2$: it is a version for the clustering VRPTW specific to the improvement phase. The fitness value is *fitness* = *sat* - $10^{-4} \times$ *length*, in order to consider the number of satisfied requests *sat* (defined above) as a main objective and *length* as a secondary objective.

4) Selection operators. Based on fitness maximization, the operator denoted *SELECT* replaces *replace* worst individuals, which 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 run.

4.2 Master loop

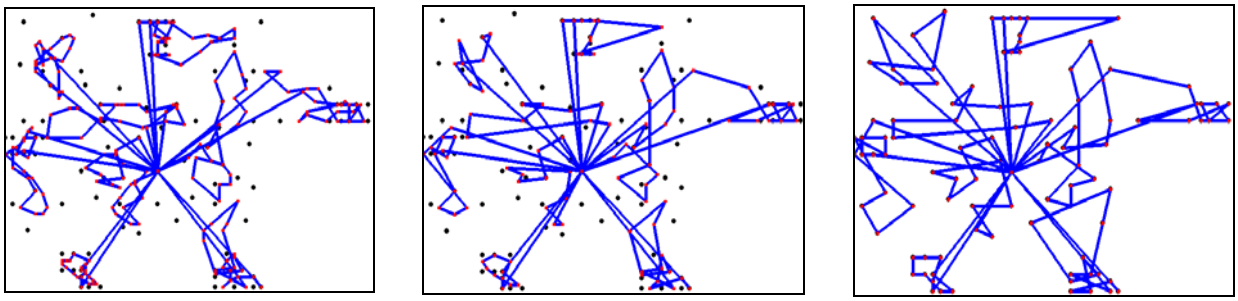
To implement a construction phase followed by an improvement phase, a master loop executes sequentially two memetic loops to respectively implement the construction and improvement phases. Once an execution terminated, the master loop performs population movements. The process of construction followed by improvement can be repeated several times starting from previously constructed solutions. Cluster centers coordinates are randomly generated into a squared area containing requests before execution.

5 Experimental results on the VRPTW

5.1 Algorithm configuration and parameters

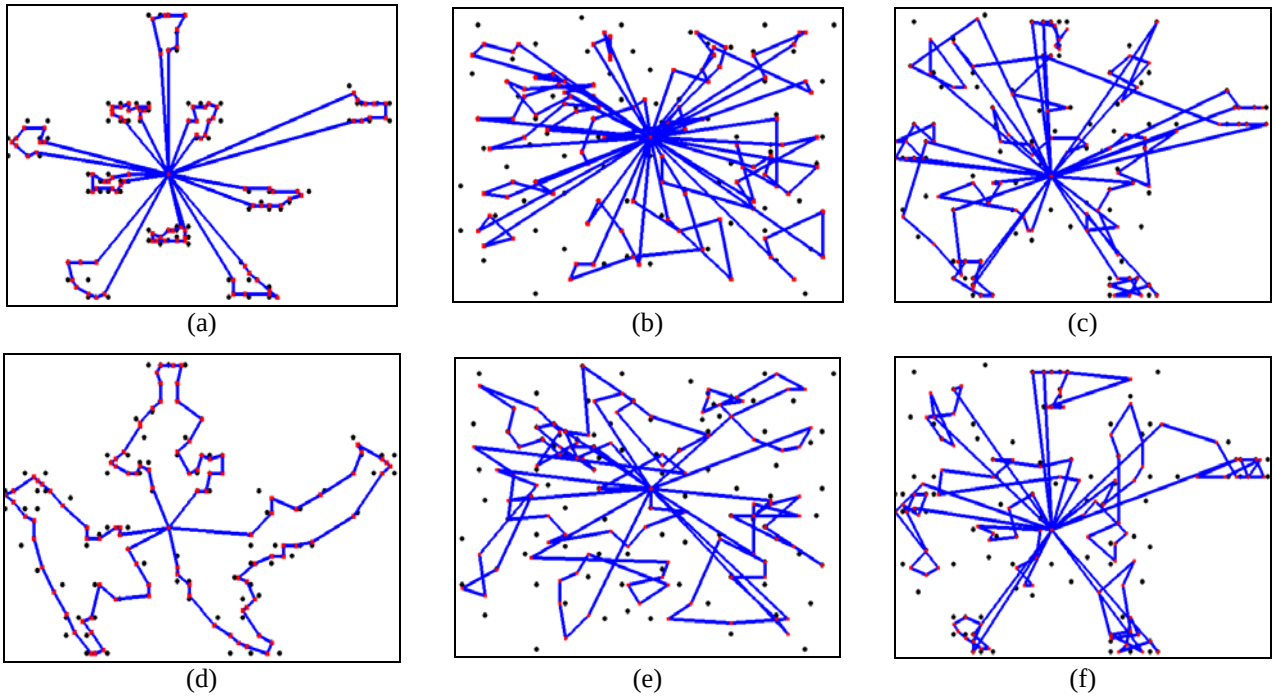
The algorithm consists of applying a construction phase followed by an improvement phase several times. Operators previously explained in section 4 are instantiated as in Tables 1-2. The number of neurons, or cluster centers, per route was set to 2*n* divided by the number of vehicle/routes. The construction loop $CONFIGTW_1$ of Table 1 generates initial solutions for the improvement loop. In experiments, such solutions present roughly 70 % of time-window constraints satisfied. The second configuration $CONFIGTW_2$ of Table 2 uses the two derived SOM operator, denoted $SOMTW_1$ and $SOMTW_2$, to perform greedy insertion

moves, whereas the standard *SOM* operator works on the transport routes to maintain topology and address length reduction. Starting at the beginning from random solutions are generated with some non null distortion (2), a classical VRPTW solution where *distortion* = 0 is derived at a final step. Fig.3(a-c) illustrates on the rc201 test case, the different steps to generate a classical VRPTW solution from an obtained solution with non null distortion. The deformable pattern generated by the algorithm is shown in (a). An intermediate result obtained by removing empty clusters is shown in (b). Then, a VRPTW solution as drawn in (c) is derived by projecting each cluster center to the location of its single assigned request.



(a) (b) (c)

Fig. 4. The steps to derive a classical VRPTW solution from a clustering VRPTW solution on the rc201 test case. (a) Obtained solution. (b) Same solution after removing the empty clusters. (c) A VRPTW solution obtained after projection of the cluster centers to their (single) assigned requests.



(a) (b) (c)

(d) (e) (f)

Fig. 3. (a)-(f) Solutions for six c101, r101, rc101, c201, r201, rc201 test cases.

solutions, the construction followed by improvement process is repeated 240 times using at each time the previously generated solutions.

5.2 Numerical results

We used the Solomon's standard VRPTW benchmark [21]. Problems are 100-customer Euclidean problems. The tests are embedded into a 100 km × 100 km geographic area, using vehicle speed of 60 km/h. Time-windows are given in minutes. Since here,

Visual patterns of the solutions obtained for six representative test cases c101, r101, rc101, c201, r201 and rc201 are shown respectively in Fig.(a)-(f). Empty clusters were removed. Table 3 presents numerical results for the six problems. Results are given for a

single run. The first column indicates the instance name and the number of vehicles. Second column presents the best known length value. Then, the number of satisfied requests, the total route length and the average distortion (*distortion* / *n)* are given in columns "sat", "length" and "avg. dist.", respectively for the obtained clustering VRPTW solutions and the derived standard VRPTW solutions. Percentage deviation to the best-known value is given within parenthesis. Here, the clustered c101 and c201 test cases are easy to solve and the algorithm finds near optimal solutions with less than 1% deviation from optimum. For the other cases, running the 120 master loop iterations, thus restarting internal coefficients many times, took approximately 5 mn on our PC 2000MHz. Deviation is less than 6 % above optimum for r101 and rc101 instances, whereas algorithm accuracy clearly diminishes with the r201 and rc201 cases which present a large horizon. Some time window lateness time is observed for few (non satisfied) requests.

Nevertheless, considering the clustering VRPTW results, lengths that are obtained are lower than the best known lengths for the classical VRPTW, while average distance from requests to their assigned vehicle cluster center (average distortion) is maintained within a narrow interval of less than 2 km. We can appreciate visually on Fig.3, how the Voronoï assignment takes place. Assignment of a request to the closest vehicle transport point is an important characteristic of the solutions generated, since a customer would like to go to its closest bus stop, and since otherwise, finding the right assignment would become a difficult problem by itself. We think that the approach leads to a new way of thinking about the VRPTW by generating underlying patterns that dispatch thru the requests, letting some place for dynamic adaptation to input modifications.

As usual with neural network applications to vehicle routing problems, the approach is far from being competitive with regards to the complex and powerful Operations Research heuristics specifically dedicated for the VRPTW, as for example the ones presented in [3][7][10][22]. For example, tabu search adaptive memory of Taillard et al. (1997) [22] produces, for a single instance where vehicle number is also an input, a solution of less than 1 % deviation to the best known within roughly less than 10 mn on average, and using a Sun Sparc 10 workstation. We are far from obtaining the same accuracy and execution time except on clustered instances of class C. But, simplicity (easy to understand) and flexibility (easy to extend) as well as their intrinsic parallelism are key points of neural networks and evolutionary algorithms that may lead in the future to computationally competitive applications for the VRPTW. Here, the results illustrate flexibility of using SOM into an evolutionary framework rather than trying to solely modify its internal law, but the two aspects may be combined in further research.

6 Conclusion

By incorporating SOM into an evolutionary algorithm, the approach extends and improves SOM based neural networks applications. This is illustrated in the paper by application to the vehicle routing problem with time windows, that is new. From the point of view of neural networks, the evolutionary framework introduces a level of supervision but it corresponds to a selection principle at a higher level with the aim to allow simplicity and flexibility and favor further parallel implantations. Operators have a similar structure based on closest point findings and simple moves performed in the Euclidean plane. In further research, improvement would be carried out by a finest parameter tuning and another combination of operators, as well as by the development of problem specific SOM learning laws. Exploiting the natural parallelism of the approach for multi-processor implantations is also a key point to address in further work.

Table 3. Results for six Solomon test cases.

	Best known	Clustering VRPTW solution			Derived VRPTW solution		
Instance - nb. of vehicles	length	sat	length	avg. dist.	sat	length	avg. dist.
c101 – 10	827.3*	101	774.6(-6.4%)	0.952	101	828.94(+0.2%)	0.0
r101 – 19	1650.80	101	1564.02(-5.3%)	1.733	98	1723.95(+4.4%)	0.0
rc101 – 14	1696.94	101	1683.96(-0.8%)	1.408	97	1793.23(+5.7%)	0.0
c201 – 3	589.1*	101	506.5(-14%)	1.323	101	595.12(+1%)	0.0
r201 – 8	1143.2*	101	1157.67(+1.3%)	1.928	100	1366.68(+19.5%)	0.0
rc201 – 9	1261.8*	101	1297.67(+2.8%)	1.948	101	1502.83(+19.1%)	0.0

* Optimum

References

[1] S. Arora, "Polynomial Time Approximation Schemes for Euclidean k-medians and related problems," *in ACM STOC'98*, 1998.

[2] K. D. Boese, A. B. Kahng, and S. Muddu, "Adaptive Multi-Start Technique for Combinatorial Global Optimization," *Journal of*

Operations Research Letters, vol. 16, pp. 101-113, 1994.

[3] O. Bräysy, W. Dullaert, and M. Gendreau, "Evolutionary Algorithms for the Vehicle Routing Problem with Time Windows," *Journal of Heuristics*, vol. 10, no 6, pp. 587-611, 2004.

[4] N. Christofides, A. Mingozzi, and P. Toth, "The vehicle routing problem," *Combinatorial Optimization*, Christofides N. et al. (eds), Wiley, 315-338, 1979.

[5] E. M. Cochrane and J. E. Beasley, "The co-adaptive neural network approach to the Euclidean Travelling Salesman Problem," *Neural Networks*, vol. 16, pp.1499-1525, 2003.

[6] R. Durbin and D. J. Willshaw, "An Analogue Approach to the Traveling Salesman problem using an Elastic Net Method," *Nature*, vol. 326, pp 689-691, 1987.

[7] L. M Gambardella, E. Taillard, and G Agazzi, "MACS-VRPTW: A Multiple Ant Colony System for Vehicle Routing Problems with Time Windows," *in New Ideas in Optimization*, D. Corne, M. Dorigo and F. Glover, editors, McGraw-Hill, UK, pp. 63-76, 1999.

[8] H. Ghaziri, "Supervision in the Self-Organizing Feature Map: Application to the Vehicle Routing Problem," *in Meta-Heuristics: Theory & Applications*, I.H. Osman and J.P. Kelly (eds), Kluwer, Boston, pp. 651-660, 1996.

[9] L. C. T. Gomes and F. J. A. Von Zuben, "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, 2002.

[10] J. Homberger and H. Gehring, "Two evolutionary metaheuristics for the vehicle routing problem with time windows," *INFOR*, 37:297‐318, 1999.

[11] D.S. Johnson and L.A. McGeoch, "The Traveling Salesman Problem: A Case Study in Local Optimization," *In Local Search in Combinatorial Optimization*, E. H. L. Aarts and J. K. Lenstra (eds.), John Wiley and Sons, London, pp. 215-310, 1997.

[12] T. Kohonen, *Self-Organization Maps and associative memory*, Springer Verlag, Berlin, 3rd edition, 2001.

[13] T. Kohonen, "Clustering, Taxonomy, and Topological Maps of Patterns," *Proceedings of the 6th International Conference on Pattern Recognition*, 1982.

[14] Y. Matsuyama, "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, 1991.

[15] A. Modares, S. Somhom, and T. Enkawa, "A self-organizing neural network approach for multiple traveling salesman and vehicle routing problems," *International Transactions in Operational Research*, vol. 6, pp. 591-606, 1999.

[16] P. Moscato, "Memetic Algorithms: A Short Introduction," *in New Ideas in Optimization*, D. Corne, M. Dorigo, and F. Glover, eds., McGraw Hill, 1999.

[17] H. Mühlenbein, "Evolution in Time and Space – The Parallel Genetic Algorithm," *in G. Rawlins (Ed.), Foundations of Genetic Algorithms*, Morgan Kaufmann, Los Altos, CA, 1991.

[18] M.G.C. Resende and C.C. Ribeiro, "Greedy randomized adaptive search procedures," *In "State of the Art Handbook in Metaheuristics*, F. Glover and G. Kochenberger, eds., Kluwer, 2002.

[19] 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*, Paris, pp. 401-406, 1995.

[20] K. A. Smith, "Neural Networks for Combinatorial Optimization: A Review of More Than a Decade of Research," *INFORMS Journal on Computing*, vol. 11, no. 1, 1999.

[21] M.M. Solomon, "Algorithms for the vehicle routing, and scheduling problems with time window constrains," *Operations Research*, vol. 35, pp. 254–264, 1987.

[22] E.D. Taillard, P. Badeau, M. Gendreau, F. Guertin, and J.Y. Potvin, "A tabu search heuristic for the vehicle routing problem with soft time windows," *Transportation science*, vol. 31, pp. 170-186, 1997.

[23] A. I. Vakhutinsky and B.L. Golden, "Solving vehicle routing problems using elastic net," *in IEEE International Conference on Neural Network*, pp. 4535-4540, 1994.

Jean-Charles Créput received his Ph.D. in Computer Science from University of Paris 6, France, in 1997. He is currently an associate professor in Computer Sciences and Engineering at the University of Technology of Belfort-Montbeliard, France, and performs research activity at the Systems and Transportation (SeT) Laboratory. His current research interest include evolutionary algorithms and neural networks applied to telecommunications and intelligent transportation services.

Abder Koukam is professor of Computer Science at the University of Technology of Belfort-Montbeliard, France. He received the Ph.D. in Computer Science from University of Nancy I, France, in 1990. He heads research activities at Systems and Transportation (SeT) Laboratory on modelling and analysis of complex systems, including software engineering, multi-agent systems and optimization.

Amir Hajjam received his Ph.D. in Computer Science from University of Haute-Alsace, France, in 1990. He is currently an associate professor in Computer Sciences and Engineering at the University of Technology of Belfort-Montbeliard, France, and performs research activity at the Systems and Transportation (SeT) Laboratory. His current research interest include evolutionary algorithms and neural networks applied to networks, telecommunications and intelligent transportation services.