Approach for optimized and dynamic medical emergency management

Amir Hajjam, Jean-Charles Creput and Abder Koukam
Systems and Transportation Laboratory, University of Technology of Belfort-Montbliard, 90010 Belfort Cedex, France
amir.hajjam@utbm.fr

Abstract—**This paper describes a new approach aiming at optimizing and improving the efficiency of the medical emergency services. The purpose is to optimize the use of human and material resources by assigning the appropriate doctor to each patient and as a result reduce costs and time reactivity. In order to do so, we tackled the problem as a Dynamic Vehicle Routing Problem within a Timeframe.**
We have developed a multi agent simulator which includes an optimization module that is able to solve vehicle routing problems. We consider our approach as original since it consists of mixing artificial intelligence methods with classical heuristics. We consider it also as result oriented. Based on the Poisson process, we are able to simulate real situations and handle requests as soon as they are entered.

Index Terms— **Evolutionary algorithm, self-organizing map, heuristics and vehicle routing.**

I. Introduction

Nowadays, Telemedicine is more and more practiced in the state of art medicine. It is beneficial for populations living in isolated communities and remote regions and it is currently being applied in virtually all medical domains (surgery, blood vessels navigation...). It involves acquiring the appropriate medical equipment. Nevertheless, there is currently a lack of medical equipment and technology for the day-to daymedical acts in this domain.

In order to help medical emergency services, the project MERCURE (Mobile and Network for the Private clinic, the Urgency or the External Residence) has been launched to create tools that optimize, follow and manage emergency interventions. The current problem is that the choice of the doctor for a patient is done by hand. The call center is not aware of the exact location nor the current state of the doctors. Thus it is rarely the best located doctor who is chosen and moreover he may not have correct equipments to heal the patient. To optimize that aspect, we have developed software allowing the optimized management of human and material medical resources.

The remainder of this paper is organized as follows. First we will present the MERCURE project. In the second section we shall introduce our optimization system with implementation details. Our optimization approach is explained in the third section. Computational results are reported in section 4 and section 5 concludes the paper.

II. Project *MERCURE*

A. Aim of the project

The project MERCURE takes part of the French pole of competitiveness therapeutic innovations. The aim of the project is to give, thanks to information technologies, an optimized and dynamic management of resources used in the scope of doctor interventions like material and human resources. The system gives a real-time tracking of current interventions, from the reception of the call to the closure of the medical record.
It optimizes resources, travel times and takes care of whole constraints relative to the domain: emergency level, pathology, medical competences, location and other specific aspects related to this profession.

The platform exploits satellite location system associated with a geographical information system (GIS) and is based on results coming from works on vehicle routing problems [7]. With present technologies we can have accurate current location of patients and doctors via GIS and A-GPS[1] respectively. The A-GPS system has 3 main uses.

1) Know the position of each medical team.
2) Help the doctor to reach quickly the intervention point.
3) Track in real-time medical teams and resources.

Here is a basic scenario when an emergency call arrives.

Call center point of view:

- Information about the patient (name, address, pathology…) are inputted in the software.
- Patient's information are processed, a set of doctors which suit to the patient's needs is created (depending of the pathology, the intervention area…).
- The selection of a doctor in the previously created set is done via an optimization algorithm. Here we focus on optimizing several criteria like distance, reaction time. . . The patient is inserted in the doctor's road.
- The selected doctor is warned by a message on his PDA[2].

Now from a doctor point of view:

1 Assisted Global Positioning System
2 Personal Digital Assistant

- The doctor receives a patient request on his PDA and he is geo-guided to the patient's location via A-GPS.

- As soon as he arrives, all information about the patient is shown: previous diseases, his allergy, current treatments. . . This information is transferred from the database via radio link like GPRS[3] or UMTS[4] for example.
- When the auscultation is finished, he inputs results and notes which are immediately transferred to the central database. Then he goes on with the next patient.

B. Improvements

This whole process improves reaction time of emergency services and thus save lives. It also provides a unique database gathering up-to-date information about patients and so facilitates the follow-up of patients. Another main improvement is that the answer fits to the patient's needs. In other words, the call is answered by a doctor-regulator who is able to help the patient to describe and specify his illness. This is a real telemedicine act and thus the software is able to select the appropriate doctor or send an ambulance. Moreover this system may suit to other emergency services like fire brigade or police department with some adaptations. There are some papers about ambulances location and relocation models written by Gendreau and al.

[10][9][11] and Brotcorne and al. [4]. But currently we are not aware of other tools for such size of emergency services. This project is realizable thanks to recent new technologies like A-GPS, wireless data communication and improvements in artificial intelligence and operations research for dynamic problems.

III. Dynamic Optimization System for doctors

In the MERCURE project, we are in charge of the optimization part for the selection of doctors and assignment of patients. We have tackled this problem as an operations research problem named Vehicle Routing Problem (VRP).

A. Problem statement

Allan Larsen said in his PhD report [16] that emergency services have 2 major criteria:

1) They are highly dynamic: most or all requests are unknown at the beginning and we don't know their arrival time.
2) The response time must be very low because lives can be in danger.

That is why we have chosen to represent the emergency problem as a Dynamic Vehicle Routing Problem with Time Window (DVRPTW) which is presented in III-A.1. This extension of the well known VRP suits very well to this kind of problem because it takes care of the 2 criteria previously stated. Time windows are perfect to consider response time and the dynamic aspect allows the system to receive requests during the optimization process

1) DVRPTW presentation: A Dynamic Vehicle Routing Problem with Time Windows is a specialization of the well known Vehicle Routing Problem where a fleet of M identical vehicles supplies goods to N customers. All vehicles have the same capacity Q, and for each customer i, $i = 1 \dots N$, the demand of goods q_i and the service time s_i to meet the demand in i are known. The parameter s_i represents the loading or unloading service time at the customer i. Each customer must be visited once, and all vehicles routes start and finish at the central depot. It is a NP-hard problem (see figure 1).

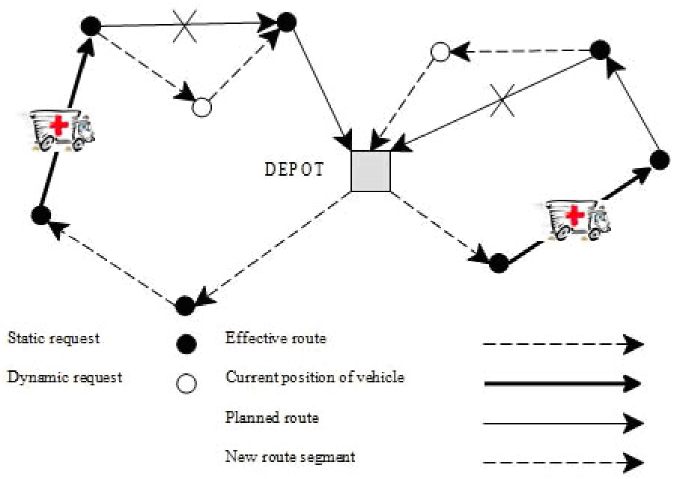

Fig. 1. Example of dynamic vehicle routing problem with 7 static requests and 2 immediate requests.

Between a VRP and a DVRPTW, 2 constraints are added:

1) Usually, the relevant information, like new patients and cancelled requests can occur all the time, even after the optimization process start. The dynamism consists in receiving several requests during the evolution of the simulation. These dynamic variations can be very important to really reduce the costs in vehicle routing problems. The date when the request i arrives is noted g_i as the generation date of request i.

2) The time windows constraint which consists in having 2 time limits associated with each requests I: $[a_i, b_i]$. The vehicle must start the customer service before b_i, but if any of them arrives at customer i before a_i, it must wait. So the smaller the time window of a request is the harder will it be to find a good insertion place in a vehicle road.

To these 2 constraints, a third one can be added depending of the instance of the problem. It comes from the fact that all doctors may not start from the same location so we must manage multi-depots instances of DVRPTW.

3 General Packet Radio Service
4 Universal Mobile Telecommunications System

3) Matching to DVRPTW: We have to affect each real entity (call center, resources and patients) to one in routing problem which are vehicles, requests and the company. The

most logical way to make them correspond is to match the doctors to the vehicles, the patients to the requests and the call center to the company.

But there is some specificity that we must consider in the problem.

First the patient may need a specialist for his illness. Not every vehicle can serve this request. It is the same thing for ambulances. In the same way, we must avoid sending a woman into a district with bad reputation. We need to have in our application different types of vehicle which is not managed in classical VRP instances where all vehicles are identical. So in our DOS a request can be dedicated to a vehicle and only this vehicle can serve it.

We must also take care of the loading of the system. We have a time constraint that is specific to emergency services. In classical VRPTW, when some requests are not served at the end of the day they are deferred to the next day. Here when the system is overloaded, we must serve most urgent requests and redirect less urgent ones to a classical doctor if possible.

B. Dynamic Optimization System

In order to solve DVRPTW, we have developed a simulator that we have called Dynamic Optimization System (DOS).

1) Architecture of the simulator: This simulator is divided in 2 distinct parts.

On the one hand we have a multi-agents simulator. Its role is to schedule main entities present in a VRP. Each entity is represented by a process.

- The environment process is dedicated to generate events during the simulation.
- The company process simulates a real company. It receives requests and plans vehicles roads.
- Vehicles processes follow roads given by the company and serve requests.

All these processes are synchronized on a same clock own by the scheduler so they advance in time simultaneously. One simulation step lasts *To* milliseconds in real time and the corresponding simulation time depends on a ratio to suit the problem. As our application domain is in real time, the ratio will be 1. So each step will last *To* millisecond in the simulation.

During a step, each process is called once to make a short action and so share CPU time. Actions that need a lot of time must be divided in several shorter actions with small execution time.

One the other hand we have the optimization part. The optimization process can be viewed as a black box, receiving the current solution (a set of vehicles) and known requests and giving back a better solution if possible. This part of our DOS is explained more precisely in IV. It is the company that has the role of asking the optimizer to optimize current solution. After a fixed time, the company reads the solution and gives new plans to the vehicles or confirms the current one. The exchanges between the two parts are done via a letterbox with exclusive access. This ensures the data transfers between two unsynchronized threads and prevents data overriding.

2) Additional features: Through this system we can also gather lots of interesting information that we can process to extract some statistical data. We can imagine optimizing the number of doctors depending of the date, the specialization the most needed and so on. Once enough requests are stored in the database, we can extract main trends and optimize human and logistic resources.

Moreover we can use that probabilistic information on future events to route doctors to their next patient by making them pass close to area with high probability of new requests. Bertsimas and al. [3] describe this kind of problems and call them Probabilistic Vehicle Routing Problem (PVRP).

IV. Optimization approach

We are now facing a DVRPTW that we must solve relatively quickly in order to be able to warn doctors of a new patient to see urgently. In a VRP problem, to get one the best solution, it almost takes a lot of time. Here we prefer having a relatively good solution quickly and then improve it.

To do that, our optimizer has a 2-level architecture. The top level uses a global meta-heuristic strategy and controls several solver agents. In the lower level we can find previously mentioned solver agents who represent different heuristics for solving VRP.

A. Global Meta-heuristic

This level aims at finding the best solution by using several solver agents. Each of these agents represents a heuristic for solving VRP (see IV-B). That can be seen as a worker with different tools (the solver agents) at his disposal for doing his job, here optimizing vehicles routes. It has to choose the strategy which suits the best to the problem for example creating the first solution. To do that, the optimizer initializes a set of selected solver agents and tells them to do the job separately. Then before a defined generation time (*Tg*) it gathers all solutions from the agents and makes a selection to keep most interesting ones depending on the strategy and then gives the best solution to the company via the letterbox. So we manage a population of solution where we keep or replace individuals like in genetic algorithms. This allows to exchanging solutions between different heuristics and so discovers new ones and get out local optima.

The solver agents are scheduled by the optimizer like the processes in the simulation part. When all used solver agents have been activated once, one step is done. So we can have several different optimization methods in parallel. The specifics of our solver agents are approached in the next part.

B. Low-level heuristics

We shall now analyze the lower level where solver agents are located. Their aim is to solve a type of VRP thanks to a specified heuristic. Each agent uses one or more heuristics which can be very basic like a 2-opt which consists in exchanging 2 roads (see figure 2a) or more complicated like neural networks or other artificial intelligence methods. The optimizer is aware of features of all solver agents.

1*) Memetic SOM*: Our main optimization algorithm we are using is based on local search [19] and self- organizing maps (SOM) [14], [12], [17] by embedding them into an evolutionary algorithm. This approach is called memetic SOM [7] by reference to memetic algorithms [18]. Its aim is to allow the massive execution of elementary local operations in the Euclidean plane and so produce an emerging phenomenon. Most of theses operations are neighbourhood operations based on spiral search [2] in order to keep a complexity in O(n) time. The standard self-organizing maps is a non directed graph $G = (N, E)$, called the network, where each vertex $n \in N$ is a neuron having a synaptic weight vector $w_n = (x,y) \in R^2$, where R^2 is the two-dimensional Euclidean space. Synaptic weight vector corresponds to the vertex location in the plane.

The training procedure structure is made of a fixed amount of t_{max} iterations applied to a graph network. Each iteration follows three basic steps. At each iteration t, a point $p(t) \in R^2$ 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 $p(t)$. Finally, the following learning law (triggering step) is applied to $n*$ and to all neurons within a finite neighbourhood of $n*$ of radius σ_t, in the sense of the topological distance d_G:

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

where α(t) is the learning rate and $h(t)$ is the function profile given by the Gaussian:

$$h_t(n*, n) = exp(\frac{-d_g(n*, n)^2}{\sigma_t^2})$$

The evolutionary algorithm embedding SOM is based on memetic loop which applies a set of operators to a population of individuals, at each iteration (called a generation). The construction loop starts its execution with solutions having randomly generated vertex coordinates, into a rectangle area containing cities. The improvement loop starts with the single best previously constructed solution, which is duplicated in the new population. The main operator is the SOM algorithm applied to the graph network. At each generation, a predefined number of SOM basic iterations are performed letting the decreasing run being interrupted and combined with application of other operators, which can be other SOM operators with their own parameters, mapping and fitness evaluation, and selection. Each operator is applied with probability prob. Details of operators are the followings:

1) Self-organizing map operator. It is the standard SOM applied to the ring network. One or more instances of the operator can be combined with their own parameter values. A SOM operator is executed performing n_{iter} basic iterations by individual, at each generation.

2) SOM derived operators. Two problem specific operators are derived from the SOM algorithm structure for dealing with the VRP especially. The first, denoted SOM VRP, is like a standard SOM but restricted to be applied on a randomly chosen vehicle, using requests already assigned to that vehicle. While capacity constraint will be considered in the mapping operator below, a SOM based operator, denoted SOM DVRP, deals with the time duration constraint. It performs a greedy insertion move.

3) Fitness/assignment operator. This operator, denoted FITNESS, generates a VRP solution and modifies the shape of the ring accordingly. The operators greedily maps customers to their nearest neuron, considering only the neurons not already assigned to a customer, and where vehicle capacity constraint is satisfied. The capacity constraint is then greedily tackled through the requests assignment. Once the assignment of requests to routes has been performed, for each individual this operator evaluates a scalar fitness value that has to be maximized and which is used by the selection operator. Taking care of time duration constraint the fitness value is computed sequentially following routes one by one and removing a request from the route assignment if it leads to a violation of the time duration constraint.

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.

The memetic SOM is very interesting because of its adaptability and flexibility due to its neighbourhoods search capabilities and simple moves performed in the plane. We can easily add or remove requests without having to launch again an optimization from the beginning because they are immediately inserted at a good position. We are currently working on the integration of this algorithm in the optimization system.

2) *Classical optimizations*: Moreover we agentified several classical optimization heuristics to make them work in our multi-agents optimization architecture. We have chosen some intra-route and inter-route heuristics like 2-opt (see figure 2a) or 1-1 exchange (see figure 2b), to improve solutions obtained by memetic SOM and also explore new solutions by mutating some of them.

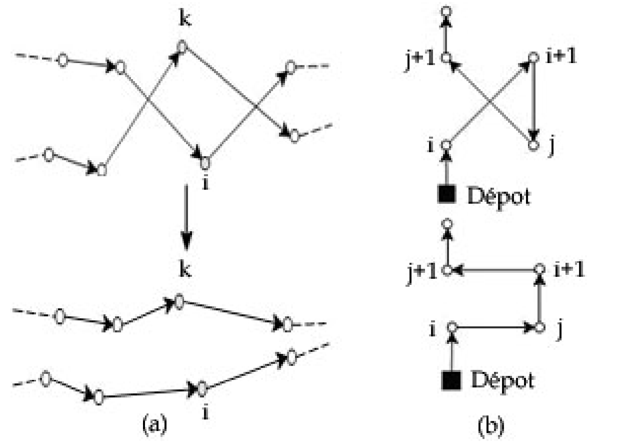

Fig. 2. (a) Example of 1-1 exchange move and (b) Example of a 2-opt operation

3) *Similar approach*: Our approach is similar to that of Kytjoki and al. [15] called variable neighbourhood search (VNS) where they create an initial solution by a cheapest insertion heuristic that is improved with a set of improvement heuristic. In a second phase they improve the solution with the same set of heuristic until there are no more improvements. With this approach they can solve very large scale VRP, up to 20,000 customers within reasonable CPU times. But their solution does not address time windows VRP and was not tested on dynamic sets.
We also think that the mixing of artificial intelligence approach with several operations research approaches can give better results than focusing on a unique one. That is why we have chosen such architecture for the optimization part to be able to add different method and see which ones go well together.

V. Experimentation

A. Computational results

We have currently only tested the Self-Organizing Maps approach on static VRP instances. The Table I reports the average results on all the 56 Solomon instances. The number of routes was set to the one of the best solution reported in the literature for each problem. Results are given on average for the six problem classes.

	Best known	VRPTW solution	
Problem set - nbr of vehicles	length	sat	length
C1 - 10	826.7	100.56	854.08(+3.32%)
R1 - 11.92	1209.89	99.83	1311.84 (+8.49%)
RC1 - 11.50	1384.16	97.63	1424.22 (+3.45%)
C2 - 3	587.69	98.88	606.43 (+3.19%)
R2 - 2.75	867.54	99.17	1187.97 (+23.88%)
RC2 - 3.25	1119.35	99.38	1428.92 (+26.8%)

TABLE I
Computational average results on all the 56 Solomon instances.

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. 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. That's why we want to improve results obtained by SOM with simple operation research heuristics to have a good quality ratio between solution quality and speed. That is on what we are currently working.

B. Dynamic benchmarks

In spite of we are not able to give results on DVRP yet; we have already defined what kind of benchmark we will use and how we will create benchmarks. Our aim is to be the nearest from realistic situations.

1) Existing benchmarks: We already have benchmarks for VRPTW as we give results in the previous part, but we did not find Dynamic VRPTW benchmarks. But we have found some Dynamic VRP benchmarks originally proposed by Kilby and al. [13], [1]. They are derived from some very popular static VRP benchmarks datasets, namely 12 problems are taken from Taillard [20], 7 problems are from Christofides and Beasley [6] and 2 problems are from Fisher and al. [8]. These problems range from 50 to 199 customers. The simulation starts at time 0 and end at time T. The number of customers can be inferred from the name of each instance. We notice that the benchmark data structure from [1] is very usefull because it is very flexible and allows all interesting feature like the use of several depots, time windows, requests generation dates. . .

All these DVRP instances are highly dynamic. It means that almost no requests are known at the beginning of the simulation. To express this information, Larsen [16] defined a ratio called degree of dynamism (*dod*) (equation 1). The ratios in the DVRP instances we have used have all a $dod = 1$ excepted two of them which have a $dod > 0.98$. This information is important because generally, the more dynamic a system is the more difficult or costly it is to generate a fast reaction.

The number of requests known in advance is denoted n_{adv} , the number of immediate requests is denoted n_{imm} and the total number of requests is denoted n_{tot} that is $n_{adv} + n_{imm}$.

$$dod = \frac{n_{imm}}{n_{tot}} \qquad (1)$$

And we can easily see that: $0 \leq edod \leq 1$

But this measurement does not take care of the arrival time of the dynamic requests. So two scenarios which have the same *dod* can be very different. It will be easier to solve the first scenario than the second one because we know requests earlier and so we can produce a better solution.

$$edod = \frac{\sum_{i=1}^{n_{imm}} (\frac{t_i}{T})}{n_{tot}} \quad (2)$$

And once again: $0 \leq edod \leq 1$

Each edod of previously stated Dynamic Benchmarks is around 0.5; this means that the distribution is relatively uniform. If all requests arrive at the beginning of the simulation, the edod will be very small (equation 3), otherwise the edod will approach 1 (equation 4).

$$\lim_{t_i \to 0 \ \forall i} edod = 0 \quad (3)$$

$$\lim_{t_i \to T \ \forall i} edod = 1 \quad (4)$$

2) Benchmark creation: As we have said before, DVRPTW fit the best to represent emergency services problems. But as far as we know, there are no such benchmarks. That is why we have chosen to transform static benchmarks to dynamic ones.
To do that, we have implemented in our simulator a tool that converts static VRP(TW) problems into a dynamic ones thanks to a Poisson Process [5]. Our aim is to generate dynamic requests sets from static sets with the generation date based on a inhomogeneous Poisson process by letting the intensity $\lambda(t)$ to vary in time where $\lambda(t)$ is a deterministic function of time. So we can take into account the requests fluctuation depending on time in order to be the nearest of real problems.

So by giving the dod and parameters for the Poisson process we can simulate realistic situations based on statistical data gathered near emergency services.

VI. Conclusion

The MERCURE project is helpfull for emergency services by giving them appropriate tools to do their job in better conditions. By representing medical emergency services by a Dynamic Vehicle Routing Problem with Time Windows, we are able to optimize human and material resources and so reduce costs, reaction time and maybe save lives.

To solve the problem we developed a multi-agents simulator that enables the real-time following of doctors thanks to GPS. The simulator is linked to an optimization part also based on agents. This part is divided in two levels. In the top level we have a global meta-heuristic which uses agents located in the lower level. Each of these agents represents a tool for solving the problem or a part of it. Tools that we have currently are based on operation research heuristics and on Self- Organizing Maps which is a kind of neural network. The mixture of classical heuristic with Self-Organizing Maps should give good results in little computing time and adaptability capacities. We are currently working on the dynamic aspect of the optimization. We have stated on how we manage dynamic requests. We are able to load dynamic requests benchmark but also generate dynamic requests from static benchmarks thanks to a Poisson process.

Another interesting aspect of our simulator is that it currently focuses on medical emergency services but it could be extended to address several kings of emergency services problems.

References

[1] APES. Dynamic vehicle routing problems instances. [Online]. Available: http://www.dcs.st-and.ac.uk/apes/apedata.html

[2] J. L. Bentley, B. W. Weide, , and A. C. Yao, “Optimal expected time algorithms for closest point problems,” in ACM Transactions on Mathematical Software, vol. 6, no. 4, pp. 563–580.

[3] D. Bertsimas, P. Jaillet, and A. R. Odoni, “A priori optimization,” in Operations Research, vol. 36, no. 6, 1990, pp. 1019–1033.

[4] L. Brotcorne, G. Laporte, and F. Semet, “Ambulance location and relocation models,” in European Journal of Operational research, 2003, vol. 147, pp. 451–463.

[5] T. C. Brown, “Poisson approximations and the definition of the poisson process,” in The American mathematical monthly. Washington, DC: Mathematical Association of America, 1984, vol. 91, pp. 116–123.

[6] N. Christofides and J. Beasley, “The period routing problem,” in Networks, 1984, vol. 14, no. 2, pp. 237–256.

[7] J.-C. Creput, A. Koukam, and A. Hajjam, “Self-organizing maps in evolutionary approach for the vehicle routing problem with time windows,” in International Journal of Computer Science and Network Security, 2007, vol. 7, no. 1, pp. 103–110.

[8] M. Fisher, R. Jakumar, and L. van Wassenhove, “A generalized assignment heuristic for vehicle routing,” in Networks, 1981, vol. 11, pp. 109–124.

[9] M. Gendreau, F. Guertin, J.-Y. Potvin, and E. Taillard, “Parallel tabu search for real-time vehicle routing and dispatching,” Transportation Science, 1999.

[10] M. Gendreau, G. Laporte, and F. Semet, “Solving an ambulance location model by tabu search,” in Location Science, 1997, vol. 5, no. 2, pp. 75–88.

[11] ——, “A dynamic model and parallel tabu search heuristic for real-time ambulance relocation,” in Parallel Computing, 2001, vol. 27, no. 12, pp. 1641–1653.

[12] H. Ghaziri, “Supervision in the self-organizing feature map: Application to the vehicle routing problem,” in Meta-Heuristics: Theory & Applications, I. Osman and J. Kelly, Eds., Boston, 1996, pp. 651–660.

[13] P. Kilby, P. Prosser, and P. Shaw, “Dynamic vrps: a study of scenarios,” University of Strathclyde,” Technical Report APES-06-1998, September 1998.

[14] T. Kohonen, in Self-Organization Maps and associative memory, 3rd ed., Springer, Ed., Berlin, 2001.

[15] J. Kyto¨ jokia, T. Nuortiob, O. Bra¨ysya, and M. Gendreauc, “An efficient variable neighborhood search heuristic for very large scale vehicle routing problems,” in Computers & Operations Research, 2007, vol. 34, pp. 2743–2757.

[16] A. Larsen, “The dynamic vehicle routing problem,” Ph.D. dissertation, Technical University of Denmark, Denmark, June 2000.

[17] A. Modares, S. Somhom, and T. Enkawa, “Self-organizing neural network approach for multiple traveling salesman and vehicle routing problems,” in International Transactions in Operational Research, 1999, vol. 6, pp. 591–606.

[18] P. Moscato, “Memetic algorithms: A short introduction,” in New Ideas in Optimisation, D. Corne, M. Dorigo, and F. Glover, Eds., McGraw Hill, 1999, ch. 14.

[19] Y. Rochat and E. D. Taillard, “Probabilistic diversification and intensification in local search for vehicle routing,” in Journal of Heuristics, 1995, vol. 1, pp. 147–167.

[20] E. Taillard, “Parallel iterative search methods for vehicle-routing problems,” in Networks, 1993, vol. 23, no. 8, pp. 661–673.