COMPETITIVE PROBABILISTIC SELF-ORGANIZING MAPS FOR ROUTING PROBLEMS

Hassan Ghaziri
AUB, OSB
Beirut, Lebanon
ghaziri@aub.edu.lb

Abstract – *In this paper, we have applied the concepts of the self-organizing map (SOM) algorithm, to a routing problem called the vehicle routing problem with backhauls (VRPB). Usually SOM is based on a single map to represent data. The main contribution of this paper is to introduce a new architecture formed of multiple competing maps and to design the corresponding learning algorithm such that the Kohonen maps can be applied to a family of routing problems. Those problems are known to be NP-Hard problems. We have applied these concepts to the vehicle routing problem (VRP) and to a variant of this problem called the VRP with Backhaul in which customers are of two sorts : linehaul and backhaul customers. In designing the routes for the various vehicles, linehaul customers should be visited first and then the backhaul customers. We benchmarked the performance of our approach with the most powerful meta-heuristics and obtained excellent results*.

Key words – Competitive self-organizing maps, Meta-heuristics, Vehicle routing problem,

1 Introduction

The VRPB problem we are considering in this paper consists of designing delivery routes to serve two types of customers; linehaul and backahul customers
The following information is given:

- **The Customers.** The position of the customers is provided through their coordinates in the Euclidean space. Along with their position, the weight or quantity of goods, of each customer and its type are also provided. There are two types, the backhaul and linehaul customers. A certain quantity of goods is delivered from the depot to the linehaul customers and a certain quantity of goods must be picked up from each backhaul customer and brought to the depot.
- **The Vehicles.** A fleet of homogeneous vehicles is given. It means that the number of vehicles is known and the capacity of all vehicles is the same. Each vehicle can not ship a quantity of goods that exceeds its capacity.
- **The Depot**. The location of the depot is known and its coordinates in the Euclidean space are given.

Each vehicle starts its route from the depot visits the linehaul customers assigned to it for delivery and then collects the corresponding goods from each backhaul customer and bring them back to the depot.

2 SOM for Routing Problems: a Brief Review

Artificial Neural networks were used to solve routing problems, namely the Traveling Salesman Problem (TSP). Hopfield (1995) network was among the first used to tackle this problem. In spite of its novelty, this approach was not practical and not competitive with exact methods and meta-heuristics. The applications were restricted to small sized problems not exceeding 20 to 50 cities.
The architecture that is usually used for routing problems and that was introduced by Fort (1988) is the ring architecture. It consists of a set of neurons placed on a deformable ring. The concept of tour is embedded in the ring architecture, since the location of a neuron on the ring can be identified with the position in the visit as shown in the figure 2. This is the most important advantage of this approach. The ring can be considered as a route for an ideal problem. The interaction of the network with its environment, here the customers and the adaptation of its neurons will force iteratively the ring to represent the real tour that will visit sequentially the customers.
From a practical point of view, the SOFM algorithm starts by specifying the architecture of the network, which consists of a one ring upon which the artificial neurons are spatially distributed. The ring is embedded in the Euclidian space. Each neuron is identified by its position in the Euclidian space and its position on the ring. The Euclidian distance will be used to compare the positions of neurons with the positions of cities. The *lateral* distance will be used to define the distance on the ring between two neurons. The lateral distance between any two neurons is defined to be the smallest number of neurons separating them plus one. In the first step, a customer is randomly picked up, his position is compared to all positions of neurons on the ring. The nearest neuron to the customer is then selected and moved towards him. The neighbors of the selected neuron move also towards the customer with a decreasing intensity controlled by the lateral function Kohonen (1982). An extensive analysis of this algorithm could be found in the works published by Fort (1988), Angeniol *et al*. (1988) and Smith (1999).

3 SOM for the VRP with Backhauls

The classical SOM approach consists of a neural network with a defined architecture that is interacting with its environment to represent it according to the following principle:
Two neighboring inputs should be represented by two neighboring neurons. The neighborhood relationship between inputs is defined according to the Euclidean distance in this paper. The concept of neighboring neurons refers to neighborhood in the map formed by the neurons. This relationship is defined according to connectivity.
From a routing perspective, it is clear that two neighboring customers should be assigned close positions in the routing schedule in order to minimize the total distance traveled by the vehicle.
In this section, we will explain how to extend the SOM to VRPB. This extension is based on the design of a new architecture in which the TSP ring is replaced by a certain number of rings. Each ring represents a vehicle. For more details refer to [1,2]. In order to represent the concept of linehaul customers and backhaul customers, each ring will consist of two parts. The first part is a sequence of neurons that will interact exclusively with linehaul customers. They are called linehaul neurons. The second part of the ring consists of backhaul neurons, which means that these neurons will interact exclusively with backhaul customers. Experiments show that this architecture lacks flexibility preventing the network from evolving adequately. Therefore, the concept of ring has been replaced by the concept of chain. There will be two types of chains. Linehaul chains are formed of linehaul neurons and backhaul chains formed of backhaul neurons. The linehaul customers will interact exclusively with the linehaul chains,

the backhaul customers will interact exclusively with the backhaul chains. It is clear that those chains will not form tours. Therefore a procedure has to be implemented in order to respect the following requirements:

- Rings should be formed to represent the sequence according to which the customers are visited
- Each ring must be formed of a linehaul sequence followed by a backhaul sequence. The linehaul sequence represents the schedule of visiting the linehaul customers served by the corresponding vehicle and the backhaul sequence represents the schedule of visiting the backhaul customers served by the same vehicle.
- Each ring must pass by the depot.

Accordingly, the procedure consists of connecting the chain such that the requirements are respected. Consequently, four types of interactions are introduced to generate a feasible VRPB tour. Let us introduce the following notations:

$\mathcal{L} = \{l_i = (x_{1i}, x_{2i}, x_{3i}), for\ i = 1,.., N_1\}$ be the set of N_1 Linehaul customers where (x_{1i}, x_{2i}) are the coordinates of the Linehaul customer l_i and x_{3i} its weight.

$B = \{b_j = (y_{1j}, y_{2j}, y_{3j}), for\ j = 1,..,N_2\}$ be the set of N_2 backhaul customers where (y_{1j}, y_{2j}) are the coordinates of the backhaul customer b_j and y_{3j} its weight.

$D = (x_d, y_d)$ be the coordinates of the depot.

$V = \{v_k, for\ k = 1,\ldots,N_v\}$ be the set of vehicles where N_v is the number of vehicles

Q is the vehicle capacity. Q is fixed because we have a homogenous fleet.

W_m: is the current amount of goods delivered to Linehaul customers by the vehicle vm , where $m=1,...,N_v$

G_n: is the current amount of goods picked up at backhaul customers by the vehicle vn, where $n=1,...,N_v$

$N_k = \{N_{kl}, N_{kb}\}$ is the set of customers served by vehicle vk, where N_{kl} is the set of linehaul customers and N_{kb} is the set of backhaul customers, where $k = 1,...,N_v$

$C_m = \{L_j^m = (X_{1j}^m, X_{2j}^m)$ for j = $1,...,N_m\}$ be the set of N_m connected neurons forming the Linehaul chain of neurons., where $m = 1,\ldots, N_v$ and (X_{1j}^m, X_{2j}^m) are the coordinates in the Euclidean space of the neuron $\boldsymbol{L_j^m}$.

$\tilde{C}_n = \{B_j^n = (Y_{1j}^n, Y_{2j}^n), for\ j = 1,...,N_n\}$ the set of N_n connected neurons forming the Backhaul chain of neurons., where $n = 1,\ldots, N_v$ and (Y_{1j}^n, Y_{2j}^n) are the coordinates in the Euclidean space of the neuron $\boldsymbol{B_j^n}$.

(i) Interaction between the chains C_m and the Linehaul customers in $\mathcal{L}$.

In this interaction, the Linehaul customers in $\mathcal{L}$ will be presented to the chains C_m one by one in a random order. In order to choose the chain that will interact with the presented customer, we have to consider two factors: 1) The distance of the nearest neuron from each chain to the customer 2) The current weight of each chain. Each time a customer is assigned to a certain chain its current weight wm will be increased by the corresponding weight. The winning chain will be selected randomly according to a probability distribution taking into consideration

these two factors. Once a chain is selected, the position of its neurons will be adjusted according to the adaptation rule.

(ii) Interaction between the chains C_n and the backhaul customers in $\mathcal{B}$.

In this interaction the backhaul customers in $\mathcal{B}$ will interact with C_n in a similar way to the interaction of type (i) and use to same adaptation rule.

(iii) Interaction between the chains C_m and C_n.

Using the previous types of interactions, the chains will evolve independently. Nothing is forcing them to be connected in order to form a feasible route. For this reason, an interaction between the two chains C_m and C_n, is introduced. We assume that each chain has a head and a tail. The tail and the head are represented by the last and the first neurons respectively. After presenting all backhaul and Linehaul customers, the chain C_m will interact with the chain C_n having the nearest neuron tail to the C_m neuron head. The objective of this interaction is to make the tail of the linehaul chain and the head of the corresponding backhaul chain converge. This convergence will allow the formation of a single ring representing a tour visiting the Linehaul and backhaul customers consecutively. The first neuron of the backhaul chain is assigned as the winner neuron in this interaction. This means that the algorithm at this level is not anymore a competitive algorithm but a supervised one in the sense that the last neuron of the linehaul chain has to be attracted by the first neuron of the backhaul chain. After this assignment, the adaptation rule has to be applied on the neurons of C_n .We apply the same procedure to the backahul chain, by presenting the first neuron of C_n to the first chain, assigning the last neuron of C_m as the winner neuron and updating the positions of the neurons of C_m according to the same adaptation rule.

(iv) Interaction between the two types of chains and the depot.

This type of interaction is similar to the last one, where the depot is presented to the linehaul chains. The first neuron of each C_m is assigned to the depot and considered as the winner neuron. Once this neuron is assigned, we update this neuron and its neighboring neurons according to the usual adaptation rule. The same procedure is applied to the last neuron of each chain C_n. The position of the neuron in the chains will give the position of the customers in the route.

3 The CP-SOM Algorithm

In this section the CP-SOM algorithm for VRPB is sketched in pseudo-code. Let us introduce the following additional notations:

dL: is the lateral distance.

δ, β: parameters to control the probability function

$\acute{\eta}, \alpha$: parameters to control adaptation rule

t : iteration number

Step 1: Initialization

Step 2: Select a city randomly

Step 3: Select a Winner neuron

If (C belongs to $\mathcal{L}$) **Then**

I. Selection of the nearest neuron for each C_m, L_m^*

Let L_m^* be the winning neuron belonging to C_m, i.e. $L_m^* = (X_{1m}^*, X_{2m}^*)$

Such that

$$(X_{1c} - X_{1m}^*)^2 + (X_{2c} - X_{2m}^*)^2 \le (X_{1c} - X_{1i})^2 + (X_{2c} - X_{2i})^2$$

$\forall\ i = 1,...,N_1$

$\forall\ m = 1,...,N_v.$

d (C , L_m^*) is the Euclidean distance between C and L_m^*

II. Select the assigned chain according to the probability:

$$P(C, C_m) = \frac{\exp(\frac{-\mathrm{d}(\mathrm{C}, L_m^*)^2}{\delta(t)}) \Big/ 1+\frac{\exp(-m+w_m+x_{3k})}{\beta(t)}}{\sum_m \exp(\frac{-\mathrm{d}(\mathrm{C}, L_m^*)^2}{\delta(t)}) \Big/ 1+\frac{\exp(-m+w_m+x_{3k})}{\beta(t)}}$$

$\forall\ m = 1,...,N_v.$

Assume C_s to be the assigned chain

$w_s = w_s + x_{3k}$

Add C to N_{sl}

III. Update the coordinates of each neuron in set C_s for example the x-coordinate is updated by the following rule:

$$X_{1j}^s\ (\mathrm{t+1}) = X_{1j}^s\ (\mathrm{t}) + \acute{\eta}\ (\mathrm{t}) \times \Gamma(\mathrm{C},\ L_s^*) \times (x_{1C} - X_{1j}^s\ (\mathrm{t})),$$

$\forall\ j = 1,..., N_s.$

where

$$\Gamma\ (C\ ,\ L_s^*) = \frac{1}{\sqrt{2}} \exp\left(\frac{-d_L^2\ (L_j^s\ ,\ L_s^*)}{2\sigma\ (t)^2} \right)$$

If (C belongs to $\mathcal{B}$) **Then**

I. Selection of the nearest neuron for each C_n, B_n^*

Let B*n be the winning neuron belonging to C_n, i.e. $B_n^* = (Y_{1n}^*, Y_{2n}^*)$ such that

$$(Y_{1c} - Y_{1m}^*)^2 + (Y_{2c} - Y_{2m}^*)^2 \le (Y_{1c} - Y_{1i})^2 + (Y_{2c} - Y_{2i})^2$$

$\forall\ i = 1,...,N_2$

$\forall\ n = 1,...,N_v.$

II. Select the assigned chain according to the probability:

$$\tilde{P}(C, C_n) = \frac{\exp(\frac{-\mathrm{d}(\mathrm{C}, B_s^*)^2}{\delta(t)}) \Big/ 1+\frac{\exp(-m+g_n+y_{3i})}{\beta(t)}}{\sum_n \exp(\frac{-\mathrm{d}(\mathrm{C}, B_s^*)^2}{\delta(t)}) \Big/ 1+\frac{\exp(-m+g_n+y_{3i})}{\beta(t)}}$$

$\forall\ n = 1,...,N_v.$

Assume $\tilde{C}_s$ to be the assigned chain

$g_s = g_s + y_{3k}$

Add C to N_{sb}

III. Update the coordinates of each neuron in set $\tilde{C}_s$ for example the x-coordinate is updated by the following rule:

$Y_{1j}^{s}\ (t+1) = Y_{1j}^{s}\ (t) + \acute{\eta}\ (t) \times \Gamma\ (C,\ B_s^*) \times (\ Y_{1C} - \ Y_{1j}^{s}\ (t))$,

$\forall\ j = 1,\ldots, N_s$.

where

$$\Gamma(C, B_s^*) = \frac{1}{\sqrt{2}} \exp\left(\frac{-d_L^2(B_j^s, B_s^*)}{2\sigma(t)^2} \right)$$

Step 4. Extremities interactions: Apply the different types of Interactions

Step 5. End-Iteration Test:

If Not {all customers are selected at the current iteration**} Then** go to Step 2.

Step 6. Stopping Criterion:

If {all customers are within $10^{-4}$of their nearest neurons in the Euclidean space**}**

Then Stop

4. Computational Experience

Our computational experience is designed to analyze the performance of the CP-SOM in terms of solution quality and computational requirements. The computational results are reported using a set of 33 instances, which were proposed in Toth & Vigo [8]. These instances are generated from the 11 classical instances of the VRP literature. The VRPB instances range in sizes between 21 and 100 customers. For each VRP problem instance, three VRPB instances are generated with ratios $\rho = \left| B/N \right|$ -the number of backhaul customers over the total number of backhaul and linehaul customers- ranging from 50, 66 and 80%. These instances were also used to report the experimental experience of many researchers, Toth and Vigo [9] and Wassan and Osman [6].

The parameters are chosen experimentally and are slowly decreased at each iteration by 1%. The Algorithm is robust in terms of parameters.

The proposed algorithms for the VRPB are coded in C and run on a PC Intel Pentium MMX 233 MHz. The quality of an algorithm is measured by the relative percentage deviation (RPD) of the solution value from its optimal solution, or best-known value published in the literature and by the average of RPDs over all instances (ARPD).

Three SOM variants, CP-SOM, CP-SOM1 and CP-SOM2, were implemented for the VRPB. The main difference between them is based on the way the local 2-opt optimization is embedded, while keeping invariant other parameters. The CP_SOM algorithm is the basic algorithm using SOM principles. In C_SOM1 a 2-opt procedure is used to improve the CP-SOM generated solution, whereasCP-SOM2 calls periodically the 2-opt procedure every 50 iterations within the CP-SOM implementation rather than only at the end as in CP-SOM1. The results comparing the performance of the 3 variants are given in table 1. We can observe that

the post-optimization has improved the results by 5% percent for large instances and around 2% for instances of 50 points. For small instances CP-SOM is still getting the best known results. This means, that the performance of the neural approach is good in the allocation stage but can be improved at the scheduling stage by a local search procedure. CPU time for post-optimization is between 2 and 3% above the CP-SOM CPU time. This confirms that the neural network provides the solution while the improvements are due to the local search. Comparing the CPU time with other techniques such as the reactive tabu search [6] is not easy because of the use of different machines and programming languages. However, using some reasonnable approximation, it can be seen that the CP-SOM algorithms require more CPU time than other algorithms when the vehicle capacity is tight and when the number of linehaul and backhaul customers are not balanced. This can be seen especially for the instances corresponding to 100 customers. However, on average CP-SOM consumes 20% CPU time less than the reactive tabu search method for a performance that is 1.66% better.

5 Conclusion

The SOM heuristic variants CP-SOM1 and CP-SOM2 are designed and implemented for the VRPB. Their comparisons with the best existing heuristics show that they are competitive with respect to solution quality, but they require more computational effort, similar to other neural networks in the literature. In particular, CP-SOM1 heuristic is the best performing algorithms for small-sized instances up to 32 customers. Therefore, it must be recommended for such instances. For the medium to large instances, the performance of CP-SOM1 was enhanced by embedding a periodic-improvement strategy within its implementation leading to CP-SOM2. In general, our SOM based neural approach is by far more powerful, flexible and simple to implement than the Hopfield-Tank neural network method.

References

[1] J.J. Hopfield, & D.W. Tank, Neural computation of decisions in optimization problem. *Biological Cybernetics*, Vol. 52, pp. 141-152, 1985.

[2] J.C. Fort, Solving a combinatorial problem via self-organizing process: An application to the traveling salesman problem. *Biological Cybernetics*, Vol. 59, pp. 33-40, 1988.

[3] H. Ghaziri, Supervision in the self-organizing feature map: application to the vehicle routing problem. In: I.H. Osman and J.P. Kelly, *Meta-Heuristics: Theory and Applications*, (pp. 651-660). Kluwer Academic publishers, Boston, 1996.

[4] H. Ghaziri, & I. Osman, A neural network algorithm for traveling sales man problem With backhauls. *Computers & Industrial Engineering*. Vol. 44, pp. 267-281, 2003.

[5]T. Kohonen, *Self-Organizing Maps*. Springer-Verlag: Berlin, 1995.

[6] I.H. Osman, & N.A. Wassan, A reactive tabu search for the vehicle routing problem with Backhauls. *Journal of Scheduling*, Vol 5, pp 263-285, 2002.

[7] K.A. Smith, Neural network for combinatorial optimization: A review of more than a decade of research. *INFORMS Journal on Computing*, Vol. 11, pp. 15-34., 1999.

[8] P. Toth, & D. Vigo, A heuristic algorithm for the vehicle routing problem with backhauls. In *Advanced Methods in Transportation Analysis*, Bianco L, pp.585-608, 1996.

[9] P. Toth, & D. Vigo, A heuristic algorithm for the symmetric and asymmetric vehicle routing problems with backhauls. *European Journal of Operational Research,* Vol. 113, pp. 528-543, 1999.

Table 1. Comparing the performance of the 3 SOM variants with the best known results

		CP-SOM1						
	N	**21**	**22**	**29**	**32**	**50**	**75**	**100**
50		0	0	0	0	4.6	2.68	3.8
66	**RPD**	0	0	0	0	6	3.23	3.05
80		0	0	0	0	4.7	3.05	3.7
	CPU	30	31.33	52	53	83	317.25	3060.83
		CP-SOM2						
50		0	0	0	0	0	0.79	3.23
66	**RPD**	0	0	0	0	2.84	1.23	3.88
80		0	0	0	0	1.22	2.35	3.65
	CPU	32.67	34.67	53.67	54	86	331	3715.17
		CP-SOM						
50		0	0	0	1.2	4.9	2.9	3.93
66	**RPD**	0	0	0	1.7	6.3	3.41	3.96
80		0	0	0	1.76	4.94	3.18	4.13
	CPU	29	31.2	49.7	51.4	79	309.6	2952.9