ALGORITHM FOR SOLVING BIPARTITE SUBGRAPH PROBLEM WITH PROBABILISTIC SELF-ORGANIZING LEARNING

Clifford Sze-Tsan CHOY and *Wan-Chi SIU*

Department of Electronic Engineering, The Hong Kong Polytechnic University,
Hung Hom, Kowloon, Hong Kong
Tel.: (852)27666229 Fax: (852) 23628439

ABSTRACT

Self-organizing model has been successfully applied to solve some combinatorial optimization problems, including travelling salesman problem, routing problem and cell-placement problem, but there has not much work reported on its application to solve graph partitioning problem. In this paper, we propose a novel mapping which has not been proposed before, with some changes to the original Kohonen's algorithm so as to enable it to solve a partitioning problem — the bipartite subgraph problem. This new approach is compared with the maximum neural network for solving the same problem, and shows that the performance of our new approach is superior over the maximum neural network.

1. INTRODUCTION

Optimization capabilities of the Kohonen's algorithm for forming self-organizing maps [1, 2] have not been completely explored, as compared to that of Hopfield's model [3, 4]. In ref. [5], Ritter and Schulten showed that the self-organizing map can be used to solve problems in three domains, namely, sensory mappings, combinatorial optimization and motor control. Combinatorial optimization problems ranging from travelling salesman problems [6, 7], routing problems [8] and cell-placement problems [9] have also been solved by using variants of Kohonen's algorithm. Yet, there has not been any algorithm based on the Kohonen's one to solve the graph partitioning problem. In this paper, we demonstrate that a variant of the Kohonen's algorithm in forming self-organizing map can be used to solve a graph partitioning problem - the bipartite subgraph problem. This problem is introduced in Section 2 and previous approaches by other authors are briefly reviewed. Then, the probabilistic self-organizing algorithm will be introduced in Section 3. Our new algorithm will be compared with the maximum neural network approach proposed by Lee et al. [10] in solving the same problem, and finally, conclusions will be drawn.

Acknowledgment: This work was supported by the Research Grant Council of the University and Polytechnic Grant Committee under the Grant HKP152/92E (PolyU340/939)

2. THE BIPARITE SUBGRAPH PROBLEM

Let $G = (V, E)$ be a graph. The graph is bipartite if its vertex set V can be partitioned into two disjoint subsets X and Y, so that each edge is incident with one element in X and one element in Y. Equivalently, there is no edge between any two elements in subset X or subset Y. Then the bipartite subgraph problem is defined as follows [11, p196]:

> Given a graph $G = (V, E)$ and a positive integer $K < |E|$, is there a subset $E' \subseteq E$ with $|E'| > K$ such that $G' = (V, E')$ is bipartite?

The objective is to delete the minimum number of edges from G, so that the remaining graph is bipartite. A particular graph can be checked to be bipartite or not in polynomial time, but the bipartite subgraph problem is solved only for some particular cases [12, 13], while it remains NP-complete in general.

In addition to its being theoretically interesting, this particular problem has found practical applications in minimizing vias in topological routing, as demonstrated by Hsu [14]. Recently, Lee et al. [10] proposed the maximum neural network, based on the Hopfield's model [4], for solving the bipartite subgraph problem, and demonstrated that its performance was better than Hsu's greedy algorithm in solving the bipartite subgraph problem.

3. THE ORIGINAL KOHONEN'S ALGORITHM AND OUR MODIFICATIONS

Considering the original Kohonen's model [2], let the number of neurons in the self-organizing map (SOM) be N, indexed from 0 to $N-1$. Let X be the set of vectors to be learnt by the SOM, and $\vec{w_i}$ be the weight vector of the i-th neuron. Let $J\left(\vec{x}, \vec{w_k}\right)$ be a distance measure between the input vector $\vec{x} \in X$, and the k-th neuron's weight vector. Then the Kohonen's algorithm is described as follows, according to Tolat [15].

0-7803-2431-5/95 $4.00 © 1995 IEEE

Algorithm KSOM

```
1. Initialize the weight vector of each neuron to a randomly chosen vector. Set $t \leftarrow 0$.
2. While $t < t_{term}$ do
   (a) Randomly select an input vector $\vec{x}$ from the set $X$.
   (b) Find a unique neuron whose index is $k$, such that
           $J\left(\vec{x}, \vec{w_k}\right) = \min_i J\left(\vec{x}, \vec{w_i}\right)$    (1-a)
       Then, let $m(i,k)$ be the output of the neuron $i$ in response to the input vector $\vec{x}$, the weights are adapted according to
           $\triangle \vec{w_i} = -\alpha m(i,k) \bigtriangledown J^2\left(\vec{x}, \vec{w_k}\right),$    (1-b)
       where $\alpha$ is the learning rate and $\bigtriangledown J$ is the gradient of $J$. If $J$ is the Euclidean distance, the corresponding equation is
           $\triangle \vec{w_i} = \alpha m(i,k)(\vec{x} - \vec{w_i}).$    (1-c)
   (c) Set $t \leftarrow t+1$.
end While
```

The function $m(i,k)$ is called the spatial mask, and is Gaussian shaped with a maximum value of one when $i = k$, and a minimum value of zero. This spatial mask defines the neighborhoodness between any neuron and the winner. The idea of Kohonen's learning is to increase the chance of the winning neuron k to win again when the same input vector $\vec{x}$ is presented. In addition, the weights of the neurons which are neighbors of the winners are given moderate amount of learning towards the input vector $\vec{x}$. After sufficient learning, the weights in the map will self-organize into an orderly manner, preserving the topology of the input space in the map.

In using Kohonen's algorithm for solving the bipartite subgraph problem, we use a neuron to represent a node in the graph to be partitioned. We use $X = \{0,1\}$ as the set of input values, to symbolize the two classes. The weights in the neurons are scalars, which represent the closeness of the neuron to a particular class, and the distance measure between an input value x and a weight w_i is given by

$$J(x, w_i) = 1 - F(x, w_i), \tag{2-a}$$

where

$$F(x, w_i) = \frac{1}{1 + \exp\left(-\frac{(x-C)\cdot(w_i - C)}{\beta}\right)} \tag{2-b}$$

The function $F(.,.)$ in (2-b) denotes the degree of association of a particular neuron to each class. The value C is the centroid of the set of input values, and is taken as 0.5. When $F(x, w_i) = 1$, the i-th neuron is fully associated with class x, while $F(x, w_i) = 0$ means that it is fully associated with class $1 - x$. Given the graph $G = (V, E)$, the neighborhood mask $m(i,j)$ is defined as

$$m(i,j) = m_{i,j} = \begin{cases} 1 & \text{if } i = j \text{ or } (i,j) \in E \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

With $|V| = N$, our algorithm is as follows.

Algorithm PSOM1

```
1. Initialize the weight of each neuron to a randomly chosen value in the range $(C-\delta, C+\delta)$ with a uniform distribution. Set $\beta \leftarrow \beta_{init}$.
2. While true do
   (a) Set $n \leftarrow 0$.
   (b) While $n < M$ do
       i. Randomly select an input value $x \in X = \{0,1\}$.
       ii. Find a unique neuron whose index is $k$, such that
               $J(x, w_k) = \min_i J(x, w_i).$    (4-a)
       iii. The weight of the i-th neuron is adapted according to
               $\triangle w_i = \alpha m_{i,k} \bigtriangledown J(x, w_i)$
               $= -\frac{\alpha}{\beta}(x - C) \cdot F(x, w_i)[1 - F(x, w_i)] m_{i,k}.$    (4-b)
       iv. Set $n \leftarrow n+1$.
   (c) Check for convergence, i.e. ,
       $\forall i \in \{0, \ldots, N-1\}$,
           $\min\{1 - F(0, w_i), F(0, w_i)\} < T,$    (4-c)
       where $T$ is the tolerance. If the network converges, terminate the algorithm.
   (d) Set $\beta \leftarrow \gamma \cdot \beta$
end While
```

Note that the sign of Δw_i in (4-b) is the reverse of that in the original Kohonen's model in (1-b) in the algorithm KSOM. Since partitioning the same graph with the maximum number of edges being cut is equivalent to solving the bipartite subgraph problem, the algorithm must try its best to allocate neurons with edges in between them in different classes, which is exactly what this sign reversal performs. Since the term $F(x, w)[1 - F(x, w)]$ becomes the maximum when $w = C$, the winner will have less learning rate than any neuron near the middle that is connected to it. This means that although with the sign reversal, in effect, the winner is not being discouraged to win when the same input value is presented again, especially when β is not large.

By starting from a large value of β, it is similar to using a large value of temperature in simulated annealing [16], so that the final state of the network is less dependent on the initial settings. Similarly, by slowly decreasing the value of β, the network can have longer time to self-organize, so that better results can be obtained. In (4-b), the learning rate $\frac{\alpha}{\beta}$ must not be an increasing value to ensure convergence. Hence, we take $\alpha = \lambda \cdot \beta$, so that a constant learning rate λ is used. The algorithm is terminated when the degree of association of all neurons to a particular class is lower than a threshold T.

Simulations of algorithm PSOM1 indicate two problems. First, by using (4-a) in chosing the winner, a minority of neurons will persistently be chosen as winners, while the rest can never win, especially when the value of β is small. Second, the values of $|w_i|$ will be getting larger and larger, especially for those neurons that persistently win, that increase the chance of these neurons to win even more. In order to solve these two problems, we introduce the probabilistic winner and saturation on values of w_i, and the revised algorithm is as follows.

Algorithm PSOM2

```
1. Initialize the weight of each neuron to a randomly chosen value in the range $(0.5 - \delta, 0.5 + \delta)$ with a uniform distribution. Set $\beta \leftarrow \beta_{init}$.
2. While true do
   (a) Set $n \leftarrow 0$.
   (b) While $n < M$ do
       i. Randomly select an input value $x \in X = \{0, 1\}$.
       ii. The probability that neuron $k$ wins is
           $$P(w_k) = \frac{F(x, w_k)}{\sum_{i=0}^{N-1} F(x, w_i)} \quad (5\text{-a})$$
       iii. The weight of the i-th neuron is adapted according to
           $$\triangle w_i = \lambda(1 - 2F(x, w_k))(x - C) \cdot F(x, w_i)[1 - F(x, w_i)]m_{i,k}. \quad (5\text{-b})$$
       iv. For each neuron, $i$, do
           A. If $w_i < 0$, set $w_i = 0$.
           B. If $w_i > 1$, set $w_i = 1$.
       v. Set $n \leftarrow n + 1$.
   (c) Check for convergence, i.e. ,
       $\forall i \in \{0, \ldots, N - 1\}$,
           $$\min\{1 - F(0, w_i), F(0, w_i)\} < T, \quad (5\text{-c})$$
       where $T$ is the tolerance. If the network converges, terminate the algorithm.
   (d) Set $\beta \leftarrow \gamma \cdot \beta$.
end While
```

Since the winner is chosen according to (5-a), there may be chances that neurons associate with another class be taken as winner. Hence, the term $(1-2F(x, w_k))$ is added to reverse the direction of $\triangle w_i$. Step 2(b)iv limits the range of the weights to the close interval $[0, 1]$.

4. RESULTS

We compare the performance of our algorithm (PSOM2) with that of the maximum neural network (MaxNet) in [10]. Graphs used in the simulations are randomly generated with undirected edges, and with number of nodes up to 400. Graphs are classified according to the *edge density*, as in [10], where edge density is the ratio between the total number of edges divided by $\frac{N(N-1)}{2}$, where N is the number of nodes in the graph. The results tabulated in Table 1 are the minimum numbers of edges to be removed from the corresponding graph to make it bipartite using each of the algorithms. The average number of iterations used by PSOM2 is tabulated in Table 2. For algorithm PSOM2, we use the following parameters:

$$M = \frac{N}{2}, T = 0.01, \gamma = 0.95, \beta_{init} = 10, \lambda = 0.2, \delta = 0.01.$$

Note that the value of M depends on the number of nodes in the graph.

Experimental results indicate that the algorithm PSOM2 can obtain better results as compared to that of MaxNet under a wide range of edge densities and node numbers. Although the average number of iterations required by PSOM2 is approximately $70N$ for the current parameters settings, its performance can be improved by varying the parameters M and/or γ at the expense of longer processing time. As shown in Table 3, the value of $M = N$ is taken while the other parameters are kept unchanged, and it is observed that better solutions are obtained with the same set of problems. Note that the number of iterations taken is just doubled, while the solutions obtained are improved in all cases.

5. CONCLUSIONS

In this paper, we have demonstrated that self-organizing learning can be applied to solve problems other than routing problems. The solution quality of our new approach is superior over that of the maximum neural network. In maximum neural network, the solution obtained depends on the initial state of the network, which is randomly generated. So, the probability of reaching a particular solution depends merely on the chance of hitting the attraction basin of the attractor corresponding to this solution. Hence, although the maximum neural

Table 1: Best solutions from MaxNet and PSOM2

	edge density			
	5%		15%	
	MaxNet	PSOM2	MaxNet	PSOM2
100	56	52	251	247
150	142	135	641	634
200	295	285	1148	1143
250	512	497	1806	1781
300	766	741	2752	2722
400	1471	1450	5090	5050

	edge density			
	25%		40%	
	MaxNet	PSOM2	MaxNet	PSOM2
100	460	457	842	842
150	1127	1118	1949	1936
200	2119	2104	3472	3452
250	3279	3253	5518	5473
300	4804	4784	8074	8027
400	8988	8926	14752	14675

Table 2: Average number of iterations for PSOM2

	edge density			
	5%	15%	25%	40%
100	7350	6600	6250	6150
150	10800	9675	9075	9000
200	14200	12600	12100	11600
250	17375	15750	15250	14625
300	20250	18750	17850	17550
400	27000	24200	23600	22800

Table 3: Best solutions from PSOM2 with $M = N$

	edge density			
	5%	15%	25%	40%
100	50	245	452	837
150	128	629	1118	1922
200	283	1129	2091	3446
250	489	1780	3229	5449
300	739	2699	4756	8006
400	1428	5002	8887	14633

network is faster than our approach, the probability of reaching a good solution depends only on the problem instance. On the contrary, for our approach, the probability of obtaining good solutions can be increased by changing the parameters M and/or γ at the expense of longer processing time, as demonstrated by the experimental results.

Furthermore, the decreasing schedule for parameters we are using in this paper is very simple, which accounts for its slowness. Surely, determining a better decreasing schedule can reduce the number of iterations to make it faster with better solutions. We hope that we can report the related findings in the near future.

Finally, the current approach introduces a new concept of mapping a problem into self-organizing learning paradigm. In past approaches, the "distances" or "costs" between the points of interests (e.g. cities in TSP) are embedded in the geometrical structure of the learning vectors, while the structural constraint is enforced by the topology of the self-organizing map (e.g. using a ring topology in solving TSP). In this new approach, we consider the reverse - the "distances" information is embedded in the topology, while the structural constraint is in the learning vectors, with suitable modifications to the learning rule. With this thinking, the current approach can be extended to solve the more general k-partite problem.

6. REFERENCES

[1] T. Kononen. "Self-organized formation of topologically correct feature maps". *Biological Cybernetics*, 43:59–69, 1982.

[2] T. Kohonen. *"Self-Organization and Associative Memory"*. Springer-Verlag, 3 edition, 1989.

[3] J.J. Hopfield. "Neurons with graded response have collective computational properties like those of two-state neurons". *Proceedings of National Academy of Science USA*, 81:3088–3092, May 1984.

[4] J.J. Hopfield and D.W. Tank. ""Neural" Computation of Decisions in Optimization Problems". *Biological Cybernetics*, 52:141–152, 1985.

[5] Helge Ritter and Klaus Schulten. "Kohonen's Self-Organizing Maps: Exploring their Computational Capabilities". In *Proceedings of IEEE International Conference on Neural Networks*, volume I, pages 109–116, 1988.

[6] Bernard Angéniol, Gaël De La Croix Vaubois, and Jean-Yves Le Texier. "Self-Organizing Feature Maps and the Travelling Salesman Problem". *Neural Networks*, 1:289–293, 1988.

[7] Clifford Sze-Tsan Choy and Wan-Chi Siu. "Approach of using a Density Equalizing Function to Self-Organizing Learning for Solving Travelling Salesman Problem". In *Proceedings of ICASSP*, volume II, pages 581–584, 1994.

[8] Hassan EL Ghaziri. "Solving Routing Problems by a Self-Organizing Map". In T. Kohonen, K.Mäkisara, O. Simula, and J. Kangas, editors, *Artifical Neural Networks*, pages 829–834. Elsevier Science Publishers B.V. (North-Holland), 1991.

[9] Sung-Soo Kim and Chong-Min Kyung. "Circuit Placement on Arbitrarily Shaped Regions Using Self-Organization Principle". *IEEE Transactions on Computer-Aided Design*, 11(7):844–854, June 1992.

[10] Kuo Chun Lee, Nobuo Funabiki, and Yoshiyasu Takefuji. "A Parallel improvement Algorithm for the Bipartite Subgraph Problem". *IEEE Transactions on Neural Networks*, 3(1):139–145, January 1992.

[11] Michael R. Garey and David S. Johnson. *"Computers and Intractability: A Guide to the Theory of NP-Completeness"*. W. H. Freeman and Company, Bell Laboratories, Murray Hill, New Jersey, 1979.

[12] F. Barahona. "On some weakly bipartite graphs". *Operations Research Letters*, 2(5):239–242, 1983.

[13] J.A. Bondy and S.C. Locke. "Largest bipartite subgraph in triangle-free graphs with maximum degree three". *J. Graph Theory*, 10:477–504, 1986.

[14] Chi-Ping Hsu. "Minimum-Via Topological Routing". *IEEE Transactions on Computer-Aided Design*, CAD-2(4):235–246, October 1983.

[15] V.V. Tolat. "An analysis of Kohonen's self-organizing maps using a system of energy functions". *Biological Cybernetics*, (64):155–164, 1990.

[16] S.Kirkpatrick, C.D.Gelatt Jr., and M.P.Vecchi. "Optimization by Simulated Annealing". *Science*, 220(4598):671–680, May 1983.