Using the General Energy Function of the Random Neural Networks to solve the Graph Partitioning Problem

Aguilar Jose
Dpto. de Computación. Facultad de Ingeniería
Universidad de los Andes. Av. Tulio Febres. Mérida-Venezuela
email: aguilar@ing.ula.ve

ABSTRACT

Typically, the neural networks are used to provide heuristic solutions to very difficult optimization problems. This is usually achieved by designing neural networks whose energy function mimics a cost function which embodies the optimization problem to be solved. In this paper, we propose to use a general energy function of the random neural network, defined in previous work, to solve the graph partitioning problem. We show as this energy function permits to define a general method to use the random neural network in the resolution of combinatorial optimization problems.

1. Introduction

Since the seminal papers of the early eighties [9, 10, 11], the study of emergent collective properties of artificial neural networks has created an exciting area for research. For instance, it is well known that for the Hopfield Network with symmetric weights, as well as for other models, each individual state change of the networks has the effect of reducing an appropriately defined cost function or energy function [11]. This elementary but subtle observation has spawned a large body of work on using neural networks to provide heuristic solutions to computationally intractable or very difficult optimization problems. This is usually achieved by designing a Hopfield (or other appropriate neural) network whose energy function mimics a cost function which embodies the optimization problem to be solved.

The basic concept is the encoding of the optimization problem in term of states that are discrete variables in an euclidean space. A real valued global energy is then defined over the set of all possible states. This energy depends on very complex interactions between the variables and has generally some physical meaning in the context of optimization. In fact, the optimal solution is the absolute minimum of this energy and one or more local minimal can be considered as acceptable solutions to the problem.

In 1989, Gelenbe has modeled the neural network using an analogy with queuing theory. This model not uses a dynamic equation, but use a scheme of interaction among neurons. It calculates the probability of activation of the neurons in the network. Signals in this model take the form of impulses which mimic what is presently known of inter-neural signals in biophysical neural networks.

The Random Neural Network (RNN) has been used in solution optimization [2, 3] and recognition problems [3]. In [8] is proposed a supervised learning procedure for the recurrent RNN model which is mainly based on the minimization of a quadratic error function. In [2, 3], we have explored the relationship between the RNN model applied to optimization and the network learning. Recently, we have applied the evolutionary learning on the RNN model [4]. In [5], we have proposed a general energy function for the RNN to solve combinatorial optimization problems.

In this paper, we propose to use the energy function defined in [5] to solve the graph partitioning problem. This work is organized as follows: in section 2, the theoretical basis of the random neural networks is reviewed. Then, we present our general energy function. In section 4 we present the energy function for the Graph Partitionning Problem. Remarks concerning future work and concluding are provided in section 5.

2. The Random Neural Model

The Random Network model has been introduced by Gelenbe [6, 7] in 1989. The model consists of a network of n neurons in which positive and negative signals circulate. Each neuron accumulates signals as they arrive, and can fire if its total signal count at a given instant of time is positive. Firing then occurs at random according to an exponential distribution of constant rate, and signals are sent out to other neurons or to the outside of the network. Each neuron i of the network is represented at any time t by its input signal potential $k_i(t)$. Positive and negative signals have different roles in the network. A negative signal reduces by 1 the potential of the neuron to which it arrives (inhibition) or has no effect on the signal potential if it is already zero; while an arriving positive signal adds 1 to the neuron potential.

Signals can either arrive to a neuron from the outside of the network or from other neurons. Each time a neuron fires, a signal leaves it depleting the total input potential of the neuron. A signal which leaves neuron *i* heads for neuron *j* with probability $p^+(i,j)$ as a positive signal (excitation), or as negative signal with probability $p^-(i,j)$ (inhibition), or it departs from the network with probability d(i). Clearly we shall have:

$$\sum_{j=1}^{n} [p^+(i,j)+p^-(i,j)] + d(i) = 1 \quad \text{for } 1\le i\le n.$$

Positive signals arrive to the i^{th} neuron according to a Poisson process of rate $\Lambda(i)$ (external excitation signals). Negative signals arrive to the i^{th} neuron according to a Poisson process of rate $\lambda(i)$ (external inhibition signals). The rate at which neuron i fires is r(i). The main property of this model is the excitation probability of a neuron *i*, q(i), which satisfy a non-linear equation:

$$q(i) = \lambda^+(i)/(r(i)+\lambda^-(i)) \tag{1}$$

where, $\lambda^+(i) = \sum_{j=1}^{n} q(j)r(j)p^+(j,i)+\Lambda(i)$

$\lambda^-(i) = \sum_{j=1}^{n} q(j)r(j)p^-(j,i)+\lambda(i)$

3. A General Energy Function for the Random Neural Model

In the Random Neural Model, q(i) depends on $\Lambda(i)$, $\lambda(i)$, $p^+(j,i)$, $p^-(j,i)$, r(i) and the other q(j)s. In the optimization, $p^+(j,i)$, $p^-(j,i)$ and r(i) are fixed and depend on the nature of combinatorial problem. Besides, in the optimization problem the relationship between two neurons is competitive or cooperative, that is either $p^+(j,i)$ or $p^-(j,i)$ is null. Of course, if there are not interaction between them, both $p^+(j,i)$ and $p^-(j,i)$ are null. On the other hand, emission of external signals is not interesting to optimization, it is better to employ the signals to inhibit or to excite the neighbor neurons, that is d(i) is null. The fire rate r(i) is obtained by the reciprocity of effect between neurons. When two neurons i and j are excited and i emits signals to j, the excitation or inhibition that i exerts over j must be the same as excitation or inhibition that i receives.

If $p^+(j,i)$, $p^-(j,i)$ and r(i) are fixed, the only way to lead the network from one stationary state to another one is to act over the inputs. This state of the RNN model is defined by (q(i), ..., q(n)). The use of two externals flows to every neuron permits a complex scaling of an external positive flow to an external negative flow [2, 3]. In optimization, the use of two flows is not interesting. We consider $\lambda(i)$ as null so that the neurons only receive external positive signals, representing the preference that the neuron belongs to the solution. By this way, q(i) and $\Lambda(i)$ become the variables of the RNN model. The general form of the energy function that we have proposed in [5] is:

$$E = \sum_{i<j} a_{ij}q(i)q(j) + \sum_{i} a_{ii}\, q(i)^2 + \sum_{i} b_i q(i) + c \qquad \text{with } i,j \in [1...n] \tag{2}$$

Where a_{ij}, b_i, c are parameters of optimization problem. It is interesting to see how this energy function definition differs from the classical approach of Hopfield. Note the additional terms which are squared in one state variable and linear in the other. Therefore, the above energy function can correspond to a quadratic cost function. Our reference to a quadratic energy function is motivated by the "usual" formulation of optimization problems with neural networks. In [2, 5] we have defined a dynamic of external positive signals in RNN model, in order to find the state that gives the minimal energy in the network:

$$\Lambda(u)^{m+1} = \Lambda(u)^{m} - \mu\, [\partial E/\partial\Lambda(u)]^{m} \qquad \text{in the m-th iteration} \tag{3}$$

The general procedure that we have proposed with the RNN is [3,5]:

```
- Initialize $\Lambda(i)$ in some appropriate manner
- Repeat
  - Solve the equation (1)
  - Using (3) and the previous results, update $\Lambda(i)$.
  - If $\Lambda(i)$ is outside of $[0, r(i)]$, replace for the nearest bounds
Until the change in the new value of $q(i)$ is smaller than some predetermined valued.
```

Thus, $$\partial E/\partial\Lambda(u) = \sum_{i\neq j} X_{ij}\partial q(i)/\partial\Lambda(u)$$

where, $X_{ij} = (2a_{ii}q(i) + b_i)1[i=j] + (a_{ij}1[j>i] + a_{ji}1[j<i])\, q(j)$

Now, we must explain $\partial q(i)/\partial\Lambda(u)$ using stationary solution of the network.

given, $w_{ji}^{+} = r(j)p^{+}(j,i)$,
$w_{ji}^{-} = r(j)p^{-}(j,i)$,
$r(i) = \sum_j (w_{ij}^{+} + w_{ij}^{-})$

then $$\partial q(i)/\partial\Lambda(u) = \sum_j Y_{ij}\partial q(j)/\partial\Lambda(u) + 1[i=u]/D(u)$$

where , $$Y_{ij} = w_{ji}^{+}/[\sum_j w_{ji}^{-}q(j) + r(i)] - w_{ji}^{-}[\sum_j w_{ji}^{+}q(j) + \Lambda(i)]/[\sum_j w_{ji}^{-}q(j) + r(i)]^2$$

That is, $\partial q/\partial\Lambda(u) = \partial q/\partial\Lambda(u)\, .\, C + e_u * 1/D(u)$ with $e_{ui} = 1[i=u]$ et $C = (\sum_j Y_{ij})$

finally, $\partial q/\partial\Lambda(u) = 1/D(u) * e_u.[1-C]^{-1}$

4. Our Energy Function on the Graph Partitioning Problem

The problem consists in dividing a graph in several subgraphs, so as to minimize a given cost function. In a very general way, to place the problem on a mathematical formulation, the following definition is necessary:

$\Pi=(N,A)$	where,	Π is a directed graph,
		N is a set of n nodes on which we can associate a weight function Q : N -> R. In ours studies Q(i)=1 for i=1,..., n,
		A = ad_{ij}, are node pairs that define the arcs. It is known as adjacency matrix.

The problem consists in dividing the graph in K different subgraphs $\Pi=\{\Pi_1,..., \Pi_K\}$, according to certain constraints. The classic constraints are: a) The subgraphs must have a specify size $N_{\Pi_1}, ..., N_{\Pi_k}$, b) The arcs with extremities in different subgraphs must be minimal. The cost function associates a real value to every subgraph configuration. We propose the cost function:

$$F = \sum_{i,j\in D} ad_{ij} + b\,(\sum_{k=1}^{K}(N_{\Pi_k} - n/K)^2)/K \tag{4}$$

where $D=\{i \in \Pi_k \,\&\, j \in \Pi_l \,\&\, l \neq k\}$

The first term minimizes the edges which belong to the cut. The second summation term will have a minimum value only when the number of nodes in the partitions are the same. The balance factor (b) defines the importance of the interconnection cost with respect to imbalance cost, and $N_{\prod_k}$ is the number of nodes in $\prod_k$ $\forall$ k=1, ..., K. The graph partitioning problem is reduced to find a subgraph configuration with minimum value for the cost function. This problem is NP-complete [12].

Our RNN for this problem:

In this approach, we will construct a RNN of the type discussed above composed of nK + K neurons, where n is the number of nodes and K is the number of subgraphs. For each (node, subgraph) pair (i,u) we will have a neuron $\mu(i,u)$ whose role is to "decide" whether node i should be assigned to subgraph u. We will denote by $q(\mu(i,u))$ the probability that $\mu(i,u)$ is excited: thus if it is close to 1 we will be encouraged to assign i to u. In order to reduce connections between subgraphs in the selected partition, $\mu(i,u)$ will tend to *excite* any neuron $\mu(j,u)$ if j is connected to i, and will tend to *inhibit* $\mu(j,v)$ if j is connected to i and $u \neq v$. Similarly, $\mu(i,u)$ will *inhibit* $\mu(j,v)$, $\forall_{v=1, \dots, K}$, if j is not connected to i. On the other hand, neurons $\mu(i,u)$ and $\mu(i,v)$, $u \neq v$, will *inhibit* each other so as to indicate that the same node should not be assigned to different subgraphs.

For each subgraph u we will have a neuron $\pi(u)$ whose role is to let us know whether u is heavily loaded with node or not. If u is very heavily loaded, it will attempt to reduce the load on subgraph u by *inhibiting* neurons $\mu(i,u)$, and it will attempt to increase the load on subgraphs $v \neq u$ by *exciting* neurons $\pi(v)$. In the same way, $\mu(i,u)$ will *excite* neuron $\pi(u)$ to increase the load on subgraph u. The parameters of the random network model expressing these intuitive criteria are chosen as follows:

- $\Lambda(\mu(i,u))$ = random, - $\lambda(\mu(i,u)) = 0$,
- $\Lambda(\pi(u)) = n/K$, to express the desirable equal load sharing property,
- $\lambda(\pi(u)) = 0$, - $r(\mu(i,u)) = nK$
- $r(\pi(u)) = n+K-1$

- $r(\mu(i,u))p^+(\mu(i,u),\mu(j,v))$	=	1 if ($ad_{ij} = 1$ or $ad_{ji} = 1$) and u=v, 0 otherwise.
- $r(\mu(i,u))\, p^-(\mu(i,u),\mu(j,v))$	=	1 if (($u \neq v$ and ($ad_{ij}= 1$ or $ad_{ji} = 1$ or i = j)) or ($ad_{ij}=0$ and $ad_{ji}=0$)), 0 otherwise.
- $r(\mu(i,u))p^+(\mu(i,u), \pi(v))$	=	1 if u=v, 0 otherwise.
- $r(\pi(u))p^-(\pi(u),\mu(i,u))$	=	1 if $q(\pi(u)) \sim 1$, 0 otherwise
- $r(\pi(u))p^+(\pi(u),\pi(v))$	=	1 if $q(\pi(u)) \sim 1$, 0 otherwise

The equation (1) for this case is:

$$q(\mu(i,u)) = \{ \textstyle\sum_{(ad_{ij} = 1 \text{ or } ad_{ji} =)1} q(\mu(j,u))r(\mu(j,u))p^+(\mu(j,u),\mu(i,u)) \} /$$
$$\{ r(\mu(i,u)) + \textstyle\sum_{v\neq u}\sum_{(ad_{ij}= 1 \text{ or } ad_{ji} = 1 \text{ or } i = j)} q(\mu(j,v))r(\mu(j,v))p^-(\mu(j,v),\mu(i,u)) +$$
$$\textstyle\sum_v \sum_{ad_{ij}=0\ \&\ ad_{ji}=0} q(\mu(j,v))r(\mu(j,v))p^-(\mu(j,v),\mu(i,u)) + q(\pi(u))r(\pi(u))p^-(\pi(u),\mu(i,u)) \}$$

$$q(\pi(u)) = \{ \Lambda(\pi(u)) + \textstyle\sum^{n}_{j=1} q(\mu(j,u))r(\mu(j,u))p^+(\mu(j,u),\pi(u)) + \sum^{K}_{v=1} q(\pi(v))r(\pi(v))p^+(\pi(v),\pi(u)) \} / r(\pi(u)) \tag{5}$$

For this problem, using the general energy function (2) and the cost function (4), we propose the following energy function:

$$E = \sum_{i,j \in D} ad_{ij}\, q(\mu(i,k))\, q(\mu(j,l)) + b \left(\sum_{k=1}^{K} \left(\sum_{i=1}^{n} q(\mu(i,k)) - {}^{n}\!/_{K} \right)^2 \right) / K + \sum_{i=1}^{n} \left(\sum_{k=1}^{K} q(\mu(i,k)) - 1 \right)^2$$

If we developed this function, we obtain the following value to a_{ij}, b_i and c:

$$a_{ij} = ad_{ij} \quad \text{if } i < j$$
$$0 \quad \text{otherwise}$$
$$a_{ii} = b/k + 1$$
$$b_i = -2(nb/K^2+1)$$
$$c = bn^2/K^3 + 1$$

Performance Evaluation:

We compare the RNN with the approximate heuristics proposed in [1, 3]: genetic algorithms (GA), simulated annealing (SA) and kernighan's heuristic (Kern). The random graphs used are defined for the average number of nodes (n) and the average degree of the successor nodes of a node (d). For each graph, the successors of a node are chosen randomly from a uniform distribution in the interval [1, d]. The execution time is in seconds.

The parameters of the simulations are the following: the total number of subgraphs (K), the mean number of nodes per graph (n), the mean number of successors per node (d) and the balance factor (b). We generate 50 random graphs for the set of parameters where n = {10, 20, 50}, K=2 and d=2.

We obtain the optimum solutions using an enumerative search algorithm. We study the following performance criteria: the execution time of the heuristics (T_l), the maximum performance (Sop_l), the percentage of optimum solutions (δ_l) and the relative error (E_l) of each heuristic, where l is the number of times that we execute the heuristic to obtain these values. These criteria are calculated as follows:

- T_l is the mean value of the computation time on a workstation for each heuristic, for l=1, ..., 10.
- Sop_l is the mean value of the solutions for a given set of parameters, for l=1,,...., 10.
- δ_l is the percentage of cases where a heuristic obtains the optimum solution, for l=1, ..., 10.
- E_l is the mean relative error of the solutions of a heuristic compared to the optimum solution,

$$E_l=\Sigma_{i,l}\,(S_{il} - S_i^{opt}) \,/\, S_i^{opt} \text{ for } l=1, ...,10 \text{ and } i=1,...,50$$

where S_i^{opt} is the optimum solution of the graph i and S_{il} is the solution of the heuristic for the graph i. Due to space limitations, the results presented in this section were chosen because they are representative of the phenomena studied.

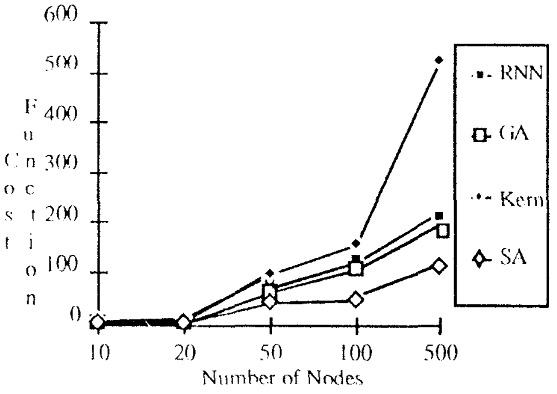

Figure 4.1. Simulation Results for K= 2, b = 1 and d = 2

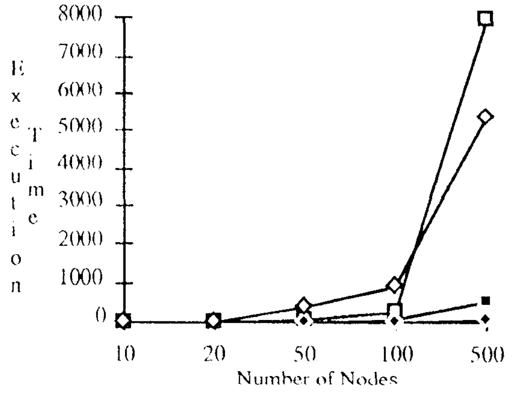

Figure 4.2 Execution Times of the heuristics for K= 2, b = 1 and d = 2

Simulated Annealing and Genetic Algorithms, for graphs of 50 or more nodes, need a very large time to reach the suboptimal solutions (Figures 4.2). For graph of little size (of 20 or fewer nodes), the difference between the exact solution and the results of the heuristics is little (figure 4.1). Otherwise, Simulated Annealing gives the best results.

Method	E	δ	T	Sop
SA	0.06	0.9	28	2.6
GA	0.12	0.8	19	2.8
RNN	0.12	0.8	9	2.8
Kernighan heuristic	1.06	0.15	3	4.1

Table 4.1. Performance criteria for l=10, n=20, d=2, b=1 and K=2

Sop and δ are approximately the same for every heuristic, but Kernighan's heuristic gives the worst results (Table 4.1). SA is the heuristic with the least mean relative error, but we obtain interesting results with RNN with short execution time.

5. Conclusions

The purpose of this paper has been to consider the formulation of a general energy function to solve combinatorial optimization problems using the random neural networks. The major advantage of this model is that it has a purely numerical and computationally fast solution, which removes the need for complex search techniques and other Montecarlo simulations based optimization methods. The definition of a general energy function for the RNN permits to describe a dynamic to search an optimum solution of a combinatorial optimization problem.

Then, we have illustrated the utilization of our general energy function on the Graph Partitioning Problem. Further work will examine the results proved for this model using this energy function in other optimization problems.

References

[1] J. AGUILAR, "Combinatorial Optimization Methods. A study of graph partitioning problem", in *Proc. of the Panamerican Workshop on Applied and Computational Mathematics-PWACM*, January 1993.

[2] J. AGUILAR, "An approach for combinatorial optimization problem based on learning in the recurrent random neural network", *Proc. of the World Congress on Neural Networks*, July 1994.

[3] J. AGUILAR, L'allocation de tâches, l'équilibrage de la charge et l'optimisation combinatoire, PhD thesis, Rene Descartes University, 1995.

[4] J. AGUILAR, "Evolutionary Learning on Recurrent Random Neural Network", in *Proc. of the World Congress on Neural Networks*, 1995.

[5] J. AGUILAR, "An Energy Function for the Random Neural Networks". Tech. Rep. The Andes University, Sept. 1995.

[6] E. GELENBE, "Random neural networks with positive and negative signals and product form solution", Neural Computation, vol. 1, no. 4, 1989.

[7] E. GELENBE, "Theory of the random neural network model", in *Neural networks: Advances and Applications*, E. Gelenbe, Ed. North-Holland, 1991.

[8] E. GELENBE, "Learning in the recurrent Random Neural Network", *Neural Computation*, vol. 5, no. 5, 1993.

[9] S. GROSSBERG, "Nonlinear neural networks: principles, mechanisms and architectures", *Neural Networks*, vol. 1, no. 1, 1988.

[10] C. HERAULT and J. NIEZ, "Neural networks and combinatorial optimization: a study of NP-complete graph problems", in *Neural networks: Advances and Applications*, E. Gelenbe, Ed. North Holland, 1991.

[11] J. HOPFIELD and D. TANK, "Neural computation of decisions in optimization problem", *Biolog. Cybern.*, no 52, 1985.

[12] B. KERNIGHAN and S. LIN, "An efficient heuristic procedure for partitioning graphs", *The Bell System Technical Journal*, February 1970.