PERGAMON

Neural Networks 11 (1998) 731–737

Neural Networks

Contributed article

Definition of an energy function for the random neural to solve optimization problems

Aguilar Jose[a,*]

[a]*Departmento de Computación, Facultad de Ingeniería, Universidad de los Andes, Mérida, Venezuela*

Received 30 June 1996; accepted 29 January 1998

Abstract

In this paper, we propose a general energy function for a new neural model, the random neural model of Gelenbe. This model proposes a scheme of interaction between the neurons and not a dynamic equation of the system. We then apply this general energy function on different optimization problems: the graph partitionning problem and the minimum node covering problem.

Keywords: Optimization problems; Neural networks; Energy function; Graph partitionning; Minimum node covering

1. Introduction

Neural network research has been popular since the 1960s. There has been a major resurgence of interest in artificial neural networks in recent years, primarily because of improved learning algorithms, theoretical foundations [due to pioneers such as Grossberg (1988)], computer systems for simulation studies and technologies that make a quick and inexpensive implementation possible.

Since the seminal papers of the early 1980s (Hopfield and Tank, 1985; Grossberg, 1988; Herault and Niez, 1991; Tagliarini et al., 1991), 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 (Hopfield and Tank, 1985). 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.

In this way, the *energy function* forms the theoretical basis for the optimization of functions using neural networks. The term *energy function* stems from an analogy between the network's behavior and that of certain physical systems. Just as physical systems may evolve towards an equilibrium state, a network of artificial neurons will evolve toward a minimum of the energy function. Therefore, the stable states of a network of neurons correspond to the local minimal of the energy function.

The basic concept is the encoding of the optimization problem in terms of states that are discrete variables in a 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 minima can be considered as acceptable solutions to the problem. The neural methods proposed to minimize such global energy function differ from the previous heuristic methods by the following characteristics:

1. noise can be introduced in the searching dynamics in order to explore any part of the solution space;
2. they are not specific to a particular optimization problem;
3. they can easily be implemented on massively parallel architectures;
4. they give good results, irrespective of the number of variables; and
5. one can easily relax the constraints of the problem.

In 1989, Gelenbe (Gelenbe, 1989, 1990) modeled the

* Corresponding author. Dpto. de Computación, Fac. de Ingeniería, Universidad de los Andes. Av. Tulio Febres, 5101 Mérida, Venezuela; E-mail: aguiai@ing.ula.ve.

0893-6080/98/$19.00
PII S0893-6080(98)00020-3

neural network using an analogy with queuing theory. This model does not use a dynamic equation, but uses an interaction scheme between neurons. It calculates the activation probability of the network neurons. The signals in this model take the form of impulses that mimic what is presently known of interneural signals in biophysical neural networks.

The random neural network (RNN) has been used in solution optimization (Gelenbe and Batty, 1992; Aguilar, 1992, 1995a; Pekergin, 1992) and recognition problems (Pekergin, 1992; Aguilar, 1995a). In (Gelenbe 1993) a supervised learning procedure is proposed for the recurrent RNN model which is mainly based on the minimization of a quadratic error function. In (Aguilar 1994, 1995a), the relationship between the RNN model applied to optimization and the network learning is explored. Recently, we have applied the evolutionary learning on the RNN model (Aguilar, 1995b).

In this paper, a general energy function for the random neural network is proposed. Then, this general energy function is applied to different optimization problems. This work is organized as follows. In Section 2, the theoretical basis of the random neural networks is reviewed. Then, our general energy function is presented. In Section 4 the energy function for two NP hard problems, graph partitionning and minimum node (graph) covering, is given. Remarks concerning future work and conclusions are provided in Section 5.

2. The random network model

The random network model was introduced by Gelenbe (Gelenbe, 1989, 1990, 1991) in 1989. This model has a remarkable property called ''*product form*'' which allows the computation of joint probability distributions of the neurons of the network.

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 occurs at random according to an exponential distribution of constant rate, and signals are sent 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 if it is already zero, while a positive signal adds 1 to the neuron potential.

Signals can either arrive at 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 leaving 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 leaves the network with probability $d(i)$. Clearly we shall have:

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

Positive signals arrive at the ith neuron according to a Poisson process with rate $\Lambda(i)$ (external excitation signals). Negative signals arrive at the ith neuron according to a Poisson process with 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:

$$q(i)=\lambda^+(i)/(r(i)+\lambda^-(i)) \tag{2}$$

where

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

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

If a unique non-negative solution exists to Eq. (2) such that each $q(i)\le 1$, then the stationary probability distribution is

$$p(k)=\Pi_{i=1}^{n}(1-q(i))q(i)^{k_i} \tag{3}$$

where $k(t)$ = vector of signal potentials at time t and $k=(k_1,...,k_n)$ = particular value of the vector.

To guarantee the stability of the RNN, the following is a sufficient condition for the existence and uniqueness of the solution to Eq. (2):

$$\lambda^+(i)<[r(i)+\lambda^-(i)] \tag{4}$$

Notice that the model is based on rates, much as natural neural systems operate. Thus, this is a ''*frequency modulated*'' model, which translates rates of signal emission into excitation probabilities via Eq. (2). For instance, $q(j)r(j)p^+(j,i)$ denotes the rate at which neuron j excites neuron i. Eq. (2) can also be translated into a special form of sigmoid which treats excitation (in the numerator) asymetrically with respect to inhibition (in the denominator).

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 the 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 is no interaction between them, both $p^+(j,i)$ and $p^-(j,i)$ are null. On the other hand, external signals are not interesting to optimization. It is better to employ 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 the 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 by acting over the inputs. A state of the RNN model is defined by $(q(i),..., q(n))$. The use of two externals flows for every neuron allows for a complex scaling of an external positive flow to an external negative flow (Pekergin, 1992; Aguilar, 1994, 1995a). In optimization, the use of two flows is not interesting. We consider $\lambda(i)$ to be 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 propose is:

$$E = \sum_{i<j} a_{ij} q(i) q(j) + \sum_i a_{ii} q(i)^2 + \sum_i b_i q(i) + c \tag{5}$$

with $i, j \in [1...n]$, where a_{ij}, b_i, c are parameters of the optimization problem.

It is interesting to see how this energy function definition differs from the classical approach of Hopfield. Note the additional terms that are squared in one state variable and linear in the other. Therefore, the above energy function corresponds to a quadratic cost function. Our reference to a quadratic energy function is motivated by the ''usual'' formulation of optimization problems with neural networks.

Now, we search to define the dynamics of external positive signals in the RNN model in order to find the state that gives the minimal energy for the network. Using the gradient descent technique, the dynamics of external excitation signals are defined as:

$$\Lambda(u)^{m+1} = \Lambda(u)^m - \mu[\partial E/\partial \Lambda(u)]^m \tag{6}$$

in the mth iteration.

Eq. (6) describes the control that is necessary to apply to the system to minimize the energy function. This method uses a learning technique in which the network learns to minimize the energy function (Eq. (5)). The general procedure that we propose with the RNN is (Aguilar, 1995a):

1. initialize $\wedge(i)$ in some appropriate manner
2. repeat
   3. solve Eq. (2)
   4. using Eq. (6) and the previous results, update $\wedge(i)$
   5. if $\wedge(i)$ is outside of $[0, r(i)]$, replace for the nearest bounds
6. until the change in the new value of $q(i)$ is smaller than some predetermined value.

Thus,

$$\partial E/\partial \Lambda(u) = \sum_{i \neq j} (a_{ij} \mathbf{1}[j > i] + a_{ji} \mathbf{1}[j < i]) q(j) \partial q(i)/\partial \Lambda(u) + \sum_i (2a_{ii} q(i) + b_i) \partial q(i)/\partial \Lambda(u) \tag{7}$$

$$\partial E/\partial \Lambda(u) = \sum_{i,j} \{(2a_{ii} q(i) + b_i) \mathbf{1}[i = j] + (a_{ij} \mathbf{1}[j > i] + a_{ji} \mathbf{1}[j < i]) q(j)\} \partial q(i)/\partial \Lambda(u)$$

given

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

then,

$$\partial E/\partial \Lambda(u) = \sum_{i \neq j} X_{ij} \partial q(i)/\partial \Lambda(u) \tag{8}$$

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

$$N(i) = \sum_j w_{ji}^+ q(j) + \Lambda(i)$$

and

$$D(i) = \sum_j w_{ji}^- q(j) + \sum_j (w_{ij}^+ + w_{ij}^-)$$

where, $w_{ji}^+ = r(j) p^+(j,i)$,

$w_{ji}^- = r(j) p^-(j,i)$,

$r(i) = \sum_j (w_{ij}^+ + w_{ij}^-)$

$q(i) = N(i)/D(i)$ is the stationary probability of excitation of neuron i, if $q(i) < = 1$, and

$$\partial q(i)/\partial \Lambda(u) = \partial N(i)/\partial \Lambda(u)[1/D(i)] - \partial D(i)/\partial \Lambda(u)[N(i)/D(i)^2] \tag{9}$$

$$\partial q(i)/\partial \Lambda(u) = [\sum_j w_{ji}^+ \partial q(j)/\partial \Lambda(u) + \mathbf{1}[i = u]]/D(i) - [\sum_j w_{ji}^- \partial q(j)/\partial \Lambda(u)] N(i)/D(i)^2$$

Given: $Y_{ij} = w_{ji}^+/D(i) - w_{ji}^- N(i)/D(i)^2$,

$$\partial q(i)/\partial \Lambda(u) = \sum_j Y_{ij} \partial q(j)/\partial \Lambda(u) + \mathbf{1}[i = u]/D(u) \tag{10}$$

That is,

$$\partial q/\partial \Lambda(u) = \partial q/\partial \Lambda(u).C + e_u * 1/D(u) \tag{11}$$

with $e_{ui} = 1[i = u]$ and $C = (\Sigma_j Y_{ij})$ and, finally, $\partial q/\partial \Lambda(u) = 1/D(u) * e_u.[I - C]^{-1}$.

4. Energy function on different optimization problems

4.1. The graph partitioning problem

The problem consists in dividing a graph in several subgraphs, so as to minimize the costs of connections between them. The problem can be complicate with weighted arcs. In this case, the sum of the weights between the subsets must be minimized. Also, we can add a weight to the nodes and define again what we want to minimize according to the particular characteristics of problem (Aguilar, 1994). In a very general way, to place the problem in a mathematical formulation, the following definition is necessary: $\Pi = (N,A)$ where Π is a directed graph, N is a set of n nodes

with which we can associate a weight function $Q{:}N \rightarrow R$. In our studies $Q(i) = 1$ for $i = 1,..., n$, $A = ad_{ij}$, are node pairs that define the arcs, known as the adjacency matrix, given by the weights of the arcs of Π.

The problem consists in dividing the graph in K different subgraphs $\Pi = \{\Pi_1,..., \Pi_K\}$, according to certain constraints. The classic constraints are:

1. the subgraphs must have a specify size $N_{\Pi 1},..., N_{\Pi K}$, or must have a weight sum of nodes less than a given value; and
2. the arcs with extremities in different subgraphs must be minimal, or the weight sum of arcs that join nodes which are in different subgraphs must be mimimal.

The cost function associates a real value to every subgraph configuration. We propose the cost function:

$$F = \sum_{i,j \in D} ad_{ij} + b\Big(\sum_{k=1}^{K} (N_{\Pi_k} - \frac{n}{K})^2\Big)/K \tag{12}$$

where

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

The first term minimizes the weight sum of 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 the imbalanced cost. $N_{\Pi k}$ is the number of nodes in $\Pi_k \forall k = 1,..., K$. The graph partitioning problem is reduced to find a subgraph configuration with minimum value for the cost function.

4.1.1. 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* $\mu(j,v), \forall_{v=} 1,..., K$, 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 nodes or not. If u is very heavily loaded, we will attempt to reduce the load on subgraph u by *inhibiting* neurons $\mu(i,u)$, and 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,
$\Lambda(\pi(u)) = n/K$, to express the desirable equal load sharing property,
$\lambda(\mu(i,u)) = 0$, $\lambda(\pi(u)) = 0$,
$r(\mu(i,u)) = nK, r(\pi(u)) = n + K\text{-}1$,

$r(\mu(i,u))p^+(\mu(i,u), \mu(j,v)) = 1$ 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

if $(ad_{ij} = 0$ 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 Eq. (2) for this case is:

$$\begin{aligned} q(\mu(i,u)) = \{&\sum_{(ad_{ij}=1 \text{ or } ad_{ji}=1)} q(\mu(j,u))r(\mu(j,u)) \\ &\times p^+(\mu(j,u), \mu(i,u))\}/ \\ \{r(\mu(i,u)) + &\sum_{v \neq u} \sum_{(ad_{ij}=1 \vee ad_{ji}=1 \vee i=j)} q(\mu(j,v))r(\mu(j,v)) \\ &\times p(\mu(j,v), \mu(i,u)) + \sum_v \sum_{ad_{ij}=0 \,\&\, ad_{ji}=0} q(\mu(j,v)) \\ &\times r(\mu(j,v))p^-(\mu(j,v), \mu(i,u)) + q(\pi(u))r(\pi(u)) \\ &\times p^-(\pi(u), \mu(i,u)) \end{aligned} \tag{13}$$

$$\begin{aligned} q(\pi(u)) = \{&\Lambda(\pi(u)) + \sum_{j=1}^{n} q(\mu(j,u))r(\mu(j,u)) \\ &\times p^+(\mu(j,u), \pi(u)) + \sum_{v=1}^{K} q(\pi(v))r(\pi(v)) \\ &\times p^+(\pi(v), \pi(u))\}/r(\pi(u)) \end{aligned}$$

For this problem, using the general energy function (Eq. (5)) and the cost function (Eq. (12)), we propose the following energy function:

$$E = \sum_{i,j \in D} ad_{ij} q(\mu(i,k))q(\mu(j,l)) + b\Big(\sum_{k=1}^{K} \Big(\sum_{i=1}^{n} q(\mu(i,k)) - \frac{n}{K}\Big)^2\Big)/K + \sum_{i=1}^{n} \left(\sum_{k=1}^{K} q(\mu(i,k)) - 1\right)^2 \tag{14}$$

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

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
Kern	1.06	0.15	3	4.1

Note:

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

$a_{ij} = ad_{ij}$ if $i < j$ 0 otherwise

$a_{ii} = b/k + 1$

$b_i = -2(nb/K^2 + 1)$

$c = bn^2/K^3 + 1$

4.1.2. Performance evaluation

We compare the RNN with the approximate heuristics proposed by (Aguilar 1995a): 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, $E_l = \Sigma_{il}, (S_{il} - S_i^{\text{opt}})/S_i^{\text{opt}}$ for $l = 1,...,10$ 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, only some results are shown which are representative of the phenomena studied (see Table 1).

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

4.2. Minimum node covering problem

The formal problem can be stated as follows. Let G be a graph with n nodes $\mathbf{N} = \{V_1,..., V_n\}$ and edges denoted by (V_i,V_j). A cover of G is a subset $\mathbf{S}$ of $\mathbf{N}$ such that for each edge (V_i,V_j) in G, either V_i or V_j is in $\mathbf{S}$. A minimum cover of G is a set $\mathbf{S^*}$ such that the number of nodes in $\mathbf{S^*}$ is no larger than the number of nodes in any cover $\mathbf{S}$ of G: $|\mathbf{S^*}| \leq |\mathbf{S}|$.

The minimum graph covering problem can also be formulated so that each node of the graph G can carry a weight, and this weight is included in the ''size'' of each cover $\mathbf{S}$. The cost function for this problem is

$$F = \sum_{i=1}^{n} \{2n\beta_i^* - [D(i)\beta_i^* - D(i)\beta_i] - \sum_{j=1}^{n} ad_{ji}\beta_i^*\beta_j + \beta_i\beta_i^*\} \quad (15)$$

where, $D(i)$ is the degree of node i, $\beta_i = 1$ if node $V_i \notin S$ in this solution, 0 otherwise; $\beta_i^* = 1$ if node $V_i \in S$ in this solution, 0 otherwise.

The first term states that there should be as few nodes as possible in the minimum cover, while the second and third terms state that we should favor nodes which have a large degree. The fourth term states that we would like to have only one of each end node of an edge in the cover. In this way, this term eliminates illegal solutions. The last term states that we cannot have the same node both in and out of the cover. Though this cost function is quite elaborated, it does provide a more detailed representation of the problem's constraints. Let us now present a random neural model for the minimum graph covering problem.

4.2.1. RNN for this problem

Each node i in the graph is represented by two neurons, $\mu(i)$ and $p(i)$. The first of these will be ''on'' if the network recommends that i be included in the cover, while the second will have the opposite role. The parameters of the random network model expressing these intuitive criteria are chosen as follows:

$\Lambda(\mu(i))$ = degree of node i, $(D(i))$,
$\lambda(\mu(i)) = 0$, $\Lambda(\pi(i))$ = random,
$\lambda(\pi(i)) = 0$, $r(\mu(i)) = 2n$,

$r(\pi(i))$ = degree of node i, $(D(i))$,
$r(\mu(i))p^-\mu(i),\pi(j)) = 2n$ if $(i = j)$, 0 otherwise.
$r(\pi(i))p^+(\pi(i),\mu(j)) = 1$ if $(ad_{ij} = 1)$ or $(ad_{ji} = 1)$,
0 otherwise.
All other weights are set to zero.

Table 2
Performance criteria

Method	Exc	δ	T
$n = 20$	$\sigma = 0.5$		
RNNnL	71	0.3	12
RNNwL	67	0.6	16
$n = 50$	$\sigma = 0.125$		
RNNnL	14	0.5	4
RNNwL	10	0.7	7

Eq. (2) for this case is:

$$q(\mu(i)) = \{\Lambda(\mu(i)) + \sum_{j=1}^{n} q(\pi(j))r(\pi(j))p^{+}(\pi(j), \mu(i))\} /r(\mu(i))$$
$$\times q(\pi(i)) = \Lambda(\pi(i))/\{r(\pi(i)) + q(\mu(i))r(\mu(i)) \times p^{-}(\mu(i), \pi(i))\} \quad (16)$$

For this problem, using the general energy function (Eq. (5)) and the cost function (Eq. (15)), we propose the following energy function:

$$E = \sum_{i=1}^{n}\{2nq(\mu(i)) - [D(i)q(\mu(i)) - D(i)q(\pi(i))] - \sum_{j=1}^{n} ad_{ji}q(\mu(i))q(\pi(j)) + q(\mu(i))q(\pi(i))\} \quad (17)$$

However, if we suppose that

$q(\mu(i)) = 1 - q(\pi(j))$ (ideal case)

the energy function is:

$$E = \sum_{i=1}^{n}\sum_{j=1}^{n} ad_{ij}q(\pi(i))q(\pi(j)) + \sum_{i=1}^{n}[2D(i) + 1 - 2n - \sum_{j=1}^{n} ad_{ji}]q(\pi(i)) - \sum_{i} q^2(\pi(i)) + 2n - \sum_{i} D(i)$$

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

$a_{ij} = ad_{ij}$ if $i_j \neq 1$, -1 otherwise

$$b_i = 2D(i) + 1 - 2n - \sum_{j=1}^{n} ad_{ji}$$

$$c = 2n + \sum_{i} D(i)$$

4.2.2. Performance evaluation

We compare the RNN (RNNwL) with an approximate heuristic based on RNN without learning (RNNnL) proposed in (Gelenbe and Batty, 1992), and with the exact solution. The number of nodes in the graphs has been varied with the following values: $n = 20, 50$. Once the size of the graphs is fixed at a value, the random generation is carried out as follows. A probability value σ is fixed; this is the probability that for any pair of nodes v_i, v_j there is an edge (v_i, v_j). Different values of σ (0.125, 0.5) were taken. These results are present in Table 2. For each value (n, σ) we generated 25 graphs at random, and the results are average values over this set. This heuristics are compared with respect to the following criteria:

δ is the percentage of cases where a heuristic obtains the minimum cover;
Exc is the average number of nodes in excess of the minimum for the set of 25 graphs, for a given (n, σ) pair; and
T is the mean value of the computation time on a workstation for each heuristic.

The results of Table 2 clearly show that the RNNwL gives the best results in all cases, but with a larger execution time. This performance improvement is particularly substantial when the graph is sparse (i.e. small σ).

5. Conclusions

The purpose of this paper has been to consider the formulation of a general energy function to solve combinatorial optimization problems using 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 Monte Carlo simulations based optimization methods. The definition of a general energy function for the RNN allows the description of a dynamic to search an optimum solution for a combinatorial optimization problem.

Then, we have illustrated the utilization of our general energy function on two optimization problems: the graph partitioning and the minimum node covering. Further work will study this energy function applied to other optimization problems.

References

Aguilar, J. (1992). Comparison between the random neural network model and other optimization combinatorial methods for large acyclic graph partitioning problem. In *Proceedings of the 7th International Symposium on Computer and Information Sciences* (pp. 565–569). Informatique Etudes en des Hautes l'ecole Paris: Presses de.

Aguilar, J. (1994). An approach for combinatorial optimization problem based on learning in the recurrent random neural network. In *World Congress on Neural Networks* (pp. 420–425). New Jersey: Lawrence Erlbaum Associates.

Aguilar, J. (1995a). L'allocation de tâches, l'équilibrage de la charge et l'optimisation combinatoire. PhD thesis. Paris: Rene Descartes University.

Aguilar, J. (1995b). Evolutionary learning on recurrent random neural network. In *World Congress on Neural Networks* (pp. 232–236). New Jersey: Lawrence Erlbaum Associates.

Gelenbe E. (1989). Random neural networks with positive and negative signals and product form solution. *Neural Computation, 1*, 502–510.

Gelenbe E. (1990). Stable random neural networks. *Neural Computation, 2*, 239–247.

Gelenbe, E. (1991). Theory of the random neural network model. In E. Gelenbe (Ed.), *Neural Networks: Advances and Applications* (pp. 25–34). Amsterdam: North-Holland.

Gelenbe E. (1993). Learning in the recurrent random neural network. *Neural Computation, 5*, 364–378.

Gelenbe, E., & Batty, F. (1992). Application of the random neural network model to minimum graph covering. In E. Gelenbe (Ed.), *Neural*

Networks: Advances and Applications 2 (pp. 85–96). Amsterdam: North Holland.

Grossberg S. (1988). Nonlinear neural networks: principles, mechanisms and architectures. *Neural Networks, 1*, 12–40.

Herault, C., & Niez, J. (1991). Neural networks and combinatorial optimization: a study of NP-complete graph problems. In E. Gelenbe (Ed.), *Neural Networks: Advances and Applications* (pp. 124–143). Amsterdam: North Holland.

Hopfield J., & Tank D. (1985). Neural computation of decisions in optimization problem. *Biolog. Cybern., 52*, 141–152.

Pekergin, F. (1992). Optimisation combinatoire par le calcul neuronal et parallelisme optimal. PhD thesis. Paris: Rene Descartes University.

Tagliarini C., Fury J., & Page E. (1991). Optimization using neural network. *IEEE Transactions on Computers, 40*, 1347–1358.