ESANN'2006 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 26-28 April 2006, d-side publi., ISBN 2-930307-06-4.

Enhanced MaxCut Clustering with Multivalued Neural Networks and Functional Annealing

E. Mérida-Casermeiro[1], D. López-Rodríguez[1] and J. M. Ortiz-de-Lazcano-Lobato[2]

1- E.T.S. Ingeniería Informática - Dept. of Applied Mathematics
Universidad de Málaga - Spain

2- E.T.S. Ingeniería Informática - Dept. of Computer Science and Artificial Intelligence
Universidad de Málaga - Spain

Abstract. In this work a new algorithm to improve the performance of optimization methods, by means of avoiding certain local optima, is described. Its theoretical bases are presented in a rigorous, but intuitive, way. It has been applied concretely to the case of recurrent neural networks, in particular to MREM, a multivalued recurrent model, that has proved to obtain very good results when dealing with NP-complete combinatorial optimization problems. In order to show its efficiency, the well-known MaxCut problem for graphs has been selected as benchmark. Our proposal outperforms other specialized and powerful techniques, as shown by simulations.

1 Introduction

In specialized literature, the MaxCut problem is defined as follows: Given an undirected weighted graph $G = (V, E)$, where $V = \{v_i\}$ is the set of N vertices and E is the set of n_e edges, and edge weights are given by matrix $C = (c_{i,j})_{i,j=1,\dots,N}$ (meaning that the weight or cost of the edge joining nodes i and j is $c_{i,j} \geq 0$), find a *maximum cut* of G, i.e., a partition of V into two sets that maximizes the total cost of the edges with endpoints in different sets.

This problem arises in the resolution of many practical or theoretical situations, including pattern recognition, clustering, statistical physics and the design of communication networks, VLSI circuits and circuit layout [2]. So, this problem is well-known in literature.

The original problem, with all its variants, is known to be NP-complete [5], making its resolution computationally intractable.

In 1997, Alberti et al. presented a Hopfield-type neural model for MaxCut [1], but its performance is worse than the one presented by Bertoni et al [3]. Takefuyi and his colleagues [9] developed a powerful neural model named 'maximum' and it proved to perform better than the rest of algorithms in solving a wide range of combinatorial optimization problems. In the last few years, Galán-Marín et al. [4] proposed a new neural model named OCHOM which obtains much more efficient solutions than 'maximum'. Moreover, it can be used for many problems and it also has the advantage of fast convergence to a valid solution without tuning any parameter. In order to make OCHOM escape from local minima, Wang et al.[10] have recently proposed a stochastic dynamics for OCHOM, permitting temporary decreases of the objective function. Recently, the best results to the moment were achieved by the application of a multivalued recurrent neural network (MREM) [8] that had proved to get very good results in some combinatorial optimization problems, see [6, 7], allowing K-partitioning of a graph.

In this work, we want to present an algorithm to improve the quality of the solutions of this neural model, enhancing its performance, since it allows the net to escape from certain local optima of its energy function.

2 Formal Description of the Problem

Let $G = (V, E)$ be an undirected graph without self-connections. $V = \{v_i\}$ is the set of vertices and E is the set of n_e vertices. For each edge in E there is a weight $c_{i,j} \in \mathbb{R}^+$. All weights can be expressed by a symmetric real matrix C, with $c_{i,j} = 0$ when it does not exist an arc with endpoints v_i and v_j.

The Maximum Cut Problem (MaxCut): consists in finding a partition of V into two subsets A_1 and A_2, such that $\sum_{v_i \in A_1, v_j \in A_2, i>j} c_{i,j}$ is maximum.

Generalization of the MaxCut Problem (K-MaxCut): It looks for a partition of V into K disjoint sets A_i such that the sum of the weights of the edges from E that have their endpoints in different elements of the partition is minimum. So, the function to be maximized is

$$\sum_{v_i \in A_m, v_j \in A_n, i>j} c_{i,j} \tag{1}$$

We will consider no restriction at all on the size of every subset A_i of the partition, but it must be noted that any constraint imposed on the problem can be easily adapted to the neural model.

3 The Neural Model

To solve the MaxCut problem, we have used the MREM neural model since this model has been successfully used for other combinatorial optimization problems [6, 7].

The MREM neural model: It consists in a series of multivalued neurons, where the state of i-th neuron is characterized by its output (s_i) that can take any value in any finite set $\mathcal{M}$. This set can be a non numerical one, but, in this paper, the neuron outputs only take value in $\mathcal{M} \subset \mathbb{Z}^+$.

The state vector $\vec{S} = (s_1, s_2, \ldots, s_N) \in \mathcal{M}^N$ describes the network state at any time, where N is the number of neurons in the net. Associated with any state vector, there is an energy function $E : \mathcal{M}^N \to \mathbb{R}$, defined by the expression:

$$E(\vec{S}) = \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} w_{i,j} f(s_i, s_j) \tag{2}$$

where $W = (w_{i,j})$ is a matrix, $f : \mathcal{M} \times \mathcal{M} \to \mathbb{R}$ is usually a similarity function since it measures the similarity between the outputs of neurons i and j. At each step, the state vector will be evolving to decrease the energy function.

To solve the MaxCut problem with this neural net, we need as many neurons as number of nodes N in the graph. Each neuron taking value $s_i \in \mathcal{M} = \{1, 2, \ldots, K\}$ points to the subset of the partition where the i-th node is assigned to.

The cost function of the K-MaxCut problem, given by Eq. (1), must be identified with the energy function of Eq. (2). So, for the general MaxCut problem, $w_{i,j} = c_{i,j}$ and $f(x, y) = \delta_{x,y}$ (Kronecker delta function), since it is equivalent to maximize the cost of the edges cut by the partition and to minimize the cost of the edges whose endpoints lie within the same group of the partition.

Initially, the state of the net is randomly selected from a subset $\mathcal{F} \subset \mathcal{M}^N$. At any time, the net is looking for a better solution than the current one, in terms of minimizing the energy function. To this end, multiple dynamics can be defined for the net, and we will discuss them in the next section.

4 Neural Dynamics for MaxCut Problem

In this work, a simple but powerful dynamics, named best-2, has been implemented.

best-2: It consists in getting the greatest decrease of the energy function just by changing the state of only two neurons at each time.

1. A state for the net is initially randomly assigned.
2. Repeat until no change in state vector:
   (a) The scheduling selects a value $d \in \{1, \ldots, \lfloor \frac{N}{2} \rfloor\}$.
   (b) The following can be made parallel: every neuron p studies all possibilities of changing neurons p and $q = (p + d) \mod (N)$, with $0 < q \leq N$, i.e., p computes the potential associated to the possible changes, and it is stored as a vector $\vec{u}_p \in \mathbb{R}^K$ whose K components are the decrease of energy associated to every possible change in the output of these two neurons, by applying the expression $-\Delta E = \sum_{i=1}^{N} (\Delta_{i,p} + \Delta_{i,q}) - \Delta_{p,q}$ where $\Delta_{i,j} = w_{i,j} \left(f(s_i, s_j) - f(s'_i, s'_j) \right)$, denoting $s_i(t) = s_i$ and $s_i(t+1) = s'_i$.
   (c) Neuron p computes $\vec{\alpha}(p) = \max \vec{u}_p$, associated to a state $\vec{\tilde{S}}_p$.
   (d) The scheduling selects the next state of the net, $\vec{S}(t+1) = \vec{\tilde{S}}_p$ for which $p = \arg\max \vec{\alpha}$.

5 Functional Annealing

In this Section, despite its neural application, we will rigorously present this optimization method, Functional Annealing (FA), giving basic theorems and results guaranteeing its convergence, although not including proofs, due to the restriction in the length of this paper.

5.1 Theoretical Foundations

Suppose that a function $F : V \to \mathbb{R}$ is to be minimized, where V is a discrete set (not necessarily numerical). We will study the possibility of some slight modifications to this function F in order to make easier its minimization.

To this end, a sequence $\{F_n\}_{n \geq 1}$ of functions defined over the same set V is considered. The only hypothesis to be verified by F_n is that for each $x \in V$, $F_n(x) \to F(x)$.

We will assume that an iterative optimization technique is used to minimize each of the functions F_n, starting from a random point $x_1^{(n)}$ and creating a sequence $x_k^{(n)}$ such that que $F_n(x_k^{(n)}) \geq F_n(x_{k+1}^{(n)})$, that verifies $\lim_{k \to \infty} x_k^{(n)} = x_*^{(n)}$.

By using this way of building sequence $x_k^{(n)}$, convergence of FA is assured by the next result:

Theorem 5.1 *Under the above assumptions, if we let $x_1^{(n+1)} = x_*^{(n)}$, with a random $x_1^{(1)}$, then the sequence $F_n(x_*^{(n)})$ is convergent and there exists x_* with $F(x_*) = \lim_{n \to \infty} F_n(x_*^{(n)}) = \lim_{n \to \infty} F(x_*^{(n)})$.*

This result states that if $x_*^{(n)}$ is taken as initial guess to minimize function F_{n+1}, that is, $x_1^{(n+1)} = x_*^{(n)}$, then the value of the successive objective functions evaluated at $x_*^{(n)}$ converges. Its limit is also a value of the original objective function F.

A trivial result states that if $x_*^{(n)}$ is a global minimum of F_n for all $n \geq n_0 \in \mathbb{N}$, then x_* is global minimum of F. This global result is hard to be verified. Some less restricting results are presented next:

Lemma 5.2 *Given $x, x' \in V$ with $F(x) < F(x')$, there exists n_0 such that if $n \geq n_0$ then $F_n(x) < F_n(x')$.*

Corollary 5.3 *There exists $n_0 \in \mathbb{N}$ such that for all $x, x' \in V$ with $F(x) < F(x')$, it is obtained $F_n(x) < F_n(x')$ for all $n \geq n_0$.*

Proposition 5.4 *There exists $n_0 \in \mathbb{N}$ such that if $n \geq n_0$, then $F_n(x) \leq F_n(x')$ implies that $F(x) \leq F(x')$.*

For this reason, it will not be necessary (in practice) to build the whole sequence F_n, but it will suffice to minimize until a certain degree of approximation indicated by F_{n_0}. In addition, this last result implies the following one:

Corollary 5.5 *If $x_*^{(n)}$ is a local minimum of F_n for all n greater than a certain $n_0 \in \mathbb{N}$, then x_* is a local minimum of F.*

It must be noted that every convergence result above only states the convergence of $\{F_n(x_*^{(n)})\}$, to reach the conclusion of $F(x_*)$ being a local minimum of F. But, under these very general assumptions, it is not strictly true that $x_*^{(n)} \to x_*$. An additional hypothesis must be assumed to this end.

Proposition 5.6 *If every local minimum of F is strict, then $x_*^{(n)}$ is convergent, and its limit is a local minimum of F.*

So, in practice, it is guaranteed that x_* is a local minimum of F and, since the sequence $F(x_*^{(n)})$ is not necessarily decreasing, it is expected that this technique escapes from certain local minima, improving so the efficiency of the algorithm.

5.2 Application of 'Functional Annealing' to Neural Networks

Analogously to what has been explained in the previous section, since the function to be minimized in case of recurrent neural networks is the energy function E, it can be identified to the objective function F. So, the idea is to look for a sequence E_n (analogue to F_n), converging to E, defined by Eq. (2), but easier to minimize than E. In fact, every E_{n+1} should be a little more difficult to minimize than the previous E_n.

Although this technique can be easily applied to Hopfield discrete network, it is more useful to get based on the model MREM previously described, since it is a more general model than Hopfield's one, and solutions to many problems, including MaxCut, are better represented with this multivalued model than with Hopfield's.

The sequence of approximated energy functions we will consider is given by:

$$E_n(\vec{S}) = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} w_{i,j}^{(n)} f^{(n)}(s_i, s_j) \tag{3}$$

where $W^{(n)} = (w_{i,j}^{(n)})$ is a sequence of matrices verifying $W^{(n)} \to W$, and $f^{(n)} : \mathcal{M} \times \mathcal{M} \to \mathbb{R}$ is a sequence of functions such that $f^{(n)}(x, y) \to f(x, y)$ for all $x, y \in \mathcal{M}$. Obviously, $E(\vec{S}) = \lim_{n\to\infty} E_n(\vec{S})$ for every state vector $\vec{S} \in \mathcal{M}^N$.

The dynamics used to minimize E_n is exactly the proposed for the MREM model above, best-2.

So, for every n, given $\vec{S}_1^{(n)}$, a sequence $\{\vec{S}_i^{(n)}\}_{i\geq 1}$, convergent to a vector $\vec{S}_*^{(n)}$, is obtained.

Theorem 5.7 *Let $\vec{S}_1^{(n+1)} = \vec{S}_*^{(n)}$, $\vec{S}_1^{(1)}$ randomly chosen, then the sequences $\{E_n(S_*^{(n)})\}$ and $\{E(S_*^{(n)})\}$ converge to $E(\vec{S}_*)$ for some state vector $\vec{S}_* = \lim_{n\to\infty} \vec{S}_*^{(n)}$, and $E(\vec{S}_*)$ is a local minimum of the energy function E..*

6 Experimental Results

For the MaxCut problem to be solved only a finite number of modified energy functions is considered: $\{E_1, E_2, \ldots, E_{n_{\text{app}}+1} = E\}$, where all weight matrices are equal to W ($W^{(n)} = W$) and similarity functions are defined by the following expression:

$$f^{(n)}(x,y) = \begin{cases} 1 & x = y \\ -\alpha_n & x \neq y \end{cases} \quad \Rightarrow f^{(n_{\text{app}}+1)}(x,y) = f(x,y) = \delta_{x,y}$$

where $\alpha_n > 0$ and $\alpha_n \to 0$. For example, $\alpha_n = \frac{-1}{n_{\text{app}}}(n - n_{\text{app}} - 1)$.

This election reinforces the fact that high-cost arcs must be cut, that is, high-weighted edges will tend to have their endpoints in different subsets of the resulting partition.

In order to show the advantage of applying FA to solve MaxCut, some random test graphs have been considered. Each of them was formed based on two parameters, $N \in \{20, 50, 100, 150, 200\}$ (the cardinality of the set of vertices), and $\rho \in \{0.05, 0.25, 0.5, 0.75, 0.9\}$ (the density of edges in the graph, meaning that n_e is approximately $\rho\frac{N(N-1)}{2}$). Weights for edges were integers randomly chosen in $[1, 5]$. For this set to be complete, the values for the parameters were chosen to cover a wide range of graphs. In this case, $K = 2$, that is, we have considered bipartitioning. Experimental results are shown in table 1. In this case, we have considered $n_{\text{app}} = 5$.

We can see that Wang's method presents the worst results on average, while FA produces the best ones. It must be noted that best-2 makes MREM a very powerful algorithm, as well as OCHOM, so even a little improvement of FA over simple MREM becomes very important.

It must also be noted that time in FA is not multiplied by 6 when compared to MREM. The time spent by FA is on average about 3.3 times the spent by MREM.

7 Conclusions and Future Work

In this work we have presented a new general optimization technique that allows to avoid certain local minima, improving considerably the quality of the solutions obtained by several algorithms.

Its theoretical foundations have been proposed, based on very applicable and general enough results, which could be the basis for a more general optimization theory. In addition, its neural application has been presented, showing its possible use in this case.

To test this new technique, the well-known MaxCut Problem has been chosen, since it is an important benchmark for combinatorial optimization algorithms, presenting many methods for its approximated resolution, some of them being very powerful, as MREM or OCHOM.

		best-2		FA		OCHOM		Wang	
N	ρ	Opt	Av	Opt	Av	Opt	Av	Opt	Av
20	0.05	42.9	42.79	42.9	42.87	42.9	42.11	42.9	31.51
	0.25	49	48.71	49	48.79	48.7	47.95	49	35.06
	0.50	54.8	54.42	54.8	54.59	54.8	53.26	53.9	37.77
	0.75	60.7	60.02	60.7	60.25	60.7	59.06	60.7	40.88
	0.90	64.4	63.93	64.4	64.01	63.9	62.84	64.2	46.45
50	0.05	51.1	50.85	51.1	50.94	51.1	50.1	51	38.95
	0.25	86.8	85.63	86.8	85.84	86.1	83.59	86.7	51.98
	0.50	120.3	119.26	120.3	119.63	120.3	117.94	120.1	92.67
	0.75	154.3	152.73	154.3	152.81	153.5	150.78	154.3	109.52
	0.90	173.7	171.62	173.7	171.99	173.4	171.11	171.9	82.24
100	0.05	80.4	79.34	80.6	79.88	79	77	78.8	57.72
	0.25	207.1	203.79	207.1	204.09	207.4	201.52	202.5	143.26
	0.50	345.7	342.92	347	343.65	345	340.5	345	182.03
	0.75	475	471.45	475	472.19	475	468.29	473.6	333.03
	0.90	534	532.03	534.7	532.59	535.6	529.86	531.2	323.28
150	0.05	127.9	126.16	127.9	126.7	124.5	121.95	125.2	101.6
	0.25	386.5	382.89	387.8	384.4	384.3	377.76	380.2	267.08
	0.50	690.6	685.03	690.6	685.4	688.6	683.1	686.8	416.94
	0.75	990.1	985.56	991.1	986.93	990	985.92	990.1	599.64
	0.90	1157.1	1153.15	1157.1	1153.56	1154.3	1150.67	1148.8	694.86
200	0.05	194.7	192.32	195.5	193.87	192.9	187.15	192.6	163.5
	0.25	6075	6009	6080	6040.7	6020	5966.7	6052	2413.2
	0.50	11260	11194.7	11303	11228.5	11269	11206.7	11113	6667.8
	0.75	16650	16579.5	16677	16601.1	16599	16529.1	16498	13198.4
	0.90	19625	19520.8	19625	19532.6	19555	19439.5	19430	9715

Table 1: 2-MaxCut comparative results.

References

[1] A. Alberti, A. Bertoni, P Campadelli, G. Grossi and R. Posenato. A neural algorithm for MAX-2SAT: performance analysis and circuit implementation, *Neural Networks*, 10-3:555-560, 1997.

[2] F. Barahona, M. Grotschel, M. Junger and G. Reinelt, An Application of combinatorial optimization to statistical physics and circuit layout design. *Operat. Research*, 36:493-513, 1988.

[3] A. Bertoni, P. Campadelli and G. Grossi, An approximation algorithm for the maximum cut problem and its experimental analysis. In Proceedings: *Algorithms and Experiments*, 9-11:137:143, 1998.

[4] G. Galán-Marín and J. Muñoz-Pérez. Design and Analysis of Maximum Hopfield Networks. *IEEE Trans. on Neural Networks*, 12(2):329-339, 2001.

[5] M.R. Garey and D.S. Johnson, *Computers and Intractability. A guide to the theory of NP-Completeness*. W. H. Freeman and Company, New York, 1979.

[6] E. Mérida-Casermeiro, G. Galán-Marín and J. Muñoz-Pérez. An Efficient Multivalued Hopfield Network for the Traveling Salesman Problem. *Neural Processing Letters* 14:203-216, 2001.

[7] E. Mérida-Casermeiro, J. Muñoz-Pérez and R. Benítez-Rochel. Neural Implementation of Dijkstra's Algorithm. Lecture Notes in Computer Science 2686, pages 342-349, Springer-Verlag, 2003.

[8] E. Mérida-Casermeiro and D. López-Rodríguez, Graph Partitioning via Recurrent Multivalued Neural Networks. In J. Mira and A. Prieto, editors, proceedings of IWANN 2005, Lecture Notes in Computer Science 3512, pages 1149-1156, Springer-Verlag, 2001.

[9] Y. Takefuyi and J. Wang, editors. *Neural computing for optimization and combinatorics.* Singapore, World Scientific, 3, 1996.

[10] J. Wang and Z. Tang. An improved optimal competitive Hopfield network for bipartite subgraph problems. *Neurocomputing*, 61:413-419, 2004.