Graph Partitioning via Recurrent Multivalued Neural Networks

Enrique Mérida-Casermeiro and Domingo López-Rodríguez

Department of Applied Mathematics,
University of Málaga, Málaga, Spain
{merida, dlopez}@ctima.uma.es

Abstract. In this work, the well-known Graph Partitioning (GP) problem for undirected weighted graphs has been studied from two points of view: maximizing (MaxCut) or minimizing (MinCut) the cost of the cut induced in the graph by the partition. An unified model, based on a neural technique for optimization problems, has been applied to these two concrete problems. A detailed description of the model is presented, and the technique to minimize an energy function, that measures the goodness of solutions, is fully described. Some techniques to escape from local optima are presented as well. It has proved to be a very competitive and efficient algorithm, in terms of quality of solutions and computational time, when compared to the state-of-the-art methods. Some simulation results are presented in this paper, to show the comparative efficiency of the methods.

1 Introduction

In classical literature, the MinCut (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 *minimum (maximum) cut* of G, i.e., a partition of V into two sets that minimizes (maximizes) the total cost of the edges with endpoints in different sets.

These problems arise in the resolution of many practical or theoretical situations. Some examples include:

– For MinCut: network reliability theory (if $c_{i,j}$ is the probability of a network edge to fail, the minimum cut ensures the minimum risk of network disconnection) [17, ?], design of compilers (communication costs must be minimized in order to reduce the swap with memory) [4, 8].
– For MaxCut: pattern recognition, clustering, statistical physics and the design of communication networks, VLSI circuits and circuit layout [2].

So, these problems are well-known in literature. Due to their wide applicability, many variants of these problems have been formulated, placing restrictions on the original formulation.

The original problems, with all the variants, are known to be NP-complete [6], making their resolution computationally intractable, but in the case of planar graphs they belong to P, that is, there exists a solution in polynomial time. So, many algorithms have appeared to tackle GP problems in the general case.

Originally, MinCut was believed to be a variant of the s-t minimum cut problem, which adds the restriction of being s and t in different sets of the partition. In the early 60's, Gomory and Hu showed that a minimum cut in G can be estimated with $N-1$ s-t minimum cut computations, see [7]. So, most of the algorithms to solve MinCut are based in the max-flow/min-cut theorem [5], which implies that a s-t max-flow solution induces a s-t min-cut solution. The efforts were then focused, for a long time, in solving max-flow problems.

In 1989, Nagamochi and Ibaraki [16] presented an algorithm that did not make use of max-flow computations. In 1993 and later in 1996, Karger et al. [9, 10] presented a class of randomized algorithms, based on the notion of edge contraction, that can find all minimum cuts with probability $1-\frac{1}{N}$.

In the recent years, METIS [11] has become one of the most powerful algorithms for this problem, achieving the best results when compared to other methods, and in a very low computational time, as proved by [20]. To our knowledge, no neural algorithm has been presented to tackle MinCut.

In 1997, Alberti et al. presented a type-Hopfield neural model for MaxCut [1], but its performance is worse than the presented by Bertoni et al [3]. Takefuyi and his colleagues [18] 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. Recently, Galán-Marín et al. 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.[19] have recently proposed a stochastic dynamics for OCHOM, permitting temporary decreases of the objective function.

In this work, we want to present a neural model, based on a recurrent network, that has been proved to get very good results in some combinatorial optimization problems, see for example [12, 13, 14, 15], allowing K-partitioning of a graph.

Note that there exists very few bibliographic references for K-partitioning (most of the references is focused in bipartition). For MinCut, only METIS and the algorithm proposed in [8] consider that possibility, no one for MaxCut.

In the next section, we will give a detailed description of GP problem , and the two variants studied in this work, MinCut and MaxCut.

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 Minimum Cut Problem (MinCut): 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 minimum.

Generalization of the MinCut Problem (K-MinCut): 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 minimized is

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

With this formulation, the trivial solution is $A_1 = V$, $A_i = \emptyset$ for $i = 2, \ldots, K$. So, some constraints have to be made in order to make this problem more interesting. In this work, we have considered a restriction to the cardinality of the subsets A_i: the number of nodes in each group A_i is constrained to be N_i, such that $\sum_{i=1}^{K} N_i = N$.

MaxCut and **K-MaxCut** are defined in a similar way: find a partition (K-partition) of V such that the cost given by the expression in (1) is maximum. Contrary to K-MinCut, there are no need for constraints in the definition of K-MaxCut.

3 The Neural Model

In order to solve the GP problem, we have used the MREM neural model since this model has been successfully used for other combinatorial optimization problems [12, 13, 14, 15].

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 $\boldsymbol{S} = (s_1, s_2, \ldots, s_N) \in \mathcal{M}^N$ 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:

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

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 GP 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-MinCut and K-MaxCut problems, given by (1), must be identified with the energy function of (2). So, for the general GP, $w_{i,j} =$

$c_{i,j}$, and $f(x,y) = 1 - \delta_{x,y}$ (Krönecker delta function) for the K-MinCut and $f(x,y) = \delta_{x,y}$ for the K-MaxCut, 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 Implementation for GP Problem

In this work, a simple dynamics, named best-2, has been firstly implemented, and then it has been combined with two methods to improve solutions: best-3 and the shake phase.

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. So, a set of neighboring states must be defined. If neurons to be changed are p and q, this set will be named $\mathcal{N}_{p,q}$. Then, if $\boldsymbol{S}(t)$ is the state of the net at time t, $\boldsymbol{S}(t+1)$ will be the vector from a $\mathcal{N}_{p,q}$ that maximizes the decrease of energy, $-\Delta E$.

An expression for the decrease of energy is here given in order to reduce the computational cost of the model. Suppose that neurons p and q are going to be changed, and that we denote $s_i(t) = s_i$ and $s_i(t+1) = s'_i$ for all i. Then, the decrease of energy is given by:

$$U_{p,q} = -\Delta E = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} w_{i,j}\left(f(s_i,s_j) - f(s'_i,s'_j)\right) = \sum_{i=1}^{N}(\Delta_{i,p} + \Delta_{i,q}) - \Delta_{p,q} \tag{3}$$

(provided the symmetry of function f), where $\Delta_{i,j} = w_{i,j}\left(f(s_i,s_j) - f(s'_i,s'_j)\right)$.

So, the dynamics best-2 can be summarized as follows:

1. A state for the net is initially randomly assigned.
2. Repeat until no change in state vector:
   (a) The scheduling selects a value $d \in \{1,\dots,\lfloor\frac{N}{2}\rfloor\}$. For $d > \lfloor\frac{N}{2}\rfloor$, all of the following computations are made twice, and this way we can save some computational effort.
   (b) The following can be made parallel: every neuron p studies all possibilities of changing neurons p and $q = (p+d) \mod (N)$, with $0 < q \le N$, i.e., p computes the potential associated to the possible changes, it is stored as a vector $\boldsymbol{u_p}$ whose components are the decrease of energy associated to any vector in $\mathcal{N}_{p,q}$, by applying (3).
   (c) Neuron p computes $\boldsymbol{\alpha}(p) = \max \boldsymbol{u_p}$, associated to a state $\widetilde{\boldsymbol{S}}_{p,q} \in \mathcal{N}_{p,q}$.
   (d) The scheduling selects the next state of the net, $\boldsymbol{S}(t+1) = \widetilde{\boldsymbol{S}}_{p,q}$ for which $p = \arg\max \boldsymbol{\alpha}$.

In order to achieve better solutions, a pair of techniques has been developed.

best-3: It is an extension of best-2, allowing changes of three neurons outputs. In this case, if the neurons to be changed are p, q and r, a neighborhood $\mathcal{N}_{p,q,r}$ must be defined, and the associate expression for the decrease of the energy is: $U_{p,q,r} = U_{p,q} + \sum_{i=1}^{N} \Delta_{i,r} - (\Delta_{p,r} + \Delta_{q,r})$. And the dynamics of best-3 can be easily derived from best-2.

So, the scheme of the complete process is as follows: iterate best-2 until achieving a solution, then iterate best-3 once. If the solution has been improved, start with best-2 again and repeat the process.

Shake phase: Given an estimated solution, a good solution for MaxCut (MinCut) usually has the following property: *"High(Low)-weighted arcs must have their endpoints in different subsets"*.

So, we can study the high(low)-weighted arcs. Let A be the set of arcs with weights greater (lower) than a threshold and endpoints in the same group. Let $V^* \subset V$ be the set of endpoints of arcs in A. Then the current solution is saved and the shake phase begins. It consists in:

– Selecting the nodes that are endpoints of arcs in A and their neighbors: $H = \{v_i / \exists v_j \in V^*, e_{i,j} = 1\}$.
– Nodes in $V - H$ are clamped to their current values, while nodes in H are randomly assigned.
– With this new initial state vector, the network evolves with the usual dynamics (best-2), but only nodes in H will be selected in order to be modified, until a new stable state is reached.
– This new solution is compared with the previous saved one and the best one is selected.

Although the shake method can be used to improve the solutions of both K-MaxCut and K-MinCut, in this work it has been tested for K-MaxCut, as an example.

So, we must concrete the definitions of $\mathcal{F}$ and the neighbors $\mathcal{N}_{p,q}$ and $\mathcal{N}_{p,q,r}$ for each problem (K-MinCut and K-MaxCut):

– **For K-MinCut:**
  1. $\mathcal{F} = \{\boldsymbol{S} \in \mathcal{M}^N$: exactly N_i of its components are equal to $i, \forall i = 1, \ldots, K\}$ is the set of feasible solutions for this problem. Therefore, the net must begin in a feasible state and must be kept inside $\mathcal{F}$ in any time.
  2. **best-2**: The only alternative for changing the states of neurons p and q, and to remain in $\mathcal{F}$ is swapping their outputs. So, if S is the current state vector of the net, we have $\mathcal{N}_{p,q} = \{\boldsymbol{S}, \overline{\boldsymbol{S}}\}$, where $\overline{\boldsymbol{S}}(p) = \boldsymbol{S}(q)$, $\overline{\boldsymbol{S}}(q) = \boldsymbol{S}(p)$, and $\overline{\boldsymbol{S}}(m) = \boldsymbol{S}(m)$ for all $m \notin \{p, q\}$. Note that in this case, we have $\Delta_{p,q} = w_{p,q}\left(f(s_p, s_q) - f(s_q, s_p)\right) = 0$.
     best-3: Something similar occurs when dealing with $\mathcal{N}_{p,q,r}$: only some permutations of the outputs of neurons p, q and r are allowed because many of them are included in some $\mathcal{N}_{p,q}$. This gives $\mathcal{N}_{p,q,r} = \{\boldsymbol{S}, \boldsymbol{S_1}, \boldsymbol{S_2}\}$, where $\boldsymbol{S}(m) = \boldsymbol{S_1}(m) = \boldsymbol{S_2}(m)$ for $m \notin \{p, q, r\}$, and $\boldsymbol{S_1}(r) = \boldsymbol{S_2}(q) = \boldsymbol{S}(p)$, $\boldsymbol{S_1}(p) = \boldsymbol{S_2}(r) = \boldsymbol{S}(q)$ and $\boldsymbol{S_1}(q) = \boldsymbol{S_2}(p) = \boldsymbol{S}(r)$.

- **For K-MaxCut:**
  1. Now, $\mathcal{F} = \mathcal{M}^N$, there are no restrictions in this case, so any state is feasible.
  2. The neighboring states are now defined as in the most general case: if current state is S, $\mathcal{N}_{p,q}$ and $\mathcal{N}_{p,q,r}$ include all possible states from $\mathcal{M}^N$ that differ from S only in the outputs of neurons p or q (or both), and p, q or r, respectively. There are K^2 vectors in $\mathcal{N}_{p,q}$, and K^3 in $\mathcal{N}_{p,q,r}$.

Some experimental results for the dynamics herein proposed are shown in the next section.

5 Simulation Results

In this work, we have tested our algorithms with the two problems exposed in the text.

A test set was formed by 240 random graphs depending on two parameters, $N \in \{20, 50, 80, 100\}$ (the cardinality of the set of vertices), and $\rho \in \{0.05, 0.15, 0.25\}$ (the density of edges in the graph, meaning that $n_e \approx \rho\frac{N(N-1)}{2}$). Weights for edges were integers randomly chosen in $[0, 5]$. For this set to be complete, the values for the parameters were chosen to cover a wide range of graphs.

For K-MinCut, we have compared our model with METIS [11], every algorithm implemented in MatLab on a Pentium IV (3.06 Ghz). In Table 1, column labelled BEST-2 shows the results of applying only the dynamics best-2, and the column labelled BEST-3, presents the results of applying the composition of best-2 and best-3, as exposed in Sec. 4.

It can be verified that our model outperforms METIS in many cases. Note that METIS is a heuristic that always produces the same solution, while the repetitive use of best-2, best-3 always obtains good solutions that can be improved with new executions.

Table 1. Best and average performance on test set over 10 runs

		BEST-2			BEST-3			METIS	
N	ρ	Best	Av.	t	Best	Av.	t	Best/Av.	t
20	0.05	0	0.405	0.0006	0	0.395	0.0111	0	0.016
20	0.15	6.3	8.945	0.0007	6.05	8.21	0.012	7.15	0.0005
20	0.25	18.2	21.75	0.0007	17.75	19.965	0.0132	19.4	0.0005
50	0.05	9.35	14.145	0.0021	8.9	13.98	0.0928	9	0.0005
50	0.15	86.1	95.64	0.0019	81.75	91.435	0.1283	90.8	0
50	0.25	180.25	194.025	0.0023	174.5	186.38	0.1428	187.75	0.002
80	0.05	42.5	51.665	0.0032	40.8	50.35	0.2974	47.95	0.0005
80	0.15	269.95	286.61	0.0039	263.2	279.24	0.3998	283.05	0.0015
80	0.25	535.7	556.09	0.003	526.2	544.54	0.4663	551.6	0.0035
100	0.05	82.8	95.66	0.0046	80.3	93.725	0.5178	86.55	0
100	0.15	452.1	472.58	0.0046	440.95	461.68	0.7451	472.6	0.003
100	0.25	869.5	898.165	0.0047	856.75	883.09	0.8182	902.5	0.004

Table 2. Best and average performance on test set

		Wang			OCHOM			MREM			MREM-shake		
N	ρ	Best	Av.	t	Best	Av.	t	Best	Av.	t	Best	Av.	t
20	0.05	26	21.8	0.003	26	24.4	0.001	26	25.4	0.024	26	25.4	0.023
20	0.15	67	29.0	0.002	69	66.5	0.001	69	68.0	0.027	69	68.0	0.027
20	0.25	80	63.6	0.002	86	78.5	0.001	86	84.4	0.024	86	84.4	0.025
50	0.05	144	113.4	0.016	142	137.4	0.005	149	143.5	0.243	149	143.5	0.265
50	0.15	278	248.8	0.015	273	264.6	0.005	284	276.7	0.234	284	277.0	0.369
50	0.25	460	397.9	0.012	476	448.8	0.006	469	460.6	0.244	472	463.9	0.482
80	0.05	270	238.0	0.031	266	258.6	0.011	279	271.5	0.713	279	271.5	1.025
80	0.15	715	702.5	0.034	739	712.4	0.014	754	735.3	0.943	754	742.2	1.954
80	0.25	1100	878.2	0.034	1106	1080.5	0.016	1117	1091.0	0.857	1117	1095.1	1.717
100	0.05	400	323.7	0.048	407	390.2	0.017	418	406.0	1.539	418	406.6	2.374
100	0.15	1071	843.6	0.068	1060	1029.1	0.023	1081	1062.3	1.629	1084	1068.5	3.257
100	0.25	1697	834.4	0.043	1728	1682.3	0.025	1741	1702.7	1.323	1741	1714.8	2.407

With respect to K-MaxCut simulations, we have compared our proposed algorithms to OCHOM and Wang's. All of them have been implemented and tested in MATLAB, on the same conditions as above. More specifically, Wang's network has been tested with its default parameter $\lambda = 30$. In the proposed model, the set A was built by including every edge $e_{i,j}$ whose cost $c_{i,j} > \overline{c} + 3\sigma$, where $\overline{c}$, σ are respectively the mean and the standard deviation of $c_{i,j}$. So, A is forced to include exclusively high-weighted edges.

Both best and average solutions obtained are shown in Table 2. So, we can verify that the proposed algorithm outperforms others, not only giving the best results, but even on average.

6 Conclusions

The aim of this work has been to present a neural model for the resolution of combinatorial optimization problems. In particular, it has been proved to be a good optimizer for some NP-complete problems, as seen in [12, 13, 14, 15].

Contrary to heuristics, producing always the same solution to the problem, with the neural approach developed in this work, improvement of solutions is feasible, because the initial state of the net can be changed in each execution, and the search for the optimum begins from a different point in the search space, avoiding some local optima.

Another important feature that is present in the model is that it allows the K-partitioning of graphs, while some other techniques are based in the bipartition. So, this model is applicable to more general situations than some other methods.

Two important techniques to escape for local optima have been exposed in this paper. When combined to the original dynamics, best-2, they improve substantially the quality of the achieved solution.

To end with, the parallelism included in the computation dynamics is a powerful tool to achieve very good results with very little time consumption.

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** (1997), 555-560.
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** (1988), 493-513.
3. A. Bertoni, P. Campadelli and G. Grossi, An approximation algorithm for the maximum cut problem and its experimental analysis. Proceedings: Algorithms and experiments. Trento, **9-11** (1998), 137-143.
4. S. Chatterjee, J. R. Gilbert, R. Schreiber, and T. J. Sheffler. Array Distribution in Data-Parallel Programs. In *Languages and Compilers for Parallel Computing*, Lecture Notes in Computer Science series, Springer-Verlag, **896** (1996), 76-91.
5. P. Elias, A. Feinstein, and C. E. Shannon. Note on Maximum Flow Through a Network. IRE Transactions on Information Theory, **IT-2** (1956), 117-199.
6. 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).
7. R. Gomory and T. C. Hu, Multi-terminal network flows, J. SIAM, **9** (1961), 551-570.
8. E. J. L. Johnson, A. Mehrotra and G. L. Nemhauser. Min-cut clustering. Mathematical Programming, **62 (1)** (1993), 133-151.
9. D. R. Karger. Minimum cuts in near-linear time. Proc. 28th Annual ACM Symposium on Theory of Computing, (1996) 56-63.
10. D. R. Karger and C. Stein. A new approach to the minimum cut problem. J. Assoc. Comput. Mach., **43(4)** (1996), 601-640.
11. G. Karypis and V. Kumar. Multilevel k-way partitioning scheme for irregular graphs. Journal of Parallel and Distributed Computing, **48(1)** (1998), 96-129.
12. D. López-Rodríguez and E. Mérida-Casermeiro. Matrix Bandwidth Minimization: A Neural Approach, in Proceedings of International Conference of Computational Methods in Science and Engineering, ICCMSE, **1** (2004), 324-327.
13. E. Mérida-Casermeiro, G. Galán-Marín and J. Muñoz-Pérez. An Efficient Multivalued Hopfield Network for the Traveling Salesman Problem. Neural Processing Letters **14** (2001), 203-216.
14. E. Mérida-Casermeiro, J. Muñoz-Pérez and R. Benítez-Rochel. Neural Implementation of Dijkstra's Algorithm. LNCS **2686** (2003), 342-349.
15. E. Mérida-Casermeiro and D. López-Rodríguez. Multivalued Neural Network for Graph MaxCut Problem, in Proceedings of International Conference of Computational Methods in Science and Engineering, ICCMSE, **1** (2004), 375-378.
16. H. Nagamochi and T. Ibaraki. Computing Edge-Connectivity in Multigraphs and Capacitated Graphs. SIAM J. Disc. Meth., **5** (1992), 54-66.
17. A. Ramanathan and C. Colbourn. Counting Almost Minimum Cutsets with Reliability Applications. Math. Prog., **39** (1987) 253-261.
18. Y. Takefuyi and J. Wang, Neural computing for optimization and combinatorics. Singapore, World Scientific, **3**, (1996).
19. Jiahai Wang and Zheng Tang. An improved optimal competitive Hopfield network for bipartite subgraph problems. Neurocomputing (In press).
20. D.Yang, Y. Chung, C. Chen and C. Liao. A Dynamic Diffusion Optimization Method for Irregular Finite Element Graph Partitioning, The Journal of Supercomputing, Kluwer Academic Publishers, **17**, (2000), 91-110.