A heuristic for max-cut in toroidal grid graphs

Claudio Gentile[1], Giovanni Rinaldi[1], Esteban Salgado[1,2], Bao Duy Tran[3]

[1] Istituto di Analisi dei Sistemi ed Informatica "Antonio Ruberti" – CNR, Italy
<name>.<surname>@iasi.cnr.it

[2] Department of Computer, Control and Management Engineering Antonio Ruberti, Sapienza University of Rome, Italy

[3] Ruprecht-Karls-University Heidelberg, Germany
baoduy.tran@uni-heidelberg.de

Abstract

The paper reports on some preliminary results obtained by using polynomial time algorithms to solve max-cut on some special graphs within a heuristic scheme known as subgraph sampling. Computational results are reported on toroidal graphs of about 100 000 nodes with edge weights generated from both a Gaussian and a bivariate uniform distribution.

Keywords : *max-cut, toroidal grid graphs, Ising spin glass, heuristics, subgraph sampling.*

1 Introduction

One of the most famous applications of the max-cut problem is the computation of the ground state of a spin glass under the Ising model, a well-studied issue in Statistical Physics.

Such a computation amounts to solving a quadratic unconstrained binary optimization or, equivalently, a max-cut problem, on a 2-dimensional $(L \times M)$ or on a 3-dimensional $(L \times M \times N)$ grid graph, whose size is typically very large in order to simulate a real physical system. The edge weights of such a graph are randomly generated either from a zero-mean Gaussian distribution or from a bivariate uniform distribution with values in the set $\{-1, 1\}$. Two are the requirements that the instances under consideration have to satisfy: a) M and N must be sufficiently large; b) to avoid the border effects that would deviate the behavior of the model from the one of a real system, the grid is "wrapped around" along each of the three dimensions. Thus, in 2D, for example, the graph to consider becomes a toroidal grid.

In the Physics community, these problems are solved with Monte Carlo techniques, which have two drawbacks: they tend to be slow to converge to an accurate solution when the graph sizes are large and do not provide any measure of how much the solution deviates from the true optimum.

Max-cut is solvable in polynomial time on planar graphs and 2D grids or 2D grids wrapped along only one direction are of this kind. Exact solutions for grid instances of sizes up to 3000×3000 are reported in [7]. The problem has been proven to be polynomially solvable on toroidal grids with bounded weights in [3], but the algorithm that achieves this result is not very practical for large instances. For example, solutions of problems up to only 50×50 are reported in [8]. Branch-and-cut methods have been proposed to solve the problem on toroidal grids (see, e.g., [6] for a survey). The largest instances reported in these studies are 150×150 for the Gaussian weights and 100×100 for the ± 1 ones. Therefore, there is a demand for fast and accurate heuristic methods that would make it possible to run simulations on a large sample of large instances and would improve the performances of the exact methods that usually benefit from the knowledge of a feasible solution of good quality.

In the context of the Markov Random Field problem, a heuristic was proposed in [4]. The idea of this heuristic, which is now referred to as the *subgraph sampling* method, was applied by

Selby in [11] to max-cut for a class of graphs known as the *Chimera graphs*. Selby's heuristic was extensively applied to solve max-cut on a large variety of instances of Chimera graphs in [5] and was shown to be very fast and able to find the optimum for all the instances of the test-bed. The central part of Selby's method relies on the ability to solve the problem on subgraphs of low tree-width efficiently. The motivation of this paper is to use the subgraph sampling method with a different type of subgraphs for which max-cut is polynomially solvable and to evaluate the efficiency of this new approach on toroidal grid instances with Gaussian and ± 1 edge weights.

2 Algorithms

Given a weighted graph $G = (V, E, w)$, the max-cut problem calls for a partition $(W : V \setminus W)$ of the node-set that defines an edge-cut of maximal weight. The two sets of the partition are called the *shores* of the cut. We represent the cut defined by $(W : V \setminus W)$ with a vector $x \in \{-1, 1\}^V$, where for $v \in V$, $x_v = 1$ if $u \in W$ and $x_v = -1$ otherwise. Then the weight of such a cut is given by

$$\sum_{ij \in E} w_{ij}(1 - x_i x_j)/2.$$

A variable transformation of the kind $x'_v = -x_v$, which is called *switching on node* v in the max-cut literature, transforms a max-cut instance into an equivalent one and will be very useful in the following because it serves a dual purpose:

a) suppose that at a given step of an algorithm we want to impose that all nodes of a subset $U \subset V$ belong to the same shore. We can apply the switching operation to some nodes of U so that all its nodes get the same x-value. At this point, we contract all nodes of U into a single node. The resulting graph, that we denote by G_U, has the property that all its cuts correspond to all cuts of G where all nodes of U belong to the same shore.

b) suppose that we need to flip the weight sign of the edges incident with a node v. Then switching on v achieves what we want.

We now outline the subgraph sampling scheme and the two heuristics that used it.

Subgraph sampling scheme.

(i) At the beginning of the algorithm, a random assignment of the nodes to the two shores is applied, i.e., a point $\hat{x} \in \{1, -1\}^V$ is randomly chosen.

(ii) A subset $U \subset V$ is "suitably" chosen. We then find the best among all cuts where the nodes in $V \setminus U$ are constrained to keep the value given in $\hat{x}$ and replace $\hat{x}$ with the value corresponding to such a cut. This is achieved by possibly applying the switching to some nodes of $V \setminus U$ and contracting the nodes of $V \setminus U$ to generate the graph $G_{V \setminus U}$. In order for the scheme to be of practical use, max-cut for the instance $G_{V \setminus U}$ has to be efficiently, or — even better — polynomially solvable.

(iii) Until a maximum number of non-improving iterations is not reached, select a new node-set $U \subset V$ and iterate step (ii).

(iv) Perturb the current vector $\hat{x}$ and repeat from step (ii) until no more improvements take place, despite perturbation.

(v) If additional computing time is allowed, the node assignment is randomly generated afresh and the process is restarted from step (ii). This way, the Borel-Cantelli lemma assures that the optimal solution is theoretically eventually attained.

(vi) Finally, return the best solution found during the whole process.

Even if this scheme is general, here we will apply it to 2-dimensional ($L \times M$) grid graphs generating two algorithms depending on the way the set U is chosen.

Planar subgraph. Given a row r, with node-set R, and a column c, with node-set C, we define $U = V \setminus \{R, C\}$. The successive selections of the set U are performed by "moving" the column and the row, i.e., $U' = V \setminus \{r', c'\}$ where $r' = (r+1) (\bmod L)$ and $c' = (c+1) (\bmod M)$. In this case, the resulting graph G_U is planar (a grid with an external node connected only to the boundary of the grid); therefore, max-cut can be solved in polynomial time on the respective G_U graph by finding a perfect matching in an auxiliary graph constructed in the same way as described in [7]. We observed that with sizes up to 100×100 the instances can be solved optimally using the algorithm described in [6] and very often *planar subgraph* solves them optimally, for example, for the bivariate case.

Negative subgraph. Let U be a non-empty node-set that induces an acyclic subgraph of G (U could even be made of a single node); then we define a *negative subgraph* $\mathcal{F}$ by the following procedure: Repeat until no more changes apply: pick a node in $v \in V \setminus U$ such that $N(v) \cap U \neq \emptyset$[1]. If the weights of all edges $(v, u) \in E$ for $u \in U$ have the same sign, add v to U.

It is easy to see that, by applying switching, we can make all edges of $\mathcal{F}$ to have a non-positive weight. Therefore, in G_U all edges, except those that are incident with the node obtained by the contraction of $V \setminus U$, have non-positive weight and max-cut can be solved in polynomial time, as shown in [9], using max-flow techniques.

The set $U = U(t, p)$ with $t \in \{1, \ldots, L\}$ and $p \in \{0, 1\}$ from which we start the construction of $\mathcal{F}$ is defined by the nodes of row t and column $(p + j) (\bmod M)$ for $j \in \{1, \ldots, M - 1\}$, by the nodes of row $(t + i) (\bmod L)$ for $i \in \{1, \ldots, L - 2\}$ and column $(p + j)$ for all even $j \in \{1, \ldots, M - 1\}$, and by the node defined by row $(t - 1) (\bmod L)$ and column $(p - 1) (\bmod M)$. The successive selections of the set U are performed by "moving" the parameters to $U(t', p')$ where $t' = (t+1) (\bmod L)$ and $p' = 1 - p$. Notice that the graph induced by $U(t, p)$ is a forest, thus it is acyclic. Likewise, interchanging rows with columns, another sequence of $U(t, p)$ sets is produced. Note that if $U(t, p)$ is defined as above, its complement also induces an acyclic subgraph. Therefore, the scheme can also be modified to solve max-cut exactly on both the graph generated by $U(t, p)$ and the one generated by its complement and then move to the next $U(t', p')$. With this modification, each step of the scheme is applied to the full graph solving two max-cut problems. We also notice that the sets generated by $U(t, p)$ and by its complement typically overlap, thus we are optimizing on overlapping subsets of V.

3 Numerical results

To measure the quality of these heuristics we tested them on 316×316 toroidal grids (with approximately 100 000 nodes), see Tab. 1. Tests were performed on a machine running `Ubuntu 18.04.4 LTS` with an Intel Xeon Gold 6136 Processor with 3.0 Ghz base frequency. The implementations described in [2] and in [1] were used for the matching and the max-flow algorithms, respectively. Graphs were randomly generated using `rudy` [10]. In the table header `p` indicates the proportion of positive/negative edges in a graph with ± 1 edge-weights, `s` denotes the seed and `G` denotes Gaussian weights. We use the value (`val`) obtained using *planar subgraph* (`P`) and compare it with the results of *negative subgraph* with and without the use of the complement. For each graph and method, we ran the algorithm 10 times and show the average (`avg`) number of iterations needed to obtain the best solution (`it`), the average (`avg`) value (`val`), the average (`avg`) and standard deviation (`std`) of the percentage relative difference with the *planar subgraph* (`%difP`) and the average (`avg`) time needed to obtain the optimal solution. For *negative subgraph* the maximum time was set to 60 minutes (the maximum time condition is verified at the beginning of each iteration).

[1] By $N(v)$ we denote the nodes of V adjacent to v.

		P		No complement				Complement			
p	**s**	**val**	**time**	**it (avg)**	**val (avg)**	**%difP (avg/std)**	**time (avg)**	**it (avg)**	**val (avg)**	**%difP (avg/std)**	**time (avg)**
40	1	89968	1612.79	6.70	89770.40	0.22 / 0.02	3681.97	3.20	89772.60	0.21 / 0.02	4013.31
	2	89978	1594.09	6.40	89768.80	0.23 / 0.01	3651.78	3.90	89779.00	0.22 / 0.01	4235.11
	3	90062	2438.43	5.70	89848.00	0.24 / 0.02	3499.65	2.80	89847.60	0.24 / 0.02	3857.10
	4	89952	1553.53	6.40	89731.60	0.25 / 0.01	3822.29	3.50	89735.80	0.24 / 0.02	3946.81
50	1	69984	1626.84	6.30	69773.20	0.30 / 0.02	3504.64	2.90	69780.00	0.29 / 0.01	3912.82
	2	70002	2444.74	6.50	69794.00	0.29 / 0.01	3701.30	3.30	69798.40	0.29 / 0.02	3977.30
	3	69956	1480.32	6.60	69741.60	0.31 / 0.02	3632.59	3.20	69749.00	0.30 / 0.02	3917.03
	4	70064	1473.25	6.50	69850.20	0.31 / 0.01	3812.63	3.30	69855.80	0.30 / 0.01	4048.55
60	1	49920	1609.70	7.40	49715.80	0.41 / 0.02	3717.88	3.60	49714.40	0.41 / 0.02	3850.08
	2	50094	1926.58	6.40	49879.20	0.42 / 0.02	3562.71	3.20	49888.80	0.40 / 0.02	3869.57
	3	50074	2485.99	6.70	49876.40	0.39 / 0.02	3662.16	2.90	49881.00	0.39 / 0.02	4035.61
	4	50000	1504.92	6.50	49791.40	0.42 / 0.01	3678.25	3.40	49804.80	0.39 / 0.02	4119.61
G	1	6528526722	1673.16	5.20	6457847194.60	1.08 / 0.02	3855.91	3.10	6452574600.70	1.16 / 0.03	4104.11
	2	6554507128	1688.89	5.50	6484463753.50	1.06 / 0.02	3871.09	3.30	6480230701.70	1.13 / 0.03	4067.86
	3	6548455479	1706.82	5.10	6476848337.70	1.09 / 0.03	3938.77	3.10	6473079819.30	1.15 / 0.03	4353.24
	4	6545055776	1659.93	5.20	6474477296.90	1.08 / 0.03	3870.45	3.50	6471527989.50	1.12 / 0.05	4297.28

TAB. 1: Results with and without complement for 316×316 toroidal grids.

As we can observe *negative subgraph* presents solutions that are comparable within a relative difference between 0.21 and 0.42 for graphs with edge-weights in $\{-1, 1\}$. When the weights are normally distributed errors rise up to 1.08. We also observe that the complement approach improves the solution on the $\{-1, 1\}$ case, which is not true with the Gaussian weights. A major difference between both approaches is that *negative subgraph* can be used for any graph.

References

[1] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *Transactions on Pattern Analysis and Machine Intelligence*, 26:1124–1137, 2004.

[2] W. Cook and A. Rohe. Computing minimum-weight perfect matchings. *INFORMS Journal of Computing*, 11:138–148, 1999.

[3] A. Galluccio, M. Loebl, and J. Vondrák. Optimization via enumeration: a new algorithm for the max cut problem. *Mathematical Programming*, 90:273–290, 2001.

[4] F. Hamze and N. de Freitas. From fields to trees. In *UAI '04 – Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, pages 243–250, Arlington, Virginia, 2004. AUAI Press. http://www.cs.ubc.ca/∼nando/papers/tree2.pdf.

[5] M. Jünger, E. Lobe, P. Mutzel, G. Reinelt, F. Rendl, G. Rinaldi, and T. Stollenwerk. Performance of a quantum annealer for Ising ground state computations on Chimera graphs. *arXiv preprint arXiv:1904.11965*, 2019.

[6] F. Liers, M. Jünger, G. Reinelt, and G. Rinaldi. Computing exact ground states of hard Ising spin glass problems by branch and cut. In Alexander K. Hartmann and Heiko Rieger, editors, *New Optimization Algorithms in Physics*, pages 47–68. Wiley-VCH, 2004.

[7] F. Liers and G. Pardella. Partitioning planar graphs: a fast combinatorial approach for max-cut. *Computational Optimization and Applications*, 51:323–344, 2012.

[8] J. Lukic, A. Galluccio, E. Marinari, O.C. Martin, and G. Rinaldi. Critical thermodinamics of the two-dimensional $\pm J$ Ising spin glass. *Phys. Rev. Lett.*, 92:117202–1 – 11702–4, 2004.

[9] S.T. McCormick, M.R. Rao, and G. Rinaldi. Easy and difficult objective functions for max cut. *Mathematical Programming*, 94:459–466, 2003.

[10] G. Rinaldi. Rudy, a graph generator, 1996. http://swmath.org/software/21923.

[11] A. Selby. Efficient subgraph-based sampling of Ising-type models with frustration. *arXiv e-prints*, page arXiv:1409.3934, 2014.