Tree-width driven SDP for MAX CUT problem

A Preprint

Sergei Anikin
Chair of Data Analysis
MIPT
Moscow, Russia
anikin.sd@phystech.edu

Alexander Bulkin
MSU
Faculty of Mechanics and Mathematics
Moscow, Russia
a.bulkin@iccda.io

Abstract

This paper addresses the well-known Max Cut problem, which has various applications both in machine learning and theoretical physics. The Max Cut problem is computationally NP-hard over general graphs. This paper presents a novel empirical approach aimed at enhancing the quality of Max-Cut approximations within polynomial time bounds. While the problem is tractable for graphs with small tree-width, its solution over general graphs often relies on Semi-Definite Programming or Lovász-Schrijver relaxations. We achieve the described improvement of approximation quality by combining relaxation approaches, the tree-width ideas and various heuristics described in the paper.

Keywords SDP · Treewidth · Max Cut · Lovász-Schrijver relaxations

1 Introduction

In this paper, we will discuss a non-asymptotic improvement of the solution to the MAX CUT problem $-$ the problem of finding the maximum cut in undirected graphs. It involves partitioning the vertices of a graph into two sets such that the number of edges between the two sets (the cut) is maximized. This problem has applications in many spheres, including machine learning, theoretical physics, and theoretical computer science. It serves as a basis for developing approximation algorithms and heuristic methods for solving other optimization problems. Currently, for graphs in general, the best solutions proposed by X. Goemans and David P. Williamson find a cut that contains at least approximately 88% of weight of the edges in the optimal cut [1]. There are families of graphs for which this bound is asymptotically optimal unless P = NP.

In this article, we focus on a non-asymptotic improvement of the solution in polynomial time on arbitrary graphs. The known solution [1] utilizes Semi-Definite Programming problems, and here, we present reasoning that allows solving them with greater accuracy by combining optimization ideas, tree-width approach, and heuristics. We decide on the efficiency of provided algorithm by comparing it with well-known ones using the different datasets[10 - 13].

2 Problem statement

We focus on weighted undirected graphs, where each edge (i, j) is assigned a weight w_{ij}. As the graph is undirected,$w_{ij} = w_{ji}$. Such graphs are represented as G = (V, E), where V denotes the set of vertices and E is the symmetric matrix with w_{ij} indicating the weight of edge (i, j). Later we will refer to weighted undirected graphs simply as graphs.

Given a fixed graph G = (V, E) with the sum of weights denoted by W, a cut in the graph is defined as a subset $S \subseteq V$. The complement of S is denoted by $T = V \setminus S$. Notably, a cut partitions the vertices into two sets: S and T. Additionally, the edges are divided into three categories: those entirely within S, those entirely within T, and those split by the cut, where one vertex lies in S and the other in T. Let's define W(S) to be the weight of the cut:

$$W(S) = \sum_{i \in S} \sum_{j \notin S} w_{ij}$$

Our goal is to find in polynomial time cut $S_{found} \subseteq V$, such that the value $W(S_{found})$ is as big as possible

$$W(S_{found}) \to \max_{S_{found}}$$

We decide on the efficiency of provided algorithm by comparing it with well-known ones using the different datasets [10 - 13].

3 Theory

We find the approximate solution using SDP solution. First of all, we show, how SDP problem is connected to Max Cut.

Let matrix L be L = $\begin{bmatrix} (\sum_{i=1}^n w_{1i}) & -w_{12} & -w_{13} & \dots & -w_{1n} \\ -w_{21} & (\sum_{i=1}^n w_{2i}) & -w_{23} & \dots & -w_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -w_{n1} & -w_{n2} & -w_{n3} & \dots & (\sum_{i=1}^n w_{ni}) \end{bmatrix}$

Later we call such matrix the Laplacian of the graph.

Then we state that $W(S) = \frac{1}{4} x \top Lx$, where $x_i = 1$ if $x \in S$ and $x_i = -1$ if $x \in T$. It is easy to see: the coefficient before each variable w_{ij} 1 in both sides of equality equals to 1 if i and j are from parts of partition and equals to 0 otherwise.

It means that our task is equivalent to finding

$$OPT = \max_{x_i^2=1} x^T Lx$$

since

$$\max_{S \subseteq V} W(S) = \frac{1}{4} \max_{x_i^2=1} x^T Lx$$

Let's look at the dual problem for OPT. We will also call SDP a problem relaxed by forgetting about the rank condition.

$$OPT = \max_{x_i^2=1} x^T Lx = \max_{\substack{x_i^2=1 \\ X=x^T x}} LX = \max_{\substack{X \succeq 0 \\ diag(X)=1_n \\ rank(X)=1}} \operatorname{Tr}(LX) \le \max_{\substack{X \succeq 0 \\ diag(X)=1_n}} \operatorname{Tr}(LX)$$

Let's find the dual for OPT problem.

$$Dual = \max_{\lambda} \min_{x} \sum_{i=1}^n \lambda_i (1 - x_i^2) - \sum_{i,j}^n x_i x_j L_{ij} = \max_{\lambda} \min_{x} \sum_{i=1}^n \lambda_i - \sum_{i,j}^n x_i x_j L_{ij} - \sum_{i=1}^n \lambda_i x_i^2$$

if $-L - Diag(\lambda) \not\succeq 0$ then

$$\min_{x} \sum_{i=1}^n \lambda_i - \sum_{i,j}^n x_i x_j L_{ij} - \sum_{i=1}^n \lambda_i x_i^2 = -\infty$$

since we can multiply the vector, which proves that $-L - Diag(\lambda) \not\succeq 0$, by constant and get the arbitrary small value.

And if $-L - Diag(\lambda) \succeq 0$, then

$$\min_{x} \sum_{i=1}^n \lambda_i - \sum_{i,j}^n x_i x_j L_{ij} - \sum_{i=1}^n \lambda_i x_i^2 = \min_{x} \sum_{i=1}^n \lambda_i$$

This means that if we denote $\xi_i := -\lambda_i$, Dual problem can be rewritten this way:

$$Dual = \max_{\lambda} \min_{x} \sum_{i=1}^n \lambda_i (1 - x_i^2) - \sum_{i,j}^n x_i x_j L_{ij} = \max_{\substack{\lambda: \\ -L - Diag(\lambda) \succeq 0}} \sum_{i=1}^n \lambda_i = \max_{\substack{\xi: \\ Diag(\xi) \succeq L}} \sum_{i=1}^n -\xi_i = \min_{\substack{\xi: \\ Diag(\xi) \succeq L}} \sum_{i=1}^n \xi_i$$

Later we will try to approximate Dual value using tree-width ideas, but first of all let's prove the following Lemma.

Lemma 3.1.

$$Dual = \min_{\substack{\xi: \\ Diag(\xi) \succeq L}} \sum_{i=1}^{n} \xi_i = \min_{L_T \succeq L} \max_{x: x_i^2 = 1} x^T L_T x = TreeRel$$

where L_T can be represented as $L_T = L_{tree} + Diagonal$, where L_{tree} corresponds to Laplacian of a tree graph and Diag is a diagonal matrix with non-negative values.

Proof. It is well-known, that tree is a bipartite graph and hence the MaxCut value equals to the total sum of edges in the graph

$$\max_{x: x_i^2=1} x^T L_{tree} x = 4 \sum_i \sum_{j>i} w_{ij}$$

Then it is easy to conclude, that if $L_T = L_{tree} + Diagonal$, then

$$\max_{x: x_i^2=1} x^T L_T x = 4 \sum_i \sum_{j>i} w_{ij} + \mathrm{Tr}(Diagonal)$$

Now we show inequalities between $Dual = \min_{\substack{\xi: \\ Diag(\xi) \succeq L}} \sum_{i=1}^{n} \xi_i$ and $TreeRel = \min_{L_T \succeq L} 4 \sum_i \sum_{j>i} w_{ij} + \mathrm{Tr}(Diagonal)$. $Dual \geq TreeRel$ is obvious since for each matrix $Diag(\xi) \succeq L$ it is possible to take $L_T = Diag(\xi)$ (which means that we simply take the tree with all the weights equal to 0.

Finally, $Dual \leq TreeRel$. In order to show that for each $L_T = L_{tree} + Diagonal \succeq L$ we can construct a vector ξ, such that $Diag(\xi) \succeq L$ and $4 \sum_i \sum_{j>i} w_{ij} + \mathrm{Tr}(Diagonal) = \sum_{i=1}^{n} \xi_i$. Let's take $\xi_i = Diagonal_{ii} + 2 \sum_{j=1}^{n} w_{ij}$. Then indeed

$$\sum_{i=1}^{n} \xi_i = 4 \sum_i \sum_{j>i} w_{ij} + \mathrm{Tr}(Diagonal)$$

Finally, we notice that $Diag(\xi) - L_T$ is SDP and hence $Diag(\xi) \succeq L_T \succeq L$ which completes the proof.

Let t_{ij} be the weight of an edge between vertexes i and j in the tree.

$$Diag(\xi) - L_T = \begin{bmatrix} (\sum_{i=1}^{n} t_{1i}) & t_{12} & t_{13} & \dots & t_{1n} \\ t_{21} & (\sum_{i=1}^{n} t_{2i}) & t_{23} & \dots & t_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ t_{n1} & t_{n2} & t_{n3} & \dots & (\sum_{i=1}^{n} t_{ni}) \end{bmatrix}$$

This matrix is symmetric and diagonally dominant with positive diagonal entries. It is known that such matrix is PSD. Or in this case it is obvious since

$$x^T (Diag(\xi) - L_T) x = \sum_{i=1}^{n} \sum_{j=i+1}^{n} t_{ij} (x_i + x_j)^2 \geq 0$$

□

Later we will refer to the Dual as SDP.

k-diagonal hierarchy Let's define k-diagonal hierarchy this way:

$$H_k = \min_{\substack{T: T = T^\top \succeq A \\ T \in (2k+1)-diag}} \max x^\top T x$$

Then we can see, that

$$OPT = H_n \leq \dots \leq H_1 = SDP$$

since inequalities are obvious, the first equality is true due to T = A being the optimal matrix for OPT and the second equality is true due to Lemma 3.1.

k-diagonal algorithm Now we are ready to describe the k-diagonal algorithm. First, we note that for fixed k if T is (2 k + 1)-diagonal matrix

$$\max_{x_i^2=1} x^\top Tx$$

can be solved in O(n) time by dynamic programming. It is possible to calculate a two-dimensional array dp, where for $1 \le i \le n$, $0 \le mask \le 2^{k-1} - 1$

$$dp[i][mask] = \max_{\substack{x_i^2=1 \\ x_{i-k+1},\dots x_{i-1},x_i=mask}} x^\top T_i x,$$

where 1)T_i, is $n \times n$ matrix which is made out of matrix T by setting all the values outside top left $k \times k$ matrix to zeros. 2) It is known, how this and the previous k - 1 vertexes are distributed between parts of cut: ones in $mask$ correspond to the vertexes from one part of the cut, and zeros in $mask$ correspond to the vertexes from another one. $dp[i][b_1, ..., b_n]$ can be easily recalculated by $dp[i][-1, b_1, ..., b_{n-1}]$, $dp[i][1, b_1, ..., b_{n-1}]$ and $L[i-k][i], L[i-k+1][i], ..., L[i][i]$

It means, that we can find the optimal value of H_k using gradient-free methods of optimization and solving $\max_{x_i^2=1} x^\top Tx$ with oracle, which uses described dynamic programming and works for O(n) time.

Finally, we can restore the final cut corresponding as the cut we get from this optimization.

4 Computational experiment

4.1 Data

We consider well known BiqMac dataset for testing our solution and comparing it with others. There are different types of graphs in this dataset:

1. g05-n.i For each dimension unweighted graphs with edge probability 0.5. n=60,80,100.
2. pm1s-n.i For each dimension weighted graphs with edge weights chosen uniformly from 0,1 and density 0.1. n=80,100
3. pm1d-n.i For each dimension weighted graphs with edge weights chosen uniformly from 0,1 and density 0.99. n=80,100
4. pwd-100.i For each density graphs with integer edge weights chosen from [0,10] and density d=0.1,0.5,0.9, n=100.

Additionally, it is essential to identify and extract word segments from certain documents to assess topic interpretability. Acquiring these specific data segments poses a challenge for the experiment due to the lack of a defined source.

4.2 Plan of experiment

1. Implement standard MaxCut solution

2. Implement new MaxCut solution

3. Compare the average ratio of cut value divided by total sum of edges with basic and new solutions using different sets of graphs

We refer to the solution described in [7] and implemented in [3] as basic solution, while the diag-dual solution refers to one solving problem which we call Dual.

(TODO: add k-diagonal results to the comparison table)

k-diagonal results provided significantly less quality than expected and show them

4.3 Comparison

Graph-type	Basic solution	Dual-diag solution
g05-60	0.6005	**0.6143**
g05-80	0.5931	**0.6121**
g05-100	0.5874	**0.6095**
pm1d-80	0.5684	**0.6030**
pm1s-100	0.5973	**0.6098**
pw01-100	**0.6212**	0.6174
pw05-100	**0.6171**	0.6160
pw09-100	0.6087	**0.6126**

As we can observe, dual-dig solution shows significant increase in accuracy on many different types of graphs.

5 Conclusion

This paper presents a new approach to finding maximum cut in the graph, using idea of approximating maxcut by tractable maxcuts of graphs, which laplacian may be presented as k-diagonal matrices. The accuracy of approximation of new approach compared with the well-known Goemans-Williamson solution.

References

TODO: change the format of all the references to proper one and change the lecture references to the

[1] Goemans-Williamson MAXCUT Approximation Algorithm by Jin-Yi Cai, Christopher Hudzik, Sarah Knoop, 2003: https://pages.cs.wisc.edu/ jyc/02-810notes/lecture20.pdf

[2] The Lovasz-Schrijver relaxation by Madhur Tulsiani, 2010: https://home.ttic.edu/ madhurt/Papers/ls.pdf

[3] https://github.com/pandrey-fr/maxcut

[4] Treewidth: https://www.cs.cmu.edu/ odonnell/toolkit13/lecture17.pdf

[5] MAX CUT approximation algorithm and UGC-hardness, Lecture by Irit Dinur and Amey Bhangale: https://www.wisdom.weizmann.ac.il/ dinuri/courses/19-inapprox/lec6.pdf

[6] Semidefinite Programming versus Burer-Monteiro Factorization for Matrix Sensing by Baturalp Yalcin, Ziye Ma, Javad Lavaei, Somayeh Sojoudi, 2022 https://arxiv.org/abs/2208.07469v1

[7] 0.878-approximation for the Max-Cut problem, Lecture by Divya Padmanabhan, 2022: https://www.iitgoa.ac.in/ sreejithav/misc/maxcut.pdf

[8] Rank optimality for the Burer-Monteiro factorization by Irène Waldspurger, Alden Waters, 2019 https://arxiv.org/abs/1812.03046

[9] Semidefinite relaxation and nonconvex quadratic optimization by Yury Nesterov, 1997 https://www.tandfonline.com/doi/abs/10.1080/10556789808805690

[10] Datasets: Texas Data Repository: https://dataverse.tdl.org/dataset.xhtml?persistentId=doi:10.18738/T8/VLTIVC

[11] Datasets: Biq Mac Library: https://biqmac.aau.at/biqmaclib.html

[12] Datasets: MaxCut and BQP Instance Library: http://bqp.cs.uni-bonn.de/library/html/index.html

[13] Datasets: MaxCut Instances: https://grafo.etsii.urjc.es/optsicom/maxcut.htmlbest-known-values

[14] Ellipsoid algorithm: https://www.cs.toronto.edu/ avner/teaching/S5-2411/ln/lecture8.pdf

[15] Michel X. Goemans and David P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. Journal of the ACM, 42(6):1115–1145, 1995