TREE-WIDTH DRIVEN SDP FOR THE MAX-CUT PROBLEM

A PREPRINT

Ivan Voronin
MIPT
Chair of Data Analysis
Moscow, Russia
voronin.ip@phystech.edu

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

ABSTRACT

This article discusses the well-studied Max-Cut problem in Graph Theory, which has found applications in various fields, particularly Machine Learning, Theoretical physics (the Ising model[17]), and VLSI design.
The initial problem is **NP**-complete, so it is of interest to propose a polynomial algorithm that gives a good approximation to the actual answer. For a long time, the best accuracy achievable by polynomial algorithms was a half of the optimal cut. Only in 1995 Goemans and Williamson introduced a probabilistic algorithm based on semidefinite relaxation for the Max-Cut problem with further decomposition of optimal matrix and randomized rounding[2]. This is the best possible approximation guarantee for the Max-Cut problem under the Unique Games Conjecture[12][7]. We propose a novel approach to solving the Max-Cut problem in the particular case where the graph's treewidth is bounded by a pre-fixed value.

Keywords Max-Cut · Unique Games Conjecture · SDP · Duality · Treewidth · Derivative-free optimization · Powell's method

1 Introduction

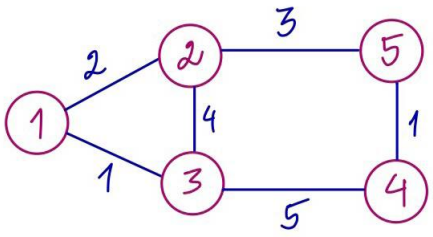
In this article we are working on the Max-Cut problem, where one is interested in finding the cut of the largest value according to a given graph i.e. partition of vertices in two parts with the maximal total weight of edges between these parts. The Max-Cut is an example of **NP**-complete problem, the study of which essentially amounts to the search of polynomial-time algorithm, providing a good approximation of true solution. For example, a simple probabilistic algorithm based on dividing the vertices of the graph into two parts by flipping a fair coin, provides the answer, expectation of which is a half of the total weight[4].
The approach, pioneered by Goemans and Williamson, has become a significant breakthrough. It has been shown that if the Unique Games Conjecture holds, then the Goemans-Williamson approximation algorithm is optimal [5].
Further we will be interested in the specific case where the graph's treewidth is bounded by a constant.

2 Problem statement

Given a weighted, undirected graph $G = (V, E)$ i.e. each edge (i, j) has a weight $w_{ij} = w_{ji}$. The set of vertices is partitioned into two parts S and $\bar{S} := V \setminus S$. Let us define the weight of this cut as the sum of the weights of edges which endpoints are contained in different parts

$$W(S) := \sum_{(i,j) \in S \times \bar{S}} w_{ij}$$

The aim is to find the cut with maximal possible weight.



S	$W(S)$
$\varnothing$	0
$\{5\}$	$1+3=4$
$\{1,2,3\}$	$3+5=8$
$\{1,2,4\}$	$1+4+3+5+1=14$
$\{1,5,4\}$	$1+2+3+5=11$
$\{1,3,5\}$	$2+4+3+1+5=15$

Example

The maximum cut is 15. Indeed, the graph is not bipartite, the total weight of all edges is 16, and there are no edges lighter than 1.
Unlike the well-studied Min-Cut problem, which is solvable in polynomial time using the Ford-Fulkerson algorithm[3], the Max-Cut is known to be **NP**-complete[9]. Rather than attempting to refute the hypothesis $\mathbf{P} \neq \mathbf{NP}$, we will focus on developing polynomial-time algorithm that provides a good approximation to the optimal solution.

3 Theory

Let us paraphrase the issue within the context of an optimization problem.

For each vertex i in the graph we define the indicator $x_i \in \{-1, +1\}$ characterizing the affiliation of $i \in S$ or $i \in \bar{S}$, respectively.
Similarly, for each edge (i, j) we define the indicator $y_{ij} = y_{ji} = x_i x_j \in \{-1, +1\}$ characterizing the belonging of the edge to the cut. Let $\mathbf{x}$ represent the vector $\mathbf{x} = (x_1, \dots, x_n)^\top$ where $n = |V|$.

Now the Max-Cut can be represented as

$$\textbf{MaxCut} := \max_{y_{ij} \in \{-1,+1\}} \frac{1}{4} \sum_{i \in V} \sum_{j \in V} w_{ij}(1 - y_{ij}) = \max_{x_i^2 = 1} \frac{1}{4} \sum_{i \in V} \sum_{j \in V} w_{ij}(1 - x_i x_j) = \max_{\substack{X = \mathbf{x}\mathbf{x}^\top \\ X_{ii} = 1}} \frac{1}{4} \sum_{i \in V} \sum_{j \in V} w_{ij}(1 - X_{ij})$$

In terms of Frobenius inner product[20],

$$\textbf{MaxCut} = \frac{1}{4} \max_{\substack{X = \mathbf{x}\mathbf{x}^\top \\ X_{ii} = 1}} \langle W, I_n - X \rangle = \frac{1}{4} \max_{\substack{X = \mathbf{x}\mathbf{x}^\top \\ Diag(X) = \mathbf{1}}} \left(\sum_{i=1}^{n} \sum_{j=1}^{n} W_{ij} - \langle W, X \rangle \right)$$

where W is the adjacency matrix of the graph

Clearly, $X_{ij} = x_i x_j$ hence $x_i^2 = X_{ii} = 1 \implies x_i \in \{-1, +1\}$

In particular, the matrix X is

1. Symmetric with units on the diagonal
2. Positive semi-definite: $\forall \mathbf{v} \in \mathbb{R}^n : \quad \mathbf{v}^\top X \mathbf{v} = \mathbf{v}^\top \mathbf{x}\mathbf{x}^\top \mathbf{v} = (\mathbf{x}^\top \mathbf{v})^\top (\mathbf{x}^\top \mathbf{v}) = (\mathbf{x}^\top \mathbf{v})^2$

3.1 Semidefinite Programming

Finally, let us consider the following problem

$$\textbf{SDP} := \max_{\substack{X = X^\top \succcurlyeq 0 \\ Diag(X) = \mathbf{1}}} \frac{1}{4} \sum_{i \in V} \sum_{j \in V} w_{ij}(1 - X_{ij}) = \max_{\substack{X = X^\top \succcurlyeq 0 \\ Diag(X) = \mathbf{1}}} \langle W, I_n - X \rangle$$

Lemma 3.1 (PSD - decomposition[15])**.**
Let $X \in \mathbb{R}^{n \times n}$ be a symmetric, positive semidefinite matrix. Then $X = VV^\top$ for some $V \in \mathbb{R}^{n \times n}$.

Proof:
Any symmetric real matrix is diagonalizable: its eigenvalues are real and its eigenvectors are orthogonal. Hence X

has an eigendecomposition $X = Q\Lambda Q^\top$, where the columns of Q are eigenvectors of X and the diagonal entries of diagonal matrix Λ are the eigenvalues of X. Also X is positive semidefinite, then all its eigenvalues are nonnegative, which means their square roots are real. Hence,

$$X = Q\Lambda Q^\top = Q\Lambda^{\frac{1}{2}}\Lambda^{\frac{1}{2}}Q^\top = \left(Q\Lambda^{\frac{1}{2}}\right)\left(Q\Lambda^{\frac{1}{2}}\right)^\top = VV^\top, \quad V \in \mathbb{R}^{n\times n}$$

Note that the rows of V are the eigenvectors of X multiplied by the square roots of the (nonnegative) eigenvalues of X. □

The difference between the **MaxCut** and **SDP** is as follows

	MaxCut	**SDP**
X	$X = \mathbf{x}\mathbf{x}^\top$, $\mathbf{x} \in \mathbb{R}^n$	$X = VV^\top$, $\mathbf{x} \in \mathbb{R}^{n\times m}$
X_{ij}	$X_{ij} = x_i x_j \in \{-1, +1\}$	X_{ij} is an arbitrary element

Hence **MaxCut** $\leqslant$ **SDP** (requirement $X = \mathbf{x}\mathbf{x}^\top$ is a particular case of $X = VV^\top$ therefore maximization in **MaxCut** is being performed on the subspace of constraint space of **SDP**)

Hereby we provide a straightforward cvxpy code solving **SDP** (W is the adjacency matrix of the graph shown in the example above):

```
import numpy as np
import cvxpy as cp

n = 5
W = np.array([[0, 2, 1, 0, 0],
              [2, 0, 4, 0, 3],
              [1, 4, 0, 5, 0],
              [0, 0, 5, 0, 1],
              [0, 3, 0, 1, 0]])
X = cp.Variable((n, n), symmetric=True)
constraints = [X >> 0]
constraints += [X[i][i] == 1 for i in range(n)]
objective = cp.Maximize(0.25 * cp.sum(cp.multiply(W, (1 - X))))
prob = cp.Problem(objective, constraints)
prob.solve()
print("The optimal value is ", round(prob.value))
print("The solution is", X.value)
```

The optimal value is 15

$$\text{The solution is } X = \begin{pmatrix} 1 & -1.00000047 & 1.0000005 & -1.00000052 & 1.00000057 \\ -1.00000047 & 1 & -1.00000027 & 1.00000038 & -1.00000037 \\ 1.0000005 & -1.00000027 & 1 & -1.00000026 & 1.0000004 \\ -1.00000052 & 1.00000038 & -1.00000026 & 1 & -1.00000042 \\ 1.00000057 & -1.00000037 & 1.0000004 & -1.00000042 & 1 \end{pmatrix}$$

Indeed, rounded matrix $\begin{pmatrix} 1 & -1 & 1 & -1 & 1 \\ -1 & 1 & -1 & 1 & -1 \\ 1 & -1 & 1 & -1 & 1 \\ -1 & 1 & -1 & 1 & -1 \\ 1 & -1 & 1 & -1 & 1 \end{pmatrix}$ corresponds to the maximum cut $S = \{1, 3, 5\}$.

In the **SDP** approach the optimal matrix X is decomposed by Cholesky algorithm in $X = LL^\top$, where the columns of the matrix L are considered as vectors in $\mathbb{R}^n$. These vectors are then divided by a random unbiased hyperplane, and x_i is assigned to 1 if the i-th vector is contained in the positive half-plane (and -1 otherwise).

Lemma 3.2.

$$\textbf{MaxCut} = \frac{1}{4}\max_{x_i^2=1} \mathbf{x}^\top L\mathbf{x}, \text{ where } L \text{ is the Laplacian[13] of the graph.}$$

Proof:

$$\max_{x_i^2=1} \mathbf{x}^\top L\mathbf{x} = \max_{\substack{X=\mathbf{x}\mathbf{x}^\top \\ Diag(X)=\mathbf{1}}} \langle L, X\rangle = \max_{\substack{X=\mathbf{x}\mathbf{x}^\top \\ Diag(X)=\mathbf{1}}} \langle D-W, X\rangle = \max_{\substack{X=\mathbf{x}\mathbf{x}^\top \\ Diag(X)=\mathbf{1}}} \left(\langle D, X\rangle - \langle W, X\rangle\right) =$$

$$= \max_{\substack{X=\mathbf{x}\mathbf{x}^\top \\ Diag(X)=\mathbf{1}}} \left(\sum_{i=1}^n D_{ii}X_{ii} - \langle W, X\rangle\right) = \max_{\substack{X=\mathbf{x}\mathbf{x}^\top \\ Diag(X)=\mathbf{1}}} \left(\sum_{i=1}^n D_{ii} - \langle W, X\rangle\right) = \max_{\substack{X=\mathbf{x}\mathbf{x}^\top \\ Diag(X)=\mathbf{1}}} \left(\sum_{i=1}^n\sum_{j=1}^n W_{ij} - \langle W, X\rangle\right) = \mathbf{MaxCut}$$

For $D = Diag(L)$, so $D_{ii} = \sum_{j=1}^n W_{ij} \implies \sum_{i=1}^n D_{ii} = \sum_{i=1}^n \left(\sum_{j=1}^n W_{ij}\right)$

□

3.2 Dual problem

$$\mathbf{MaxCut} = \frac{1}{4}\max_{x_i^2=1} \mathbf{x}^\top L\mathbf{x} = \frac{1}{4}\max_{\boldsymbol{\lambda}\geqslant 0}\min_{\mathbf{x}}\left(\mathbf{x}^\top L\mathbf{x} + \sum_{i=1}^n \lambda_i(1-x_i^2)\right) = \frac{1}{4}\max_{\boldsymbol{\lambda}\geqslant 0}\min_{\mathbf{x}}\left(\mathbf{x}^\top\left(L - Diag(\boldsymbol{\lambda})\right)\mathbf{x} + \mathbf{1}^\top\boldsymbol{\lambda}\right) \leqslant \text{[16]}$$

$$\leqslant \frac{1}{4}\min_{\boldsymbol{\lambda}\geqslant 0}\max_{\mathbf{x}}\left(\mathbf{x}^\top\left(L - Diag(\lambda)\right)\mathbf{x} + \mathbf{1}^\top\boldsymbol{\lambda}\right) = \frac{1}{4}\min_{\substack{Diag(\boldsymbol{\lambda})\geqslant L \\ \boldsymbol{\lambda}\geqslant 0}} \mathbf{1}^\top\boldsymbol{\lambda} =: \mathbf{Dual}$$

Last transition is valid since if $Diag(\boldsymbol{\lambda}) - L \not\succcurlyeq 0$, then $\mathbf{x}^\top\left(L - Diag(\boldsymbol{\lambda})\right)\mathbf{x} \underset{\substack{\mathbf{x}=\alpha\mathbf{v} \\ \alpha\longrightarrow+\infty}}{\longrightarrow} +\infty$ when $\mathbf{v}$ is a negative eigenvector of $Diag(\boldsymbol{\lambda}) - L$.

Lemma 3.3.

$$\mathbf{SDP} = \mathbf{Dual}$$

Proof:

$$4\cdot\mathbf{SDP} = \max_{\substack{X=X^\top\succcurlyeq 0 \\ Diag(X)\equiv\mathbf{1}}} \langle L, X\rangle = \min_{\boldsymbol{\lambda}\geqslant 0}\max_{X=X^\top\succcurlyeq 0}\left(\langle L, X\rangle + \sum_{i=1}^n \lambda_i(1-X_{ii})\right) = \min_{\boldsymbol{\lambda}\geqslant 0}\max_{X=X^\top\succcurlyeq 0}\left(\langle L - Diag(\boldsymbol{\lambda}), X\rangle\right) + \mathbf{1}^\top\boldsymbol{\lambda}$$

$$= \max_{X=X^\top\succcurlyeq 0}\min_{\boldsymbol{\lambda}\geqslant 0}\left(\langle L - Diag(\boldsymbol{\lambda}), X\rangle\right) + \mathbf{1}^\top\boldsymbol{\lambda} \overset{3.2}{=} \min_{\substack{Diag(\boldsymbol{\lambda})\geqslant L \\ \boldsymbol{\lambda}\geqslant 0}} \mathbf{1}^\top\boldsymbol{\lambda} = 4\cdot\mathbf{Dual}$$

□

Here is a cvxpy solver for **Dual** problem(L is the Laplacian matrix of the graph shown in the example above):

```
import numpy as np
import cvxpy as cp

n = 5
L = np.array([[3, -2, -1, 0, 0],
              [-2, 9, -4, 0, -3],
              [-1, -4, 10, -5, 0],
              [0, 0, -5, 6, -1],
              [0, -3, 0, -1, 4]])
X = cp.Variable((n, n), diag=True)
constraints = [X >> L]
objective = cp.Minimize(0.25 * cp.trace(X))
prob = cp.Problem(objective, constraints)
prob.solve()
print("The optimal value is", round(prob.value))
print(" The solution is\n", X.value.toarray())
```

The optimal value is 15

$$\text{The solution is } X = \begin{pmatrix} 3.99998606 & 0 & 0 & 0 & 0 \\ 0 & 17.99998614 & 0 & 0 & 0 \\ 0 & 0 & 17.99998614 & 0 & 0 \\ 0 & 0 & 0 & 11.9999861 & 0 \\ 0 & 0 & 0 & 0 & 7.99998608 \end{pmatrix}$$

The question arises as to how to retrieve real cut from solution matrix X.

3.3 k-diagonal extension

Instead of solving

$$\mathbf{Dual} = \min_{\substack{Diag(\boldsymbol{\lambda}) \succcurlyeq L \\ \boldsymbol{\lambda} \geqslant 0}} \frac{\mathbf{1}^\top \boldsymbol{\lambda}}{4} = \min_{\substack{Diag(\boldsymbol{\lambda}) \succcurlyeq L \\ \boldsymbol{\lambda} \geqslant 0}} \max_{x_i^2=1} \frac{\mathbf{x}^\top Diag(\boldsymbol{\lambda})\mathbf{x}}{4}$$

Let us expand the space of minimization to the k-diagonal matrices (symmetric matrices having at most $\frac{k-1}{2}$ leading diagonals on both sides of main diagonal)

$$\mathbf{D_k} := \min_{\substack{T:\ T=T^\top \succcurlyeq L \\ T \text{ is } k\text{-diagonal}}} \frac{1}{4} \max_{x_i^2=1} x^\top T x, \qquad k \underset{2}{\equiv} 1$$

Therefore,

$$\mathbf{SDP} = \mathbf{Dual} = \mathbf{D_1} \ \geqslant\ \mathbf{D_3} \ \geqslant\ \mathbf{D_5} \ \geqslant\ \ldots \ \geqslant\ \mathbf{D_{2n-1}} = \mathbf{MaxCut}$$

Let us define an inner function as

$$f(T) := \frac{1}{4} \max_{x_i^2=1} x^\top T x$$

Hence, the aim is to solve

$$\min_{\substack{T:\ T=T^\top \succcurlyeq L \\ T \text{ is } k\text{-diagonal}}} f(T)$$

The obstacle here is that $f(T)$ is not easily differentiable. Let us now consider derivative-free methods.

Note that $f(T)$ is computable in $O(2^k \cdot n)$ time using dynamic programming, where T is a k-diagonal matrix. Therefore, optimization for this problem could be performed by using a polynomial-time oracle to compute $f(T)$ on each iteration.

3.4 Optimizers comparison

Let us now turn our attention to the nevergrad package in Python with ready-made implementations of a wide range of gradient-free optimizers. Visualization of some optimizers' work here[18].

1. CMA[14]
2. Powell's method[1]
3. TBPSA[11]
4. TwoPointsDE[19]

We have selected the Powell's method due to its numerous benefits.

The pertinent issue is how to derive a real cut from the optimal matrix T^* for $\mathbf{D_k}$. The following approach is suggested: we will run our dynamic oracle on T^* remembering all transitions that led to the optimal value.

3.5 k-treewidth extension

Another approach is to perform minimization among k-treewidth graphs (having a treewidth of at most k)[10].

$$\mathbf{H_k} := \min_{\substack{T:\ T=T^\top \succcurlyeq L, \\ T \text{ is } k\text{-treewidth}}} \frac{1}{4} \max_{x_i^2=1} x^\top T x$$

Here again, inner function can be solved by dynamic programming.
Therefore,

$$\mathbf{SDP} = \mathbf{Dual} = \mathbf{D_1} \geqslant \mathbf{H_1} \geqslant \mathbf{H_2} \geqslant \mathbf{H_3} \geqslant \ldots \geqslant \mathbf{H_n} = \mathbf{D_{2n-1}} = \mathbf{MaxCut}$$

Moreover, $\mathbf{D_k} \geqslant \mathbf{H_k}, \quad k \underset{2}{\equiv} 1$

And again, the question arises as to how to derive a real cut from the optimal matrix T^* for $\mathbf{H_k}$.
This time we proceed with a similar to the k-diagonal case approach, with the exception of the new dynamic oracle.

4 Computational experiment

4.1 Datasets

In our research we are using the Biq Mac Library[6]

1. g05_60.i: For each dimension ten unweighted graphs with edge probability 0.5. n=60
2. g05_80.i: For each dimension ten unweighted graphs with edge probability 0.5. n=80
3. g05_100.i: For each dimension ten unweighted graphs with edge probability 0.5. n=100
4. pwd_100.i: For each density ten graphs with integer edge weights chosen from [0,10] and density d=0.1, 0.5, 0.9. n=100

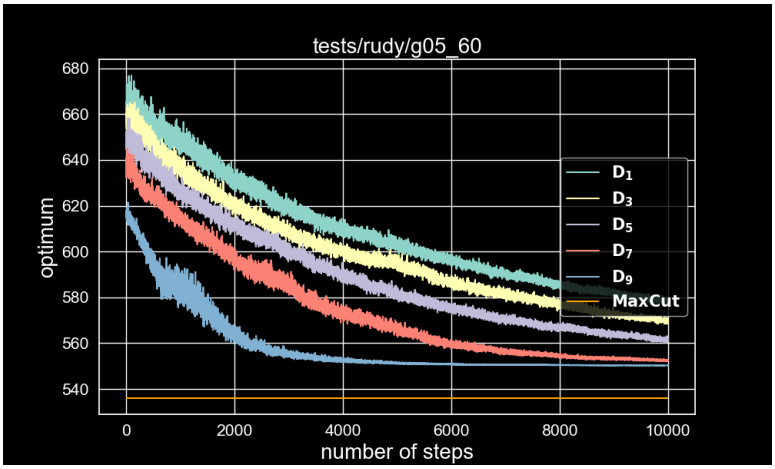
4.2 Roadmap

1. For each graph from dataset
   - 1.1 Use Goemans and Williamson approach to solve $\mathbf{SDP}$ and retrieve cut, then calculate ratio $\frac{\mathbf{SDP}^{cut}}{\mathbf{MaxCut}}$.
   - 1.2 For $k \in \{1, 3, 5, 7, 19\}$ run k-diagonal solver and retrieve cut from optimum for $\mathbf{D_k}$, then calculate ratio $\frac{\mathbf{D_k}^{cut}}{\mathbf{MaxCut}}$.
   - 1.3 Solve $\mathbf{Dual}$ and retrieve cut using 1-diagonal solver, then calculate ratio $\frac{\mathbf{Dual}^{cut}}{\mathbf{MaxCut}}$.
2. Examine error analysis on a graph of each type.
3. Compare the obtained results on the plots and in the table.

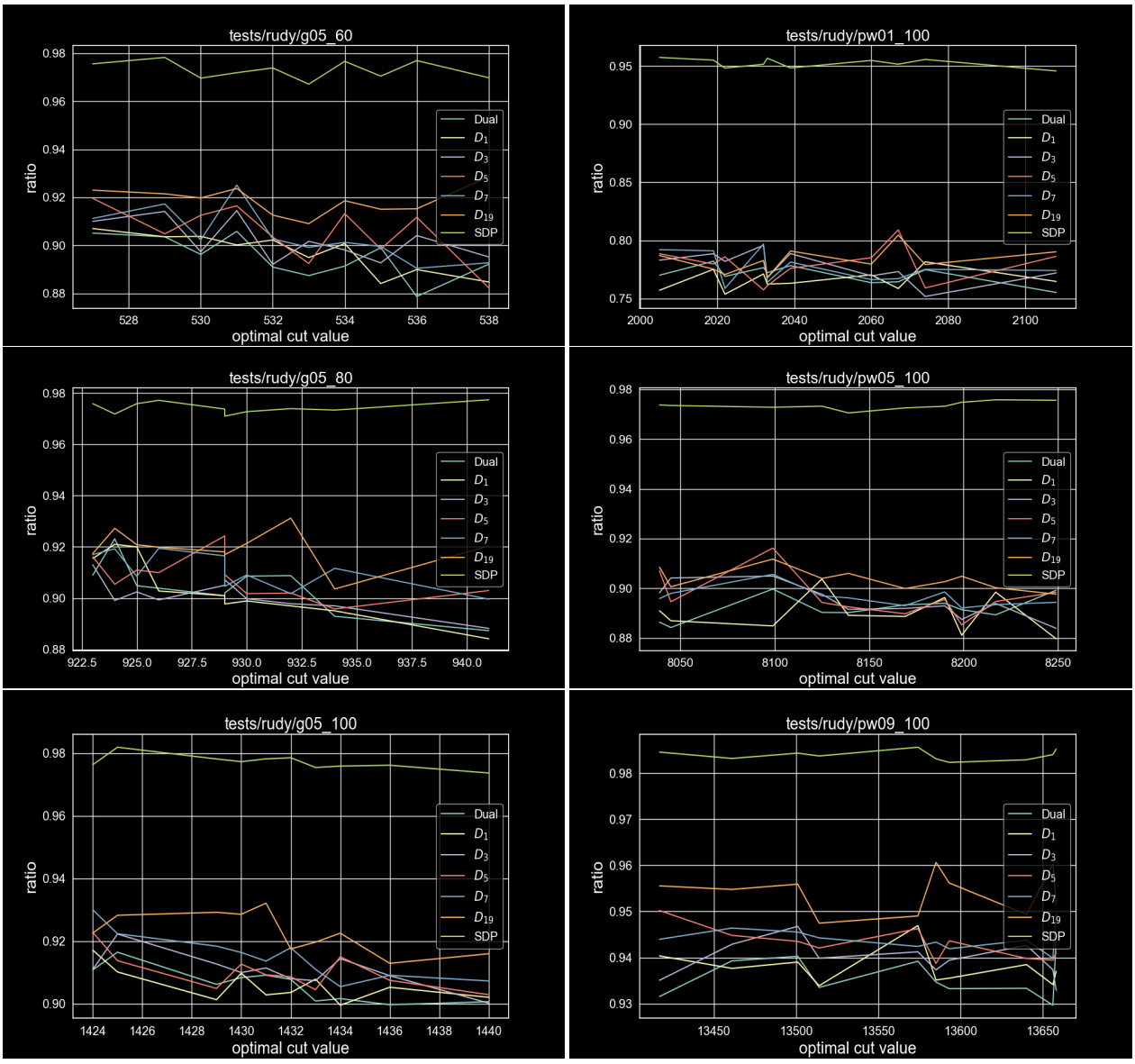
5 Results

5.1 Error analysis

The optimizer demonstrates good convergence as the number of iterations increases.



5.2 Results comparison



Average ratios

Test	Dual	D_1	D_3	D_5	D_7	D_{19}	SDP
g05-60	0.8951	0.8971	0.902	0.9055	0.9042	0.9187	0.9731
g05-80	0.9042	0.9033	0.9009	0.9079	0.9107	0.9197	0.9743
g05-100	0.9062	0.906	0.9107	0.9102	0.9152	0.923	0.9772
pw01-100	0.7706	0.7658	0.7772	0.7788	0.7767	0.7828	0.9525
pw05-100	0.8919	0.89	0.8947	0.8969	0.8964	0.9036	0.9736
pw09-100	0.9359	0.9378	0.9396	0.943	0.9441	0.9541	0.9839

6 Conclusion

In this paper, we have discussed a novel approach to solving the Max-Cut problem under specific constraints on the graph's Laplacian. Unfortunately, the implemented k-diagonal solver has not outperformed the $\mathbf{SDP}$. In our opinion, the issue lies in the use of gradient-free optimizers which do not operate properly under the specified constraints.
Further research into this topic could be pursued in two directions. First, one could attempt to implement specific gradient-free technique tailored to the task. Apart from that, the solution of the k-treewidth case requires the implementation of a new oracle that solves $\mathbf{H_k}$, based on a polynomial tree decomposition[8].

References

[1] Willard I. Zangwill. "Minimizing a function without calculating derivatives". In: *Universitetet i Bergen* (1967). URL: https://www.ii.uib.no/~lennart/drgrad/Zangwill1967.pdf.

[2] David P. Williamson Michel X. Goemans. "Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems Using Semidefinite Programming". In: *MIT* (1995). URL: https://math.mit.edu/~goemans/PAPERS/maxcut-jacm.pdf.

[3] Wikipedia. "Minimum cut". In: *cs.princeton.edu* (2004). URL: https://www.cs.princeton.edu/courses/archive/spr04/cos226/lectures/maxflow.4up.pdf.

[4] Eli Upfal Michael Mitzenmacher. *Probability and Computing*. Cambridge University Press, 2005. URL: https://www.cs.purdue.edu/homes/spa/courses/pg17/mu-book.pdf.

[5] Elchanan Mossel Subhash Khot Guy Kindler and Ryan O'Donnell. *Probability and Computing*. Cambridge University Press, 2005. URL: https://www.cs.purdue.edu/homes/spa/courses/pg17/mu-book.pdf.

[6] Angelika Wiegele. *Biq Mac Library*. 2007. URL: http://bqp.cs.uni-bonn.de/library/html/instances.html.

[7] Instructor: Irit Dinur and Scribe: Tom Ferster Amey Bhangale. "MAXCUT approximation algorithm and UGC-hardness". In: *Weizmann Institute of Science, Faculty of Mathematics and Computer Science* (2010). URL: https://www.wisdom.weizmann.ac.il/~dinuri/courses/19-inapprox/lec6.pdf.

[8] Hauke Brinkop and Klaus Jansen. "Solving Cut-Problems in Quadratic Time for Graphs With Bounded Treewidth". In: *arxiv.org* (2012).

[9] David Steurer. "Reduction from 3 SAT to MAX CUT". In: *cs.cornell.edu* (2014). URL: https://www.cs.cornell.edu/courses/cs4820/2014sp/notes/reduction-maxcut.pdf.

[10] Gerrod Voigt. "Tree Decompositions, Treewidth, and NP-Hard Problems". In: *math.mit.edu* (2016). URL: https://math.mit.edu/~apost/courses/18.204-2016/18.204_Gerrod_Voigt_final_paper.pdf.

[11] Olivier Teytaud Marie-Liesse Cauwet. "Population Control meets Doob's Martingale Theorems: the Noise-free Multimodal Case". In: *arxiv.org* (2017). URL: https://arxiv.org/pdf/2005.13970.

[12] Scribe: Wenzheng Li Lecturer: Aviad Rubinstein. "Introduction to Unique Games Conjecture". In: *Stanford University* (2019). URL: https://web.stanford.edu/class/cs354/scribe/lecture07.pdf.

[13] Hanchen Li. "Properties and Applications of Graph Laplacian". In: *mathematics.uchicago.edu* (2022). URL: https://math.uchicago.edu/~may/REU2022/REUPapers/Li,Hanchen.pdf.

[14] Nikolaus Hansen Inria. "The CMA Evolution Strategy: A Tutorial". In: *arxiv.org* (2023). URL: https://arxiv.org/pdf/1604.00772.

[15] math.stackexchange.com. *Decomposition of a positive semidefinite matrix*. URL: https://math.stackexchange.com/questions/1801403/decomposition-of-a-positive-semidefinite-matrix.

[16] *Minimax Inequality*. URL: https://inst.eecs.berkeley.edu/~ee127/sp21/livebook/l_dual_weak_minimax.html.

[17] Qiskit. *Max-Cut and Traveling Salesman Problem*. URL: https://qiskit-community.github.io/qiskit-optimization/tutorials/06_examples_max_cut_and_tsp.html.

[18] SimonBlanke. *Gradient-Free-Optimizers*. URL: https://github.com/SimonBlanke/Gradient-Free-Optimizers.

[19] Wikipedia. *Differential evolution*. URL: https://en.wikipedia.org/wiki/Differential_evolution.

[20] Wikipedia. *Frobenius inner product*. URL: https://en.wikipedia.org/wiki/Frobenius_inner_product.