Spectral Graph Theory and the Max-Cut Problem: Approximation Algorithms and Applications in Machine Learning

Prathamesh Chatorikar
University of California, Santa Cruz

Abstract

The Max-Cut problem asks how to split a graph into two parts so that the number or total weight of edges between the parts is as large as possible. This problem is hard to solve exactly, but there are good ways to approximate it. In this paper, I explain how methods based on semidefinite programming and spectral graph theory help solve Max-Cut. I focus on one recent method that uses much less memory while still giving strong results. I also look at how Max-Cut ideas are used in machine learning, especially for clustering and graph neural networks.

1 Introduction

The Max-Cut problem is a common problem in graph theory. It asks how to split the nodes of a graph into two groups so that the total weight of the edges between the groups is as large as possible. This problem is known to be hard, meaning we can't solve it quickly for large graphs.

Even though it's hard to solve exactly, we can still find good enough solutions using special methods. One famous method is from Goemans and Williamson, who used a tool called semidefinite programming (SDP) to get results that are close to the best possible [1]. Another way is to use spectral methods, which involve looking at the eigenvalues and eigenvectors of the graph's matrices [3, 4].

Recently, Shinde et al. [7] showed a new way to solve Max-Cut and related problems using much less memory. Their method keeps the results close to the older, more expensive ones while making it easier to run on big graphs.

In this paper, I explain what the Max-Cut problem is, how these methods work, and where they are used in machine learning, especially in tasks like clustering and working with graph data.

2 Problem Definition: Max-Cut and Spectral Relaxations

The Max-Cut problem is a classic question in graph theory. The goal is to divide the nodes of a graph into two groups so that the number (or total weight) of edges between the two groups is as large as possible. This is known as the "cut" of the graph.

Formally, let $G = (V, E)$ be an undirected graph with a set of nodes V and edges E. Each edge $(i, j) \in E$ has a non-negative weight w_{ij}. A *cut* is a partition of the vertex set V into two disjoint subsets S and $V \setminus S$. The value of the cut is the total weight of edges that have one endpoint in each part:

$$\mathrm{cut}(S) = \sum_{\substack{(i,j)\in E \\ i\in S, j\in V\setminus S}} w_{ij}$$

The Max-Cut problem asks for the subset S that maximizes this sum.

Binary Formulation

We can also represent this problem using binary variables. For each node i, let $x_i \in \{-1, +1\}$ indicate which side of the cut it belongs to. An edge is cut if $x_i \neq x_j$, and we can express the cut value as:

$$\max_{\mathbf{x}\in\{-1,+1\}^n} \frac{1}{2} \sum_{(i,j)\in E} w_{ij}(1 - x_i x_j)$$

This form is useful for developing algorithms, but it is still NP-hard to solve exactly.

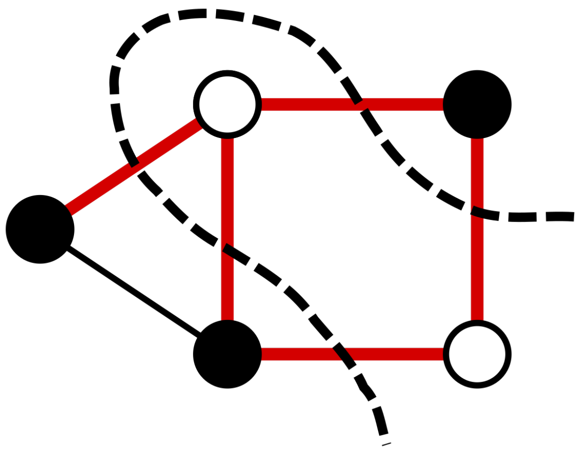

Figure 1: Example of a Max-Cut on a graph. The nodes are split into two sets (black and white), and the red edges are the ones that cross between the sets. These are the "cut" edges. The goal is to maximize the total weight (or count) of these red edges.

Semidefinite Programming (SDP) Relaxation

Goemans and Williamson [1] proposed a way to relax the problem so it can be solved using semidefinite programming. Instead of assigning ± 1 values, each node is assigned a unit vector $v_i \in \mathbb{R}^n$. The relaxed problem becomes:

$$\begin{aligned} \max_{X\succeq 0} \quad & \frac{1}{2} \sum_{(i,j)\in E} w_{ij}(1 - X_{ij}) \\ \text{s.t.} \quad & X_{ii} = 1 \quad \text{for all } i \end{aligned}$$

Here, X is a matrix where $X_{ij} = v_i \cdot v_j$, and $X \succeq 0$ means it must be positive semidefinite. After solving this relaxed problem, a rounding step assigns each node to one side of the cut based on the direction of a randomly chosen hyperplane.

Spectral Relaxation

Another way to approximate Max-Cut is using spectral methods. These methods look at the eigenvalues and eigenvectors of matrices like the adjacency matrix or the graph Laplacian. A simple method is to use the sign of the entries in the eigenvector corresponding to the smallest eigenvalue of the adjacency matrix to assign nodes to the two groups.

Trevisan [3] proposed a spectral algorithm that breaks the 50% approximation barrier. Soto [4] later improved the analysis and showed this method gives at least about 61.4% of the optimal cut.

Why Relaxations Are Needed

The original Max-Cut problem is NP-hard, meaning it can't be solved efficiently for large graphs. Relaxations like SDP and spectral methods make it possible to get good approximate solutions in a reasonable amount of time. They also give useful insights into the structure of the graph and help in machine learning tasks like clustering, as I explore in later sections.

3 Spectral and SDP Techniques for Max-Cut

There are two main ways I found for solving Max-Cut approximately: using semidefinite programming (SDP) and using spectral methods.

SDP Method

The Goemans-Williamson algorithm [1] gives the best known approximation ratio of about 0.878. It uses semidefinite programming to relax the problem, followed by a rounding step using a random hyperplane. This method is slow and memory-intensive but gives strong results. It was also extended to other problems like Max-k-Cut [2].

Spectral Methods

Spectral methods work by analyzing the eigenvalues and eigenvectors of a matrix that represents the graph. Trevisan [3] introduced a simple spectral algorithm that achieves an approximation ratio of at least 0.531. Soto [4] improved it to about 0.614. These methods run much faster but give weaker guarantees compared to SDP.

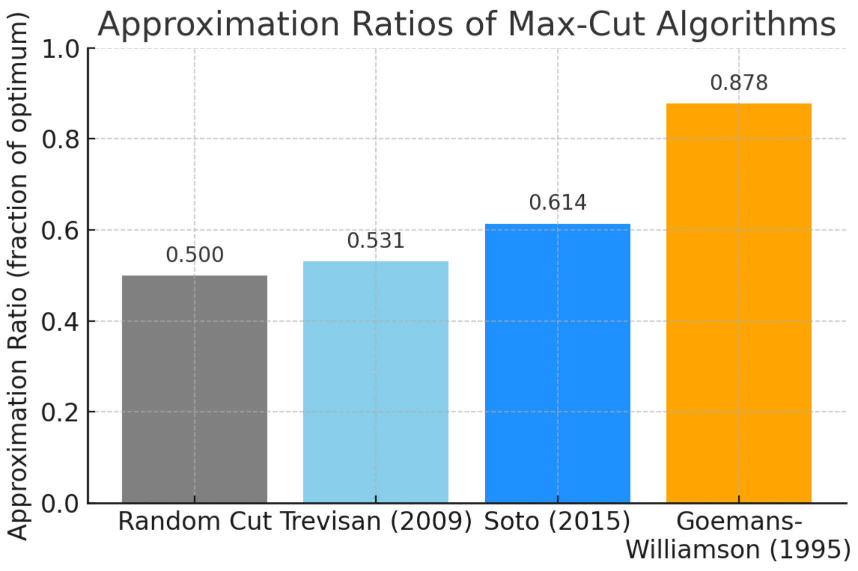

Figure 2: Approximation ratios of different Max-Cut algorithms. Goemans-Williamson's SDP method achieves the best known ratio of 0.878, while spectral methods like Trevisan and Soto offer faster but slightly weaker performance.

Low-Rank and Memory-Efficient SDP

Solving full SDP can be hard for large graphs. Burer and Monteiro [5] suggested solving it using low-rank matrix factorization, and Boumal et al. [6] showed this often works well. More recently, Shinde et al. [7] developed a memory-efficient way to approximate SDP solutions using Gaussian sampling. Their method achieves nearly the same performance as Goemans-Williamson but uses much less memory.

When to Use What

If accuracy is the top priority and the graph is not too big, I recommend using SDP-based methods. If the graph is large or time is limited, spectral methods are a better choice. Newer low-memory methods try to balance both.

4 Applications in Machine Learning and Data Clustering

The Max-Cut problem and related ideas are useful in several machine learning tasks, especially when working with graph data. Here are a few areas where they are applied.

4.1 Correlation Clustering

In correlation clustering, we are given a graph where edges can be marked as "similar" (positive) or "different" (negative), and the goal is to group the nodes so that similar nodes are in the same group and different ones are in different groups.

This problem is closely related to Max-Cut. If we only allow two groups and treat negative edges as ones we want to cut, the problem becomes a version of Max-Cut.

The original correlation clustering problem was introduced by Bansal et al. [8]. Later, Charikar et al. [9] used semidefinite programming to give better approximation algorithms. These methods are useful when we don't know the number of clusters in advance.

Shinde et al. [7] applied their memory-efficient SDP method to correlation clustering. It worked well for large graphs and gave results close to the best known algorithms.

4.2 Spectral Clustering

Spectral clustering is a method where we group nodes in a graph using information from the graph's eigenvalues and eigenvectors. This is often used when we want to find tightly connected groups in data.

While spectral clustering usually tries to minimize cuts (to keep similar nodes together), Max-Cut does the opposite—it tries to cut as many edges as possible. Still, both use similar tools like graph Laplacians and eigenvectors.

Because spectral methods are fast and easy to use, they are common in unsupervised learning tasks like image segmentation, text clustering, and document grouping.

4.3 Graph Neural Networks (GNNs)

In deep learning on graphs, one challenge is how to "pool" or reduce the graph to fewer nodes while keeping important structure. One solution is to use a cut-based idea to decide how to group nodes.

Bianchi et al. [10] proposed MinCutPool, a method that learns how to group nodes in a graph by minimizing the number of edges within each group. This is related to the Max-Cut idea, just reversed.

The method adds a cut-based loss to the model, helping the network learn better structure for graph classification tasks. This shows how classic graph problems like Max-Cut can be used in modern machine learning.

4.4 Insights

Max-Cut and its relaxations are useful in ML for:

- Grouping nodes based on similarity or dissimilarity (correlation clustering).
- Finding good clusters quickly using spectral methods.
- Helping deep learning models process graph data more effectively.

These methods work well in practice, especially when graphs are large or when we don't know the number of clusters in advance.

5 Discussion: Insights, Strengths, and Limitations

In this section, I share what I learned from studying different methods for solving the Max-Cut problem. I explain what these methods do well, where they struggle, and what open questions remain.

5.1 What I Learned

Max-Cut is hard to solve exactly, but there are good ways to find close answers. The two main types of methods I focused on are:

- **Semidefinite programming (SDP)** — gives strong results but uses more memory and time.
- **Spectral methods** — faster and easier to run, but not as accurate in the worst case.

Each method works better depending on the size of the graph and how good a solution is needed.

5.2 Strengths of SDP Methods

The Goemans-Williamson algorithm [1] gives a cut that is at least 87.8% as good as the best possible one. This is still the best-known approximation guarantee for Max-Cut.

Other researchers have shown how SDP can be extended to other problems like Max-k-Cut [2]. To make SDP faster, Burer and Monteiro [5] suggested a low-rank version that needs fewer variables. Boumal et al. [6] showed this version works well in many cases.

More recently, Shinde et al. [7] created a method that avoids storing large matrices by using random vectors. I found this especially useful for large graphs where normal SDP would not be practical.

5.3 Strengths of Spectral Methods

Spectral methods, such as those from Trevisan [3] and Soto [4], work by looking at the graph's eigenvalues. These methods are fast and easy to run on large graphs. Although they don't reach the 87.8% performance of SDP, they are still better than random guessing. Soto's improved version guarantees about 61.4% of the optimal cut.

I found spectral methods especially helpful in machine learning, where fast clustering of large graphs is more important than perfect accuracy.

5.4 Use in Machine Learning

Max-Cut ideas are used in machine learning tasks like correlation clustering [8,9], where the goal is to group similar nodes together. I also studied how Max-Cut shows up in graph neural networks. One example is MinCutPool [10], where the network learns how to group nodes while keeping important structure.

In these settings, exact solutions are less important. What matters is speed, scalability, and good enough performance on real-world data.

5.5 Limitations

Even with all these improvements, there are still some clear limitations:

- **SDP is slow and uses a lot of memory.** This makes it hard to use on very large graphs without simplifications.
- **Spectral methods are fast but not always accurate.** They can miss important structure if the graph is noisy or unbalanced.

- **No method works best in all cases.** Sometimes cuts are too uneven or don't reflect the real structure of the data.
- **Learning-based methods depend on training data.** In MinCutPool [10], if the model isn't trained well, the learned cuts may not be meaningful.

5.6 Key Takeaways

SDP-based methods offer the best theoretical performance, but they are slow and memory-heavy. Spectral methods are faster and scale better, but don't always produce the best cuts. New approaches like low-rank optimization and learning-based models help balance both sides. I found that Max-Cut is not just a hard graph problem—it's a useful tool in machine learning, and there is still a lot of room to improve how we solve and apply it.

6 Future Directions and Open Questions

While studying the Max-Cut problem and its applications, I found several areas where more research and progress are still possible. These are important both for improving algorithms and for using them better in real-world tasks.

6.1 Better Approximations Without SDP

The best known Max-Cut approximation is about 87.8%, achieved by the Goemans-Williamson SDP method [1]. One open question is whether this same guarantee can be reached using simpler or faster methods, like spectral algorithms. So far, no non-SDP method has matched this result.

6.2 Combining Spectral Methods with Learning

Spectral methods are fast, and learning-based models are flexible. I believe combining them could help improve both speed and accuracy. For example, a learning model could suggest which parts of the graph to cut, and spectral methods could refine those suggestions.

6.3 Solving Max-Cut on Changing Graphs

Many real-world graphs change over time, like social networks or communication systems. It's still hard to solve Max-Cut when edges or nodes are added or removed over time. More work is needed to design Max-Cut algorithms that can update quickly as the graph changes.

6.4 Using Max-Cut Ideas in Neural Networks

Methods like MinCutPool [10] show how Max-Cut ideas can help graph neural networks learn better node groupings. But there is room to improve. For example, can these networks learn closer to optimal cuts without needing labels? Or can we add cut-based objectives to other types of graph models?

6.5 Making Algorithms More Scalable

Even with improvements like low-rank optimization [5, 6] and memory-efficient sampling [7], it is still hard to run Max-Cut algorithms on massive graphs. Finding new ways to reduce time and memory usage while keeping high-quality results is an ongoing challenge.

6.6 Better Understanding of When Methods Work Well

Not all graphs behave the same. Some have clear community structures; others are more random. I think it's important to understand which types of graphs work best with which methods. This could help choose the right algorithm for a given task instead of always using the same one. Max-Cut is an old problem, but it's still full of opportunities for new research. By improving approximation methods, mixing learning and math, and making algorithms faster and easier to use, I believe we can make these tools more useful for real-world applications, especially in machine learning.

7 Conclusion: Putting It All Together

In this paper, I studied the Max-Cut problem and the different methods used to solve it approximately. Max-Cut is a well-known graph problem where the goal is to divide a graph into two parts in a way that maximizes the total weight of edges between the parts. Since it is NP-hard, finding the exact solution is not practical for large graphs. However, several smart approximation techniques make it possible to get close to the best solution.

I focused on two main approaches: semidefinite programming (SDP) and spectral methods. The Goemans-Williamson algorithm [1] is the most well-known SDP-based method and gives the best known approximation ratio of about 87.8%. It is widely used in theory, but it can be slow and memory-heavy. To deal with this, researchers have proposed ways to make SDP more efficient, such as using low-rank matrix factorizations [5, 6] or memory-efficient sampling techniques [7]. These new methods make it easier to apply SDP to large-scale graphs.

On the other hand, spectral methods use the eigenvalues and eigenvectors of graph matrices to find cuts quickly. Although they don't provide the same level of guarantees as SDP, they are much faster and scale well. Trevisan [3] and Soto [4] showed that these methods can give decent results, especially when graphs have structure.

I also looked at how Max-Cut is used in machine learning. In correlation clustering [8, 9], Max-Cut helps group similar or dissimilar items. In deep learning, Max-Cut ideas are used in methods like MinCutPool [10], which help graph neural networks group nodes in a useful way during training.

Through this survey, I learned that each method has its strengths. SDP methods offer strong guarantees, but they require more resources. Spectral methods are lightweight and practical, though sometimes less accurate. New research is trying to combine the best of both—finding solutions that are both fast and reliable.

References

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

[2] Frieze, A., & Jerrum, M. (1997). Improved approximation algorithms for MAX k-CUT and MAX BISECTION.
Algorithmica, 18(1), 67–81.

[3] Trevisan, L. (2012). Max Cut and the smallest eigenvalue.
SIAM Journal on Computing, 41(6), 1769–1786.

[4] Soto, J. A. (2014). Improved analysis of a Max-Cut algorithm based on spectral partitioning. arXiv preprint arXiv:0910.0504.

[5] Burer, S., & Monteiro, R. D. (2003). A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization.
Mathematical Programming, 95(2), 329–357.

[6] Boumal, N., Voroninski, V., & Bandeira, A. S. (2016). The non-convex Burer–Monteiro approach works on smooth semidefinite programs.
In *Advances in Neural Information Processing Systems*, 29, 2757–2765.

[7] Shinde, N., Narayanan, V., & Saunderson, J. (2021). Memory-Efficient Approximation Algorithms for Max-k-Cut and Correlation Clustering.
In *Advances in Neural Information Processing Systems*, 34, 20743–20755.

[8] Bansal, N., Blum, A., & Chawla, S. (2004). Correlation clustering.
Machine Learning, 56(1-3), 89–113.

[9] Charikar, M., Guruswami, V., & Wirth, A. (2005). Clustering with qualitative information.
Journal of Computer and System Sciences, 71(3), 360–383.

[10] Bianchi, F. M., Grattarola, D., & Alippi, C. (2020). Spectral clustering with graph neural networks for graph pooling.
In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, 874–884.