Goemans and Williamson Algorithm for MAX-CUT

Areen Mahich

April 21, 2025

Acknowledgments

I would like to thank Sutanu Gayen and Debjyoti Dey for their valuable contributions and support.

Outline

Problem Statement Formal Definition and Mathematical Formulation

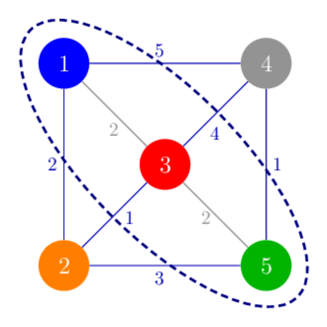
Abstract

- The Max-Cut problem is a classical combinatorial optimization problem.
- Given a weighted undirected graph, the objective is to partition the vertices into two disjoint sets such that the total weight of edges between the sets is maximized.
- The problem is NP-hard, but approximation algorithms can yield near-optimal results.
- We present the Goemans-Williamson algorithm achieving a 0.878-approximation.

Problem Statement

- **Input:** An undirected graph $G = (V, E)$ with edge weights $w_{ij} \geq 0$.
- **Objective:** Partition V into two sets S and $\bar{S}$ to maximize the total weight of edges between the sets:

$$\text{maximize} \quad \sum_{i \in S, j \in \bar{S}} w_{ij}$$



Motivation and Importance

- It is one of the classic graph problems and has real-world relevance.
- The Max-Cut problem appears in many fields, such as:
    - Network Analysis
    - Statistical physics
    - Image segmentation

Formal Definition of Max-Cut

- Let $G = (V, E)$ be an undirected graph with weight function $w : E \to \mathbb{R}_{\geq 0}$.
- A cut is defined by a partition of the vertex set V into two disjoint sets S and $\bar{S}$.
- The weight of the cut is the sum of weights of edges crossing the partition:

$$\text{Cut}(S) = \sum_{\substack{(i,j)\in E \\ i\in S,\, j\in \bar{S}}} w_{ij}$$

- The **Max-Cut problem** is to find a partition $(S, \bar{S})$ that maximizes $\text{Cut}(S)$.

Mathematical Formulation

- Define $x_i \in \{-1, +1\}$ to indicate which side of the cut vertex i belongs to.
- Then, the cut weight can be written as:

$$\frac{1}{2} \sum_{(i,j) \in E} w_{ij}(1 - x_i x_j)$$

Max-Cut as an Optimization Problem

- So, the Max-Cut problem becomes:

$$\max_{x \in \{-1,1\}^n} \frac{1}{4} \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij}(1 - x_i x_j)$$

- This is an **Integer programming problem** problem.

Computational Complexity

- The Max-Cut problem is one of Karp's 21 classic **NP-Complete problems**.
- Exact brut-force algorithms exist, but they are exponential in the worst case.

Approximation is the Key

- Goal: Find a solution that is close to optimal, with provable guarantees.
- Notably, the **Goemans-Williamson algorithm** provides a 0.878-approximation.

Step 1: Integer Programming Formulation

- Original Max-Cut program:

$$(\text{P1}) \qquad \max_{x_i \in \{-1,1\}} \frac{1}{4} \sum_{i,j} w_{ij}(1 - x_i x_j)$$

- Equivalent to finding a rank-1 matrix:

$$X = xx^\top \quad \text{where } X_{ij} = x_i x_j,\ X_{ii} = 1$$

- Objective becomes:

$$\frac{1}{4} \sum_{i,j} w_{ij}(1 - X_{ij})$$

Step 2: Vector Reformulation

- Key insight: Any rank-1 PSD matrix with $X_{ii} = 1$ can be written as:
$$X = xx^\top \quad \text{for } x \in \{-1, 1\}^n$$
- Thus, (P1) is equivalent to:

$$\text{(P2)} \quad \max_X \frac{1}{4} \sum_{i,j} w_{ij}(1 - X_{ij}) \quad \text{s.t. } X = xx^\top, X_{ii} = 1$$

Step 3: SDP Relaxation

- Relax the rank-1 constraint while keeping PSD:

$$(\text{P3}) \quad \max_X \frac{1}{4} \sum_{i,j} w_{ij}(1 - X_{ij}) \quad \text{s.t. } X \succeq 0, X_{ii} = 1$$

- Now an SDP
- Three key changes:
  1. Rank-1 $\rightarrow$ general PSD
  2. $X_{ij} \in \{-1, 1\} \rightarrow X_{ij} \in \mathbb{R}$
  3. Exact $\rightarrow$ approximate solution

Algorithm Implementation

Step 1: Original Formulation

We aim to solve the Max-Cut problem:

$$\max \sum_{i<j} w_{ij} \cdot \frac{1 - x_i x_j}{2} \quad \text{subject to } x_i \in \{-1, 1\}$$

Rewriting using matrix notation $X = xx^T$, where $X_{ij} = x_i x_j$:

$$\max \frac{1}{4} \sum_{i,j} w_{ij}(1 - X_{ij}) \quad \text{subject to } X = xx^T,\ x_i \in \{-1, 1\}$$

Step 2: SDP Relaxation

Relaxing the constraint $X = xx^T$ to general PSD matrices:

- $X \succeq 0$ (positive semidefinite)
- $X_{ii} = 1 \quad \forall i$

The relaxed SDP:

$$\max \frac{1}{4} \sum_{i,j} w_{ij}(1 - X_{ij}) \quad \text{subject to } X \succeq 0,\ X_{ii} = 1$$

Step 3: Solving the SDP

We solve the above semidefinite program using an SDP solver to obtain an optimal matrix X^* such that:

$$X^* \succeq 0, \quad X^*_{ii} = 1$$

Step 4: Cholesky Decomposition

Perform Cholesky decomposition on X^*:

$$X^* = RR^T$$

The rows of $R \in \mathbb{R}^{n \times n}$ represent vector embeddings r_i for each vertex i.

Step 5: Hyperplane Rounding

- Sample a random unit vector $u \sim \mathcal{N}(0, I_n)$ by:
- For each vertex i, define:

$$x_i = \begin{cases} 1 & \text{if } \langle r_i, u \rangle \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

- Vertices are split by the hyperplane $\langle r_i, u \rangle = 0$, inducing a cut.

Cutting a Sphere

- Each variable v_i lies on the unit sphere $\mathbb{S}^{n-1}$
- Closer v_i and v_j are, the smaller $1 - \langle v_i, v_j \rangle$
- Maximizing the objective means placing v_i, v_j "as opposite as possible" for high-weight edges

Relaxation Intuition

- Binary variables $x_i \in \{-1, 1\}$ are relaxed to vectors $v_i \in \mathbb{R}^n$
- Captures more geometric freedom
- Final cut is obtained by converting vector solution back to binary — using rounding

Algorithm Overview

1. Solve the relaxed SDP to get unit vectors $v_1, v_2, \dots, v_n$
2. Sample a random hyperplane through the origin
3. Partition the graph by assigning:

$$x_i = \begin{cases} 1 & \text{if } v_i \text{ is on one side of the hyperplane} \\ -1 & \text{otherwise} \end{cases}$$

4. This gives a valid cut of the graph

GW Algorithm (1/3): Input and SDP Formulation

Algorithm 1 Goemans-Williamson Max-Cut Approximation

Require: Weighted undirected graph $G = (V, E)$ with edge weights $w_{ij} \geq 0$

Ensure: Cut $(S, V \setminus S)$ with $\mathbb{E}[\text{cut}] \geq 0.878 \cdot \text{OPT}$

0: **Step 1: Construct Weight Matrix** $\{\mathcal{O}(|E|)\}$

0: $n \leftarrow |V|$

0: Initialize $W \leftarrow n \times n$ zero matrix

0: **for** $(i, j) \in E$ **do**

0:     $W_{ij} \leftarrow w_{ij}$, $W_{ji} \leftarrow w_{ij}$

0: **end for**

0: **Step 2: Setup SDP** $\{$Formulation: $\mathcal{O}(n^2)\}$

0: Define symmetric matrix $X \in \mathbb{R}^{n \times n}$ with:

    $X \succeq 0$ (positive semidefinite), $X_{ii} = 1$

0: Objective: $\max \frac{1}{4} \sum_{i,j} W_{ij}(1 - X_{ij})$ =0

GW Algorithm (2/3): SDP Solver and Vector Extraction

Algorithm 2 *

(continued)

Step 3: Solve the SDP $\{$Time: $\mathcal{O}(n^3)$ to $\mathcal{O}(n^4)\}$
$X^* \leftarrow$ SDP_Solver($X \succeq 0,\ X_{ii} = 1,$ objective)
Step 4: Extract Vectors from X^* $\{\mathcal{O}(n^3)\}$
Try Cholesky Decomposition: $X^* = V^\top V$
if Cholesky fails **then**
    Use Eigen-Decomposition: $X^* = Q\Lambda Q^\top$
    $V \leftarrow Q\sqrt{\Lambda}$
end if=0

GW Algorithm (3/3): Rounding and Final Cut

Algorithm 3 *

(continued)

0: **Step 5: Hyperplane Rounding** $\{\mathcal{O}(n)\}$

0: Sample $r \sim \mathcal{N}(0, I_n)$

0: Normalize: $r \leftarrow r/\|r\|$

0: $S \leftarrow \{i \in V \mid \langle V_i, r\rangle \geq 0\}$

0: **Return** $(S, V \setminus S) = 0$

Time Complexity Summary

- Step 1: Build weights — $\mathcal{O}(|E|)$
- Step 2: SDP setup — $\mathcal{O}(n^2)$
- Step 3: Solve SDP — $\mathcal{O}(n^3)$ to $\mathcal{O}(n^4)$
- Step 4: Decomposition — $\mathcal{O}(n^3)$
- Step 5: Rounding — $\mathcal{O}(n)$

Establishing the Approximation Ratio and Proof of Correctness

Approximation Guarantee

Assume that the algorithm returns S on input $G = (V, E, w)$.
For ease of analysis, assume:

- `SolveSDP` returns optimal matrix X.
- `Decomposition` returns exact matrix U (where $X = UU^T$).

Our goal is to show:

$$\mathbb{E}[W(S)] \geq \alpha \cdot OPT(G) \quad \text{where} \quad \alpha > 0.878$$

Defining Random Variables

Let Y_{ij} be the indicator random variable for edge (i, j):

$$Y_{ij} = \begin{cases} 1 & \text{if } i \in S, j \in T \text{ or vice versa} \\ 0 & \text{otherwise} \end{cases}$$

Then,

$$\mathbb{E}[W(S)] = \frac{1}{2} \sum_{i,j \in V} w_{ij} \cdot \mathbb{E}[Y_{ij}]$$

Rewriting Expectation

$$\mathbb{E}[W(S)] = \frac{1}{2}\sum_{i,j\in V} w_{ij}\cdot\mathbb{E}[Y_{ij}] = \frac{1}{2}\sum_{i,j\in V}\frac{w_{ij}\cdot\mathbb{E}[Y_{ij}]}{(1-X_{ij})}\cdot(1-X_{ij})$$

$\geq \min_{i,j\in V}\left\{\frac{2\,\mathbb{E}(Y_{ij})}{1-X_{ij}}\right\}\cdot\frac{1}{4}\sum_{i\in V}\sum_{j\in V} w_{ij}(1-X_{ij})$

$= \min_{i,j\in V}\left\{\frac{2\Pr(Y_{ij}=1)}{1-X_{ij}}\right\}\cdot\frac{1}{4}\sum_{i\in V}\sum_{j\in V} w_{ij}(1-X_{ij})$

$\geq \min_{i,j\in V}\left\{\frac{2\Pr(Y_{ij}=1)}{1-X_{ij}}\right\}\cdot\mathrm{OPT}(G)$

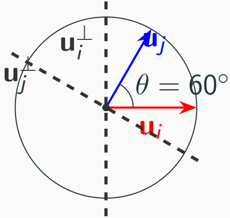
Expression for $\mathbf{Pr}(\mathbf{Y}_{ij} = 1)$

The event $\Pr(Y_{ij} = 1)$ occurs only when the i^{th} and j^{th} vertices lie on different sides of the cut. This is possible only when they give different signs for the inner product with the selected random vector $\mathbf{r}$.

Let θ be the angle between the vectors $\mathbf{u}_i$ and $\mathbf{u}_j$. We show the following:

$$\Pr(\mathbf{u}^\top \mathbf{r}_i \geq 0 \text{ and } \mathbf{u}^\top \mathbf{r}_j < 0) = \frac{\theta}{2\pi}$$

Contd...



Due to symmetry $\Pr(\mathsf{Y}_{ij} = 1) = 2 \cdot \Pr(\mathbf{u}^\top \mathbf{r}_i \geq 0 \text{ and } \mathbf{u}^\top \mathbf{r}_j < 0) = \frac{\theta}{\pi}$

Ratio calculation

Since $X = RR^\top$, we have

$$X_{ij} = \mathbf{r}_i^\top \mathbf{r}_j = \cos\theta_{ij}$$

From the analysis, we get

$$\Pr(Y_{ij} = 1) = \frac{\theta_{ij}}{\pi}$$

Using this, we compute:

$$\frac{2 \cdot \Pr(Y_{ij} = 1)}{1 - X_{ij}} = \frac{2\theta}{\pi(1 - X_{ij})} = \frac{2\theta}{\pi(1 - \cos\theta)}$$

Finally, we observe that:

$$\min_\theta \left\{ \frac{2\theta}{\pi(1 - \cos\theta)} \right\} > 0.878 \quad \text{(by simple calculus)}$$

0.878 Approximation

- Let OPT_{SDP} be the solution of the relaxed problem
- Expected value of rounded solution:

$$\mathbb{E}[\text{Cut}] \geq \alpha_{GW} \cdot \text{OPT}_{\text{SDP}}, \quad \alpha_{GW} \approx 0.878$$

- This is the best-known approximation ratio assuming the Unique Games Conjecture

Extensions and Conclusion

Key Takeaways

- Max-Cut is NP-hard, but approximation is possible
- SDP relaxation gives powerful convex optimization tool
- Goemans-Williamson algorithm achieves 0.878 ratio
- Impact spans theory and practice, with deep ties to complexity assumptions

Thank You!

Questions are welcome.