The Maximum Cut Problem

Hongyue Li

Abstract

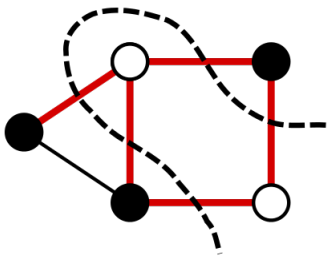
We introduce the Maximum-Cut problem and present a beautiful approximation algorithm for it.

1 Introduction

Definition 1.1 (Maximum Cut Problem(Max-Cut))**.**

Given an undirected graph $G = (V, E)$ with n vertices and m edges, a cut in G is a partition of the vertices V into two disjoint subsets $S, \bar{S} \subseteq V$, where $\bar{S} = V \setminus S$. Let $E(S, \bar{S})$ denote the set of edges with one vertex in S and one vertex in $\bar{S}$.

The Max-Cut problem is to find the cut $(S, \bar{S})$ that maximizes $|E(S, \bar{S})|$. In the weighted version of Max-Cut, we are also given an edge weight function $w : E \to \mathbb{R}$, and the problem is to find a cut with maximum weight. The Max-Cut problem is hard to solve. In contrast, the Min-Cut problem is solvable in polynomial time, e.g., by reducing to the s-t min-cut problem, which is dual to the Max-Flow problem, and can be solved efficiently.



Remark 1.2. *Max-Cut is NP-hard.*
The decision Max-Cut problem: For a given graph G and an integer k, does there exist a cut of size $\geq k$ in G?

There exist polynomial time reductions from the independent set problem, k-clique problem, 3-SAT problem to the decision Max-Cut problem.

Max-Cut is APX-hard. Assuming $P \neq NP$, there is no polynomial time algorithm that can compute $1-\epsilon$-approximation to the optimal solution for arbitrary $\epsilon > 0$. Furthermore, assuming $P \neq NP$, there is no polynomial time algorithm that can approximate Max-Cut to within a factor of $\frac{16}{17}$.

2 Three Simple Algorithms For Max-Cut

Algorithm 1: Random Algorithm

Input: Graph $G = (V, E)$
Output: Partition $(S, \bar{S})$
Assign each vertices $v \in V$ to $(S, \bar{S})$ at random with equal probability;
Output: $(S, \bar{S})$;

Remark 2.1. $\mathbb{E}[|E(S, \bar{S})|] = \frac{m}{2}$, *hence the random algorithm is a randomized 1/2-approximation algorithm for Max-Cut.*

Algorithm 2: Greedy Algorithm

Input: Graph $G = (V, E)$
Output: Partition $(S, \bar{S})$
Order the vertices $v_1, v_2, \ldots, v_n$;
Initialize two empty bins $S_1 \leftarrow \emptyset$, $S_2 \leftarrow \emptyset$;
`// Greedily assign all vertices`
for $i \leftarrow 1$ **to** n **do**
    if $|E(S_1 \cup \{v_i\}, S_2)| > |E(S_1, S_2 \cup \{v_i\})|$ **then**
        $S_1 \leftarrow S_1 \cup \{v_i\}$;
    else
        $S_2 \leftarrow S_2 \cup \{v_i\}$;
Output: (S_1, S_2);

Remark 2.2. *$|E(S_1, S_2)| \geq \frac{m}{2}$, hence the greedy algorithm is a deterministic 1/2-approximation algorithm for Max-Cut.*

Algorithm 3: Local Search

Input: Graph $G = (V, E)$

Output: Partition $(S, \bar{S})$ of V with a relatively large cut

```
1 Start with some arbitrary S ⊆ V;
2 while True do
3     if ∃v ∈ S such that moving v to V \ S increases the cut size then
4         S ← S \ {v};
5     if ∃v ∈ S̄ such that moving v to S increases the cut size then
6         S ← S ∪ {v};
7     if no vertices can be moved to improve S then
8         Terminate and output: (S, S̄);
```

Remark 2.3. *$|E(S, \bar{S})| \geq \frac{m}{2}$, hence the local search algorithm is a 1/2-approximation algorithm for Max-Cut.*

3 A Beautiful Approximation Algorithm For Max-Cut

Step 1: Semidefinite Programming

The Max-Cut problem can be formulated as the following Integer Programming problem:

$$\begin{aligned} \text{Maximize} \quad & \frac{1}{2} \sum_{(i,j) \in E} w_{ij}(1 - x_i x_j) \\ \text{subject to} \quad & x_i \in \{-1, +1\} \quad \forall i = 1, 2, \ldots, n \end{aligned}$$

We can relax the domain of each x_i from the discrete set $\{-1, +1\}$ to the (n-1)-dimensional sphere $\mathbb{S}^{n-1}$, such that $\{-1, +1\}$ can be embedded as a pair of antipodal points on $\mathbb{S}^{n-1}$.

$$\begin{aligned} \text{Maximize} \quad & \frac{1}{2} \sum_{(i,j)\in E} w_{ij}(1 - x_i \cdot x_j) \\ \text{subject to} \quad & x_i \in \mathbb{S}^{n-1} \quad \forall i = 1, 2, \ldots, n \end{aligned}$$

Let $P = X^T X$, where $X = [x_1, ..., x_n] \in \mathbb{R}^{n\times n}$, then P is a positive semidefinite matrix, denoted $P \succeq 0$. Conversely, for any $Q \succeq 0$, $Q = Y^T Y$ for some $Y \in \mathbb{R}^{n\times n}$. So the relaxed Max-Cut problem can be realized as the following Semidefinite Programming problem:

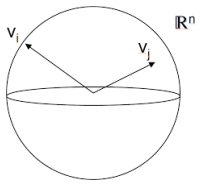
$$\begin{aligned} \text{Maximize} \quad & \frac{1}{2} \sum_{(i,j)\in E} w_{ij}(1 - P_{ij}) \\ \text{subject to} \quad & P \succeq 0 \\ & P_{ii} = 1 \quad \forall i = 1, 2, \ldots, n \end{aligned}$$

This is a convex optimization problem and can be solved to any degree of accuracy in polynomial time.

After we solve P, we can recover $X = \sqrt{\Lambda} Q^T$ where $P = Q\Lambda Q^T$ is the spectral decomposition of P.

Step 2: Random Hyperplane Rounding

Given $X = [x_1, ..., x_n] \in \mathbb{R}^{n\times n}$, we use random hyperplane rounding to construct a partition.



1. Sample a random direction r in $\mathbb{R}^n$: $r \sim \mathbb{N}(0, I_n)$.
2. For each vertex i=1,...,n, the decision rule is:

$$i \in \begin{cases} S & \text{if } x_i \cdot r \geq 0 \\ \overline{S} & \text{if } x_i \cdot r < 0 \end{cases}$$

3. Output the cut $(S, \bar{S})$

Theorem 3.1. $\mathbb{E}[|E(S,\bar{S})|] \geq \alpha^*$ *MaxCut, where*

$$\alpha^* = \min_{\theta \in [0,\pi]} \frac{\theta/\pi}{\frac{1-\cos\theta}{2}} \approx 0.878 \text{ (min at } \theta^* \approx 2.33).$$

We denote the true value of the maximum cut MaxCut, the value of the relaxed semidefinite program relaxed MaxCut.

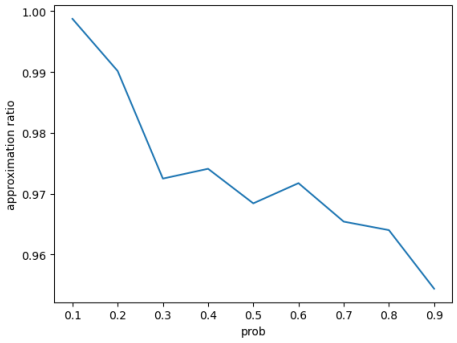
Proof.

$$\begin{aligned}
\mathbb{E}[|E(S,\overline{S})|] &= \sum_{(i,j)\in E} w_{ij} \text{Prob}(x_i \text{ and } x_j \text{ are separated by a random hyperplane}) \\
&= \sum_{(i,j)\in E} w_{ij} \frac{\arccos x_i \cdot x_j}{\pi} \\
&\geq \sum_{(i,j)\in E} w_{ij} \alpha^* \frac{1 - x_i \cdot x_j}{2} \\
&= \alpha^* \sum_{(i,j)\in E} w_{ij} \frac{1 - x_i \cdot x_j}{2} \\
&= \alpha^* \text{ relaxed MaxCut} \\
&\geq \alpha^* \text{ MaxCut}
\end{aligned}$$

□

Remark 3.2. α^* *is also the integrality gap of the semidefinite programming problem above, which means the randomized hyperplane rounding method is optimal.*

Example 3.3. *We generate random graphs $G(n,p)$ with n=10 nodes for various probabilities p ranging from 0.1 to 0.9 and plot the approximation ratio over 100 iterations. The approximation ratio in our simulation decreases to 0.95 as p decreases but is always greater than $\alpha^* \approx 0.878$.*



4 Hardness of Approximation: the Unique Game Conjecture

Definition 4.1 (The Probabilistically Checkable Proof(PCP) System)**.** *We say that there exists a PCP system* $PCP_{c,s}[r,q]_\Sigma$*(*$0 \le s \le c \le 1, r, q : \mathbb{N} \to \mathbb{N}$ *are integer functions) for a language* L *over alphabet* Σ *(denoted* $L \in PCP_{c,s}[r,q]_\Sigma$*) if the following holds:*

- *Given input* x*, the prover* P *writes down the proof* $\pi \in \Sigma^*$*. The polynomial-time probabilistic verifier* V *looks at* x*, accesses* $r(|x|)$ *random bits and then does some computation to choose* $q := q(|x|)$ *locations in the proof and to produce a deterministic test(predicate)* ψ *on the* q *chosen locations. Then* V *queries the* q *chosen locations in the proof* π*, runs the predicate* ψ *on these* q *bits, and accepts or rejects.*
- *Completeness: For* $x \in L, \exists \pi, \Pr[V(x,\pi) = 1] \ge c$*.*
- *Soundness: For* $x \notin L, \forall \pi, \Pr[V(x,\pi) = 1] \le s$*.*

Theorem 4.2 (PCP Theorem)**.**

$$NP = PCP_{1,\frac{1}{2}}[O(\log n), O(1)]_{\{0,1\}}.$$

Conjecture 4.3 (Unique Game Conjecture)**.** *For every sufficiently small pair of constants* $1-c, s > 0$*, there exists an alphabet* Σ *of constant size such that*

$$NP = PCP_{c,s}[O(\log n), 2]_\Sigma.$$

Theorem 4.4. *Assuming the Unique Games Conjecture, it is NP-hard to approximate Max-Cut to any factor greater than* α^**.*

5 Statistical Physics of Max-Cut

The Ising model for Max-Cut is:

$$H_{Ising} = \sum w_{ij} s_i s_j.$$

Minimizing this energy is equivalent to maximizing the cut.

In the quantum version, we encode each vertex as a qubit. A qubit in $|0\rangle$ or $|1\rangle$ indicates whether the vertex is in S or $\bar{S}$. The Pauli Z gate acts on the qubit such that $Z|0\rangle = |0\rangle, Z|1\rangle = -|1\rangle$. The Hamiltonian of Max-Cut is:

$$\begin{aligned} H_C &= \sum_{(u,v)\in E} \frac{Z_u + I}{2} + \frac{Z_v + I}{2} - 2 \cdot \frac{Z_u + I}{2} \frac{Z_v + I}{2} \\ &= \sum_{(u,v)\in E} \frac{Z_u + Z_v + 2I - (Z_u Z_v + Z_u + Z_v + I)}{2} \\ &= \sum_{(u,v)\in E} \frac{I - Z_u Z_v}{2}. \end{aligned}$$

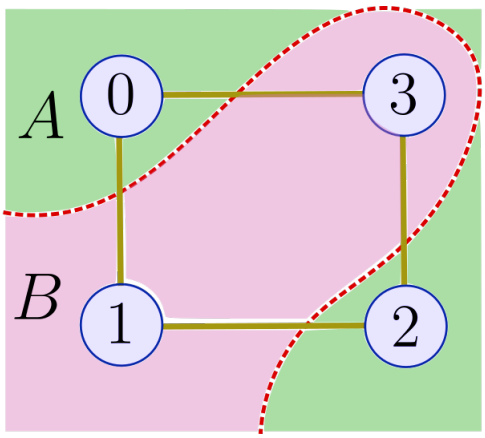
The expected value for a given state ψ is:

$$\begin{aligned} \langle\psi|H_C|\psi\rangle &= \langle\psi| \sum_{(u,v)\in E} \frac{I - Z_u Z_v}{2} |\psi\rangle \\ &= \langle\psi| \sum_{(u,v)\in E} \frac{I}{2} |\psi\rangle - \langle\psi| \sum_{(u,v)\in E} \frac{Z_u Z_v}{2} |\psi\rangle \\ &= \frac{|E|}{2} - \frac{1}{2} \langle\psi| \sum_{(u,v)\in E} Z_u Z_v |\psi\rangle. \end{aligned}$$

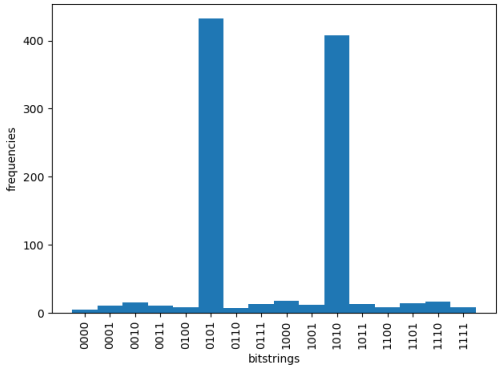
Our goal is to maximize this value, which is equivalent to minimizing the energy of the Hamiltonian $H = -\sum_{(u,v)\in E} Z_u Z_v$.

Example 5.1. *We can use the Quantum Approximate Optimization Algorithm to minimize this energy. We did a simple experiment on a small graph with 4 nodes and 4 edges.*

The states $|0101\rangle, |1010\rangle$ have the highest frequencies, both corresponding to the cut $\{0, 2\}, \{1, 3\}$.



(a) Graph



(b) Relative Frequencies For Each State

Figure 1: Overall Caption for Both Graphs

6 Conclusions

We studied the maximum cut problem, presented various approximation algorithms, and realized the hardness of approximation.

7 Code

The code is here: `https://github.com/ordinarylhy/MaxCut`.