1 Introduction

Finding the max cut in a graph has many applications, including circuit design [1]. Studying Max Cut has also given insight into how to solve other combinatorial optimization problems (for example, [2] has used similar ideas to solve Max Cut and MAX-3SAT).

There have historically been two approaches to the max cut problem. One is a more extremal approach – finding bounds for the max cut for graphs with certain property (i.e. triangle-free, K_r-free, H-free). The other approach has been to algorithmically compute max cuts by finding approximations of it. The focus of this survey will be on the latter, although it will touch a little bit on a recent application of SDPs to the extremal max cut problem

This survey will present the approach taken by (Goemans et al. 1995 [2]) and will discuss improvements made upon that, as well as open directions, for the Max Cut problem.

We will be dealing with weighted undirected graphs in this survey (although some of the examples will restrict to graphs where the edges have weight 1). For a graph $G = (V, E)$, and for $(i, j) \in \mathbb{N}\times\mathbb{N}$, let the weight w_{ij} equal the weight of the edge from i to j (if $(i, j) \in E$) or 0 otherwise. Define the weight of a subgraph H to be $w(H) = \sum_{i,j\in V(H),i<j} w_{ij}$. Define a *cut* to be a partition of $V(G)$ into S and $\overline{S}$ for some $S \subset V(G)$. Define the weight of the cut $w(S, \overline{S})$ to be $\sum_{i\in S} \sum_{j\in\overline{S}} w_{ij}$.

2 Problem statement

Given a weighted undirected graph $G = (V, E)$ and a weight function $w : v \times V \to \mathbb{R}^+$, we want to find the max cut, i.e.

$$\max_{S\in V} w(S, \overline{S}) = \max_{S\subset V} \sum_{i\in S} \sum_{j\notin S} w_{ij}$$

The entire problem is NP-hard (even for all weights equal to 1), so one thing we are interested in is finding approximations for the max cut with particular guarantees. We will require the following definition which is crucial in the study of approximation algorithms:

Definition (α-approximation): For a maximization problem, let OPT denote the optimal solution. Define an algorithm to be an α-approximation algorithm if it can return a solution with value at least $\alpha \cdot OPT$.

From now on, we will be using OPT to refer to the optimal solution to Max Cut, and Z to refer to the optimal solution of a relaxation.

3 0.5-approximation algorithm for Max Cut

There is a trivial algorithm to find a 0.5-approximation. Label the vertices of G as $v_1, v_2, ..., v_n$. Take each vertex one-by-one, and assign it to either S or $\overline{S}$. When adding the ith vertex, the total weight of the edges that gets added to $G[S \cup \overline{S}]$ is equal to $W = w(G[N_G(v_i) \cap \{v_1, ..., v_{i-1}\}])$. Assign v_i to either S or $\overline{S}$ such that at least $\frac{1}{2}W$ gets added to the cut.

Thus, we guarantee that at least half of the total weight gets added to the cut, so the above is a 0.5-approximation for Max Cut.

(Goemans et al. [2]) stated that some earlier works were able to obtain slightly better approximation algorithms:

- $\frac{1}{2} + \frac{1}{2m}$ (Vitányi 1981)
- $\frac{1}{2} + \frac{n-1}{4m}$ (Poljak and Turzík 1982)
- $\frac{1}{2} + \frac{1}{2n}$ (Haglin and Venkatesan 1991
- $\frac{1}{2} + \frac{1}{2\Delta}$ (Hofmeister and Lefmann 1995)

As we will see in the next section, we can do much better.

4 Using SDPs to approximate the max cut

We will open with an explanation of the ideas used in the paper (Goemans et al. 1995, [2]) that motivated a significant amount of future work related to using semidefinite programming for combinatorial optimization. We will present a .878-approximation algorithm for Max Cut in this section.

4.1 Quadratic program formulation

The Max Cut problem can be expressed as the following integer quadratic program:

$$\begin{aligned} \text{maximize} \quad & \frac{1}{2} \sum_{(i,j)\in E} w_{ij}(1 - x_i x_j) \\ \text{subject to} \quad & x_i \in \{-1, 1\}, \qquad i = 1, \dots, n \end{aligned} \tag{Q}$$

Here, $x_i = 1$ if $v_i \in S$ and -1 otherwise. We can verify that the objective is equivalent to $w(S, S')$. For $(i, j) \in E$, if $v_i and v_j$ are in the same set in (S, S'), this contributes 0 to the term $1 - x_i x_j$. If one is in S and the other in S', this contributes 2 to $1 - x_i x_j$ and thus contributes w_{ij} to the objective. This integer quadratic program is NP-hard to solve.

4.2 Relaxation to SDP

Each x_i can be thought of as a unit vector aligned with the ith axis in $\mathbb{R}^n$. We will relax the quadratic program such that each x_i will be any arbitrary vector in S^n (the unit n-dimensional sphere).

$$\begin{aligned} \text{maximize} \quad & \frac{1}{2} \sum_{(i,j)\in E} w_{ij}(1 - v_i \cdot v_j) \\ \text{subject to} \quad & v_i \in S^n, \qquad\qquad i = 1, \ldots, n \end{aligned} \tag{SDP-CUT}$$

[3] refers to this formulation of semidefinite programming as *vector programming.*

We can see that the solution to Q is feasible to the solution of SDP-CUT, i.e. when $v_i \in \pm e_i$ for each i. Thus, the optimal objective in SDP-CUT is an upper bound for OPT.

4.3 Algorithm

(Goemans et al. 1995) gives the following algorithm for approzimating the max cut:

(i) Solve SDP-CUT and obtain vectors $v_1, v_2, ..., v_n \in \mathbb{R}^n$.

(ii) Obtain a cut $(S, \overline{S})$ as follows. Sample a vector r uniformly from S^n, and for each i, assign vertex i to S if $\langle v_i, r\rangle > 0$, and assign vertex i to $\overline{S}$ otherwise.

The technique in step (ii) is now well-known as *random hyperplane rounding.* Visually, what is happening is that a random hyperplane that contains the origin is sampled (r is a vector orthogonal to this hyperplane), and each half-space contains the vertices on one side of the cut.

It can be shown (will not prove here) that step (ii) can equivalently be seen as sampling from n independent standard normal distributions and projecting onto the unit sphere. This is useful for more advanced techniques like RPR^2, which will be discussed in Section 6.

4.4 Guarantees

Theorem: In general, the above algorithm can produce an α-approximation, where $\alpha = \min_{\theta>0} \frac{\theta(1-\cos\theta)}{2\pi} \approx$ 0.878-approximation to Max Cut.

Proof:

Let $v_1, v_2, ..., v_n$ be the vectors in the optimal solution in SDP-CUT, and let Z be the value of the objective function in the optimal solution.

We will consider the contribution of each edge to the max cut. For an edge (i, j), define θ_{ij} to be the angle between v_i and v_j. The contribution of this edge to Z is equal to $\frac{1}{2}w_{ij}(1 - v_i \cdot v_j) = \frac{1}{2}w_{ij}(1 - \cos\theta_{ij})$, where θ_{ij} is the angle between vectors v_i and v_j. Using the notation in [4], call this contribution $SDP(\theta_{ij})$

It can be proven (and intuitively seen) that randomly sampling r gives that $sign(\langle v_i, r\rangle) \neq sign(\langle v_j, r\rangle)$ with probability $\frac{\theta_{ij}}{\pi}$. We can ignore the case where $\theta_{ij} = 0$ (since this will always contribute 0 to the max cut), so the expected contribution of this edge will be $w_{ij}\frac{\theta_{ij}}{\pi}$

To lower bound the approximation constant, we can just lower bound $\frac{w_{ij}\frac{\theta_{ij}}{\pi}}{SDP(\theta_{ij})} = \min_{\theta>0}\frac{\theta(1-\cos\theta)}{2\pi} \approx 0.878$. Thus, $E[cut] \geq \alpha \cdot Z \geq \alpha \cdot OPT$, as desired. $\square$

We have that the bound is tight when θ_{ij} is a minimizer of $\frac{\theta(1-\cos\theta)}{2\pi}$, i.e. $\theta_{ij} \approx 2.33$ or when $\theta_{ij} = 0$.

4.5 Guarantees when the max cut is large

Goemans et al. 1995 [2] also give guarantees when $Z = tW$, where Z is the optimal objective function in SDP-CUT, and W is the total weight of all edges.

A function $h(t) = \frac{\arccos 1-2t}{\pi}$ is defined, and for $t > \gamma = .84458$ (the optimal value of $h(t)$), the following guarantee holds:

Theorem (adapted from Goemans-Williamson 1995) When $Z = tW$, then

$$E[cut] \geq \alpha_t Z$$

where $\alpha_t = \frac{h(t)}{t}$

It can be verified that $\alpha_t \geq \alpha$, and equality is attained only if $t = \alpha$.

From now on, we will assume the edge weights to be 1. Many of the below results can be generalized to positive real edge weights, although some may possibly be with worse guarantees.

5 A brief note about hardness of approximation of MAX CUT

Hastad proved (using a variant of the PCP theorem) that a $(\frac{16}{17} + \epsilon)$-approximation for max cut implies that $P = NP$. In fact, the Unique Games Conjecture implies that there an $(\alpha + \epsilon)$-approximation implies that $P = NP$. More details can be seen in [5].

6 Some Improvements in Special Cases

6.1 Graphs with bounded degree

Feige et al. 2002 [6] study approximations of max cut for graphs with maximum degree $\leq \Delta$ by using an algorithm similar to Goemans and Williamson, but with extra constraints and a greedy update step. Essentially it is the following:

$$\begin{aligned} \text{maximize} \quad & \frac{1}{2} \sum_{(i,j) \in E} (1 - v_i \cdot v_j) \\ \text{subject to} \quad & (1) v_i \in S^n, \qquad\qquad i = 1, \dots, n \\ & (2) \langle v_i, v_j \rangle + \langle v_i, v_k \rangle + \langle v_j, v_k \rangle >= 1 \\ & \langle v_i, v_j \rangle - \langle v_i, v_k \rangle - \langle v_j, v_k \rangle \geq 1, \qquad \forall i, j, k \in [n] \end{aligned} \tag{SDP-Delta}$$

Constraint (2) is also referred to as the "triangle" constraint.

The paper defines a vertex to be "misplaced" if it is on the same side as at least half of its neighbors. [6] argues that constraint (2) allows one to lower bound misplaced vertices (after solving the relaxation and performing random hyperplane rounding), and proposes a greedy strategy to get rid of the misplaced vertices.

For vertices of degree at most 3 (with an additional SDP constraint that $\langle v_i, v_j \rangle + \langle v_i, v_k \rangle + \langle v_j, v_k \rangle = -1 \forall (i, j, k)$ such that i is misplaced and j and k are neighbors of i), [6] gives a 0.921-approximation for Max Cut. For general maximum degree $\leq \Delta$, they prove that SDP-Delta results in an $\alpha + \Omega(\frac{1}{\Delta^2})$-approximation.

6.2 Outward Rotation

Zwick 1999 [7] gives a modification to the Goemans-Williamson algorithm for tighter guarantees when the max cut is small. The high-level idea is projecting the solution to the SDP algorithm to a higher dimensional space and performing random hyperplane rounding from there.

Recall SDP-CUT (remember we are now assuming the weights are equal to 1):

$$\begin{aligned} \text{maximize} \quad & \frac{1}{2} \sum_{(i,j) \in E} (1 - v_i \cdot v_j) \\ \text{subject to} \quad & v_i \in S^n, \qquad\qquad i = 1, \dots, n \end{aligned}$$

For some $\gamma \in (0, 1)$, the algorithm does the following:

(i) Solve SDP-CUT to obtain $v_1, v_2, ..., v_n$.

(ii) For $e_1, e_2, ...e_n$ (vectors orthogonal to $v_1, v_2, ..., v_n$ in $\mathbb{R}^{2n}$), create (for each i) $\hat{v_i} = \sqrt{1-\gamma}v_i + \sqrt{\gamma}e_i$.

(iii) Perform random hyperplane rounding on $\hat{v_1}, ..., \hat{v_n}$ to obtain an optimal solution to Max Cut.

For $Z = tW$ with $t < 0.844$, [7] proves that this algorithm guarantees an approximation ration of $\alpha_t > \alpha$.

6.3 RPR^2 Rounding Technique

Fiege et al. 2006 [4] give a general method to lessen the number of misplaced vertices: RPR^2 (random projection, randomized rounding). While the analysis doesn't carry to the above case where the vertices have bounded degree, it does apply to other cases, such as when the max cut is within $0.6|E(G)|$. The idea is similar Goemans-Williamson, but the random hyperplane rounding step is more general.

Recall it was mentioned in Section 4.3 that sampling a random vector in S^n is equivalent to sampling n unit Gaussians independently and projecting their cartesian product onto S^n. Here, we will avoid the projection step and assign vertices to a cut with respect to a function f (rather than assigning based on $\langle v_i, r \rangle$.

Below is the formal statement of the RPR^2 rounding technique (with respect to a function f):

(i) Solve SDP-CUT to obtain $v_1, v_2, ..., v_n$.

(ii) Sample $r_1, r_2, ..., r_n$ independently from $\mathcal{N}(0, 1)$, and let $r = (r_1, r_2, ..., r_n)$.

(iii) Create $x_1, x_2, ..., x_n$ where $x_i = \langle v_i, r \rangle$.

(iv) With probability $f(x_i)$, assign vertex i to S. Otherwise assign vertex i to $\overline{S}$.

Note that when $f(x_i) = 1$ if $x_i > 0$ or 0 otherwise, this is exactly the Goemans-Williason algorithm.

It is in fact the case that outward rotation is a special case of RPR^2 (where $f(x) = \Phi\left(x\sqrt{\frac{1-\gamma}{\gamma}}\right)$). Feige was able to make improvements to the approximation constant for light max cut.

The RPR technique has also been used to make improvements in other problems (see section 7).

7 Related problems

7.1 Max bisection

Max Bisection is exactly the Max Cut problem, but with the additional constraint that the vertices in both partitions have the equal. The RPR^2 technique was used to get the approximation constant up to .7028 by Fiege et al. 2006 [4]. There have been further improvements using the Lasserre Hierarchy and relaxations of it (up to 0.8776, Austrin et al. 2012 [8]).

7.2 Other problems

There are also other problems that fall in a similar vein, such as max bisection with unlimited balance, max k-cut, and generalizations of these problems to the hypergraph setting.

8 Application to the extremal Max Cut problem: Lower bounds on the max cut for H-free graphs

Semidefinite programming has recently been applied to improve classical bounds ([9]) regarding the largest bipartite subgraph of a graph. Denote $b(G)$ to indicate the largest bipartite subgraph of G. It is well known that $b(G) \geq \frac{1}{2}$, and an extensively researched problem has been to analyze the error term $b(G) - \frac{1}{2}$, or *surplus* for different classes of graphs.

The following recent theorem, which is a novel application of SDPs to the extremal max cut problem, trivializes some earlier results.

Theorem (Carlson et al. 2020 [9]) Let G be a graph with n vertices and m edges, and for each $i \in [n]$, let V_i be a subset of i's neighbors, and let $\epsilon_i \leq \frac{1}{\sqrt{|V_i|}}$. Then

$$b(G) \geq \frac{m}{2} + \sum_{i=1}^{n} \frac{\epsilon_i |V_i|}{4\pi} - \sum_{(i,j)\in E} \frac{\epsilon_i \epsilon_j |V_i \cap V_j|}{2}$$

This is proved using SDP's as follows:

Proof sketch (see [9] for the full details):

For each $i \in [n]$, create a vector $\hat{v^{(i)}} \in \mathbb{R}^n$ such that $v_j^{(i)} = 1$ if $i = j$, $-\epsilon_i$ if $j \in V_i$, or 0 otherwise. Then, letting $v_i := \frac{v^{(i)}}{||v_i||}$, it can be shown that $v_1, v_2, ..., v_n$ form a solution to SDP-CUT. Performing random hyperplane rounding, [9] lower bounds the probability of any arbitrary edge being in the cut, and summing gives Theorem 1.1.

This theorem has been used to prove and conjecture a couple of lower bounds (mentioned in [9]).

Recall Shearer's bound:

Theorem (Shearer 1992) For a triangle-free graph with m edges, $b(G) \geq \frac{1}{2}m + \frac{1}{8\sqrt{2}} \sum_{v \in V} \sqrt{d(v)}$.

It is actually an immediate consequence of Carlson et al. 2020 (up to a constant) by letting V_i be the neighborhood of i, and let $\epsilon_i = d_i$, where d_i is the degree of vertex i.

The above theorem also implies the following result about K_r-free graphs:

Theorem (Carlson et al. 2020) Let G be a d-degenerate $K_r - free$ graph $r \geq 3$. Then, there exists a constant $c = c(r) > 0$ such that

$$b(G) \geq \left(\frac{1}{2} + \frac{c}{d^{\frac{(l-1}{2r-4}}} \right) m$$

Carlson et al. also propose the following two conjectures from the above theorems:

Conjecture 1 For d-degenerate H-free graphs, there exists $c = c(H) > 0$ such that

$$b(G) \geq \left(\frac{1}{2} + \frac{c}{\sqrt{d}} \right) m$$

It can be shown that the above conjecture implies the following conjecture:

Conjecture 2 For H-free graphs, there exists $\epsilon = \epsilon(H) > 0, c = c(H) > 0$ such that

$$b(G) \geq \frac{m}{2} + cm^{\frac{3}{4}+\epsilon}$$

9 Future work

For a little over the last two decades, semidefinite programming has been frequently used in combinatorial optimization, but it would be interesting to analyze the extent to which it is used in extremal problems, like the ones in Section 8.

The Lasserre hierarchy (a structured way of making more and more tight relaxations) has been useful for obtaining strong approximation ratios for Max Bisection [8], and it could be worthwhile to investigate how well it works for other variants of Max Cut and how well it compares with others.

References

[1] Michael Jünger Francisco Barahona, Martin Grötschel and Gerhard Reinelt. An application of combinatorial optimization to statistical physics and circuit layout design. 1987.

[2] Michael X. Goemans and David P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. 1995.

[3] David P. Williamson and David B. Shmoys. *The Design of Approxiation Algorithms*. 2010.

[4] Langberg M. Feige U. The rpr^2 rounding technique for semidefinite programs. 2006.

[5] Jake Wellens. Fourier analysis and inapproximability for max-cut: a case study. 2016.

[6] Langberg M. Feige U, Karpinski M. Improved approximation of max-cut on graphs of bounded degree. 2002.

[7] Uri Zwick. Outward rotations: A tool for rounding solutions of semidefinite programming relaxations, with applications to max cut and other problems. In *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing*, STOC '99, page 679–687, New York, NY, USA, 1999. Association for Computing Machinery.

[8] Per Austrin, Siavosh Benabbas, and Konstantinos Georgiou. Better balance by being biased: A 0.8776-approximation for max bisection, 2012.

[9] Charles Carlson, Alexandra Kolla, Ray Li, Nitya Mani, Benny Sudakov, and Luca Trevisan. Lower bounds for max-cut in h-free graphs via semidefinite programming, 2020.