SDPs for Max Cut Approximations

Arvind Ramaswami

July 25, 2021

Max Cut Problem

Definiton: 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.

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

Max Cut Problem

Definiton: 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.

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

Computing the optimal solution to MAX CUT is NP-hard (even when all the weights are equal to 1).

Max Cut Problem

Definiton: 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.

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

Computing the optimal solution to MAX CUT is NP-hard (even when all the weights are equal to 1).

There has however been work to approximate the optimal solution in polynomial time.

Approximation definition

Definition: 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 $\alpha \cdot OPT$.

Approximation definition

Definition: 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 $\alpha \cdot OPT$.

$\frac{1}{2}$**-approximation algorithm for MAX CUT**: Assign each vertex one by one to either S or $\overline{S}$ (pick whichever side will result in a larger cut size with the vertices that are already assigned). This will return a solution of size $\geq \frac{1}{2} \sum_{i \in S} \sum_{j \notin S} w_{ij} = \frac{1}{2} \cdot OPT$.

Approximation definition

Definition: 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 $\alpha \cdot OPT$.

$\frac{1}{2}$**-approximation algorithm for MAX CUT**: Assign each vertex one by one to either S or $\overline{S}$ (pick whichever side will result in a larger cut size with the vertices that are already assigned). This will return a solution of size $\geq \frac{1}{2} \sum_{i \in S} \sum_{j \notin S} w_{ij} = \frac{1}{2} \cdot OPT$.

Slightly improved approximation algorithms for unweighted MAX CUT (listed in [1]):

- $\frac{1}{2} + \frac{1}{2m}$ (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)

Approximation definition

Definition: 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 $\alpha \cdot OPT$.

$\frac{1}{2}$**-approximation algorithm for MAX CUT**: Assign each vertex one by one to either S or $\overline{S}$ (pick whichever side will result in a larger cut size with the vertices that are already assigned). This will return a solution of size $\geq \frac{1}{2} \sum_{i \in S} \sum_{j \notin S} w_{ij} = \frac{1}{2} \cdot OPT$.

Slightly improved approximation algorithms for unweighted MAX CUT (listed in [1]):

- $\frac{1}{2} + \frac{1}{2m}$ (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)

Major improvement using SDPs: 0.878-approximation (Goemans Williamson 1995 [1])

Improved MAX CUT approximation with SDPs

Quadratic Programming Formulation

Let the vertices of the graph be enumerated as $v_1, ..., v_n$.

Define $x_1, x_2, ..., x_n \in -1, 1$ such that $x_i = 1$ if $x_i \in S$ in the partition $(S, \overline{S})$, and let $x_i = -1$ otherwise.

Quadratic Programming Formulation

Let the vertices of the graph be enumerated as $v_1, ..., v_n$.

Define $x_1, x_2, ..., x_n \in -1, 1$ such that $x_i = 1$ if $x_i \in S$ in the partition $(S, \overline{S})$, and let $x_i = -1$ otherwise.

We can express the max cut problem as the following integer quadratic program:

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

Quadratic Programming Formulation

Let the vertices of the graph be enumerated as $v_1, ..., v_n$.

Define $x_1, x_2, ..., x_n \in -1, 1$ such that $x_i = 1$ if $x_i \in S$ in the partition $(S, \overline{S})$, and let $x_i = -1$ otherwise.

We can express the max cut problem as the following integer quadratic program:

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

Solving an integer quadratic program is NP-hard, so we want to reduce this to something more tractible.

SDP Relaxation

We want an approximate solution to

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

We can think about each x_i as unit vector in an axis.

SDP Relaxation

We want an approximate solution to

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

We can think about each x_i as unit vector in an axis.

We can relax it as follows:

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

SDP Relaxation

We want an approximate solution to

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

We can think about each x_i as unit vector in an axis.

We can relax it as follows:

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

Let Z be the optimal value attained by **??**. Every solution in Q is feasible in SDP-CUT, so Z is an upper bound to OPT.

Goemans-Williamson Max Cut Approximation

Recall the SDP relaxation:

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

Goemans-Williamson Max Cut Approximation

Recall the SDP relaxation:

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

It can be shown that the solution Z can be solved (within ϵ) in polynomial time. Consider the following algorithm for obtaining 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.

Goemans-Williamson Max Cut Approximation

Recall the SDP relaxation:

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

It can be shown that the solution Z can be solved (within ϵ) in polynomial time. Consider the following algorithm for obtaining 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.

Step (ii) is often referred to as "random hyperplane rounding."

Proof of Goemans-Williamson

Let $v_1, ..., v_n$ be the vectors, and let Z be the value of the obtimal objective of SDP-CUT.

Proof of Goemans-Williamson

Let $v_1, ..., v_n$ be the vectors, and let Z be the value of the obtimal objective of SDP-CUT.

For an edge (i, j), let θ_{ij} be the angle between v_i and v_j. The probability that they are located on opposite sides of the cut is equal to $\frac{\theta_{ij}}{\pi}$

Proof of Goemans-Williamson

Let $v_1, ..., v_n$ be the vectors, and let Z be the value of the obtimal objective of SDP-CUT.

For an edge (i, j), let θ_{ij} be the angle between v_i and v_j. The probability that they are located on opposite sides of the cut is equal to $\frac{\theta_{ij}}{\pi}$

The contribution of θ_{ij} to the objective function is equal to $w_{ij}(1 - v_i \cdot v_j) = \frac{w_{ij}(1-cos\theta_{ij})}{2}$. Call this $SDP(\theta_{ij})$

Proof of Goemans-Williamson

Let $v_1, ..., v_n$ be the vectors, and let Z be the value of the obtimal objective of SDP-CUT.

For an edge (i, j), let θ_{ij} be the angle between v_i and v_j. The probability that they are located on opposite sides of the cut is equal to $\frac{\theta_{ij}}{\pi}$

The contribution of θ_{ij} to the objective function is equal to $w_{ij}(1 - v_i \cdot v_j) = \frac{w_{ij}(1-cos\theta_{ij})}{2}$. Call this $SDP(\theta_{ij})$

It suffices to minimize $\frac{E[Cut_{ij}]}{SDP(\theta_{ij})}$, which is $\min_{\theta>0} \frac{\theta(1-\cos\theta)}{2\pi} \approx 0.878$. Thus, Goemans-Williamson gives a solution that is $\geq \alpha Z \geq \alpha OPT$, where $\alpha \approx 0.878$.

This is proven to be tight for general graphs since there are cases where $OPT = \alpha * Z$ (i.e. the program has *integrality gap* α).

Guarantees when the max cut is large

Goemans and Williamson also showed that when $Z = tW$, where W is the total weight of all edges, for $t > .84458$, the algorithm returns a cut of size $\geq \alpha_t * OPT$, where $\alpha_t = \frac{h(t)}{t}$, where $h(t) = \frac{\arccos 1-2t}{\pi}$.

Guarantees when the max cut is large

Goemans and Williamson also showed that when $Z = tW$, where W is the total weight of all edges, for $t > .84458$, the algorithm returns a cut of size $\geq \alpha_t * OPT$, where $\alpha_t = \frac{h(t)}{t}$, where $h(t) = \frac{\arccos 1-2t}{\pi}$. The above ratio is strictly $> \alpha$ unless $t = \alpha$.

A note about hardness of approximation

It has been proven that if there is an $\frac{16}{17} + \epsilon$-approximation for Max Cut, then $P = NP$ (proved using theory related to PCP hardness).

It has also been shown that if there is an $\alpha + \epsilon$-approximation, then this would disprove the Unique Games Conjecture.

A detailed survey about this exists in [2].

Improved techniques

Graphs of bounded degree

We will now assume edge weights have value 1.

[3] uses the following SDP formulation:

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

(SDP-Delta)

Graphs of bounded degree

We will now assume edge weights have value 1.

[3] uses the following SDP formulation:

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

(SDP-Delta)

(2) is referred to as the "triangle constraint" – allows guarantees of misplaced vertices in the analysis, so one can make greedy steps afterward to improve the cut. Obtained bounds were 0.921 for $\Delta(G) \leq 3$ and $\alpha + \Omega(\frac{1}{\Delta^2})$ for general Δ.

RPR^2 technique

Similar to Goemans-Williamson but more general (does hyperplane rounding based on a function f). Developed by Fiege 2006 [4]

RPR^2 technique

Similar to Goemans-Williamson but more general (does hyperplane rounding based on a function f). Developed by Fiege 2006 [4]

Algorithm:

(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 $\overline{S}$.

This is just Goemans-Williamson if we let $f(x_i)$ be 1 when $x_i > 0$ and 0 otherwise.

Has improved guarantees for when the $Z = tW$ for $t < 0.844$ (i.e. when the max cut is small) over other algorithms like outward rotation (see survey).

Related problems

Max Bisection: Max Cut but with the constraint that the both sides of the cut must have equal size.

Related problems

Max Bisection: Max Cut but with the constraint that the both sides of the cut must have equal size.

Fiege et al. 2006 [4] used RPR^2 for a $.7028$ approximation ratio.

Related problems

Max Bisection: Max Cut but with the constraint that the both sides of the cut must have equal size.

Fiege et al. 2006 [4] used RPR^2 for a $.7028$ approximation ratio.

There have been further improvements using the Lasserre Hierarchy and relaxations of it up to 0.8776 [5]

Related problems

Max Bisection: Max Cut but with the constraint that the both sides of the cut must have equal size.

Fiege et al. 2006 [4] used RPR^2 for a $.7028$ approximation ratio.

There have been further improvements using the Lasserre Hierarchy and relaxations of it up to 0.8776 [5]

Other related problems: Max k-cut, Max cut with limited unbalance, generalizations of these problems to hypergraphs

Application to the extremal Max Cut problem

Denote $b(G)$ to indicate the largest bipartite subgraph of G.

Application to the extremal Max Cut problem

Denote $b(G)$ to indicate the largest bipartite subgraph of G.

New result based on semidefinite programming:

Theorem (Carlson et al. 2020 [6]) 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

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

Note: plugging in $|V| = \sqrt{d}_i$ and $\epsilon = \frac{1}{\sqrt{d_i}}$ implies Shearer's theorem

Application to the extremal Max Cut problem (bounding $b(G)$ for H-free graphs)

Denote $b(G)$ to indicate the largest bipartite subgraph of G.

Application to the extremal Max Cut problem (bounding $b(G)$ for H-free graphs)

Denote $b(G)$ to indicate the largest bipartite subgraph of G.

New result based on semidefinite programming:

Theorem (Carlson et al. 2020 [6]) 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

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

Note: plugging in $|V| = \sqrt{d}_i$ and $\epsilon = \frac{1}{\sqrt{d_i}}$ implies Shearer's theorem

Application to the extremal Max Cut problem (bounding $b(G)$ for H-free graphs)

Denote $b(G)$ to indicate the largest bipartite subgraph of G.

New result based on semidefinite programming:

Theorem (Carlson et al. 2020 [6]) 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

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

Note: plugging in $|V| = \sqrt{d}_i$ and $\epsilon = \frac{1}{\sqrt{d_i}}$ implies Shearer's theorem

Proof outline: 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. This is an optimal SDP solution, so lower bound $b(G)$ after applying random hyperplane rounding.

Extremal Max Cut problem – implications

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

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

Extremal Max Cut problem – implications

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

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

This leads to new conjectures

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

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

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

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

Future work

It would be interesting to analyze the extent to which semidefinite programming used in extremal min cut problems, like the one earlier.

The Lasserre hierarchy (a structured way of making more and more tight relaxations) has brought good ratios for Max Bisection [5]. It would be interesting to see how well it does on other similar problems.

Bibliography

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

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

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

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

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

[6] 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.