Tree-width driven SDP for Max Cut

Сергей Аникин

Московский физико-технический институт

Курс: Автоматизация научных исследований
(практика, В. В. Стрижов)/Группа Б05-125

Научный руководитель: Александр Булкин
ProjectLink

Май 2024

Motivation

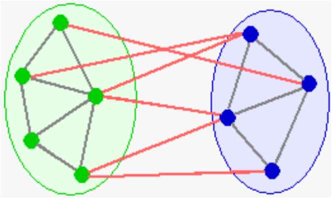
- Max Cut problem has applications in many spheres, including machine learning, theoretical computer science, and even theoretical physics.
- It serves as a basis for developing approximation algorithms and heuristic methods for solving other optimization problems.

Problem statement

Let's define W(S) to be the weight of the cut:

$$W(S) = \sum_{i \in S} \sum_{j \notin S} w_{ij}$$

Our goal is to find in polynomial time cut $S_{found} \subseteq V$, such that the value $W(S_{found})$ is as big as possible



$$W(S_{found}) \to \max$$

Our goal

If Unique Games Conjecture is true, it is known to be impossible to find solution, which is asymptotically better, than the one with constant 0.878...
Our goal is to develop a non-asymptotic improvement of the solution in polynomial time on arbitrary graphs by applying tree-width approach to the methods for solving SDP.

Problem statement

Let matrix L be L =
$$\begin{bmatrix} (\sum_{i=1}^{n} w_{1i}) & -w_{12} & -w_{13} & \dots & -w_{1n} \\ -w_{21} & (\sum_{i=1}^{n} w_{2i}) & -w_{23} & \dots & -w_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -w_{n1} & -w_{n2} & -w_{n3} & \dots & (\sum_{i=1}^{n} w_{ni}) \end{bmatrix}$$
Later we call such matrix the Laplacian of the graph.
Our task is equivalent to finding

$$OPT = \max_{x_i^2=1} x^T Lx$$

since

$$\max_{S \subseteq V} W(S) = \frac{1}{4} \max_{x_i^2=1} x^T Lx$$

Standard solution

Common plan:

SDP: solve semidefinite programming problem

Decompose solution X with Cholesky decomposition

Round the solution by introducing random hyperplane

Get the approximate Max Cut solution

The quality of approximation will be at least 0.878 of optimum.

$$OPT = \max_{x_i^2=1} x^T L x \leq \max_{\substack{X \succeq 0 \\ diag(X)=1_n}} (LX) = SDP$$

Theoretical results

$$OPT = \max_{x_i^2=1} x^T L x = \max_{\substack{x_i^2=1 \\ X=x^T x}} LX = \max_{\substack{X \succeq 0 \\ diag(X)=1_n \\ rank(X)=1}} (LX)$$

We find the dual for OPT problem. (all the proofs can be found here: Paper)

$$Dual = \min_{\substack{\xi: \\ Diag(\xi) \succeq L}} \sum_{i=1}^{n} \xi_i$$

Theoretical results

Lemma

$$Dual = \min_{\substack{\xi:\\ Diag(\xi) \succeq L}} \sum_{i=1}^{n} \xi_i = \min_{L_T \succeq L} \max_{x: x_i^2 = 1} x^T L_T x = TreeRel$$

where L_T can be represented as $L_T = L_{tree} + Diagonal$, where L_{tree} corresponds to Laplacian of a tree graph and Diag is a diagonal matrix with non-negative values.

Tree-width approach

$$H_k = \min_{\substack{T : T = T^\top \succeq A \\ \mathrm{tw}(T) \leq k}} \max x^\top T x, \qquad OPT = H_k \leq \cdots \leq H_1 = Dual$$

where optimization is taken over all graph with tree-width less than k.

k-diagonal approach

$$D_k := \min_{\substack{T:\ T=T^\top \succeq L \\ T\ is \leq 2k+1-diagonal}} \max_{x_i^2=1} x^\top T x$$

Then,

$$Dual = D_1 \geq D_2 \geq D_3 \geq \ldots \geq D_n = OPT$$

It will be possible to find the values of D_k if we manage to implement oracul, which will calculate

$$\max_{x_i^2=1} x^\top T x$$

in polynomial time. Then we can use gradient-free methods of optimization in order to find D_k.

Oracul

We note that for fixed k if T is (2 k + 1)-diagonal matrix

$$\max_{x_i^2=1} x^\top T x$$

can be solved in O(n) time by dynamic programming. It is possible to calculate a two-dimensional array dp, where for $1 \leq i \leq n$, $0 \leq mask \leq 2^{k-1} - 1$

$$dp[i][mask] = \max_{x_{i-k+1},\ldots x_{i-1},x_i = mask} x^\top T_i x,$$

where 1) T_i, is $n \times n$ matrix which is made out of matrix T by setting all the values outside top left $k \times k$ matrix to zeros. 2) It is known, how this and the previous k - 1 vertexes are distributed between parts of cut: ones in *mask* correspond to the vertexes from one part of the cut, and zeros in *mask* correspond to the vertexes from another one.

k-diagonal algorithm

$dp[i][b_1, ..., b_n]$ can be easily recalculated by $dp[i][0, b_1, ..., b_{n-1}]$, $dp[i][1, b_1, ..., b_{n-1}]$ and $L[i-k][i], L[i-k+1][i], ..., L[i][i]$
It means, that we can find the optimal value of D_k using gradient-free methods of optimization and solving $\max_{x_i^2=1} x^\top Tx$ with oracle, which uses described dynamic programming and works for $O(n2^k)$ time.
Finally, we can restore the final cut corresponding as the cut we get from this optimization.

Computational results

1. g05-n.i For each dimension unweighted graphs with edge probability 0.5. n=60,80,100.
2. pw-100.i For each density graphs with integer edge weights chosen from [0,10] and density d=0.1,0.5,0.9, n=100.

Test	*Dual*	$\mathbf{D_1}$	$\mathbf{D_3}$	$\mathbf{D_5}$	$\mathbf{D_7}$	$\mathbf{D_{19}}$	*SDP*
g05-60	0.8951	0.8971	0.902	0.9055	0.9042	0.9187	0.9731
g05-80	0.9042	0.9033	0.9009	0.9079	0.9107	0.9197	0.9743
g05-100	0.9062	0.906	0.9107	0.9102	0.9152	0.923	0.9772
pw01-100	0.7706	0.7658	0.7772	0.7788	0.7767	0.7828	0.9525
pw05-100	0.8919	0.89	0.8947	0.8969	0.8964	0.9036	0.9736
pw09-100	0.9359	0.9378	0.9396	0.943	0.9441	0.9541	0.9839

Таблица: Results comparision

Future plans

$$H_k = \min_{\substack{T:T=T^\top \succeq A \\ \mathrm{tw}(T)\le k}} \max x^\top T x, \qquad OPT = H_k \le \cdots \le H_1 = Dual$$

Implement the similar oracul for graphs with bounded tree-width.

Try different heuristics, for example: Having T, optimal approximation with $\mathrm{tw}(T) = 1$ (tree), incrementally add edges of the graph with the largest weight to T keeping $\mathrm{tw}(T) \le k$.

Literature review

[1] 0.878-approximation for the Max-Cut problem by Divya Padmanabhanx'
[2] Michel X. Goemans and David P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. Journal of the ACM, 42(6):1115–1145, 1995
[3] Semidefinite relaxation and nonconvex quadratic optimization by Yury Nesterov, 1997
[4] Convex Optimization, Lieven Vandenberghe, Stephen Boyd, Stanford University