SDP and its applications in optimization

Konin Georgiy
Optimization Class Project. MIPT

Introduction

Semidefinite programming (*SDP*) is the most exciting development in mathematical programming in the 1990's. *SDP* has applications in such diverse fields as traditional convex constrained optimization, control theory, and combinatorial optimization.

Semidefinite Programming

Let $X \in S^n$.

def. $C(X)$ is a a linear function of X if

$$C(X) = C \bullet X = \sum_{i=1}^{n}\sum_{j=1}^{n} C_{ij}X_{ij}$$

If X is a symmetric matrix, there is no loss of generality in assuming that the matrix C is also symmetric.

A semidefinite program (SDP) is an optimization problem of the form:

$$SDP: \ C \bullet X \to min$$

$$s.t. \ A_i \bullet X = b_i, \ i = 1, ..., m; \ X \succeq 0 \ (X \in S_+^n),$$

$C, A_1, \ldots, A_m$ is the symmetric matrices, b is the m-vector.

It is easy to see that a linear program LP is a special instance of an SDP.

0.5-Approximation Algorithm for MAXCUT

For any vertex, we toss a coin in order to decide which part of the section to attribute this vertex to. It is expected that half of the edges are cutting. This algorithm can be randomized using the conditional probability method. Thus, there is a simple deterministic polynomial-time algorithm with $\frac{1}{2}$-approximation. One such algorithm starts with an arbitrary partitioning of the vertices of a given graph $G = (N, E)$ and moves one vertex in one step from one part of the cut to another, improving the solution at each step as long as improvement is possible. The number of iterations of the algorithm does not exceed$|E|$, because the algorithm improves the cut by at least one edge. When the algorithm stops working, at least half of the edges incident to any vertex belong to the cut, otherwise transferring the vertex would improve the cut (increase the size of the cut). Thus, the incision includes at least $\frac{|E|}{2}$ edges.

Interior-point Methods for SDP

At the heart of an interior-point method is a barrier function that exerts a repelling force from the boundary of the feasible region. For SDP, we need a barrier function whose values approach $+\infty$ as points X approach the boundary of the semidefinite cone S_+^n.

A natural barrier function to use to repel X from the boundary of S_+^n then is

$$-\sum_{j=1}^{n} \ln(\lambda_j(X)) = -\ln(\prod_{j=1}^{n} \lambda_j(X)) = -\ln(det(X)),$$

$\lambda_j(X)$ is the i-th eigenvalue of the X. Consider the logarithmic barrier problem $BSDP(\theta)$ parameterized by the positive barrier parameter θ:

$$BSDP(\Theta): \ C \bullet X - \Theta \ln(det(X)) \to min$$

$$s.t. \ A_i \bullet X = b_i, \ i = 1, ..., m; \ X \succ 0$$

having considered the logarithmic barrier problem for SDP(BSDP), and having compiled the Karush-Kuhn-Tucker conditions, we obtain the following algorithm.

Algorithm

Internal point search algorithm:

Step 0 . Initialization. Data is $(X^0, y^0, S^0, \theta^0)$. $k = 0$. Assume that (X^0, y^0, S^0) is a β-approximate solution of $BSDP(\theta^0)$ for some known value of β that satisfies $\beta < 1$.

Step 1. Set Current values. $(\bar{X}, \bar{y}, \bar{S}) = (X^k, y^k, S^k)$, $\theta = \theta^k$.

Step 2. Shrink θ. Set $\theta' = \alpha\theta$ for some $\alpha \in (0, 1)$. In fact, it will be appropriate to set

$$\alpha = 1 - \frac{\sqrt{\beta} - \beta}{\sqrt{\beta} + \sqrt{n}}$$

Step 3. Compute Newton Direction and Multipliers. Compute the Newton step D' for $BSDP(\theta')$ at $X = \bar{X}$ by factoring $\bar{X} = \bar{L}\bar{L}^T$ and solving the following system of equations in the variables (D, y):

$$\begin{cases} C - \theta'\bar{X}^{-1} + \theta'\bar{X}^{-1}D\bar{X}^{-1} = \sum\limits_{i=1}^{m} y_i A_i \\ \\ A_i \bullet D = 0, \quad i = 1, \ldots, m. \end{cases} \tag{7}$$

Denote the solution to this system by (D', y').

Step 4. Update All Values.

$$X' = \bar{X} + D'$$

$$S' = C - \sum_{i=1}^{m} y_i' A_i$$

Step 5. Reset Counter and Continue. $(X^{k+1}, y^{k+1}, S^{k+1}) = (X', y', S')$. $\theta^{k+1} = \theta'$. $k \leftarrow k + 1$. Go to Step 1.

Some of the unresolved issues regarding this algorithm include:

- how to set the fractional decrease parameter α
- the derivation of the Newton step D' and the multipliers y'
- whether or not successive iterative values (X^k, y^k, S^k) are β-approximate solutions to $BSDP(\theta^k)$, and
- how to get the method started in the first place.

An SDP Relaxation of the MAXCUT Problem

Let G be an undirected graph with nodes $N = 1, \ldots, n$, and edge set E. Let $w_{ij} = w_{ji}$ be the weight on edge (i, j), for $(i, j) \in E$.We assume that $w_{ij} \geq 0 \ \forall (i, j) \in E$. The MAX CUT problem is to determine a subset S of the nodes N for which the sum of the weights of the edges that cross from S to its complement $\overline{S}$ is maximized (where $\overline{S} := N \backslash S$).

Let's write the MAXCUT problem in matrices

$$MAXCUT: \ \frac{1}{4}(\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij} - W \bullet Y) \to max_{Y,x}$$

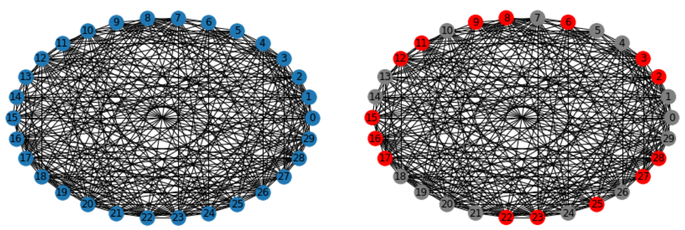
$$s.t. \ x_j \in \{-1, 1\}, \ j = 1, ..., n; \ Y = xx^T$$

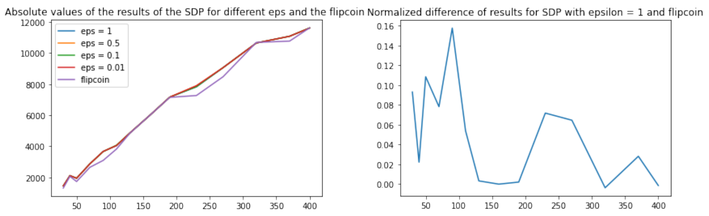
It is obvious that the matrix $Y = xx^T$ is symmetric positive definite of rank 1. By relaxing this condition, we obtain the MAXCUT relaxation, which is an SDP problem

$$RELAX: \ \frac{1}{4}(\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij} - W \bullet Y) \to max_{Y}$$

$$s.t. \ Y_{jj} = 1, \ j = 1, ..., n; \ Y \succeq 0$$

Analysis of the results





Conclusion

- For the SDP, the truth of the main inequality was checked. It is being executed.
- For the SDP, we looked at the results of work at different eps and chose (for reasons of the speed of the algorithm and the small difference in the result) the desired value for comparative analysis.
- We visually compared the results of the work of two algorithms on the same dataset of sparse and non-sparse graphs.
- The results of the two algorithms were compared numerically. It turned out that SDP copes with the task better by 1-16
- It is worth noting that the SDP processed the dataset about 5 times longer than the flipcoin, which is a fee for quality:)