Large-Scale Optimization and Applications
Course Project
Combinatorial optimization: Max-Cut, Min Uncut and Sparsest Cut Problems

M. Danilova, N. Puchkin, A. Shilova

28 March 2017

1 Combinatorial optimization problems

We consider three types of combinatorial optimization problems: Max-Cut, Min UnCut and uniform Sparsest Cut. Detailed formulation of these problems is given below in corresponding subsections. Now we only mention, that all these problems are NP-hard and cannot be solved efficiently if $P \neq NP$.

1.1 Max-Cut problem

The maximal cut problem or Max-Cut can be formulated as follows: given an undirected weighted graph $G = (V, E, W)$, where $V = \{1, \dots, n\}$ is a set of vertices, $E \subseteq V \times V$ is a set of edges and W is a matrix of weights, one wants to find a partition $(S, V \backslash S)$ maximizing the sum of weights of edges in the cut. This problem can be formulated as a combinatorial optimization

$$\begin{cases} \frac{1}{2} \sum\limits_{(i,j) \in E} w_{ij}(1 - x_i x_j) \longrightarrow \max \\ x_i \in \{-1, 1\}, \quad i = \overline{1, n} \end{cases} \tag{1}$$

This problem is indeed Max-Cut, since the partition can be defined by the formula $S = \{i : x_i \geq 0\}$. An edge (i, j) belongs to cut if and only if $x_i x_j = -1$. The Max-Cut problem 1 can be rewritten as follows:

$$\begin{cases} \frac{1}{4}\vec{1}^T W \vec{1} - \frac{1}{4} x^T W x \longrightarrow \max \\ x \in \{-1, 1\}^n, \end{cases} \tag{2}$$

where $\vec{1}$ stands for the vector of ones.

1.2 Min UnCut problem

The minimal uncut problem is a complement of the Max-Cut problem and can be formulated as follows. Given an undirected weighted graph $G = (V, E, W)$, where

$V = \{1, \dots, n\}$ is a set of vertices, $E \subseteq V \times V$ is a set of edges and W is a matrix of weights, one wants to find a partition $(S, V\backslash S)$ minimizing the sum of weights of edges in S and $V\backslash S$. In terms of optimization this problem can be formulated as follows:

$$\begin{cases} \frac{1}{4}\vec{1}^T W \vec{1} + \frac{1}{4} x^T W x \longrightarrow \min \\ x \in \{-1, 1\}^n \end{cases} \tag{3}$$

Indeed, the sum of all edges in the graph G is given by the formula $\frac{1}{2}\vec{1}^T W \vec{1}$. Subtracting the sum of weights of edges in the cut $\frac{1}{4}\vec{1}^T W \vec{1} - \frac{1}{4} x^T W x$, one arrives at the formulation 3.

1.3 Sparsest Cut problem

Finally, we give a formulation of the uniform Sparsest Cut problem. Let $G = (V, E)$, $|V| = n$ stand for a undirected graph and let $S \subseteq V$. Denote $E(S)$ a set of edges in G belonging to the cut $(S, V\backslash S)$. Then the sparsest cut problem can be formulated as follows:

$$\frac{|E(S)|}{|S| \cdot |V\backslash S|} \to \min_S \tag{4}$$

Sparsest Cut is related to the following problem. Denote $\varphi(S) = \frac{|E(S)|}{|S|}$. The number $\varphi(S)$ is called a sparsity of the cut S. The problem of sparsity minimization is formulated as follows:

$$\varphi(S) = \frac{|E(S)|}{|S|} \longrightarrow \min_{S : |S| \leq \frac{n}{2}} \tag{5}$$

Since for every cut S, $|S| \leq \frac{n}{2}$, it holds $\frac{n}{2} \leq |V\backslash S| \leq n$, then obviously

$$\frac{\varphi(S)}{n} \leq \frac{|E(S)|}{|S| \cdot |V\backslash S|} \leq \frac{2\varphi(S)}{n} \tag{6}$$

and the problem 5 can be considered as an upper-bound of the problem 4 up to a factor $\frac{2}{n}$.

Modify the problem 4. Consider $x \in -1, 1^n$ and define $S = \{i : x_i > 0\}$. Then the number of edges in the cut $|E(S)|$ can be found according to the formula

$$|E(S)| = \frac{1}{4}\vec{1}^T A \vec{1} - \frac{1}{4} x^T A x,$$

where A is an adjacency matrix of the graph G. Now we find expressions for $|S|$ and $|V\backslash S|$. Note, that

$$\sum_{i=1}^{n} x_i = |S| - |V\backslash S|$$

Then $|S|$ and $|V\backslash S|$ can be found from the system of equations

$$\begin{cases} |S| - |V\backslash S| = \sum\limits_{i=1}^{n} x_i \\ |S| + |V\backslash S| = n \end{cases}$$

Solving this system, one obtains

$$|S| = \frac{1}{2}\left(n + \sum_{i=1}^{n} x_i\right) \tag{7}$$

$$|V \backslash S| = \frac{1}{2}\left(n - \sum_{i=1}^{n} x_i\right) \tag{8}$$

Then the problem 4 can be formulated as follows:

$$\begin{cases} \frac{\vec{1}^T A \vec{1} - x^T A x}{n^2 - \left(\sum\limits_{i=1}^{n} x_i\right)^2} \longrightarrow \min \\ x \in \{-1, 1\}^n \end{cases} \tag{9}$$

2 Greedy algorithm

In this section we provide a naive greedy algorithm of combinatorial optimization. Advantage of this method is simple implementation. The pseudocode is given below

Algorithm 1 Naive greedy algorithm

Require: $F(\cdot)$ – a functional to minimize
Ensure: x_{greedy}, $F(x_{greedy})$ – optimal values

```
Initialize x randomly
while True do
    Get a neighbourhood 𝒩(x) of the point x
    Find y* = arg min_{y∈𝒩(x)\{x}} F(y)
    if F(y*) ≤ F(x) then
        x ← y*
    else
        break
return x, F(x)
```

Objectives $F(\cdot)$ for Max-Cut, Min UnCut and Sparsest Cut can be taken from problems 2, 3 and 9 respectively with addition a factor -1, if necessary. It remains to describe, how we choose a neighbourhood $\mathcal{N}(x)$ of a point x. For arbitrary $x \in \{-1, 1\}^n$ define $\mathcal{N}(x) = \{y : \|y - x\|_1 \leq 2\}$. In other words, we call y a neighbour of x, if it differs from x no more, than in one position. Since initial point is generated randomly, one can run the greedy algorithm several times and choose the best value of the objective.

It is easy to observe, that in case of Max-Cut the greedy algorithm on each step chooses a vertex v, such that the number of edges from v in the cut is less, than a number of edges from v outside the cut. Similarly in case of Min UnCut the greedy algorithm on each step chooses a vertex v, such that the number of edges from v outside the cut is less, than a number of edges from v in the cut.

3 LP relaxations

3.1 LP relaxation of Max - Cut problem

We first design an integer program for MAXCUT and then relax it to get a LP.

Integer Program Version Define variables $x_u, u \in V$ and $e_{uv}, (u,v) \in E$ as follows which are supposed to imply the following.

$$e_{uv} = \begin{cases} 1 & \text{if } (u,v) \text{ is in cut} \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

$$x_u = \begin{cases} 1 & \text{if } u \in S \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

MAXCUT can now be phrased as the following integer program.

$$\begin{cases} \sum\limits_{(u,v)\in E} e_{uv} \longrightarrow \max \\ e_{uv} \leq \begin{cases} x_u + x_v \\ 2 - (x_u + x_v) \end{cases} \quad \forall (u,v) \in E \\ x_u \in \{0,1\} \quad \forall u \in V \\ e_{uv} \in \{0,1\} \quad \forall (u,v) \in E \end{cases} \tag{12}$$

Notice that $e_{uv} \neq 0 \iff x_u \neq x_v$.

We now relax $e_{uv} \in \{0,1\}$ to $0 \leq e_{uv} \leq 1$ and $x_{uv} \in \{0,1\}$ to $0 \leq x_{uv} \leq 1$ to obtain the following LP relaxation.

$$\begin{cases} \sum\limits_{(u,v)\in E} e_{uv} \longrightarrow \max \\ e_{uv} \leq \begin{cases} x_u + x_v \\ 2 - (x_u + x_v) \end{cases} \quad \forall (u,v) \in E \\ x_u \in [0,1] \quad \forall u \in V \\ e_{uv} \in [0,1] \quad \forall (u,v) \in E \end{cases} \tag{13}$$

We can now solve the above LP using an LP-solver. Every solution to the Integer Program is also a solution to the Linear Program.

LP relaxation for MaxCut:

$$\begin{cases} \frac{1}{4}\vec{1}^T W \vec{1} - \frac{1}{4} x^T W x \longrightarrow \max \\ e_{uv} \leq \begin{cases} x_u + x_v \\ 2 - (x_u + x_v) \end{cases} \quad \forall (u,v) \in E \\ x_u \in [0,1] \quad \forall u \in V \\ e_{uv} \in [0,1] \quad \forall (u,v) \in E \end{cases} \tag{14}$$

3.2 LP relaxation of Min-UnCut problem

For Min-UnCut SDP relaxation we used the idea described in [6] and section 3.2.

Min UnCut is equivalent to the problem of finding a minimum symmetric cut ($S, \bar{S} = -S$) in G.

LP relaxation for Min-UnCut:

$$\begin{cases} \frac{1}{8}\vec{1}^T\tilde{W}\vec{1} + \frac{1}{8}x^T\tilde{W}x \longrightarrow \min \\ e_{uv} \geq \begin{cases} x_u + x_v \\ 2 - (x_u + x_v) \end{cases} \quad \forall (u,v) \in E \\ x_u \in [0,1] \quad \forall u \in V \end{cases} \tag{15}$$

where

$$\tilde{W} = \begin{pmatrix} O & W \\ W & O \end{pmatrix}$$

3.3 LP relaxation for Sparsest Cut:

The sparsest cut problem seeks to find

$$\Phi^* = \min_{S \subseteq V} \frac{\bar{w}\ \bar{\delta}_S}{\bar{D}\ \bar{\delta}_S} \tag{16}$$

where $\delta_{S_{ij}}$ - cut metric

$$\delta_{S_{ij}} = \begin{cases} 0 & \text{if } i,j \in S \text{ or } i,j \in \bar{S} \\ 1 & \text{otherwise} \end{cases} \tag{17}$$

non-negative demands D_{ij} between every pair of vertices (in our case $D_{ij} = 1$)

non-negative edge costs (or capacities) w_{ij} between every pair of vertices

To form the LP relaxation, we relax the requirement of minimizing over the cut metrics to minimizing over all metrics. That is,

$$\lambda^* = \min_{\text{metrics d}} \frac{\bar{w}\ \bar{d}}{\bar{D}\ \bar{d}} \tag{18}$$

The above relaxation is solved by the following linear program:

$$\begin{cases} \sum\limits_{i,j} w_{ij}d_{ij} \longrightarrow \min \\ \sum\limits_{i,j} D_{ij}d_{ij} = 1 \\ d_{ij} + d_{jk} \geq d_{ik} \quad \forall i,j,k \\ d_{ij} \geq 0 \quad \forall i,j \end{cases} \tag{19}$$

4 SDP relaxations

4.1 SDP relaxation of Max-Cut problem

As it was already written earlier the Max-Cut can be formulated as the problem 2. In such formulation it is not convex as x can only be equal to -1 or to 1. Let's notice

that $\forall x \hookrightarrow x^T W x = \mathrm{Tr}(W x x^T)$, therefore if we introduce $X = xx^T$ and $\mathbb{I}$ a matrix containing only ones as elements, then the whole optimisation problem can be rewritten in the following way

$$\begin{cases} \frac{1}{4}\left(\mathrm{Tr}(W\mathbb{I}) - \mathrm{Tr}(WX)\right) \longrightarrow \max, \\ \mathrm{diag}(X) = 1, \\ X \succeq 0, \\ \mathrm{rank}(X) = 1, \\ X_{i,j} \in \{-1, 1\} \qquad\qquad \forall i, j. \end{cases}$$

It is still not a convex problem, but we will obtain one if the last two constraints are thrown away. Hence, the SDP relaxation of Max-Cut looks like

$$\begin{cases} \frac{1}{4}\left(\mathrm{Tr}(W\mathbb{I}) - \mathrm{Tr}(WX)\right) \longrightarrow \max \\ \mathrm{diag}(X) = 1, \\ X \succeq 0. \end{cases} \tag{20}$$

4.2 Min UnCut relaxation

For Min-UnCut SDP relaxation we used the idea described in [6]. In the lecture it was suggested to use new graph $G'(V', E')$. In order to construct it, new vertices $\{-1, \ldots, -n\}$ should be added to the initial graph. We connect two vertices i and j with an edge in G' if and only if i and $-j$ or $-i$ and j are connected with an edge in G. Then the initial problem of finding the cut such that the sum of weights of edges not in the cut was minimal is equivalent to the problem of finding the minimal symmetric cut in G', i.e. $(S', T' = -S')$ and $S' \cup T' = V'$, where $S' \equiv \{-i |\ i \in S'\}$. That is true if every cut (S, T) in the graph G corresponds to the cut $(S \cup (-T), (-S) \cup T)$ in G' and on the other hand if we have a cut (S', T') in G', then one can easily get the cut in G, where $S = \{i |\ i \geq 0,\ i \in S'\}$ and $T = \{i |\ i \geq 0,\ i \in T'\}$. We can exploit it and then our problem is equivalent to

$$\begin{cases} \frac{1}{8}\left(\mathrm{Tr}(\tilde{W}\mathbb{I}) - \mathrm{Tr}(\tilde{W}X)\right) \longrightarrow \min, \\ \mathrm{diag}(X) = 1, \\ X_{i,-i} = -1, \\ X \succeq 0, \\ \mathrm{rank}(X) = 1, \\ X_{i,j} \in \{-1, 1\} \qquad\qquad \forall i, j. \end{cases}$$

Here we took into account the fact that in the extended graph the number of edges doubles, therefore the coefficient $\frac{1}{8}$ is present. X_{ij} has the similar meaning as for Max-Cut problem, but $X_{ij} = -1$ if and only if vertices i and j are separated by the cut in G'. Moreover, in this statement

$$\tilde{W} = \begin{pmatrix} O & W \\ W & O \end{pmatrix}$$

where W is the matrix of weights for G. At last, the third constraint binds our cut to be symmetric or to satisfy $(S', T' = -S')$.

As in the previous case two last constraints prevent this problem from being convex, hence, getting rid of them, we will have the SDP relaxation for Min-UnCut problem.

$$\begin{cases} \frac{1}{8}\left(\mathrm{Tr}(\tilde{W}\mathbb{I}) - \mathrm{Tr}(\tilde{W}X)\right) \longrightarrow \min, \\ \mathrm{diag}(X) = 1, \\ X_{i,-i} = -1, \\ X \succeq 0. \end{cases} \tag{21}$$

4.3 Sparsest Cut relaxation

Basing on (9), we can get the relaxed problem by properly dealeing with the denominator. We want to find t, such that $t\left\{n^2 - \left(\sum_{i=1}^{n} x_i\right)^2\right\} = 1$. Then introducing as in the previous cases the matrix X with elements $X_{ij} = x_i x_j$ and also a new vector $y = \sqrt{t}x$ and $Y = \{Y_{ij}\}_{i,j=1}^{n}$, where $Y_{ij} = y_i y_j = t x_i x_j = t X_{ij}$, one can see that the problem of finding the sparsest cut is equivalent to the following

$$\begin{cases} t\sum\limits_{i,j=1}^{n} w_{ij} - \mathrm{Tr}(WY) \longrightarrow \min, \\ tn^2 - \sum\limits_{i,j=1}^{n} Y_{ij} = 1, \\ \mathrm{diag}(Y) = t, \\ t > 0, \\ \mathrm{rank}(Y), \\ Y_{ij} \in \{\sqrt{t}, -\sqrt{t}\} \qquad\qquad \forall i, j. \end{cases}$$

Here, as in the cases listed above two last constraints aren't suitable for SDP programming, therefore they should be thrown away. Furthermore, the first constraint can be replaced by $t\left\{n^2 - \sum\limits_{i,j=1}^{n} X_{ij}\right\} \geq 1$ equivalently. Finally, in order to make our relaxation be narrower, one may notice that as the number $\left\{n^2 - \left(\sum\limits_{i=1}^{n} x_i\right)^2\right\}$ corresponds to the value $|S||V\backslash S|$. There are known upper and lower bounds for this value. The upper bound is $\frac{n^2}{4}$ if n is an even number and $\frac{n^2-1}{4}$ if n is odd. The lower bound is $n-1$. Taking into account all the mentioned, we obtain the SDP relaxation for our problem:

$$\begin{cases} t\sum\limits_{i,j=1}^{n} w_{ij} - \mathrm{Tr}(WY) \longrightarrow \min, \\ tn^2 - \sum\limits_{i,j=1}^{n} Y_{ij} \geq 1, \\ \mathrm{diag}(Y) = t, \\ \dfrac{1}{4(n-1)} \geq t \geq \dfrac{1}{n^2}. \end{cases} \tag{22}$$

4.4 Gradient descent

Consider a SDP problem

$$\begin{cases} \mathrm{Tr}\left(W^\top X\right) \longrightarrow \min \\ \mathrm{diag}\,(X) = 1 \\ X \succeq 0, \end{cases} \tag{23}$$

Let's reformulate this problem

$$\begin{cases} \mathrm{Tr}\left(W^\top X\right) - \mu \log \det X \longrightarrow \min \\ \mathrm{diag}\,(X) = 1, \end{cases} \tag{24}$$

Gradient

$$df = \mathrm{Tr}\left[(W - \mu X^{-1})\delta X\right]$$
$$\nabla f = (W - \mu X^{-1})$$

Step

$$\Delta X = \pi_L\left(\alpha(W - \mu X^{-1})\right) = \alpha\pi_L\left(W - \mu X^{-1}\right)$$

for MaxCut problem

$$L = \{z \mid z_{ii} = 0 \quad \forall i\}$$

for Min UnCut problem

$$L = \{z \mid z_{ii} = 0, z_{i,n+i} = 0, z_{n+i,i} = 0 \quad \forall i\}$$

4.5 Primal-dual central path tracing interior point method

Consider an SDP problem

$$\begin{cases} \langle C, X\rangle \longrightarrow \min \\ X \in \mathcal{L} - B \\ X \succeq 0, \end{cases} \tag{25}$$

where $\langle C, X\rangle = \mathrm{Tr}(C^T X)$. The dual problem can be formulated as follows

$$\begin{cases} \langle B, S\rangle \longrightarrow \max \\ S \in \mathcal{L}^\perp + C \\ S \succeq 0 \end{cases} \tag{26}$$

If problems 25 and 26 are strictly feasible, then the strong duality takes place. We describe an interior point method based on primal-dual central path tracing. Consider a pair of modified primal and dual problems

$$X_\mu = \arg\min_{X\in\mathcal{L}} \langle C, X\rangle - \mu \log\det X \tag{27}$$

$$S_\mu = \arg\max_{S\in\mathcal{L}^\perp} \langle B, X\rangle + \mu \log\det S \tag{28}$$

It is known, that $X_\mu S_\mu = S_\mu X_\mu = I$, and duality gap will be equal to $\langle X_\mu, S_\mu\rangle = \mu n$. The set $\{(X_\mu, S_\mu) : \mu > 0\}$ defines a so-called primal-dual central path. If one can move along

the primal-dual central path, then one can easily solve optimization problems 25, 26 with arbitrary predefined accuracy tending μ to zero. Unfortunately, we cannot stay on the primal-dual central path, but we can try to move in its vicinity. The following algorithm allows to trace the primal-dual central path.

Algorithm 2 Primal-dual central path tracing IPM

Require: Matrices B and C as in problems 25, 26; projectors $\pi_{\mathcal{L}}(\cdot, Q)$, $\pi_{\mathcal{L}^\perp}(\cdot, Q)$ on linear spaces $Q\mathcal{L}Q$ and $Q^{-1}\mathcal{L}^\perp Q^{-1}$; parameters $0 \leq \chi \leq \kappa \leq 0.1$; initial values $X \in \mathcal{L} - B$, $S \in \mathcal{L}^\perp + C$, $X, S \succ 0$ and t, which fulfil

$$\|tX^{\frac{1}{2}}SX^{\frac{1}{2}} - I\|_2 \leq \kappa;$$

tolerance ε and (or) number of iterations $N_{\max}$

Ensure: ε-optimal solution X

1: Compute duality gap $\langle X, S\rangle$
2: **while** (Duality gap $> \varepsilon$) or (number of iterations $< N_{\max}$) **do**
3: $t \leftarrow \left(1 - \frac{\chi}{\sqrt{k}}\right)^{-1} t$
4: Choose a Nesterov-Todd scaling matrix

$$Q = P^{\frac{1}{2}} \tag{29}$$

$$P = X^{-\frac{1}{2}}\left(X^{-\frac{1}{2}}S^{-1}X^{-\frac{1}{2}}\right)^{\frac{1}{2}}X^{\frac{1}{2}}S \tag{30}$$

5: $\hat{X} \leftarrow QXQ$
6: Solve a Lyapunov matrix equation

$$\hat{X}Z + Z\hat{X} = \frac{2}{t}I - 2\hat{X}^2 \tag{31}$$

7: $\Delta\hat{X} \leftarrow \pi_{\mathcal{L}}(Z, Q)$, $\Delta\hat{S} \leftarrow \pi_{\mathcal{L}^\perp}(Z, Q)$
8: $\Delta X \leftarrow Q^{-1}\Delta\hat{X}Q^{-1}$, $\Delta S \leftarrow Q\Delta\hat{S}Q$
9: Update $X \leftarrow X + \Delta X$, $S \leftarrow S + \Delta S$, $t \leftarrow \frac{k}{\langle X,S\rangle}$
10: Compute duality gap $\langle X, S\rangle = \frac{k}{t}$
11: **return** X

The Nesterov-Todd scaling matrix defined by 29 – 30 ensures, that matrices $\hat{X} = QXQ$ and $\hat{S} = Q^{-1}SQ^{-1}$ commute and, moreover, $\hat{X} = \hat{S}$. The equation 31 can be considered as a linearization of the equation

$$\tilde{X}\tilde{S} + \tilde{S}\tilde{X} = \frac{2}{t}I$$

at the point $(\hat{X}, \hat{S}) = (QXQ, Q^{-1}SQ^{-1})$ w.r.t. X and S, and $Z = Q\Delta XQ + Q^{-1}\Delta SQ^{-1}$. Note, that the algorithm guarantees, that $\Delta X \in \mathcal{L}$ and $\Delta S \in \mathcal{L}^\perp$, so the output X is feasible. We refer to lectures of Ben-Tal and Nemirovski [2] for more details.

The next theorem guarantees consistency of the procedure 2.

Theorem 1 (4.5.2 in [2]) *Let the parameters χ, κ of 2 satisfy*

$$0 < \chi \leq \kappa \leq 0.1$$

Let (X, S) be a pair of strictly feasible primal and dual solutions to 25, 26, such that the triple (X, S, t) satisfies

$$\|tX^{\frac{1}{2}}SX^{\frac{1}{2}} - I\|_2 \leq \kappa; \tag{32}$$

Then the updated pair $(X + \Delta X, S + \Delta S)$ is well-defined (i. e. equation 31 has a unique solution), $X + \Delta X$, $S + \Delta S$ are strictly feasible solutions to 25, 26 respectively, and the triple $\left(X + \Delta X, S + \Delta S, t = \frac{k}{\langle X+\Delta X, S+\Delta S\rangle}\right)$ satisfies 32.

4.6 Relaxations with addition of triangle constraints

If we introduce the metric for our partitioning tasks:

$$d(i,j) = \begin{cases} 0 & \text{if } i \text{ and } j \text{ are vertices in the same part of the graph,} \\ 1 & \text{otherwise,} \end{cases}$$

which satisfies all the metric conditions, then the variables in the problems mentioned above are connected with the metric in the following way

$$X_{ij} = 1 - 2d(i,j).$$

Taking into account this connection, one can obtain the triangle constraints with our variables, writing those for the metric itself:

$$d(i,j) + d(j,k) \geq d(i,k) \Leftrightarrow X_{ij} + X_{jk} - 1 \leq X_{ik}.$$

Those expressions should be true for any i, j and k. Let's denote for any j $E_j = (\mathbf{e}_j, \ldots, \mathbf{e}_j)$ and $\mathbf{e}_j$ is a unit vector with 1 on the j-th place. From this follows that the constraint above can be rewritten as following:

$$XE_j + E_j^T X - 1 \leq X \qquad \forall j.$$

Beside all the mentioned above, one could notice that for the metric $d(i,j)$ it also holds the fact that

$$d(i,j) + d(j,k) + d(i,k) \leq 2 \qquad \forall i, j, k.$$

Therefore, applying the link between the metric and X, we have

$$X_{ij} + X_{jk} + X_{ik} \geq -1 \qquad \forall i, j, k.$$

Rewriting it in the matrix way, we will derive the last triangle constraint:

$$XE_j + E_j^T X + X \geq -1 \qquad \forall j.$$

As for the SDP relaxation of the sparsest cut problem, in our problem statement there are variables Y and t and Y is connected with X by $Y = tX$ expression. Hence, the formulas presented above should be rewritten for the sparsest cut problem in the following way

$$\begin{aligned} YE_j + E_j^T Y + Y &\geq -t & \forall j \\ YE_j + E_j^T Y - t &\leq Y & \forall j \end{aligned}$$

For the sparsest cut we used the relaxation for unit ℓ_2-representation of graph proposed in [1]

$$\begin{cases} \frac{1}{n}\sum\limits_{(i,j)\in E} ||v_i - v_j||^2 \longrightarrow \min \\ \|v_i\|^2 = 1 \\ \sum\limits_{i<j} \|v_i - v_j\|^2 = 1 \\ \|v_i - v_j\|^2 + \|v_j - v_k\|^2 \geq \|v_i - v_k\|^2, \quad \forall\, i,j,k \end{cases} \tag{33}$$

5 Numerical experiments and results

We conducted experiments of described methods on Erdos-Renyi random graphs. Let $n \in \mathbb{N}$ and $p \in (0,1)$. Erdos-Renyi random graph $G(n,p)$ is a graph with n vertices, where each of $\frac{n(n-1)}{2}$ edges is presented independently with probability equal to p.

For MaxCut, Min UnCut and Sparsest Cut problems a crucial point is connectivity of a graph, otherwise these problems become trivial. So, in most of our experiments we chose p fulfilling

$$p > \frac{2\log(n-1)}{n} \tag{34}$$

Condition 34 guarantees, that the graph $G(n,p)$ will be connected with high probability as n tends to infinity. We refer to [4], section 5.3.3 for details.

5.1 Theoretical guarantees for Max Cut and Min Uncut

In this section we provide theoretical guarantees for mathematical expectation of Max Cut in random graphs. Let $\text{max\_cut}(G(n,p))$ be a maximal cut of Erdos-Renyi random graph. Denote $f_{cut}(n,p) = \mathbb{E}\ \text{max\_cut}(G(n,p))$. Note, that the condition 34 implies

$$\frac{np}{2} > \log(n-1),$$

and for large n this number is also large. Thus, we don't deal with the phase transition case. Theoretical estimations of $f_{cut}(n,p)$ are given by [3].

Theorem 2 (High-density random Max-Cut [3]) *For large $\frac{np}{2}$,*

$$\left(\frac{np}{4} + \sqrt{\frac{4np}{9\pi}}\right) n \leq f_{cut}(n,p) \leq \left(\frac{np}{4} + \sqrt{\frac{np\ln 2}{4}}\right) n$$

Theorem 3 (Low-density random Max-Cut [3]) *For any fixed $\varepsilon > 0$ it holds*

$$\left(\frac{1}{2}\varepsilon - \frac{16}{3}\varepsilon^3\right) n \leq f_{cut}\left(n, \frac{1+2\varepsilon}{n}\right) \leq \left(\frac{1}{2} + \varepsilon - \Omega\left(\frac{\varepsilon^3}{\ln \varepsilon^{-1}}\right)\right) n$$

The last theorem is more suitable for the case close to the phase transition. During the experiment we computed bounds given by both theorems 2, 3 and chose the tightest one.

Finally note, that for this model bounds for Min UnCut follow immediately, since the expectation of sum of edges in cut and out of cut is equal to np.

5.2 Theoretical guarantees for Sparsest Cut

Let $G(n,p)$ be an Erdos-Renyi graph and let L stand for its Laplacian. Denote $\lambda_2 = \lambda_2(L)$ the second smallest eigenvalue of L. Define a sparsity of a graph $\phi(S) = \min\limits_{S:|S|\leq\frac{n}{2}} \frac{|E(S)|}{|S|}$. The lower bound for sparsest cut can be taken from inequality 6. The lower bound for $\varphi(S)$ is given by the Cheeger's inequality.

Theorem 4 (Cheeger's inequality)

$$\frac{\lambda_2}{2} \leq \phi(S) \leq \sqrt{2\lambda_2}$$

So, the number $\frac{1}{n}\sqrt{2\lambda_2}$ is taken for the upper bound of sparsest cut.

The next theorem, which allows to estimate λ_2, follows directly from [5].

Theorem 5 (Jiang [5])

$$\frac{\lambda_2 - np}{\sqrt{p(1-p)n\log n}} \to -\sqrt{2} \quad a.\,s.$$

$$\limsup_{n\to\infty} \frac{\lambda_2 - np}{\sqrt{p(1-p)n\log n}} = -\sqrt{2} \quad a.\,s.$$

$$-\sqrt{3} \leq \liminf_{n\to\infty} \frac{\lambda_2 - np}{\sqrt{p(1-p)n\log n}} \leq -\sqrt{\frac{8}{3}} \quad a.\,s.$$

The theorem 5 provides good guarantees for the second smallest eigenvalue of Laplacian in case of large n. Namely, combined with 4, it gives $O(1)$-approximation, $n \to \infty$, of Sparsest Cut provided p is fixed. However, in experiments we did not use these results, since n was relatively small, and all eigenvalues if Laplacian could be found fast numerically.

5.3 Numerical results

Numerical results of quality of algorithms and times of computations are provided in Appendix A. We used theoretical guarantees provided in sections 5.1, 5.2 to estimate the quality of relaxations. Namely, we took values of Max-Cut, Min UnCut and Sparsest Cut, returned by procedures, and divided them by the theoretical upper-bounds. Computations were executed on the laptop with 2.7 GHz and 8 GB RAM. All methods performed on the same set of graphs, which was generated at the beginning of tests. Relaxations with triangle constraints were restricted by 100 iterations due to computational issues. Manually implemented gradient descent and interior-point method were restricted by 1000 iterations.

We can observe, that the greedy algorithm was exremely fast, but at the same time showed the worst performance.

SDP-relaxations seems to be the best choice for the Max-Cut problem, since they showed a very good performance. However, they are computationally more costly, than LP-relaxations, which deliver some kind of trade-off between computational complexity and quality of relaxation.

For Min Uncut and Sparsest Cut algorithms performed not so well. Note, that there is a huge difference in values, returned by Sparsest Cut relaxations with and without constraints.

We also implemented gradient descent and interior-point methods. The second method gives more accurate results, but it works significantly slower.

Final Remarks

See `github.com/npuchkin/LSOptProject` for he report and implementation of described algorithms.

Work split for this project was the following: Alena implemented SDP-relaxations for Max-Cut, Min Uncut and Sparsest Cut and SDP-relaxations with triangle constraints; Marina implemented LP-relaxations and gradient descent method for SDP-relaxations of Max-Cut and Min Uncut; and finally, Nikita implemented greedy algorithm, primal-dual tracing path IPM, SDP-relaxation for sparsest cut based on ℓ_2-representation and studied theoretical results of Max-Cut, Min UnCut and Sparsest Cut.

References

[1] S. Arora, S. Rao, U. Vazirani *Expander Flows, Geometric Embeddings and Graph Partitioning*, (2004),
`https://www.cs.princeton.edu/~arora/pubs/arvfull.pdf`

[2] A. Ben-Tal, A. Nemirovski, *Lectures on Modern Convex Optimization*, (2013)
`http://www2.isye.gatech.edu/~nemirovs/Lect_ModConvOpt.pdf`

[3] D. Coppersmith et al., *Random Max Sat, Random Max Cut, and Their Phase Transitions* (2003),
`https://arxiv.org/abs/math/0306047`

[4] J. A. Tropp, *An Introduction to Matrix Concentration Inequalities*, (2014)
`https://arxiv.org/pdf/1501.01571.pdf`

[5] T. Jiang, *Low eigenvalues of Laplacian matrices of large random graphs* (2012), Probab. Theory Relat. Fields

[6] Sanjeev Arora, *Lecture 8: Approximating Min UnCut and Min-2CNF Deletion*, (2005)
`http://www.cs.princeton.edu/courses/archive/spr05/cos598B/lecture8.pdf`

[7] A. Agarwal, K. Makarychev, M. Charikar, Y. Makarychev, *Approximation Algorithms for Min UnCut, Min2CNF Deletion, and Directed Cut Problems*, (2005)
`http://ttic.uchicago.edu/~yury/papers/min2cnf.pdf`

[8] A. Gupta, *Lecture: Algorithms for Sparsest Cut*, (2008)
`http://www.cs.cmu.edu/~anupamg/adv-approx/lecture27.pdf`

[9] A. Gupta, *Lecture: Sparsest Cut and L1 Embeddings*, (2008)
`http://www.cs.cmu.edu/~anupamg/adv-approx/lecture19.pdf`

[10] P. Harsha, *Lecture: MAXCUT and Introduction to Inapproximability*, (2010)
http://www.tcs.tifr.res.in/~prahladh/teaching/2009-10

Appendix

A Quality of algorithms' performance

A.1 Quality of greedy algorithms' performance

$p\backslash n$	20	40	60	80	100
0.2	0.005184	0.001810	0.000783	0.000438	0.000291
0.4	0.006924	0.001736	0.000863	0.000466	0.000330
0.6	0.007916	0.001861	0.000864	0.000505	0.000330
0.8	0.007310	0.001943	0.000881	0.000501	0.000336

Table 1: Quality of greedy algorithm for Max-Cut

$p\backslash n$	20	40	60	80	100
0.2	0.064299	0.006877	0.002178	0.001037	0.000622
0.4	0.029885	0.004285	0.001753	0.000851	0.000562
0.6	0.025171	0.003879	0.001542	0.000826	0.000510
0.8	0.019763	0.003676	0.001458	0.000769	0.000490

Table 2: Quality of greedy algorithm for Min UnCut

$p\backslash n$	20	40	60	80	100
0.2	0.185306	0.108240	0.072286	0.056744	0.047290
0.4	0.188165	0.095604	0.083093	0.059591	0.056778
0.6	0.207449	0.120564	0.084791	0.070886	0.063973
0.8	0.164967	0.115637	0.091512	0.077024	0.069975

Table 3: Quality of greedy algorithm for Sparsest Cut

A.2 Quality of LP-relaxations' performance

$p\backslash n$	20	40	60	80	100
0.2	0.337637	0.516379	0.519232	0.508041	0.539972
0.4	0.489825	0.501423	0.550420	0.569032	0.591920
0.6	0.536125	0.545757	0.576205	0.601525	0.608043
0.8	0.529586	0.567689	0.587893	0.600959	0.622985

Table 4: Quality of Max-Cut LP-relaxation

$p\backslash n$	20	40	60	80	100
0.2	0.291439	0.172465	0.119498	0.092908	0.078675
0.4	0.261286	0.135953	0.114061	0.087785	0.078202
0.6	0.243999	0.151122	0.106579	0.088867	0.079501
0.8	0.188348	0.129494	0.103023	0.087423	0.077947

Table 5: Quality of Sparsest Cut LP-relaxation

$p\backslash n$	20	40	60	80	100
0.2	6.980001	3.270138	2.406798	2.002486	1.918795
0.4	3.523489	2.062452	1.863596	1.731282	1.679314
0.6	2.840803	1.895721	1.713624	1.640804	1.566960
0.8	2.386046	1.790277	1.621432	1.537556	1.517116

Table 6: Quality of Min UnCut LP-relaxation

A.3 Quality of SDP-relaxations' performance

$p\backslash n$	20	40	60	80	100
0.2	0.761003	0.979086	0.951926	0.928473	0.973150
0.4	0.916130	0.871677	0.919312	0.941996	0.961611
0.6	0.886489	0.866162	0.916332	0.964963	0.949561
0.8	0.824357	0.797939	0.904660	0.988222	1.006969

Table 7: Quality of Max-Cut SDP-relaxation

$p\backslash n$	20	40	60	80	100
0.2	0.802952	1.023859	0.983911	0.950392	0.975133
0.4	0.948670	0.898792	0.950017	0.945025	0.961480
0.6	0.913459	0.897152	0.915049	0.931280	0.933983
0.8	0.841636	0.866825	0.881889	0.890260	0.910257

Table 8: Quality of Max-Cut SDP-relaxation with triangle constraints

$p\backslash n$	20	40	60	80	100
0.2	1.797336	0.965480	0.634725	0.581404	0.568665
0.4	1.750255	0.271793	0.105309	0.199872	0.265164
0.6	1.546344	0.221130	-0.000755	-0.000672	-0.000405
0.8	1.302217	0.208395	-0.002666	0.015526	0.097146

Table 9: Quality of Min Uncut SDP-relaxation

$p\backslash n$	20	40	60	80	100
0.2	1.797336	0.965480	0.634725	0.581404	0.568665
0.4	1.750255	0.271793	0.105309	0.199872	0.265164
0.6	1.546344	0.221130	-0.000755	-0.000672	-0.000405
0.8	1.302217	0.208395	-0.002666	0.015526	0.097146

Table 10: Quality of Min Uncut SDP-relaxation with triangle constraints

$p\backslash n$	20	40	60	80	100
0.2	0.028236	0.028697	0.025743	0.023292	0.022666
0.4	0.061513	0.052685	0.042748	0.041511	0.037735
0.6	0.087093	0.063291	0.059973	0.054090	0.048579
0.8	0.126918	0.088076	0.072834	0.064304	0.058238

Table 11: Quality of Sparsest Cut SDP-relaxation

$p\backslash n$	20	40	60	80	100
0.2	3.103853	4.107249	4.253099	4.377912	4.668124
0.4	3.391525	3.670304	4.700984	4.839431	5.423894
0.6	3.610752	4.699704	5.016535	5.623423	6.277126
0.8	3.081788	4.491410	5.416955	6.147781	6.921715

Table 12: Quality of Sparsest Cut SDP-relaxation with triangle constraints

$p\backslash n$	20	40	60	80	100
0.2	0.316362	0.556524	0.578102	0.583320	0.630353
0.4	0.505950	0.565402	0.639184	0.670508	0.711235
0.6	0.541884	0.608817	0.667366	0.706982	0.729679
0.8	0.512161	0.618446	0.660501	0.697750	0.731807

Table 13: Quality of Max-Cut SDP-relaxation for gradient descent method

$p\backslash n$	20	40	60	80	100
0.2	6.930888	2.882311	2.067429	1.700380	1.608799
0.4	3.123833	1.753610	1.551708	1.428076	1.376428
0.6	2.442479	1.579533	1.409037	1.339802	1.274523
0.8	2.025600	1.477239	1.324676	1.249753	1.228988

Table 14: Quality of Min UnCut SDP-relaxation for gradient descent method

$p\backslash n$	20	40	60	80	100
0.2	0.803032	1.023921	0.983963	0.950437	0.975169
0.4	0.948734	0.898833	0.950075	0.945078	0.961524
0.6	0.913493	0.897195	0.915093	0.931339	0.934016
0.8	0.841681	0.866872	0.881917	0.890294	0.910294

Table 15: Quality of Max-Cut SDP-relaxation for IPM

B Times of execution

B.1 Greedy algorithms times of execution

$p\backslash n$	20	40	60	80	100
0.2	0.070913	0.083236	0.146062	0.385446	0.317414
0.4	0.057796	0.080928	0.140337	0.379452	0.446185
0.6	0.046991	0.092468	0.141134	0.294798	0.401081
0.8	0.043774	0.082179	0.151729	0.211405	0.349521

Table 16: Time of execution of greedy algorithm for Max-Cut

$p\backslash n$	20	40	60	80	100
0.2	0.070913	0.083236	0.146062	0.385446	0.317414
0.4	0.057796	0.080928	0.140337	0.379452	0.446185
0.6	0.046991	0.092468	0.141134	0.294798	0.401081
0.8	0.043774	0.082179	0.151729	0.211405	0.349521

Table 17: Time of execution of greedy algorithm for Min UnCut

$p\backslash n$	20	40	60	80	100
0.2	0.070913	0.083236	0.146062	0.385446	0.317414
0.4	0.057796	0.080928	0.140337	0.379452	0.446185
0.6	0.046991	0.092468	0.141134	0.294798	0.401081
0.8	0.043774	0.082179	0.151729	0.211405	0.349521

Table 18: Time of execution of greedy algorithm for Sparsest Cut

B.2 LP-relaxations times of execution

$p\backslash n$	20	40	60	80	100
0.2	0.013428	0.028338	0.047363	0.086294	0.148773
0.4	0.011856	0.056529	0.050932	0.125852	0.236400
0.6	0.012826	0.028367	0.071739	0.186027	0.354341
0.8	0.012313	0.032294	0.068754	0.134669	0.247148

Table 19: Time of execution of Max-Cut LP-relaxation

$p\backslash n$	20	40	60	80	100
0.2	0.029958	0.111946	0.347665	0.509658	0.765721
0.4	0.028274	0.113997	0.318581	0.606641	0.901005
0.6	0.032696	0.103831	0.280712	0.534633	1.001389
0.8	0.036837	0.108554	0.322009	0.584454	1.082593

Table 20: Time of execution of Min UnCut LP-relaxation

$p\backslash n$	20	40	60	80	100
0.2	0.117843	0.718436	2.739953	7.626631	12.672801
0.4	0.096559	0.749443	2.699008	7.180649	13.982816
0.6	0.096460	0.777362	2.926412	8.385034	14.511220
0.8	0.105442	0.776333	2.823051	8.213174	13.481095

Table 21: Time of execution of Sparsest Cut LP-relaxation

B.3 SDP-relaxations times of execution

$p\backslash n$	20	40	60	80	100
0.2	0.276537	1.874202	6.227408	15.865747	31.877150
0.4	0.238374	2.002168	5.987302	16.097594	34.618038
0.6	0.313747	1.640704	7.582998	16.550395	31.338326
0.8	0.244984	1.833730	6.129244	15.752917	37.577764

Table 22: Time of execution of Max-Cut SDP-relaxation

$p\backslash n$	20	40	60	80	100
0.2	0.034280	0.101418	0.331691	0.535867	1.057492
0.4	0.032636	0.080624	0.260028	0.684516	1.370277
0.6	0.022795	0.113248	0.288867	0.911601	1.373014
0.8	0.036301	0.148022	0.370161	0.757674	1.972998

Table 23: Time of execution of Max-Cut SDP-relaxation with triangle constraints

$p\backslash n$	20	40	60	80	100
0.2	1.823601	15.769429	59.879660	166.685619	337.450467
0.4	1.749810	15.894278	57.654459	155.694652	332.730629
0.6	2.042828	18.929642	57.812336	163.010889	347.442831
0.8	1.757870	16.502796	64.940528	160.101380	335.736556

Table 24: Time of execution of Min Uncut SDP-relaxation

$p\backslash n$	20	40	60	80	100
0.2	1.823601	15.769429	59.879660	166.685619	337.450467
0.4	1.749810	15.894278	57.654459	155.694652	332.730629
0.6	2.042828	18.929642	57.812336	163.010889	347.442831
0.8	1.757870	16.502796	64.940528	160.101380	335.736556

Table 25: Time of execution of Min Uncut SDP-relaxation with triangle constraints

$p\backslash n$	20	40	60	80	100
0.2	0.125547	0.493920	4.248190	7.880980	13.090590
0.4	0.093592	1.823017	4.367517	7.604911	13.282790
0.6	0.074742	0.339860	4.145274	7.525525	13.302072
0.8	0.087599	0.554415	3.954314	7.646360	12.731183

Table 26: Time of execution of Sparsest Cut SDP-relaxation

$p\backslash n$	20	40	60	80	100
0.2	0.156529	0.762021	3.032832	7.277937	13.798227
0.4	0.132919	0.771237	2.990910	6.944147	14.230462
0.6	0.135523	0.781156	2.912829	7.215524	15.103758
0.8	0.128842	0.819107	3.443282	7.230506	13.938156

Table 27: Time of execution of Sparsest Cut SDP-relaxation with triangle constraints

$p\backslash n$	20	40	60	80	100
0.2	0.415974	1.750458	3.857480	9.020489	15.484402
0.4	0.482390	1.643881	4.201674	9.163862	16.069376
0.6	0.441821	1.816941	3.813231	9.104874	17.500542
0.8	0.406073	1.561782	4.013681	8.934394	15.571183

Table 28: Time of execution of Max-Cut SDP-relaxation for gradient descent method

$p\backslash n$	20	40	60	80	100
0.2	1.825647	8.452745	23.353299	41.375080	78.893934
0.4	1.559878	8.446443	24.290554	38.089894	85.483747
0.6	1.501492	8.705670	25.692618	39.091669	86.943622
0.8	1.807911	7.605453	21.413884	37.643582	80.052236

Table 29: Time of execution of Min UnCut SDP-relaxation for gradient descent method

$p\backslash n$	20	40	60	80	100
0.2	7.433283	21.958996	50.476675	118.892647	219.795809
0.4	7.876525	24.343558	49.547392	119.840315	193.288305
0.6	7.009986	25.542976	49.774089	109.004840	212.017449
0.8	6.606875	24.981019	48.753915	113.483592	235.657828

Table 30: Time of execution of Max-Cut SDP-relaxation for IPM