Relaxations of the Maximum Cut Problem

Demetrios V. Papazaharias, Carter Mann, Luca Wrabetz
Department of Industrial and Systems Engineering
University at Buffalo, Bell Hall, Buffalo, New York, 14260
{dvpapaza, cjmann3, lucawrab}@buffalo.edu

December 2019

Abstract

Maximum cut is a classic NP-Hard problem in combinatorial optimization. In this work we present several relaxations for the maximum cut problem.

1 Introduction

The maximum cut problem (maxcut) is a classical problem in combinatorial optimization in which we are given a graph $G = (V, E)$, with node set $V = \{1, ..., n\}$ and edge set $E = \{i, j\} \subseteq \binom{V}{2}$. We are also given, for each edge $\{i, j\} \in E$, a non-negative weight w_{ij} . The maxcut problem consists in determining a subset of $S \subseteq V$ nodes for which the sum of the weights of the edges that cross from S to its complement $\bar{S}$ (i.e., the weight of the edges in the cut $(S, \bar{S})$ is maximized. In this particular case, we are assuming that the edges are undirected; however, this problem can be easily generalized to the directed case.

Unlike the minimization variant to this problem, which can be solved in polynomial time, the maxcut problem is NP-Hard. Given its nature, the maxcut problem is a highly studied problem in Graph Theory. Although no polynomial algorithm exists to solve maxcut (Unless we discover $P = NP$) a multitude of approximation algorithms and heuristics have been developed in an attempt to solve this problem to a relative level of optimality. One simplistic method is a local search algorithm, which starts with an arbitrary set S and then iteratively either adds or removes an edge, as long as it helps the objective function. A more complex method would be the Goemans and Williamson algorithm, which combines semi-definite programming and a rounding procedure to produce an approximate solution to the max-cut problem.

In this paper we will focus our attention on IP formulations to the maxcut, along with a variety of relaxations and a heuristic which will help us find upper and lower bounds respectively. In section 2 we will discuss two separate formulations to the maxcut. Section 3 will feature a variety of relaxations to the maxcut, which will help us find upper bounds to our problem. Section 4 will discuss a heuristic which utilizes two-swap simulated annealing approach to find a sufficient lower bound to our problem. Section 5 will display the computational analysis for our study, and finally section 6 will be the conclusion to our findings.

2 Mathematical Formulations

2.1 Formulation 1

In this formulation, we define binary variables x_i for each $i \in V$ that indicate whether node i belongs to S. we also define the binary variables y_{ij} to indicate whether edge $\{ij\}$ is part of the cut. Using these variables, we can formulate this problem as follows:

$$
\begin{aligned}
z^* = \max \sum_{i,j \in E} w_{ij} y_{ij} \qquad & & (1) \\
st. \quad y_{ij} \leq 2 - x_i - x_j, \quad & \{ij\} \in E & (2) \\
y_{ij} \leq x_i + x_j, \quad & \{ij\} \in E & (3) \\
y_{ij} \geq x_i - x_j, \quad & \{ij\} \in E & (4) \\
y_{ij} \geq x_j - x_i, \quad & \{ij\} \in E & (5) \\
x_i \in \{0, 1\}, \quad & i \in V & (6) \\
y_{ij} \in \{0, 1\}, \quad & \{i, j\} \in E & (7)
\end{aligned}
$$

where constraint (2) imposes that if both ends of an edge are in S, this edge is not in the cut, (3) imposes that if both ends are in $\bar{S}$, this edge is not in the cut, (4) and (5) impose that if the ends of this edge are in different sets, this edge is in the cut, and (6) and (7) define the domain of the variables. In our study, we run this formulation using Gurobi and terminate with an optimality gap after 500 seconds of runtime.

2.2 Formulation 2

In this formulation, we define variables z_i for each $i \in V$ to take the value of -1 if $i \in S$ and 1 if $i \notin S$. Using these variables, we can formulate this problem as follows:

$$z^* = \frac{1}{4} \max \sum_{i \in V} \sum_{j \in V} w_{ij}(1 - z_i z_j) \tag{8}$$

$$st. \quad z_i \in \{-1, 1\}, \qquad i \in V \tag{9}$$

3 Relaxations for Max Cut

3.1 Eigenvalue Relaxation

Proposition 1. *The optimal value of z^* in the maxcut problem defined on graph $G = (V, E)$ satisfies*

$$z^* \leq z^{ev} := \frac{n}{4} \lambda_{max}(L(G))$$

Proof. Consider the maximization problem

$$\max\{\mathbf{z}^\top L(G)\mathbf{z} \,|\, \mathbf{z} \in \{-1, 1\}^n\}$$

Since $L(G) = D(G) - A(G)$, then $L(G)_{ii} = \sum_{j \in V} w_{ij}$ for $i \in V$ and $L(G)_{ij} = -w_{ij}$ for $i, j \in V$ such that $i \neq j$. We expand the objective function and obtain

$$\max_{\mathbf{z} \in \{-1,1\}} \sum_{i \in V} \sum_{j \in V} w_{ij} z_i z_i - \sum_{i \in V} \sum_{j \in V} w_{ij} z_i z_j$$

Furthermore,

$$\max_{\mathbf{z} \in \{-1,1\}} \sum_{i \in V} \sum_{j \in V} w_{ij}(z_i z_i - z_i z_j)$$

Since $z_i \in \{-1, 1\}$ then $z_i z_i = 1$, we can simplify the objective function to

$$\max_{\mathbf{z} \in \{-1,1\}} \sum_{i \in V} \sum_{j \in V} w_{ij}(1 - z_i z_j)$$

From the previous section we saw an almost identical formulation for max cut. The only difference being a multiplier of $\frac{1}{4}$. Let z^* represent the optimial solution for max cut

$$z^* = \frac{1}{4} \max\{\mathbf{z}^\top L(G)\mathbf{z} \,|\, \mathbf{z} \in \{-1, 1\}^n\} \tag{10}$$

Taking the continuous relaxation of (**??**), namely $z_i \in [-1,1]$ for $i \in V$, is equivalent to maximizing over the norm infinity, $\|\mathbf{z}\|_\infty \leq 1$. We can relax this further by maximizing over a ball of radius $\sqrt{n}$. In other words, our region is now defined where $\|\mathbf{z}\| \leq \sqrt{n}$.

$$z^* \leq \frac{1}{4}\max\{\mathbf{z}^\top L(G)\mathbf{z} \,|\, \mathbf{z} \leq \sqrt{n}\} \tag{11}$$

We can define our problem over the unit ball with a simple transformation. Let $\|\mathbf{z}\| = \sqrt{n}\mathbf{x}$ and we now have

$$z^* \leq \frac{n}{4}\max\{\mathbf{x}^\top L(G)\mathbf{x} \,|\, \|\mathbf{x}\| \leq 1\} \tag{12}$$

Since for any symmetric matrix $\mathbf{A}$, $\max\{\mathbf{x}^\top \mathbf{A}\mathbf{x} \,|\, \|\mathbf{x}\| \leq 1\} = \lambda_{\max(\mathbf{A})}$ and $L(G)$ is symmetric, then we have

$$z^* \leq z^{ev} := \frac{n}{4}\lambda_{max}(L(G)) \tag{13}$$

□

Proposition 2. *The optimal value of z^* in the maxcut problem defined on graph $G = (V,E)$ satisifes*

$$z^* \leq z^{ev} := -\frac{1}{4}\sum_{i=1}^{n} u_i + \frac{n}{4}\lambda_{max}(L(G) + diag(\mathbf{u}))$$

for all $\mathbf{u} \in \mathbb{R}^n$

Proof. Consider the maximization problem

$$\max\{\mathbf{z}^\top(L(G) + diag(\mathbf{u}))\mathbf{z} | \mathbf{z} \in \{-1,1\}\} \tag{14}$$

From the previous proof, we perform a similar technique to (**??**) to obtain the following optimization problem

$$\max_{\mathbf{z}\in\{-1,1\}} \sum_{i\in V}\left(\sum_{j\in V} w_{ij} + u_i\right) z_i z_i - \sum_{i\in V}\sum_{j\in V} w_{ij} z_i z_j$$

Rearranging terms and using the fact that $z_i z_i = 1$,

$$\max_{\mathbf{z}\in\{-1,1\}} \sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}(1 - z_i z_j) + \sum_{i=1} u_i \tag{15}$$

Which is closely related to the objective function in our prevoius formulation. We obtain the following relationship between (**??**) and z^*:

$$z^* = \frac{1}{4}\max\{\mathbf{z}^\top(L(G) + diag(\mathbf{u}))\mathbf{z} \,|\, \mathbf{z} \in \{-1,1\}\} - \frac{1}{4}\sum_{i=1}^{n} u_i$$

We repeat the process from the previous proof of relaxing the variables so that they are continuous and then further relaxing this problem onto a ball of radius $\sqrt{n}$.

$$z^* \leq \frac{1}{4}\max\{\mathbf{z}^\top(L(G) + diag(\mathbf{u}))\mathbf{z} \,|\, \|\mathbf{z}\| \leq \sqrt{n}\} - \frac{1}{4}\sum_{i=1}^{n} u_i$$

Let $\mathbf{z} = \sqrt{n}\mathbf{x}$.

$$z^* \leq \frac{n}{4}\max\{\mathbf{x}^\top(L(G) + diag(\mathbf{u}))\mathbf{x} \mid \|\mathbf{x}\| \leq 1\} - \frac{1}{4}\sum_{i=1}^{n} u_i$$

Finally, we obtain the inequality

$$z^* \leq z^{ev}(\mathbf{u}) = -\frac{1}{4}\sum_{i=1}^{n} u_i + \frac{n}{4}\lambda_{max}(L(G) + diag(\mathbf{u})) \tag{16}$$

$\square$

3.2 Lagrangian Dual

In the previous section we have proved several upper bounds on max cut based on the Laplace $L(G)$, including one which changes of the diagonal of $L(G)$ with some vector $\mathbf{u} \in \mathbb{R}^n$. In this section we will compute the Lagrangian dual bound of maximum cut, which corresponds to the values of $\mathbf{u}$ such that $z^{ev}(\mathbf{u})$ is minimized.

$$z^{LD} = \min\left\{ -\frac{1}{4}\sum_{i=1}^{n} u_i + \frac{n}{4}\lambda_{max}(L(G) + diag(\mathbf{u})) \,\middle|\, \mathbf{u} \in \mathbb{R}^n \right\}$$

Consider the gradient of our objective function, $\nabla z^{ev}(\mathbf{u})$. In order to compute $\nabla\lambda_{max}(L(G)+diag(\mathbf{u}))$ we first use the fact that $L(G)+diag(\mathbf{u})$ is a symmetric square matrix and therefore $\lambda_{max}(L(G) + diag(\mathbf{u})) = \max\{\mathbf{x}^\top(L(G) + diag(\mathbf{u}))\mathbf{x} \mid \|\mathbf{x}\| \leq 1\}$. This optimization problem can further be expanaded to

$$\max_{\|\mathbf{x}\|\leq 1} \sum_{i\in V}\left(\sum_{j\in V} w_{ij} + u_i\right) x_i x_i - \sum_{i\in V}\sum_{j\in V} w_{ij}x_i x_j$$

We can see that this objective value is maximized for $x_k = 1$ where $k = \arg\max_i\{\sum_{j\in V} w_{ij} + u_i \mid \sum_{j\in V} w_{ij} + u_i > 0\}$, and $x_i = 0$ for $i \in V \setminus \{k\}$. If there exists no positive value of $\sum_{j\in V} w_{ij} + u_i$ then the objective function is maximized for $\mathbf{x} = \mathbf{0}$. Therefore, $\nabla\lambda_{max}(L(G)+diag(\mathbf{u})) = e_k$ if $\lambda_{max}(L(G) + diag(\mathbf{u})) \geq 0$ and $\mathbf{0}$ otherwise and $\nabla z^{ev}(\mathbf{u})$ can be computed as

$$\nabla z^{ev}(\mathbf{u})_k = \begin{cases} \frac{n-1}{4} & \text{if } k = \arg\max_i \left\{ \sum_{j\in V} w_{ij} + u_i \,\middle|\, \sum_{j\in V} w_{ij} + u_i > 0 \right\} \\ -\frac{1}{4} & \text{otherwise} \end{cases}$$

3.2.1 Subgradient Algorithm

We implement the following subgradient algorithm in order to compute z^{LD}. We postpone details on our implementation decisions such as stopping criteria, initial solutions and choice of sequence h_k for a later section.

Algorithm 1: Subgradient Algorithm for MaxCut

```
Result: Computes the Lagrangian dual bound $z^{LD}$
Select an initial solution $\mathbf{u}_0$ and appropriate sequence $\{h_k\}_{k=0}^{\infty}$;
for $k \geq 0$ do
    Compute $z^{ev}(\mathbf{u})$ and $\nabla z^{ev}(\mathbf{u})$;
    $\mathbf{u}_{k+1} \leftarrow \mathbf{u}_k - h_k \frac{\nabla z^{ev}(\mathbf{u})}{\|\nabla z^{ev}(\mathbf{u})\|}$;
end
```

3.3 Semi Definite Relaxation

The semi-definite relaxation of formulation 2 is expressed as follows:

$$z^* = \frac{1}{4} max \quad \sum_{i \in V} \sum_{j \in V} w_{ij}(1 - Y_{ij}) \tag{17}$$

$$s.t \quad Y_{ii} = 1, \quad i \in Y \tag{18}$$

$$\mathbf{Y} \succeq 0 \tag{19}$$

The formulation is relaxed by introducing the matrix $\mathbf{Y}$. The elements Y_{ij} of this matrix correspond to the multiplication of the decision variables z_i and z_j from the original formulation. Additionally, the variables are relaxed such that $Y_{ij} \in \mathbb{R}$. Constraint (18) enforces the fact that $z_i = z_i$ in the original unrelaxed formulation, as the multiplication of 1 and 1 or -1 and -1 results in 1. This also makes sense conceptually, as the same node can't be in two different sides of the cut. Thus, we will maintain the diagonal of the matrix $\mathbf{Y}$ fixed during the solution of this formulation. Constraint (19), forces the values of Y_{ij} variables to stay between -1 and 1, and ultimately converge close to either 1 or -1 in feasible solutions.

To solve the formulation, we implement the ellipsoid algorithm. The algorithm is generally efficient and effective for convex problems, such as the relaxed formulation. However, we know that its nature causes it to be slow in practice in many cases, as it can iterate between infeasible and feasible solutions for a large number of steps. We will discuss this further in our computational analysis.

The ellipsoid algorithm relies on affine transformations between a unit "ball", $B(y_k, 1)$, and ellipsoids $E(y_k, \mathbf{H_k})$. Notice, that the value $\mathbf{y}$ is the current proposed solution at iteration k. Ultimately, the algorithm works on each iteration and its current $\mathbf{y_k}$ in the space of $B(0, 1)$, and transforms new solutions in the ellipsoid space to continually shrink and shift the ellipsoids and converge to better, feasible solutions. At each iteration, if the current solution is infeasible, a separating cut is added to move the next $\mathbf{y_k}$ towards feasibility: if it is not PSD, the eigenvector associated with the most violating eigenvalue is used to calculate the direction to move in for y_{k+1}. If the solution is feasible, we move in the direction of the objective function's gradient.

Suppose we define the feasible region for our semi-definite relaxation to be the set $K = \{y_{ij} : Y_{ii} = 1, Y_{ij} = Y_{ji}, Y \succeq 0, i \in V, j \in V\}$. In reality, when the algorithm is implement, we only need to maintaint a vector of variables $y_i, i \in V$, since the constraints $Y_{ii} = 1, Y_{ij} = Y_{ji}$ can be trivially disgarded without loss of generality, and the matrix $\mathbf{Y}$

can be constructed and converted back to the list $\mathbf{y}$ at every iteration. The algorithm is initialized by finding a value $R \in \mathbb{R}$ such that $K \subseteq B(0, R)$, and thus, our initial solution, $\mathbf{y_0}$, is the zero vector, and $\mathbf{Y}$ is the identity. Thus, we wish for our initial ellipsoid to be $B(0, R)$, so we initialize $\mathbf{H_0}$ to be $R^2\mathbf{Y}$. Thus, we iterate over the algorithm described earlier as follows:

Algorithm 2: Ellipsoid Algorithm for MaxCut.

```
Result: Solution to Semi-Definite Relaxation
$R \leftarrow 2n$;
$\mathbf{Y_0} \leftarrow$ identity;
$\mathbf{y_0} \leftarrow \mathbf{0}$ ;
$\mathbf{H_0} \leftarrow R^2\mathbf{Y}$;
$\mathbf{g} \leftarrow$ gradient(z);
$\epsilon \leftarrow 0.001$;
$change \leftarrow 100$;
$k \leftarrow 0$ while $change > \epsilon$ do
    if $\mathbf{Y_k} \succeq 0$ then
        $\mathbf{y_{k+1}} \leftarrow$ ellipsoidshift($\mathbf{Y_k}$, $\mathbf{g}$, $\mathbf{H_k}$) ;
        $\mathbf{H_{k+1}} \leftarrow$ newH($\mathbf{Y_k}$, $\mathbf{g}$, $\mathbf{H_k}$) ;
    else
        $\overline{g} \leftarrow$ separatingcut(min_eigenvector) ;
        $\mathbf{y_{k+1}} \leftarrow$ ellipsoidshift($\mathbf{Y_k}$, $\overline{\mathbf{g}}$, $\mathbf{H_k}$) ;
        $\mathbf{H_{k+1}} \leftarrow$ newH($\mathbf{Y_k}$, $\overline{\mathbf{g}}$, $\mathbf{H_k}$) ;
    end
    $k \leftarrow k + 1$ ;
    $change \leftarrow ||\mathbf{y_{k+1}} - \mathbf{y_k}||$ ;
end
return $\mathbf{y}$ ;
```

The side funtions ellipsoid shift, and new H, are implementations of the closed-form matrix operations from the project guidelines. The vector $\mathbf{g}$ remains constant, and is the gradient of the objective function. This is the direction we want to move in when our solution is already feasible. $\overline{\mathbf{g}}$ is calculated by taking the gradient of the most violating eigenvector in the function separatingcut.

4 Lower Bound Heuristic

5 Computational Analysis

6 Conclusion