CS 491I Approximation algorithms
Lecture notes
April 19th & 24th ,2001
Dejan Desovski

Semidefinite Programming

So far, we have used ***Integer Programs*** and their relaxation to ***Linear Programs***, in order to obtain approximation algorithms for ***NP Complete*** problems. We will demonstrate how we can use ***Quadratic Integer Programs*** (**QIP**) for the same purpose, producing approximation algorithms for **NP Complete** problems, with better bounds than the previously devised algorithms.

The general form of a **QIP** is the following:

$$\min z = \vec{x}^T \mathbf{Q}\vec{x} + \vec{c}\vec{x}$$
$$s.t.\ \mathbf{A}\vec{x} \le \vec{b}$$
$$x_i \in \{-1,1\}$$

QIP, like **IP** is **NP Complete** and thus it is highly unlikely that a polynomial time algorithm for solving **QIP**'s exists, unless **NP** = **P** is proven.

Having a **QIP** for some problem we can relax it to ***Quadratic Program*** (**QP**). The general form of a **QP** is the following:

$$\min z = \vec{x}^T \mathbf{Q}\vec{x} + \vec{c}\vec{x}$$
$$s.t.\ \mathbf{A}\vec{x} \le \vec{b}$$
$$\vec{x} \ge \vec{0}$$

Unfortunately, even **QP** is **NP Complete**. The proof that **QP** $\in$ **NP** is difficult. We will assume that this is true, and just provide proof of the **NP hardness** by reducing it to the 3-SAT problem.

Theorem 1: **IP** $\le_p$ **QP**.

Proof: We assume that we have proven that **QP** $\in$ **NP**. We start with the 3-SAT problem, we can formulate an **IP** formulation by assigning a integer variable $x_i \in \{0,1\}$ for each boolean variable a_i in the 3-SAT instance. Then, each clause is transformed into inequality by replacing each boolean variable a_i with its integer counterpart x_i, each negated boolean variable $\overline{a_i}$ with $1 - x_i$, and the sum of the variables in each clause should be greater than or equal to 1 in order that clause to be satisfied. An example of this process is the following transformation:

$\overline{a_1} \vee a_2 \vee a_3 \rightarrow 1 - x_1 + x_2 + x_3 \ge 1$

Thus we will get the following IP formulation of 3-SAT:

$$\mathbf{A}\vec{x} \ge \vec{b}$$
$$x_i \in \{0,1\}$$

This proves that 3-SAT $\le_p$ IP.

Using this IP we can write a QP formulation of the 3-SAT problem. First observe that the minimum of the function $f(x) = x(1-x)$, $x \in [0,1]$, is 0, and it is obtained when *x* is 0 or 1. Using this, we can formulate the 3-SAT problem as following:

$$\min z = \sum_{i=1}^{n} x_i(1-x_i) = \vec{x}^T \begin{bmatrix} -1 & 0 & 0 & & 0 \\ 0 & -1 & 0 & \cdots & 0 \\ 0 & 0 & -1 & & 0 \\ & \vdots & & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & -1 \end{bmatrix} \vec{x} + \begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix} \vec{x} = \vec{x}^T(-\mathbf{I})\vec{x} + \vec{1}^T\vec{x}$$

$$A\vec{x} \geq \vec{b}$$

$$0 \leq x_i \leq 1$$

So, we have proved that **IP** $\leq_p$ **QP**.

Fortunately, there are some **QP** instances which can be solved in polynomial time. These are the instances where the matrix **Q** is positive semidefinite matrix, and these programs are called **Semidefinite *Programs*** (**SDP**). We will demonstrate these ideas on the Max-Cut problem.

SDP Max-Cut

Problem Statement: Given graph ***G* = <*V*, *E*>**, **| *V* | = *n***, **| *E* | = *m***. Each edge $(i, j) \in$ ***E*** has cost w_{ij} associated with it. Partition the vertices of ***G*** in two sets ***V*₁** and ***V*₂**, such that, the cost of the edges with one vertex in $\boldsymbol{V_1}$ and the other in $\boldsymbol{V_2}$ is maximal.

We begin by writing **QIP** formulation of the Max-Cut problem. We assign a variable y_i to each vertex v_i, defined in the following way:

$$y_i = \begin{cases} 1, \text{if } v_i \in V_1 \\ -1, \text{if } v_i \in V_2 \end{cases}$$

If two vertices are in the opposite sets then the edge connecting them increases the objective function. We can model this statement with the following formula:

$$\frac{1}{2}(1 - y_i y_j)w_{ij}$$

We obtain the following **QIP** formulation for the Max-Cut problem:

$$\max z = \sum_{j=1}^{n} \sum_{i=1}^{j-1} w_{ij}(1 - y_i y_j)$$

$$y_i \in \{-1, 1\}, \forall i = 1, \ldots, n$$

Now we can relax the **QIP** to **QP**, by letting the variables y_i to be vectors on the unit circle. Thus we are expanding the one-dimensional unit sphere $S = \{-1, 1\}$ to the two-dimensional unit sphere $S^2 = \{(a,b) | a^2 + b^2 = 1\}$ (see **Figure 1**).

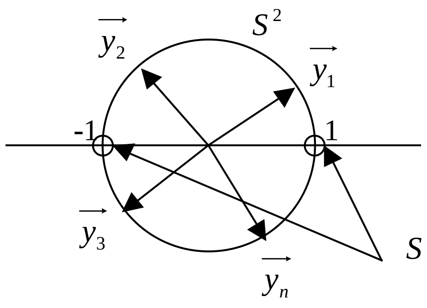

Figure 1: Relaxing the domain of the variables

The **QP** formulation, obtained by this relaxation is the following:

$$\max z = \sum_{j=1}^{n} \sum_{i=1}^{j-1} w_{ij} \left(1 - \overrightarrow{y_i}\overrightarrow{y_j}\right)$$

$$\overrightarrow{y_i} \in S^2, \forall i = 1, \ldots, n$$

We notice that each solution of the **QIP** is also solution of the **QP**, because we can get vectors that lie on the unit circle from the **QIP** solutions by padding them with 0, i.e. $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$ or $\begin{bmatrix} -1 \\ 0 \end{bmatrix}$. But the reverse is not true, not any vector on the unit circle represents a feasible solution to the **QIP**, thus we need to perform rounding of the **QP** solution in order to obtain a feasible solution to the **QIP** and consequently – solution for the Max-Cut problem.

The following is the proposed algorithm:

```
1. Solve the QP
2. we get $\left(\overrightarrow{y_1}, \overrightarrow{y_2}, \ldots, \overrightarrow{y_n}\right)$
3. Select a random vector $\vec{r}$
4. for each vertex $v_i$
       if $\overrightarrow{y_i} \cdot \vec{r} < 0$ then put $v_i$ in $V_1$
       else put $v_i$ in $V_2$
```

Algorithm 1: **QP** Max-Cut

This basically represents a randomization algorithm – the random vector $\vec{r}$ is used to partition the other vectors in two sets induced by its normal (see **Figure 2**), and thus obtain a feasible solution.

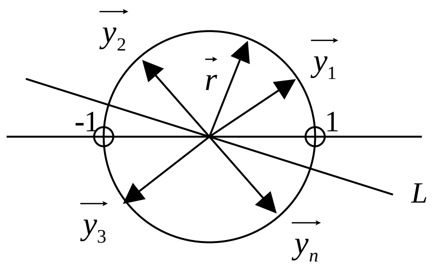

Figure 2: Partitioning of $\overrightarrow{y_i}$ induced by L

We have showed that **QP** cannot be efficiently solved in general, but later we will show that this particular instance is a **SDP** and can be solved in polynomial time. Lets focus on the analysis of the algorithm.

Analysis

First we will need to find the minimum value β of the following function:

$$\beta = \min_{0<\alpha\leq\pi} \frac{2\alpha}{\pi(1-\cos\alpha)}$$

It is difficult to calculate this minimum by algebraic means because of the transcendental nature of the first derivative of this function, but by using numerical methods we can obtain the following result: $\beta > 0.8785$.

Next we calculate the expected cost of the approximation algorithm.

$$E[m_H] = \sum_{j=1}^{n}\sum_{i=1}^{j-1} w_{ij} \cdot Pr\left[\operatorname{sgn}\left(\overrightarrow{y_i}\cdot\vec{r}\right) \neq \operatorname{sgn}\left(\overrightarrow{y_j}\cdot\vec{r}\right)\right]$$

where: $\operatorname{sgn}(x) = \begin{cases} 1, \text{if } x \geq 0 \\ -1, \text{if } x < 0 \end{cases}$

$Pr\left[\operatorname{sgn}\left(\overrightarrow{y_i}\cdot\vec{r}\right) \neq \operatorname{sgn}\left(\overrightarrow{y_j}\cdot\vec{r}\right)\right]$ denotes the probability that the normal *L* cuts the smaller of the angles between the vectors $\overrightarrow{y_i}$ and $\overrightarrow{y_j}$, lets call this angle α_{ij}, (i.e., the vectors lie on opposite sides of the normal *L* see **Figure 3**). This probability is equal to:

$$Pr\left[\operatorname{sgn}\left(\overrightarrow{y_i}\cdot\vec{r}\right) \neq \operatorname{sgn}\left(\overrightarrow{y_j}\cdot\vec{r}\right)\right] = \frac{\alpha_{ij}}{2\pi} + \frac{\alpha_{ij}}{2\pi} = \frac{\alpha_{ij}}{\pi}$$

That is, the first end *P* of the normal lies in the smaller angle between the vectors, or the second end *T* of the normal lies in the smaller angle between the vectors.

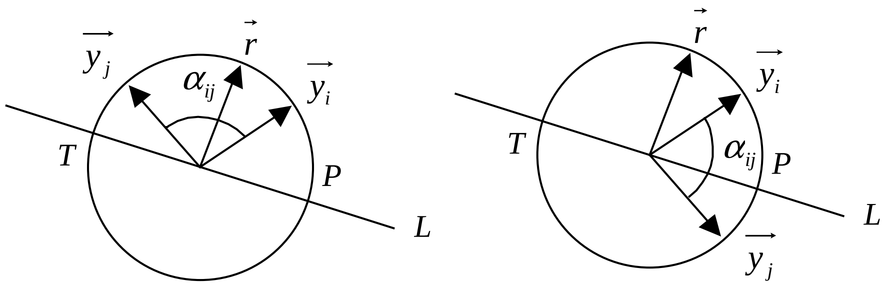

Figure 3: When the normal L cuts the smaller angle between two vectors

Thus we have:

$$E[m_H] = \sum_{j=1}^{n}\sum_{i=1}^{j-1} w_{ij}\frac{\alpha_{ij}}{\pi} = \sum_{j=1}^{n}\sum_{i=1}^{j-1} w_{ij}\frac{\cos^{-1}\left(\overrightarrow{y_i}\,\overrightarrow{y_j}\right)}{\pi}$$

using that: $\overrightarrow{y_i}\cdot\overrightarrow{y_j} = \left\|\overrightarrow{y_i}\right\|\cdot\left\|\overrightarrow{y_j}\right\|\cdot\cos(\alpha_{ij}) = 1\cdot 1\cdot\cos(\alpha_{ij}) = \cos(\alpha_{ij})$.

Now we can use the previously calculated β.
We know that:

$$\beta \le \frac{2\cos^{-1}\left(\overrightarrow{y_i}\overrightarrow{y_j}\right)}{\pi\left(1-\overrightarrow{y_i}\overrightarrow{y_j}\right)}$$

by rewriting we get: $\frac{\cos^{-1}\left(\overrightarrow{y_i}\overrightarrow{y_j}\right)}{\pi} \ge \frac{\beta}{2}\left(1-\overrightarrow{y_i}\overrightarrow{y_j}\right)$

and substituting for the $E[m_H]$ we get:

$$E[m_H] = \frac{\beta}{2}\sum_{j=1}^{n}\sum_{i=1}^{j-1} w_{ij}\left(1-\overrightarrow{y_i}\overrightarrow{y_j}\right) \ge \beta \cdot OPT_{QP} \ge \beta \cdot OPT$$

First we notice that we can extend the range of the **QP** from S^2 to S^n - here S^n denotes *n*-dimensional unit sphere. This relaxation does not affect the analysis and the obtained approximation bound stays the same. In order to justify this relaxation we need the following results from the linear algebra.

Definition 1: A symmetric matrix **M** is said to be positive semidefinite if $\vec{x}^T \mathbf{M}\vec{x} \ge 0$ for all $\vec{x}$.

Theorem 2 (Cholesky factorization): A symmetric matrix **M** is positive semidefinite if and only if there exists a matrix **P** such that **M** = **P**^{T}**P**. Moreover, if **M** is positive semidefinite, then the matrix **P** can be computed in polynomial time. The proof of this theorem and some other properties of semidefinite matrices can be found in **[2]**.

Theorem 3: Given *n* vectors $\overrightarrow{y_1},\ldots,\overrightarrow{y_n} \in S^n$, the matrix **M** defined as $\mathbf{M}_{i,j} = \overrightarrow{y_i}\overrightarrow{y_j}$ is positive semidefinite. The proof is left as an exercise to the reader.

On other hand, from **Theorem 2**, it follows that, given a positive semidefinite matrix **M** such that $\mathbf{M}_{i,j} = 1$ for $i = 1,\ldots,n$, it is possible to compute, in polynomial time, a set of *n* vectors $\overrightarrow{y_1},\ldots,\overrightarrow{y_n} \in S^n$ such that $\mathbf{M}_{i,j} = \overrightarrow{y_i}\overrightarrow{y_j}$.

In order words, the **QP** is equivalent to the following **SDP**:

$$\max z = \frac{1}{2}\sum_{j=1}^{n}\sum_{i=1}^{j-1} w_{ij}\left(1-\mathbf{M}_{i,j}\right)$$

$$s.t.\ \ \mathbf{M}\ \text{is positive semidefinite}$$

$$\mathbf{M}_{i,i} = 1, 1 \le i \le n$$

This **SDP** can be solved in polynomial time, which is proportional to *n* (the number of variables) and $\log\left(\frac{1}{\varepsilon}\right)$ where ε is the rounding error of the computation.

By solving this SDP we calculate the elements $\mathbf{M}_{i,j} = \overrightarrow{y_i}\overrightarrow{y_j}$ in the matrix **M**, and then by **Theorem 2** we can calculate the actual vectors $\overrightarrow{y_i}$.

Reference

[1] G. Ausiello, e.t. all. *Complexity and Approximation - Combinatorial Optimization Problems and Their Approximability Properties*. pp[162-168]. Springer-Verlag, New York, 1999.

[2] Ivar Tammeraid, Jüri Majak, Seppo Pohjolainen, Tero Luodeslampi**.** *LINEAR ALGEBRA***.**
http://www.math.ut.ee/~toomas_l/linalg
http://www.cs.ut.ee/~toomas_l/linalg/lin2/node24.html, May 2001.