MA 796s-OR 791k Project Report December 2007

Max-cut Problem

Lecturer: Kartik Sivaramakrishnan Scribe: Sahar Karimi

1 Introduction

Max-cut problem is one of many NP-hard graph theory problems which attracted many researchers over the years. Though there is almost no hope in finding a polynomial-time algorithm for max-cut problem, various heuristics, or combination of optimization and heuristic methods have been developed to solve this problem. Among them is the efficient algorithm of Goemans and Williamson. Their algorithm combines Semi-definite programming and a rounding procedure to produce an approximate solution to the max-cut problem. This approximate solution, however, turns out to be quite good and its error is bounded by 0.878.
This report is organized as follows. We first define the max-cut problem, and review the notation in 1.1 and 1.2. In section 2, we formulate the max-cut problem. Section 3 is dedicated to the Goemans-Williamson algorithm. Finally in section 4, we talk about "Dual-scaling algorithm". This algorithm is proposed for solving large-scale sparse SDP's by Benson et al. in [2]. We explain the algorithm, and we show how it can be used to solve the SDP relaxation of max-cut problem.

1.1 Max-cut Problem

A cut in a weighted undirected graph $G_w = (V,E)$, is defined as partition of the vertices of G into two sets; and the weight of a cut is the summation of weights of the edges that has an end point in each set (i.e. the edges that connect vertices of one set to the vertices of the other). Trivially, one can define the max-cut problem as the problem of finding a cut in G with maximum weight.

1.2 Notation

In this report, $G_w = (V,E)$ stands for a weighted undirected graph, and $w_{ij} \geq 0$ is the weight of edge $e_{ij} \in E$.
Lower case letters are vectors and upper case letters denote matrices. e_i is the unit vector which has 1 in the ith position and zeros elsewhere; e is the all ones vector, and I is the identity matrix.
$\mathbb{R}^n$ represents the space of real vectors of size n, and $\mathbb{S}^n$, $\mathbb{S}^n_+$, and $\mathbb{S}^n_{++}$ represent the space of symmetric matrices of size n, symmetric positive semidefinite matrices of size n, and symmetric positive definite matrices of size n, respectively.
For any two vectors v and w, $v.w = v^t w$ is the inner product of these two vectors; and vw^t is the outer product of these two vectors. Similarly, for any two matrices A and B,

$A \bullet B = trace(A^t B) = \sum_{\forall i,j} A_{ij} B_{ij}$ is the inner product of matrices A and B.
$||X||_\infty$ for matrix X is the infinity norm of matrix X, and $||X||$ represents the Frobenius norm of X.
Finally we define the operator $\mathscr{A} : \mathbb{S}^n \to \mathbb{R}^n$, and its adjoint $\mathscr{A}^t : \mathbb{R}^m \to \mathbb{S}^n$ as follows:

$$\mathscr{A}(X) = \begin{pmatrix} A_1 \bullet X \\ \vdots \\ A_n \bullet X \end{pmatrix}$$

$$\mathscr{A}^t(y) = \sum_{i=1}^{m} y_i A_i$$

2 Formulation of the Max-cut Problem

In this section, we first model the max-cut problem, then we show how we can obtain the semidefinite relaxation of the max-cut problem.

Let us assign a variable x_i to each vertex of $G_w = (V,E)$, and define this variable as follows:

$$x_i = \begin{cases} 1 & \text{if the } i\text{th vertex is in } S \\ -1 & \text{if the } i\text{th vertex is in } \overline{S} \end{cases}$$

where S is a subset of V, and $\overline{S}$ is the complement of S.

Now we can model the max-cut problem as

$$\begin{aligned} Max \quad & \tfrac{1}{2} \textstyle\sum_{i<j} w_{ij}(1 - x_i x_j) \\ s.t. \quad & x_i \in \{+1, -1\} \quad i = 1 \dots n \end{aligned} \tag{1}$$

The feasible region of model (1) obviously defines a cut, so we only need to show that the objective function gives the value of this cut. Notice that for any i and j $(1 - x_i x_j)$ is 0 if vertices i and j are in the same set, and 2 otherwise. This means that $\sum_{i<j} w_{ij}(1 - x_i x_j)$ is twice as much as the weight of the cut. Dividing this value by 2 gives the weight of the cut. It is worth mentioning here that we can write the same objective function as $\frac{1}{4} \sum_{i,j} w_{ij}(1 - x_i x_j)$.

Model (1) can be translated into vector notation as follows:

$$\begin{aligned} Max \quad & \tfrac{1}{4} x^t L x \\ s.t. \quad & x_i^2 = 1 \quad i = 1 \dots n \end{aligned} \tag{2}$$

where L is the laplacian matrix of the graph.

Let's define a new variable $X = xx^t$. It's easy to show that $X \in S^n_+$. Now we can write model (2) as follows:

$$\begin{array}{ll} Max & \frac{1}{4}L \bullet X \\ s.t. & Diag(X) = e \\ & rank(X) = 1 \\ & X \in S^n_+ \end{array} \tag{3}$$

This problem is still hard to solve because of the rank constraint. Relaxing this constraint gives us the following relaxation:

$$\begin{array}{ll} Max & \frac{1}{4}L \bullet X \\ s.t. & Diag(X) = e \\ & X \in S^n_+ \end{array} \tag{4}$$

Model (4) is an SDP problem which can efficiently be solved in polynomial-time, and gives us an upper bound on the max-cut problem.

3 Goemans-Williamson Algorithm

As we previously mentioned Goemans-Williamson algorithm is a rounding procedure for finding an approximate solution to the max-cut problem. In what follows we first review the algorithm, then we show how the 0.878 bound can be obtained.

Goemans-Williamson Algorithm:

1. Solve model (4). As mentioned this model can be solved efficiently. Let's call the optimal solution to this model X^*.
2. Compute the Cholesky factorization of X^*; i.e. $X^* = V^t V$. Let $v_i, i = 1 \dots n$ represent the normalized columns of V (i.e. $||v_i|| = 1$ for $i = 1 \dots n$).
3. Rounding Procedure:
   set $S = \emptyset$
   (a) Uniformly generate a random vector r on the unit n-sphere.
   (b) For $i = 1 \dots n$: If $v_i^t r > 0$ assign vertex i to S; else assign vertex i to $\overline{S}$.
   (c) Find the weight of the obtained cut by $\frac{1}{4}L \bullet X$, where $X = xx^t$.
4. Repeat the rounding procedure.

This approximation algorithm has guaranteed performance of 0.878; however the solution obtained by this algorithm is better than 0.878 in many cases. In what follows we show how this bound is achieved.

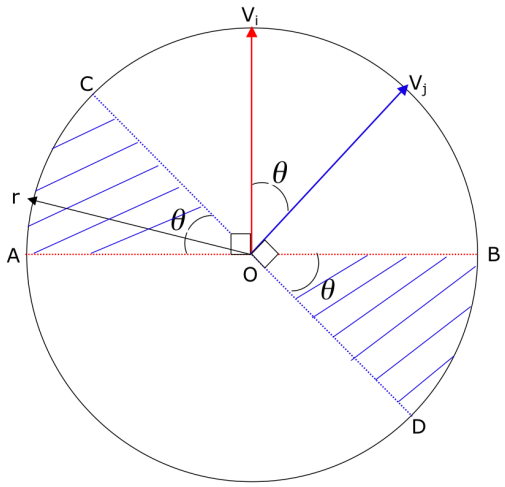

Figure 1: Finding the Probability

Fact 1. *The projection of r onto a plane is uniformly distributed on a unit circle.*

Lemma 1. $Pr[i \in S, j \notin \overline{S} \text{ or } i \in S, j \in \overline{S}] = \frac{1}{\pi} arccos(v_i v_j)$.

Proof. Let H denote the plane obtained by v_i and v_j. Figure 1 shows the unit circle containing v_i and v_j on this plane.

Let $r = r_\perp + r'$, where r' is the projection of r on H, and $r_\perp$ is the projection of r on the normal vector of H. Using fact 1, we know that r' is uniformly distributed on the unit circle. Also we know that for any vector v on H, $v^t r = v^t r'$ since $v^t r_\perp = 0$.

Let θ denote the angle between v_i and v_j, then $\theta = arccos(v_i^t v_j)$. Furthermore, $\widehat{COA}$ and $\widehat{BOD}$ are also θ since the sides of each of these angles are perpendicular to the sides of $\widehat{v_i O v_j}$.

Finally notice that if r lies anywhere on the arc ACB, then $v_i^t r \geq 0$, else $v_i^t r \leq 0$. Similarly if r lies anywhere on the arc CBD, then $v_j^t r \geq 0$, else $v_j^t r \leq 0$. Therefore the only region in the unit circle in which $v_i^t r > 0, v_j^t r < 0$ or $v_i^t r < 0, v_j^t r > 0$ is the region marked with blue lines in figure 1.

As a result,

$Pr[i \in S, j \notin \overline{S} \text{ or } i \in S, j \in \overline{S}] = \frac{2\theta}{2\pi} = \frac{1}{\pi} arccos(v_i^t v_j)$

□

Lemma 2. $\min_{-1 \leq x \leq 1} \frac{\frac{1}{\pi} arccos(x)}{\frac{1}{2}(1-x)} \geq 0.878$. $(\Rightarrow \frac{1}{\pi} arccos(x) \geq 0.878(\frac{1}{2}(1-x))$.

Proof. Can be easily verified using any mathmatical software. □

Theorem 1. *Goemans-Williamson algorithm is a 0.878 approximation algorithm.*

Proof. $E[W] = \sum_{i<j} w_{ij} Pr[i \in S, j \notin \overline{S} \text{ or } i \in S, j \in \overline{S}] = \sum_{i<j} w_{ij} \frac{1}{\pi} arccos(v_i^t v_j) \geq 0.878 \frac{1}{2} \sum_{i<j} w_{ij}(1 - v_i^t v_j) = 0.878 Z_{SDP} \geq 0.878 Z_{opt}$.
where Z_{SDP} is the objective value obtained by the SDP relaxation, and Z_{opt} is the optimal value of the max-cut problem. □

4 Dual-scaling Algorithm

As mentioned before, Dual-scaling algorithm is proposed by Benson et al. in [2] for solving large-scale sparse SDP's. This algorithm, which is basically a potential-reduction method, uses a dual potential function instead of the primal-dual potential function to solve the problem. In many combinatorial applications, the primal problem is dense while its dual can be very sparse. Using a potential function which captures only the dual problem, enables one to take the advantage of sparsity of the problem. Savings in computational cost, and exploring sparsity are major benefits of dual-scaling algorithm. Recall that the max-cut SDP relaxation and its dual had the following format:

$$\begin{array}{lll} min & C \bullet X & \\ s.t. & Diag(X) = e & \\ & X \in S_+^n & \end{array} \quad \longrightarrow \quad \begin{array}{lll} Max & e^t y & \\ s.t. & Diag(y) + S = C & \text{where } C = -\frac{1}{4}L \\ & S \in S_+^n & \end{array}$$

Generally C is very sparse for large sparse graphs and $Diag(y)$ is just a diagonal matrix, so S can be as sparse as C, while the primal variable X can be totally dense. Therefore dual-scaling algorithm can be very beneficial in solving large and sparse max-cut instances. As reported in [2] this method could solve max-cut instances with n up to 3000.

In subsection 4.1 we go over the dual-scaling algorithm and major results presented in [2]. We skip the proofs here and we refer interested readers to [2] for proofs of the results discussed here. Then in subsection 4.2, we explain dual-scaling algorithm for the max-cut problem. Finally in subsection 4.3 we go over two algorithms proposed for finding the Gram matrix.

4.1 Dual-scaling: The algorithm and theory

Dual-scaling algorithm uses dual potential function

$$\psi(y, \bar{z}) = \rho ln(\bar{z} - b^t y) - ln(det(S)) \tag{5}$$

with gradient

$$\bigtriangledown \psi(y, \bar{z}) = \frac{\rho}{\bar{z} - b^t y} b + \mathscr{A}(S^{-1}) \tag{6}$$

instead of Tanabe-Todd-Ye primal-dual potential function

$$\psi_{pd}(X,S) = \rho ln(X \bullet S) - ln(det(X)) - ln(det(S)). \tag{7}$$

When $\rho > n$ in (7), the infimum of the potential function occurs at an optimal solution. Also notice that the first term in this function captures the duality gap, while the second and third terms of this function capture the primal feasibility, and dual feasibility, respectively.
In (5), $\bar{z}$ is an upper bound for the objective function of the dual problem; i.e. it's $\bar{z} = C \bullet X$ for some feasible X. So in this function the first term still captures the duality gap and the second term captures dual feasibility. Same as what we said for the primal-dual potential function, the infimum of (5) happens at an optimal solution when $\rho > n + \sqrt{n}$.

An overestimator for the potential function can be obtained as

$$\psi(y,\bar{z}^k) - \psi(y^k,\bar{z}^k) \leq \bigtriangledown \psi(y^k,\bar{z}^k)^t (y - y^k) + \frac{||(s^k)^{-\frac{1}{2}}(\mathscr{A}^t(y-y^k))(s^k)^{-\frac{1}{2}}||}{2(1-||(s^k)^{-\frac{1}{2}}(\mathscr{A}^t(y-y^k))(s^k)^{-\frac{1}{2}}||_\infty} \tag{8}$$

where (y^k, S^k) is the current iteration and $\bar{z}^k$ is the current upper bound.
Using the overestimator (8) and having strictly feasible (y^k, S^k) and upper bound $\bar{z}^k$ each iteration solves:

$$\begin{array}{ll} min & \bigtriangledown \psi(y^k,\bar{z}^k)^t (y - y^k) \\ s.t. & ||(s^k)^{-\frac{1}{2}}(\mathscr{A}^t(y-y^k))(s^k)^{-\frac{1}{2}}|| \leq \alpha \end{array} \tag{9}$$

where constant $\alpha < 1$.

Using first order KKT conditions and ellipsoidal constraint, the solution of (9), y^{k+1}, can be obtained using

$$\begin{array}{ll} direction & d(\bar{z}^k)_y = -(M^k)^{-1} \bigtriangledown \psi(y^k,\bar{z}^k) \\ stepsize & \beta = \frac{\alpha}{\sqrt{\bigtriangledown \psi(y^k,\bar{z}^k) d(\bar{z}^k)_y}} \end{array} \tag{10}$$

where M^k is the Gram matrix and can be computed as

$$M^k = \begin{pmatrix} A_1(S^k)^{-1} \bullet (S^k)^{-1}A_1 & \cdots & A_1(S^k)^{-1} \bullet (S^k)^{-1}A_m \\ \vdots & \ddots & \vdots \\ A_m(S^k)^{-1} \bullet (S^k)^{-1}A_1 & \cdots & A_m(S^k)^{-1} \bullet (S^k)^{-1}A_m \end{pmatrix} \tag{11}$$

In subsection 4.3, we discuss two algorithms presented in [2] for finding M^k efficiently in special classes of problems.

To find a feasible primal point, the following least squares model is solved at each iteration.

$$\begin{array}{ll} min & ||(s^k)^{\frac{1}{2}} X (s^k)^{\frac{1}{2}} - \frac{\bar{z}^k - b^t y}{\rho} I|| \\ s.t. & \mathscr{A}(X) = b \end{array} \tag{12}$$

This problem looks for a matrix $X(\bar{z}^k)$ near to the central path, and its solution can be computed by

$$X(\bar{z}^k) = \frac{\bar{z}^k - b^t y}{\rho} (S^k)^{-1} (\mathscr{A}^t (d(\bar{z}^k)_y) + S^k)(S^k)^{-1} \tag{13}$$

however in dual-scaling algorithm we don't need to compute $X(\bar{z}^k)$ at each iteration; we just need to compute the objective value at $X(\bar{z}^k)$. We can compute $C \bullet X(\bar{z}^k)$ by the following equation and leave computing $X(\bar{z}^k)$ to the very end of the algorithm.

$$C \bullet X(\bar{z}^k) = b^t + X(\bar{z}^k) \bullet S^k = b^t y^k + \frac{\bar{z}^k - b^t y}{\rho}(d(\bar{z}^k)_y^t \mathscr{A}((S^k)^{-1}) + n) \tag{14}$$

since we have computed $d(\bar{z}^k)_y$ and $\mathscr{A}((S^k)^{-1})$ for finding direction and gradient before, the cost of finding the new upper bound is just a dot product.

Let us define

$$P(\bar{z}^k) = \frac{\rho}{\bar{z}^k - b^t y}(S^k)^{\frac{1}{2}} X(\bar{z}^k)(S^k)^{\frac{1}{2}} - I \tag{15}$$

Now we can have the following Lemma.

Lemma 3. *Let* $\mu^k = \frac{\bar{z}^k - b^t y^k}{n}, \mu = \frac{X(\bar{z}^k) \bullet S^k}{n} = \frac{C \bullet X(\bar{z}^k) - b^t y^k}{n}, \rho > n + \sqrt{n}$, *and* $\alpha < 1$. *If*

$$||P(\bar{z}^k)|| < min(\alpha \sqrt{(\frac{n}{n+\alpha^2}}, 1 - \alpha) \tag{16}$$

then:

1. $X(\bar{z}^k) \succ 0$ *(i.e.* $X(\bar{z}^k)$ *is strictly feasible)*
2. $||(S^k)^{\frac{1}{2}} X(\bar{z}^k)(S^k)^{\frac{1}{2}} - \mu I|| \leq \alpha\mu$
3. $\mu \leq (1 - \frac{\alpha}{2\sqrt{n}})\mu^k$

The above lemma is used both as a certificate for updating the upper bound and as a useful element in updating y^{k+1}. Using $||P(\bar{z}^k)||$, we can show that

$$||P(\bar{z}^k)|| = \sqrt{- \bigtriangledown \psi(y^k, \bar{z}^k) d(\bar{z}^k)_y} \tag{17}$$

and

$$\alpha ||P(\bar{z}^k)|| = - \bigtriangledown \psi(y^k, \bar{z}^k)(y^{k+1} - y^k) \tag{18}$$

By (17), and (18) we can update y^{k+1}, and S^{k+1} as follows:

$$\begin{array}{c} y^{k+1} = y^k + \beta d(\bar{z}^k)_y = y^k + \frac{\alpha}{||P(\bar{z}^{k+1})||} d(\bar{z})_y \\ S^{k+1} = C - \mathscr{A}^t(y^{k+1}) \end{array} \tag{19}$$

If $\alpha < 1$ then the positive definiteness of S^{k+1} is guaranteed, so we'll have a strictly feasible solution; and the reduction in the potential function satisfies

$$\psi(y^{k+1}, \bar{z}^{k+1}) - \psi(y^k, \bar{z}^k) \leq -\alpha ||P(\bar{z}^k)|| + \frac{\alpha^2}{2(1-\alpha)} \tag{20}$$

We are now ready to present the complete algorithm.

Dual-Scaling Algorithm
Given an upper bound $\bar{z}^0$, strictly feasible (y^0, S^0), $\rho > n + \sqrt{n}, \alpha \in (0,1)$

1. Compute $\mathscr{A}((S^k)^{-1})$ and M^k

2. Find $d(\bar{z}^k)_y$ using (10)
3. Compute $||P(\bar{z}^k)||$ using (17)
4. If $||P(\bar{z}^k)||$ satisfies (16),
   $X^{k+1} = X(\bar{z}^k), z^{k+1} = C \bullet X^{k+1}$, and $(y^{k+1}, S^{k+1}) = (y^k, S^k)$.
   Else,
   $y^{k+1} = y^k + \frac{\alpha}{||P(\bar{z}^k)||} d(\bar{z}^{k+1})_y, \quad S^{k+1} = C - \mathscr{A}^t(y^{k+1})$,
   $(X^{k+1}, Z^{k+1}) = (X^k, Z^k)$.
5. Repeat until $\bar{z}^k - b^t y^k < \varepsilon$

Theorem 2. *Using the above algorithm,*

$$\psi(X^{k+1}, S^{k+1}) \leq \psi(X^k, S^k) - \delta$$

where $\delta > \frac{1}{50}$ *for a suitable* α.

Corollary 1. *The algorithm terminates in at most* $O((\rho - n) ln(\frac{X^0 \bullet S^0}{\varepsilon}))$ *iterations.*

4.2 Dual-scaling Algorithm for Max-cut Problem

For max-cut problem, the dual potential function and its gradient can be written as follows:

$$\psi(y, \bar{z}) = -\rho ln(\bar{z} - e^t y) - ln(det(S)) \tag{21}$$

$$\bigtriangledown \psi(y, \bar{z}) = \frac{\rho}{\bar{z} - b^t y} e + diag(S^{-1}). \tag{22}$$

Same as what mentioned in the general algorithm we obtain the next iteration direction and step size by (10); however notice that for max-cut problem $M^k = (S^k)^{-1} \circ (S^k)^{-1}$, where $\circ$ represents the Hadamard (component wise) product.
To find direction $d(\bar{z}^k)_y$, we follow the following equations:

$$\begin{gathered} dy_1 = (M^k)^{-1} e \\ dy_2 = (M^k)^{-1} diag((S^k)^{-1}) \\ d(\bar{z}^k)_y = \frac{\rho}{z^k - e^t y^k} dy_1 - dy_2 \end{gathered} \tag{23}$$

This way of computing direction gives us flexibility in recomputing direction when the upper bound changes. We only need to change the denominator of the coefficient of dy_1 if $\bar{z}^k$ changes. Another advantage of the max-cut problem is in finding improved upper bounds. In the general algorithm, we used condition (16) in lemma 3 (i.e. if $||P(\bar{z}^k)||$ is sufficiently small) as a certificate that the new bound is feasible. However, we frequently have an improved upper bound for larger values of $||P(\bar{z}^k)||$. Luckily in the max-cut problem we can check the feasibility of the solution. Remember that we found primal solution in (13), and it is

$$X(\bar{z}^k) = \frac{\bar{z}^k - e^t y}{\rho} (S^k)^{-1} (Diag(d(\bar{z}^k)_y) + S^k)(S^k)^{-1}$$

with objective value of

$$C \bullet X(\bar{z}^k) = b^t + X(\bar{z}^k) \bullet S^k = b^t y^k + \frac{\bar{z}^k - e^t y}{\rho} (d(\bar{z}^k)_y^t diag((S^k)^{-1}) + n) \tag{24}$$

for the max-cut problem.

Since $X(\bar{z}^k) \succ 0$ if and only if $Diag(d(\bar{z}^k)_y) + S^k \succ 0$, we can easily check the feasibility of the solution whenever $\bar{z} < \bar{z}^k$. Notice that sparsity of S^k makes factorizing $Diag(d(\bar{z}^k)_y) + S^k$, and thus checking the positiveness of $X(\bar{z}^k)$, fast and easy.

Using the advantages of the max-cut problem discussed above, we can now slightly modify the dual-scaling algorithm for solving max-cut problems.

Specialized Dual-scaling Algorithm
Given $\bar{z}^0 = C \bullet I$, strictly feasible (y^0, S^0), $\rho > n + \sqrt{n}, \alpha = 0.99$

1. Compute $diag((S^k)^{-1})$ and M^k
2. Find $d(\bar{z}^k)_y$ using (23)
3. Find the new upper bound $\bar{z}$
4. If $\bar{z} < \bar{z}^k$ and $(Diag(d(\bar{z}^k))_y + S^k) \succ 0$
   $(\bar{z}^{k+1} = \bar{z}^k)$, recompute $d(\bar{z}^k)_y$ using (23)
   Else,
   $\bar{z}^{k+1} = \bar{z}^k$
5. Compute $||P(\bar{z}^k)||$; Select $\beta > \frac{\alpha}{||P(\bar{z}^k)||}$ such that
   $y^{k+1} = y^k + \beta d(\bar{z}^{k+1})_y$ and $S^{k+1} = C - Diag(y^{k+1}) \succ 0$
6. Repeat until $\frac{\bar{z}^k - e^t y^k}{|e^t y^k| + 1} < \varepsilon$

4.3 Finding M^k Efficiently

We mentioned before that for special classes of problems we can find Gram matrix M^k efficiently in each iteration. To be more precise, when $A_i = a_i a_i^t$ (i.e they are rank one matrices) we can compute matrix M^k and vector $\mathscr{A}((S^k)^{-1})$ (required for finding $\bigtriangledown \psi(y^k, \bar{z}^k)$; check equation (6)) efficiently without computing $(S^k)^{-1}$. For max-cut problem, for instance, $A_i = e_i e_i^t$.
The first algorithm presented in [2] is as follows:

Algorithm 1

1. Factor $S^k = L^k (L^k)^t$.
2. For i=1 ... m
   (a) Solve $L^k w_i = a_i$
   (b) $\mathscr{A}((S^k)^{-1})_i = w_i^t w_i$; and $M_{ii}^k = (\mathscr{A}((S^k)^{-1})_i)^2$
   (c) For $j = 1 \ldots i-1 : M_{ij}^k = (w_i^t w_j)^2$; end
3. End

This algorithm saves a lot in computational cost, but it needs to store w_j's. As an alternative the following algorithm is proposed.

Algorithm 2

1. Factor $S^k = L^k (L^k)^t$.

2. For i=1 ... m
   (a) Solve $S^k = L^k (L^k)^t w_i = a_i$
   (b) $\mathscr{A}((S^k)^{-1})_i = w_i^t w_i$; and $M_{ii}^k = (\mathscr{A}((S^k)^{-1})_i)^2$
   (c) For $j = i+1 \ldots m : M_{ij}^k = (w_i^t a_j)^2$; end
3. End

This algorithm does not need to store w_j's, however it is more expensive in step 2(a) (since we need one more back-solve).

5 Summary

In this report, we presented the max-cut problem and its formulation. Then we discussed the Goemans-Williamson algorithm for solving the max-cut problem. At the end, we presented dual-scaling algorithm for solving large instances of max-cut SDP relaxations.

References

[1] David Williamson, *Lecture Notes on Approximation Algorithms*, IBM Research Report RC 21273, February 1999.

[2] Steven Benson, Yinyu Ye, and Xiong Zhang, *Solving large-scale sparse semidefinite programs for combinatorial optimization*, SIAM Journal on Optimization, 10(2000), pp. 443-461.

[3] Stephen Wright, *Primal-Dual Interior Point Methods* , SIAM, Philadelphia, 1997.