An Efficient Approximation Algorithm for Max-Cut

Abdullah Al-Malaise Al-Ghamdi
Deptt of IS, FCIT
King Abdul Aziz University,
Jeddah, Saudi Arabia

Pawan Kumar Patel
Deptt of CSE
Indian Institute of Technology,
Kanpur, India

Kunal Gupta
Higher College of Technology,
Muscat,
Oman

ABSTRACT
Significant research effort has been devoted in the study of approximation algorithms for NP-hard problems. Approximation algorithm for Max-Cut problem with performance guarantee of 0.87856 is long known. In this work we study balanced Max-Cut problem. We give a balancing factor β for given α such that the approximate solution is at least 0.87856 times the optimal α-balanced cut and it is itself β-balanced.

General Terms
Approximation Algorithms.

Keywords
Approximation algorithm, Balancing factor in Max-Cut, Graphs Partitioning.

1. INTRODUCTION

The paper will work on solving the NP-hard problem, namely the *MAX-CUT* problem. Good approximation algorithms are known in the literature for this problem. The objective is to adopt an existing approximation algorithm for Max-Cut for a variant of this problem.

A. Max-Cut of undirected edge-weighted graphs

Max-Cut of an undirected edge weighted graph seeks to partition the graph into two sets of vertices so

that the sum of the weights of the edges joining these two sets is maximized.

Several approaches are known for this problem which have different objectives and/or technique:

• Whether an exact solution is desired, an approximation with a performance guarantee is sought or simply a good heuristic method is required.

• Some methods work with the input graph while others reduce the problem to other problem and then solve it.

1) Exact Methods: The naive approach to an exact solution for the Max-Cut problem enumerates the 2^n possible cuts of graph, calculates the weight of each cut and note the maximum weight cut, where *n* is number of vertices in graph.

A heuristic approach can be developed based on the enumeration technique. In order to compute a close to optimum cut(S, $\bar{S}$), we proceed to build S one vertex at a time. We traverse the binary decision tree in which the two branches denote the two options: to include or exclude a given vertex. A heuristic function is evaluated based on the current state to decide which branch to traverse. Such heuristics functions are of two-types: upper bound based and lower bound based. In the former case if the currently known weight is greater then the upper bound for a branch, then that branch is abandoned. In the latter case if the currently known weight is lower than the lower bound of a branch, then that branch is pursued and the current value of the weight is updated. Examples of this approach and a variety of bounds used to limit the search in this way are found in [1], [2].

Another approach involves decomposing the problem into some subproblems. In this approach the problem is solved by removing vertices of lower degree. The characteristics of the rest of the vertices are suitably modified to account for the removed vertices. Subsequently enumeration technique is applied to the smaller graph. Gramm et.al. [3] showed that this approach helps reduce the time complexity to $poly(m).2^{m/3}$, where m is the number of edges.

*2) Approximation methods with performance guarantee:*Relaxing the requirement for an exact solution leads to the study of *p*-approximation algorithms, or polynomial time algorithms that provide a solution that is at least *p* times the optimal value of a maximizing problem. The constant *p* is called the performance guarantee of the algorithm.

Sahni and Gonzales [4] presented a 0.5-approximation algorithm for the max cut problem. Their algorithm iterates through the vertices and assigns each vertex to maximize the partial cut that has resulted from the assignment of the preceding vertices.

Goemans and Williamson [5] were the first to apply semidefinite programming to the solution of Max-Cut problem. Algorithm proposed by Geomans and Williamson that has the performance factor $p = 0.878$ for the problem. Their algorithm uses a randomized rounding of the solution to a non-linear relaxation of the Max-Cut.

3) Reformulation into different problem domain: The Max- Cut problem can be reformulated in numerous ways. One can map Max-Cut problem into an equivalent Max-2-Sat problem. A Max-2-Sat problem takes a set of clauses in

conjunctive normal form each containing at most two literals and asks for the maximum number of clause that can be simultaneously satisfied.

Alber, Gramm and Niedermeier [6] have observed that Max-Cut problem is theoretically equivalent to the Max-2-Sat problem, in practice, random Max-Cut problems reformulated as Max-2-Sat problems are generally harder to solve than random Max-2-Sat problems owing to the special structure of the transformed problem.

4) The variant of Max-Cut: In this thesis we study the variant of the Max-Cut problem where the ratio of the two vertex sets of the cut have a bounded ratio. In this version we are given a parameter $0 < \alpha < 0.5$ such that $\alpha < mm\{ |S| / |\bar{S}| , |\bar{S}| / |S| \}$

2. Goemans and Williamson's solution of MaxCut

Goemans and Williamson gave a randomized approximation algorithm [5] for the Maximum cut problem (MaxCut) and proved that their algorithm gives a cut of size at least 0.875856 times the optimal value. They used vector relaxation and semi-definite programming to solve a quadratic program. They pioneered this technique which was later used in many approximation algorithms. They used randomized rounding technique to compute an integral approximation.

MaxCut of a undirected edge-weighted graph $G = (V, E)$ is defined as to find a set of vertices S such that the sum of the edges between the vertices of S and $\bar{S}$ maximum. The set of edges between the vertices of S and $\bar{S}$ are called a *cut.* The problem is known to NP-hard.

Goemans and Willaimson's algorithm computes a cut with expected weight equal to $(\gamma - \epsilon)$ times the optimal cut weight where

$$\gamma = \frac{2}{\pi} \min_{0 \le \theta \le \pi} \frac{\theta}{1 - cos\theta} \geq .87856$$

and e is any positive scalar. The corresponding semidefinite program can be solved in time polynomial in n (number of vertices) and log $1/\epsilon$ using ellipsoid algorithm [7].

A. The Randomized Approximation algorithm for MaxCut

Given undirected edge-weighted graph with verices $x_1,\ldots, x_n$ and nonnegative weights $w_{ij} = w_{ji}$ for each edge $(i,j) \in E$ the formulation of the problem by the integer quadratic program is as follows.

Maximize

$$\frac{1}{2} \sum_{1 \le i \le j \le n} w_{ij}(1 - y_i y_j)$$

Subject to

$$y_i^2 = 1, x_i \in V ,$$

$$y_i \in \mathbb{Z}, x_i \in V.$$

Here y_i is the indicator variable for vertex x_i which is constrained to either +1 or — 1, indicating whether x_i is in S or in $\bar{S}$. Solving this integer quadratic program is NP-hard so we consider the relaxation of the scalar variables y_i to vectors v_i in an n-dimensional vector space. The products $y_i y_j$ are replaced by the inner-product $v_i \cdot v_j$.

Maximize

$$\frac{1}{2} \sum_{1 \le i \le j \le n} w_{ij}(1 - v_i \cdot v_j)$$

Subject to

$$v_i \cdot v_i = 1, v_i \in V.$$

Observe that if the solution vectors are restricted to a 1-dimensional subspace, then the vector-program reduces to the original quadratic program. Hence the optimal value of the objective function in the vector program is an upper bound for the optimal value of the original program.

A vector program can be converted to a semi-definite program and can be solved to an arbitrary accuracy ϵ in time polynomial in n^2 and $\log(1/\epsilon)$. The optimal solution vectors $a_1,\ldots, a_n$ can not directly be used to approximate the values for y_i's. Goemans and Willamson used a randomized technique to "round" these vectors to a suitable integer in the set {1,-1}. The corresponding semi-definite program is as follows.

Maximize

$$\frac{1}{2} \sum_{1 \le i \le j \le n} w_{ij}(1 - Y_{ij})$$

Subject to

$$Y_{ii} = 1, 1 \le i \le n$$

$$Y \succeq 0$$

Let $\theta_{i\,j}$ denote the angle between the unit vectors a_i and a_j. The contribution of $a_i \cdot a_j$ to the object function increases as the angle approaches π. Hence we would like to place x_i and x_j in different sets of the cut if $\theta_{i\,j}$ is large. In the randomized rounding a random vector r is selected with equal probability in every direction. Then in the approximation solution of the quadratic program y_i is set to 1 if and only if $a_i \cdot r \geq 0$. We repeat this randomized algorithm a certain number of times to improve the probability of getting a large cut.

B. Analysis of Algorithm

In this section we will show that cut computed by the above algorithm is at least 87.856 percent of the maximum cut in the graph with probability at least 1 — 1/e.

Lemma 1: Let the solution of the vector program be $v_i = a_i$ for all i. Then the probability that vertex x_i and x_j are separated is

$\theta_{i\ j}/\pi$ where $\theta_{i\ j}$ is the angle between a_i and a_j.
Proof: Project the randomly generated vector r on the plane formed by a_i and a_j. The two vertices will be separated if and only if the projection passes through the arc between a_i and a_j or that between $-a_i$ and $-a_j$. Since r is uniformly chosen, the probability of the projection intersecting one of the arcs is $2\theta_{ij}/2\pi$. ■

Lemma 2: Let W be the weight of the cut computed by one run of the above algorithm. Then $E[W] > \gamma Opt_v$, where Opt_v is the optimal value of the objective function in the vector program and $\gamma = (2/\Pi)\min_{0 \le \theta \pi} \theta/(1 - \cos\theta)$.

Proof: By the definition of γ, for any $0 \le \theta \le \pi$, $\theta/\pi \ge \gamma(1 - \cos\theta)/2$. Using the result of Lemma 1 we have $E[W] = \sum_{i<j} w_{ij} Pr$ (x_i and x_j are separated) $= \sum_{i<j} w_{ij}\theta_{ij}/\pi \ge \gamma \sum_{i<j} (1/2) w_{ij} (1 - \cos\theta_{ij}) = \gamma Opt_v$.

The optimal value after relaxation opt_v is upper bound of the MaxCut opt_{mc}. So we conclude that $E[W] > \gamma \cdot opt_v \ge \gamma \cdot opt_{mc}$.

Theorem 1: [5] There is a randomized approximation algorithm for MaxCut which, with high probability, achieves an approximation factor of at least $\gamma > 0.87856$.

Proof: Let T denotes the sum of the weights of all edges in graph, define a parameter a such that $E[W] = aT$. Let p denote $P_r[W < (1 - \epsilon)aT]$. Hence $E[W] = aT \le p(1 - \epsilon)aT + (1 - p)T$ for any $\epsilon > 0$. Therefore, $p \le (1 - a)/(1 - a + a\epsilon)$. The unfriendly partition of the graph vertices shows that $T/2 \le opt_{mc}$.

Hence

$$T \ge E[W] = aT \ge \gamma opt_v \ge \gamma opt_{mc} \ge \gamma T/2.$$

Therefore, $1 \ge a \ge \gamma/2$. Plugging the lower bound for a in the inequality for p we get $p \le 1 - (e\gamma/2)/(1 - (1 - \epsilon)\gamma/2) = 1 - c$, where $c = (\epsilon\gamma/2)/(1 - (1 - \epsilon)\gamma/2)$. If W' is the best MaxCut after running the algorithm 1/c times, then $Pr[W' \ge (1 - \epsilon)aT] \ge 1 - (1 - c)^{1/c} \ge 1 - 1/e$. ■

3. EXTENDING THE SOLUTION TO BALANCED MAXCUT

Given a parameter $0 \le \alpha \le 0.5$, an α-cut is a cut $(V_1, V \setminus V_1)$ such that $\alpha |V| \le |V_i| \le (1 - \alpha)|V|$. The Balanced Max Cut (BMC) problem is to find the maximum α-cut, for a given α. In this section we describe a suitable modification to Goemans and Williamson's randomized approximation algorithm to approximate a BMC.This algorithm computes a β-cut which is at least 0.87856 of the maximum α-cut where $\beta \le \alpha$. In particular we show that the proposed algorithm computes a 0.325-balanced cut which is 0.87856 approximation of the maximum bisection-cut (0.5-balanced cut).

Continuing with the Goemans and Williamson formulation, we use variables y_i to define the cut where $y_i=1$ if the vertex x_i belongs to the first set and $y_i = -1$ otherwise. Then we have the following characterization for α-balanced cut.

Lemma 3: A cut is a-balanced cut *iff* $\sum_{ij} y_i y_j \le n^2(1 - 2\alpha)^2$.

Proof: Let the cut be $(V_1, V \setminus V)$ where $|v_1| = n_1 \le |V \setminus V_1| = n - n_1$

so,

$$\sum_{i,j} y_i y_j = n_1^2 + n_2^2 - 2n_1 n_2$$

$$= n_1^2 + (n - n_1)^2 - 2n_1(n - n_1)$$

$$= n_1^2 + n^2 + {n_1}^2 - 2nn_1 - 2nn_1 + 2n_1^2$$

$$= 4n_1^2 - 4nn_1 + n^2$$

$$= (n - 2n_1)^2$$

$$= n^2(1 - 2n_1/n)^2$$

If the cut is a-balanced then,

$$\sum_{i,j} y_i y_j \le n^2(1 - 2\alpha)^2$$

Conversely if $n^2(1 - 2n_1/n)^2 \le n^2(1 - 2\alpha)^2$ then,

$(n_1/n)^2 - n_1/n \le \alpha^2 - \alpha$,

So,

$$\frac{n_1}{n} = \frac{1}{2} - \frac{1}{2}\sqrt{(1 - 2\alpha)^2 - 4\alpha}, for\ some\ \varepsilon \ge 0$$

$$\frac{n_1}{n} \ge \alpha$$

So in any a-balanced cut $\sum_{i\ j} y_i y_j \le n^2(1 - 2\alpha)^2$. This condition captures the balancing requirement. Adding it to the original conditions in the quadratic program proposed by Geomens Williamson we get the following program.

Maximize

$$\frac{1}{2} \sum_{1 \le i \le j \le n} w_{ij}(1 - y_i\, y_j)$$

Subject to

$$y_i^2 = 1, x_i \in V,$$

$$\sum_{i,j} y_i y_j \le n^2(1 - 2\alpha)^2, x_i x_j \in V_{ji}$$

$$y_i \in \mathbb{Z}, x_i \in V.$$

Relaxing the quadratic program into a vector program leads to

Maximize

$$\frac{1}{2}\sum_{1\le i\le j\le n} w_{ij}(1 - v_i\, v_j)$$

Subject to

$$v_i \,.\, v_i = 1, x_i \epsilon V,$$

$$\sum_{i,j} v_i v_j \;\le n^2(1-2\alpha)^2, x_i x_j \in V$$

$$v_i \in \mathbb{Z}, x_i \in V.$$

where v_i are vectors in $\mathbb{R}^n$. The semidefinite program to solve this vector program is as follows.

Maximize $C \bullet Y$

subject to $Di \bullet Y = d_i,\ 1 \le i \le n,$

$$0 \le D_0 \bullet Y \le n^2 (1 - 2\alpha)^2 ,$$
$$Y \succeq 0$$
$$Y \epsilon M_n$$

Here the operator • denotes Hadamard product. The D_0 is the *nxn* matrix, in which all entries is 1 and D_i is also *nxn* matrix where *(i, i)* -entry is 1 and remaining entries are zero. *C* is the weight matrix o f *nxn* where *(i,* j)-entry is —w_{ij}.

We use the randomized relaxation technique proposed by Goemens and Williamson to compute an approximate solution for the balanced MaxCut problem.

A. Analysis of Algorithm

Suppose after randomized relaxation the computed values of y_i are y_i'. Suppose the resulting cut is β-balanced and let $\bar{\beta}$ denote the expected value of *β*. Then E[$\sum_{i\,j} y_i' y_j'$] = n^2 (1-2 $\bar{\beta}$)^2

Let θ_{ij} denote the angle between the vectors v_i and v_j. Then the probability of x_i and x_j getting separated is θ_{ij}/π.

$P\gamma[x_i$ and x_j are separated] = $P\gamma[y_i' y_j' = -1]$ =

$$\begin{cases} \dfrac{\theta_{ij}}{\pi} & if\ y_i' y_j' = -1 \\ 1 - \dfrac{\theta_{ij}}{\pi} & if\ y_i' y_j' = 1 \end{cases}$$

So the expected value of $y_i' y_j'$ is E[$y_i' y_j'$] = 1 — $\frac{2\theta_{ij}}{\pi}$

v_i and v_j are unit vectors so the internal product of these two vectors is $v_i.v_j = cos\theta_{ij}$. By definition of γ we have $\frac{2\theta}{\pi} \le c(1 - cos\theta)$ for every 0 ≤ θ ≤ π. Hence E[$y_i' y_j'$] = $1 - \frac{2\theta_{ij}}{\pi} \le 1 - \gamma(1 - cos\theta_{ij})$

$$\sum_{i,j} E\,[y_i' y_j'] \;\le \sum_{i,j} 1 - \gamma(1 - cos\theta_{ij})$$

$$\le (1-\gamma)n^2 + \sum_{i,j} \gamma(v_i.v_j\,)$$

$$\le (1-\gamma)n^2 + \gamma(1-2\alpha)^2 n^2$$

From the definition of $\bar{\beta}$ we have $n^2(1 — 2\,\bar{\beta})^2 \le (1 — \gamma)n^2 + \gamma(1 — 2\alpha)^2.n^2$. Simplifying this inequality we get

$$(1 - 2\bar{\beta})^2 \le (1 - \gamma) + \gamma(1-2\alpha)^2$$

$$\bar{\beta}^2 - \bar{\beta} \le \gamma\alpha^2 - \gamma\alpha$$

$$\bar{\beta}^2 - \bar{\beta} - \gamma\alpha^2 + \gamma\alpha \le 0$$

$$\frac{1}{2} - \frac{1}{2}\sqrt{1 + 4\gamma\alpha^2 - 4\gamma\alpha} \le \bar{\beta}$$

In particular, if α = 0.5, then $\bar{\beta}$ = 0.325. The analysis for the expected size of the cut computed by this algorithm remains same as in unbalanced case. Hence the expected size of the computed cut is greater than or equal to γ times the maximum feasible cut. Due to the additional condition all feasible cuts are α-cuts. Hence this algorithm computes a cut which is at least 0.87856 times the maximum α-cut.

B. Experimental Results

We performed some experiments on small graphs with various values of α. Following table gives the optimal α-cut, computed cut and the balance of the computed cut b.

Table 1. Result Table

Number of	**A**	**b**	**Optimal α-cut**	**Computed**
10	.2	.3 .5	1287	1041
	.3	.5 .5		
	.4		1490	1381
	.5			
			1514	1512
12	.2	.416	2341	1811
	.3	.416		
	.4	.416	2263	2149
	.5	.416		
			2393	2359

14	.2 .3 .4 .5	.428 .428 .428 .428	2967 2967 2793	2072 2783 3015
16	.2 .3 .4 .5	.5 .437 .5	4110 4004 4110	3049 3466 4004
18	.2 .3 .4 .5	.444 .444 .388	4696 4501 4643	3342 4308 4696
20	.2 .3 .4 .5	.5 .5 .5 .45	5955 5878 5847	3880 4989 5669

4. CONCLUSION

In this work we revisited the Williamson and Geomans' algorithm for MAX-CUT problem that used semidefinite programming to give a performance guarantee of .878. We presented a modification of their algorithm in which for a given α, we compute a cut which is at leats 0.87856 of the optimum α-balanced cut and itself is a $\frac{1}{2} - \frac{1}{2}\sqrt{1 + 4\gamma\alpha^2 - 4\gamma\alpha} \leq \bar{\beta}$ balance cut.

5. REFERENCES

[1] H. Shen and H. Zhang, "Improving exact algorithms for max-2-sat," Annals of Mathematics and Artificial Intelligence, vol. 44, April 2004.

[2] "Study of lower bound functions for max-2-sat," American Association for Artificial Intelligence, April 2004.

[3] J. Gramm, E. A. Hirsch, R. Niedermeier, and P. Rossmanith, "New worst-case upper bounds for max-2-sat with application to max-cut," ECCC, 2000.

[4] S. Sahni and T. Gonzalez, "P-complete approximation problems," Journal of The ACM, pp. 555–565, 1976.

[5] M. X. Goemans and D. Williamson, "Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming," Journal of the ACM, vol. 42, pp. 1115–1145, 1995.

[6] J. Alber, J. Gramm, and R. Niedermeier, "Faster exact algorithms for hard problems: A parameterized point of view," Discrete Mathematics, vol. 229, pp. 3–27, 2000.

[7] M. Grtschel, L. Lovsz, and A. Schrijver, "The ellipsoid method and its consequences in combinatorial optimization," Combinatorica, pp. 169– 197, 1981.