Semidefinite Programming for Combinatorial Optimization

David P. Williamson
IBM T.J. Watson Research Center

SIAM Optimization Prize Presentation
May 11, 1999

$\mathcal{S}.\mathcal{D}.\mathcal{G}.$

Michel X. Goemans and David P. Williamson, "Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems Using Semidefinite Programming," *Journal of the ACM*, Vol. 42, pages 1115–1145, November 1995.

See also the survey:
Michel X. Goemans, "Semidefinite Programming and Combinatorial Optimization," *Mathematical Programming*, Vol. 79, pages 143–161, 1997.

Available from:
`www.research.ibm.com/people/w/williamson`.

Outline

I. What is the maximum cut problem?

II. What is an approximation algorithm?

III. What is semidefinite programming?

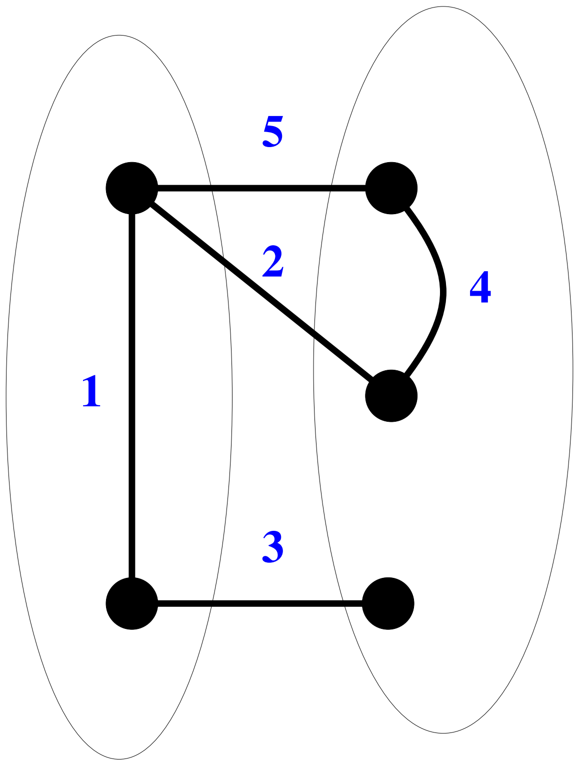
IV. How do these all relate?

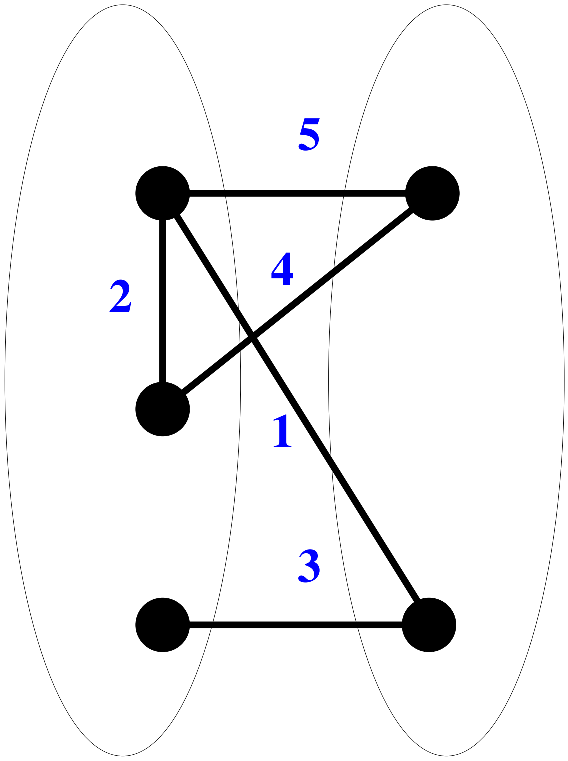
V. Where does this lead?

The maximum cut problem

Input: A graph $G = (V, E)$ with rational weights w_{ij} for $(i, j) \in E$. Assume $w_{ij} = 0$ if $(i, j) \notin E$.

Goal: Divide the nodes into two parts so as to maximize the weight of the edges whose nodes are in different parts (edges *in the cut*).





The maximum cut problem (MAX CUT) is NP-hard, even when $w_{ij} \in \{0, 1\}$; poly-time solvable if the graph is planar.

Why do we care?

The maximum cut problem arises in several different domains:

- Finding the ground state of magnetic particles subject to a field at 0K in the Ising spin glass model.
- Minimizing the number of vias (holes) drilled in a two-sided circuit board.
- Solving network design problems.
- Finding "radius of nonsingularity" of matrices.

(Barahona, Grötschel, Jünger, Reinelt 1988)
(Barahona 1996)
(Poljak and Rohn 1993)

Solution approaches

Various approaches to solving the problem:

- Integer/linear programming enumerative techniques (e.g. branch and cut)
- Heuristics
- *approximation algorithms*

Approximation algorithms

An α-approximation algorithm for an NP-hard optimization problem:

- runs in polynomial time
- returns a solution with value within a factor α of an optimal solution

Can consider randomized versions, in which expected value of solution is within a factor α of optimal.

A $\frac{1}{2}$-approximation algorithm by Sahni and Gonzalez (1976) was the best known for MAX CUT until the .878-approximation algorithm of this paper.

Consider the following model of MAX CUT:

$$\text{Max} \quad \frac{1}{2} \sum_{i<j} w_{ij}(1 - y_i y_j)$$
$$y_i \in \{-1, 1\}$$

Then

$$\begin{aligned}
&\frac{1}{2} \sum_{i<j} w_{ij}(1 - y_i y_j) \\
&\quad = \frac{1}{2} \sum_{i<j: y_i = y_j} w_{ij}(1 - y_i y_j) + \frac{1}{2} \sum_{i<j: y_i \neq y_j} w_{ij}(1 - y_i y_j) \\
&\quad = \frac{1}{2} \sum_{i<j: y_i \neq y_j} w_{ij}(1 - y_i y_j) \\
&\quad = \sum_{i<j: y_i \neq y_j} w_{ij} \qquad \text{is weight of the cut.}
\end{aligned}$$

A brief review

Definition: A matrix $A \in \Re^{n \times n}$ is positive semidefinite iff $x^T A x \geq 0$ for all $x \in \Re^n$. We say $A \succeq 0$.

The following are equivalent for a symmetric A:

1. $A \succeq 0$;
2. A has only non-negative eigenvalues;
3. $A = B^T B$ for some matrix $B \in \Re^{m \times n}$.

Then following model is equivalent:

$$
\begin{aligned}
\text{Max} \quad & \frac{1}{2} \sum_{i<j} w_{ij}(1 - x_{ij}) \\
& X = yy^T = (y_i y_j) \\
& y_i \in \{-1, 1\} \\
& x_{ii} = 1 \\
& X \succeq 0 \qquad\qquad\qquad\qquad X \text{ symm}
\end{aligned}
$$

What is semidefinite programming?

If we drop the constraints involving y, we obtain a semidefinite program which is an upper bound on MAX CUT.

$$
\begin{array}{lll}
\text{Max/Min} & \displaystyle\sum_{i,j} c_{ij} x_{ij} & \\
& \displaystyle\sum_{i,j} a_{ijk} x_{ij} = b_k & \forall k \\
& X = (x_{ij}) & \text{symm, psd}
\end{array}
$$

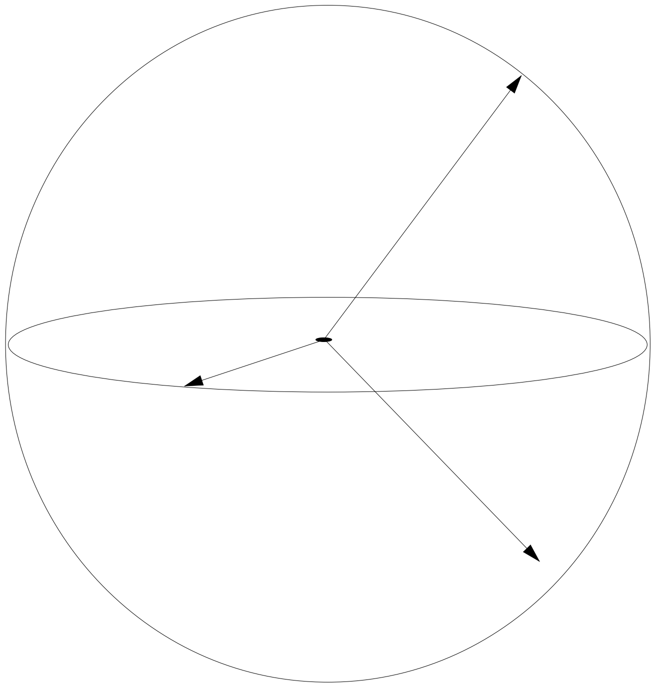
Semidefinite programs (SDP) are solvable (to within some small error) in polynomial time.

(Grötschel, Lovász, Schrijver '88)
(Alizadeh '95)
(Nesterov & Nemirovskii '94).

A geometric interpretation

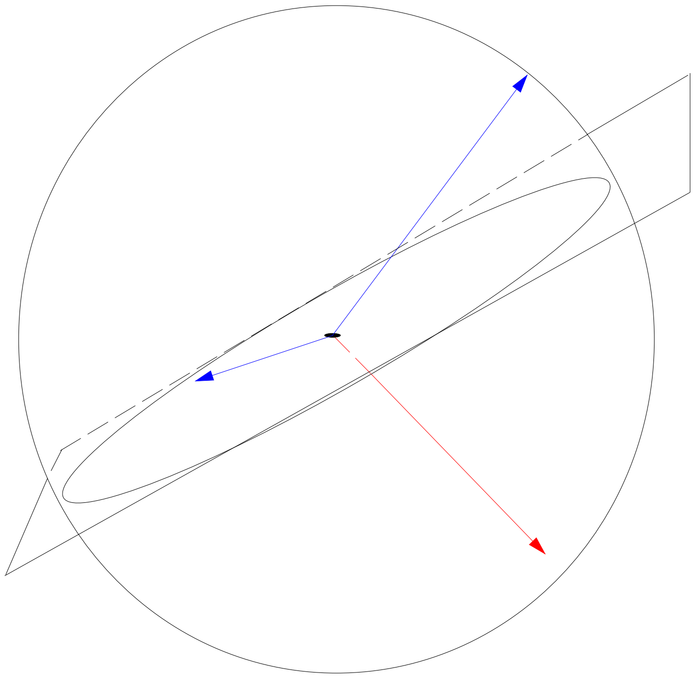
Using the fact that $X = V^T V$ for some V, get a geometric interpretation with v_i the ith column vector of V. Then can view soln. as n unit vectors.

$$\begin{array}{lll} \text{Max} & \displaystyle\frac{1}{2}\sum_{i<j} w_{ij}(1 - v_i^T v_j) & \\ & v_i^T v_i = 1 & \forall i \\ & v_i \in \Re^n & \end{array}$$



How does this all fit together?

To get a solution for MAX CUT, take optimal vectors, choose a vector r uniformly at random and let this be the normal to a plane through the origin; this cuts the sphere in half.

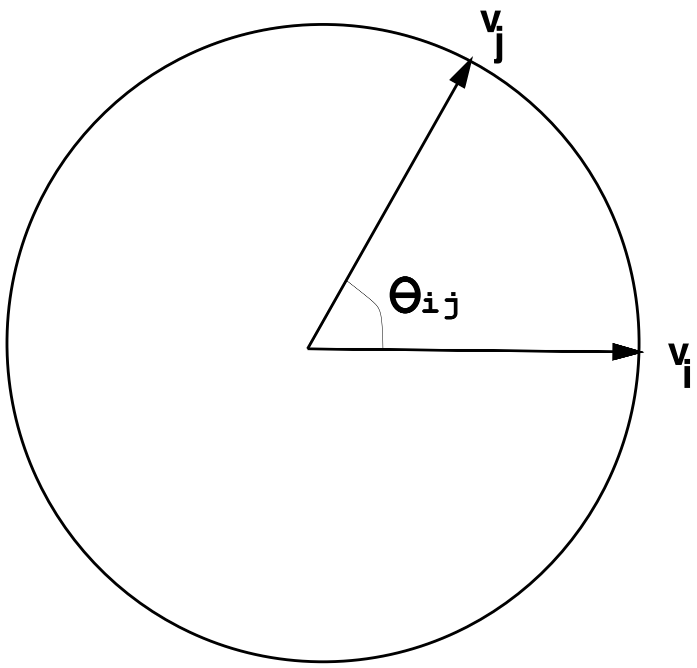


This naturally partitions the vectors (and thus the nodes) into two parts.

Lemma 1

Lemma 1:

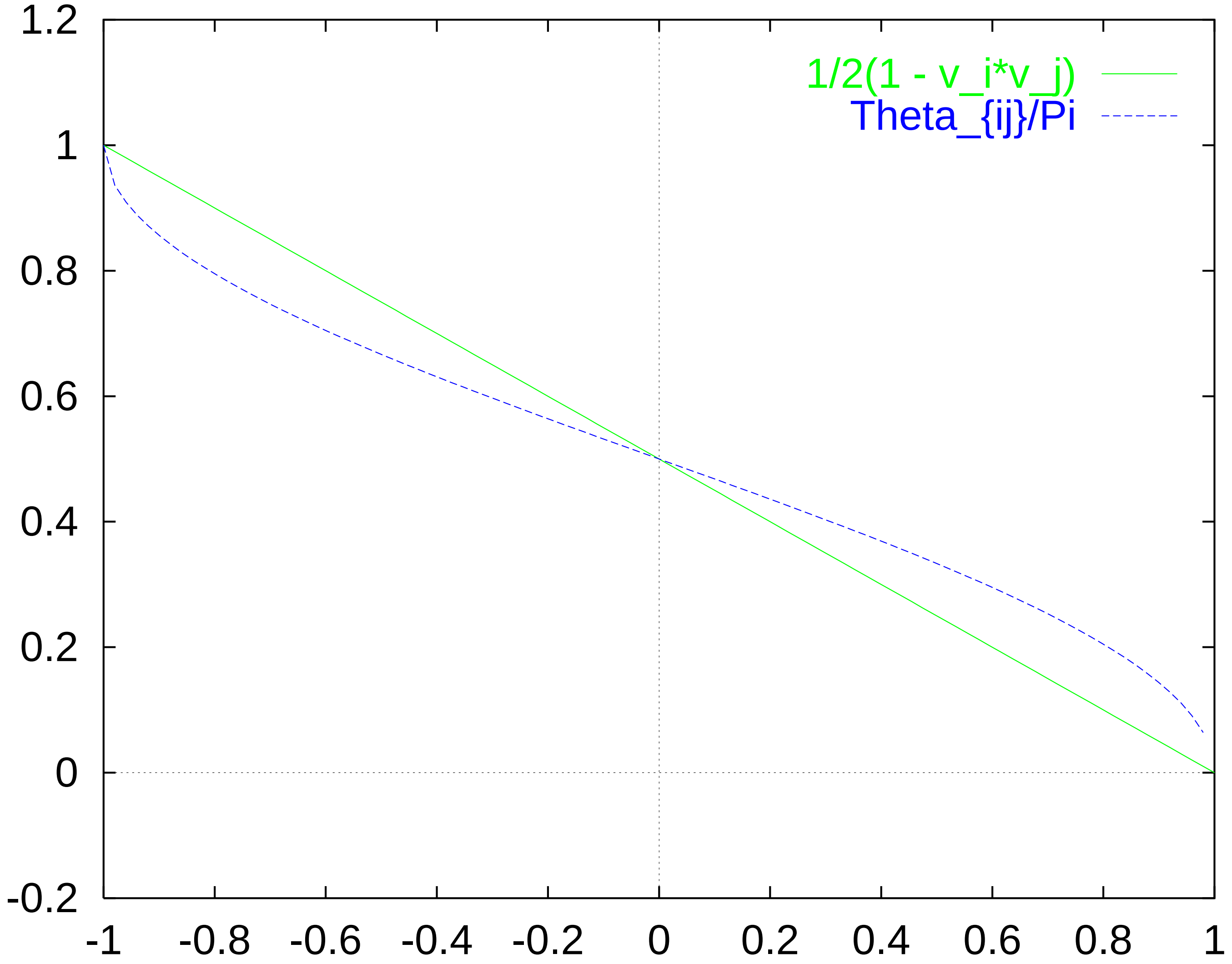
$$\Pr[\text{Edge } (i,j) \text{ in the cut}] = \frac{1}{\pi}\theta_{ij} = \frac{1}{\pi}\arccos(v_i^T v_j)$$



So the expected value of the solution is

$$\sum_{i<j} w_{ij}\frac{1}{\pi}\arccos(v_i^T v_j).$$

Lemma 2



At worst,

$$\frac{\frac{1}{\pi}\arccos(v_i^T v_j)}{\frac{1}{2}(1 - v_i^T v_j)} \geq .878$$

$$
\begin{aligned}
\sum_{i<j} w_{ij} \cdot \frac{1}{\pi} \arccos(v_i^T v_j) &\geq .878 \cdot \frac{1}{2} \sum_{i<j} w_{ij}(1 - v_i^T v_j) \\
&\geq .878 \cdot OPT,
\end{aligned}
$$

or no more than 14% away from optimal.

(Goemans, Williamson '95)

Can we do better?

Delorme and Poljak '93 give a graph for which the ratio of optimum cut to SDP value is .884.

Karloff '96 gives instances for which the ratio of the expected value to the optimum is exactly

$$\min \frac{\frac{1}{\pi}\arccos(v_i^T v_j)}{\frac{1}{2}(1 - v_i^T v_j)},$$

even if any valid inequality is added to the SDP.

Håstad '97, Trevisan, Sorkin, Sudan, Williamson '96 have shown that if a .94-approximation algorithm exists, then a polynomial-time algorithm for MAX CUT exists.

Where did this all come from?

The use of SDP was inspired by a paper of Lovász and Schrijver (1990), who considered linear programming relaxations of 0-1 integer programming problems.

Idea: Multiply each inequality $\sum_j a_{kj} y_j - b_k \geq 0$ by $1 - y_i \geq 0$ or $y_i \geq 0$. For example,

$$(1 - y_i) \left(\sum_j a_{kj} y_j - b_k \right) \geq 0.$$

As before, replace $y_i y_j$ with x_{ij} and impose constraint $X \succeq 0$, $x_{ii} = y_i$.

This was shown to give a strong relaxation for stable set problem; also, n repetitions of gives convex hull of any 0-1 problem.

Where has it led?

One direction of research: further exploration of computational use of SDP for max cut problems.

- Goemans & Williamson '95: random and Euclidean graphs, up to 200 nodes (roughly a CPU hour). Solns. within 4-5% of bounds.

- Homer & Peinado '95: parallelized version of gradient descent on CM5, IBM SP2. Up to 8,000 nodes (roughly a CPU hour). GW + local search usually outperforms simulated annealing.

- Berry & Goldberg '95: compares GW solns with simulated annealing, Kernighan-Lin on random graphs, finds GW inferior in quality.

- Helmberg & Rendl '95: SDP and cutting planes in a branch and bound setting. Up to 450 nodes. Optimal solns. on 100 nodes, 3 CPU hours.

- Benson, Ye, and Zhang '97: Dual scaling algorithm. Up to 3000 nodes, up to 3 CPU hours on a PC.

- Burer & Monteiro '98: Projected gradient. Up to 3000 nodes, up to 1 CPU hour. Usually within 6-7% of bounds.

Experimental summary

- The algorithm is better than its worst-case guarantee: usually within 6% of optimum.
- It usually beats local search techniques if a local min is found after the algorithm is run.
- Algorithms are gradually giving reasonable running times on large instances.
- CAVEAT: All these tests on random graphs.

Where has it led?

Additionally, these techniques have led to numerous improved approximation algorithms for combinatorial optimization problems.

- Goemans & Williamson '95, Asano, Ono, Hirata '96, Asano '97, Asano & Williamson '99 give improved approx. algs. for MAX SAT.
- Feige & Goemans '94: .931-approx. alg. for MAX 2SAT.
- Frieze & Jerrum '94: improved approx. alg. for MAX k-CUT.
- Chor & Sudan '95: improved approx. algs. for the "betweeness" problem.
- Karger, Motwani, and Sudan '95: improved approx. alg. for coloring k-colorable graphs.

- Nesterov '97, Ye '97,'98, Zhang '98, others: approx. algs. for various quadratic programming problems.

- Karloff & Zwick '97: $\frac{7}{8}$-approx. alg. for MAX 3SAT.

- Halperin & Zwick '98: .872-approx. alg. for MAX 4SAT.

- Zwick '98: $\frac{1}{2}$-approx. alg. for MAX 3CSP.

- Skutella '98: improved approx. alg. for scheduling.

- Blum, Konjevod, Ravi, and Vempala '98: $O(\sqrt{n}\log n)$-approx. alg. for the minimum bandwidth problem.

- Ye '99: .699-approx. alg. for MAX BISECTION.

Concluding thoughts

- Continuous optimization can be useful for solving discrete problems.

- Convex programs can give provably stronger bounds on discrete problems than traditionally used linear programs.

- Semidefinite programming may play the role in enumerative techniques that linear programming does now.