Lecture 6: Approximation Algorithms

Angsheng Li

BeiHang University

29, Oct., 2019

Advanced Algorithms

1. Definition of algorithms?
   How many definitions you can give? Traditional five principles: input, output, finiteness, determinism, mechanism
2. What are the roles of algorithms?
   - The engine of computer
3. What is advanced? approximately after 1990
   - Randomized, after 1980
   - Approximation, after 1990
   - Local algorithms, after 2000
   - Dynamical algorithms, after 2000
   - Streaming algorithms, after 2000
   - Distributed algorithms, revisit
4. New direction?
   Data structure and algorithms of big data - New Science

The goal of the course

1. Better understanding of the art
2. The key of algorithms?
   - New ideas
   - New applications
   - New understanding of the concept of computation
     - Computation vs Game
     - Computing vs Learning
     - Computation vs Evolution
     - Computation vs Information
3. The lectures: good science

Secrecies of Algorithm

1. Algorithms are naturally linked to structure
2. Algorithmic ideas come from new applications, new concepts and hard problem
3. Breakthrough in a wide TCS areas usually comes from algorithmic new ideas
4. Proofs of new algorithms need new mathematics

Outline - Approximation

1. Why approximation?
2. Approximation, combinatorial approach
3. Linear thinking
4. Provable approximation via linear programming (LP)
5. Semidefinite programs (SDPs) approach to approximation algorithms

Approximation

1. Why approximation? $\min$, $\max$
2. What is approximation?
3. How to approximate?
   - Linear thinking
   - Provable approximation via linear programming
   - Semidefinite programs (SDPs) and approximation algorithms
   - Duality
   - Combinatorial approach
   - Probabilistic approximation
4. The limit of approximation - the great achievement after NP completeness, 1998
5. The role of approximation - theory and practice, and future

Definition

For a maximum problem *P*, for $\alpha \leq 1$, an α-approximation algorithm is a polynomial time algorithm, finding, for any instance *x*, a solution achieving

$$\geq \alpha \cdot \mathrm{OPT}.$$

For minimum problem *P*, for for $\alpha \geq 1$, the algorithm finds a solution with

$$\leq \alpha \cdot \mathrm{OPT}$$

for every instance.

Cardinality vertex cover

Vertex cover (VC) Given a graph $G = (V, E)$, find a set $S \subset V$ of minimum size that *covers* V, meaning that every edge must link to a vertex $s \in S$.

Algorithm

1) Find a maximal matching M of G
2) Let S be the set of all the vertices incident to M.

Then:

(1) S is a vertex cover
(2) $|M| \leq \mathrm{OPT}$, and $|S| \leq 2 \cdot |M|$.

Proof

Proof.

For (1). Towards a contradiction, if there is an edge e fails to link to any vertex $s \in S$, then $M + e$ is a matching. But M is a maximal matching.

For (2). $|M| \leq \mathrm{OPT}$, and $|S| \leq 2 \cdot |M|$.

Let X be a vertex cover. For every edge $e \in M$, e links to a vertex $x \in X$, since X is a vertex cover. By the assumption, M is a matching, different edges in M must link to different vertices in X. So $|M| \leq |X|$. This gives $|M| \leq \mathrm{OPT}$.

Clearly, $|S| \leq 2 \cdot |M|$. Therefore,

$$|S| \leq 2 \cdot |M| \leq 2 \cdot \mathrm{OPT}.$$

This gives a 2-approximation algorithm for the vertex cover problem. □

Metric Steiner tree

1. (Steiner tree)(ST) Given an undirected graph $G = (V, E)$ with nonnegative edge costs and whose vertices are partitioned into two sets, the **required** R and the **Steiner** S, find a minimum cost tree T in G that contains the required vertices R and any subset of S, meaning two vertices in R are connected in T.
2. The **Metric Steiner tree** problem:
   - Complete graph G
   - The edge costs c satisfy the *triangle inequality*: For any vertices x, y, z,

$$\mathrm{cost}(x, y) \leq \mathrm{cost}(x, z) + \mathrm{cost}(z, y). \tag{1}$$

   Find the minimum cost tree that contains the required vertices R.

Reducing ST to metric ST

Theorem

There is an approximation factor preserving reduction from the Steiner tree problem to the metric Steiner tree problem.

Proof.

Given a Steiner tree problem $G = (V, E)$ with required set R, let G' be the complete graph on the vertex set V. For every $u, v \in V$, let $\mathrm{cost}(u, v) = d^G(u, v)$, where $d^G(u, v)$ is the distance from u to v in G. The required set $R' = R$. Then G' is a metric Steiner tree instance.

$$\mathrm{OPT}_{G'} \leq \mathrm{OPT}_G \tag{2}$$

$$\mathrm{OPT}_G \leq \mathrm{OPT}_{G'} \tag{3}$$

□

Proof

Proof.

For (2). For any edge $(x, y) \in E$, its cost in G' is no more than its cost in G. Therefore, the cost of an optimum solution in G' does not exceed the cost of an optimum solution in G.

For (3). Given a Steiner tree T' in G', for every two vertices x, y, replace the edge (x, y) by a shortest path from x to y in G. This constructs a subgraph H of G. Then for any $u, v \in R$, u and v are connected in H. Let T be a tree of H. Then the cost of T is at most the cost of T'. □

Steiner tree is a sub-problem of metric Steiner tree problem

Steiner tree problem is reduced to a metric Steiner tree problem.

Minimum spanning tree approach to metric Steiner tree

- Minimum spanning tree, written **MST**, is in **P**
  There is no requirement of the required set R of vertices!
- Metric Steiner tree is **NP** hard

However, minimum spanning tree is a good approximation for the metric Steiner tree.

Approximation algorithm for the metric Steiner tree based on MST

Let R be the set of **required** vertices. A minimum spanning tree on R is a feasible solution for the metric Steiner tree problem. And this feasible solution is a good approximation.
The **Algorithm** is to simply find a minimum spanning tree T on R
The induced subgraph by the vertices R, or written by G_R

Proof - 1

Theorem

The cost of the algorithm, i.e., minimum spanning tree on R, is at most $2 \cdot \mathrm{OPT}$*, where* OPT *is the cost of the metric Steiner tree. This gives a factor* 2 *approximation algorithm for the metric Steiner tree problem.*

Proof.

1) Let T be a Steiner tree of cost OPT.
2) Double the edges of T to get T',
3) Find an Eulerian graph connecting all the required vertices R from T'

Euler cycle: A closed trail containing all edges. Every vertex has even degree.

□

Proof - 2

Proof.

4) Obtain a Hamiltonian cycle on the required vertices R using the Euler tour, and short-cutting the Steiner vertices and visited vertices, this gives a Hamiltonian cycle C
Using triangle inequality here
Hamilton cycle: Travels the vertices
5) delete one edge from C to get a path P.
Then:

- P is a spanning tree on the required set R
- the cost of P is at most

$$2 \cdot \mathrm{OPT}$$

The minimum spanning tree MST on G_R is at most the cost of P, which is at most $2 \cdot \mathrm{OPT}$.

Multiway cut

Given $G = (V, E)$, edge weights $w : E \to R^+$, a set $S = \{s_1, s_2, \cdots, s_k\} \subset V$, find a minimum weight set of edges whose removal disconnects all the vertices in S.

An approximation algorithm

Algorithm

1) For each i, find a set C_i of edges with minimum weight that isolates s_i

Let t_i be a new vertex by merging the vertices $S \setminus \{s_i\}$ into one.

C_i be the set of edges disconnect s_i and t_i.

This is a max flow/min cut strategy.

2) Discard the cut of the largest weight.

3) Output the union of the remaining cuts

Approximation $2(1-\frac{1}{k})$

Let A be the optimum cut of G. Remove A, then G becomes k connected component.
Let $A_i \subset A$ be the set isolating s_i.
Then

$$\sum_{i=1}^{k} w(A_i) = 2w(A).$$

By the algorithm for each i, $w(C_i) \leq w(A_i)$.
Therefore

$$w(C) \leq (1-\frac{1}{k})\sum_{i=1}^{k} w(C_i) \leq 2(1-\frac{1}{k})w(A).$$

k-Center

1) Given a set of cities, with intercity distances specified, pick k cities for locating warehouses in so as to minimize the maximum distance of a city from its closest warehouse.
2) In a network, find k vertices at which we set virus controller such that whenever there is a virus spreading in the network, it is always there is a controller that quickly detects the virus. (Open)

Definition

(Metric k-center) Let $G = (V, E)$ be a complete undirected graph with edge costs satisfying the triangle inequality, and k be a natural number. For any set $S \subseteq V$ and vertex $v \in V$, define $\mathrm{connect}_G(S; v)$ to be the cost of the cheapest edge from v to a vertex in S. The problem is to find a set S of size k so as to minimize

$$\max_{v}\{\mathrm{connect}_G(S; v)\}.$$

Parametric pruning for metric k-center

1. Sort the edges of G in nondecreasing order of cost, i.e.,

$$\mathrm{cost}(e_1) \le \mathrm{cost}(e_2) \le \cdots \le \mathrm{cost}(e_m).$$

2. Let $G_i = (V, E_i)$, where $E_i = \{e_1, \cdots, e_i\}$.
3. Find the least i such that G_i has a dominating set of size at most k.
   Let i^* be such an i. Then $\mathrm{cost}(e_{i^*})$ is the cost of the optimal k-center.

Square of graph

Define the square of graph H to be the graph containing an edge (u, v) if there is a path at most 2 from u to v in H. Written H^2.

Lemma

Given a graph H, let I be an independent set in H^2. Then

$$|I| \leq \mathrm{dom}(H), \tag{4}$$

where $\mathrm{dom}(H)$ is the minimum dominating set of H.

Proof

Proof.

Let D be a minimum dominating set in H.Then H contains $|D|$ stars spanning all vertices. Since each of these stars will be a clique in H^2, H^2 contains $|D|$ cliques spanning all vertices. Clearly, I can pick at most one vertex from each clique, and the lemma follows. $\square$

Approximation algorithm for metric k-center

1. Construct $G_1^2, G_2^2, \cdots, G_m^2$
2. Compute a maximal independent set M_i, in each G_i^2
3. Find the least i such that $|M_i| \leq k$, j say.
4. Output M_j.

Proof

Lemma

For the j found by the algorithm, $\mathrm{cost}(e_j) \le \mathrm{OPT}$*.*

Proof.

For every $i < j$, $|M_i| > k$, so $j \le i^*$. □

Approximation 2

Theorem

The algorithm is an approximation 2 algorithm.

Proof.

A maximal independent set, I say, is also a dominating set. Thus there exist stars in G_j^2, centered on the vertices in M_j, covering all vertices. By the triangle inequality, each edge used in constructing these stars has cost at most $2 \cdot \mathrm{cost}(e_j)$. The theorem follows. $\square$

More combinatorial approach

- Many problems have approximation by this approach
- The most successful result is the PTAS for Euclidean TSP
- No theoretical foundation yet

Linear vs nonlinear

1. Linear systems of equations is easy to solve in polynomial time
2. Non-linear systems of equations is NP-hard

Vertex set cover -linear

Given a graph $G = (V, E)$, find a vertex cover set $S \subset V$. For each i, define $x_i = 1$ if $i \in S$, and 0 otherwise.
Linear:

$$\min \sum_{i=1}^{n} x_i$$

Subject to:
(1) For each i, $x_i = 0$ or 1,
(2) For each edge $\{i, j\}$, $x_i + x_j \geq 1$.

Vertex set cover -nonlinear

Nonlinear:
The system of the equations:

$$\min \sum_{i=1}^{n} x_i^2$$

Subject to:
(1) For every $i \in V$

$$x_i(1 - x_i) = 0$$

(2) For every edge $\{i, j\}$ of G

$$(1 - x_i)(1 - x_j) = 0$$

Idea

- Algebraic approach
- Linear approach

Solving systems of linear equations

Given an $m \times n$ coefficient matrix A and a vector b, then the following are equivalent

1. the linear system $Ax = b$ is feasible
2. b is in the span of the column vectors of A
3. $\mathrm{rank}(A) = \mathrm{rank}(A, b)$
   (A, b) is the matrix of A with a new last column b.

Time complexity: $O(n^3)$

Systems of linear inequalities

A space R is called convex, if for every $x, y \in R$,

$$\lambda \cdot x + (1 - \lambda) \cdot y \in R$$

for all λ with $0 \leq \lambda \leq 1$.

A *linear programming* (LP) is to solve the problem of the following form:

$$\min c^T \cdot x$$

$$A \cdot x \geq b$$

How to solve the LPs?

Basic algorithms

1) Naive way
2) Simplex method
Khachiyan, 1979, the first polynomial time algorithm
Applications of the linear programming, 1939, former Soviet Union, programming economics,
Nobel economics award

Why linearity?

Taylor expansion:
For a well-defined function f

$$f(x_1, x_2, \cdots, x_n) = f(0, 0, \cdots, 0) + \sum_i x_i \frac{\partial f}{\partial x_i}(0) + \sum_{i_1, i_2} x_{i_1} x_{i_2} \frac{\partial f}{\partial x_{i_1} \partial x_{i_2}}(0, 0) + \cdots \tag{5}$$

Poly time

Gaussian elimination is a polynomial time procedure.
Prove it!

Provable approximation by LP

I Most NP-hard optimization problems involve finding 0/1 solutions
II Using LP, we can find a fractional solution
A general approach of approximation algorithms

Deterministic rounding

- $< \frac{1}{2}$ — 0
- $\geq \frac{1}{2}$ — 1

Weighted vertex cover Given $G = (V, E)$ with weight $w : V \to R^+$.

The goal is to find a vertex cover with minimum weights.

The LP relaxation of weighted vertex cover

$$\min \sum_{i=1}^{n} w_i x_i$$

$$\text{Subject to} : 0 \le x_i \le 1, \forall i \in V$$

and

$$x_i + x_j \ge 1, \forall \{i, j\} \in E$$

Let OPT_f be the optimum value of the LP.
Set

$$S = \{i \mid x_i \ge \frac{1}{2}\}$$

Proofs

(1) S is a vertex cover of V

For $\{i,j\} \in E$, $x_i + x_j \geq 1$, so one of the x_i and $x_j \geq \frac{1}{2}$, $i \in S$ or $j \in S$.

(2) $w(S) \leq 2 \cdot \mathrm{OPT}_f \leq 2 \cdot \mathrm{OPT}$.

$$\begin{aligned} \mathrm{OPT}_f &= \sum_i w_i \cdot x_i = \sum_{i \in S} w_i x_i + \sum_{i \notin S} w_i x_i \\ &\geq \frac{1}{2} \sum_{i \in S} w_i + \sum_{i \notin S} w_i x_i \\ &= \frac{1}{2} w(S) + \sum_{i \notin S} w_i x_i \end{aligned} \tag{6}$$

Hence

$$\frac{1}{2} w(S) \leq \mathrm{OPT}_f \leq \mathrm{OPT}.$$

Randomized and derandomization

Let ϕ be a CNF formula with Boolean variables $x_1, x_2, \cdots, x_n$ and clauses $c_1, c_2, \cdots, c_m$.

Random assignment σ: For each x_i, assign $x_i = 1$ with probability $\frac{1}{2}$.

For each clause c_j, define X_j to be 1 if c_j is satisfied, and 0 otherwise.

Let $X = \sum_{j=1}^{m} X_j$.

For k, define $\alpha_k = 1 - 2^{-k}$. For each $k \geq 1$, $\alpha_k \geq \frac{1}{2}$.

Lemma For clause c_j with k literals, $E[X_j] = \alpha_k$.

Then $E[X] = \sum_{j=1}^{m} E[X_j]$.

Therefore, there is an assignment that satisfies at least $E[X]$ clauses of ϕ.

Derandomization - 1

Design a deterministic algorithm to find an assignment that satisfies at least $E[X]$ clauses of ϕ.

Notice that given a CNF formula $\phi : c_1, c_2, \cdots, c_m$, where each c_j is a cause of k_j literals. Define

$$X_j = \begin{cases} 1, & \text{if } c_j \text{ is satisfied,} \\ 0, & \text{otherwise.} \end{cases} \tag{7}$$

$$X = \sum_{j=1}^{m} X_j. \tag{8}$$

$$N(\phi) = E[X], \tag{9}$$

where $E[X]$ is the expectation of X.

Derandomization - 2

$N(\phi)$ can be computed only by counting the number of literals in each of the causes of ϕ.
Suppose that

$$x_1, x_2, \cdots, x_n$$

is a fixed ordering of all the propositional variables appeared in ϕ.
Consider the assignment of x_1.
We consider two cases:
Case 1. $x_1 = 0$.
Let N_0 be the number of causes of ϕ in which $\neg x_1$ occurs, and let ϕ_0 be the formula obtained from ϕ by deleting the causes that are already satisfied, and by deleting x_1 from each of the causes in which x_1 appears.
Let $M_0 = N_0 + N(\phi_0)$.

Derandomization - 3

Case 2. $x_1 = 1$.
Let N_1 be the number of causes of ϕ in which x_1 occurs, and let ϕ_1 be the formula obtained from ϕ by deleting the causes that are already satisfied, and by deleting $\neg x_1$ from each of the causes in which $\neg x_1$ appears.
Let $M_1 = N_1 + N(\phi_1)$.
If $M_0 > M_1$, then set $x_1 = 0$, otherwise, then set $x_1 = 1$.
Continue the procedure, we find an assignment $\sigma = a_1, a_2, \cdots, a_n$ for the list of variables $x_1, x_2, \cdots, x_n$.
Clearly, the assignment σ satisfies at least $N(\phi)$ causes of ϕ.
Notice that $N(\phi) \geq \frac{m}{2}$.
The method of finding the assignment σ above is called a self-reducibility method, which is an important algorithmic method in many problems.

Max 2SAT

A 2 CNF formula consists of:
n Boolean variables $x_1, x_2, \cdots, x_n$ and m clauses of the form $y \vee z$, where y, z are literals, i.e., a variable or its negation.
MAX 2SAT: Given a 2CNF formula, find an assignment to satisfy the maximum number of clauses.
It is interesting to note that:

- 2CNF is in P
- MAX2SAT is NP-hard

LP for MAX2SAT

Given a 2CNF ϕ, let J be the set of clauses and y_{j_1}, y_{j_2} be the two literals of the j-th clause. Let ϕ be a 2CNF of n variables $x_1, \cdots, x_n$ and m clauses each of which contains at most two literals. For each clause C_j, we introduce a variable z_j such that if C_j is satisfied, then $z_j \geq 1$, and 0, otherwise.
For a variable y, the value of an occurrence of y is y, and the value of an occurrence of $\neg y$ is $1 - y$.
The LP relaxation is:

$$\max \sum_{j \in J} z_j$$

$$0 \leq x_i \leq 1 : \forall x_i$$

$$y_{j_1} + y_{j_2} \geq z_j : \forall C_j$$

Randomized rounding: Max 2SAT

Suppose that $\{r_i\}$ is the fractional solution of the variables x_i of the LP.
With probability r_i, set

$$x_i = 1$$

Then $E[x_i] = r_i$.
Let N be the number of the clauses satisfied by the the random rounding of the LP solution. N is a random variable.
Let $N(\phi) = E[N]$.

Proofs - I

We show that

$$E[N] \geq \frac{3}{4} \cdot \mathrm{OPT}_f \geq \frac{3}{4} \cdot \mathrm{OPT}.$$

It suffices to prove that for each clause C_j, the probability that C_j is satisfied is at least $\frac{3}{4} \cdot z_j$.

Case 1 $C_j = x_r$

By the definition of the rounding, the probability that C_j is satisfied is x_r. By constraints, this is $\geq z_j$.

Proofs - II

Case 2. Let $C_j = x_r \vee x_s$. Then $x_r + x_s \geq z_j$.
The probability that C_j is satisfied is

$$p_j = 1 - (1 - x_r) \cdot (1 - x_s) = x_r + x_s - x_r x_s$$

Using $4x_r x_s \leq (x_r + x_s)^2$, we have

$$\begin{aligned} p_j &\geq x_r + x_s - \frac{1}{4}(x_r + x_s)^2 \\ &\geq z_j - \frac{1}{4}z_j^2 \geq \frac{3}{4}z_j. \end{aligned} \tag{10}$$

Note: Using quadratic function $x - \frac{1}{4}x^2 = 0$, the roots are 0 and 4. Consider the curve of $f(x) = x - \frac{1}{4}x^2$, when $0 \leq x \leq 1$, the curve is above $\frac{3}{4}x$.

A $\frac{3}{4}$ approximation algorithm for MAX2SAT

Given a 2CNF formula ϕ, using LP, let $N = \frac{3}{4} \cdot \mathrm{OPT}_f(\phi)$.
Let ϕ_a be the 2CNF formula obtained from ϕ by deleting the clauses that have already been satisfied by $x_1 = a$, and by deleting the x_1 from each of the causes containing the literal of x_1. Let M_a be the number of clauses that are satisfied simply by $x_1 = a$.
If $M_a + N(\phi_a) \geq N(\phi)$, then set $x_1 = a$.
Continuing the procedure, the algorithms finds an assignment σ that satisfies at least $\frac{3}{4}$ fraction of the OPT many clauses of ϕ.
Self-reducibility method: It works for a number of important problems.

MAX-SAT

Given a CNF formula ϕ, let C be the set of all clauses. For each $c \in C$, let S_c^+ be the set of variables that positively occur in c, and S_c^- be the set of negatively occurring variables in c.
Let z_c be the variable. z_c takes value 1 if c is satisfied, and 0 otherwise.
The MAX-SAT is

$$\max \sum_{c \in C} w_c z_c$$

subject to

$$\forall c \in C : \quad \sum_{i \in S_c^+} y_i + \sum_{i \in S_c^-} (1 - y_i) \geq z_c$$

$$\forall c : \quad z_c \in \{0, 1\}$$

$$\forall i : \quad y_i \in \{0, 1\}$$

LP of MAX-SAT

The LP relaxation is:

$$\max \sum_{c \in C} w_c z_c$$

subject to

$$\forall c \in C : \sum_{i \in S_c^+} y_i + \sum_{i \in S_c^-} (1 - y_i) \geq z_c$$

$$\forall c : \quad 0 \leq z_c \leq 1$$

$$\forall i : \quad 0 \leq y_i \leq 1$$

Solving the LP, let (y, z) be the optimum solution for the LP.
Independently, set $x_i = 1$ with probability y_i

Lemma

Let W_c be the weight contributed by clause c, and W be the sum of all W_c.
For $k \geq 1$, define

$$\beta_k = 1 - (1 - \frac{1}{k})^k.$$

Lemma. If c contains k literals, then

$$E[W_c] \geq \beta_k w_c z_c.$$

Proofs

Suppose without loss of the generality that $c = x_1 \vee x_2 \vee \cdots \vee x_k$. Then the probability that c is satisfied is:

$$\begin{aligned} 1 - \prod_{i=1}^{k}(1 - y_i) &\geq 1 - \left(\frac{\sum_{i=1}^{k}(1 - y_i)}{k}\right)^k \\ &= 1 - \left(1 - \frac{\sum y_i}{k}\right)^k \\ &\geq 1 - \left(1 - \frac{z_c}{k}\right)^k \geq \beta_k. \end{aligned} \tag{11}$$

where the first inequality follows from the arithmetic/geometric mean inequality, and the second from the constraints of the LP. Let $g(z) = 1 - (1 - \frac{z}{k})^k$. Then g is concave with $g(0) = 0$ and $g(1) = \beta_k$.

Proofs

$$E[W] = \sum_c E[W_c] \geq \beta_k \sum w_c z_c = \beta_k \mathrm{OPT}_f \geq \beta_k \mathrm{OPT}.$$

For all k,

$$(1 - \frac{1}{k})^k < \frac{1}{e}.$$

So this is a $1 - 1/e$ factor approximation algorithm for MAX-SAT. By the self-reducibility of CNF formula, we can give a deterministic approximation algorithm for MAX-SAT with approximation ratio $1 - 1/e$. (Exercise)

Convex program

A set of points K is called *convex*, if for every two $x, y \in K$, the line segment joining x, y lies entirely inside K.
A function $f : R^n \to R^n$ is *convex* if:

$$f(\frac{x+y}{2}) \leq \frac{1}{2}(f(x) + f(y))$$

for all x, y.
A *convex program* consists of a convex function f and a convex body K.
The goal is to find $x \in K$ to minimise $f(x)$. That is

$$\min f(x), x \in K$$

Positive semidefinite

A symmetric $m \times n$ matrix M is *positive semi-definite* (PSD), if there exists a matrix A such that

$$M = AA^{\mathrm{T}}$$

where A^{T} is the transpose of A
This is a generalisation of linear programming.
The semi-definite programming can be solved with error ϵ in time polynomial in n, $\log \frac{1}{\epsilon}$, using the *ellipsoid algorithm*.

MAX CUT

Given an undirected graph $G = (V, E)$, edge weights $w : E \to Q+$, find a partition $(S, \bar{S})$ of V to maximize the total weight of the edges in the cut, that is, the edges between S and $\bar{S}$, the complement of S.

Quadratic programs

Given an instance of MAX CUT, i.e., a graph G with n nodes and m edges.
For each i, assign $y_i = \pm 1$, set $S = \{i \mid y_i = 1\}$.
Then if i, j in the same side, then $y_i y_j = 1$, and -1, otherwise.
The MAX CUT is:

$$\max \quad \frac{1}{2} \cdot \sum_{1 \leq i < j \leq n} w_{i,j}(1 - y_i y_j)$$

$$y_i^2 = 1, \forall i \in V$$

$$y_i \in Z, \forall i \in V$$

Vector program

We interpret y_i as a vector v_i in R^n.
Then the vector program is:

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

$$v_i \cdot v_i = 1$$

$$v_i \in R^n$$

for all $i \in V$.
v_i are in the n-dimensional unit sphere.
The program is solvable with error ϵ in time polynomial in n and $\log(\frac{1}{\epsilon})$.

Algebraic properties

Given $n \times n$ matrix A, it is called *positive semi-definite*, if: $\forall x \in R^n$, $x^{\mathrm{T}} A x \geq 0$.

Theorem 3.1 Let A be $n \times n$ real symmetric matrix. Then the following are equivalent:

1) For all $x \in R^n$, $x^{\mathrm{T}} A x \geq 0$.
2) All eigenvalues of A are ≥ 0.
3) There is an $n \times n$ real matrix W such that

$$A = W^{\mathrm{T}} W$$

Proofs of Theorem 3.1

For $1) \Rightarrow 2)$.
$Ax = \lambda x$ implies $0 \leq x^{\mathrm{T}} Ax = \lambda x^{\mathrm{T}} x$, giving $\lambda \geq 0$, since $x^{\mathrm{T}} x > 0$.
For $2) \Rightarrow 3)$.
Suppose $\lambda_1, \lambda_2, \cdots, \lambda_n$ are the eigenvalues of A with corresponding eigenvectors $v_1, v_2, \cdots, v_n$, which are orthonormal.
Let U be the matrix with columns $v_1, v_2, \cdots, v_n$. Let Λ be the diag matrix of the $\lambda_1, \lambda_2, \cdots, \lambda_n$.
$AU = U\Lambda$, $UU^{\mathrm{T}} = I$.
$W = U(\Lambda)^{\frac{1}{2}}$ works.
$3) \Rightarrow 1)$.
Easy.

Decomposition

Define Λ, U as before.

$$A = U \Lambda U^{\mathrm{T}}$$

A PSD iff in the above decomposition, all $\lambda_i > 0$. The decomposition is found in poly time.

Proposition

If A, B are positive semidefine, then so it $A + B$.

(Exercise)

Definitions

Given $A \in R^{n\times n}$, the trace of A,

$$\mathrm{tr}(A) = \sum_{i=1}^{n} a_{i,i}.$$

Let A, B real, $n \times n$ matrices.
The *Frobenius inner product* of A and B is:

$$A \odot B = \mathrm{tr}(A^{\mathrm{T}} B) = \sum_{i,j} a_{i,j} b_{i,j}.$$

The semi-definite programming problem

Let Y be $n \times n$ real valued variables with y_{ij} the (i, j)-th entry.
Suppose $C, D_1, \cdots, D_k$ are positive semi-definite, $d_1, d_2, \cdots, d_k$ are reals.
The SDP is:

$$\max \quad C \odot Y$$

$$D_i \odot Y = d_i, 1 \leq i \leq k$$

$$Y \geq 0$$

i) If $C, D_1, \cdots, D_k$ are diagonal, it is the linear programming.
ii) The set of all feasible solutions form a convex.

Solving SDPs

Theorem

Let $\mathcal{S}$ be a semi-definite programming problem, and A be a point in $R^{n \times n}$. We can determine, in poly time, whether or not A is a feasible solution for $\mathcal{S}$, and if it is not, then find a separating hyperplane.

Solving SDPs - Proof

Proof.

Easy if A is a feasible solution. Otherwise:

Case 1. A is not symmetric. If $a_{ij} > a_{j.i}$, then $y_{ij} \leq y_{ji}$ is a separating hyperplane.

Case 2. A is not positive SDP. There is an eigenvalue $\lambda < 0$ with eigenvector v. Then $v^{\mathrm{T}} Y v \geq 0$ is a separating hyperplane.

Case 3. If any of the linear constraints is violated, then it directly gives a separating hyperplane. □

SDP for MAX-Cut

Lemma Vector program $\mathcal{V}$ is equivalent to $\mathcal{S}$.
Easy.

$$\max \quad \frac{1}{2} \cdot \sum_{1 \leq i < j \leq n} w_{ij}(1 - y_i y_j)$$

subject to:

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

Suppose that $a_1, a_2, \cdots, a_n$ are the optimal solution and OPT_v be the optimal value of the vector programming.
Note the vectors lie on the n-dimensional unit sphere.

Randomized rounding

For i, j, let θ_{ij} be the angle between a_i and a_j.
The contribution of a_i and a_j to OPT_v is:

$$\frac{w_{ij}}{2}(1 - \cos\theta_{ij}).$$

If $\theta_{ij} \approx \pi$, then $\cos\theta_{i,j} \approx -1$, the contribution is large.
This means that is θ_{ij} is large, then v_i, v_j are split.
Rounding: Pick a random r on the unit sphere.
Define

$$S = \{v_i \mid a_i \cdot r \geq 0\}.$$

Probability of splitting

Lemma

The probability that v_i and v_j are separated is exactly $\frac{\theta_{ij}}{\pi}$.

Proof.

v_i and v_j are separated if and only if they are separated by r, which occurs with prob $\frac{\theta_{ij}}{\pi}$. □

Algorithm for MAX-CUT

Clearly, $\langle v_i, v_j \rangle = \cos\theta_{ij}$.
The algorithm for MAX-CUT proceeds as follows:

1. Solve the vector programming. Let $a_1, a_2, \cdots, a_n$ be the optimum solution.
2. Pick r randomly and uniformly
3. Let $S = \{i \mid a_i \cdot r \geq 0\}$.

Let W be the weight of the cut $(S, \bar{S})$, and

$$\alpha = \frac{2}{\pi} \cdot \min_{0 \leq \theta \leq \pi} \frac{\theta}{1 - \cos\theta}$$

Exercise. Using calculus, show that $\alpha > 0.87856$.

Proofs

Lemma

$$E[W] \geq \alpha \cdot \mathrm{OPT}_v \geq \alpha \cdot \mathrm{OPT}.$$

Proof.

$$E[W] = \sum_{0 \leq i < j \leq n} w_{ij} \frac{\theta_{ij}}{\pi}$$

$$\geq \alpha \cdot \sum_{1 \leq i < j \leq n} \frac{1}{2} w_{ij} (1 - cos\theta_{ij}) = \alpha \cdot \mathrm{OPT}_v. \tag{12}$$

$\square$

Randomized approximation for MAX-CUT

Theorem

There exists a randomized approximation algorithm for MAX-CUT achieving an approximation factor of 0.87856*.*

Proof.

Let T be the total weight, let a be such that $E[W] = aT$.
Let $p = \Pr[W < (1-\epsilon)aT]$.
We have

$$aT \leq p(1-\epsilon)aT + (1-p)T$$

giving

$$p \leq \frac{1-a}{1-a+a\epsilon}.$$

$\square$

Proofs

Now

$$T \geq E[W] = aT \geq \alpha \cdot \mathrm{OPT}_v \geq \alpha \cdot \mathrm{OPT} \geq \frac{\alpha \cdot T}{2}$$

so $1 \geq a \geq \alpha/2$ and

$$p \leq 1 - c$$

for $c = \frac{\epsilon\alpha/2}{1+\epsilon\alpha/2-\alpha/2}$
Run the algorithm $\frac{1}{c}$ many times, let W' be the maximal weight, then

$$\Pr[W' \geq (1-\epsilon)aT] \geq 1 - (1-c)^{1/c} \geq 1 - \frac{1}{e}.$$

Conclusions

1. Approximation is a general approach to hard problems
2. There are general methodology for approximation:
   - Randomness is useful
   - Linear programming
   - Semi-definite programming
     Idea To amplify solution space, so that the optimum of the generalised problem is solvable, and then find the closest solution to the generalised solution, prove a guaranteed bound.
   - Combinatorial approach - open, no theoretical foundation
     Clever ideas for understanding the structures of the solutions, from which good algorithms are found
   - Is there a direction of approximation algorithms by embedding, extracting, and abstracting? Open question.

Questions

1. Real applications of approximation algorithms
2. The limit of the approximation
3. New method for approximation
4. Further reading:
   Vijay V. Vazirani
   Approximation Algorithms, Springer, 2003

Exercise

1. Give a factor $\frac{1}{2}$ approximation algorithm for the MAX-CUT. A simple greedy
2. Give a deterministic approximation algorithm for MAX-SAT with approximation ratio $1 - 1/e$.
3. Design a fast algorithm to approximate the diameter of a connected graph.

Thank You!