Efficient Techniques in Geometric Optimization: An Overview

Pankaj K. Agarwal

Center for Geometric Computing

http://www.cs.duke.edu/CGC/

May 1999

Geometric Optimization

Combinatorial Optimization

☆ Maximize (or minimize) a function $\varphi(x_1, \ldots, x_m)$

☆ subject to a large number of equality and inequality constraints.

☆ *Examples:* Linear and integer programming

Geometric Optimization

☆ Constant number of variables
- Fixed dimensional linear programming

☆ Constraints are induced by geometric objects
- Geometric set cover, Euclidean matching/TSP in $\mathbb{R}^2$.

Can geometry help in speeding up the algorithms?

An Example: Clustering

S: Set of (demand) points in $\mathbb{R}^2$

k: An integer

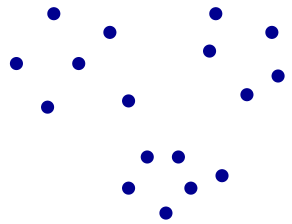
Find a set Σ of k supply points s.t.

☆ *k-center*: minimize $\Phi(S) = \max_{p \in S} \min_{q \in \Sigma} d(p, q)$.

- Cover S by k congruent disks of the smallest radius.

☆ *Discrete k-center*: $\Sigma \subseteq S$ and $\Phi(S)$ is minimized.

☆ *k-median*: minimize $\Phi(S) = \sum_{p \in S} \min_{q \in \Sigma} d(p, q)$.

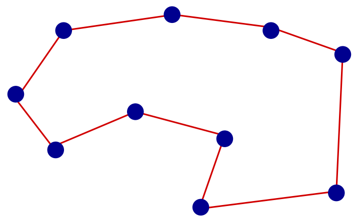


Euclidean Matching & TSP

S: Set of points in $\mathbb{R}^2$

☆ $G_S = (S, S \times S)$

☆ $w : S \times S \to \mathbb{R}$,
$w(p, q) = d(p, q)$



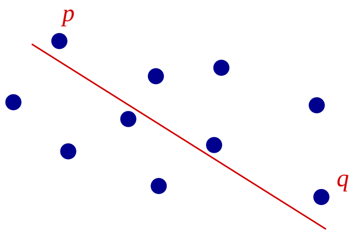
☆ Find a minimum-weight tour in G_S (a cycle that visits each point exactly once).

- NP-Hard (Papadimitriou, 77), efficient heuristics (Karp, 77)
- 1.5-approximation (Christofides, 76)
- $(1 + \varepsilon)$-approximation (Arora, 98; Rao-Smith,98)

☆ Find a minimum-weight matching in G_S.

- $\tilde{O}(n^{1.5})$ (Varadarajan, 98)
- Approximation: $\tilde{O}(n/\varepsilon^3)$ (A.-Varadarajan, 99)

Statistical Estimators



S: Set of points in $\mathbb{R}^d$
($d = 2, 3$)

☆ *Diameter*
$\mathrm{diam}(S) = \max_{p,q} d(p, q)$

☆ *Hyperplane fitting:* Find a hyperplane h s.t.
$\max_{p \in S} d(p, h)$ is minimum.
- Same as the width problem.

☆ *Sphere fitting:* Find a sphere C s.t.
$\max_{p \in S} d(p, C)$ is minimum.
- Same as the minimum-width annulus (shell) problem.

Optimization Problems & Basis

☆ S: set of n objects

☆ $W : 2^S \to \mathbb{R}$
$W(A)$: cost of the optimal solutions for a subset $A \subseteq S$.
- Radius of disks in k-center
- Length of the tour in TSP

☆ *Basis:* A smallest subset B s.t. $W(B) = W(S)$.

☆ Either compute B or $W(B) = W(S)$.

☆ In many geometric optimization problems $|B| = O(1)$.
- Smallest enclosing disk

Optimization & Decision Problems

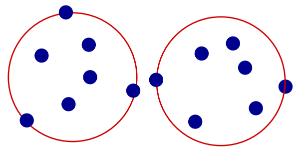
Ø: optimization problem; minimizes a univariate function.

☆ $\mathcal{C}$: A finite set of candidate values for $W(S)$

- E.g., $\mathcal{C} = \{W(A) \mid A \subseteq S \,\&|A| = |\text{Basis}(Ø, S)|\}$

☆ For k-center, $\mathcal{C}$ consists of two parts

- $d(p,q)/2$, $p,q \in S$
- Radii of circles passing through every triple of points in S.



Decision Problem, $D(r)$:

☆ Decide whether $W(S) < r$, $W(S) = r$, $W(S) > r$.

☆ k-center: Can S be covered by k disks of radius r each?

☆ $A(r)$: An algorithm for the decision problem.

Searching Over $\mathcal{C}$

☆ Perform a binary search over $\mathcal{C}$ using $A(r)$ at each step.

☆ $\mathcal{C}$ is too large to be computed explicitly.

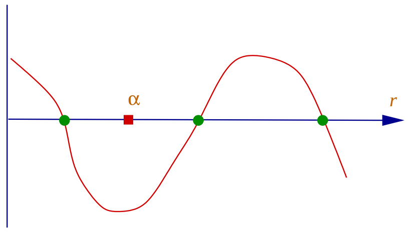
- Smallest enclosing disk, 2-center
- diameter of S

☆ Perform an *implcit binary search*

- Parametric searching
- Random selection
- expanders, $(1/r)$-cuttings
- Matrix searching

Parametric Searching

- ☆ $A(r)$: Comparison based algorithm for the decision problem
  - Each step performs a comparison of the form $p(r) > 0$, $p(r) = 0$, or $p(r) < 0$; p is a constant-degree polynomial.
  - Need some additional assumptions on the behavior of A.
  - T: running time of A.
- ☆ Use $A(r)$ itself to perform an implcit binary search over $\mathcal{C}$.
- ☆ A comparison $p(\alpha)$ can be resolved without knowing the exact value of α.
  - Find two adjacent roots ξ, ζ of p s.t. $\xi < \alpha < \zeta$



Parametric Searching

- ☆ Simulate A at $r = r^* = W(S)$ generically without knowing the exact value of r^*.
  - Need to compute the sign of $p(r^*)$.
  - Call the decision algorithm A at each root of p to determine the two consecutive roots α, β of $p(\cdot)$ s.t. $r^* \in (\alpha, \beta)$.
- ☆ Implicitly performs a binary search on $\mathcal{C}$.
- ☆ **Claim:** One of the calls to the decision algorithm $A(\cdot)$ is at $r = r^*$.
- ☆ Total time: $O(T^2)$.

Parametric Searching

- ☆ $A_P(r)$: Parallel implementation of A.
  T_P: running time, Π: # processors.
- ☆ Run A_P generically at r^*.
- ☆ For each parallel step of A_P do the following:
  - $p_1, \ldots, p_\Pi$: polynomials whose signs to be determined at r^*.
  - Compute & sort the roots of all the polynomials whose signs to be computed at r^*: $z_1 < z_2 < \cdots < z_u$
  - Perform a binary search on these roots using A.
  - Compute the i s.t. $r^* \in (z_i, z_{i+1})$.
  - Signs of all p_i's is the same for all $r \in (z_i, z_{i+1})$ and can be evaluated in $O(1)$ time.

Parametric Searching

- ☆ Resolving all comparisons at each parallel step requires $O(T \log \Pi)$ time.
- ☆ Total time: $O(T_P \Pi + T_P \cdot T \log \Pi)$.

Main challenge: Develop sequential and parallel algorithms for the decision problem.
Requires many sophisticated geometric techniques.

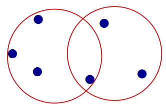
Typically $T_P = \log^{O(1)} n$ and $\Pi = n^{O(1)}$.

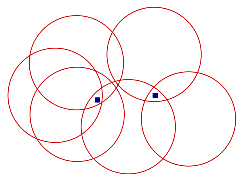
Extensions

- ☆ Multidimensional parametric searching
  - Recurse on dimension at each step
  - k-dimensional opitmization algorithm is used for the decision problem in dimension $(k+1)$.
- ☆ Parametric searching used for query-type problems.

Example: 2-center

Decision algorithm:
Can S be be covered by two disks of radius 1?
$D = \{B(p, 1) \mid p \in S\}$.
Can all disks in D be hit by two points?





☆ (A.-Sharir, 1994)
- $O(n^2 \log n)$ sequential algorithm
- $O(\log n)$ time, $O(n^2)$ processor

☆ (Hershberger, 1993)
- $O(n^2)$ squential algorithm

☆ (Sharir, 1996; Eppstein 1997)
- $O(n \log^2 n)$ squential algorithm

☆ Discrete 2-center:
Can D be hit by two points of S?
- $O(n^{4/3} \log^3 n)$ (A., Sharir, Welzl, 1998)

Open Problems

☆ 2-line center: Cover a set S of points in the plane by two congruent strips of the minimum width.
Is there a subquadratic algorithm.

☆ Projective clustering:: Approximate a set S of points in $\mathbb{R}^d$ by a few k-flats.

- No efficient approximation algorithm is known.
- Useful for higher dimensional searching problems.

Random Selection

☆ $\mathcal{C}$: Set of candidate values for $r^* = W(S)$.

☆ Maintain an interval $I = [\alpha, \beta]$ s.t. $r^* \in I$.

☆ Select a random element $\sigma \in I \cap \mathcal{C}$.
- Nontrivial step if $\mathcal{C}$ is not computed explicitly.
- Matoušek presents algorithms in many cases.

☆ Use the decision algorithm A to compare r^* with σ.
- If $r^* = \sigma$, return σ.
- If $r^* > \sigma$, set $I = [\sigma, \beta]$.
- If $r^* < \sigma$, set $I = [\alpha, \sigma]$.

☆ Expected value of $|I_{new} \cap \mathcal{C}| \leq 3/4 \cdot |I_{old} \cap \mathcal{C}|$.

☆ Expected running time: $O(T \log n)$.

Deterministic Approaches

☆ Use expanders instead of selecting a random element.
- Katz-Sharir (1993, 1997)
- Varadarajan-A. (1998)

☆ Use $(1/r)$-cuttings or ε-nets to choose an element σ.
- Bronniman-Chazelle (1994)

☆ Matrix searching; Frederickson-Johnson (1983, 1984)
Works if $\mathcal{C}$ can be represented as an $m \times m$ matrix M s.t.

- Rows and colums of M are sorted
- Given $1 \leq i, j \leq m$, $M[i, j]$ can be computed efficiently.
- Performs a binary search using the decision algorithm A
- Running time: $O(T \log m)$

Clarkson-Shor Technique

- ☆ S: n objects, $\mu : S \to \mathbb{R}$
- ☆ Compute $\max_{p \in S} \mu(p)$
- ☆ For the diameter problem:
  - S: n points in $\mathbb{R}^3$ $\mu(p) = \max_{q \in S} d(p, q)$
  - $\mathrm{diam}(S) = \max_{p \in S} \mu(p)$

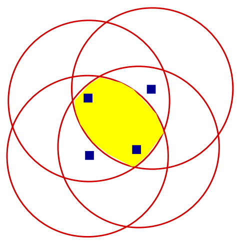
```
⟨p_1, ..., p_n⟩:  a random permutation of n
curr = −∞
for i = 1 to n do
    if μ(p_i) > curr
        curr = μ(p_i)        (⋆)
return curr
```

- ☆ Expected # times *curr* is updated: $O(\log n)$.
- ☆ Useful when checking $\mu(p_i) > curr$ (using some preprocessing) is cheaper than computing $\mu(p_i)$.

Diameter in $\mathbb{R}^3$

- ☆ Computing $\mu(p)$: $O(n)$ time.
- ☆ Deciding $\mu(p) > curr$:
  - $\mathcal{I} = \bigcap_{i=1}^{n} B(p, curr)$
  - $\mu(p) \le curr \Leftrightarrow p \in \mathcal{I}$.
  - $\mathcal{I}$: requires $O(n \log n)$ expected time.

    Point location: $O(\log n)$ time.



- ☆ $\mathrm{diam}(S)$ can becomputed in $O(n \log^2 n)$ expected time

  Running time can be improved to $O(n \log n)$.
- ☆ Deterministic algorithm: $O(n \log^2 n)$.
- ☆ *Develop an $O(n \log n)$ deterministic algorithm.*

Extensions of Clarkson-Shor Technique

☆ (A., Sharir, 1996) Deciding $\mu(p) \leq curr$ is not always easy

- *Example:* S is a set of lines in $\mathbb{R}^3$.

- Extended the approach to such situations.

☆ (Chan, 1998) Developed a more a general framework under which this approach works.

Linear Programming

☆ Prune-and-search (Megiddo, 1983, 1984)

- $O(3^{d^2} n)$ algorithm for d-dim LP (Clarkson, 1986)
- Approach works for many other problems, e.g., 1-center.

☆ Iterative weighted random sampling (Clarkson, 1995)

- $O(d^2 n) + d^{d/2+O(1)} \log n$ for d-dim LP

☆ Randomized incremental algorithm (Matoušek, Sharir, Welzl, 1996)

- Subexponential running time
- Present an abstract framework called LP-type problems Works even for some instances of nonconvex programming.
- (Matoušek, 1995) Extended to finding a solution that violates $\leq k$ constraints.
- (Amenta, 1994) Relationship between Helly-type theorem and LP-type problems.

Clarkson's LP Algorithm

☆ H: Set of n constraints in $\mathbb{R}^d$

☆ $\mu : H \to \mathbb{Z} \quad \mu_h = 1 \quad \forall h \in H$

```
function procedure RANDOM_LP (H)
    if n ≤ 6d² then
        return Simplex(H)
    else
        r = 6d²
        repeat
            choose random R ∈ (H choose r)
            x_R := Simplex(R)
            V := {h ∈ H | x_R violates h}
            if μ(V) ≤ 3μ(H)/d then
                for all h ∈ V do μ_h := 2μ_h
        until V = ∅
        return x_R
```


Clarkson's LP Algorithm

☆ B: Set of d constraints forming incident to the optimal vertex.

☆ Each *successful* iteration doubles the weight of at least one constraint in B.

☆ Expected # iterations $= O(d \log n)$

☆ Best known deterministic algorithm: $d^{O(d)} \cdot n$.

☆ *Is there a subexponential deterministic algorithm for LP?*

Approximation Algorithms

☆ Computing optimal solutions is hard in most cases.

☆ Optimization criteria are typically artificial.

Approximation Techniques

☆ Discretization

☆ Greedy algorithm and Clarkson's algorithm for set cover

☆ Canonical solutions and dynamic programming

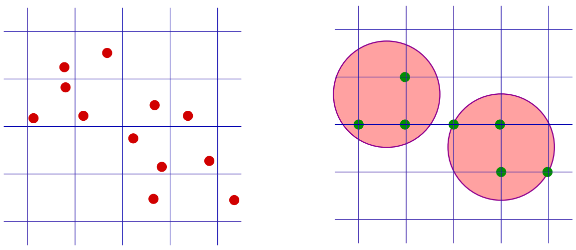
☆ Relaxation techniques and randomized rounding

☆ Primal-dual methods

☆ Monte Carlo algorithms

Discretization

Discretize the input space



k**-center**: $(1+\varepsilon)$-approximation algorithm

☆ Draw a grid of appropriate size.

☆ Map each point p to its nearest grid point p'.

☆ Solve the k-center problem for $S' = \{p' \mid p \in S\}$.

☆ If the grid size is chosen properly, then

- $|S'| = O((k/\varepsilon)^2)$
- the solution for S' approximates that of S.

Discretization

Discretize the output space

- ☆ Suppose we want to fit a line through a point set S in $\mathbb{R}^2$.
- ☆ Choose a finite set Δ of directions.
- ☆ Compute the best-fit line in one of the chosen directions.

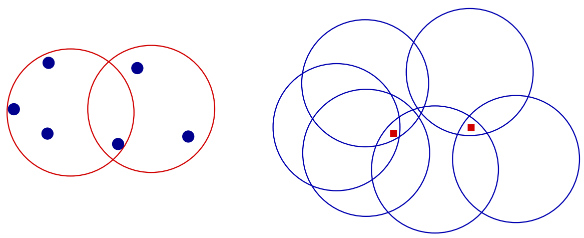
- ☆ A uniform sampling of the solution spacedoes not always give a good approximation.

- ☆ Clever discretization techniques are needed.
  - $(1+\varepsilon)$-approximation for the minimum width annulus in $O(n/\varepsilon^2)$-time (A., Aronov et al., 99).
  - $(1+\varepsilon)$-approximation for various clustering algorithms.
  - *An efficient approximation algorithm for the min width cylindrical shell?*

Geometric Set Cover

- ☆ S: n points in $\mathbb{R}^2$

- ☆ Cover S by the minimum number of unit-radius disks.



Hitting set is the dual of set cover.

- ☆ D: Set of disks in the plane
- ☆ V: Set of intersection points of D.
- ☆ k^*: Minimum number of points that hit all disks of D.
- ☆ Greedy algorithm will return a hitting set of size $O(k^* \log n)$.

Clarkson's Algorithm

Repeat the following steps

```
function procedure HITTING_SET (D)
    choose random R ⊆ V of size O(k log k)
    if R hits all disks of D
      return R
    δ:  δ ∩ R = ∅
    if μ(V ∩ δ) ≤ μ(V)/k
        for all p ∈ δ ∩ V do μ_p := 2μ_p
```

☆ $H \subseteq V$: an optimal hitting set of D.

☆ Each *successful* iteration doubles the weight of at least one point in H.

☆ Expected # iterations: $O(k \log n)$

☆ One iteration: $O(n^2)$

Clarkson's Algorithm

Extensions

☆ Bronniman, Goodrich (1996)
- Extended to any set system with finite VC-dimension.
- Output size: Size of $(1/k^*)$-net.

☆ A., Desikan (1997); A., Procopiuc (1999)
- Computing the set system explicitly is expensive
- Described algorithms for maintaining the set system implcitly.

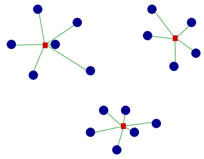
☆ *Is there a $O(1)$-approximation algorithm for geometric set cover?*

Canonical Solutions & Dynamic Programming

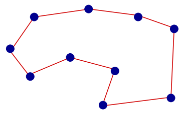
- ☆ Show that there exists a *canonical* solution whose cost is near optimal and that satisfies some nice properties.
- ☆ Find an optimal canonical solution.
- ☆ Typically dynamic programming is used to find an optimal canonical solution.

k-median problem: Find a set Σ of k points s.t.

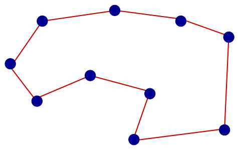
$$\min \sum_{p \in S} \min_{\sigma \in \Sigma} d(p, \sigma).$$



Euclidean TSP: Compute a minimum lenth tour of S.



Approximation Algorithm for TSP

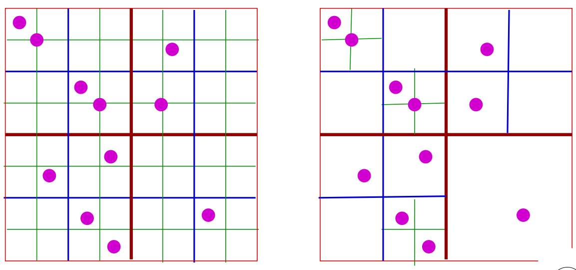


- ☆ S: Set of n points in $\mathbb{R}^2$
- ☆ $\varepsilon > 0$: a parameter
- ☆ *Compute a tour of S whose length is at most $(1+\varepsilon)OPT$.*

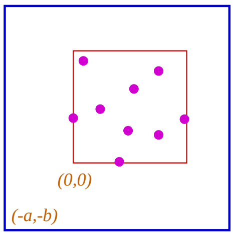
Can make the following assumptions on S

- ☆ Distance between closest pair is ≥ 8.
- ☆ All points have integer coordinates.
- ☆ Length of the smallest square enclosing S is $L = O(n/\varepsilon)$.
- ☆ Bottom-left corner of the bounding box is at $(0, 0)$.

Dissection and Quad-tree



Arora's Algorithm



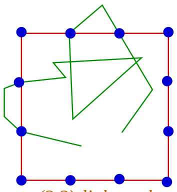
☆ Choose two integers $a, b \in [0, L]$.

☆ B: Square of length $2L$ whose bottom-left corner is at $(-a, -b)$.

☆ Construct a quad-tree Q on B.

☆ *m-regular portals* for a quad tree:

- Set of points on the edges of the squares of the quad tree.
- Each square has a point on each corner and m other equally spaced points on each edge



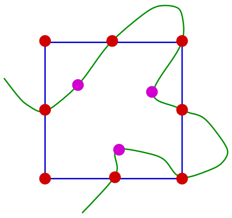
(2,2)-light path

☆ A path Π is *(m, r)-light* if

- Π crosses an edge of every square C of $Q \leq r$ times,
- all crossing points Π and C lie on m-regular portals.

Arora's Algorithm

The edges of a tour of S are allowed to be polygonal chains.



Claim: *With probability* $\geq 1/2$, $\exists$ *an* (m, r)*-light tour* Π *of* S *s.t.*

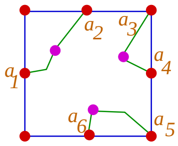
$$|\Pi| \leq (1 + \varepsilon) \cdot OPT,$$

$$m = O((\log L)/\varepsilon) \qquad r = O(1/\varepsilon).$$

☆ Compute a (m, r)-light tour of S of the minimum length.

☆ For a square $C \in Q$ and for a sequence $a_1, a_2, \ldots, a_{2p}$, where $2p \leq 4r$, define $\Pi(C, a_1, \ldots, a_{2p})$

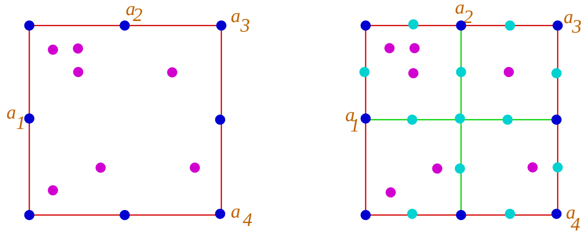
- Collection of p paths, the ith path from a_{2i-1} to a_{2i},
- paths lie inside C,
- they cover all the points of $S \cap C$.



Arora's Algorithm

- ☆ # subproblems: $O(n \log n) \times m^r = n \log^{O(1/\varepsilon)} n$
- ☆ Leaves of Q can be handled easily.
- ☆ For a nonleaf square C, compute $\Pi(C, a_1, \ldots, a_p)$ from the four children $C_1, \ldots, C_4$ of C.

  Guess the crossing points of Π and C_i's



Time: $O(m^r) = \log^{O(1/\varepsilon)} n$.

- ☆ Total time: $n \log^{O(1/\varepsilon)} n$
- ☆ Running time is improved by Rao-Smith (1998) to

$$2^{O(1/\varepsilon)} n + O(n \log n).$$

Arora's Algorithm

Extensions

- ☆ $(1+\varepsilon)$-approx for Euclidean minimum weight matching
  $\tilde{O}(n/\varepsilon^3)$ (A., Varadarajan, 99)
- ☆ $(1+\varepsilon)$-approx for k-median
  $n^{O(1/\varepsilon)}$ (Arora et al., 98)

Limitations

- ☆ An $\tilde{O}(n/\varepsilon^{O(1)})$-time $(1+\varepsilon)$-approx algorithm for Euclidean minimum weight bipartite matching?
- ☆ An efficient $(1+\varepsilon)$-approx algorithm for minimum weight triangulation?
- ☆ Efficient $(1+\varepsilon)$-approx algorithm for more generalized clusterings (e.g., minimize $\sum_{p \in S} \min_{q \in \Sigma} d^2(p,q)$)

Relaxation & Rounding

- ☆ State the optimization problem as an instance of integer programming.
- ☆ Relax the integer constraints and compute a fractional solution.
  - Use LP or SDP
- ☆ Round the fractional solution to an integer solution.
  - Use randomized rounding
- ☆ *Examples:*
  - Packing & covering
  - k-median
  - max-cut, MAXSAT, coloring, ...

k-Median

$$\min_{1 \le i,j \le n} d_{ij} x_{ij}$$

subject to

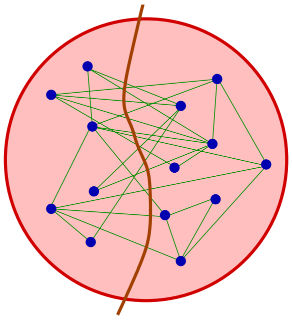
$$\begin{aligned}
\sum_{i=1}^{n} x_{ij} &= 1 && 1 \le j \le n \\
x_{ij} &\le y_i && 1 \le i,j \le n \\
\sum_{j=1}^{n} y_j &\le k \\
x_{ij}, y_i &\in \{0,1\}
\end{aligned}$$

- ☆ Lin-Vitter, 1992 $(1+\varepsilon)$-approx, # facilities $\le (1+\varepsilon)k \log n$

$\vdots$

- ☆ Charikar *et al.*, 1999 6.67-approximation

Max-Cut



$G = (V, E)$, $w_{ij} : E \to \mathbb{R}$

Find a maximum weight cut in G.

- ☆ x_i: A variable for the node i.
- ☆ $\max \sum_{i<j} w_{ij} \frac{1 - x_i x_j}{2}$
- ☆ *subject to* $x_i \in \{-1, +1\}$, $1 \le i \le n$.
- ☆ Relax: $x_i \longrightarrow \mathbf{v}_i \in \mathbb{S}^{n-1}$
- ☆ $\max \sum_{i<j} w_{ij} \frac{1 - \mathbf{v}_i \cdot \mathbf{v}_j}{2}$
- ☆ *subject to* $\mathbf{v}_i \in \mathbb{S}^{n-1}$, $1 \le i \le n$.

Max-Cut

- ☆ $\mathbf{v}_i \cdot \mathbf{v}_j \longrightarrow y_{ij}$.
- ☆ $\max \sum_{i<j} w_{ij} \frac{1 - y_{ij}}{2}$
- ☆ *subject to* $\mathbf{Y} = (y_{ij})$ being semidefinite & $y_{ii} = 1$
- ☆ $\mathbf{v}_i$'s can be recovered from $\mathbf{Y}$.

Rounding:

- ☆ Choose a random vector $\mathbf{r} \in \mathbb{S}^{n-1}$.
- ☆ $x_i = \mathrm{sign}(\mathbf{v}_i \cdot \mathbf{r})$

Gives 0.878-approx algorithm [Goemans-Williamson]

Conclusions

- ☆ Lower bounds for geometric approximation algorithms.
- ☆ Can SDP and primal-dual methods be used for geometric optimization problems?
- ☆ Monte Carlo algorithms for geoemtric optimization.
- ☆ More practical approximation algorithms for TSP, matching, etc.
  - Polynomial dependancy on ε
  - Randomization instead of dynamic programming