

Semidefinite Programming for Approximation

Yinyu Ye

Department of Management Science and Engineering

Stanford University

Stanford, CA 94305, U.S.A.

<http://www.stanford.edu/yye>

Outline

- Radii of High-Dimension Points and SDP Approximation
- 2-Catalog Segmentation and SDP Approximation
- Facility Location and Approximation

Approximate the Radii of Projected Point Sets

- **Input.** A set P of $2n$ symmetric points in Euclidean space R^d If $p \in P$ then $-p \in P$.
- **Objective.** To minimize the outer k -radius of P

$$R_k(P) = \min_{F \in F^k} \max_{p \in P} d(p, F),$$

where F^k is the collection of all k -dimensional subspaces of R^d , and $d(p, F)$ is the radius of the projection of p onto F .

- **Mathematical formulation.** The square of $R_k(P)$ can be defined as the minimum of, over all sets of k orthogonal unit vectors $\{x_1, x_2, \dots, x_k\}$,

$$\max_{p \in P} \sum_{i=1}^k (p^T x_i)^2.$$

Figure 1: Radius of points

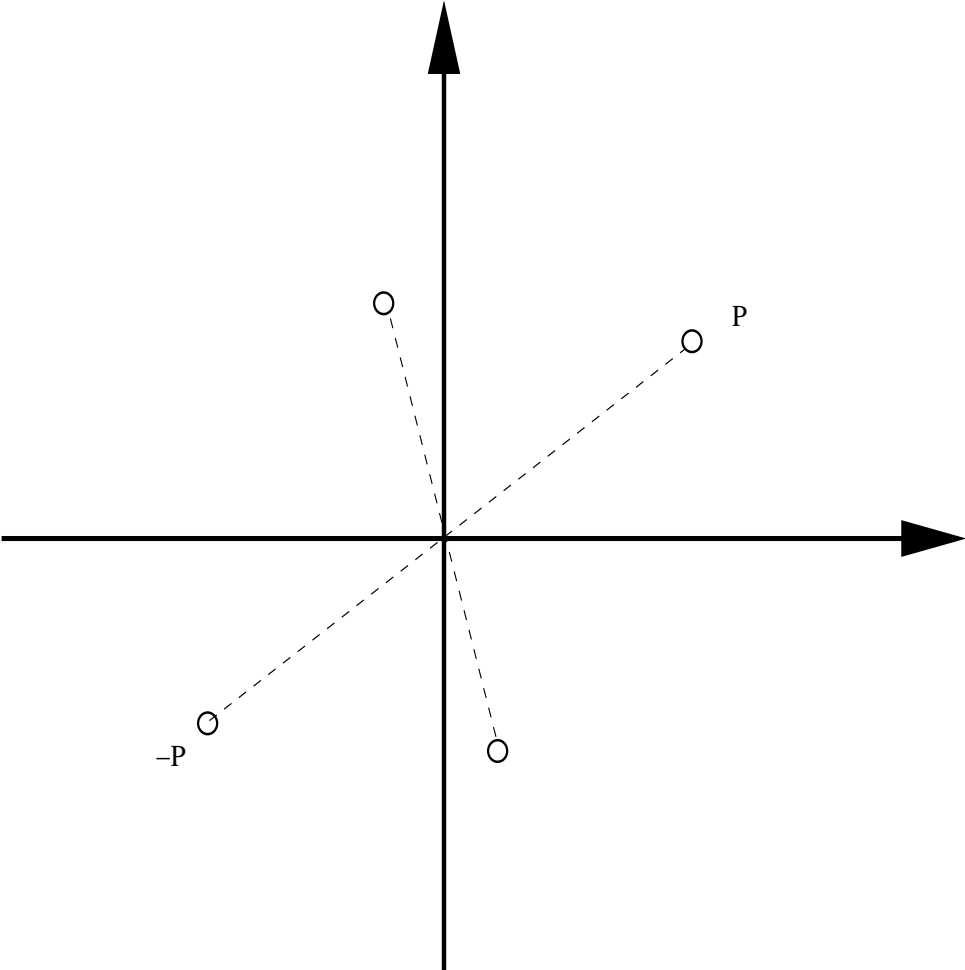
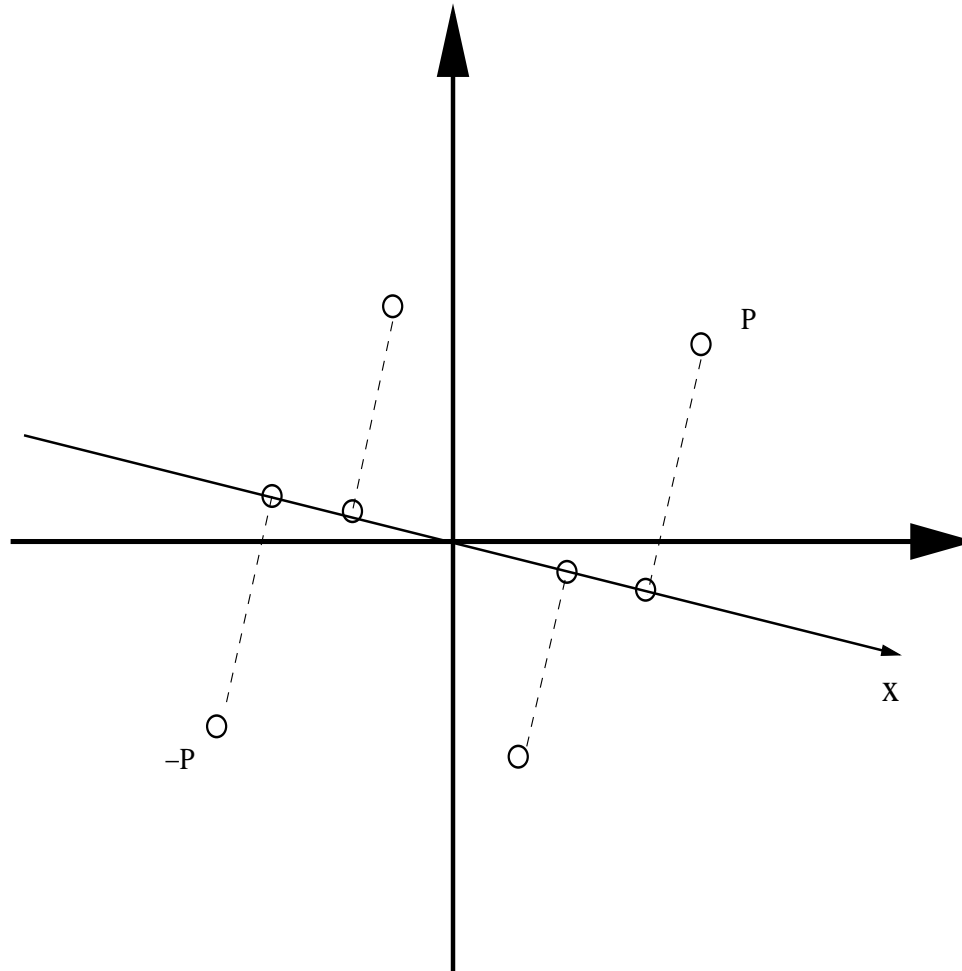


Figure 2: Radius of projected points on one-dimensional x 

Previous Results

- Fundamental problem in computational convexity and has applications in data mining, statistics and clustering (Gritzmann and Klee 1993, 1994).
- When d (the dimension) is a constant The problem is polynomial time solvable (Faigle et al 1996). It can also be approximated by a factor of $(1 + \epsilon)$ by saving the running time (Barequet and Har-Peled 2001, Har-Peled and Varadarajan 2001).
- When d is a variable when $d - k$ is a constant, the problem can be approximated by $(1 + \epsilon)$ (Badoiu et al 2002, Har-Peled and Varadarajan 2002).
- For $k = 1$, there is a randomized $O(\log n)$ algorithm for $R_k(P)^2$ (Implied from Nemirovskii et al. 1999, Nesterov 1998)
- Nothing is known when $d - k$ varies. A few hardness results shows that it is NP-hard to approximate the problem (Briden 2000, 2002).

Varadarajan/Venkatesh/Zhang Results (FOCS2002)

- There is a poly-time algorithm that approximates $R_k(P)^2$ within a factor of $O(\log n \cdot \log d)$ for any $1 \leq k \leq d$.
- There exists a constant $\delta > 0$ such that the following holds for any $0 \leq \epsilon < 1$ there is no quasi-polynomial time algorithm that approximates $R_k(P)$ within $(\log n)^\delta$ for all k such that $k \leq d - d^\epsilon$ unless $NP \subseteq DTIME[2^{(\log m)^{O(1)}}]$.
- For any $\epsilon > 0$ and any constant $c \geq 1$. There is no quasi-polynomial time algorithm that approximates $R_k(P)$ within $(\log d)^c$ for all k such that $k \leq d - d^\epsilon$ unless $NP \subseteq DTIME[2^{(\log m)^{O(1)}}]$.

Tools in their Approach

- Using SDP relaxation and a double rounding
- The first rounding uses the Johnson-Linderstrauss Lemma
- The second one uses another randomized rank reduction

Johnson-Linderstrauss Lemma

Lemma 1 *Let p be a vector in R^d , let v_1, \dots, v_r be a set of orthogonal unit vectors. Let G be a random s -dimensional subspace spanned by v_1, \dots, v_r . Let u_1, \dots, u_s be any orthonormal basis for G . Then there exists a constant C such that if $s \geq C \log n$, then inequality*

$$\frac{r}{s} \cdot \frac{\langle q, u_1 \rangle^2 + \dots + \langle q, u_s \rangle^2}{\langle q, v_1 \rangle^2 + \dots + \langle q, v_r \rangle^2} \leq 2$$

holds with probability at least $1 - \frac{1}{n^2}$.

New Result (Ye and Zhang 2003)

There is a poly-time algorithm that approximates $R_k(P)^2$ within a factor of $O(\log n)$ for any $1 \leq k \leq d$. This is a factor $\log d$ reduction from the previous best $O(\log n \cdot \log d)$, and the new bound is independent of d .

- Using SDP relaxation
- Using a deterministic partition
- Using a randomized rank reduction

Quadratic Representation

$$R_k(P)^2 = \text{Minimize } \alpha$$

Subject to

$$\sum_{i=1}^k (p^T x_i)^2 \leq \alpha, \forall p \in P,$$
$$\|x_i\|^2 = 1, i = 1, \dots, k,$$
$$(x_i)^T x_j = 0, \forall i \neq j.$$

SDP Relaxation

Consider the matrix $X = (x_1x_1^T + x_2x_2^T + \cdots + x_kx_k^T)$, we get an SDP relaxation

$$\alpha_k^* = \text{Minimize } \alpha$$

$$\text{Subject to } pp^T \bullet X \leq \alpha, \forall p \in P,$$

$$I \bullet X = k,$$

$$I - X \succeq 0,$$

$$X \succeq 0.$$

Orthogonal Decomposition

Let X^* be an SDP optimizer with rank r . Then, considering λ_i and x_i being the eigenvalues and eigenvectors of X^* , we can compute, in “polynomial time”, a set of non-negative reals $\lambda_1, \dots, \lambda_d$ and a set of orthogonal unit vectors x_1, \dots, x_r in R^d such that

- $\sum_{i=1}^r \lambda_i = k$
- $\max_i \lambda_i \leq 1$
- $\sum_{i=1}^r \lambda_i (p^T x_i)^2 \leq \alpha_k^* \leq R_k(P)^2$ for all $p \in P$.

Note that $r \geq k$. (Why?)

Case of $k = 1$ 1-Randomized Rounding

Generates a random r -dimensional vector u where, independently,

$$u_i = \begin{cases} \sqrt{\lambda_i} & \text{with probability } .5 \\ -\sqrt{\lambda_i} & \text{otherwise.} \end{cases}$$

Assign

$$\hat{x} = \sum_{i=1}^r u_i \cdot x_i.$$

Notice that

$$I \bullet \hat{x}\hat{x}^T = \sum_{i=1}^r \lambda_i = 1.$$

Probabilistic Analysis

Therefore, \hat{x} is a unit vector. Moreover,

$$\mathbb{E}[\hat{x}\hat{x}^T] = X^*,$$

so that

$$\mathbb{E}[pp^T \bullet \hat{x}\hat{x}^T] = pp^T \bullet X^* \leq \alpha_1^* \quad \forall p \in P.$$

Moreover, with probability at least $1/2$

$$pp^T \bullet \hat{x}\hat{x}^T \leq O(\log n) \cdot pp^T \bullet X^* \leq O(\log n) \cdot \alpha_1^*, \quad \forall p \in P.$$

Recall that n is the number of $(p, -p)$ in P , and the analysis is similar to Nemirovskii, Roos and Terlaky (1999).

Case of general k

Partition λ_i s into k groups, I_1, \dots, I_k , such that

$$\sum_{i \in I_j} \lambda_i \geq \frac{1}{2}, \quad \forall j = 1, \dots, k.$$

Let

$$X_j = \sum_{i \in I_j} \lambda_i x_i x_i^T, \quad j = 1, \dots, k.$$

Then

$$\sum_{j=1}^k p p^T \bullet X_j \leq \alpha_k^*, \quad \forall p \in P.$$

Note that X_j s are subspaces orthogonal to each other.

Rank Reduction

Apply the same 1-randomized rounding to X_j , independently, to generate \hat{x}_j such that

$$I \bullet \hat{x}_j \hat{x}_j^T = I \bullet X_j \geq \frac{1}{2}$$

and

$$pp^T \bullet \hat{x}_j \hat{x}_j^T \leq O(\log n) \cdot pp^T \bullet X_j, \forall p \in P, j = 1, \dots, k.$$

Note that \hat{x}_j s are orthogonal to each other, and $\|\hat{x}_j\|^2 \geq 1/2$ for all j .

Normalizing every \hat{x}_j such that $\|\hat{x}_j\|^2 = 1$, we still have

$$pp^T \bullet \hat{x}_j \hat{x}_j^T \leq O(\log n) \cdot pp^T \bullet X_j, \quad \forall p \in P, j = 1, \dots, k.$$

Finally

$$\begin{aligned} \sum_{j=1}^k pp^T \bullet \hat{x}_j \hat{x}_j^T &\leq O(\log n) \cdot \sum_{j=1}^k pp^T \bullet X_j \\ &\leq O(\log n) \cdot \alpha_k^*, \quad \forall p \in P. \end{aligned}$$

2-Catalog Segmentation

- **Input** A bipartite graph $G = (A, B, E)$ with $|A| = n$ and $|B| = m$,
- **Output** A partition of $B = B_1 \cup B_2$, and two subsets A_1, A_2 of A such that $|A_1| = |A_2| = k$
- **Objective** To maximize $w(A_1, B_1) + w(A_2, B_2)$.

$A \longleftrightarrow I$ and $B \longleftrightarrow$ Customers;

A_1 and A_2 are the two catalogs one of which will be sent to each of the customers such that the (total) satisfaction is maximized.

Figure 3: Graph Representation

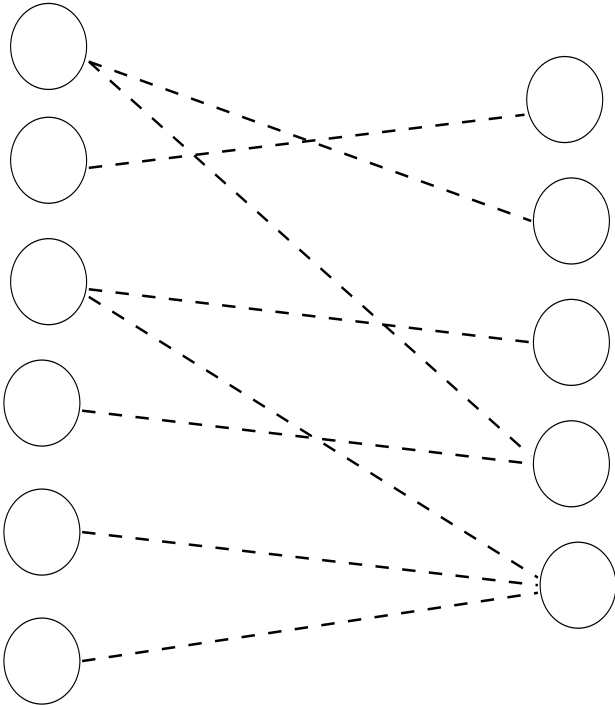


Figure 4: 2-Catalog Segmentation

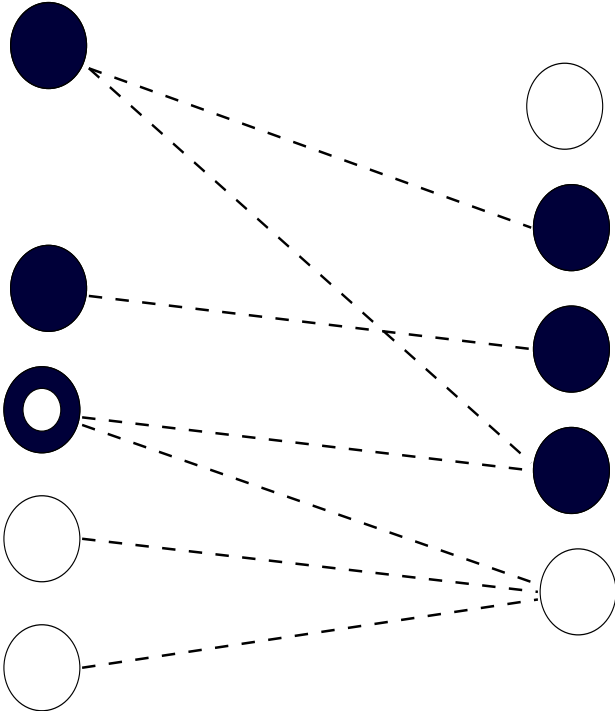
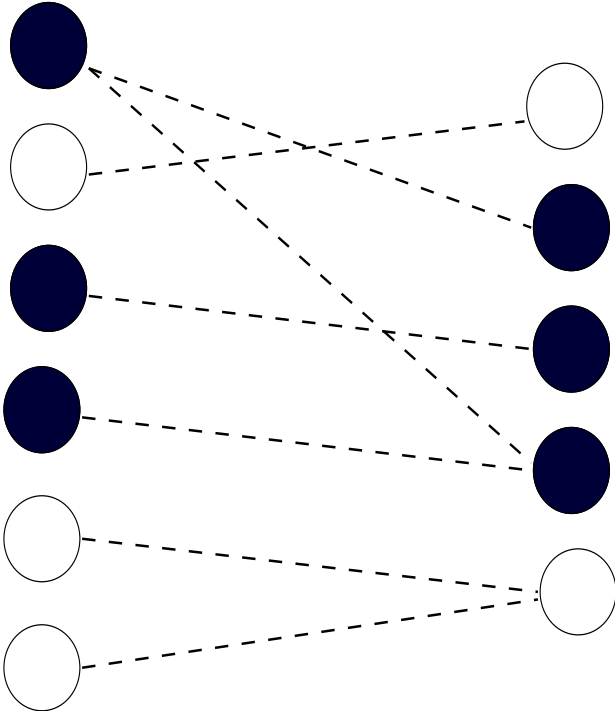


Figure 5: Disjoint 2-Catalog Segmentation



Previous Results

- The problem was proposed by Kleinberg, Papadimitriou and Raghavan (1998).
- A simple greedy algorithm with performance guarantee (KPR 1998) simply selects $A_1 \in A$ to be the k nodes of largest degrees, $A_2 \in A$ be any k nodes in A , $B_1 = B$, $B_2 = \emptyset$.
- A polynomial time approximation scheme (KPR 1998) for all dense instances of 2-CatSP, in which each node of B has a degree of a constant fraction of n .
- 0.56-approximation algorithm for the case $k = n/2$ by Dodis, Guruswami and Khanna (1999).

Our Results (Xu, Ye and Zhang)

A randomized approximation algorithm based on the semidefinite programming relaxation which has performance guarantee

- $\frac{1}{2}$ for all $k \in [1, n]$;
- strictly better than $\frac{1}{2}$ when $k \geq \frac{n}{3}$;
- 0.67 when $k = \frac{n}{2}$.

Key Techniques

- A new integer programming formulation which allows us to obtain a SDP relaxation.
 - The main difficulty was that $A_1 \cap A_2 = \emptyset$ is not required.
 - The new IP model uses two-dimensional vector variables.
- A new variants of the Goemans-Williamson rounding technique which allows us to take the advantage of the underline structure of the problem
 - the graph is bipartite;
 - the cardinality constraints are on the set of A only

A Special Case

- $k = \frac{n}{2}$ and it is required that $A_1 \cap A_2 = \emptyset$ (Disjoint Case).
- Each of the node in A is either in A_1 or A_2 . For $i \in A$, $x_i = 1$ iff $i \in A_1$, otherwise $x_i = -1$; for $j \in B$, $x_j = 1$ iff $j \in B_1$.
- The problem can be formulated as

$$w_* = \text{Maximize } \frac{1}{2} \sum_{i \in A, j \in B} w_{ij} (1 + x_i x_j) \tag{1}$$

$$\text{subject to } \sum_{i \in A} x_i = 0,$$

$$x_i^2 = x_j^2 = 1 \quad i \in A, j \in B$$

Relationship with Max-Bisection

- This special case is very similar to the well-known Max-Bisection problem

$$w_* = \text{Maximize } \frac{1}{2} \sum_{i \in A, j \in B} w_{ij} (1 - x_i x_j) \tag{2}$$

$$\text{subject to } \sum_i x_i = 0, \\ x_i^2 = 1$$

- Is fairly easy to solve since the known SDP based approximation algorithms can also applied to this special case of 2-Catalog problem (DGK 1999).
- 0.65-approximation by the result of Frieze and Jerrum (1995).
0.699-approximation by the results of Ye (1999) in running time $O(n^{3.5})$,
and 0.701 by Halperin and Zwick (2001) in $O(n^{9.5})$.

A Harder Special Case

- $k = \frac{n}{2}$ but $A_1 \cap A_2 = \emptyset$ is not required (General Case).
- One dimensional binary variables are not sufficient to model this problem.
- DGK (1999) reduce this problem to the Disjoint Case, and show that the 0.651-approximation for the latter implies a 0.56-approximation for the General Case.
- If we use the 0.70-approximation for Disjoint case, we get a 0.58-approximation for the General Case.

General Case

- Each node i of A has four possible choices $A_1 \setminus A_2$, $A_2 \setminus A_1$, $A_1 \cap A_2$ and $A \setminus (A_1 \cup A_2)$.
- One binary variable cannot characterize all choices of node i .
- We design a two-dimensional vector (x_i, y_i) for each node $i \in A$ such that $x_i = 1$ iff $i \in A_1$, otherwise $x_i = -1$; and $y_i = 1$ iff $i \in A_2$, otherwise $y_i = -1$. We assign variable z_j for each node $j \in B$ such that $z_j = 1$ iff $i \in B_1$, otherwise $z_j = -1$.

An Integer Programming Formulation

A quadratic formulation can be obtained by using these vector variables together with an additional reference variable u_0 , which makes the objective function and constraints homogeneous

$$w_* = \text{Maximize } \frac{1}{4} \sum_{i \in A, j \in B} w_{ij} (2 + u_0 x_i + u_0 y_i + x_i z_j - y_i z_j)$$

$$\text{subject to } \sum_{i \in A} u_0 x_i = \sum_{i \in A} u_0 y_i = 2k - n,$$

$$u_0^2 = x_i^2 = y_i^2 = z_j^2 = 1 \quad i \in A, j \in B$$

(3)

SDP Relaxation

Let vector $u = (u_0; x \in R^n; y \in R^n; z \in R^m)$ and consider matrix $U = uu^T$. Then, it is straightforward to obtain an SDP relaxation of (3)

$$\text{Max} \quad \sum_{1 \leq i \leq n, 2n+1 \leq j \leq 2n+m} w_{ij} \frac{(2 + U_{0i} + U_{0(n+i)} + U_{ij} - U_{(n+i)j})}{4}$$

$$\begin{aligned} \text{s.t.} \quad & \sum_{i=1}^n U_{0i} = \sum_{i=n+1}^{2n} U_{0i} = 2k - n, \\ & \sum_{1 \leq i, j \leq n} U_{ij} = \sum_{n+1 \leq i, j \leq 2n} U_{ij} = (2k - n)^2, \\ & U_{ii} = 1, \quad i = 0, 1, \dots, 2n + m \\ & U \succeq 0. \end{aligned}$$

(4)

A High-Level Algorithm

- Solve the SDP relaxation and obtain an optimal matrix \bar{U} which can be write as

$$\bar{U} = \begin{pmatrix} \bar{U}_{00} & \bar{U}_{0x} & \bar{U}_{0y} & \bar{U}_{0z} \\ \bar{U}_{x0} & \bar{U}_{xx} & \bar{U}_{xy} & \bar{U}_{xz} \\ \bar{U}_{y0} & \bar{U}_{yx} & \bar{U}_{yy} & \bar{U}_{yz} \\ \bar{U}_{z0} & \bar{U}_{zx} & \bar{U}_{zy} & \bar{U}_{zz} \end{pmatrix}$$

according to sub-blocks corresponding to u_0 , x , y , and z .

- Use some (randomized) rounding technique to obtain a $\{-1, 1\}$ solution \hat{x}, \hat{y}, z, u which produce $\hat{A}_1, \hat{A}_2, B_1, B_2$. (The solution may not be feasible, i.e, the cardinality constraints may not be satisfied.)
- Use a greedy adjusting procedure to get a feasible solution A_1, A_2, B_1, B_2 from the above solution.

Goemans-Williamson Rounding Technique

- Factorize the optimal matrix $\bar{U} = V^T V$ where $V = (v_0, V_x, V_y, V_z)$ is a $(2n + m + 1) \times (2n + m + 1)$ matrix.
- Randomly choose a unit vector u from S^{2n+m} .
- For each node i , if $u \cdot v_i$ has the same sign as $u \cdot v_0$, then it is 1; otherwise it is -1 .
- This randomized rounding produces a solution which has an expected objective value close to the optimal one, and the expected number of nodes in \hat{A}_1 or \hat{A}_2 are close to k as desired.
- **The problem is that the variances of $|\hat{A}_1|$ or $|\hat{A}_2|$ could be very large.**

A Combined Rounding

- The identity matrix I can be used as the rounding matrix which has a better bound on the expected values and variances of $|\hat{A}_1|$ or $|\hat{A}_2|$, but a worse bound on the objective value. (For simplicity, we only consider $k = \frac{n}{2}$ here).
- We have shown that a careful combination of the optimal matrix \bar{U} and the identity matrix I can improve the overall approximation ratio, i.e., to balance the bounds on the objective value and the sizes.
- We can apply this combined rounding technique to our problem.
- Can we do better by exploiting the structure of the problem?

Structure of the Problem

- Recall that our graph is bipartite and the cardinality constraints are only on the set A .
- For the nodes in A , we can apply combined rounding matrix.
- For the nodes in B , we do not want to use the combined rounding since it has a negative effect on the objective value.
- A new rounding matrix may take advantage of this fact. But note that it must be semi-definite.

New Rounding Matrix

We use the matrix $\bar{U}(\theta) + (1 - \theta)P$ as the rounding matrix, where $\theta \in [0, 1]$,

$$\bar{U}(\theta) = \begin{pmatrix} \bar{U}_{00} & \sqrt{\theta}\bar{U}_{0x} & \sqrt{\theta}\bar{U}_{0y} & \bar{U}_{0z} \\ \sqrt{\theta}\bar{U}_{x0} & \theta\bar{U}_{xx} & \theta\bar{U}_{xy} & \sqrt{\theta}\bar{U}_{xz} \\ \sqrt{\theta}\bar{U}_{y0} & \theta\bar{U}_{yx} & \theta\bar{U}_{yy} & \sqrt{\theta}\bar{U}_{yz} \\ \bar{U}_{z0} & \sqrt{\theta}\bar{U}_{zx} & \sqrt{\theta}\bar{U}_{zy} & \bar{U}_{zz} \end{pmatrix}$$

and

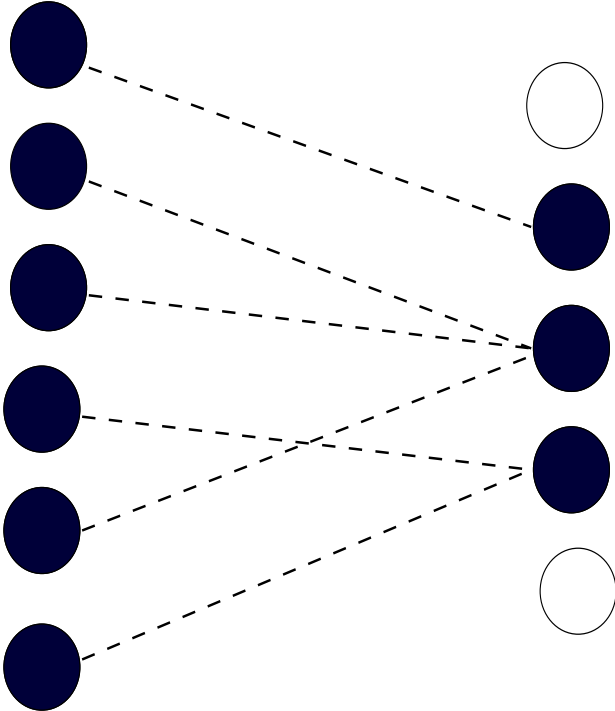
$$P = \begin{pmatrix} 0 & 0_{1 \times n} & 0_{1 \times n} & 0_{1 \times m} \\ 0_{n \times 1} & I_{n \times n} & 0_{n \times n} & 0_{n \times m} \\ 0_{n \times 1} & 0_{n \times n} & I_{n \times n} & 0_{n \times m} \\ 0_{m \times 1} & 0_{m \times n} & 0_{m \times n} & 0_{m \times m} \end{pmatrix}$$

Facility Location

In the uncapacitated facility location problem (UFLP), we have

- A set \mathcal{F} of n_f facilities, where for every facility $i \in \mathcal{F}$, a nonnegative number f_i is given as the *opening cost* of facility i .
- A set \mathcal{C} of n_c cities, where for every city $j \in \mathcal{C}$ and facility $i \in \mathcal{F}$, we have a *connection cost* (a.k.a. service cost) c_{ij} between city j and facility i .
- The objective is to open a subset of the facilities in \mathcal{F} , and connect each city to an open facility so that the total cost is minimized.
- We will consider the *metric* version of this problem, i.e., the connection costs satisfy the triangle inequality.

Figure 6: Uncapacitated Facility Location



Does SDP Help?

- The similar bipartite graph structure.
- The problem is simple if know which facilities to open.
- Binary variables can be used to represent facilities.

We cannot get it to work, so as to try others...

Binary integer program

$$\begin{aligned}
 &\text{Minimize} && \sum_{i \in \mathcal{F}} f_i y_i + \sum_{i \in \mathcal{F}} \sum_{j \in \mathcal{C}} c_{ij} x_{ij} \\
 &\text{subject to} && \sum_{i \in \mathcal{F}} x_{ij} = 1 \quad \text{for each } j \in \mathcal{C}, \\
 &&& x_{ij} \leq y_i \quad \text{for each } i \in \mathcal{F}, j \in \mathcal{C}, \\
 &&& x_{ij} \in \{0, 1\} \quad \text{for each } i \in \mathcal{F}, j \in \mathcal{C}, \\
 &&& y_i \in \{0, 1\} \quad \text{for each } i \in \mathcal{F}.
 \end{aligned} \tag{5}$$

This problem is denoted as $FLP(c, f)$.

Approximation Results

| approx. factor | reference | technique/running time |
|----------------|------------------------------|---|
| $O(\ln n_c)$ | Hochbaum | greedy algorithm/ $O(n^3)$ |
| 3.16 | Shmoys, Tardos, Aardal | LP rounding |
| 2.41 | Guha and Khuller | LP rounding + greedy |
| 1.736 | Chudak | LP rounding |
| $5 + \epsilon$ | Korupolu, Plaxton, Rajaraman | local search/ $O(n^6 \log(n/\epsilon))$ |
| 3 | Jain and Vazirani | primal-dual method/ $O(n^2 \log n)$ |
| 1.853 | Charikar and Guha | primal-dual method + greedy/ $O(n^3)$ |
| 1.728 | Charikar and Guha | LP rounding + primal-dual method + greedy |
| 1.861 | Mahdian et al. | greedy algorithm/ $O(n^2 \log n)$ |
| 1.61 | Jain et al. | greedy algorithm/ $O(n^3)$ |
| 1.582 | Sviridenko | LP rounding |
| 1.517 | Mahdian, Ye, Zhang | greedy algorithm + greedy/ $O(n^3)$ |

Table 1: Approximation Algorithms for UFLP

Hardness Results

Guha and Khuller proved that it is impossible to get an approximation guarantee of 1.463 for the uncapacitated metric facility location problem, unless $\text{NP} \subseteq \text{DTIME}[n^{O(\log \log n)}]$.

Cost-Splitting Approximation

An algorithm is called a (γ_f, γ_c) -approximation algorithm for UFLP, if for every instance \mathcal{I} of UFLP, and for every solution SOL for \mathcal{I} with facility cost F_{SOL} and connection cost C_{SOL} , the cost of the solution found by the algorithm is at most $\gamma_f F_{SOL} + \gamma_c C_{SOL}$.

Let $\gamma_f \geq 1$. Then $\gamma_c \leq \sup_k \{z_k\}$, where z_k

$$\max \frac{\sum_{i=1}^k \alpha_i - \gamma_f f}{\sum_{i=1}^k d_i}$$

$$\text{s.t. } \forall 1 \leq i < k \quad \alpha_i \leq \alpha_{i+1}$$

$$\forall 1 \leq j < i < k \quad r_{j,i} \geq r_{j,i+1}$$

$$\forall 1 \leq j < i \leq k \quad \alpha_i \leq r_{j,i} + d_i + d_j$$

$$\forall 1 \leq i \leq k \quad \sum_{j=1}^{i-1} \max(r_{j,i} - d_j, 0) + \sum_{j=i}^k \max(\alpha_i - d_j, 0) \leq f$$

$$\forall 1 \leq j \leq i \leq k \quad \alpha_j, d_j, f, r_{j,i} \geq 0.$$

Overhead Charge δ

We solve $FLP(c, \delta \cdot f)$ instead of $FLP(c, f)$. After a solution is obtained, gradually reduce δ to 1 while look for more facilities to open.

Theorem 1 *If there is a*

$$(\gamma_f, \gamma_c)$$

approximation algorithm for UFLP, then there is a

$$\left(\gamma_f + \ln \delta, 1 + \frac{\gamma_c - 1}{\delta}\right)$$

approximation for UFLP.

Prove $(\gamma_f, \gamma_c) = (1.104, 1.7805)$ and select $\delta = 1.5107$.

Extended Results

| Problem | Previous ratio | Our ratio | Reduced Problem |
|---------|----------------|-----------|-----------------|
| SFLP | 3 | 2 | LFLP |
| K-FLP | 3 | | LP |
| | 4.83 | 3.27 | UFLP |
| 2-FLP | 4.83 | 1.77 | UFLP |
| 3-FLP | 4.83 | 2.51 | UFLP |
| 4-FLP | 4.83 | 2.81 | UFLP |
| SMFLP | 9 | 3.98 | UFLP |
| CCFLP | 4.59 | 3.97 | UFLP |

Table 2: Approximation Algorithms for FLP Variants