# Approximation Algorithms using ILP

Subhas C. Nandy
(nandysc@isical.ac.in)

Advanced Computing and Microelectronics Unit
Indian Statistical Institute
Kolkata 700108, India.

# Organization

# Introduction

- A problem is said to be tractable if there exists an algorithm for that problem whose worst case time complexity is a polynomial function in $n$ (the input size).
- Otherwise, the problem is said to be intractable.

# Introduction

- A problem is said to be tractable if there exists an algorithm for that problem whose worst case time complexity is a polynomial function in $n$ (the input size).
- Otherwise, the problem is said to be intractable.

## Complexity Classes

P: A decision problem that can be solved in polynomial time.

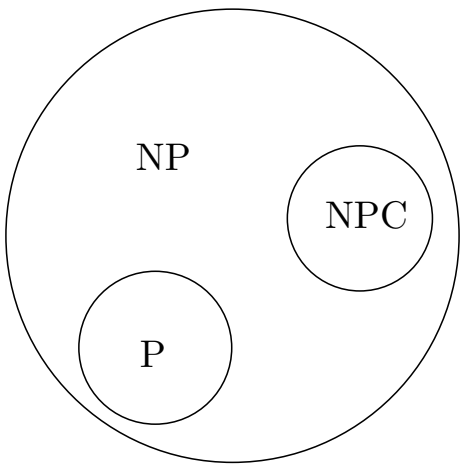
NP: A decision problem that, for an input, can be verified in polynomial time.

# NP-Completeness

## An important question

Whether **P = NP** or **P** $\neq$ **NP** ?

NP-Complete Class: A class of decision problems in NP such that if one of them can be solved in polynomial time, all other problems can also be solved in polynomial time.



# NP-Complete Problems

- Circuit satisfiability,
- 3-SAT (satisfiability where the expression is in CNF, and each clause is of length at most 3),
- Clique decision problem,
- Vertex Cover decision,
- Travelling salesman problem,
- Knapsack problem,
- Bin packing
- Art gallery problem

to name only a few.

# Handling the NP-Hard Problems

## Brute-force algorithm

Design a clever enumeration strategy, so that

- it guarantees the optimality of the solution, but
- no guarantee on the running time.

# Handling the NP-Hard Problems

## Brute-force algorithm

Design a clever enumeration strategy, so that

- it guarantees the optimality of the solution, but
- no guarantee on the running time.

## Heuristic

Develop an intuitive algorithm, such that

- it guarantees to run in polynomial time, but
- no guarantee on the quality of the solution.

# Approximation Algorithm

## Approximation Algorithm

- Guaranteed to run in polynomial time, and
- guaranteed to produce a "high quality" solution.

# Approximation Algorithm

## Approximation Algorithm

- Guaranteed to run in polynomial time, and
- guaranteed to produce a "high quality" solution.

**But, what is a high quality solution?**

# Approximation Algorithm

## Approximation Algorithm

- Guaranteed to run in polynomial time, and
- guaranteed to produce a "high quality" solution.

**But, what is a high quality solution?**

**Is the solution** $= OPT + \alpha$, or
$\alpha \times OPT$, or
$(1 + \epsilon) \times OPT$

where,
$OPT \rightarrow$ the optimum solution,
$\alpha \rightarrow$ a constant,
$\epsilon \rightarrow$ a very small constant.

# Approximation Algorithm

**Note:** Designing approximation algorithm for a problem having polynomial time algorithm is also relevant if the time complexity of the problem is very high.

# Approximation Algorithm

**Note:** Designing approximation algorithm for a problem having polynomial time algorithm is also relevant if the time complexity of the problem is very high.

**Difficulty:** One needs to prove that the solution is close to optimum, without knowing the optimum solution.

# Approximation Algorithm

## $f(n)$-approximation Algorithm

Let $P$ be a problem of size $n$. An algorithm $\mathcal{A}$ for a problem $P$ is said to be an $f(n)$-approximation algorithm if and only if for every instance $I$ of $P$, $|\mathcal{A}(I) - OPT|/OPT \leq f(n)$. It is assumed that $OPT \geq 0$.

**Note:** $f(n)$ can be a constant also.

# Approximation Algorithm

## $f(n)$-approximation Algorithm

Let $P$ be a problem of size $n$. An algorithm $\mathcal{A}$ for a problem $P$ is said to be an $f(n)$-approximation algorithm if and only if for every instance $I$ of $P$, $|\mathcal{A}(I) - OPT|/OPT \leq f(n)$. It is assumed that $OPT \geq 0$.

**Note:** $f(n)$ can be a constant also.

## Performance ratio:

$$f(n) = \begin{cases} \mathcal{A}(I)/OPT & \text{in case } P \text{ is a minimization problem} \\ OPT/\mathcal{A}(I) & \text{in case } P \text{ is a maximization problem} \end{cases}$$

We will show the use of ILP in designing Approximation Algorithms

# Integer Linear Programming

## The Problem

Objective Function: Maximize $8x_1 + 11x_2 + 6x_3 + 4x_4$

Subject to:

$$5x_1 + 7x_2 + 4x_3 + 3x_4 \leq 14$$

$$7x_1 + 2x_3 + x_4 \leq 10$$

$$x_i \in \mathbb{Z}^+ \text{ for all } i = 1, 2, 3, 4$$

# Vertex Cover Problem

**Input:** $G = (V, E) \longrightarrow$ a graph, where each vertex is associated a positive weight.
**Output:** A subset $C \subseteq V$ such that every edge $e \in E$ is covered by some vertex in $C$, and $w(C)$ is minimized.

## ILP Formulation

Minimize: $\sum_{v \in V} w_v x_v$
Subject to: $x_u + x_v \geq 1$ for each edge $e = (u, v) \in E$
$x_v \in \{0, 1\}$ for all $v \in V$

## LP Relaxation

Minimize: $\sum_{v \in V} w_v x_v$
Subject to: $x_u + x_v \geq 1$ for each edge $e = (u, v) \in E$
$0 \leq x_v \leq 1$ for all $v \in V$

This LP can be solved in polynomial time
Let the optimum solution be $OPT_f$

# Vertex Cover Problem

Finally, round the fractional solution $x$ to an integer solution

$$x^* : C = \{v | x_v \geq \frac{1}{2}\}$$

**Observation 1**

$C$ is a feasible solution of the vertex cover problem.

Follows from the fact: every edge $e = (u, v)$ satisfies $x_u + x_v \geq 1$

**Observation 2**

$w(C) \leq 2OPT_f \leq 2OPT$

$$\begin{aligned} w(C) &= \sum_{v \in C} w(v) \\ &= \sum_{v \in V | x_v \geq 1/2} w(v) \\ &\leq 2 \times \sum_{v \in V | x_v \geq 1/2} w(v)x(v) \\ &\leq 2 \times \sum_{v \in V} w(v)x(v) \\ &= 2OPT_f \leq 2OPT. \end{aligned}$$

# Integrality Gap

Given a set of instances $\mathcal{I}$ of the problem $P$

**The integrality gap is defined by**

$sup_{I \in \mathcal{I}} \frac{OPT(I)}{OPT_f(I)}$ for minimization problem, and $inf_{I \in \mathcal{I}} \frac{OPT(I)}{OPT_f(I)}$ for maximization problem.

**Note:** We can not get an approximation factor better than the integrality gap.

# Integrality Gap

## Vertex cover problem has integrality gap $2 - \frac{2}{n}$

- Let $G$ be a complete graph with $n$ vertices
- $OPT_f = \frac{n}{2}$
- However, the optimum solution of the vertex cover problem is $n - 1$

Thus, the Integrality Gap is $\frac{n-1}{n/2} = 2 - frac2n$

## Our approximation bound is Tight

- Consider a bipartite graph $G = (V_1 \cup V_2, E)$
- Solve the corresponding LP and $OPT_f$ satisfies $x_v = 1/2$ for all $v \in V$
- This implies, the solution of our Heuristic is $n$
- But, $OPT = \frac{n}{2}$

# Set Cover - Randomized Rounding

## Set Cover Problem

Input: A universe of elements $U = \{u_1, u_2, \ldots, u_n\}$,
a family of subsets $\mathcal{T} = \{S_1, S_2, \ldots S_m\}$, and
a cost function $c : T \longrightarrow R^+$

Objective: To find a collection of subsets that cover all the elements in

## ILP Formulation

Minimize: $\sum_{S \in \mathcal{T}} c(S) x_S$

Subject to: $\sum_{S:e \in S} x_S \geq 1$ for each element $e \in U$
$x_S \in \{0, 1\}$ for all $S \in \mathcal{T}$

# Set Cover - Randomized Rounding

## LP Relaxation

Minimize: $\sum_{S \in \mathcal{T}} c(S) x_S$

Subject to: $\sum_{S: e \in S} x_S \geq 1$ for each element $e \in U$

$0 \leq x_S \leq 1$ for all $S \in \mathcal{T}$

Let $x^*$ be the optimum solution of the LP.

- Unlike the case of vertex cover, we cannot round $x_i^*$ to its nearest integer.
  Reason: If an element $u$ belongs to 100 sets, it could be that $x_i = 1/100$ for each of those sets, and we would be rounding all those numbers to zero, leaving the element u not covered.
- However, if we know that every element $u$ belongs to at most $k$ sets, then we can round the numbers $\geq 1/k$ to 1, and the numbers $< 1/k$ to zero.

# Set Cover - Randomized Rounding

- This would give a feasible cover, and we can prove that we achieve a $k$-approximation.
- Unfortunately, $k$ could be very large, even $n/2$. So, a $k$-factor approximation is not desirable.

We try to have an approximation guarantee that is never worse than $O(\log n)$.

# Set Cover - Randomized Rounding

Assume $S$ is included in the set-cover with probability $x_S$.

Thus, Expected Cost $E(cost) = \sum_{S \in \mathcal{T}} c(S)x_S = OPT_f$

Remark: No guarantee that the solution obtained is feasible.

**Result**

The randomized rounding covers each element with probability at least $1 - \frac{1}{e}$.

**Proof:**

- Consider an element $a$, which can be covered by $k$ sets.
- Each of which is selected in the algorithm with probability $p_1, p_2, \ldots, p_k$
- Thus, $\sum_{i=1}^{k} p_i \geq 1$
- Prob[$a$ is not covered] $= \Pi_{1 \leq i \leq k}(1 - p_i) \leq (1 - \frac{1}{k})^k \leq \frac{1}{e}$.
- Prob[a is covered] $= 1 - \frac{1}{e}$

# Set Cover - Randomized Rounding

We repeat this procedure $d \log n$ time, and report the union of set covers.

Now, we have

- Prob[$a$ is not covered] $= (\frac{1}{e})^{d \log n} \leq \frac{1}{4n}$, for some appropriate choice of $d$
- Prob[at least one element is not covered] $\leq \frac{1}{4}$, by applying union bound

$\Rightarrow$ Prob[every element is covered] $\geq \frac{3}{4}$

- Total expected cost $E[cost] = d \log n \times OPT_f$
- Now, applying Markov inequality, we have

$$\text{Prob}[\text{cost} > 4d \log n \times OPT_f] \leq \frac{d \log n \times OPT_f}{4d \log n \times OPT_f} = \frac{1}{4}$$

- Prob[cost $\leq 4d \log n \times OPT_f] \geq \frac{3}{4}$

# Set Cover - Randomized Rounding

## Problems in this method

- Solution might be infeasible
- Solution may be of very high cost

However, probability of both these events is upper bounded by $\frac{1}{4}$. Implying, their union is upper bounded by $\frac{1}{2}$.

## Result

With probability at least $\frac{1}{2}$, one can get $4d \log n$-approximation of the set cover problem using $d \log n$ iteration of randomized rounding.

# Set Cover - Randomized Rounding

## A feasible algorithm

Repeat the iterations until both the conditions (feasibility and targetted cost) are satisfied.

- This may not terminate in polynomial time.
- Although the expected running time is polynomial.
- This class of algorithms belong to the complexity class $ZPP$, which consists of all languages which can be decided in expected polynomial time.

# Set Cover - Randomized Rounding

## A feasible algorithm

Repeat the iterations until both the conditions (feasibility and targetted cost) are satisfied.

- This may not terminate in polynomial time.
- Although the expected running time is polynomial.
- This class of algorithms belong to the complexity class $ZPP$, which consists of all languages which can be decided in expected polynomial time.

## Integrality Gap of LP-relaxation for Set Cover

$$Sup_G \frac{OPT(G)}{OPT_f(G)} \leq 4d \log n$$

# Rectangle Stabbing Problem[1]

**Problem Statement:**

**Given a set $\mathcal{R}$ of $n$ axis-parallel rectangles, find the minimum number of axis-parallel lines to stab all the members in $\mathcal{R}$.**

**A rectangle $r \in \mathcal{R}$ is given using a pair of coordinates $[(a, b), (c, d)]$ corresponding to its (bottom-left, top-right) diagonal.**

For the sake of simplicity, we assume that the coordinates are integer valued.

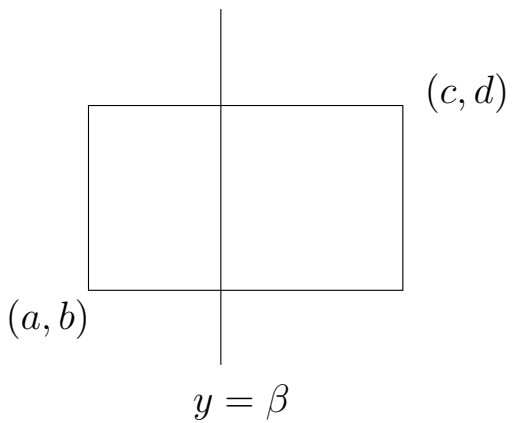
[1] Gaur, Ibaraki and Krishnamurthy, J. of Algorithms, 2002

# Rectangle Stabbing Problem

**Definition**

**An axis-parallel (horizontal/vertical) line $\ell$ stabs a rectangle $r = [(a, b), (c, d)] \in \mathcal{R}$ if $\ell$ passes through the interior of the rectangle $r$. if**

**Example:**




**Here the line $y = \beta$ stabs the rectangle.**
**This implies $a + 1 \leq \beta \leq c - 1$**

# Status of the Problem and Our Objective

**Status:**

**The problem is NP-hard**

Reference: Hasin and Megiddo, Discrete Appl. Math., 1991

**Objective:** To design a constant factor approximation algorithm

**Result available:**

**A 2-factor approximation algorithm that runs in $O(n^5)$ time.**

**Tools used:** LP-relaxation.

# Integer Programming Formulation

**Solution Space:** $H \bigcup V$.

$H$ – Set of $2n$ horizontal lines at $y = b_i + \epsilon$ and $y = d_i - \epsilon$, where $b_i$ and $d_i$ are $y$-coordinates of bottom and top boundaries of $i$-th rectangle.

$V$ – Set of $2n$ vertical lines at $x = a_i + \epsilon$ and $y = c_i - \epsilon$, where $a_i$ and $c_i$ are $x$-coordinates of left and right boundaries of $i$-th rectangle.

Take 4n decision variables, namely $x_1, x_2, \ldots, x_{2n}$, $y_1, y_2, \ldots, y_{2n}$.

$x_i$ – corresponds to $i$-th vertical line, and

$y_j$ – corresponds to $j$-th horizontal line.

These decision variables can assume values 0 and 1 only.

# Integer Programming Formulation

$H_k$ – Set of horizontal lines that stab the rectangle $r_k$, and

$V_k$ – Set of vertical lines that stab the rectangle $r_k$.

**Integer Programming Problem – $P$**

**Objective Function:**

min $\sum_{i \in V} x_i + \sum_{j \in H} y_j$

**Constraints:**

For each rectangle $r_k$, $k = 1, 2, \ldots, n$, we have the constraint

$\sum_{i \in V_k} x_i + \sum_{j \in H_k} y_j \geq 1$

$x_i \in [0, 1]$, for all $i = 1, 2, \ldots, 2n$, and

$y_j \in [0, 1]$, for all $j = 1, 2, \ldots, 2n$.

# LP Relaxation

## Linear Programming Problem – $\bar{P}$

**Objective Function:**

$$\min \sum_{i \in V} x_i + \sum_{j \in H} y_j$$

**Constraints:**

$$\sum_{i \in V_k} x_i + \sum_{j \in H_k} y_j \geq 1, \quad \text{for all } k = 1, 2, \ldots, n$$

$$x_i \geq 0, \quad \text{for all } i = 1, 2, \ldots, 2n, \text{ and}$$

$$y_j \geq 0, \quad \text{for all } j = 1, 2, \ldots, 2n.$$

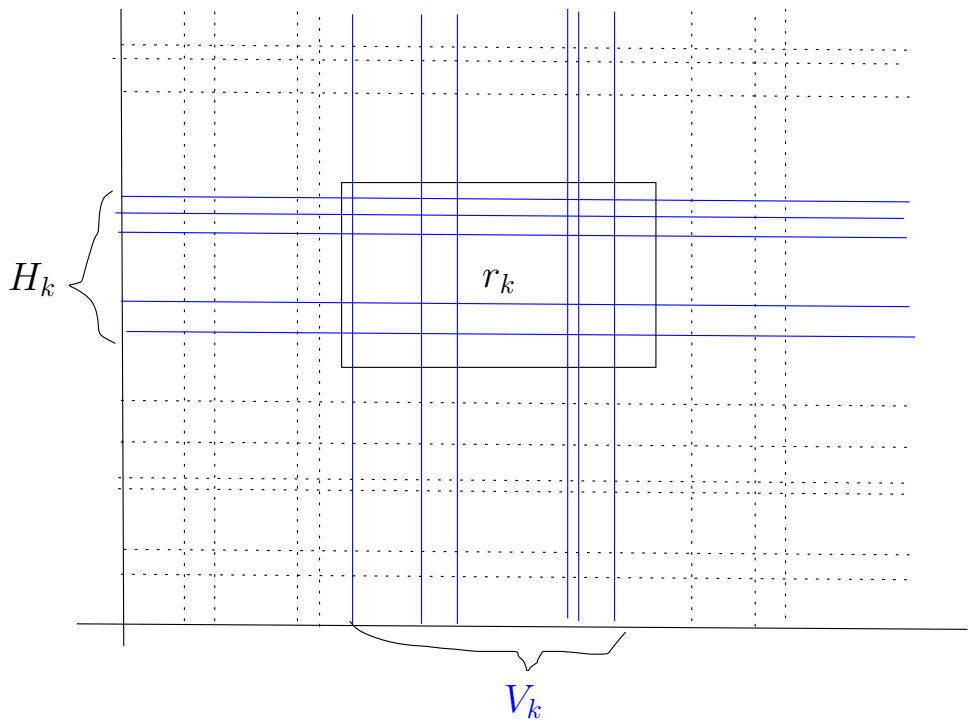
# Analysis of LP Solution

Let $\bar{x}_i, i = 1, 2, \ldots, 2n$ and $\bar{y}_j, j = 1, 2, \ldots, 2n$ be an optimal fractional solution of the LP problem $\bar{P}$.

For each rectangle $r_k$, $k = 1, 2, \ldots, n$, we have

either $\sum_{i \in V_k} \bar{x}_i \geq \frac{1}{2}$

or $\sum_{j \in H_k} \bar{y}_j \geq \frac{1}{2}$.



# Analysis of LP Solution

Let $R_H$ be the set of all $k$ such that $\sum_{i \in V_k} \bar{x}_i \geq \frac{1}{2}$, and

$R_V$ be the set of all $k$ such that $\sum_{j \in H_k} \bar{y}_j \geq \frac{1}{2}$.

**Implication:**

- The set of rectangles in $R_H$ will be stabbed by horizontal lines, and
- The set of rectangles in $R_V$ will be stabbed by vertical lines.

Thus, we have the following two problems

$P_H$: Compute the minimum clique cover of an interval graph with the vertical intervals corresponding to the rectangles in $R_H$, and

$P_V$: Compute the minimum clique cover of an interval graph with the horizontal intervals corresponding to the rectangles in $R_V$.

**$P_H$ and $P_V$ can be optimally solved in polynomial time.**

# Analysis of Approximation Factor

Let

$Q$: Optimum solution of the integer programming problem $P$,

$\hat{Q}$: Optimum solution of the linear programming problem $\hat{P}$,

$Q_H$: Optimum solution of the clique cover problem $P_H$,

$Q_V$: Optimum solution of the clique cover problem $P_V$.

**Theorem**

$Q_H + Q_V \leq 2Q$.

# Analysis of Approximation Factor

**Proof:** Let $\hat{Q} = (\hat{x}, \hat{y})$ be an optimal fractional solution of $\hat{P}$

$\Longrightarrow Q_V^* = 2\hat{x}$ and $Q_H^* = 2\hat{y}$ are feasible solutions of $P_H$ and $P_V$.

**Reason:** For every $k \in R_H$, we have $\sum_{i \in H_k} \hat{y}_i \geq \frac{1}{2}$
$\Longrightarrow \sum_{i \in H_k} 2\hat{y}_i \geq 1$.

We have,

- $Q_H + Q_V \leq Q_H^* + Q_V^*$
  [since $Q_H$ and $Q_V$ are optimum solutons and $Q_H^*$ and $Q_V^*$ are feasible solutions of the same minimization problems.]
- $Q_H^* + Q_V^* = 2(\hat{x} + \hat{y}) = 2\hat{Q}$.
- $\hat{Q} \leq Q$
  [Since optimum solution of an LP minimization problem is less the optimum solution of its corresponding IP problem]

Thus, we have the proof of the result.

# Uncapacited Facility Location

## The Problem

Input:

- A set $V$ of demand location,
- a set $F$ of facility location,
- a cost function $c : (V \cup F) \times (V \cup F) \longrightarrow Q^+$,
- cost $f_j$ of opening a facility $j \in F$, and
- a demand $d_i$ of client $i \in V$

Goal

- To find a subset $S \subseteq F$
  such that $\sum_{j \in S} f_j + \sum_{i \in V} c_{i\sigma_i} d_i$ is minimized.

# Uncapacited Metric Facility Location

Metric Condition $c_{ij} = c_{ji}$, $c_{ij} \leq c_{ik} + c_{kj}$, $c_{ii} = 0$

- There is a bound on total demand (or the no. of clients) a facility can serve, and ,
- There is a bound on the number of facilities that can be opened.

**STATUS: NP-COMPLETE**

# ILP Formulation

For any $j \in F$, define a variable $y_j$, where

$$y_j = 1 \quad \text{if facility } j \text{ is opened,}$$
$$= 0 \quad \text{otherwise}$$

**Assumption:** A client is assigned to its nearest facility

For each client $i \in V$, $\sigma(i) =$ nearest $j$ such that $y_j = 1$

Define $x_{ij} = 1$ if $i \in V$ is served by $j \in F$
$= 0$ otherwise

## The ILP

Minimize $\sum_{j \in F} f_j y_j + \sum_{i \in V} x_{ij} c_{ij}$
Subject to

- $x_{ij} \leq y_j$ for all $i \in V$, $j \in F$
- $\sum_{j \in V} x_{ij} \geq 1$ for all $i \in V$

# LP Relaxation

Minimize $\sum_{j \in F} f_j y_j + \sum_{i \in V} x_{ij} c_{ij}$
Subject to

- $x_{ij} \leq y_j$ for all $i \in V$, $j \in F$
- $\sum_{j \in V} x_{ij} \geq 1$ for all $i \in V$
- $x_{ij}, y_j \geq 0$, for all $i \in V$, $j \in F$

Suppose $x^*, y^* \longrightarrow$ optimum solution for the above LP.

# Algorithm

1. Solve the LP. Let $(x^*, y^*)$ be the optimum solution.

# Algorithm

1. Solve the LP. Let $(x^*, y^*)$ be the optimum solution.
2. For each client $i \in V$ do
   - $r_i = 2 \sum_j x^*_{ij} c_{ij}$
   - $B_i = \{j \in F | c_{ij} \leq r_i\}$

# Algorithm

1. Solve the LP. Let $(x^*, y^*)$ be the optimum solution.
2. For each client $i \in V$ do
   - $r_i = 2 \sum_j x^*_{ij} c_{ij}$
   - $B_i = \{j \in F | c_{ij} \leq r_i\}$
3. Sort $r_i$ in non-decreasing order. Let $r_1 \leq r_2 \leq \ldots \leq r_n$, and $V' = \{r_1, r_2, \ldots, r_n\}$ such that the index set of $V'$ represents the clients.

# Algorithm

1. Solve the LP. Let $(x^*, y^*)$ be the optimum solution.
2. For each client $i \in V$ do
   - $r_i = 2\sum_j x^*_{ij} c_{ij}$
   - $B_i = \{j \in F | c_{ij} \leq r_i\}$
3. Sort $r_i$ in non-decreasing order. Let $r_1 \leq r_2 \leq \ldots \leq r_n$, and $V' = \{r_1, r_2, \ldots, r_n\}$ such that the index set of $V'$ represents the clients.
4. Let $i = 1$, $F' = \emptyset$
5. For each client $i \in V'$ do
   - Let $j \in B_i$ with $x^*_{ij} > 0$ and $f_j$ minimum. We set $F' = F' \cup \{j\}$, and $\sigma(i) = j$.
   - Define $N_i = \{r_k \in V' | B_i \cap B_k \neq \emptyset\}$
   - For each client $k$ such that $r_k \in N_i$ do $\sigma(k) = j$
   - Set $V' = V' \setminus N_i$
   - If $V' = \emptyset$, Break

# Algorithm

1. Solve the LP. Let $(x^*, y^*)$ be the optimum solution.
2. For each client $i \in V$ do
   - $r_i = 2\sum_j x^*_{ij} c_{ij}$
   - $B_i = \{j \in F | c_{ij} \leq r_i\}$
3. Sort $r_i$ in non-decreasing order. Let $r_1 \leq r_2 \leq \ldots \leq r_n$, and $V' = \{r_1, r_2, \ldots, r_n\}$ such that the index set of $V'$ represents the clients.
4. Let $i = 1$, $F' = \emptyset$
5. For each client $i \in V'$ do
   - Let $j \in B_i$ with $x^*_{ij} > 0$ and $f_j$ minimum. We set $F' = F' \cup \{j\}$, and $\sigma(i) = j$.
   - Define $N_i = \{r_k \in V' | B_i \cap B_k \neq \emptyset\}$
   - For each client $k$ such that $r_k \in N_i$ do $\sigma(k) = j$
   - Set $V' = V' \setminus N_i$
   - If $V' = \emptyset$, Break
6. Output $F'$ and $\sigma$

# Algorithm

1. Solve the LP. Let $(x^*, y^*)$ be the optimum solution.
2. For each client $i \in V$ do
   - $r_i = 2\sum_j x^*_{ij} c_{ij}$
   - $B_i = \{j \in F | c_{ij} \leq r_i\}$
3. Sort $r_i$ in non-decreasing order. Let $r_1 \leq r_2 \leq \ldots \leq r_n$, and $V' = \{r_1, r_2, \ldots, r_n\}$ such that the index set of $V'$ represents the clients.
4. Let $i = 1$, $F' = \emptyset$
5. For each client $i \in V'$ do
   - Let $j \in B_i$ with $x^*_{ij} > 0$ and $f_j$ minimum. We set $F' = F' \cup \{j\}$, and $\sigma(i) = j$.
   - Define $N_i = \{r_k \in V' | B_i \cap B_k \neq \emptyset\}$
   - For each client $k$ such that $r_k \in N_i$ do $\sigma(k) = j$
   - Set $V' = V' \setminus N_i$
   - If $V' = \emptyset$, Break
6. Output $F'$ and $\sigma$

**Time Complexity Result**

The running time of this algorithm is polynomial in the size of the input.

# Approximation Factor

**Claim: For all $i \in V$, $\sum_{j \in B_i} y_j^* \geq \frac{1}{2}$**

$LP_i = \sum_j x_{ij}^* c_{ij} \longrightarrow$ the service cost of client $i \in V$ in the LP solution.

Assume that the claim is not true, i.e., $\sum_{j \in B_i} y_j^* < \frac{1}{2}$.

We have $LP_i \geq \sum_{j \notin B_i} x_{ij}^* c_{ij}$
$\geq \sum_{j \notin B_i} x_{ij}^* r_i$ (since $c_{ij} > r_i$ for all $j \notin B_i$
$= r_i \sum_{j \notin B_i} x_{ij}^*$
$> \frac{1}{2} r_i$ (since $\sum_{j \in B_i} x_{ij}^* \leq \sum_{j \in B_i} y_j^* < \frac{1}{2}$.)

Thus, $LP_i > \frac{1}{2} r_i = \frac{1}{2} \times 2 \sum_j x_{ij}^* c_{ij} = LP_i \longrightarrow$ Contradiction.

# Approximation factor

**Result:**

If $LP_f$ = Facility cost of the LP solution $= \sum_{j \in F} f_j y_j^*$, then

$$\sum_{j \in F'} f_j \leq 2 \times LP_f$$

- For each $j \in F'$, let $i_j$ is the client who was considered when $j$ was added in $F'$.
- Consider the ball $B_{i_j}$
- Then $\sum_{\ell \in B_{i_j}} y_\ell^* f_\ell \geq \sum_{\ell \in B_{i_j}} y_\ell^* f_j = f_j \times \sum_{\ell \in B_{i_j}} y_\ell^* \geq \frac{1}{2} f_j$
- Reason $\longrightarrow \sum_{\ell \in B_{i_j}} y_\ell^* \geq \frac{1}{2}$.

# Approximation factor

Adding over all members of $F'$,

$$\frac{1}{2}\sum_{j\in F'} f_j \leq \sum_{j\in F'}\sum_{\ell\in B_{i_j}} y^*_\ell f_\ell \leq \sum_{j\in F} f_j y^*_j$$

(since all facilities are selected from non-overlapping balls)

Thus, $\sum_{j\in F'} f_j \leq 2 \times LP_F$.

# Approximation factor

**Result**

If $LP_i = \sum_j x^*_{ij} c_{ij}$ is the service cost of client $i$ in the LP solution, then $c_{i\sigma(i)} \leq 6 \times LP_i$

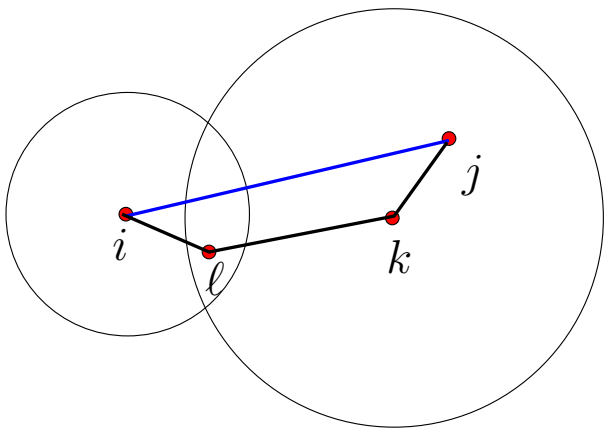
Consider two possible cases for client $i$

Case 1: Client $i$ is such that some $j \in B_i$ was selected. Then,

$$c_{i\sigma(i)} \leq r_i = 2LP_i$$

Case 2: Client $i$ is served by a facility $j \in B_k$ chosen for client $k$ since $B_i \cap B_k \neq \emptyset$. Thus, $\sigma(i) = \sigma(k) = j$

- Surely $r_i > r_k$.
- Let $\ell \in B_i \cap B_k$.
- Thus, $c_{ij} < c_{i\ell} + c_{\ell k} + c_{kj}$ $\leq r_i + 2r_k \leq 3r_i$ [By triangle inequality]




Thus, $c_{ij} \leq 6LP_i \implies c_{i\sigma(i)} \leq 6LP_i$

# Approximation factor

Thus, $\sum_{i \in V} c_i \sigma(i) \leq 6 \times LP_S$,
($LP_S$ = Service cost of LP Solution)

**Theorem**

Total cost $= \sum_{j \in F'} f_j + \sum_{i \in V} c_{i\sigma(i)} \leq 6 \times (LP_f + LP_S)$