Finding Facilities Fast

Saurav Pandit   Sriram V. Pemmaraju

Department of Computer Science
The University of Iowa
[spandit, sriram]@cs.uiowa.edu
www.cs.uiowa.edu/∼[spandit, sriram]/

ICDCN 2009

Results Overview

- A *constant*-factor approximation algorithm for facility location on *unit disk graphs (UDGs)*.
- A *distributed* implementation of this algorithm in *constant* rounds, while still maintaining a constant-factor approximation.

Results Overview

- A *constant*-factor approximation algorithm for facility location on *unit disk graphs (UDGs)*.
- A *distributed* implementation of this algorithm in *constant* rounds, while still maintaining a constant-factor approximation.

Facility Location: Standard Form

- The input is a complete bipartite graph $G = (F, C, E)$, where F is the set of facilities and C is the set of cities.
- Opening costs $f : F \rightarrow \mathbb{R}^+$, and connection costs $c : E \rightarrow \mathbb{R}^+$.
- Find a set of facilities $I \subseteq F$ to open and a function $\phi : C \rightarrow I$ that assigns every city to an open facility so as to minimize $\sum_{i \in I} f(i) + \sum_{j \in C} c(j, \phi(j))$.

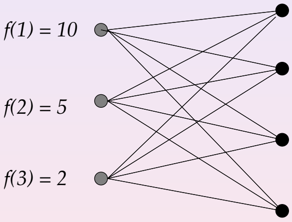
Facility Location: Standard Form

- The input is a complete bipartite graph $G = (F, C, E)$, where F is the set of facilities and C is the set of cities.
- Opening costs $f : F \rightarrow \mathbb{R}^+$, and connection costs $c : E \rightarrow \mathbb{R}^+$.
- Find a set of facilities $I \subseteq F$ to open and a function $\phi : C \rightarrow I$ that assigns every city to an open facility so as to minimize $\sum_{i \in I} f(i) + \sum_{j \in C} c(j, \phi(j))$.

Facility Location: Standard Form

- The input is a complete bipartite graph $G = (F, C, E)$, where F is the set of facilities and C is the set of cities.
- Opening costs $f : F \rightarrow \mathbb{R}^+$, and connection costs $c : E \rightarrow \mathbb{R}^+$.
- Find a set of facilities $I \subseteq F$ to open and a function $\phi : C \rightarrow I$ that assigns every city to an open facility so as to minimize $\sum_{i \in I} f(i) + \sum_{j \in C} c(j, \phi(j))$.

Facility Location
An Illustration

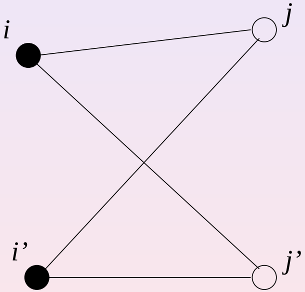


	1	2	3
a	4	1	6
b	7	1	9
c	5	1	7
d	4	6	1

c(i,x) values

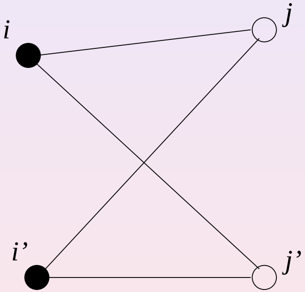
OPT: open facilities 2 and 3 with cities *a*, *b* and *c* connected to facility 2 and city *d* to facility 3.

Metric Facility Location



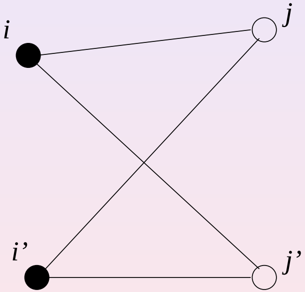
$$c(i,j) \leq c(i,j') + c(i',j') + c(i',j).$$

Metric Facility Location



$$c(i,j) \leq c(i,j') + c(i',j') + c(i',j).$$

Metric Facility Location



$$c(i,j) \leq c(i,j') + c(i',j') + c(i',j).$$

Facility Location on Wireless Networks

- A facility can be opened at any node.
- Each node is a city.
- Connection cost $c(i, j)$ between neighbors i and j depends on Euclidean distance $|ij|$.
- Cost of connecting non-neighbors is ∞.

Facility Location on Wireless Networks

- A facility can be opened at any node.
- Each node is a city.
- Connection cost $c(i,j)$ between neighbors i and j depends on Euclidean distance $|ij|$.
- Cost of connecting non-neighbors is ∞.

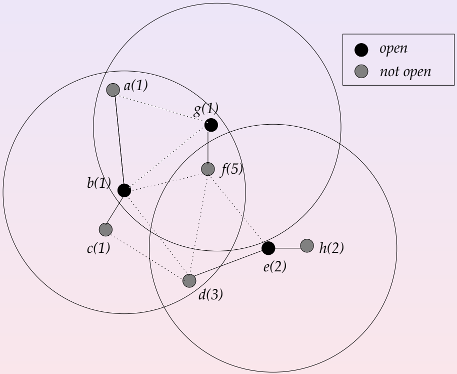
Facility Location on Wireless Networks

- A facility can be opened at any node.
- Each node is a city.
- Connection cost $c(i,j)$ between neighbors i and j depends on Euclidean distance $|ij|$.
- Cost of connecting non-neighbors is ∞.

Facility Location on Wireless Networks

- A facility can be opened at any node.
- Each node is a city.
- Connection cost $c(i,j)$ between neighbors i and j depends on Euclidean distance $|ij|$.
- Cost of connecting non-neighbors is ∞.

Facility Location on Wireless Networks
An Illustration



The cost of this solution is 4 units (for opening facilities) plus $|fg| + |ab| + |cb| + |de| + |he|$.

Past Work

- Non-metric facility location: $O(\log n)$-approximation [Hochbaum, 1982].
- Cannot be approximated to better than $\Omega(\log n)$ (reduction from SET COVER).
- Metric facility location: $O(1)$-approximation [Shmoys-Tardos-Aardal, Jain-Vazirani, Charikar-Guha, etc].
- Cannot be approximated to better than 1.463 [Guha-Khuller, 1998].

Past Work

- Non-metric facility location: $O(\log n)$-approximation [Hochbaum, 1982].
- Cannot be approximated to better than $\Omega(\log n)$ (reduction from `SET COVER`).
- Metric facility location: $O(1)$-approximation [Shmoys-Tardos-Aardal, Jain-Vazirani, Charikar-Guha, etc].
- Cannot be approximated to better than 1.463 [Guha-Khuller, 1998].

Past Work

- Non-metric facility location: $O(\log n)$-approximation [Hochbaum, 1982].
- Cannot be approximated to better than $\Omega(\log n)$ (reduction from `SET COVER`).
- Metric facility location: $O(1)$-approximation [Shmoys-Tardos-Aardal, Jain-Vazirani, Charikar-Guha, etc].
- Cannot be approximated to better than 1.463 [Guha-Khuller, 1998].

Past Work

- Non-metric facility location: $O(\log n)$-approximation [Hochbaum, 1982].
- Cannot be approximated to better than $\Omega(\log n)$ (reduction from SET COVER).
- Metric facility location: $O(1)$-approximation [Shmoys-Tardos-Aardal, Jain-Vazirani, Charikar-Guha, etc].
- Cannot be approximated to better than 1.463 [Guha-Khuller, 1998].

UDG Facility Location
Connection Costs

- We assume that there is a function $g : [0,1] \rightarrow \mathbb{R}^+$ such that each edge $\{i,j\} \in E$ has a connection cost $c(i,j) = g(|ij|)$.
- We assume that $g(\cdot)$ is a monotonically increasing function with *bounded growth*, i.e., for some constant $B \geq 1$, $g(x) \leq B \cdot g(x/3)$ for all $x \in [0,1]$.
- **Example 1:** If connection costs equal Euclidean distance, then $g(x) = x$ and therefore $B = 3$.
- **Example 2:** If connection costs represent *power consumption*, e.g., $g(x) = \beta \cdot x^\gamma$ for constants β and γ, then $B = 3^\gamma$.

UDG Facility Location
Connection Costs

- We assume that there is a function $g : [0,1] \rightarrow \mathbb{R}^+$ such that each edge $\{i,j\} \in E$ has a connection cost $c(i,j) = g(|ij|)$.
- We assume that $g(\cdot)$ is a monotonically increasing function with *bounded growth*, i.e., for some constant $B \geq 1$, $g(x) \leq B \cdot g(x/3)$ for all $x \in [0,1]$.
- **Example 1:** If connection costs equal Euclidean distance, then $g(x) = x$ and therefore $B = 3$.
- **Example 2:** If connection costs represent *power consumption*, e.g., $g(x) = \beta \cdot x^\gamma$ for constants β and γ, then $B = 3^\gamma$.

UDG Facility Location
Connection Costs

- We assume that there is a function $g : [0,1] \rightarrow \mathbb{R}^+$ such that each edge $\{i,j\} \in E$ has a connection cost $c(i,j) = g(|ij|)$.
- We assume that $g(\cdot)$ is a monotonically increasing function with *bounded growth*, i.e., for some constant $B \geq 1$, $g(x) \leq B \cdot g(x/3)$ for all $x \in [0,1]$.
- **Example 1:** If connection costs equal Euclidean distance, then $g(x) = x$ and therefore $B = 3$.
- **Example 2:** If connection costs represent *power consumption*, e.g., $g(x) = \beta \cdot x^\gamma$ for constants β and γ, then $B = 3^\gamma$.

UDG Facility Location
Connection Costs

- We assume that there is a function $g : [0,1] \to \mathbb{R}^+$ such that each edge $\{i,j\} \in E$ has a connection cost $c(i,j) = g(|ij|)$.
- We assume that $g(\cdot)$ is a monotonically increasing function with *bounded growth*, i.e., for some constant $B \geq 1$, $g(x) \leq B \cdot g(x/3)$ for all $x \in [0,1]$.
- **Example 1:** If connection costs equal Euclidean distance, then $g(x) = x$ and therefore $B = 3$.
- **Example 2:** If connection costs represent *power consumption*, e.g., $g(x) = \beta \cdot x^\gamma$ for constants β and γ, then $B = 3^\gamma$.

Main Results Revisited

- A sequential $(6 + B + \varepsilon)$-approximation algorithm for `UDG-FacLoc`.
- A distributed $16 \cdot (2 + B)$-approximation algorithm for `UDG-FacLoc` running in constant-rounds in the $\mathcal{LOCAL}$ model.

Main Results Revisited

- A sequential $(6 + B + \varepsilon)$-approximation algorithm for `UDG-FacLoc`.
- A distributed $16 \cdot (2 + B)$-approximation algorithm for `UDG-FacLoc` running in constant-rounds in the $\mathcal{LOCAL}$ model.

Algorithm: High Level Description
Step 1

- Convert the given instance of `UDG-FacLoc` to a standard non-metric instance of facility location.
- Run the primal-dual algorithm of Jain-Vazirani and obtain a solution (I, ϕ)

Note

(I, ϕ) may have cost ∞ because some connections between cities and facilities may cost ∞.

Algorithm: High Level Description

Step 1

- Convert the given instance of `UDG-FacLoc` to a standard non-metric instance of facility location.
- Run the primal-dual algorithm of Jain-Vazirani and obtain a solution (I, ϕ)

Note

(I, ϕ) may have cost ∞ because some connections between cities and facilities may cost ∞.

Algorithm: High Level Description

Step 1

- Convert the given instance of `UDG-FacLoc` to a standard non-metric instance of facility location.
- Run the primal-dual algorithm of Jain-Vazirani and obtain a solution (I, ϕ)

Note

(I, ϕ) may have cost ∞ because some connections between cities and facilities may cost ∞.

Algorithm: High Level Description
Step 2

- Assign to each node i the weight $f(i)$.
- Compute a light *weighted dominating set* of G and obtain a set D.

Note

Use the algorithm of Huang et al. [J. Comb. Opt., 2008] to obtain a $(6+\varepsilon)$-approximation of a lightest dominating set.

Algorithm: High Level Description

Step 2

- Assign to each node i the weight $f(i)$.
- Compute a light *weighted dominating set* of G and obtain a set D.

Note

Use the algorithm of Huang et al. [J. Comb. Opt., 2008] to obtain a $(6+\varepsilon)$-approximation of a lightest dominating set.

Algorithm: High Level Description
Step 2

- Assign to each node i the weight $f(i)$.
- Compute a light *weighted dominating set* of G and obtain a set D.

Note

Use the algorithm of Huang et al. [J. Comb. Opt., 2008] to obtain a $(6 + \varepsilon)$-approximation of a lightest dominating set.

Algorithm: High Level Description
Step 3

- Open all nodes in D as facilities.
- Each node i, that uses a connection of cost ∞ is *reconnected* to an arbitrary neighbor in D.

Algorithm: High Level Description
Step 3

- Open all nodes in D as facilities.
- Each node i, that uses a connection of cost ∞ is *reconnected* to an arbitrary neighbor in D.

Analysis: High Level Ideas

Step 1

- Recall (I, ϕ) is solution produced by the primal-dual step (i.e., Step 1) of the algorithm.
- Let $cost'(I, \phi)$ denote the cost of this solution with all ∞ cost connections excluded.

Jain-Vazirani

$cost'(I, \phi) \leq OPT$

Analysis: High Level Ideas

Step 1

- Recall (I, ϕ) is solution produced by the primal-dual step (i.e., Step 1) of the algorithm.
- Let $cost'(I, \phi)$ denote the cost of this solution with all ∞ cost connections excluded.

Jain-Vazirani

$cost'(I, \phi) \leq OPT$

Analysis: High Level Ideas
Step 1

- Recall (I, ϕ) is solution produced by the primal-dual step (i.e., Step 1) of the algorithm.
- Let $cost'(I, \phi)$ denote the cost of this solution with all ∞ cost connections excluded.

Jain-Vazirani

$cost'(I, \phi) \leq OPT$

Analysis: High Level Ideas
Step 2

- Let D^* be a dominating set of minimum weight. Then $\text{wt}(D^*) \leq OPT$.
- Recall that in Step 2, we construct a dominating set D such that $\text{wt}(D) \leq (6 + \varepsilon) \cdot \text{wt}(D^*)$.

Lemma: Dominating set is cheap

$\text{wt}(D) \leq (6 + \varepsilon) \cdot OPT$

Analysis: High Level Ideas

Step 2

- Let D^* be a dominating set of minimum weight. Then $\text{wt}(D^*) \leq OPT$.
- Recall that in Step 2, we construct a dominating set D such that $\text{wt}(D) \leq (6 + \varepsilon) \cdot \text{wt}(D^*)$.

Lemma: Dominating set is cheap

$\text{wt}(D) \leq (6 + \varepsilon) \cdot OPT$

Analysis: High Level Ideas
Step 2

- Let D^* be a dominating set of minimum weight. Then $\text{wt}(D^*) \leq OPT$.
- Recall that in Step 2, we construct a dominating set D such that $\text{wt}(D) \leq (6+\varepsilon) \cdot \text{wt}(D^*)$.

Lemma: Dominating set is cheap

$\text{wt}(D) \leq (6+\varepsilon) \cdot OPT$

Analysis: High Level Ideas

Step 3

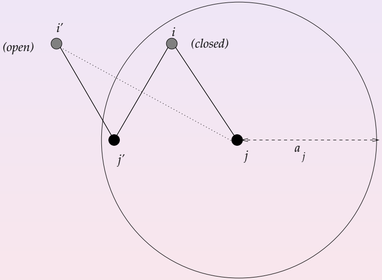
The primal-dual algorithm assigns to each node j a “dual” value α_j such that $\sum_j \alpha_j \leq OPT$.

Lemma: Reconnected nodes have high dual cost

For any node j such that $c(j, \phi(j)) = \infty$, j is *reconnected* to some $i' \in D$ and $c(j, i') \leq B \cdot \alpha_j$.

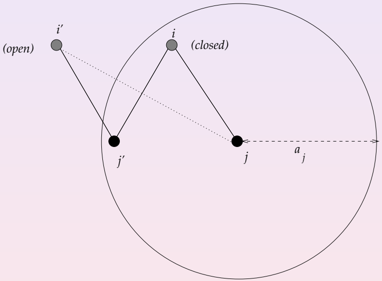
Let *Rcost* denote the total cost of all reconnections.
$Rcost \leq \sum_j \alpha_j \leq OPT$.

Intuition for the Lemma



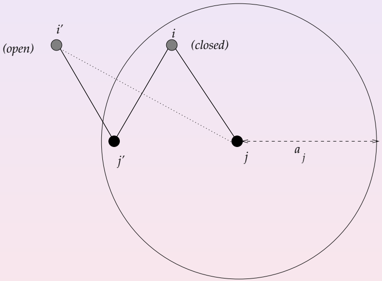
- $\alpha_j \geq \max\{c(i,j), c(i,j'), c(i',j')\}$
- $\max\{c(i,j), c(i,j'), c(i',j')\} \geq 1/3$

Intuition for the Lemma



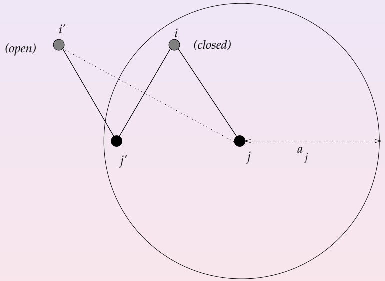
- $\alpha_j \geq \max\{c(i, j), c(i, j'), c(i', j')\}$
- $\max\{c(i, j), c(i, j'), c(i', j')\} \geq 1/3$

Intuition for the Lemma



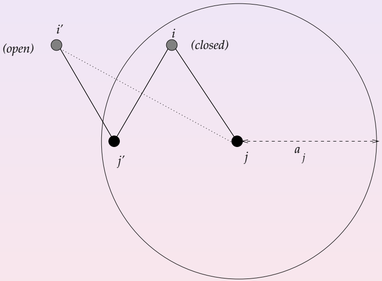
- $\alpha_j \geq \max\{c(i,j), c(i,j'), c(i',j')\}$
- $\max\{c(i,j), c(i,j'), c(i',j')\} \geq 1/3$

Intuition for the Lemma



- $\alpha_j \geq \max\{c(i,j), c(i,j'), c(i',j')\}$
- $\max\{c(i,j), c(i,j'), c(i',j')\} \geq 1/3$

Intuition for the Lemma



- $\alpha_j \geq \max\{c(i,j), c(i,j'), c(i',j')\}$
- $\max\{c(i,j), c(i,j'), c(i',j')\} \geq 1/3$

Analysis: Putting it Together

$$\begin{aligned}\text{cost of solution} &= cost'(I, \phi) + \mathrm{wt}(D) + Rcost \\ &\leq OPT + (6+\varepsilon)\cdot OPT + B \cdot OPT \\ &= (7 + B + \varepsilon)\cdot OPT\end{aligned}$$

Slightly more sophisticated analysis yields the $(6 + B + \varepsilon)\cdot OPT$ upper bound.

Local Subproblems

- Partition the plane into $\frac{1}{\sqrt{2}} \times \frac{1}{\sqrt{2}}$ size square cells.
- Let $V_s \subseteq V$ be the set of vertices which lie in square s.
- Let $N(V_s)$ denote the set of all vertices in $V \setminus V_s$ that are adjacent to some vertex in V_s.
- $\texttt{UDG-FacLoc}_s$ is the subproblem in which we are allowed to open facilities from $V_s \cup N(V_s)$ with the aim of connecting all the nodes in V_s to these facilities.

Local Subproblems

- Partition the plane into $\frac{1}{\sqrt{2}} \times \frac{1}{\sqrt{2}}$ size square cells.
- Let $V_s \subseteq V$ be the set of vertices which lie in square s.
- Let $N(V_s)$ denote the set of all vertices in $V \setminus V_s$ that are adjacent to some vertex in V_s.
- $\texttt{UDG-FacLoc}_s$ is the subproblem in which we are allowed to open facilities from $V_s \cup N(V_s)$ with the aim of connecting all the nodes in V_s to these facilities.

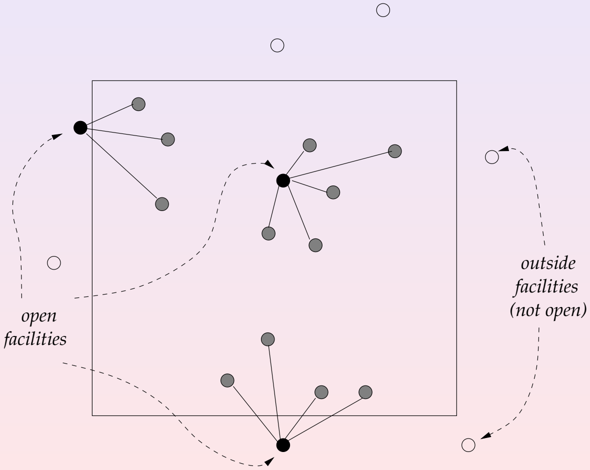
Local Subproblems

- Partition the plane into $\frac{1}{\sqrt{2}} \times \frac{1}{\sqrt{2}}$ size square cells.
- Let $V_s \subseteq V$ be the set of vertices which lie in square s.
- Let $N(V_s)$ denote the set of all vertices in $V \setminus V_s$ that are adjacent to some vertex in V_s.
- $\texttt{UDG-FacLoc}_s$ is the subproblem in which we are allowed to open facilities from $V_s \cup N(V_s)$ with the aim of connecting all the nodes in V_s to these facilities.

Local Subproblems

- Partition the plane into $\frac{1}{\sqrt{2}} \times \frac{1}{\sqrt{2}}$ size square cells.
- Let $V_s \subseteq V$ be the set of vertices which lie in square s.
- Let $N(V_s)$ denote the set of all vertices in $V \setminus V_s$ that are adjacent to some vertex in V_s.
- $\texttt{UDG-FacLoc}_s$ is the subproblem in which we are allowed to open facilities from $V_s \cup N(V_s)$ with the aim of connecting all the nodes in V_s to these facilities.

Local Subproblems
An Illustration



Locality of UDG-FacLoc

Lemma

For each square s, let OPT_s denote the cost of an optimal solution to $\texttt{UDG-FacLoc}_s$ and let $\{F_s, \phi_s\}$ be a solution to $\texttt{UDG-FacLoc}_s$ such that for some c, $cost(\{F_s, \phi_s\}) \leq c \cdot OPT_s$. Then $cost(\cup_s \{F_s, \phi_s\}) \leq 16c \cdot OPT$.

Distributed Algorithm: High Level Description

Step 1. Each node v gathers information about the subgraph induced by its 2-neighborhood.

Step 2. Each node v in square s then identifies V_s and $N(V_s)$.

Step 3. Each node v locally computes the solution of $\texttt{UDG-FacLoc}_s$, thereby determining whether it should be opened as a facility and if not which neighboring facility it should connect to.

Distributed Algorithm: High Level Description

Step 1. Each node v gathers information about the subgraph induced by its 2-neighborhood.

Step 2. Each node v in square s then identifies V_s and $N(V_s)$.

Step 3. Each node v locally computes the solution of $\texttt{UDG-FacLoc}_s$, thereby determining whether it should be opened as a facility and if not which neighboring facility it should connect to.

Distributed Algorithm: High Level Description

Step 1. Each node v gathers information about the subgraph induced by its 2-neighborhood.

Step 2. Each node v in square s then identifies V_s and $N(V_s)$.

Step 3. Each node v locally computes the solution of $\texttt{UDG-FacLoc}_s$, thereby determining whether it should be opened as a facility and if not which neighboring facility it should connect to.

Open Questions

- Can we get a constant-approximation to `UDG-FacLoc` without access to geometric information?
- Can we get a constant-approximation to `UDG-FacLoc` in a model with bounded message sizes (e.g., the $\mathcal{CONGEST}$ model)?

Open Questions

- Can we get a constant-approximation to `UDG-FacLoc` without access to geometric information?
- Can we get a constant-approximation to `UDG-FacLoc` in a model with bounded message sizes (e.g., the $\mathcal{CONGEST}$ model)?

The End

- Thank you for coming.
- Any questions?

The End

- Thank you for coming.
- Any questions?