Large-Scale Distributed Algorithms for Facility Location with Outliers

Shreyas Pai

The University of Iowa

December 17, 2018

Joint work with Tanmay Inamdar and Sriram V. Pemmaraju

(Metric Uncapacitated) Facility Location

(Metric Uncapacitated) Facility Location

Input:

- A set F of facilities with opening costs $f : F \to \mathbb{R}^+$

(Metric Uncapacitated) Facility Location

Input:

- A set F of facilities with opening costs $f : F \to \mathbb{R}^+$
- A set C of clients

(Metric Uncapacitated) Facility Location

Input:

- A set F of facilities with opening costs $f : F \to \mathbb{R}^+$
- A set C of clients
- A metric space d on $F \cup C$.

(Metric Uncapacitated) Facility Location

Input:

- A set F of facilities with opening costs $f : F \to \mathbb{R}^+$
- A set C of clients
- A metric space d on $F \cup C$.

Objective:

$$\min_{S \subseteq F} \quad \sum_{i \in S} f_i + \sum_{j \in C} d(j, S)$$

(Metric Uncapacitated) Facility Location

A well known NP-hard problem.

(Metric Uncapacitated) Facility Location

A well known NP-hard problem.

There are two sequential 3-approximation algorithms by –

(Metric Uncapacitated) Facility Location

A well known NP-hard problem.

There are two sequential 3-approximation algorithms by –

- Jain and Vazirani (Primal-Dual Algorithm)
- Mettu and Plaxton (Greedy Algorithm)

Facility Location with Outliers

Facility Location with Outliers

Some clients may be inherently too costly to serve i.e. outliers

Facility Location with Outliers

Some clients may be inherently too costly to serve i.e. outliers

Outliers significantly change the structure of the output

Facility Location with Outliers

Some clients may be inherently too costly to serve i.e. outliers

Outliers significantly change the structure of the output

How do we identify (and eliminate) outliers?

Facility Location with Outliers

Charikar et al. [1] proposed two variants of Facility Location with Outliers which we consider –

[1] Charikar, Khuller, Mount, Narasimhan: Algorithms for Facility Location Problems with Outliers, SODA 2001

Facility Location with Outliers

Charikar et al. [1] proposed two variants of Facility Location with Outliers which we consider –

- Robust Facility Location
- Facility Location with Penalties

[1] Charikar, Khuller, Mount, Narasimhan: Algorithms for Facility Location Problems with Outliers, SODA 2001

Facility Location with Outliers

Charikar et al. [1] proposed two variants of Facility Location with Outliers which we consider –

- Robust Facility Location
- Facility Location with Penalties

They also gave $O(1)$ approximation algorithms for both these problems.

[1] Charikar, Khuller, Mount, Narasimhan: Algorithms for Facility Location Problems with Outliers, SODA 2001

Robust Facility Location

Robust Facility Location

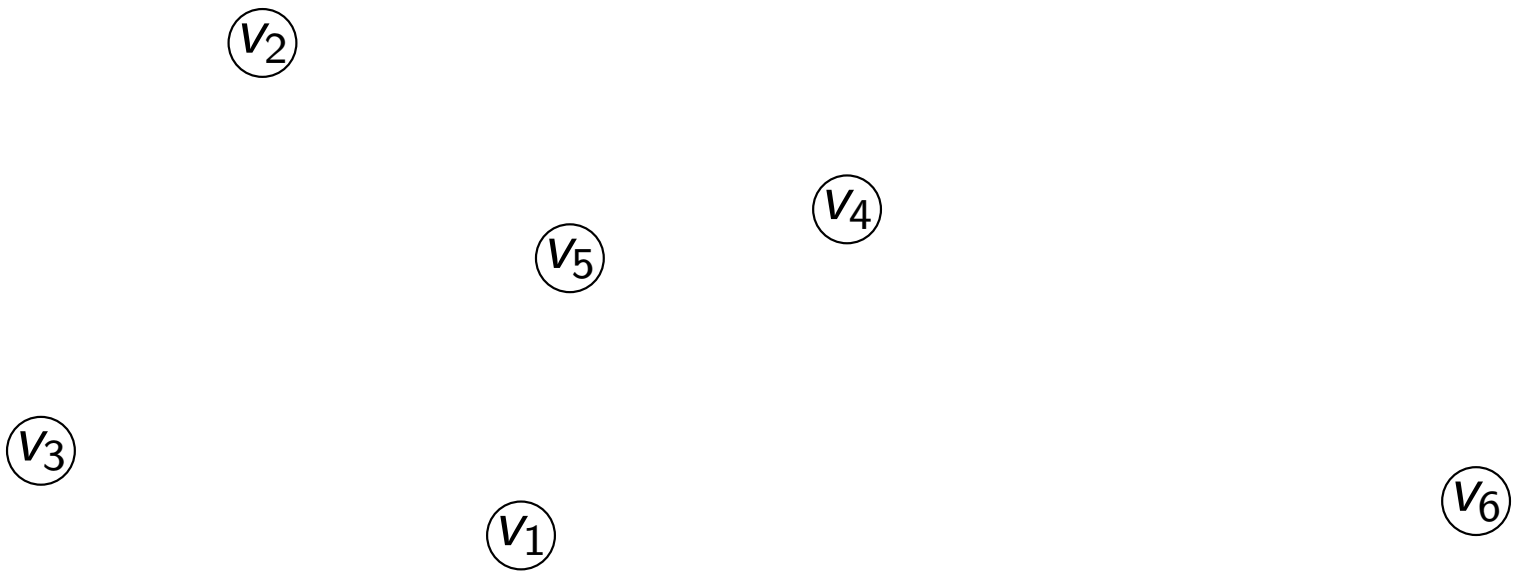
We are given a coverage requirement of $n - \ell$ clients.

Robust Facility Location

We are given a coverage requirement of $n - \ell$ clients.

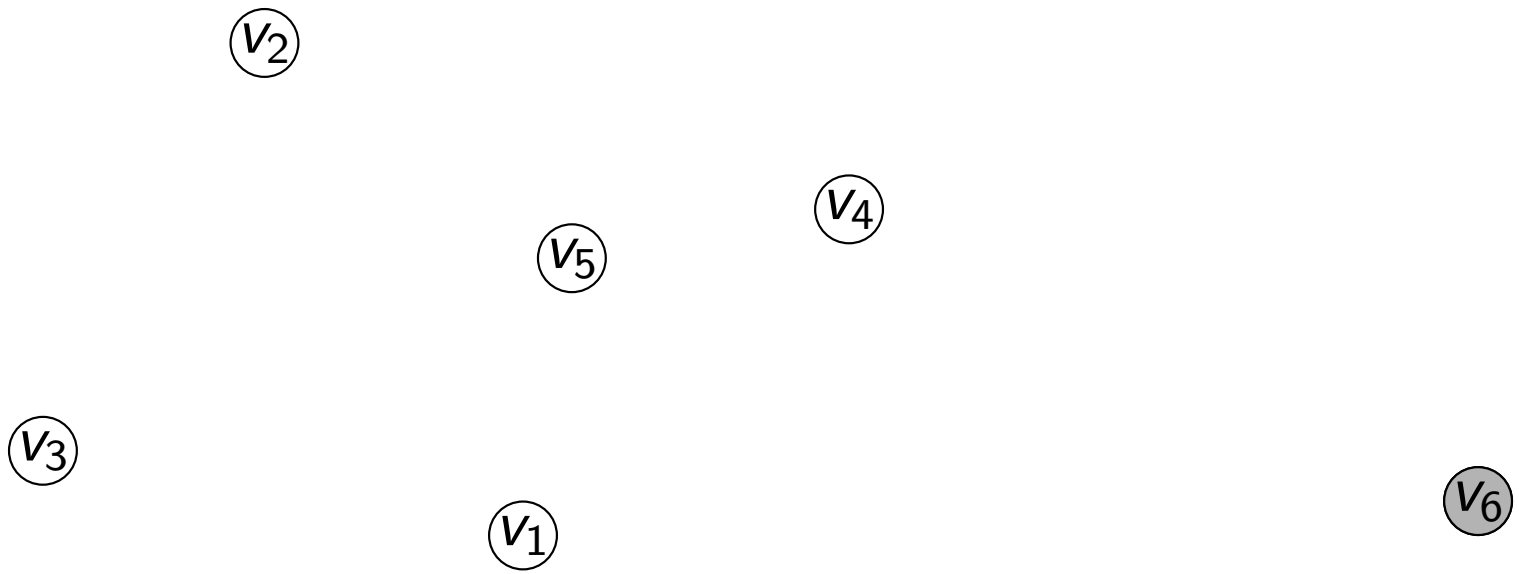
We want to minimize Facility Location objective while not serving at most ℓ clients (outliers).

An Example $\ell = 1$



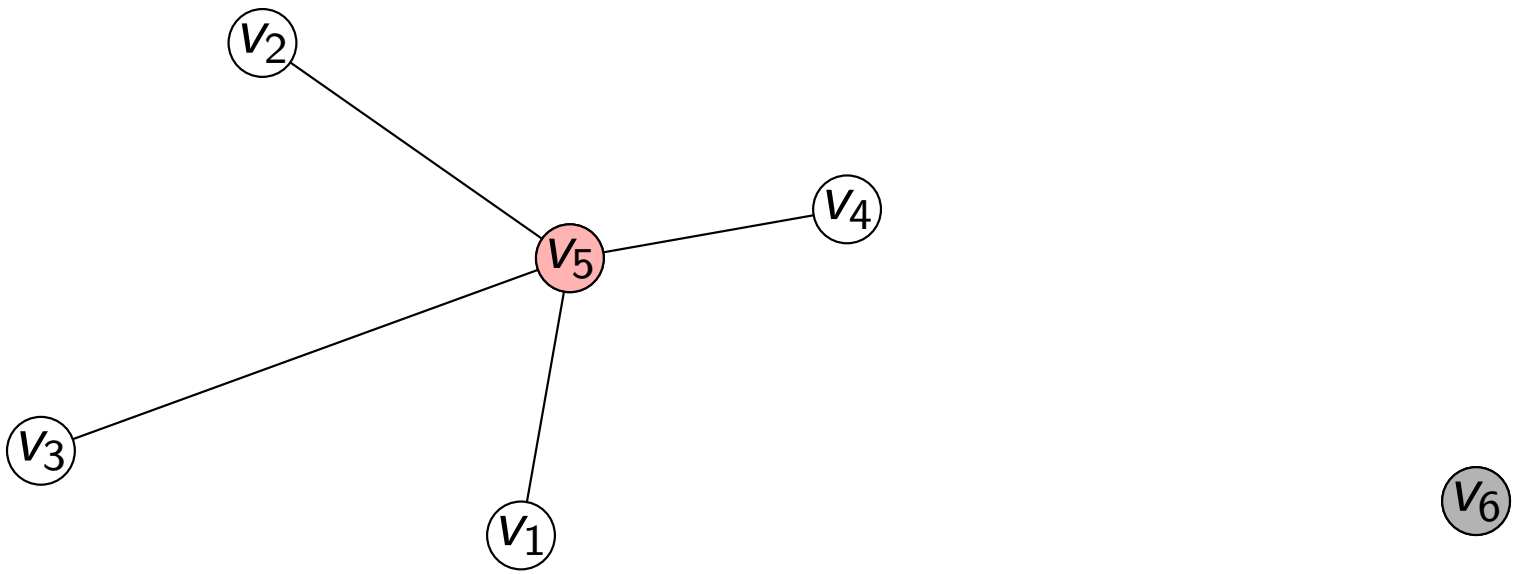
Facilities and Clients are co-located

An Example $\ell = 1$



Facilities and Clients are co-located

An Example $\ell = 1$



Facilities and Clients are co-located

Metric Representations

Metric Representations

In distributed settings, the complexity of the problem can be quite sensitive to the input representation.

Metric Representations

In distributed settings, the complexity of the problem can be quite sensitive to the input representation.

We consider two alternate ways of specifying the input to the problem:

Metric Representations

In distributed settings, the complexity of the problem can be quite sensitive to the input representation.

We consider two alternate ways of specifying the input to the problem:

- **Explicit Metric:** The metric is specified as a $|F| \times |C|$ matrix distributed among the machines

Metric Representations

In distributed settings, the complexity of the problem can be quite sensitive to the input representation.

We consider two alternate ways of specifying the input to the problem:

- **Explicit Metric:** The metric is specified as a $|F| \times |C|$ matrix distributed among the machines
- **Implicit Metric:** The metric is specified as the shortest path metric of a given edge-weighted graph whose vertex set is $F \cup C$

Why use Implicit Metric?

Why use Implicit Metric?

The graph specifying the metric can be quite sparse (e.g., having $O(|F| + |C|)$ edges).

Why use Implicit Metric?

The graph specifying the metric can be quite sparse (e.g., having $O(|F| + |C|)$ edges).

Pro: We get a linear amount of memory savings in such sparse settings.

Why use Implicit Metric?

The graph specifying the metric can be quite sparse (e.g., having $O(|F| + |C|)$ edges).

Pro: We get a linear amount of memory savings in such sparse settings.

Con: Need to perform shortest path computations to get distances between nodes.

Handling Outliers

Handling Outliers

Charikar et al. obtain $O(1)$ approximation algorithms for Facility Location with Outliers by modifying the Jain-Vazirani Algorithm.

Handling Outliers

Charikar et al. obtain $O(1)$ approximation algorithms for Facility Location with Outliers by modifying the Jain-Vazirani Algorithm.

Can implement Jain-Vazirani in distributed settings in poly-logarithmic rounds.

Handling Outliers

Charikar et al. obtain $O(1)$ approximation algorithms for Facility Location with Outliers by modifying the Jain-Vazirani Algorithm.

Can implement Jain-Vazirani in distributed settings in poly-logarithmic rounds.

The Mettu-Plaxton algorithm allows much more efficient distributed implementations.

Main Technical Contribution

Main Technical Contribution

We show that the Mettu-Plaxton algorithm can be modified in a similar manner to obtain $O(1)$ approximation algorithms for

- Robust Facility Location
- Facility Location with Penalties

Our Results

Our Results

We give constant approximation algorithms in three models of distributed computing:

Our Results

We give constant approximation algorithms in three models of distributed computing:

- Congested Clique Model
- Massively Parallel Computation (MPC) Model with $\tilde{O}(n)$ memory
- k-machine Model

Our Resuts

Round Complexity of our algorithms in different models:

	Implicit	Explicit
Congested Clique	$O(\text{poly} \log n)$	$O(\log \log \log n)$
MPC, $\tilde{O}(n)$ memory	$O(\text{poly} \log n)$	$O(\log \log \log n)$
k-machine	$\tilde{O}(n/k)$	$\tilde{O}(n/k)$

The Mettu-Plaxton Algorithm

The Mettu-Plaxton Algorithm

- **Radius Computation Phase.**
  - For each facility $v \in F$, compute radius r_v.
- **Greedy Phase.**
  - Consider facilities $v \in F$ in non-decreasing order of radii.
  - Starting with $S = \emptyset$, add v to S if $d(v, S) > 2r_v$.

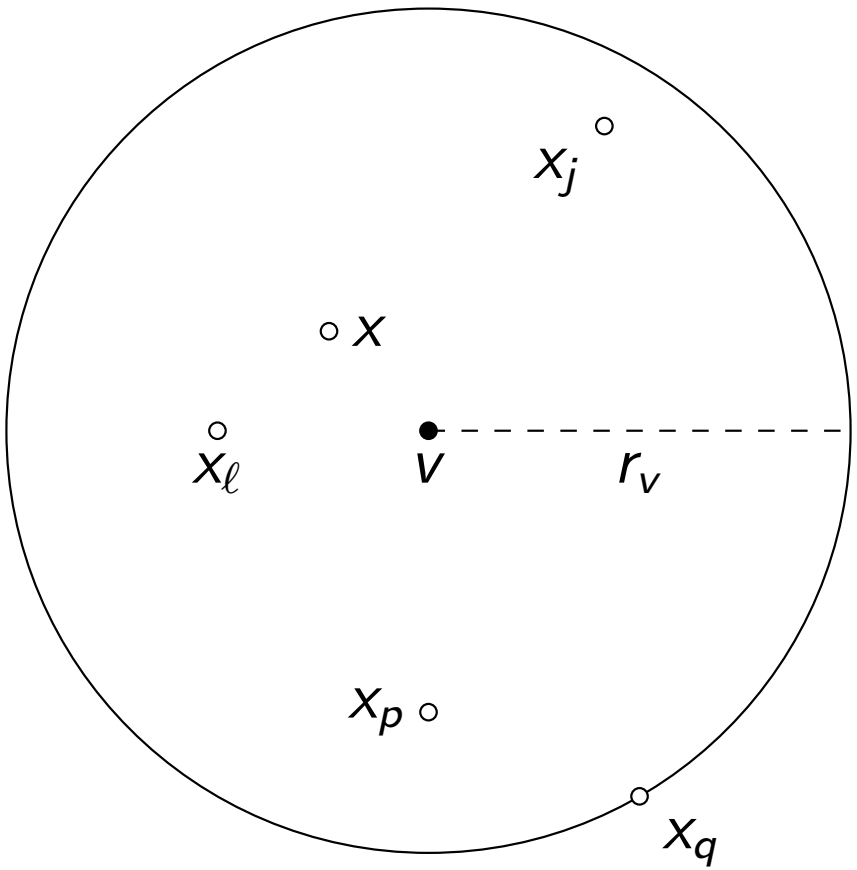
Radius Definition

Radius Definition

$$r_v : f_v = \sum_{x \in B(v, r_v)} r_v - d(x, v)$$

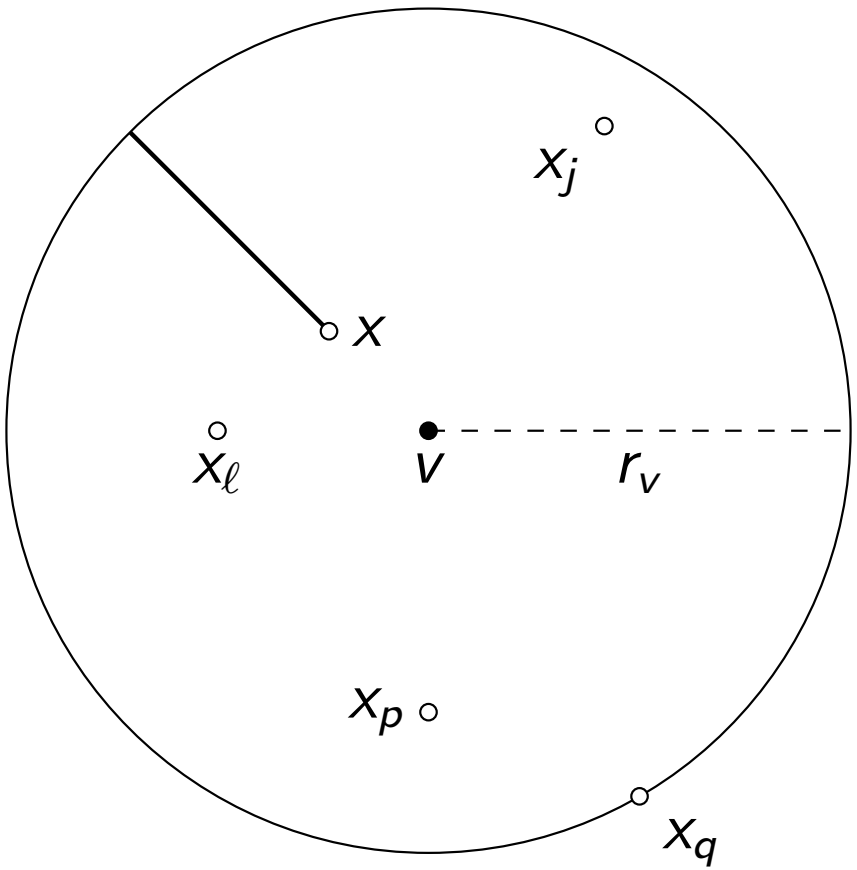
Radius Definition

$$r_v : f_v = \sum_{x \in B(v, r_v)} r_v - d(x, v)$$



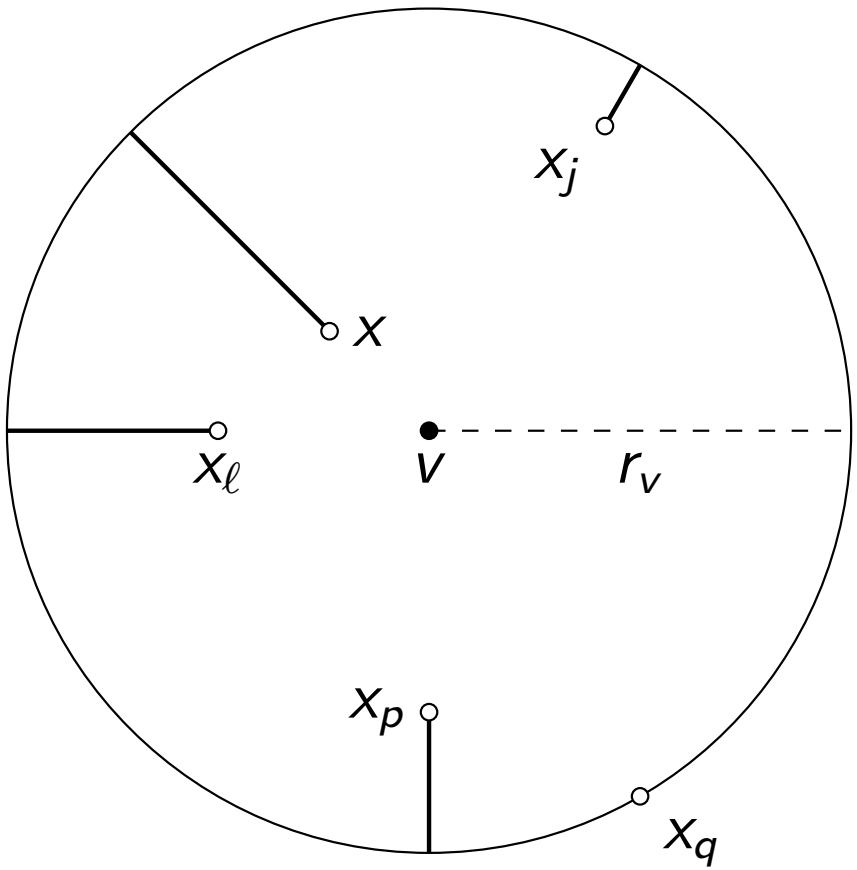
Radius Definition

$$r_v : f_v = \sum_{x \in B(v, r_v)} r_v - d(x, v)$$



Radius Definition

$$r_v : f_v = \sum_{x \in B(v, r_v)} r_v - d(x, v)$$



Radius Computation

Radius Computation

Explicit metric: radius computation can be done locally.

Radius Computation

Explicit metric: radius computation can be done locally.

Implicit metric: radius computation can be done using polylog n calls to a $(1+\epsilon)$-approximate SSSP subroutine [2].

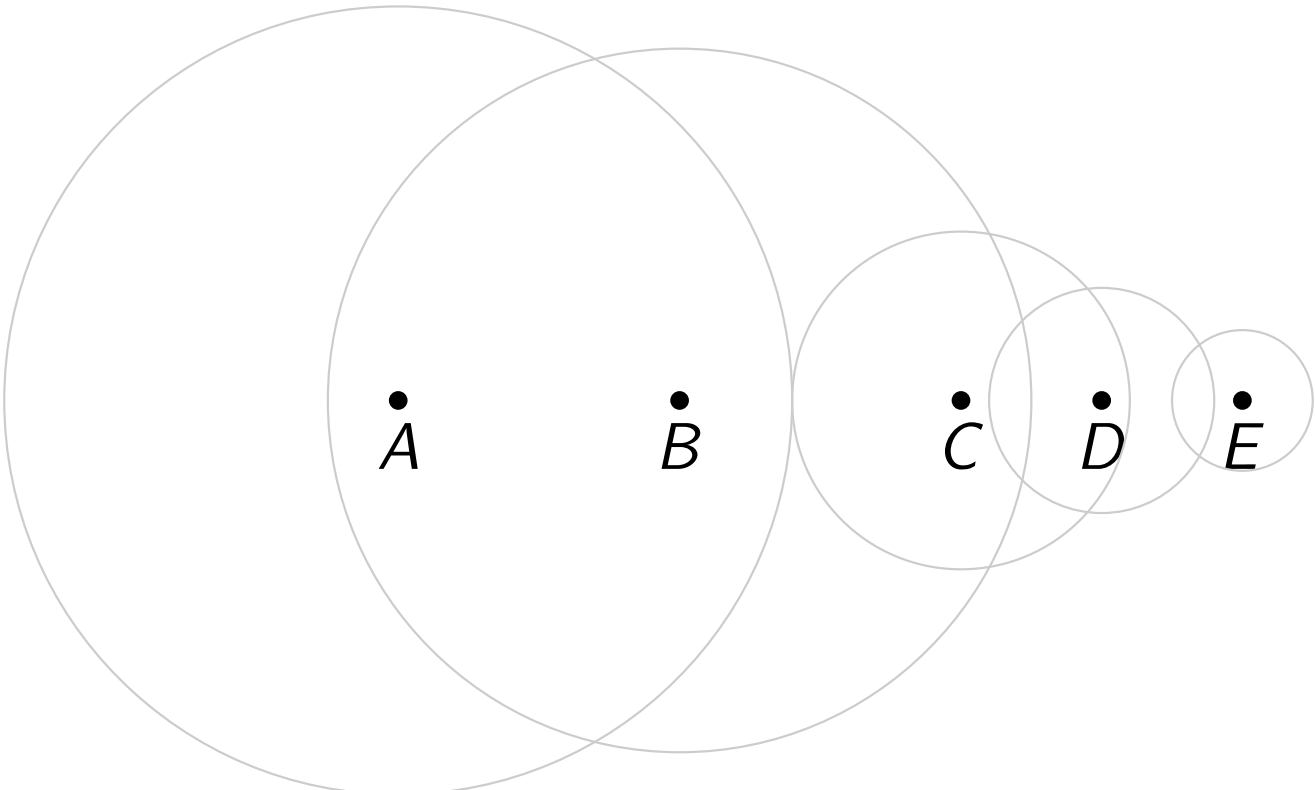
[2] Bandyapadhyay, Inamdar, Pai, Pemmaraju: Near-Optimal Clustering in the k-Machine Model, ICDCN 2018

Greedy Phase

- Consider facilities $v \in F$ in non-decreasing order of radii.
- Starting with $S = \emptyset$, add v to S if $d(v, S) > 2r_v$.

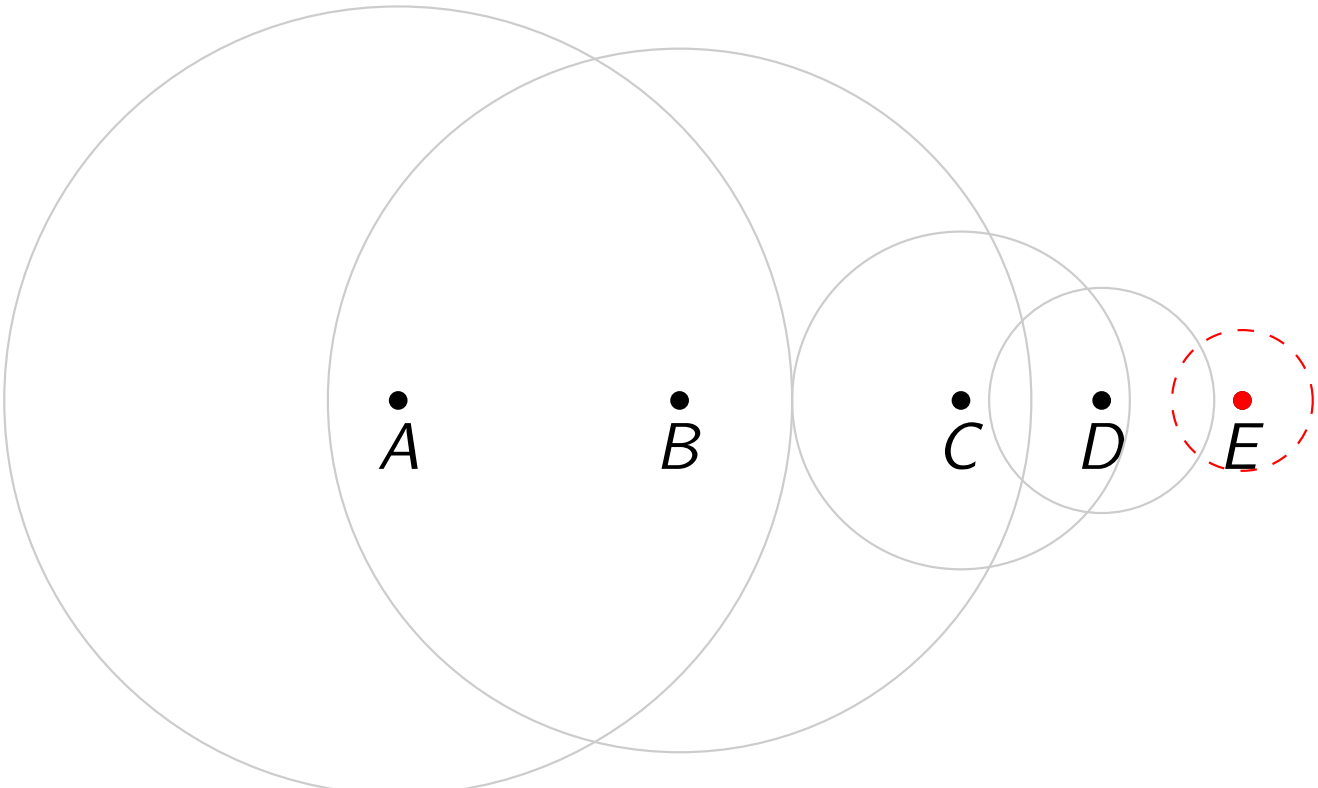
Greedy Phase

- Consider facilities $v \in F$ in non-decreasing order of radii.
- Starting with $S = \emptyset$, add v to S if $d(v, S) > 2r_v$.



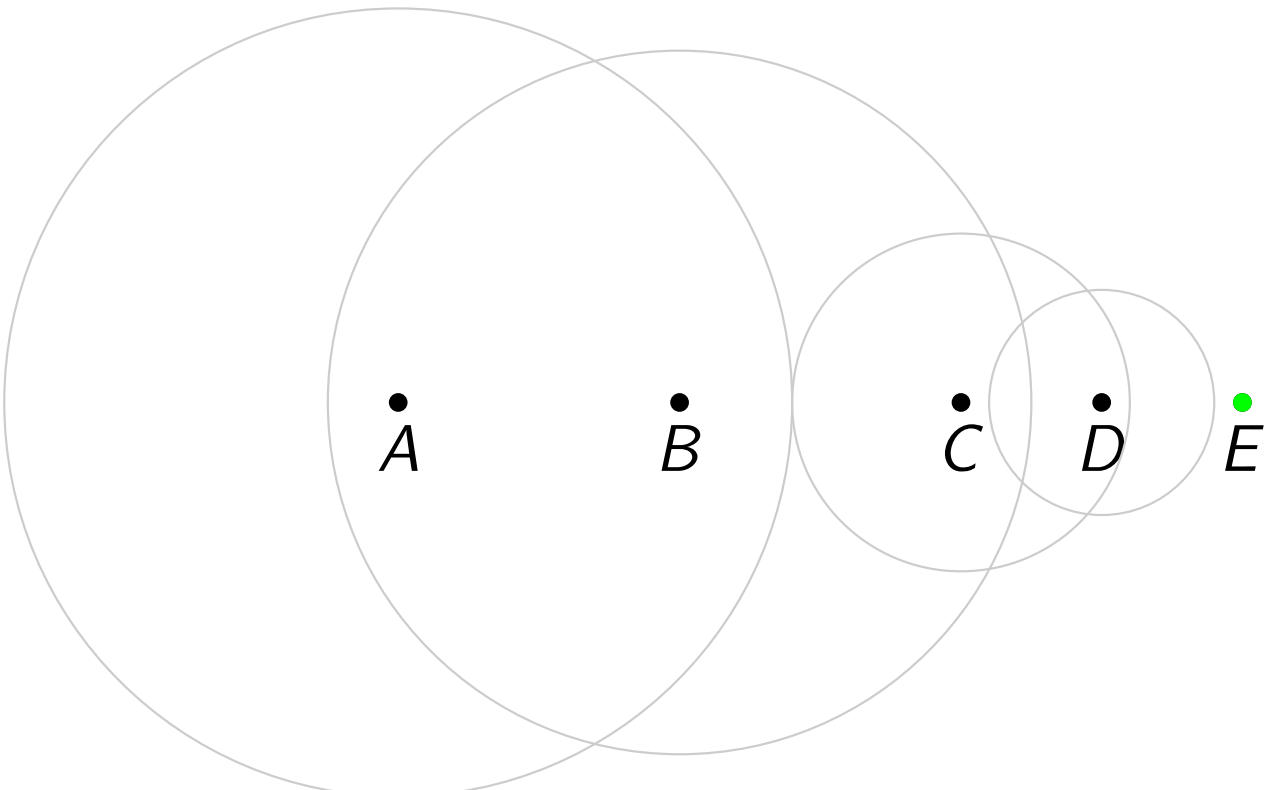
Greedy Phase

- Consider facilities $v \in F$ in non-decreasing order of radii.
- Starting with $S = \emptyset$, add v to S if $d(v, S) > 2r_v$.



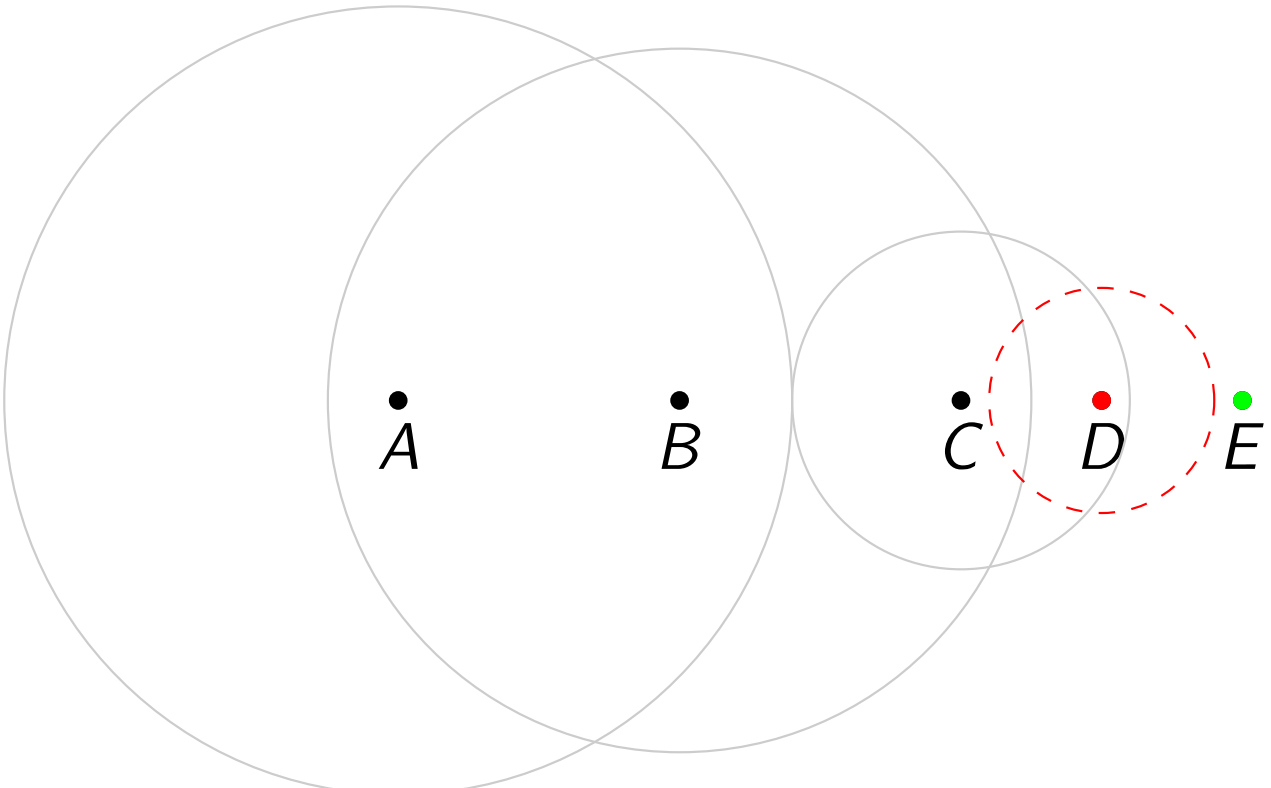
Greedy Phase

- Consider facilities $v \in F$ in non-decreasing order of radii.
- Starting with $S = \emptyset$, add v to S if $d(v, S) > 2r_v$.



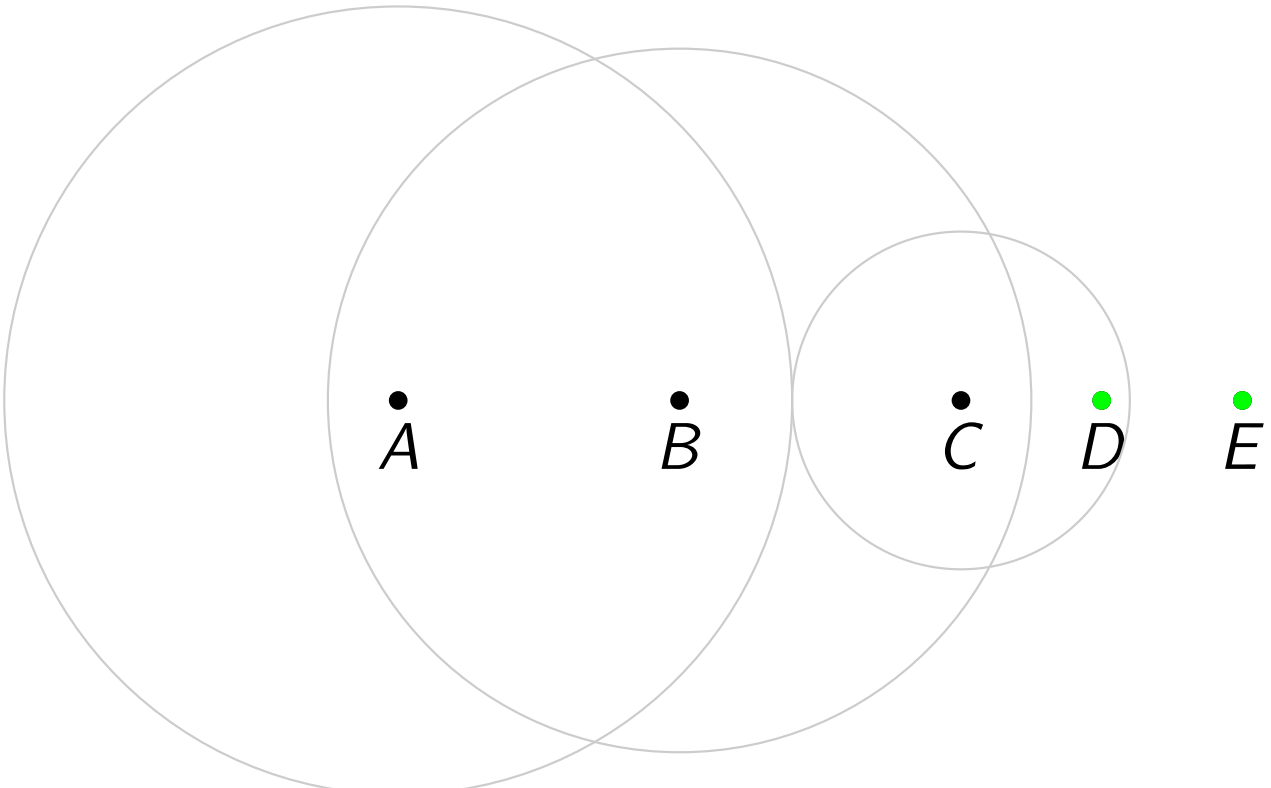
Greedy Phase

- Consider facilities $v \in F$ in non-decreasing order of radii.
- Starting with $S = \emptyset$, add v to S if $d(v, S) > 2r_v$.



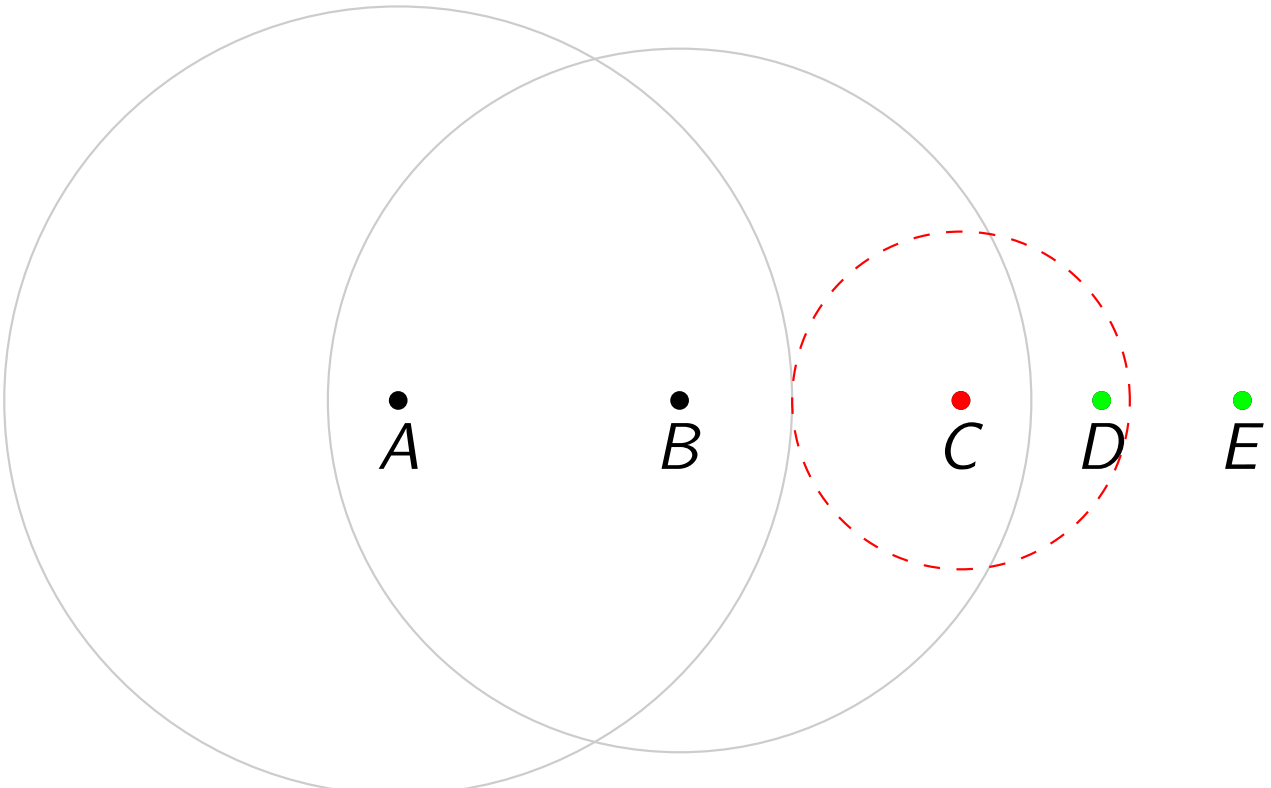
Greedy Phase

- Consider facilities $v \in F$ in non-decreasing order of radii.
- Starting with $S = \emptyset$, add v to S if $d(v, S) > 2r_v$.



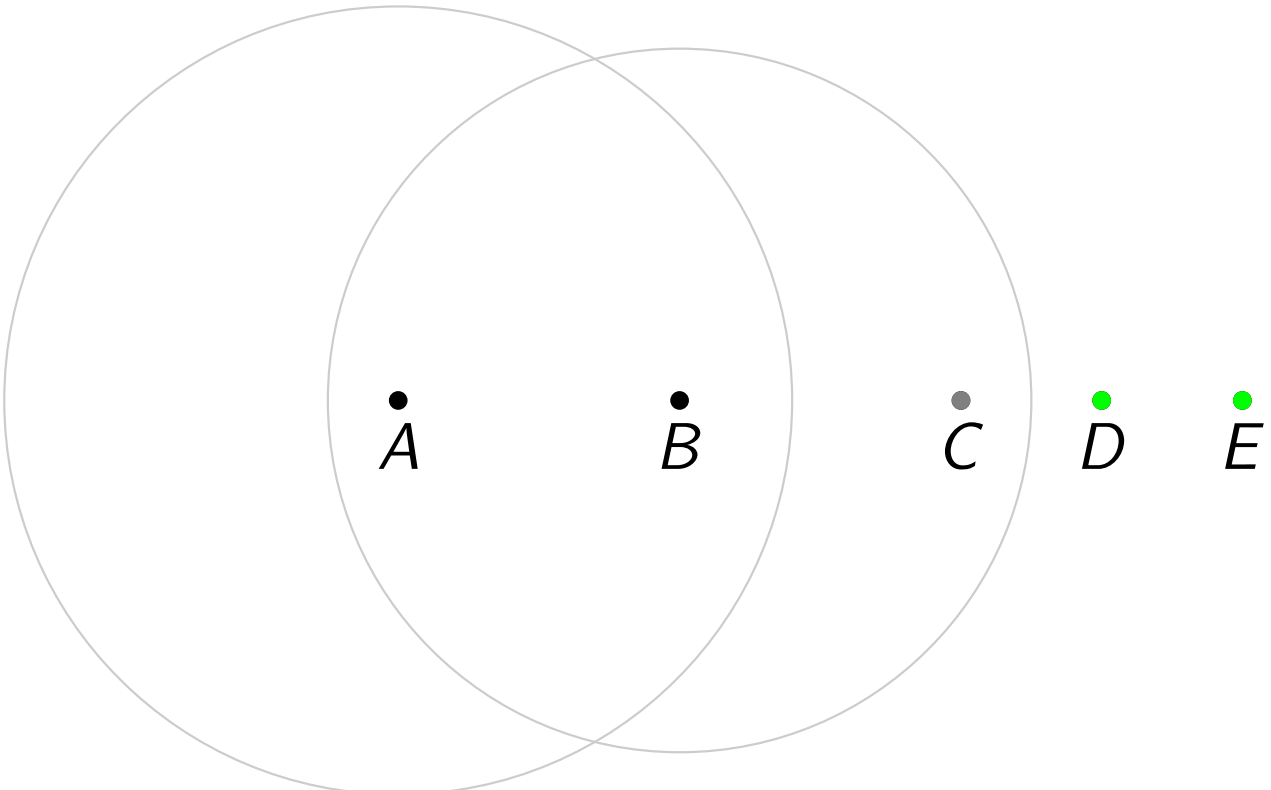
Greedy Phase

- Consider facilities $v \in F$ in non-decreasing order of radii.
- Starting with $S = \emptyset$, add v to S if $d(v, S) > 2r_v$.



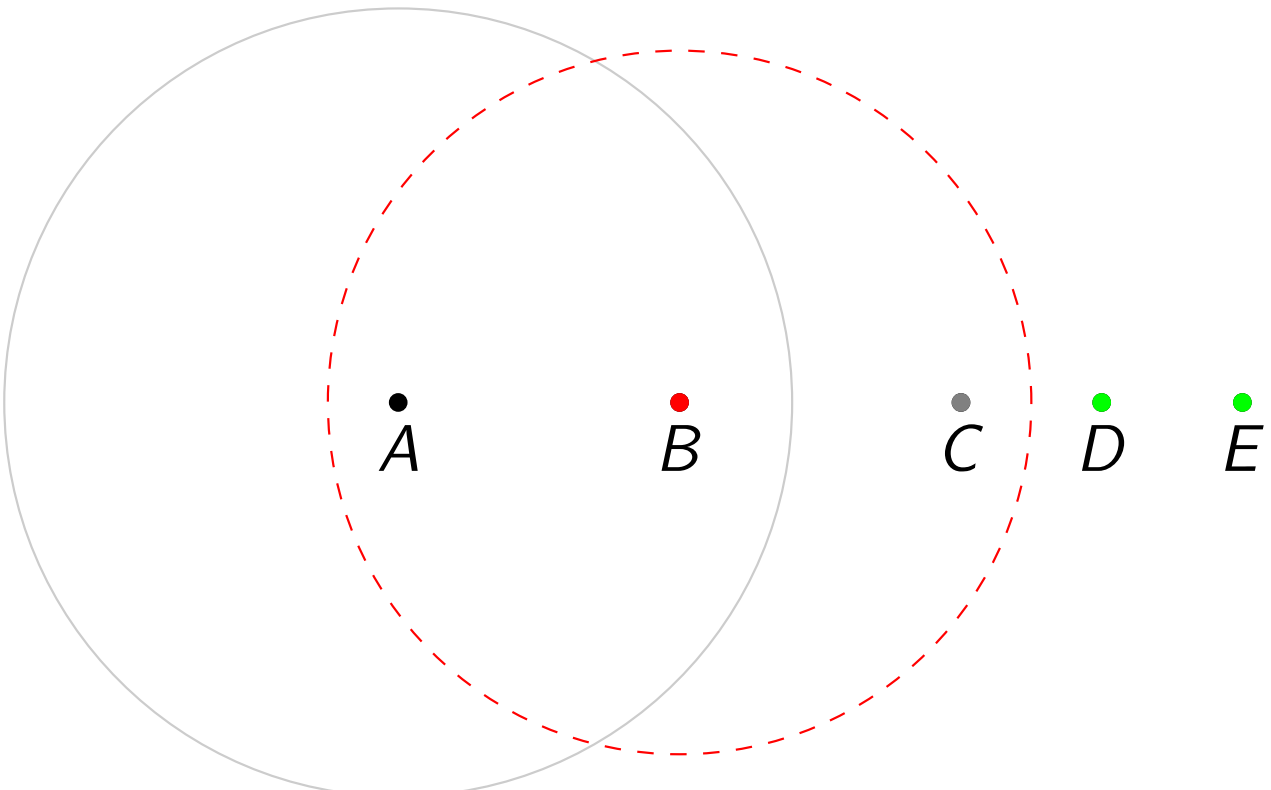
Greedy Phase

- Consider facilities $v \in F$ in non-decreasing order of radii.
- Starting with $S = \emptyset$, add v to S if $d(v, S) > 2r_v$.



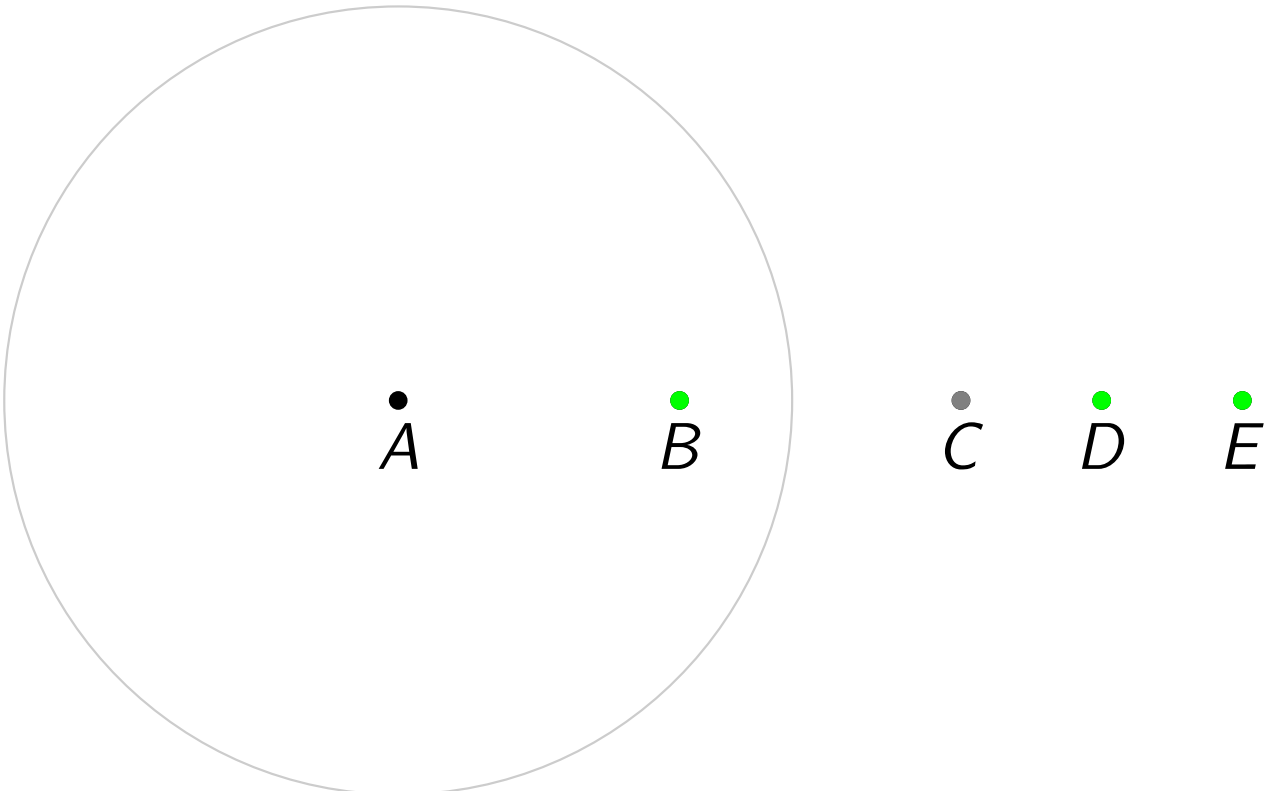
Greedy Phase

- Consider facilities $v \in F$ in non-decreasing order of radii.
- Starting with $S = \emptyset$, add v to S if $d(v, S) > 2r_v$.



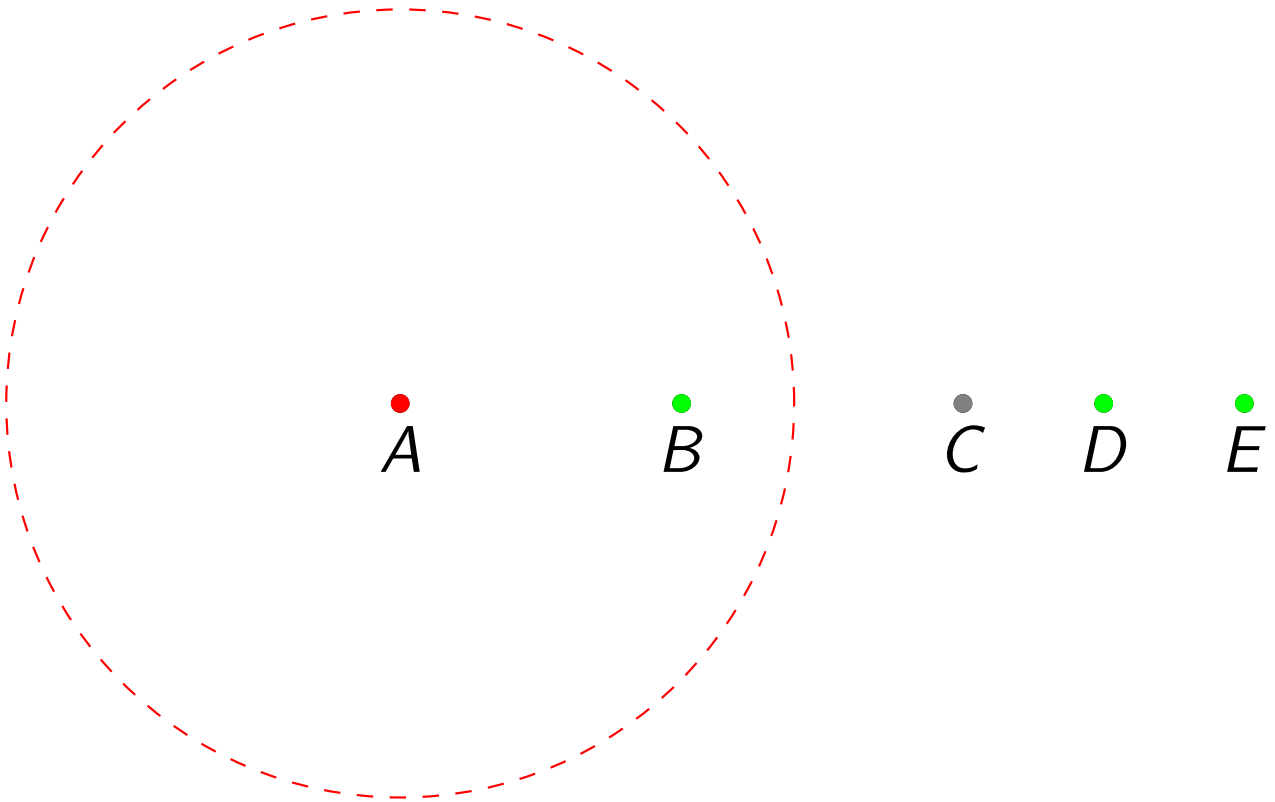
Greedy Phase

- Consider facilities $v \in F$ in non-decreasing order of radii.
- Starting with $S = \emptyset$, add v to S if $d(v, S) > 2r_v$.



Greedy Phase

- Consider facilities $v \in F$ in non-decreasing order of radii.
- Starting with $S = \emptyset$, add v to S if $d(v, S) > 2r_v$.



Greedy Phase

- Consider facilities $v \in F$ in non-decreasing order of radii.
- Starting with $S = \emptyset$, add v to S if $d(v, S) > 2r_v$.

Implementing Greedy Phase

Implementing Greedy Phase

The Greedy Phase can be implemented as a Maximal Independent Set (MIS) algorithm.

Implementing Greedy Phase

The Greedy Phase can be implemented as a Maximal Independent Set (MIS) algorithm.

In iteration i, we process facilities with r-value in range $[(1+\epsilon)^{i-1}, (1+\epsilon)^{i})$.

Implementing Greedy Phase

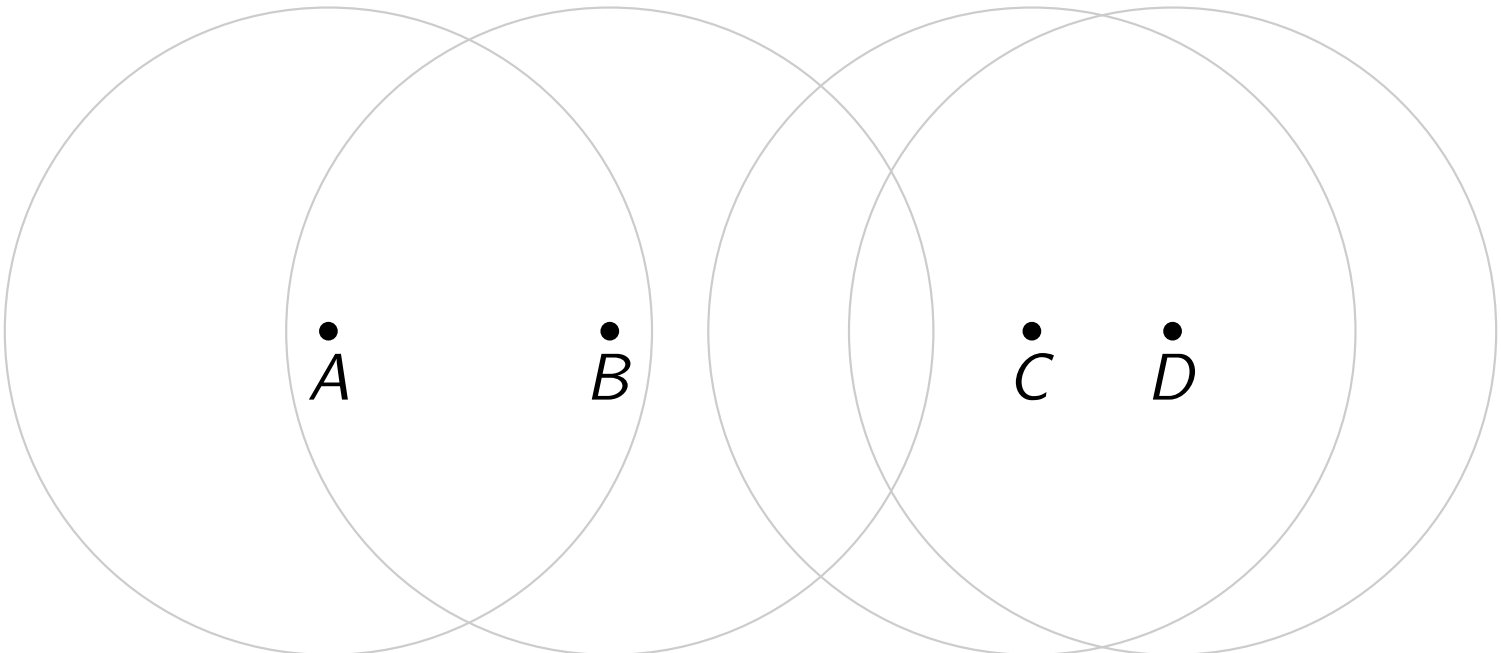
The Greedy Phase can be implemented as a Maximal Independent Set (MIS) algorithm.

In iteration i, we process facilities with r-value in range $[(1+\epsilon)^{i-1}, (1+\epsilon)^{i})$.

Implementing Greedy Phase

The Greedy Phase can be implemented as a Maximal Independent Set (MIS) algorithm.

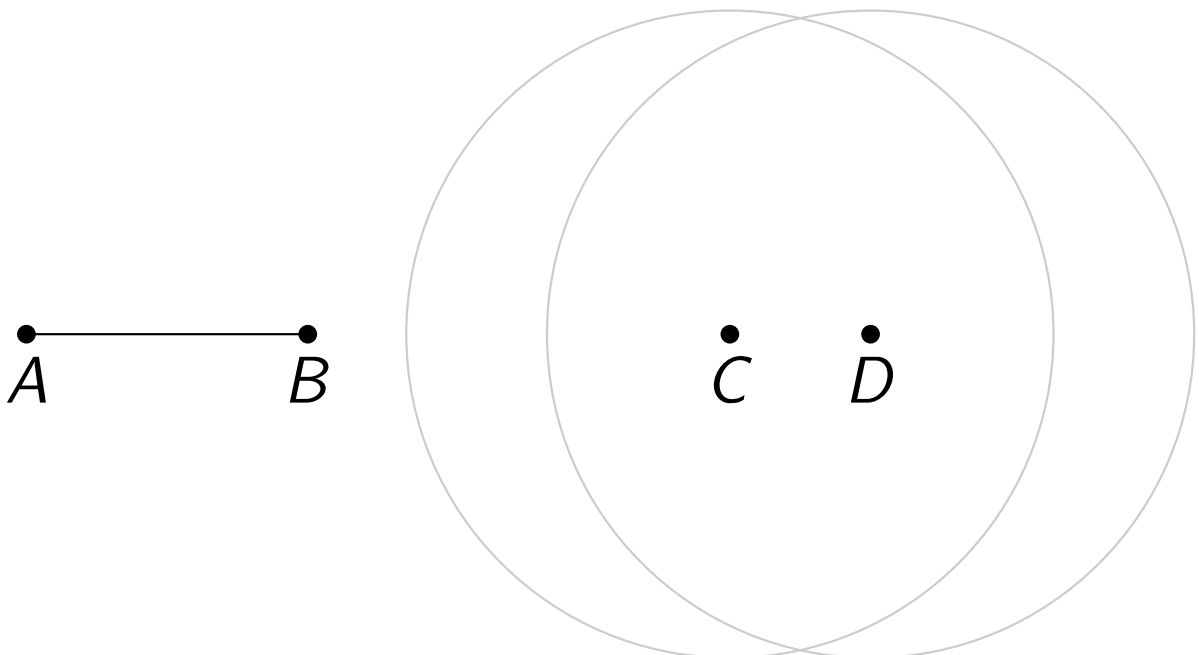
In iteration i, we process facilities with r-value in range $[(1+\epsilon)^{i-1}, (1+\epsilon)^i)$.



Implementing Greedy Phase

The Greedy Phase can be implemented as a Maximal Independent Set (MIS) algorithm.

In iteration i, we process facilities with r-value in range $[(1+\epsilon)^{i-1}, (1+\epsilon)^{i})$.



Implementing Greedy Phase

The Greedy Phase can be implemented as a Maximal Independent Set (MIS) algorithm.

In iteration i, we process facilities with r-value in range $[(1+\epsilon)^{i-1}, (1+\epsilon)^{i})$.

Implementation in Explicit Metric

Implementation in Explicit Metric

Running time is dominated by the Greedy Phase.

Implementation in Explicit Metric

Running time is dominated by the Greedy Phase.

If we compute a t-ruling set instead of MIS, we get an $O(t)$ approximation.

Implementation in Explicit Metric

Running time is dominated by the Greedy Phase.

If we compute a t-ruling set instead of MIS, we get an $O(t)$ approximation.

A t-ruling set is a generalization of MIS (which is a 1-ruling set)

Implementation in Explicit Metric

Running time is dominated by the Greedy Phase.

If we compute a t-ruling set instead of MIS, we get an $O(t)$ approximation.

A t-ruling set is a generalization of MIS (which is a 1-ruling set)

We show how to compute a 2-ruling set in $O(\log \log \log n)$ rounds of Congested Clique with high probability.

Implementation in Implicit Metric

Implementation in Implicit Metric

Running time is dominated by time required to learn relevant parts of the metric.

Implementation in Implicit Metric

Running time is dominated by time required to learn relevant parts of the metric.

Need to make polylogarithmic calls $(1+\epsilon)$-approximate SSSP subroutine.

Implementation in Implicit Metric

Running time is dominated by time required to learn relevant parts of the metric.

Need to make polylogarithmic calls $(1+\epsilon)$-approximate SSSP subroutine.

SSSP can be solved in $O(\text{poly}(\epsilon^{-1})\log^3 n)$ rounds of Congested Clique [3].

[3] Becker, Karrenbauer, Krinninger, and Lenzen: Near-Optimal Approximate Shortest Paths and Transshipment in Distributed and Streaming Models, DISC 2017

Extention to Robust Facility Location

Extention to Robust Facility Location

- Guess the costliest facility j in the optimal solution

Extention to Robust Facility Location

- Guess the costliest facility j in the optimal solution
- Add it to our solution and solve the problem by modifying facility opening costs to be

Extention to Robust Facility Location

- Guess the costliest facility j in the optimal solution
- Add it to our solution and solve the problem by modifying facility opening costs to be

$$f_i' = \begin{cases} +\infty & \text{if } f_i > f_j \\ 0 & \text{if } i = j \\ f_i & \text{otherwise} \end{cases}$$

Extention to Robust Facility Location

Extention to Robust Facility Location

Robust Facility Location —> $O(\log n)$ Facility Location computations

Extention to Robust Facility Location

Robust Facility Location —> $O(\log n)$ Facility Location computations

- Works with the Mettu-Plaxton Algorithm.

Extention to Robust Facility Location

Robust Facility Location —> $O(\log n)$ Facility Location computations

- Works with the Mettu-Plaxton Algorithm.
- Can perform all $O(\log n)$ computations in parallel.

Future Directions

Future Directions

- Can we get t-ruling set algorithms that run in $O(\log^* n)$ or $O(1)$ rounds of Congested Clique?

Future Directions

- Can we get t-ruling set algorithms that run in $O(\log^* n)$ or $O(1)$ rounds of Congested Clique?
- Can we compute exact SSSP in $O(\operatorname{poly}\log n)$ rounds of Congested Clique?

Thank you

Questions?