TOR VERGATA
UNIVERSITÀ DEGLI STUDI DI ROMA

Advanced Topics on Algorithms
2025/2026

Clustering is Hard Only When it Doesn't Matter

Lecturer: Alessandro Straziota

In previous episodes...

Why do we need algorithm analysis?

In previous episodes...

Why do we need algorithm analysis?

1. Performance Prediction

In previous episodes...

Why do we need algorithm analysis?

1. Performance Prediction
2. Identify Optimal Algorithms

In previous episodes...

Why do we need algorithm analysis?

1. Performance Prediction
2. Identify Optimal Algorithms
3. Design New Algorithms

In previous episodes...

Why do we need algorithm analysis?

1. Performance Prediction
2. Identify Optimal Algorithms
3. Design New Algorithms

Worst Case Analysis can rank algorithms inaccurately: sometime fails in identifying the right algorithm to use in practice.

In previous episodes...

Why do we need algorithm analysis?

1. Performance Prediction
2. Identify Optimal Algorithms
3. Design New Algorithms

Worst Case Analysis can rank algorithms inaccurately: sometime fails in identifying the right algorithm to use in practice.

Example (Online Paging): both `LRU` and `FWF` (or `FIFO`) have an optimal **competitive ratio**.

In previous episodes...

Why do we need algorithm analysis?

1. Performance Prediction
2. Identify Optimal Algorithms
3. Design New Algorithms

Worst Case Analysis can rank algorithms inaccurately: sometime fails in identifying the right algorithm to use in practice.

Example (Online Paging): both `LRU` and `FWF` (or `FIFO`) have an optimal **competitive ratio**.

$\Longrightarrow$ the theory cannot differentiate `LRU` from `FWF` or `FIFO`.

In previous episodes...

Why do we need algorithm analysis?

1. Performance Prediction
2. Identify Optimal Algorithms
3. Design New Algorithms

Worst Case Analysis can rank algorithms inaccurately: sometime fails in identifying the right algorithm to use in practice.

Example (Online Paging): both `LRU` and `FWF` (or `FIFO`) have an optimal **competitive ratio**.

$\Longrightarrow$ the theory cannot differentiate `LRU` from `FWF` or `FIFO`.

But `FWF` is clearly worse than `LRU`!

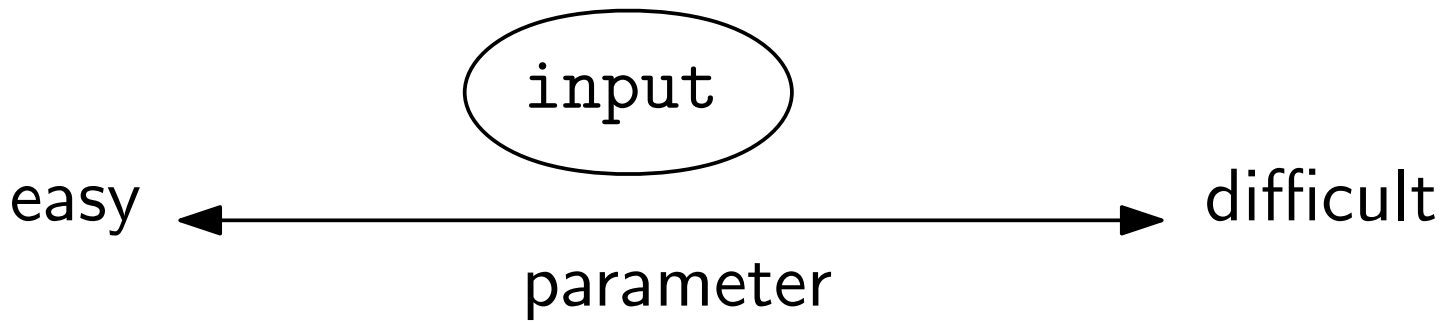
In previous episodes...

Parameterized analysis: fine-grained understanding of the performance of an algorithm on all inputs, not just on worst-case inputs.

In previous episodes...

Parameterized analysis: fine-grained understanding of the performance of an algorithm on all inputs, not just on worst-case inputs.

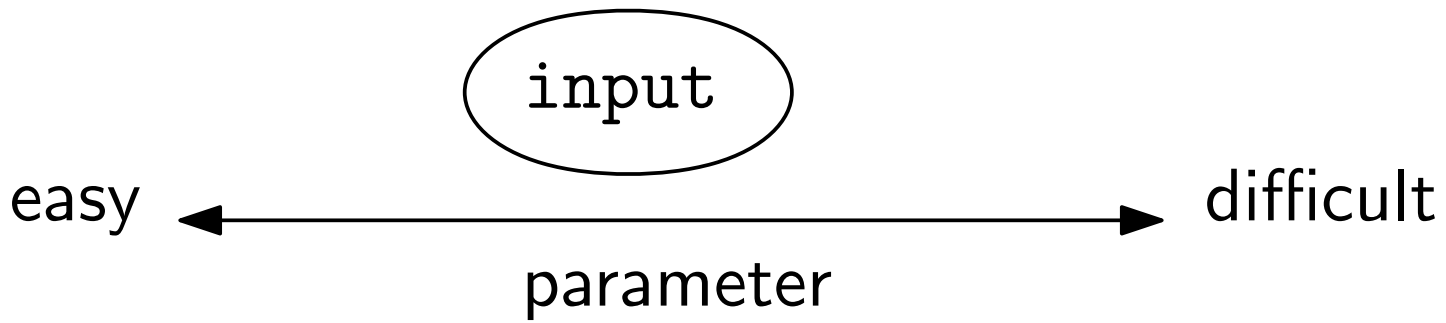
Idea: some problems are difficult only for some parameter values.



In previous episodes...

Parameterized analysis: fine-grained understanding of the performance of an algorithm on all inputs, not just on worst-case inputs.

Idea: some problems are difficult only for some parameter values.

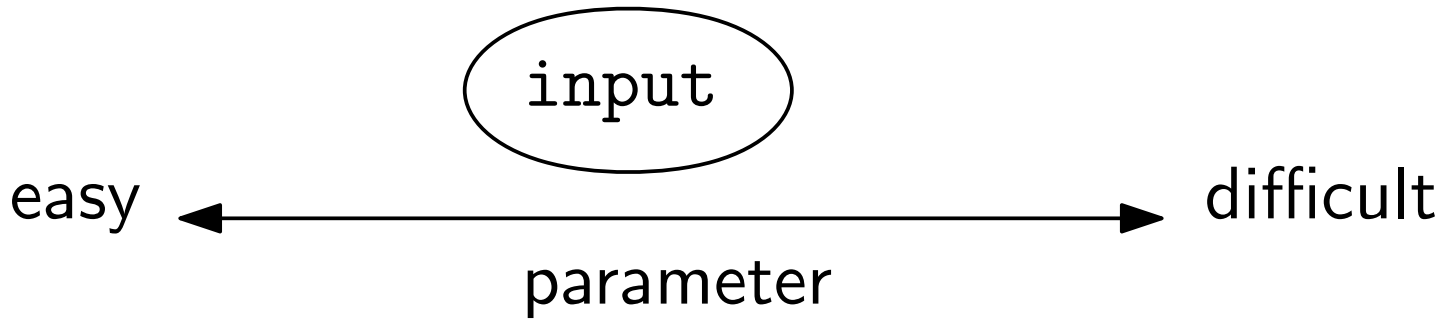


Example (Working Set Model): the maximum number of **distinct** page requests in every window length w.

In previous episodes...

Parameterized analysis: fine-grained understanding of the performance of an algorithm on all inputs, not just on worst-case inputs.

Idea: some problems are difficult only for some parameter values.



Example (Working Set Model): the maximum number of **distinct** page requests in every window length w.

Claim. `LRU` has a **page fault rate** strictly lower than `FIFO`.

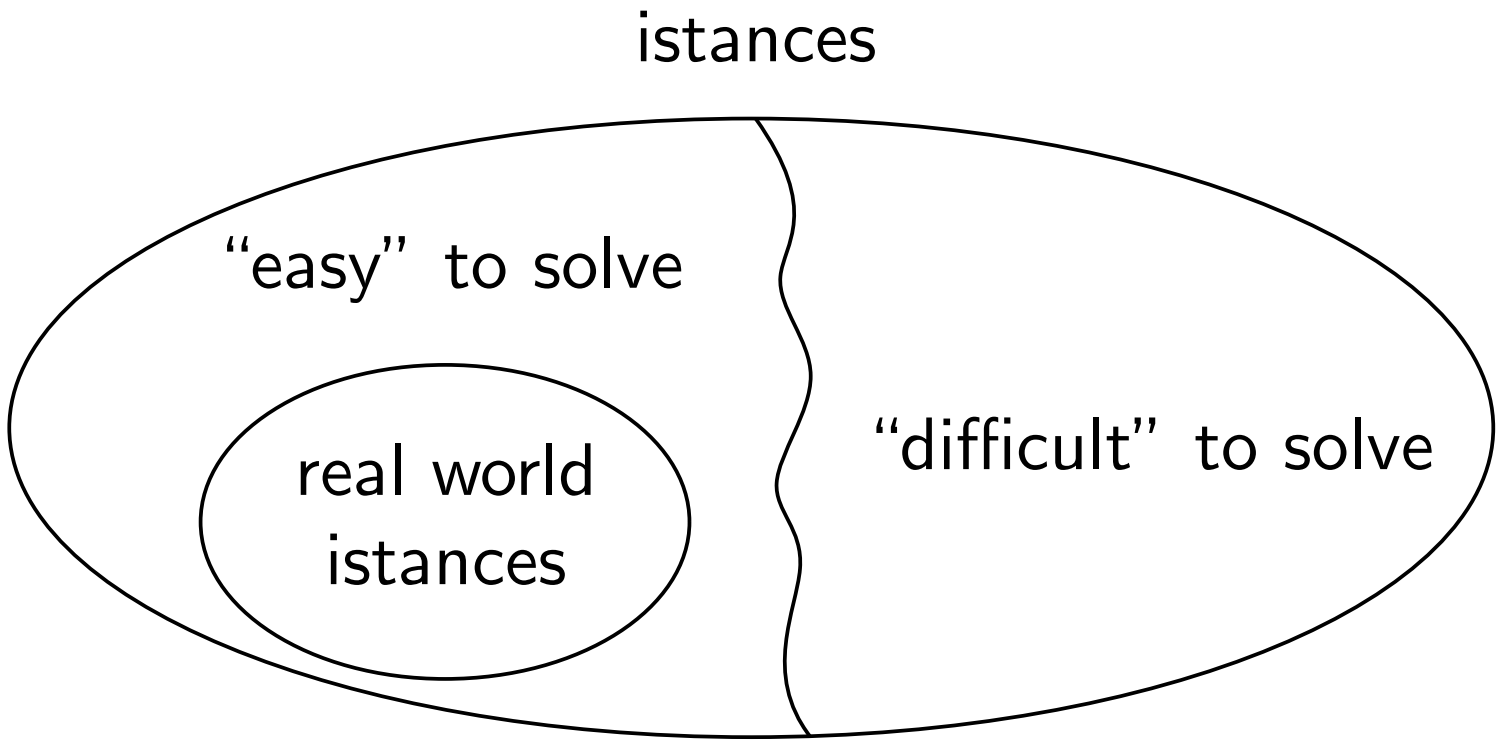
In this episode...

In this episode...

Insight: in most applications we only care about instances in which there is a "meaningful solution", that are computationally easier to solve than worst-case instances.

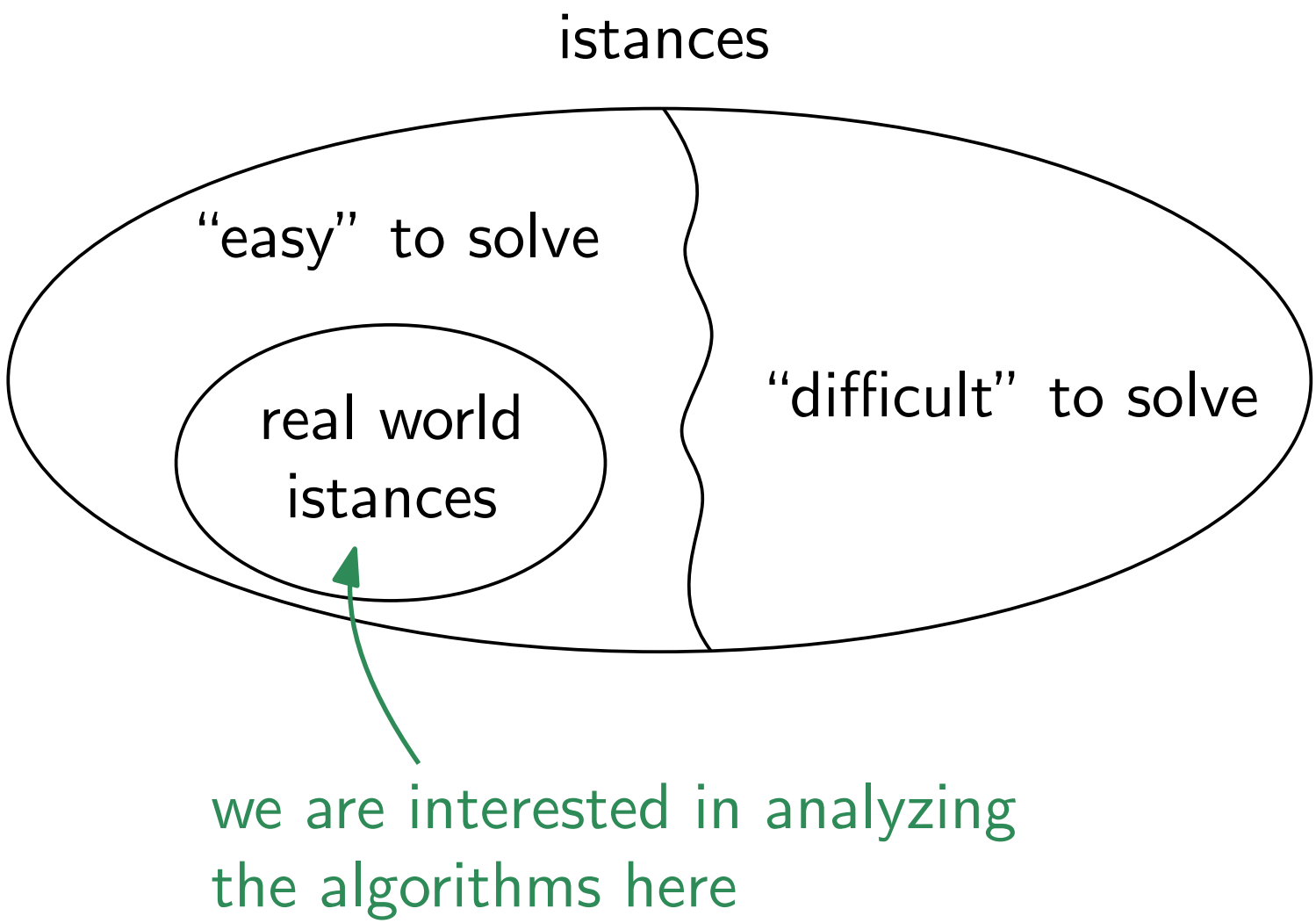
In this episode...

Insight: in most applications we only care about instances in which there is a "meaningful solution", that are computationally easier to solve than worst-case instances.



In this episode...

Insight: in most applications we only care about instances in which there is a "meaningful solution", that are computationally easier to solve than worst-case instances.



Beyond Worst-Case Analysis
Episode III

Clustering is Hard Only When it Doesn't Matter

Clustering

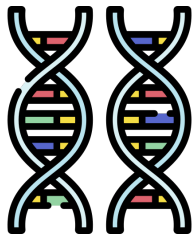
Input: a set of data.

Clustering

Input: a set of data.

documents

figures



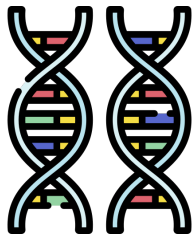
genomes

Clustering

Input: a set of data.

documents

figures



genomes

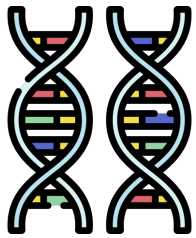
Goal: group homogeneous data into sets (**clusters**) based on their similarity.

Clustering

Input: a set of data.

documents

figures



genomes

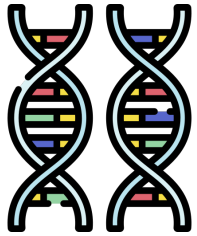
Goal: group homogeneous data into sets (**clusters**) based on their similarity.

Clustering

Input: a set of data.

documents

figures



genomes

Goal: group homogeneous data into sets (**clusters**) based on their similarity.

Cluster 1

Cluster 2

Cluster 3

Clustering

Is it a simple problem?

Clustering

Is it a simple problem?

Is it simple?

Clustering

Is it a simple problem?

... and this?

Clustering

Is it a simple problem?

... and what about this ???

Clustering

How do we represent data?

Clustering

How do we represent data?

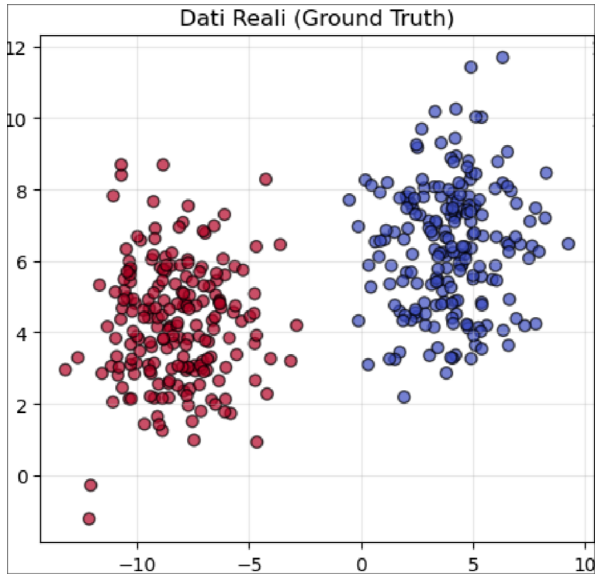
We represent our data as a set $\mathcal{X} = \{x_1, \ldots, x_n\}$ of n points in a **metric space** $(\mathcal{M}, d)$.

Clustering

How do we represent data?

We represent our data as a set $\mathcal{X} = \{x_1, \ldots, x_n\}$ of n points in a **metric space** $(\mathcal{M}, d)$.

data



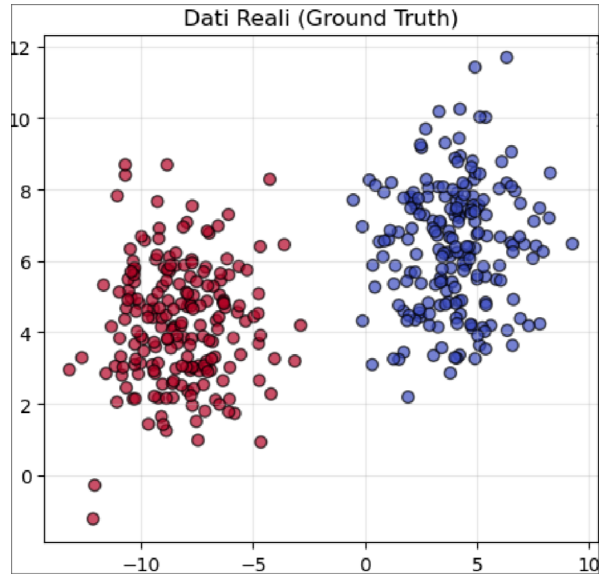
vector representation

Clustering

How do we represent data?

We represent our data as a set $\mathcal{X} = \{x_1, \ldots, x_n\}$ of n points in a **metric space** $(\mathcal{M}, d)$.

data



vector representation

Distance function: $d : \mathcal{M} \times \mathcal{M} \to \mathbb{R}^+$ such that

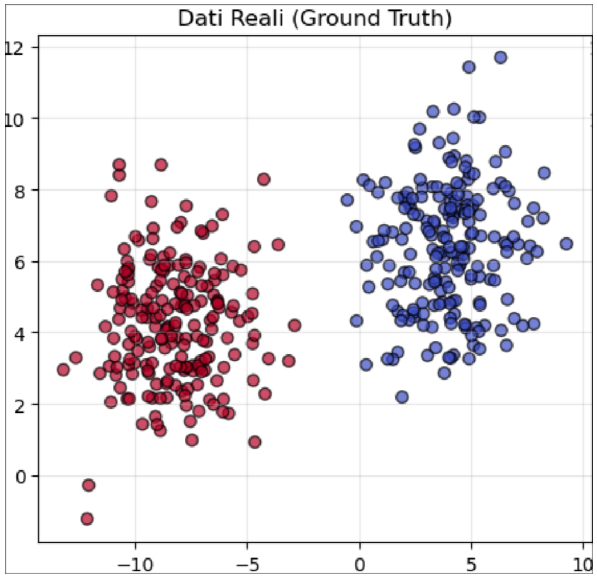
- $d(x, x) = 0$
- $d(x, y) = d(y, x)$
- $d(x, y) \leq d(x, z) + d(z, y)$

Clustering

How do we represent data?

We represent our data as a set $\mathcal{X} = \{x_1, \ldots, x_n\}$ of n points in a **metric space** $(\mathcal{M}, d)$.

data



vector representation

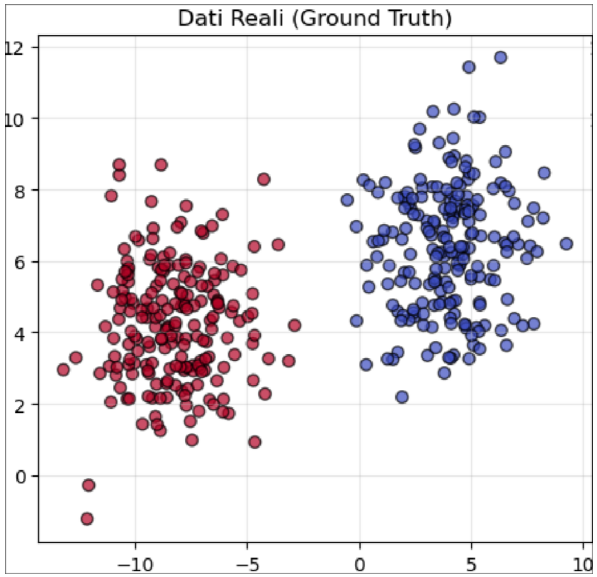
How to group similar points?

Clustering

How do we represent data?

We represent our data as a set $\mathcal{X} = \{x_1, \ldots, x_n\}$ of n points in a **metric space** $(\mathcal{M}, d)$.

data



vector representation

How to group similar points?

Through an optimization problem!

k-Median Clustering

k-Median Clustering

Input: a set $\mathcal{X} = \{x_1, \ldots, x_n\}$ of n points in a metric space $(\mathcal{M}, d)$.

k-Median Clustering

Input: a set $\mathcal{X} = \{x_1, \ldots, x_n\}$ of n points in a metric space $(\mathcal{M}, d)$.

Goal: a set $S \subseteq \mathcal{X}$ of k **centers** $c_1, \ldots, c_k$, which minimize the sum of the distances of the points from the nearest center.

$$\min_{\substack{S \subseteq \mathcal{X} \\ |S|=k}} \sum_{x \in \mathcal{X}} \min_{c_i \in S} d(x, c_i)$$

k-Median Clustering

Input: a set $\mathcal{X} = \{x_1, \ldots, x_n\}$ of n points in a metric space $(\mathcal{M}, d)$.

Goal: a set $S \subseteq \mathcal{X}$ of k **centers** $c_1, \ldots, c_k$, which minimize the sum of the distances of the points from the nearest center.

$$\min_{\substack{S \subseteq \mathcal{X} \\ |S|=k}} \sum_{x \in \mathcal{X}} \min_{c_i \in S} d(x, c_i)$$

Obs: each center c_i defines a cluster C_i, and $C_1, \ldots, C_k$ are a partition of $\mathcal{X}$.

k-Median Clustering

Input: a set $\mathcal{X} = \{x_1, \ldots, x_n\}$ of n points in a metric space $(\mathcal{M}, d)$.

Goal: a set $S \subseteq \mathcal{X}$ of k **centers** $c_1, \ldots, c_k$, which minimize the sum of the distances of the points from the nearest center.

$$\min_{\substack{S \subseteq \mathcal{X} \\ |S|=k}} \sum_{x \in \mathcal{X}} \min_{c_i \in S} d(x, c_i)$$

Obs: each center c_i defines a cluster C_i, and $C_1, \ldots, C_k$ are a partition of $\mathcal{X}$.

Does it work well?

k-Median Clustering

Input: a set $\mathcal{X} = \{x_1, \ldots, x_n\}$ of n points in a metric space $(\mathcal{M}, d)$.

Goal: a set $S \subseteq \mathcal{X}$ of k **centers** $c_1, \ldots, c_k$, which minimize the sum of the distances of the points from the nearest center.

$$\min_{\substack{S \subseteq \mathcal{X} \\ |S|=k}} \sum_{x \in \mathcal{X}} \min_{c_i \in S} d(x, c_i)$$

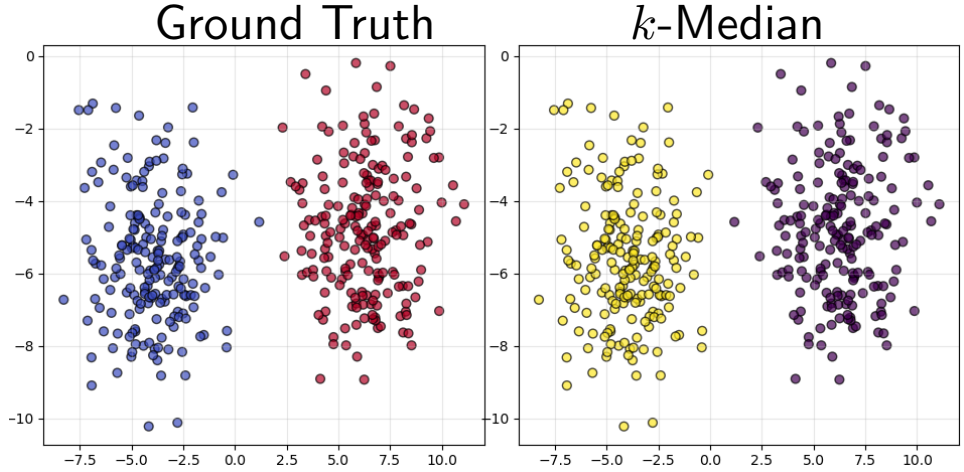
Obs: each center c_i defines a cluster C_i, and $C_1, \ldots, C_k$ are a partition of $\mathcal{X}$.

Does it work well?

it depends...

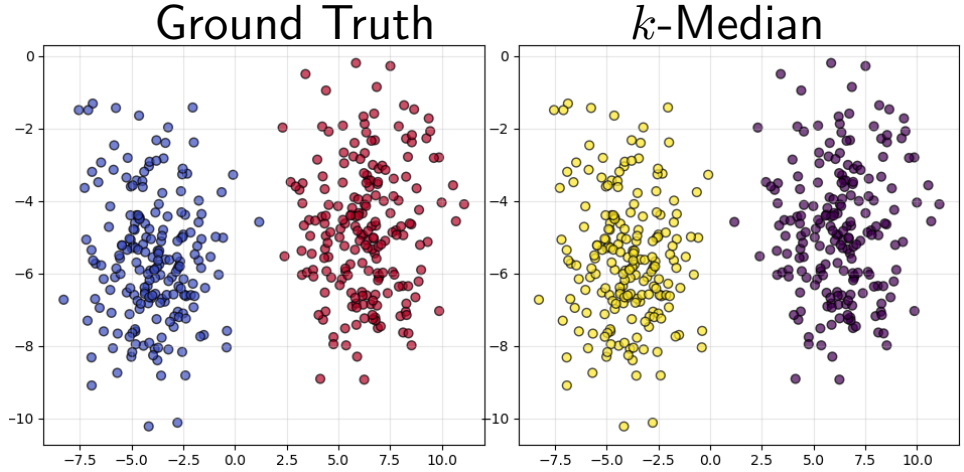
k-Median Clustering

Sometimes it works well, capturing intrinsic patterns in the data.

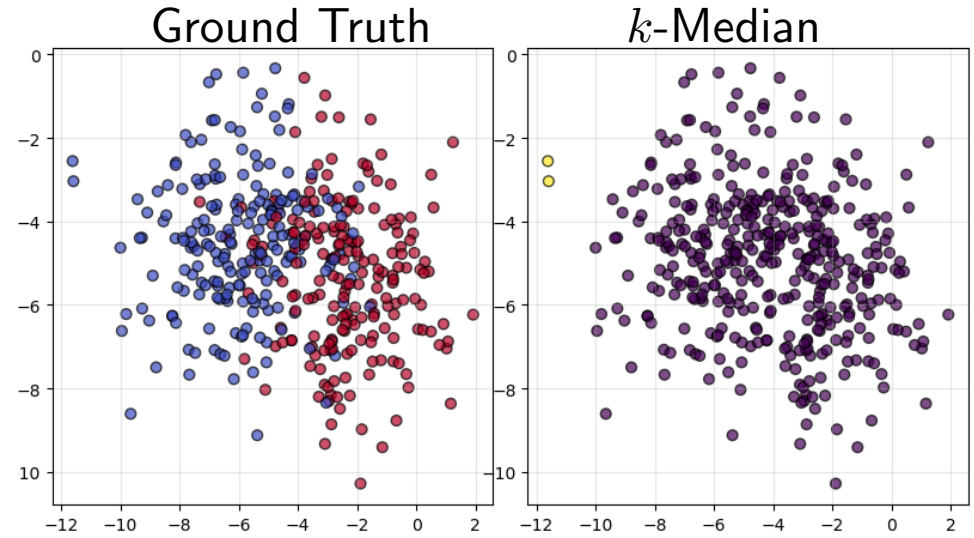


k-Median Clustering

Sometimes it works well, capturing intrinsic patterns in the data.

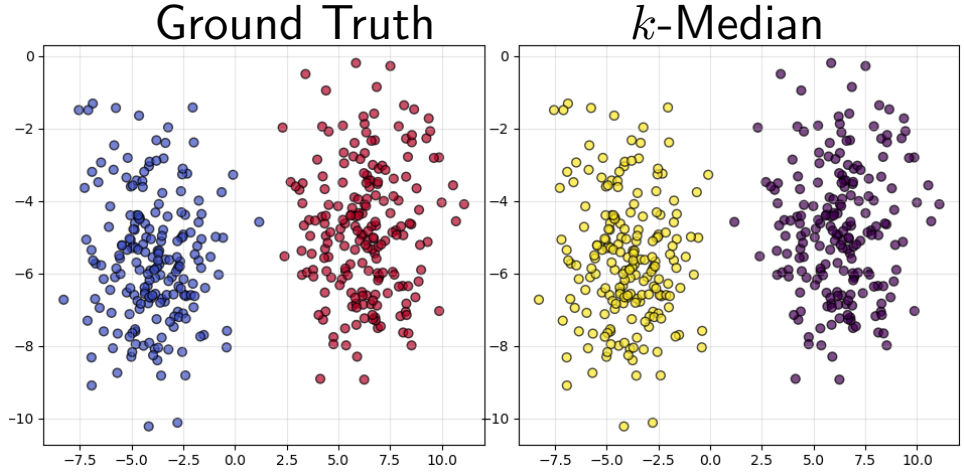


Sometimes it works really bad.

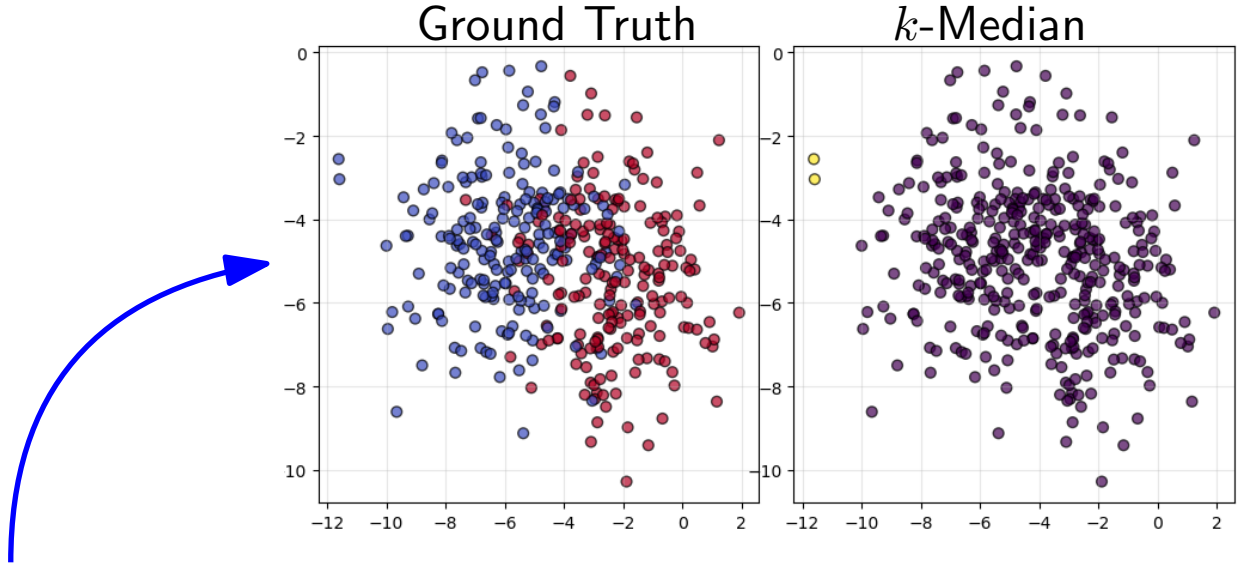


k-Median Clustering

Sometimes it works well, capturing intrinsic patterns in the data.



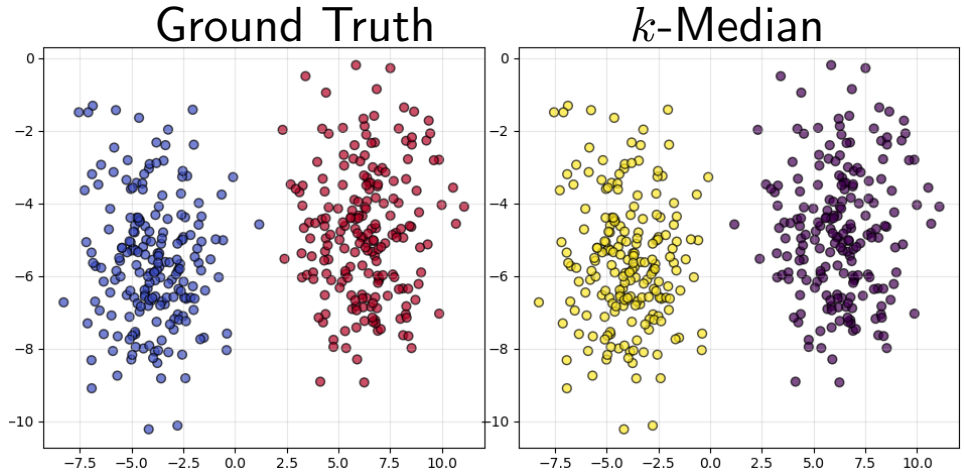
Sometimes it works really bad.



Good news: it's not an interesting case :)

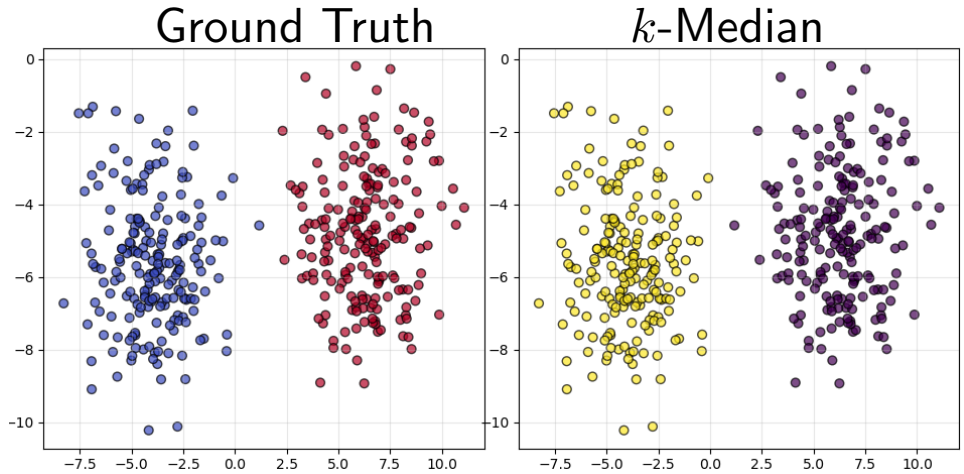
k-Median Clustering

If we are doing clustering, it means we are interested in instances for which exists a "**meaningful solution**".



k-Median Clustering

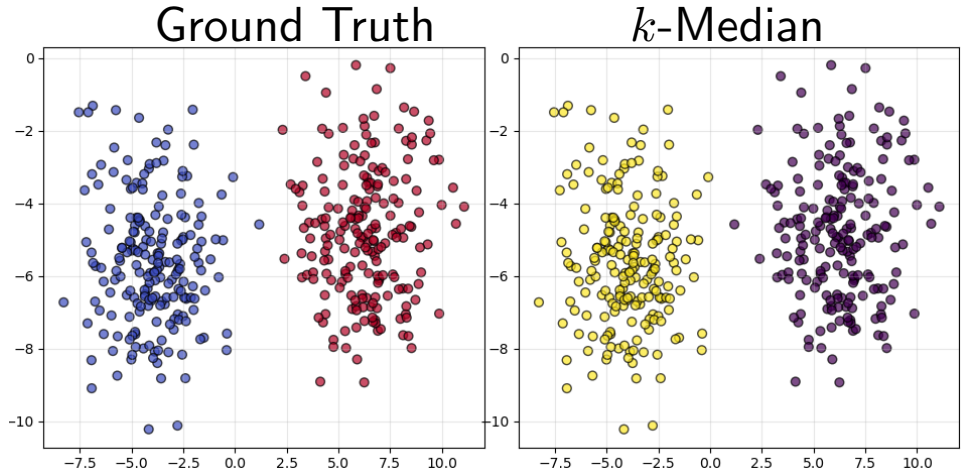
If we are doing clustering, it means we are interested in instances for which exists a "**meaningful solution**".



Hardness: k-media clustering is NP-hard, even to approximate better then $\approx 1.736...$

k-Median Clustering

If we are doing clustering, it means we are interested in instances for which exists a "**meaningful solution**".

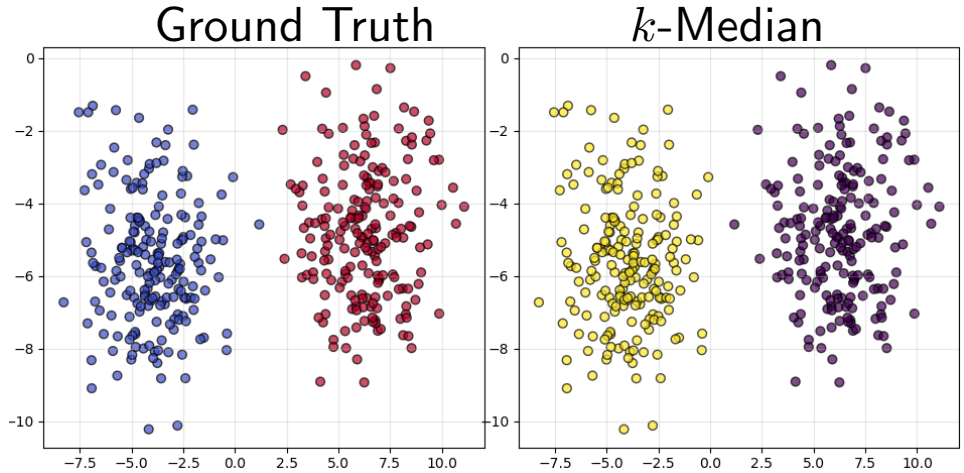


Hardness: k-media clustering is NP-hard, even to approximate better then $\approx 1.736...$

... but if we restrict to the significant istances, it can be solved in polynomial time!

k-Median Clustering

If we are doing clustering, it means we are interested in instances for which exists a "**meaningful solution**".



Hardness: k-media clustering is NP-hard, even to approximate better then $\approx 1.736...$

... but if we restrict to the significant istances, it can be solved in polynomial time!

clustering is hard only when it doesn't matter!

Perturbation Stability

Perturbation Stability

Def. (γ-perturmation) For $\gamma \geq 1$, a γ-perturbation of a metric space $(\mathcal{M}, d)$ is another metric space $(\mathcal{M}, d')$ such that

$$d(x, y) \leq d'(x, y) \leq \gamma d(x, y) \;\; \forall x, y \in \mathcal{M}.$$

Perturbation Stability

Def. (γ-perturmation) For $\gamma \geq 1$, a γ-perturbation of a metric space $(\mathcal{M}, d)$ is another metric space $(\mathcal{M}, d')$ such that

$$d(x, y) \leq d'(x, y) \leq \gamma d(x, y) \ \ \forall x, y \in \mathcal{M}.$$

Def. (Perturbation stability) A k-median istance (X, d) is γ-perturbation-stabel if there exists a solution $S^* = \{c_1^*, \ldots, c_k^*\}$ such that, for every γ-perturbation (X', d), S^* is the unique optimal solution.

Perturbation Stability

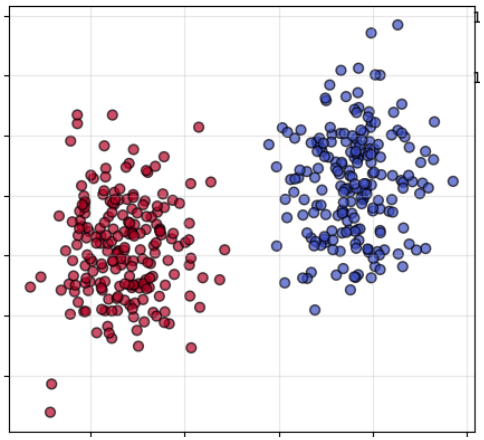
Def. (γ-perturmation) For $\gamma \geq 1$, a γ-perturbation of a metric space $(\mathcal{M}, d)$ is another metric space $(\mathcal{M}, d')$ such that

$$d(x, y) \leq d'(x, y) \leq \gamma d(x, y) \quad \forall x, y \in \mathcal{M}.$$

Def. (Perturbation stability) A k-median istance (X, d) is γ-perturbation-stabel if there exists a solution $S^* = \{c_1^*, \ldots, c_k^*\}$ such that, for every γ-perturbation (X', d), S^* is the unique optimal solution.

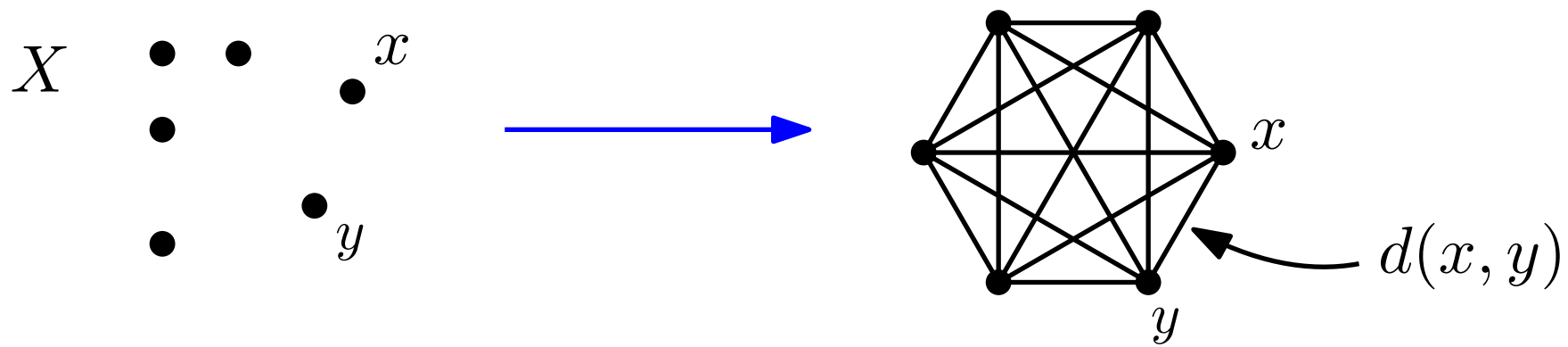
Intuition: the clusters are well separated, so even changing the metric (provided it is not too different) the solution S^* remains optimal (and unique).

interesting istance



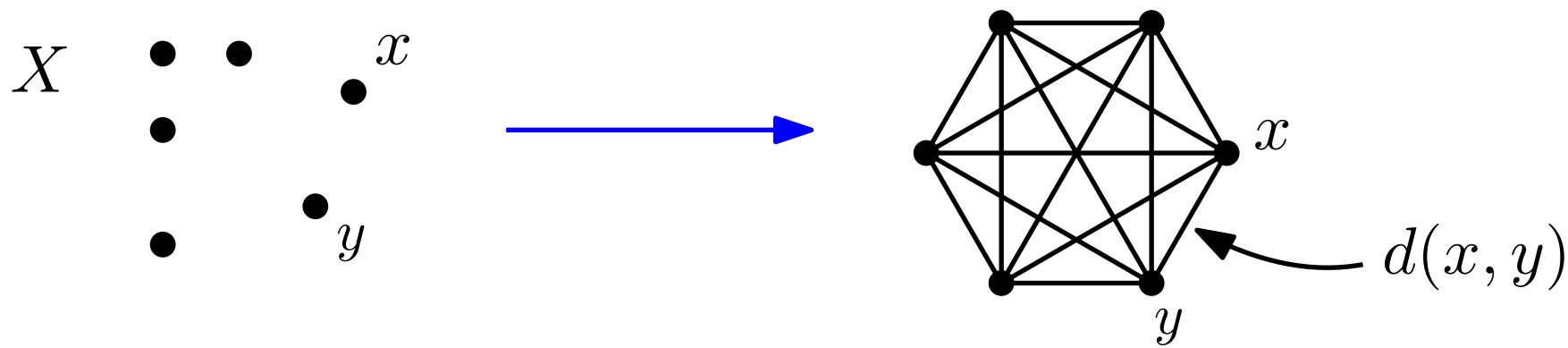
Single-Link Algorithm

Setp 1. Think the input (X, d) as a complete graph with vertices X and edge weights given by d.

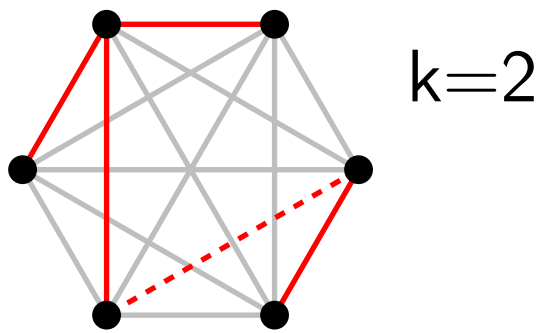


Single-Link Algorithm

Setp 1. Think the input (X, d) as a complete graph with vertices X and edge weights given by d.

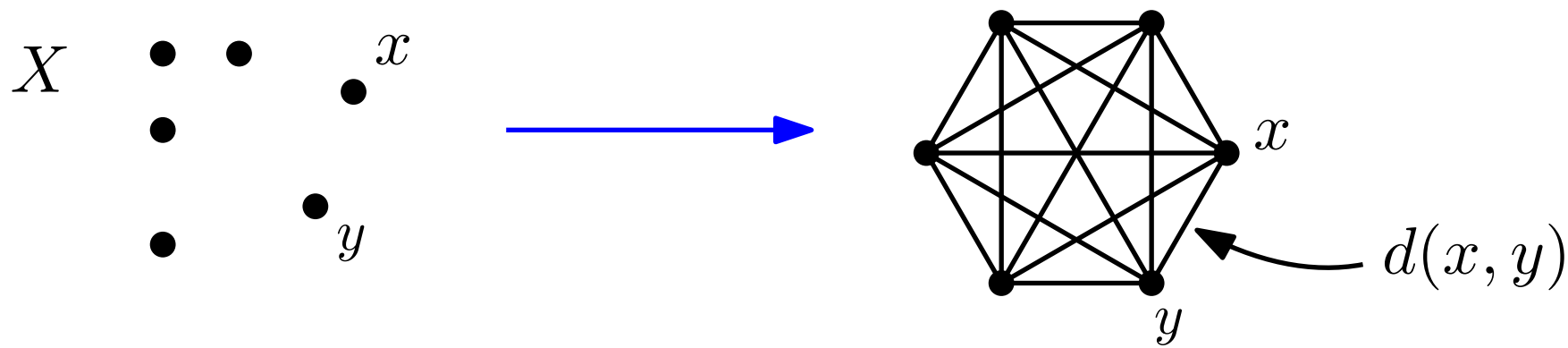


Step 2. runs Kruskal's minimum spanning tree (MST) algorithm, but stop when there are k connected components. That is, we skip the final $k-1$ edge additions of Kruskal's algorithm.

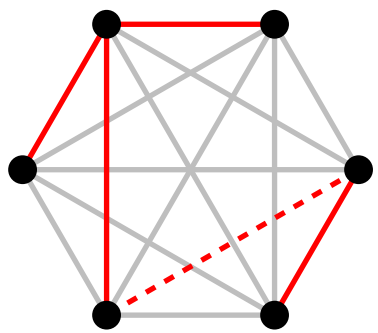


Single-Link Algorithm

Setp 1. Think the input (X, d) as a complete graph with vertices X and edge weights given by d.



Step 2. runs Kruskal's minimum spanning tree (MST) algorithm, but stop when there are k connected components. That is, we skip the final $k-1$ edge additions of Kruskal's algorithm.

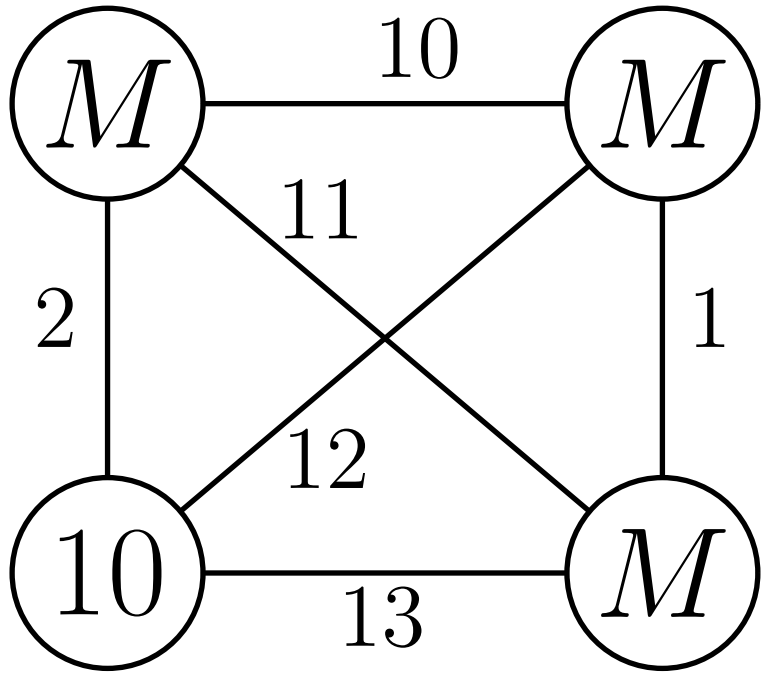


Does it give a goos solution?

Counterexample

$k = 3, M \gg 10$

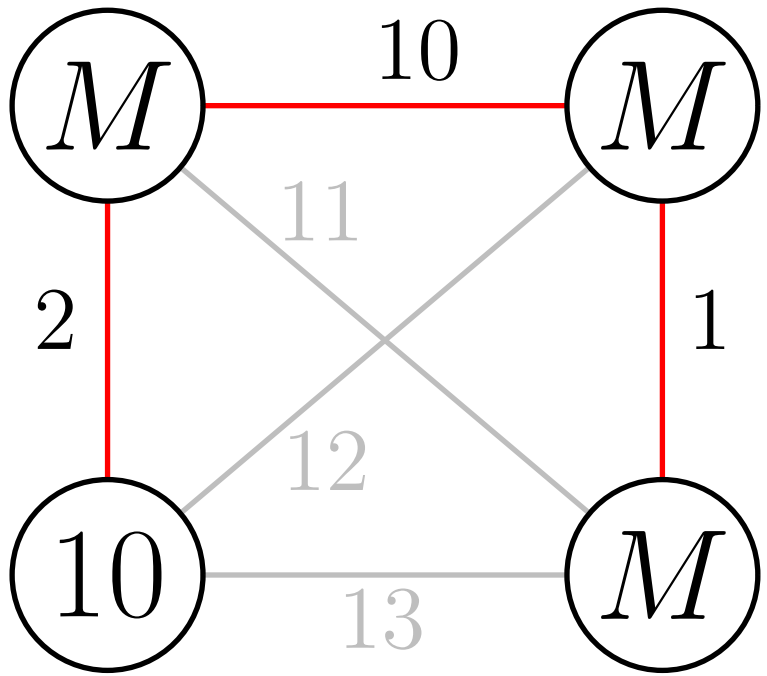
The circle represent M (or 10) co-located points



Counterexample

$k = 3, M \gg 10$

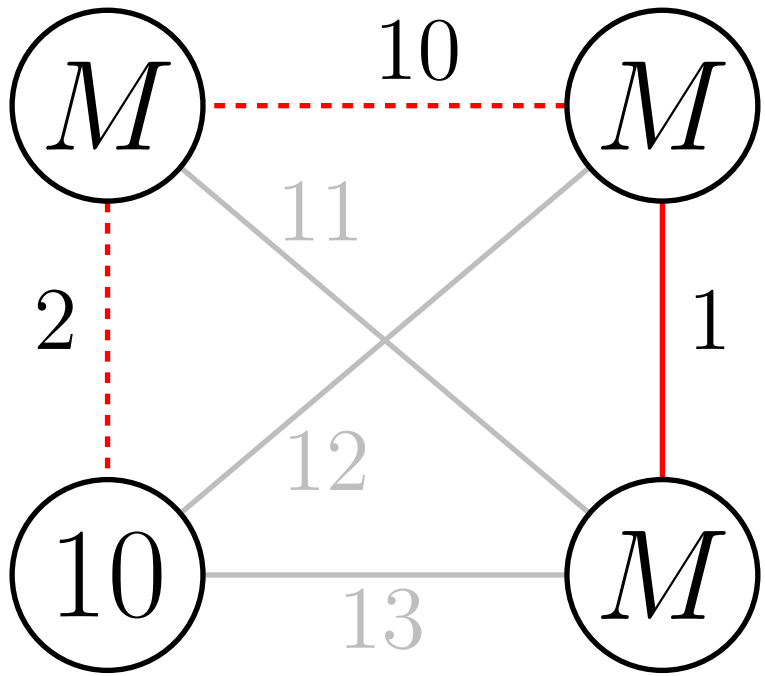
The circle represent M (or 10) co-located points



Counterexample

$k = 3, M \gg 10$

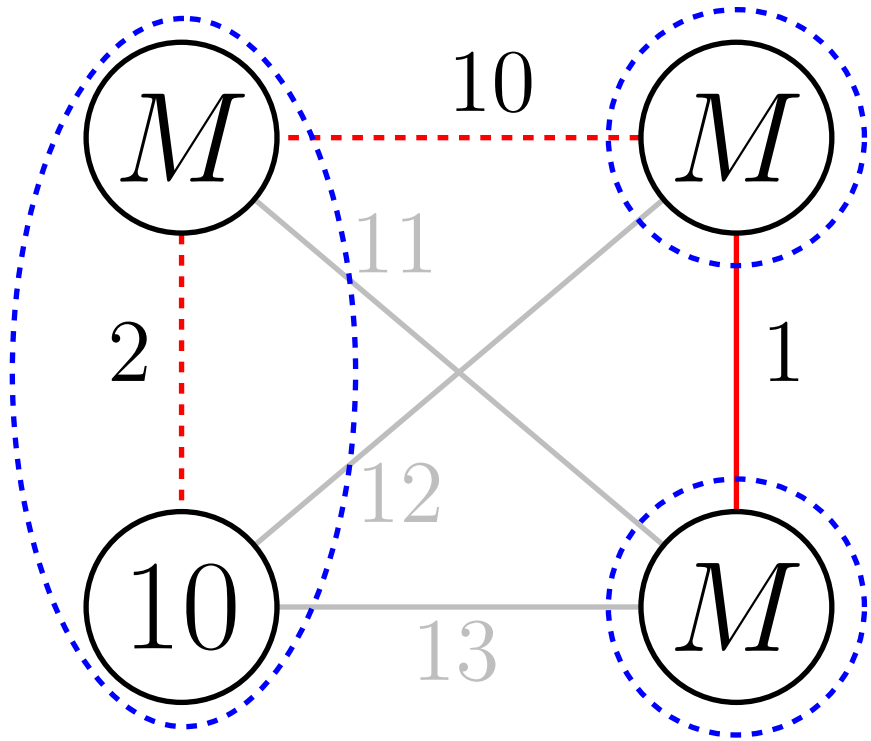
The circle represent M (or 10) co-located points



Counterexample

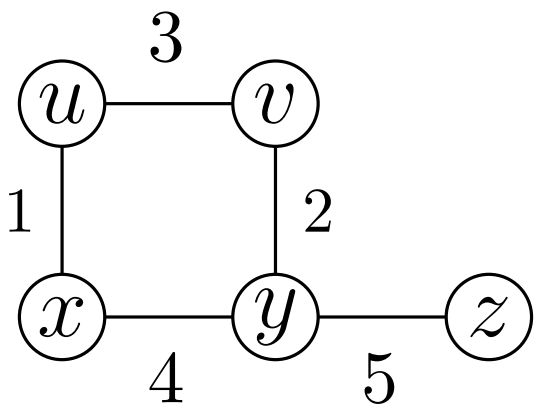
$k = 3, M \gg 10$

The circle represent M (or 10) co-located points



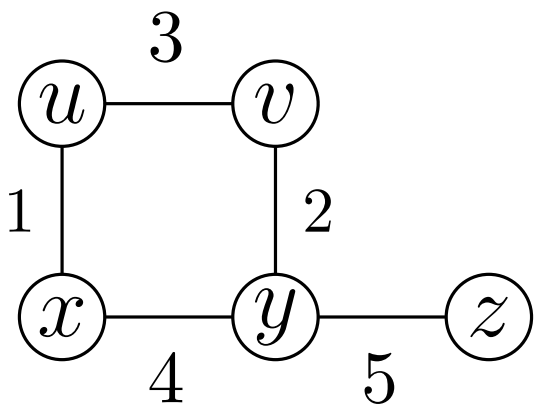
optimal solution

Hierarchical Cluster Tree



G

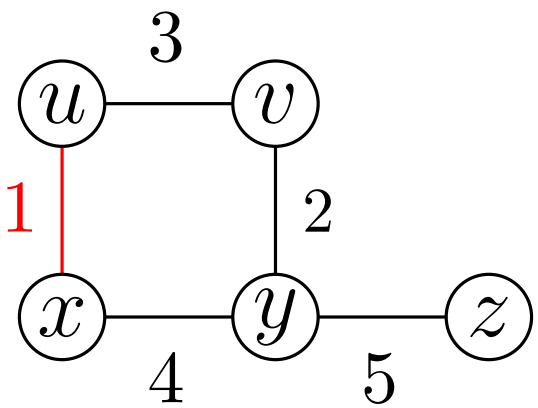
Hierarchical Cluster Tree



G

H

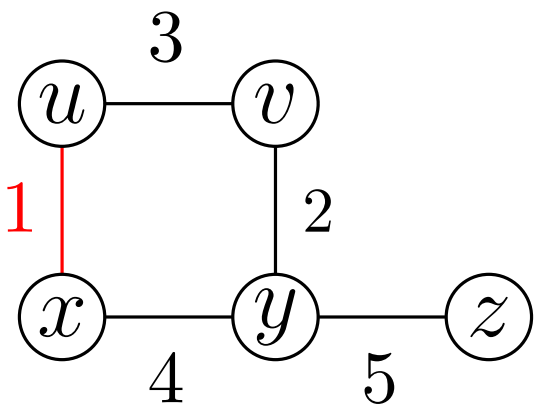
Hierarchical Cluster Tree



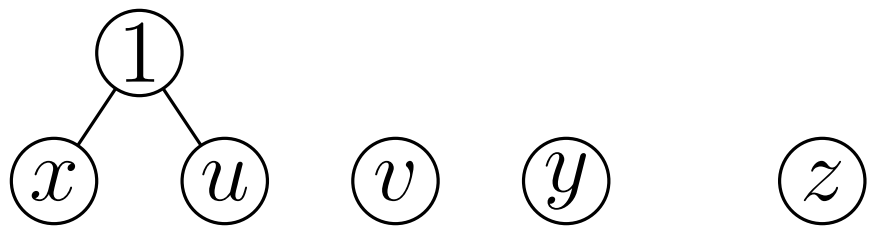
G

H

Hierarchical Cluster Tree

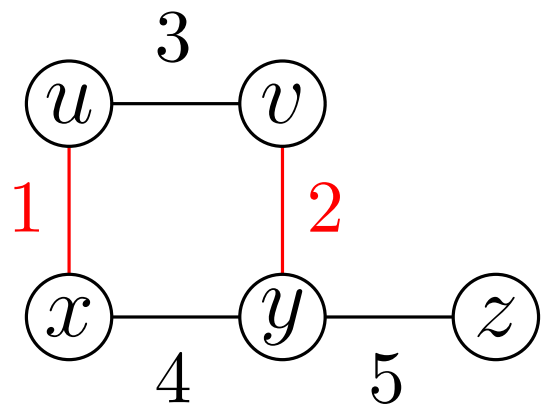


G

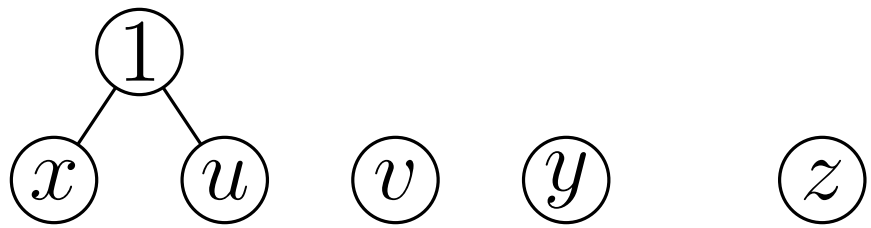


H

Hierarchical Cluster Tree

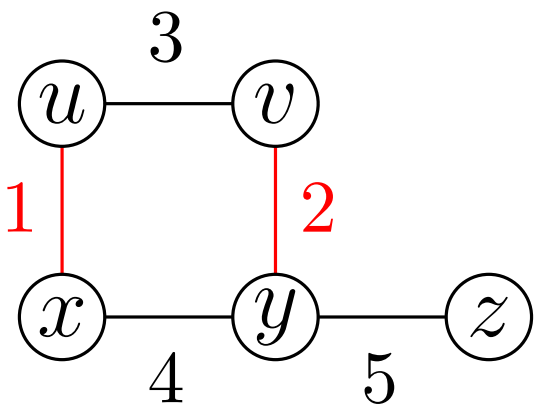


G

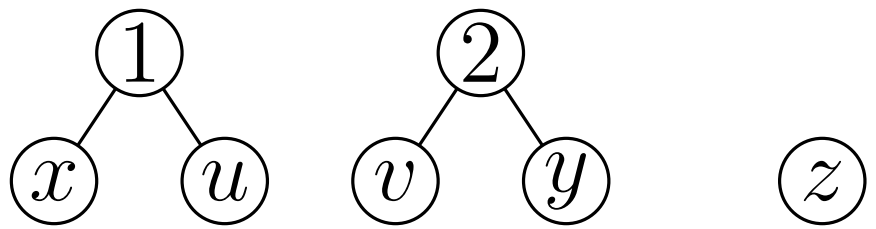


H

Hierarchical Cluster Tree

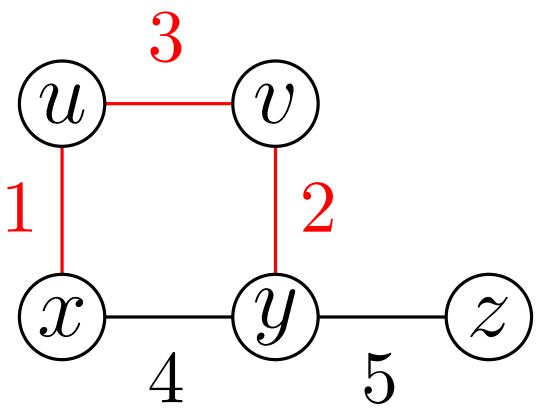


G

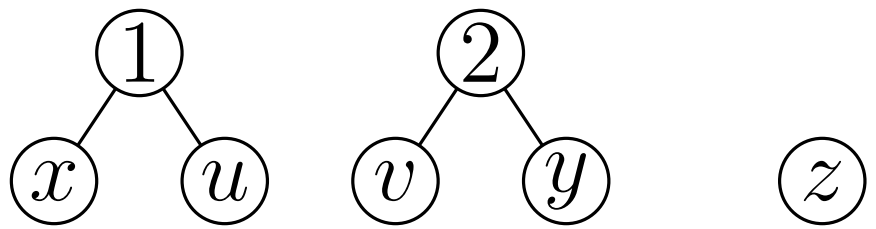


H

Hierarchical Cluster Tree

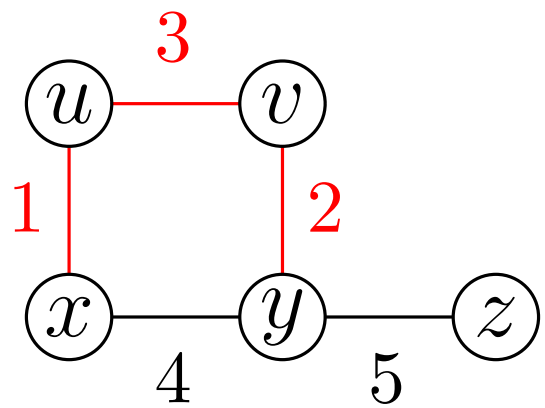


G

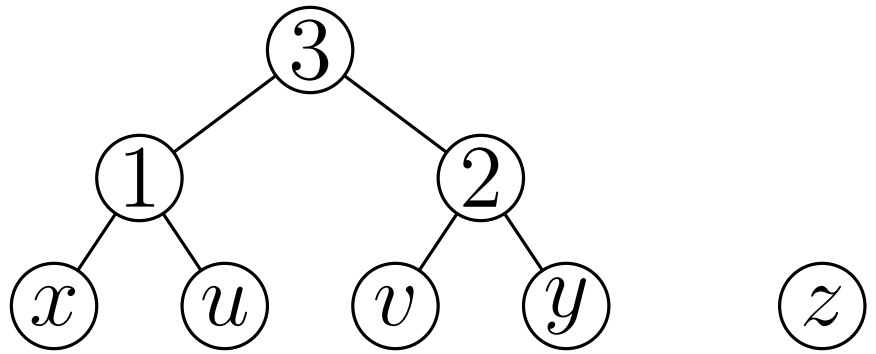


H

Hierarchical Cluster Tree

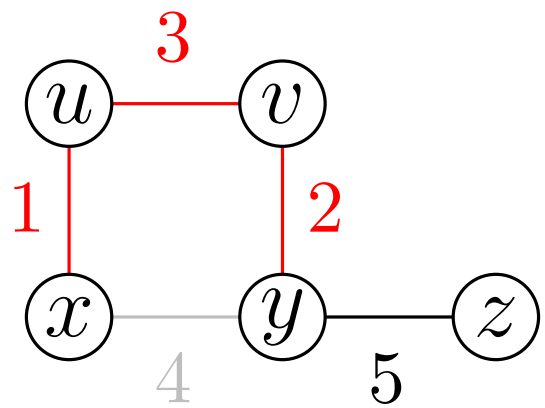


G

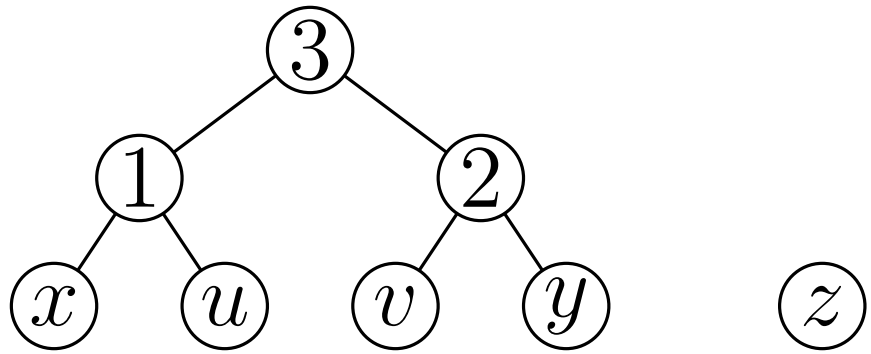


H

Hierarchical Cluster Tree

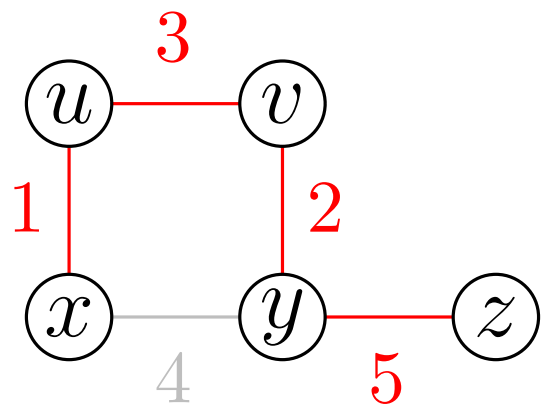


G

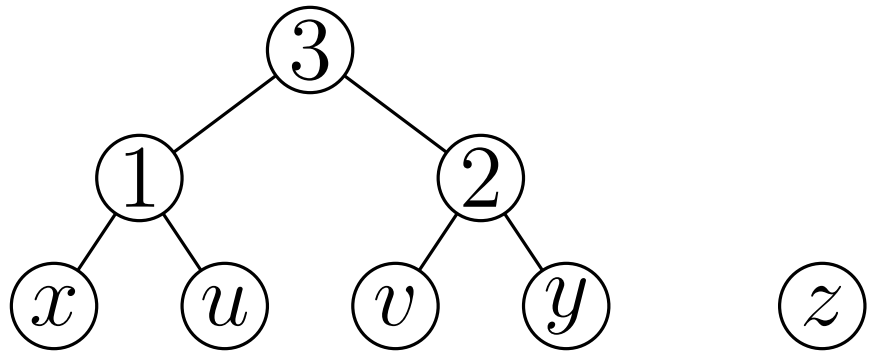


H

Hierarchical Cluster Tree

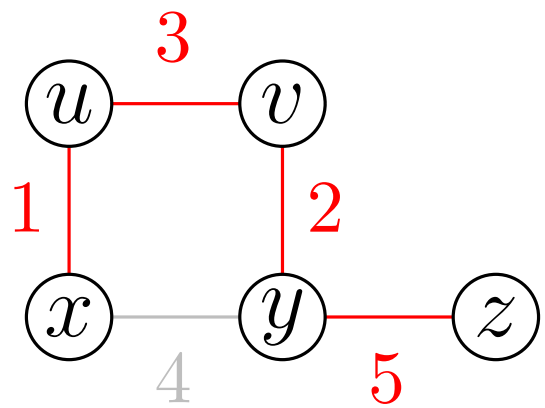


G

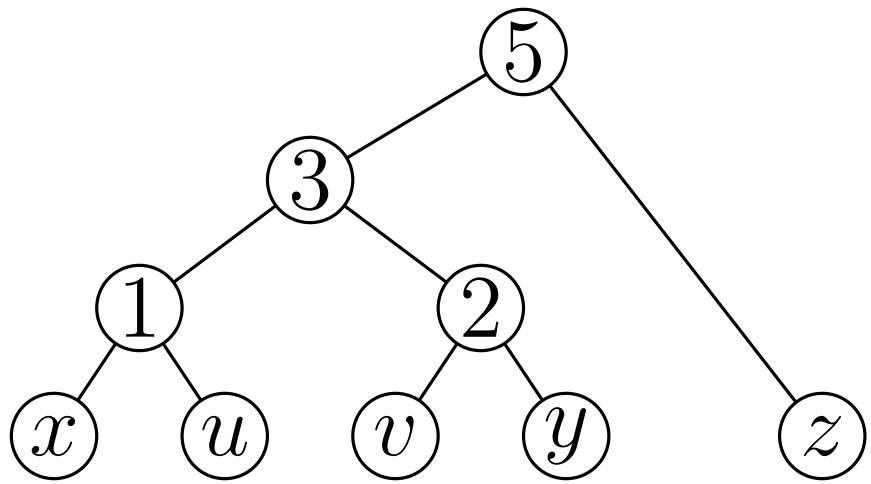


H

Hierarchical Cluster Tree



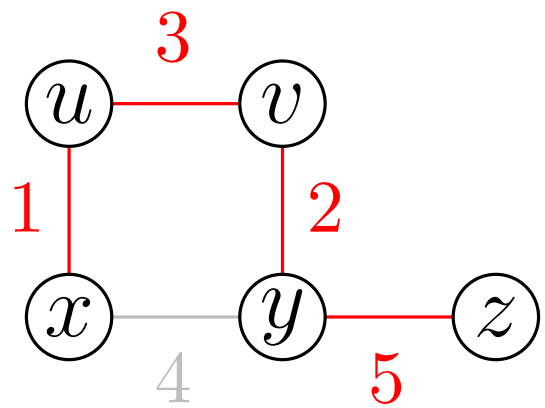
G



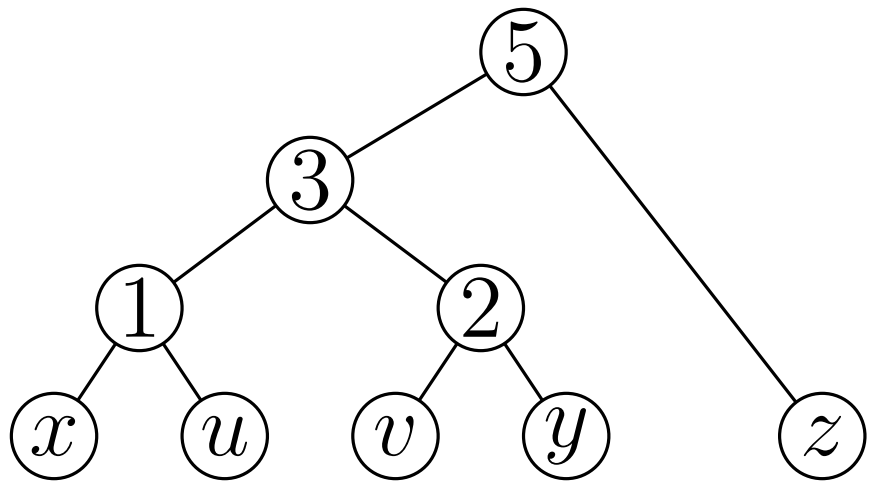
H

Hierarchical Cluster Tree

Property 1: Each of the $n-1$ edges added by Kruskal's algorithm corresponds to an internal node of H.



G

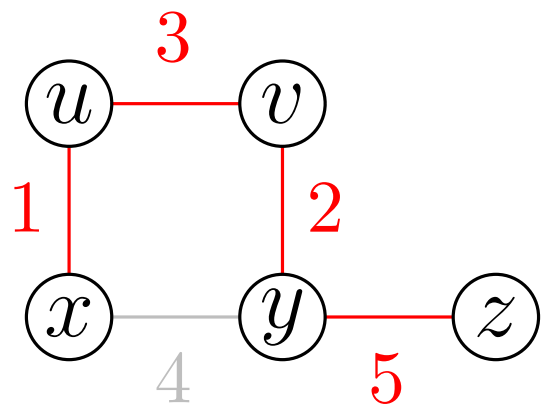


H

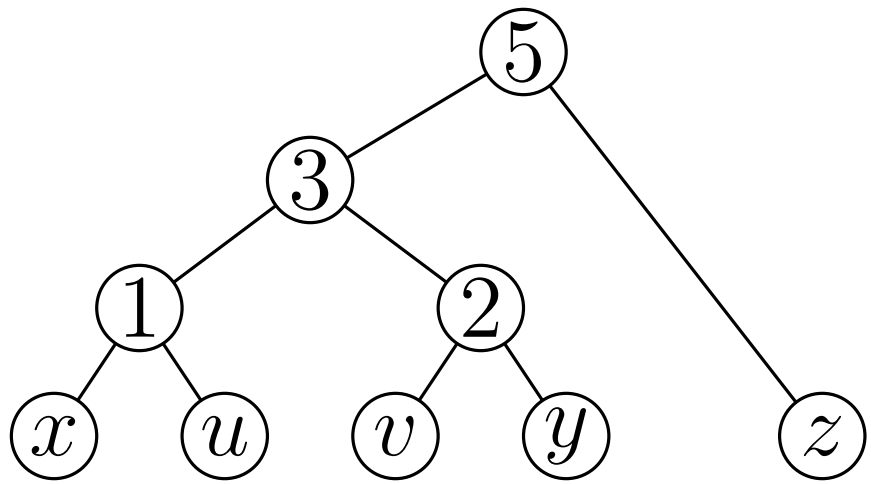
Hierarchical Cluster Tree

Property 1: Each of the $n-1$ edges added by Kruskal's algorithm corresponds to an internal node of H.

Property 2: the (sets of leaves of) subtrees of H correspond exactly to the connected components that appear at some point in the execution of Kruskal's algorithm.



G

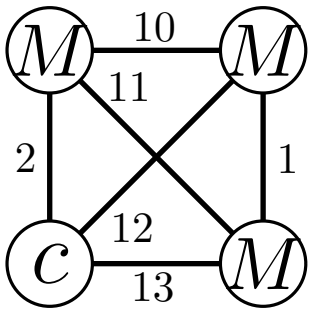


H

Where Does Single-Link Fail?

$k = 3$

$c \ll M$

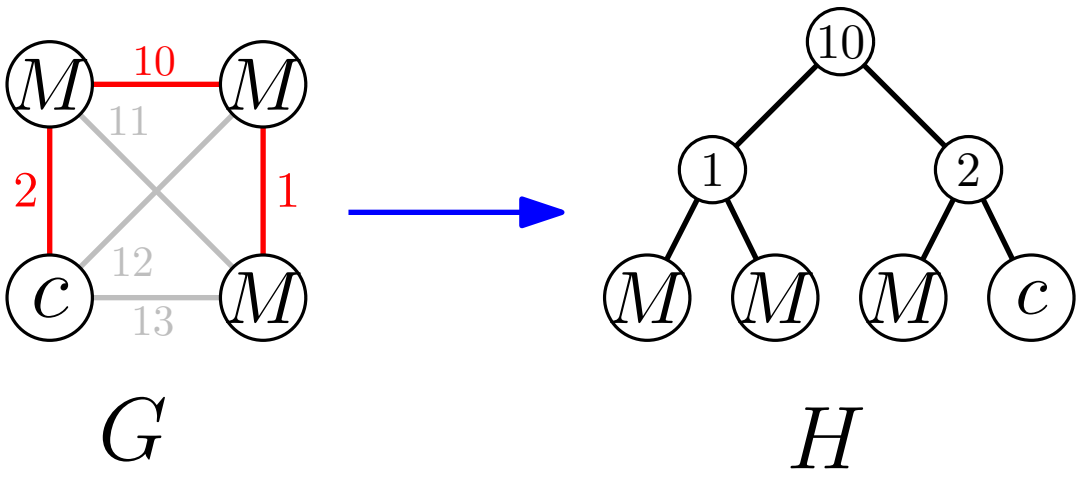


G

Where Does Single-Link Fail?

$k = 3$

$c \ll M$

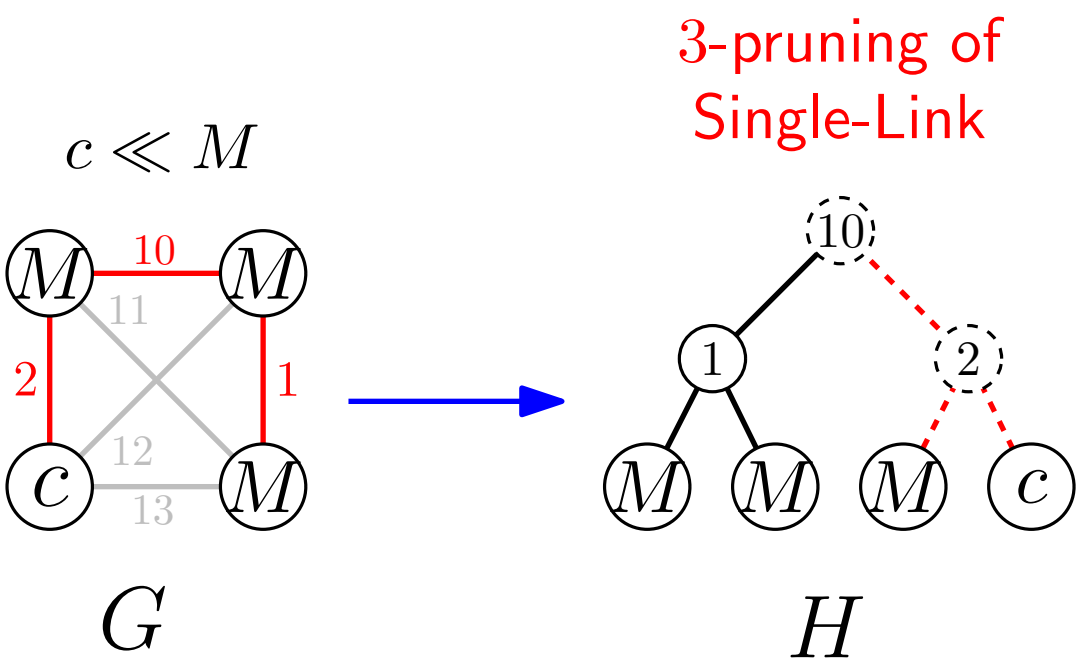


G H

Where Does Single-Link Fail?

$k = 3$

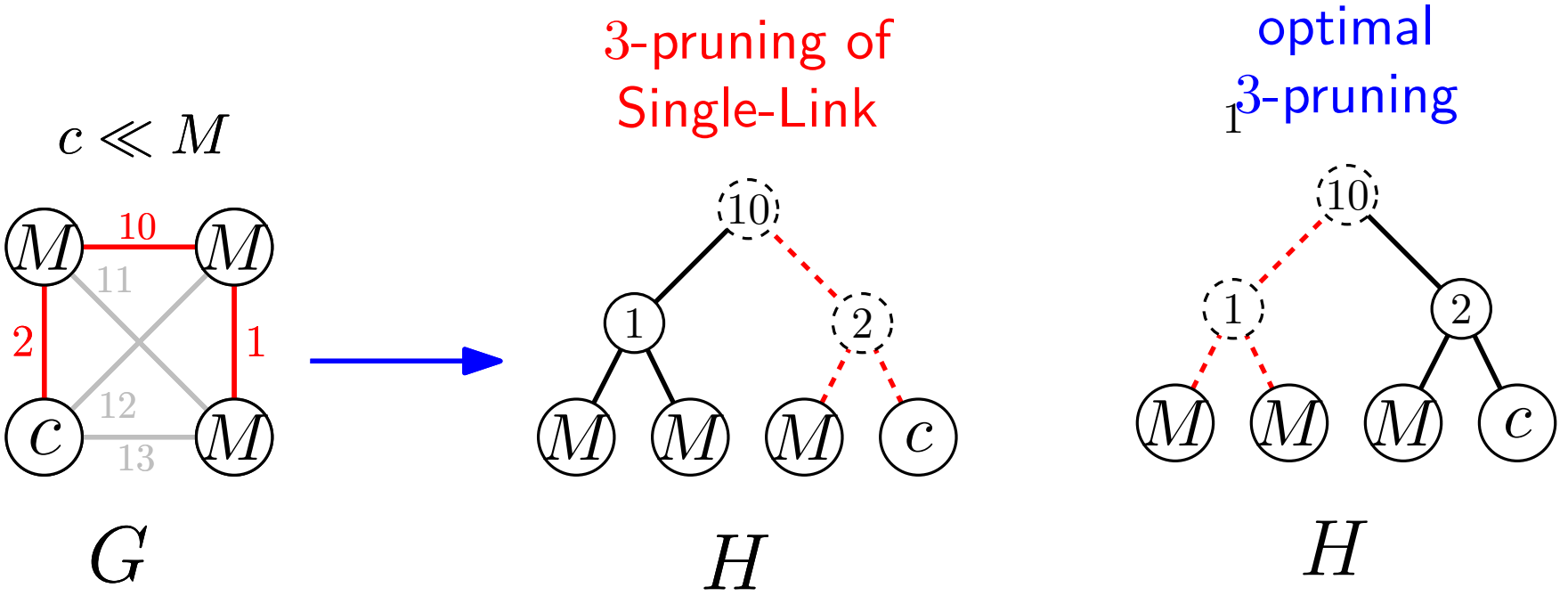
Def. k-pruning of H $(k \geq 1)$ as a k-clustering induced by the removal of a set S of k internal nodes "from the top" of H, meaning that if $x \in S$ then parent$(x) \in S$ as well.



Where Does Single-Link Fail?

$k = 3$

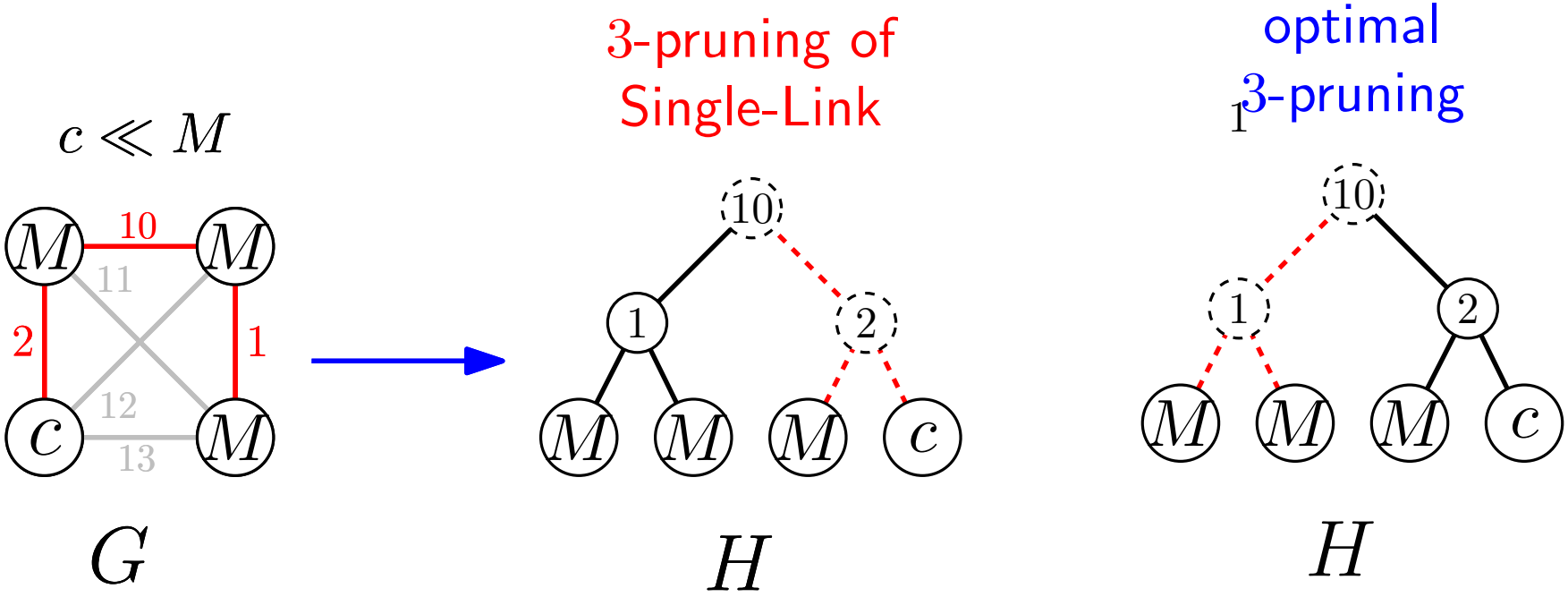
Def. k-pruning of H $(k \geq 1)$ as a k-clustering induced by the removal of a set S of k internal nodes "from the top" of H, meaning that if $x \in S$ then parent$(x) \in S$ as well.



Where Does Single-Link Fail?

$k = 3$

Def. k-pruning of H ($k \geq 1$) as a k-clustering induced by the removal of a set S of k internal nodes "from the top" of H, meaning that if $x \in S$ then parent$(x) \in S$ as well.

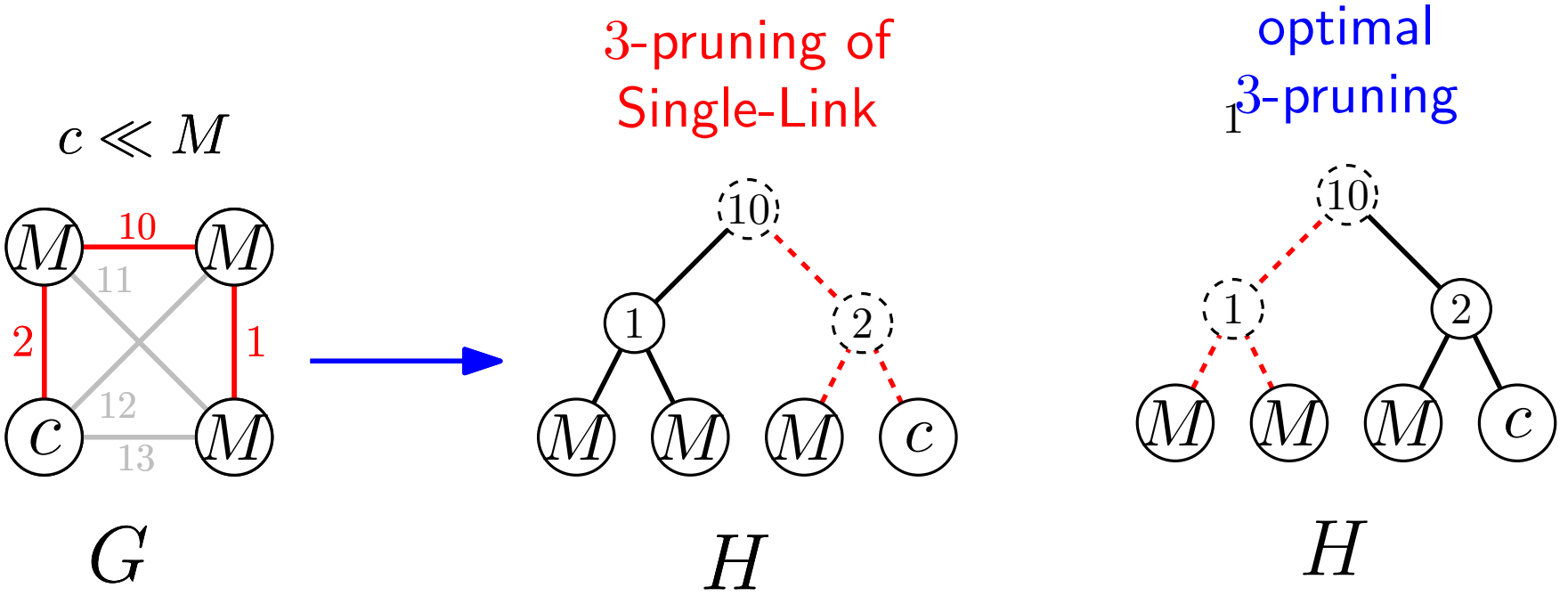


Good news: it can be done in polynmial time vial **dynamic programming**!

Where Does Single-Link Fail?

$k = 3$

Def. k-pruning of H ($k \geq 1$) as a k-clustering induced by the removal of a set S of k internal nodes "from the top" of H, meaning that if $x \in S$ then parent$(x) \in S$ as well.



Good news: it can be done in polynmial time vial **dynamic programming**! (exercise)

Single-Link++ (SL++)

Single-Link++ Algorithm:

Single-Link++ (SL++)

Single-Link++ Algorithm:

1. compute the hierarchical cluster tree H of G

Single-Link++ (SL++)

Single-Link++ Algorithm:

1. compute the hierarchical cluster tree H of G
2. compute the optimal k-pruning of H via DP algorithm

Single-Link++ (SL++)

Single-Link++ Algorithm:

1. compute the hierarchical cluster tree H of G
2. compute the optimal k-pruning of H via DP algorithm
3. return a cluster for each connected component of H

Single-Link++ (SL++)

Single-Link++ Algorithm:

1. compute the hierarchical cluster tree H of G
2. compute the optimal k-pruning of H via DP algorithm
3. return a cluster for each connected component of H

Theorem. For every γ-stable k-median instance with $\gamma \geq 3$, the SL++ algorithm recovers the optimal solution.

Single-Link++ (SL++)

Single-Link++ Algorithm:

1. compute the hierarchical cluster tree H of G
2. compute the optimal k-pruning of H via DP algorithm
3. return a cluster for each connected component of H

Theorem. For every γ-stable k-median instance with $\gamma \geq 3$, the SL++ algorithm recovers the optimal solution.

Lemma. for every $\varepsilon > 0$, there is a $(3 - \varepsilon)$-stable k-median instance such that the SL++ algorithm does not recover the optimal solution.

Single-Link++ (SL++)

Single-Link++ Algorithm:

1. compute the hierarchical cluster tree H of G
2. compute the optimal k-pruning of H via DP algorithm
3. return a cluster for each connected component of H

Theorem. For every γ-stable k-median instance with $\gamma \geq 3$, the SL++ algorithm recovers the optimal solution.

Lemma. for every $\varepsilon > 0$, there is a $(3 - \varepsilon)$-stable k-median instance such that the SL++ algorithm does not recover the optimal solution. (the theorem analysis is thight)

Single-Link++ (SL++)

Single-Link++ Algorithm:

1. compute the hierarchical cluster tree H of G
2. compute the optimal k-pruning of H via DP algorithm
3. return a cluster for each connected component of H

Theorem. For every γ-stable k-median instance with $\gamma \geq 3$, the SL++ algorithm recovers the optimal solution.

Lemma. for every $\varepsilon > 0$, there is a $(3 - \varepsilon)$-stable k-median instance such that the SL++ algorithm does not recover the optimal solution. (the theorem analysis is thight)

Fact. Fa variation of the k-pruning procedure can achieves $\gamma = 1 + \sqrt{2}$.

Single-Link++ (SL++)

Single-Link++ Algorithm:

1. compute the hierarchical cluster tree H of G
2. compute the optimal k-pruning of H via DP algorithm
3. return a cluster for each connected component of H

Theorem. For every γ-stable k-median instance with $\gamma \geq 3$, the SL++ algorithm recovers the optimal solution.

Lemma. for every $\varepsilon > 0$, there is a $(3 - \varepsilon)$-stable k-median instance such that the SL++ algorithm does not recover the optimal solution. (the theorem analysis is thight)

Fact. Fa variation of the k-pruning procedure can achieves $\gamma = 1 + \sqrt{2}$.

Theorem. for $\varepsilon > 0$, no polynomial-time algorithm always recovers the optimal solution in $(2 - \varepsilon)$-stable k-median instances, unless $P = NP$.

Single-Link++ (SL++)

Single-Link++ Algorithm:

1. compute the hierarchical cluster tree H of G
2. compute the optimal k-pruning of H via DP algorithm
3. return a cluster for each connected component of H

Theorem. For every γ-stable k-median instance with $\gamma \geq 3$, the SL++ algorithm recovers the optimal solution.

Lemma. for every $\varepsilon > 0$, there is a $(3 - \varepsilon)$-stable k-median instance such that the SL++ algorithm does not recover the optimal solution. (the theorem analysis is thight)

Fact. Fa variation of the k-pruning procedure can achieves $\gamma = 1 + \sqrt{2}$.

Theorem. for $\varepsilon > 0$, no polynomial-time algorithm always recovers the optimal solution in $(2 - \varepsilon)$-stable k-median instances, unless $P = NP$.

Analysis

When Does Single-Link++ Succeed?

Let $C_1^*, \ldots, C_k^*$ be the optimal clustering of the k-median istance (X, d).

When Does Single-Link++ Succeed?

Let $C_1^*, \ldots, C_k^*$ be the optimal clustering of the k-median istance (X, d).

(a) for every i, the cluster C_i^* appears as a connected component at some stage of Kruskal's algorithm.

When Does Single-Link++ Succeed?

Let $C_1^*, \ldots, C_k^*$ be the optimal clustering of the k-median istance (X, d).

(a) for every i, the cluster C_i^* appears as a connected component at some stage of Kruskal's algorithm.

(a) $\implies$ SL++ computes the optimal solution

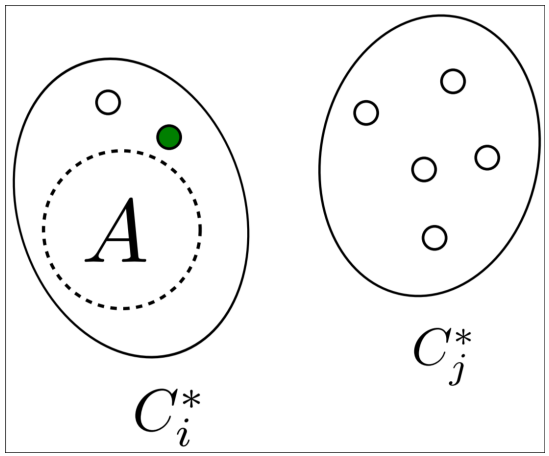
When Does Single-Link++ Succeed?

Let $C_1^*, \ldots, C_k^*$ be the optimal clustering of the k-median istance (X, d).

(a) for every i, the cluster C_i^* appears as a connected component at some stage of Kruskal's algorithm.

(a) $\implies$ SL++ computes the optimal solution

(b) for every i, and every strict subset $A \subset C_i^*$, the point $x^* = \arg\max_{x \in X \setminus A} d(x, A)$ of $X \setminus A$ closest to A lies in C_i^*



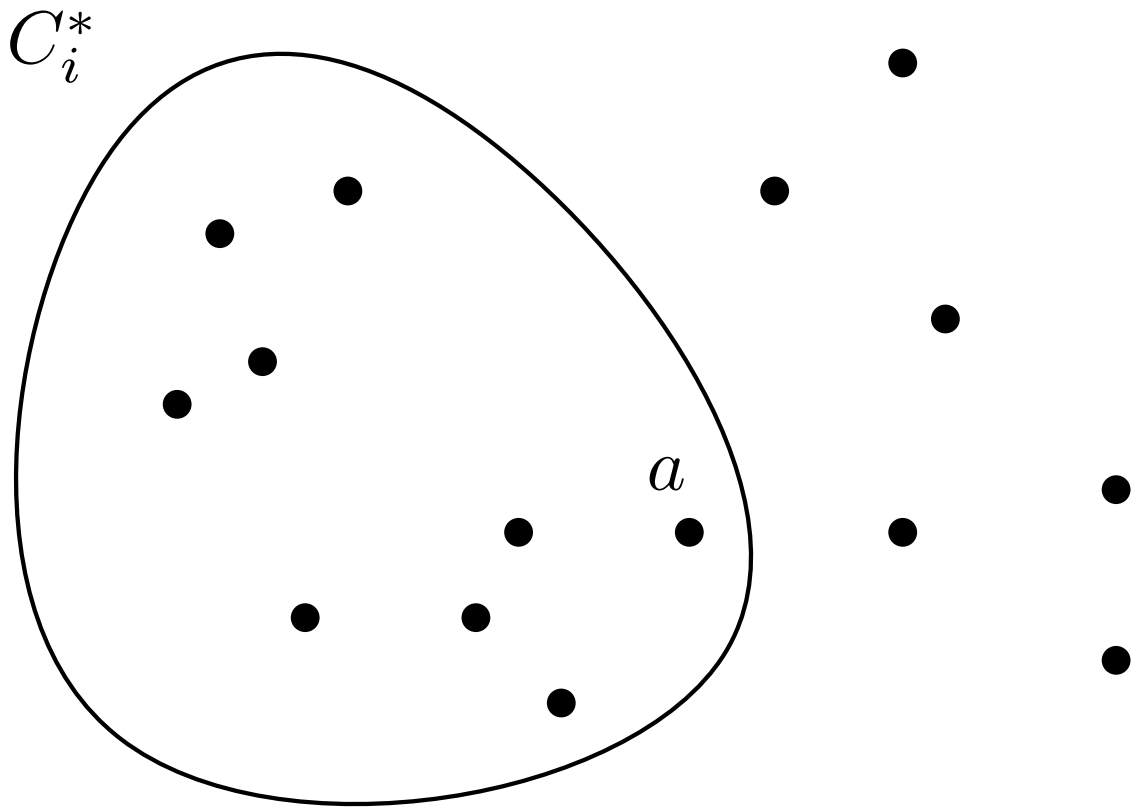
(b) $\implies$ *(a)* $\implies$ SL++ computes the optimal solution

$(b) \implies (a)$

Assume (b) is true.

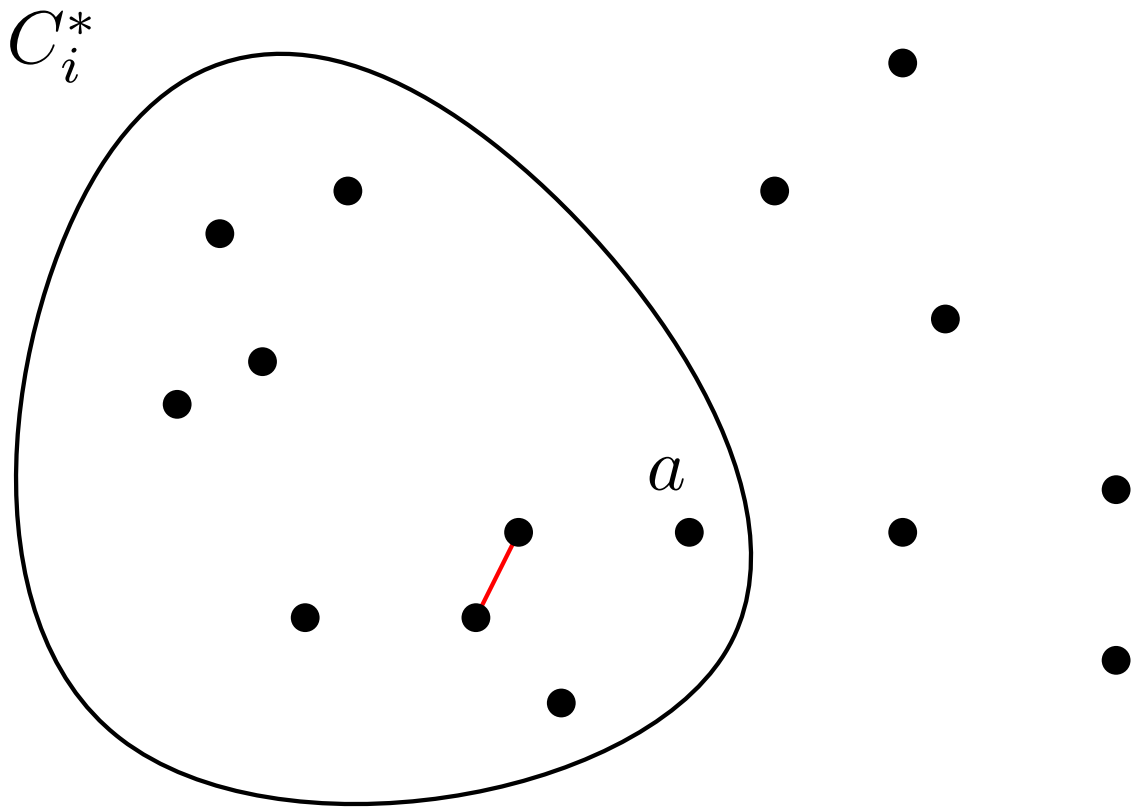
(b) $\Longrightarrow$ *(a)*

Assume *(b)* is true.



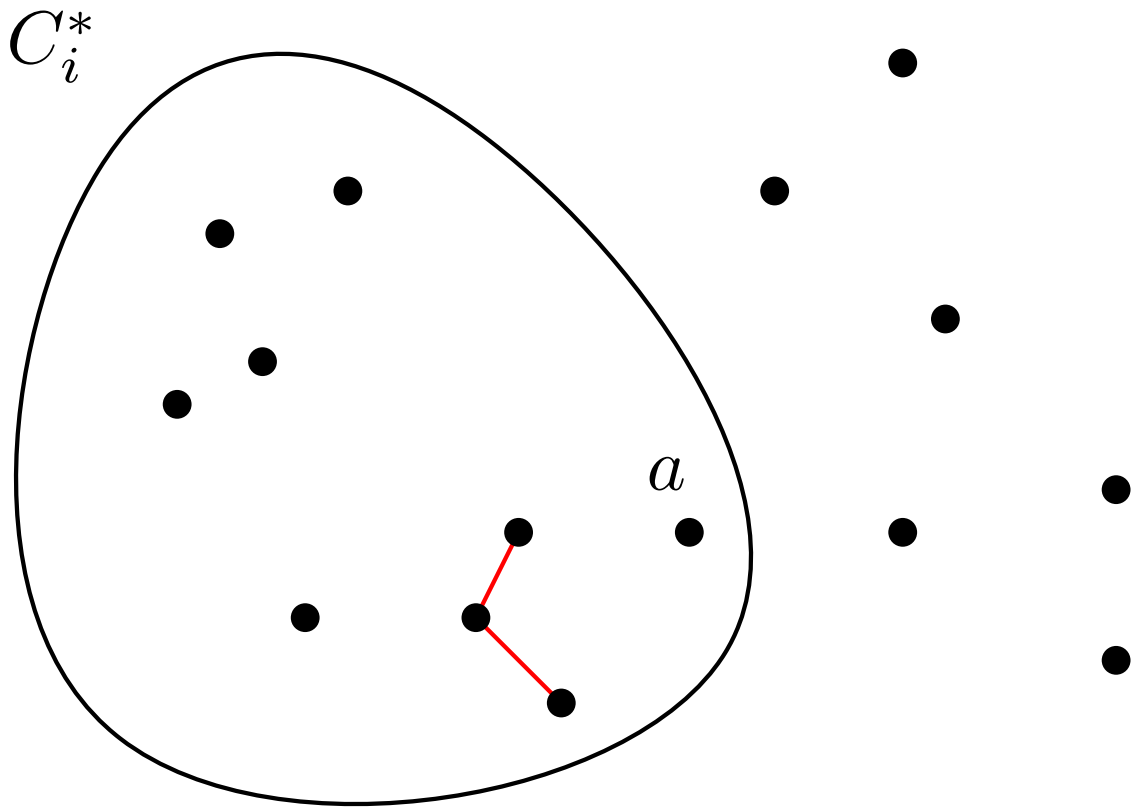
(b) $\Longrightarrow$ *(a)*

Assume *(b)* is true.



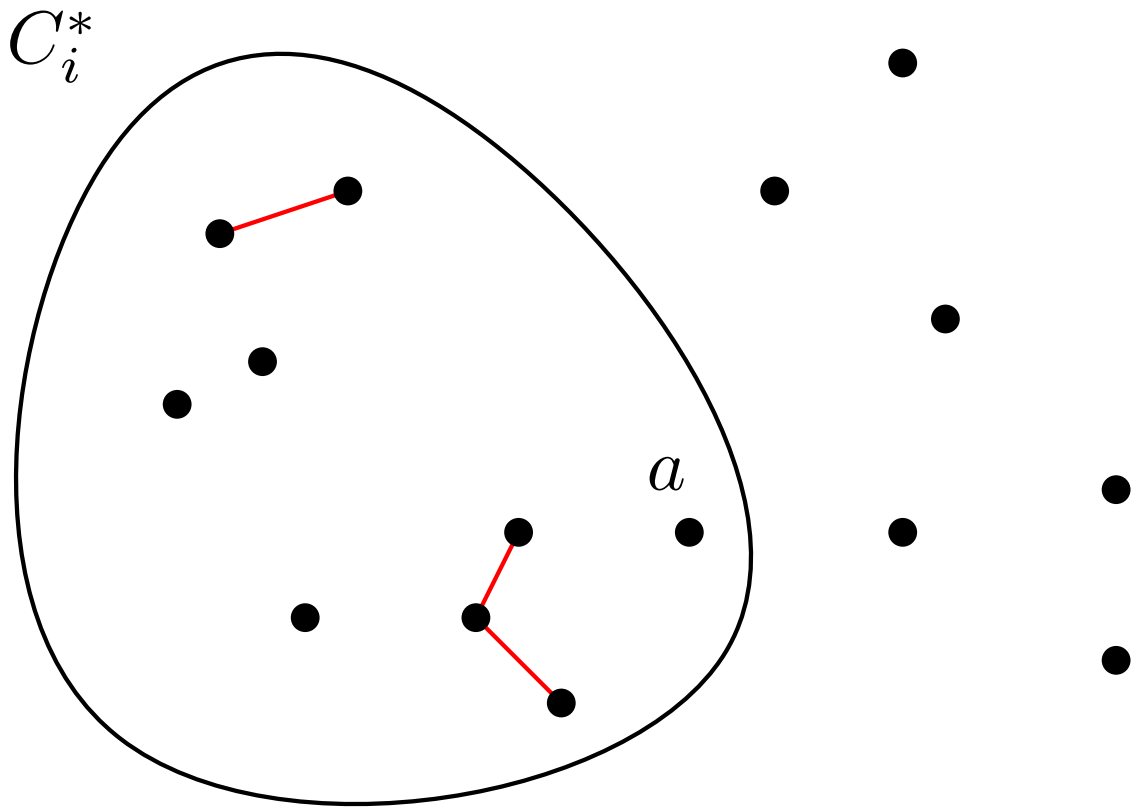
$(b) \implies (a)$

Assume (b) is true.



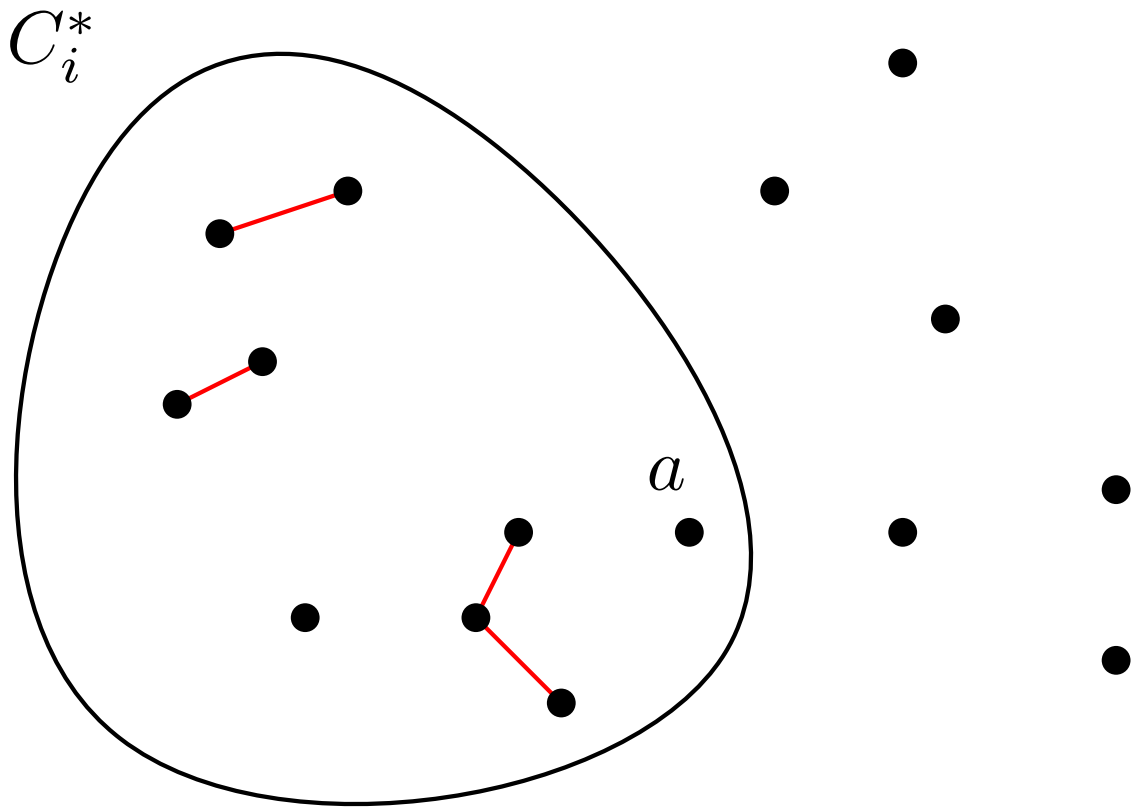
(b) $\Longrightarrow$ *(a)*

Assume *(b)* is true.



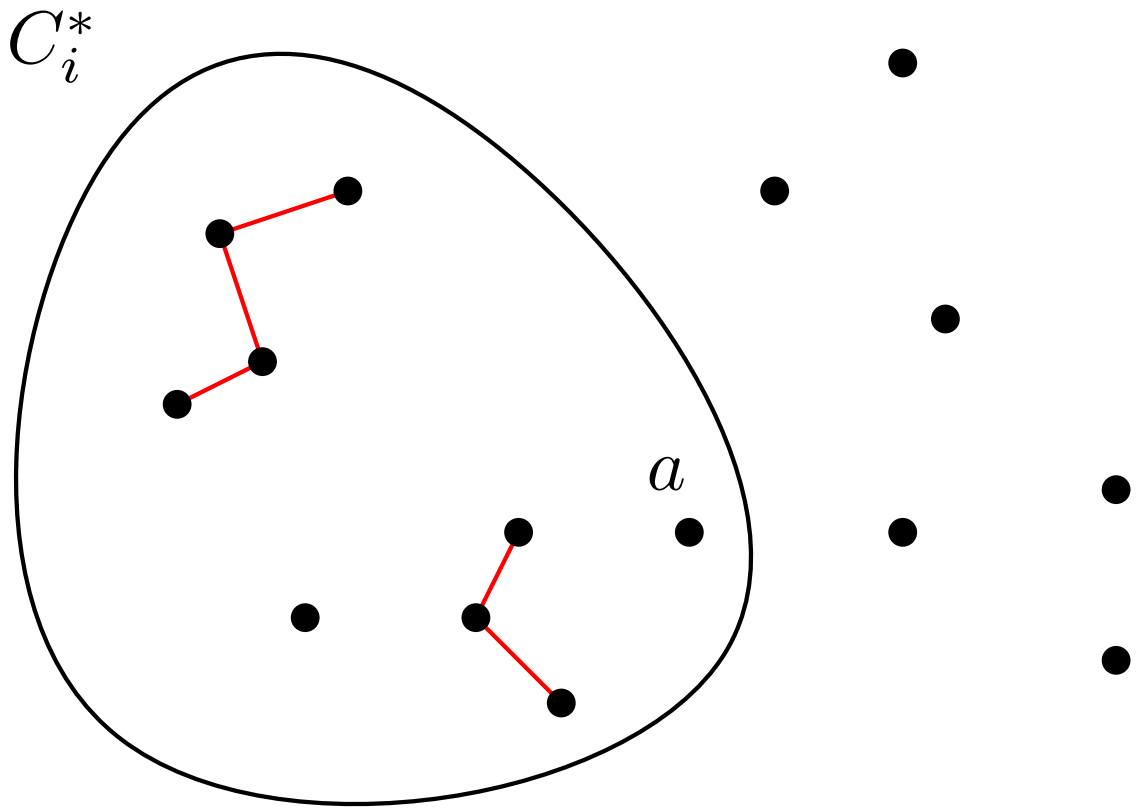
$(b) \implies (a)$

Assume (b) is true.



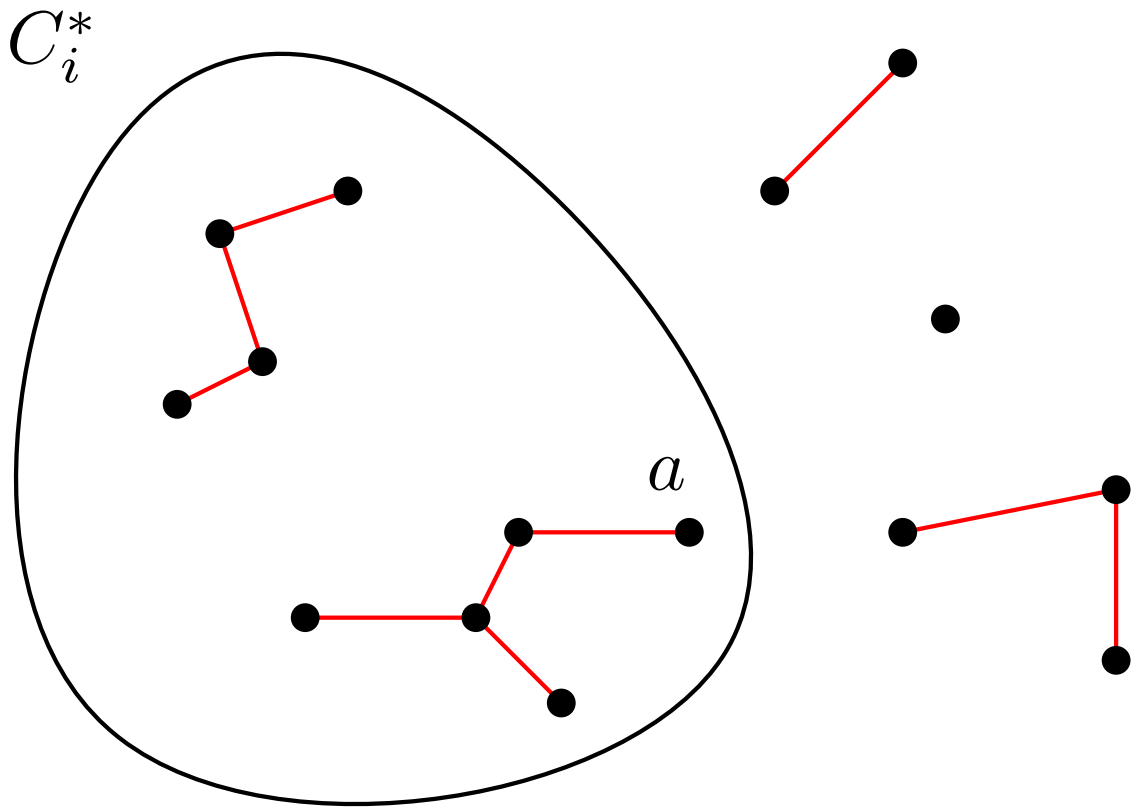
(b) $\Longrightarrow$ *(a)*

Assume *(b)* is true.



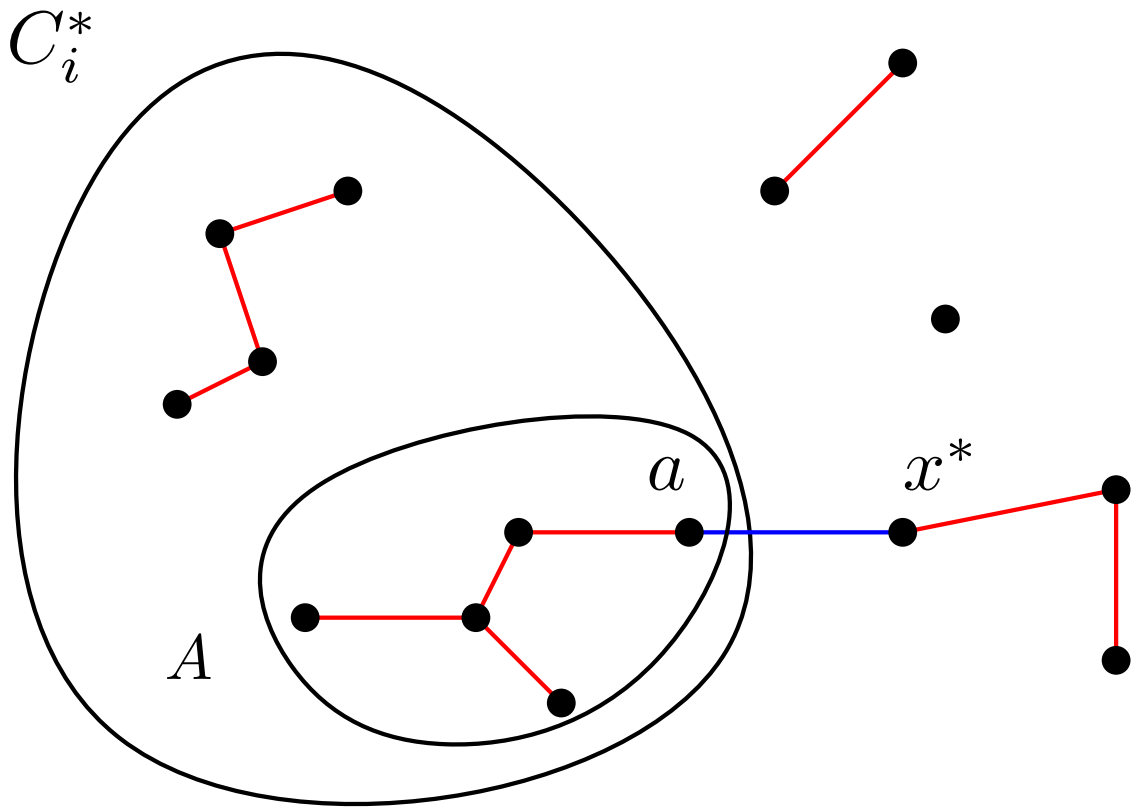
$(b) \implies (a)$

Assume (b) is true.



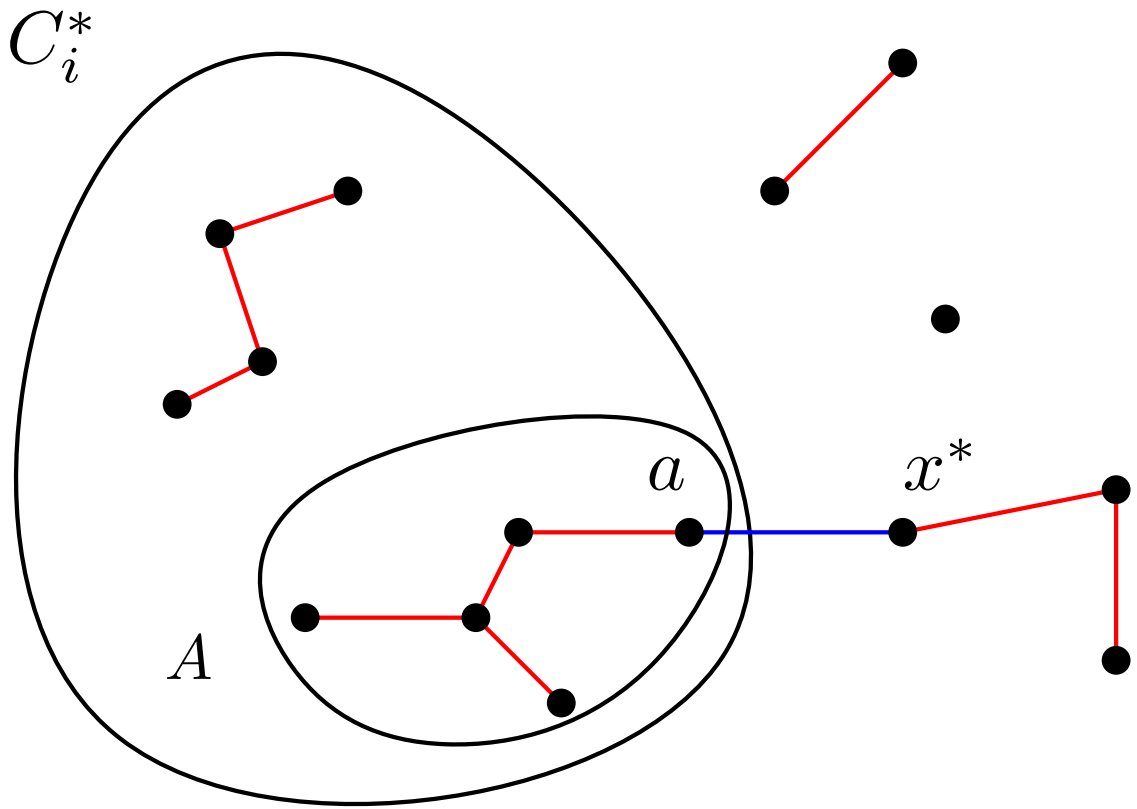
$(b) \implies (a)$

Assume (b) is true.



$(b) \implies (a)$

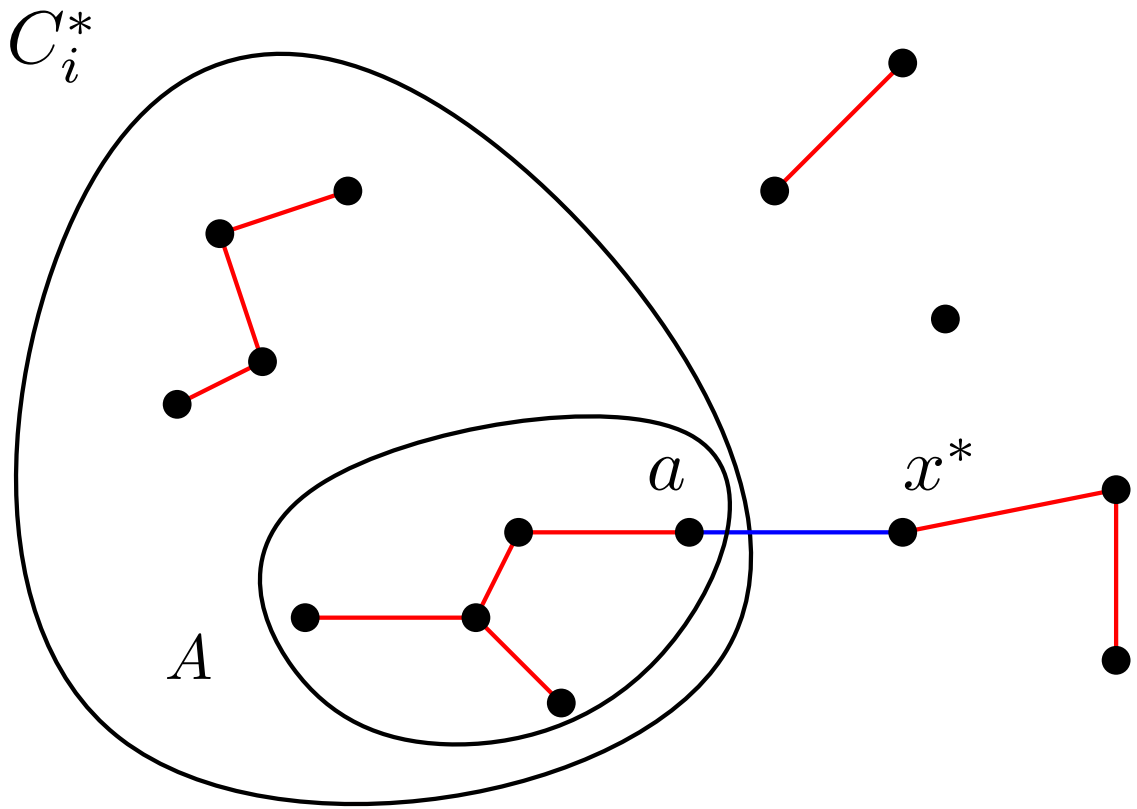
Assume (b) is true.



By definition $x^* = \arg\max_{x \in X \setminus A} d(A, x)$

(b) $\Longrightarrow$ *(a)*

Assume *(b)* is true.

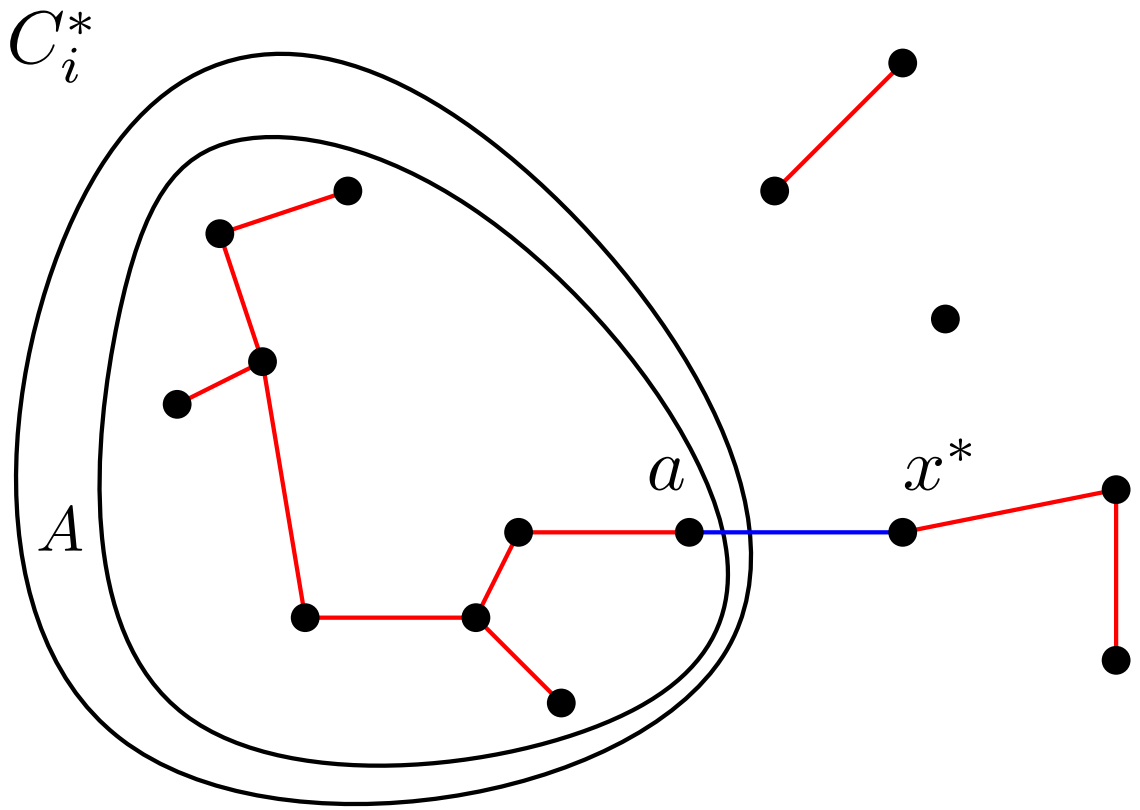


By definition $x^* = \arg\max_{x \in X \setminus A} d(A, x)$

By assumption *(b)*, if $A \subset C_i^*$ then x^* must be in C_i^*

(b) $\Longrightarrow$ *(a)*

Assume *(b)* is true.



By definition $x^* = \arg\max_{x \in X \setminus A} d(A, x)$

By assumption *(b)*, if $A \subset C_i^*$ then x^* must be in C_i^*

$\Longrightarrow$ $A \not\subset C_i^*$ $\Longrightarrow$ $A = C_i^*$ $\Longrightarrow$ C_i^* appears as a connected component during Kruskal.

Proof of *(b)*

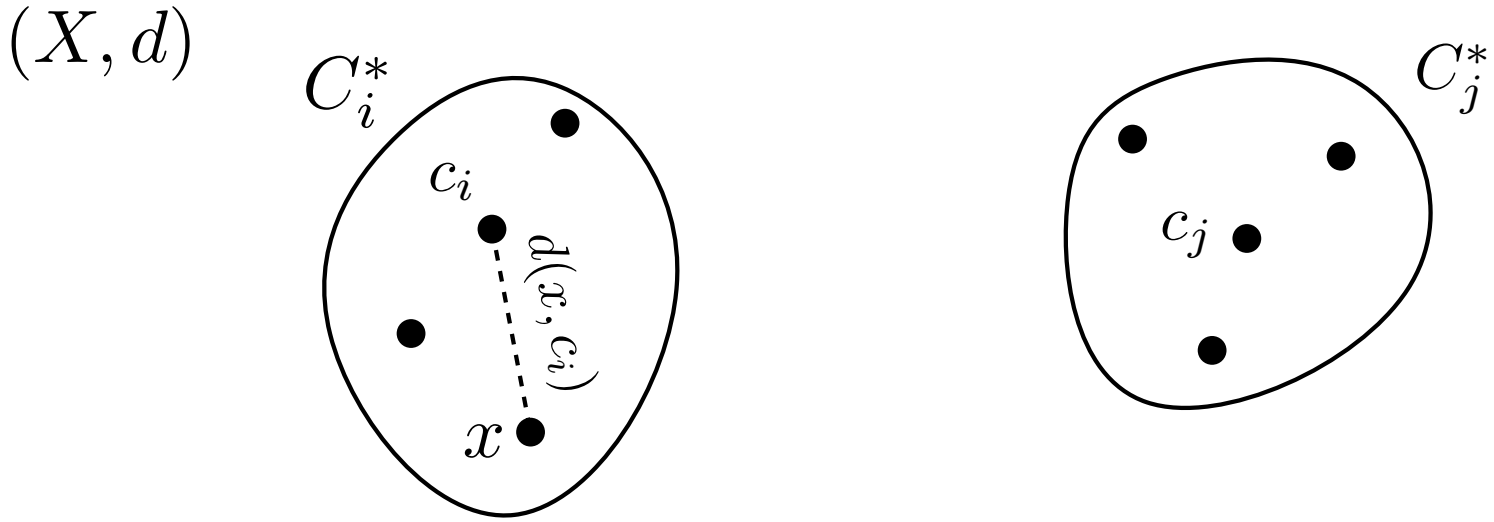
Claim 1. In every γ-stable instance with $\gamma \geq 3$, for every point $x \in C_i^*$ we have

$$d(x, c_j) > 3d(x, c_i), \quad \forall j \neq i.$$

Proof of *(b)*

Claim 1. In every γ-stable instance with $\gamma \geq 3$, for every point $x \in C_i^*$ we have

$$d(x, c_j) > 3d(x, c_i), \quad \forall j \neq i.$$

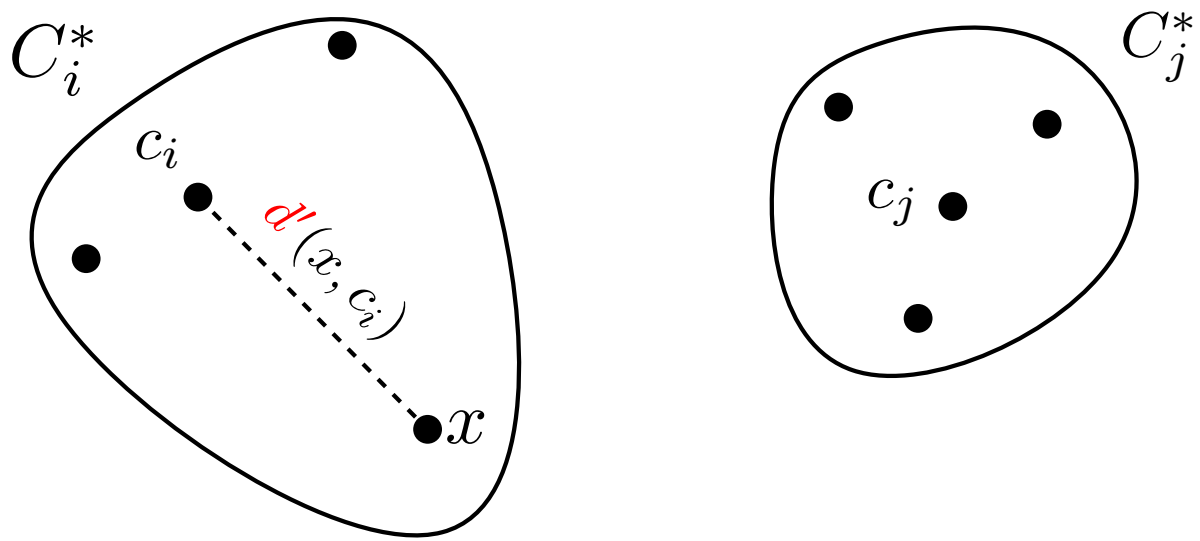


Proof of *(b)*

Claim 1. In every γ-stable instance with $\gamma \geq 3$, for every point $x \in C_i^*$ we have

$$d(x, c_j) > 3d(x, c_i), \quad \forall j \neq i.$$

(X, d')



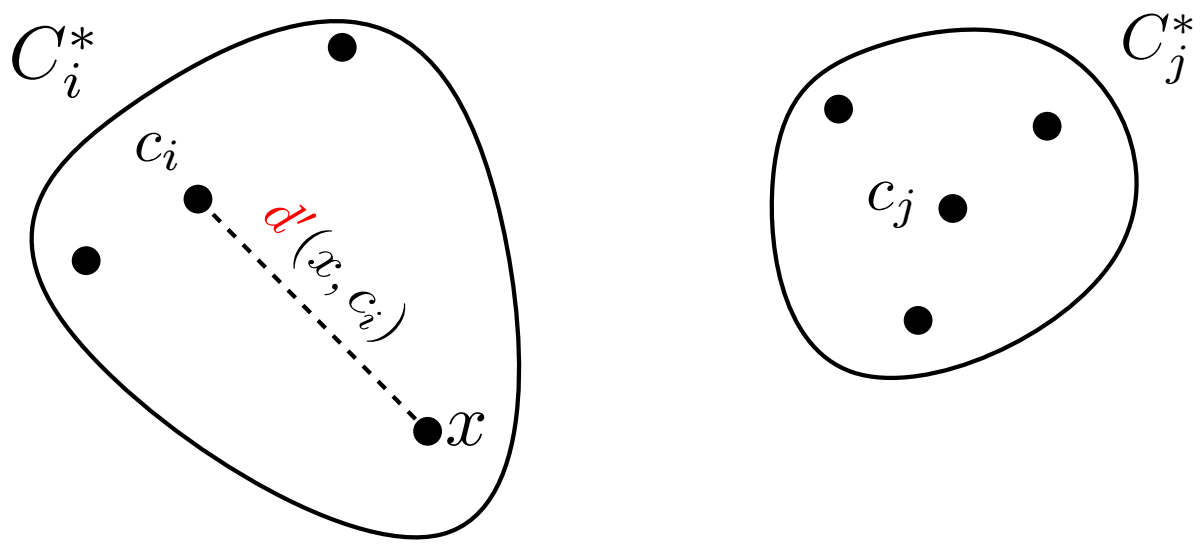
Blow up C_i^* by a factor γ, leaving all the other distances the same.

Proof of *(b)*

Claim 1. In every γ-stable instance with $\gamma \geq 3$, for every point $x \in C_i^*$ we have

$$d(x, c_j) > 3d(x, c_i), \quad \forall j \neq i.$$

(X, d')



Blow up C_i^* by a factor γ, leaving all the other distances the same.

$$d(x, c_j) = d'(x, c_j) > d'(x, c_i) = \gamma d(x, c_i) \geq 3d(x, c_i)$$

■

Proof of *(b)*

Claim 2. For every $x \in C_i^*$ and $y \in C_j^*$, with $j \neq i$, we have

$$d(x, y) > 2 \max\{d(x, c_i), d(y, c_j)\}.$$

Proof of *(b)*

Claim 2. For every $x \in C_i^*$ and $y \in C_j^*$, with $j \neq i$, we have

$$d(x, y) > 2 \max\{d(x, c_i), d(y, c_j)\}.$$

$$d(x, y) + d(x, c_i) \geq d(x, c_j) > 3d(x, c_i)$$

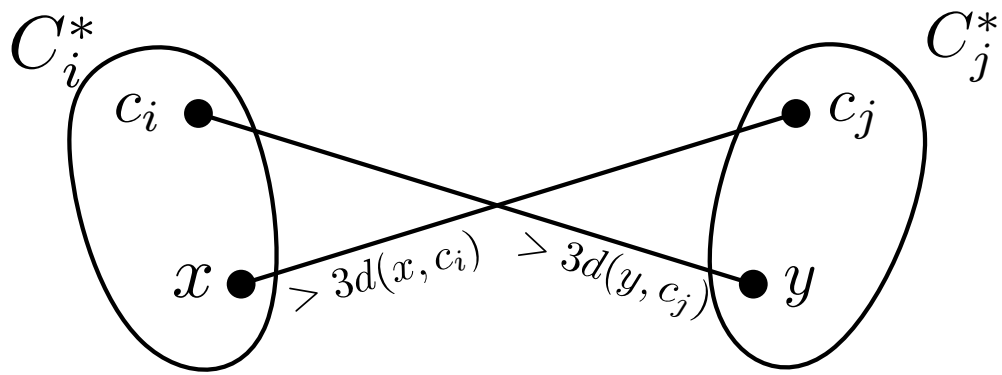
Proof of *(b)*

Claim 2. For every $x \in C_i^*$ and $y \in C_j^*$, with $j \neq i$, we have

$$d(x,y) > 2\max\{d(x,c_i), d(y,c_j)\}.$$

triangular inequality

$$d(x,y) + d(x,c_i) \geq d(x,c_j) > 3d(x,c_i)$$



Proof of *(b)*

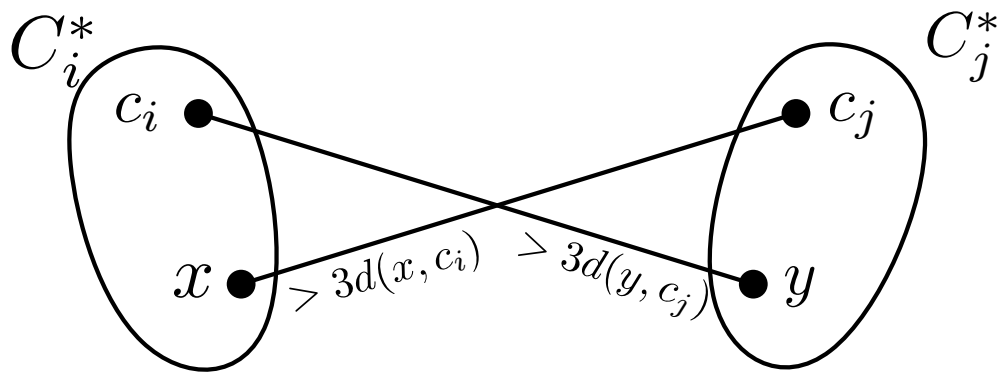
Claim 2. For every $x \in C_i^*$ and $y \in C_j^*$, with $j \neq i$, we have

$$d(x,y) > 2\max\{d(x,c_i), d(y,c_j)\}.$$

triangular inequality

claim 1

$$d(x,y) + d(x,c_i) \geq d(x,c_j) > 3d(x,c_i)$$



Proof of *(b)*

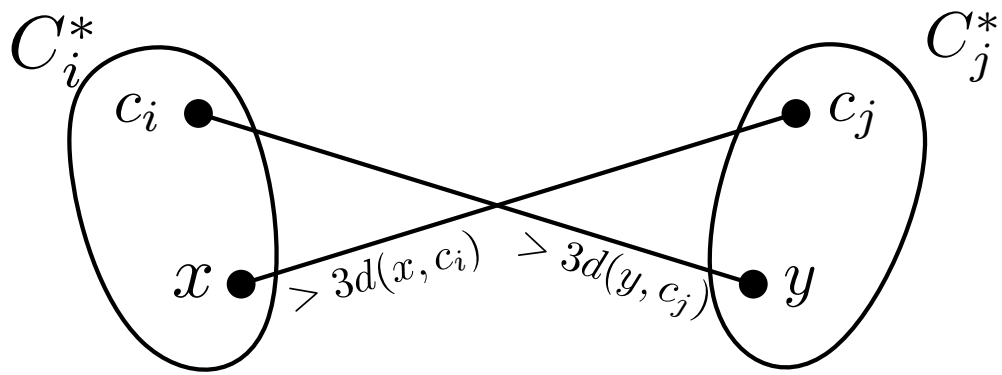
Claim 2. For every $x \in C_i^*$ and $y \in C_j^*$, with $j \neq i$, we have

$$d(x, y) > 2 \max\{d(x, c_i), d(y, c_j)\}.$$

triangular inequality

claim 1

$$d(x, y) + d(x, c_i) \geq d(x, c_j) > 3d(x, c_i)$$



rearranging: $d(x, y) > 2d(x, c_i)$

Proof of *(b)*

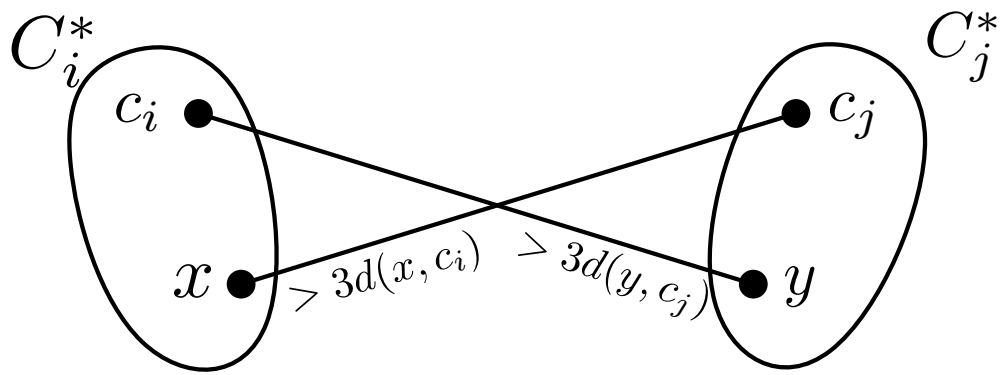
Claim 2. For every $x \in C_i^*$ and $y \in C_j^*$, with $j \neq i$, we have

$$d(x,y) > 2\max\{d(x,c_i), d(y,c_j)\}.$$

triangular inequality

claim 1

$$d(x,y) + d(x,c_i) \geq d(x,c_j) > 3d(x,c_i)$$



rearranging: $d(x,y) > 2d(x,c_i)$

by symmetry: $d(x,y) > 2d(y,c_j)$

Proof of *(b)*

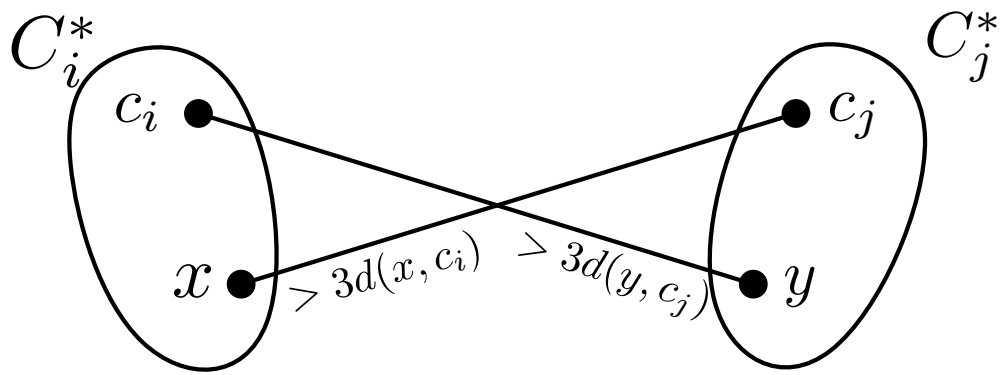
Claim 2. For every $x \in C_i^*$ and $y \in C_j^*$, with $j \neq i$, we have

$$d(x,y) > 2\max\{d(x,c_i), d(y,c_j)\}.$$

triangular inequality

claim 1

$$d(x,y) + d(x,c_i) \geq d(x,c_j) > 3d(x,c_i)$$



rearranging: $d(x,y) > 2d(x,c_i)$

by symmetry: $d(x,y) > 2d(y,c_j)$

$$\Longrightarrow \; d(x,y) > 2\max\{d(x,c_i), d(y,c_j)\}$$

■

Proof of *(b)*

We need to prove that, for every C_i^* and every strict subset $A \subset C_i^*$, the closest point in $X \setminus A$ to A lies in C_i^*.

Proof of *(b)*

We need to prove that, for every C_i^* and every strict subset $A \subset C_i^*$, the closest point in $X \setminus A$ to A lies in C_i^*.

Trivial case: $c_i \notin A$.

Proof of *(b)*

We need to prove that, for every C_i^* and every strict subset $A \subset C_i^*$, the closest point in $X \setminus A$ to A lies in C_i^*.

Trivial case: $c_i \notin A$. (why?)

Proof of *(b)*

We need to prove that, for every C_i^* and every strict subset $A \subset C_i^*$, the closest point in $X \setminus A$ to A lies in C_i^*.

Trivial case: $c_i \notin A$. (why?)

Assume $c_i \in A$.

Proof of *(b)*

We need to prove that, for every C_i^* and every strict subset $A \subset C_i^*$, the closest point in $X \setminus A$ to A lies in C_i^*.

Trivial case: $c_i \notin A$. (why?)

Assume $c_i \in A$.

By contraddiction: assume the closest point to A is $y \in C_j^*$, with $j \neq i$.

Proof of *(b)*

We need to prove that, for every C_i^* and every strict subset $A \subset C_i^*$, the closest point in $X \setminus A$ to A lies in C_i^*.

Trivial case: $c_i \notin A$. (why?)

Assume $c_i \in A$.

By contraddiction: assume the closest point to A is $y \in C_j^*$, with $j \neq i$.

Since $A \subset C_i^*$, there exists $z \in C_i^* \setminus A$.

Proof of *(b)*

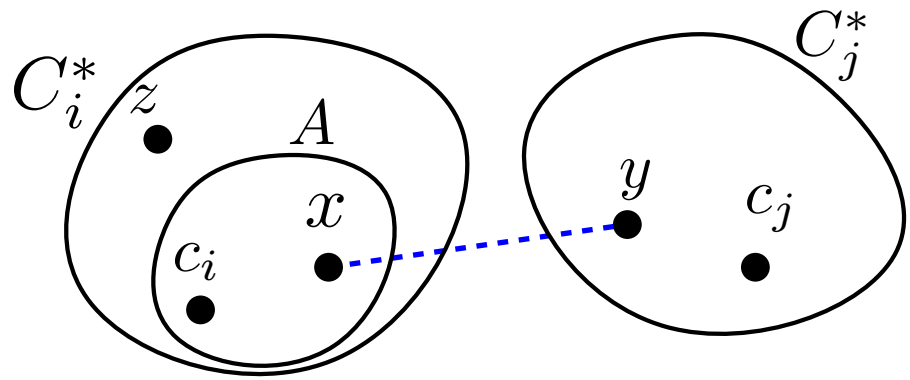
We need to prove that, for every C_i^* and every strict subset $A \subset C_i^*$, the closest point in $X \setminus A$ to A lies in C_i^*.

Trivial case: $c_i \notin A$. (why?)

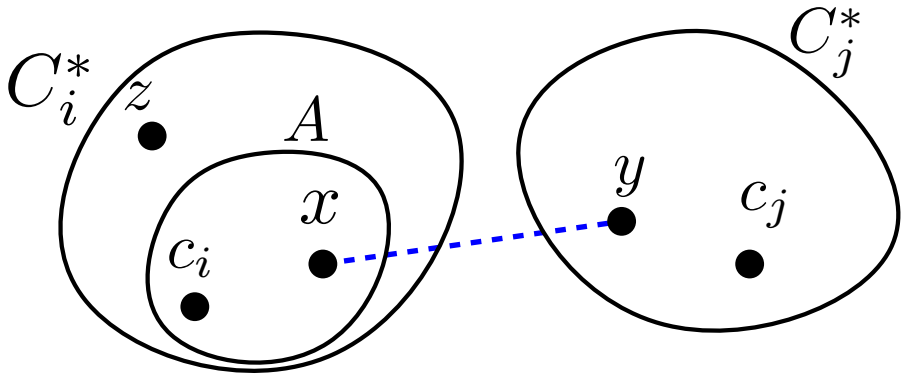
Assume $c_i \in A$.

By contraddiction: assume the closest point to A is $y \in C_j^*$, with $j \neq i$.

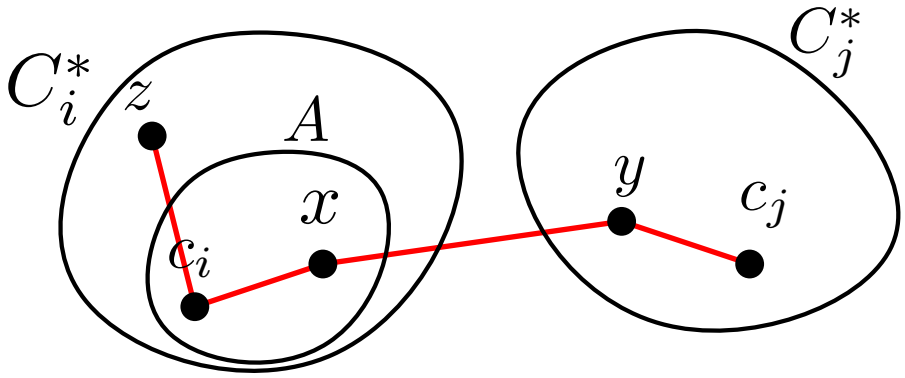
Since $A \subset C_i^*$, there exists $z \in C_i^* \setminus A$.



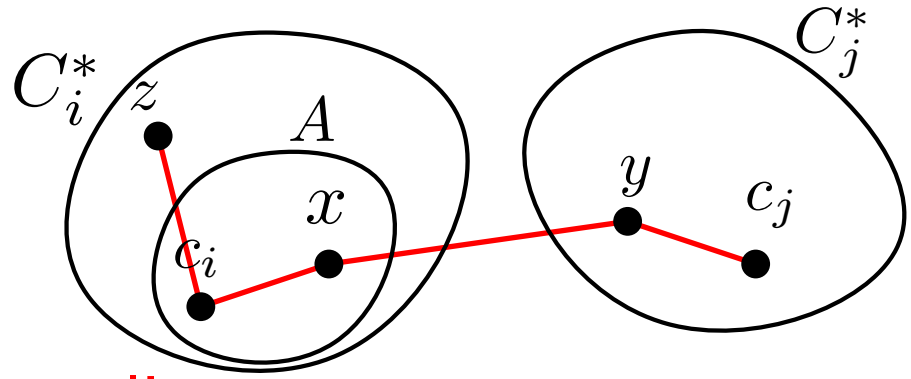
Proof of *(b)*



Proof of *(b)*



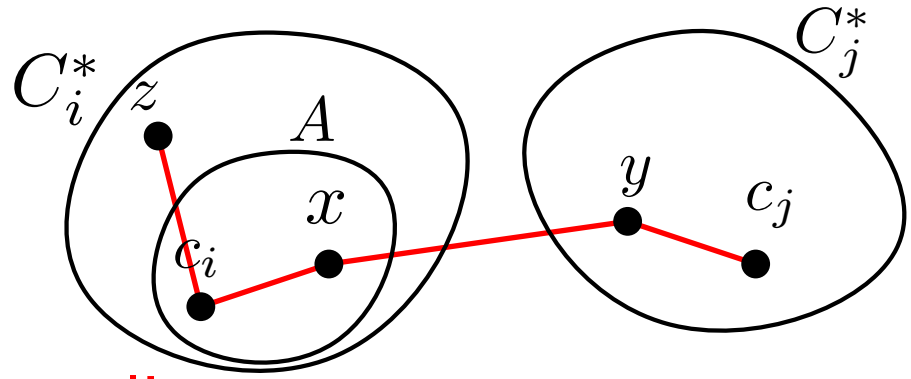
Proof of *(b)*



triangular inequality

$$d(z, c_j) \leq d(z, c_i) + d(c_i, x) + d(x, y) + d(y, c_j)$$

Proof of *(b)*



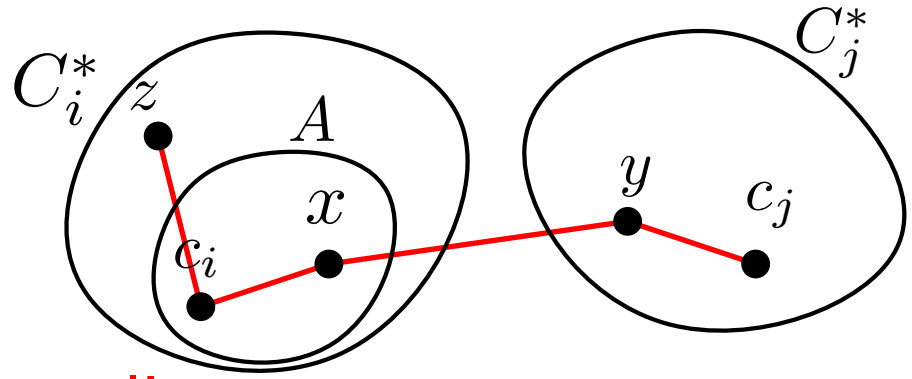
triangular inequality

$$d(z, c_j) \leq d(z, c_i) + d(c_i, x) + d(x, y) + d(y, c_j)$$

$$< \tfrac{1}{2} d(x, y) \qquad < \tfrac{1}{2} d(x, y)$$

calim 2

Proof of *(b)*



triangular inequality

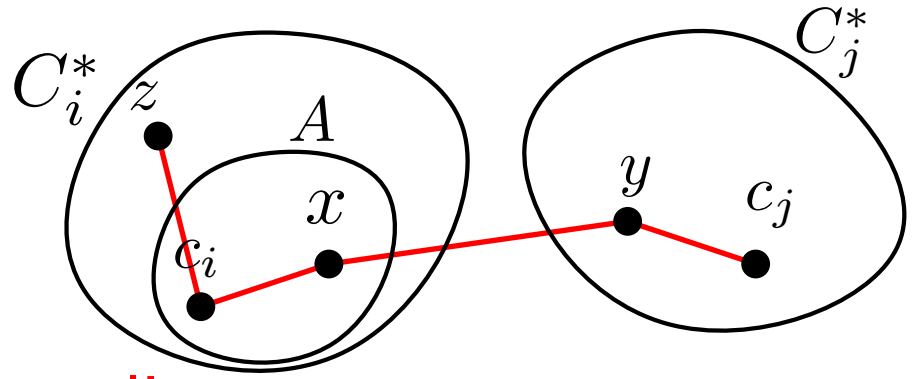
$$d(z, c_j) \leq d(z, c_i) + d(c_i, x) + d(x, y) + d(y, c_j)$$

$$< \tfrac{1}{2} d(x, y) \qquad < \tfrac{1}{2} d(x, y)$$

calim 2

$$< d(z, c_i) + 2d(x, y)$$

Proof of *(b)*



triangular inequality

$$d(z, c_j) \leq d(z, c_i) + d(c_i, x) + d(x, y) + d(y, c_j)$$

$$< \tfrac{1}{2} d(x, y) \qquad < \tfrac{1}{2} d(x, y)$$

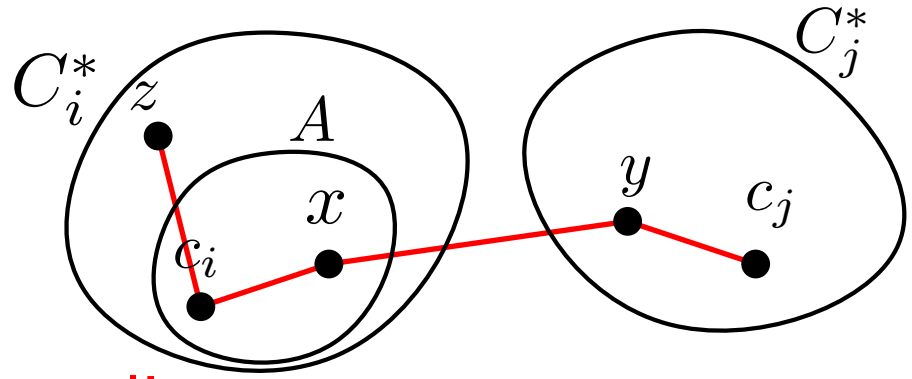
calim 2

$$< d(z, c_i) + 2d(x, y)$$

Since $c_i \in A$, $z \notin A$, and x, y are the closest point between A and C_j, we have

$d(x, y) < d(z, c_i)$

Proof of *(b)*



triangular inequality

$$d(z, c_j) \leq d(z, c_i) + d(c_i, x) + d(x, y) + d(y, c_j)$$

$$< \tfrac{1}{2} d(x, y) \qquad < \tfrac{1}{2} d(x, y)$$

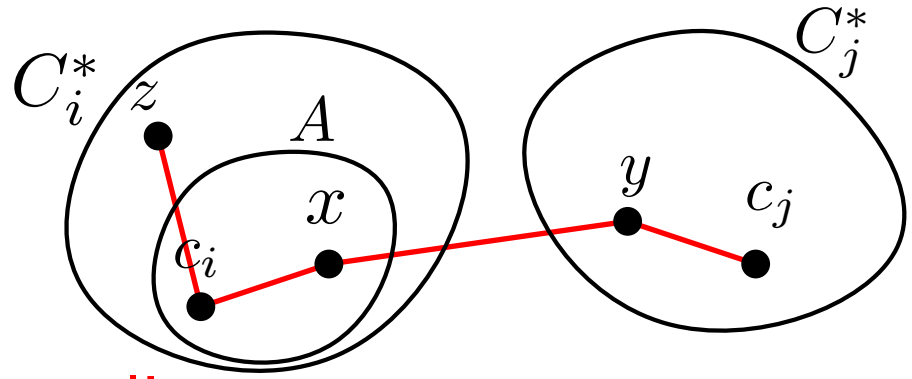
calim 2

$$< d(z, c_i) + 2d(x, y)$$

Since $c_i \in A$, $z \notin A$, and x, y are the closest point between A and C_j, we have

$$d(x, y) < d(z, c_i) \implies d(z, c_j) < 3d(z, c_i) = d'(z, c_i)$$

Proof of *(b)*



triangular inequality

$$d(z,c_j) \le d(z,c_i) + d(c_i,x) + d(x,y) + d(y,c_j)$$

$$< \tfrac{1}{2}d(x,y) \qquad < \tfrac{1}{2}d(x,y)$$

calim 2

$$< d(z,c_i) + 2d(x,y)$$

Since $c_i \in A$, $z \notin A$, and x, y are the closest point between A and C_j, we have

$$d(x,y) < d(z,c_i) \implies d(z,c_j) < 3d(z,c_i) = d'(z,c_i)$$

$\implies$ (X,d) is not γ-stable (absurd!)

■

Other Stability Difinitions

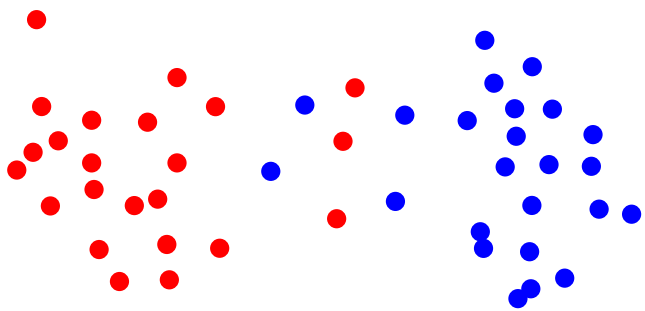
Other Stability Difinitions

Def. (Approximation Stability) The target clustering of a k-median instance is (c, ε)-approximation stable if every c-approximate k-clustering is ε-accurate, meaning that it agrees with the target clustering on at least a $1 - \varepsilon$ fraction of the points.

Other Stability Difinitions

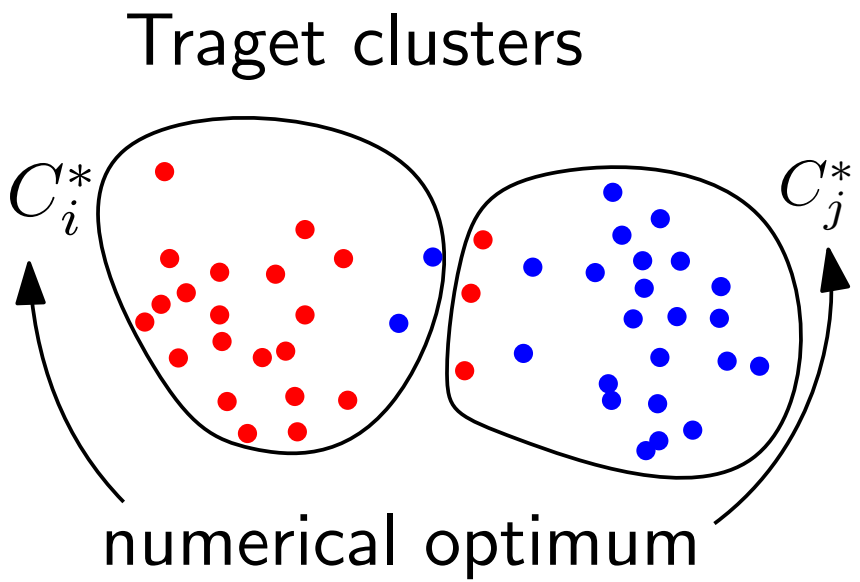
Def. (Approximation Stability) The target clustering of a k-median instance is (c, ε)-approximation stable if every c-approximate k-clustering is ε-accurate, meaning that it agrees with the target clustering on at least a $1 - \varepsilon$ fraction of the points.

Traget clusters



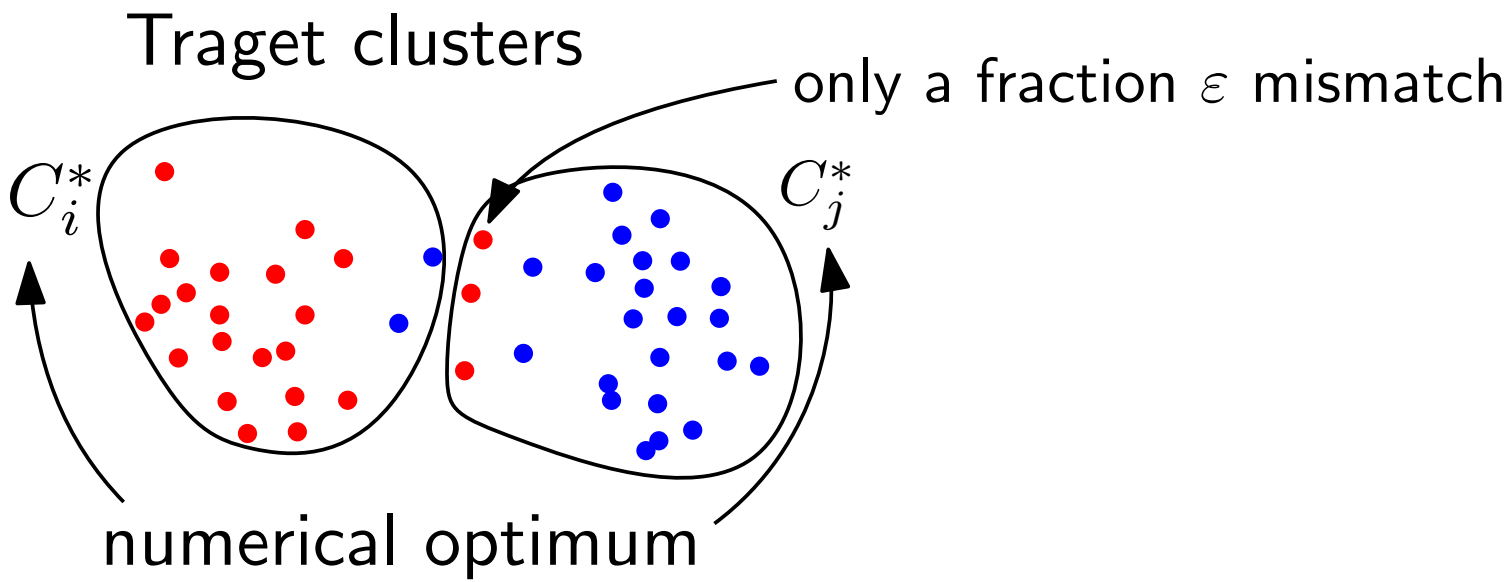
Other Stability Difinitions

Def. (Approximation Stability) The target clustering of a k-median instance is (c, ε)-approximation stable if every c-approximate k-clustering is ε-accurate, meaning that it agrees with the target clustering on at least a $1 - \varepsilon$ fraction of the points.



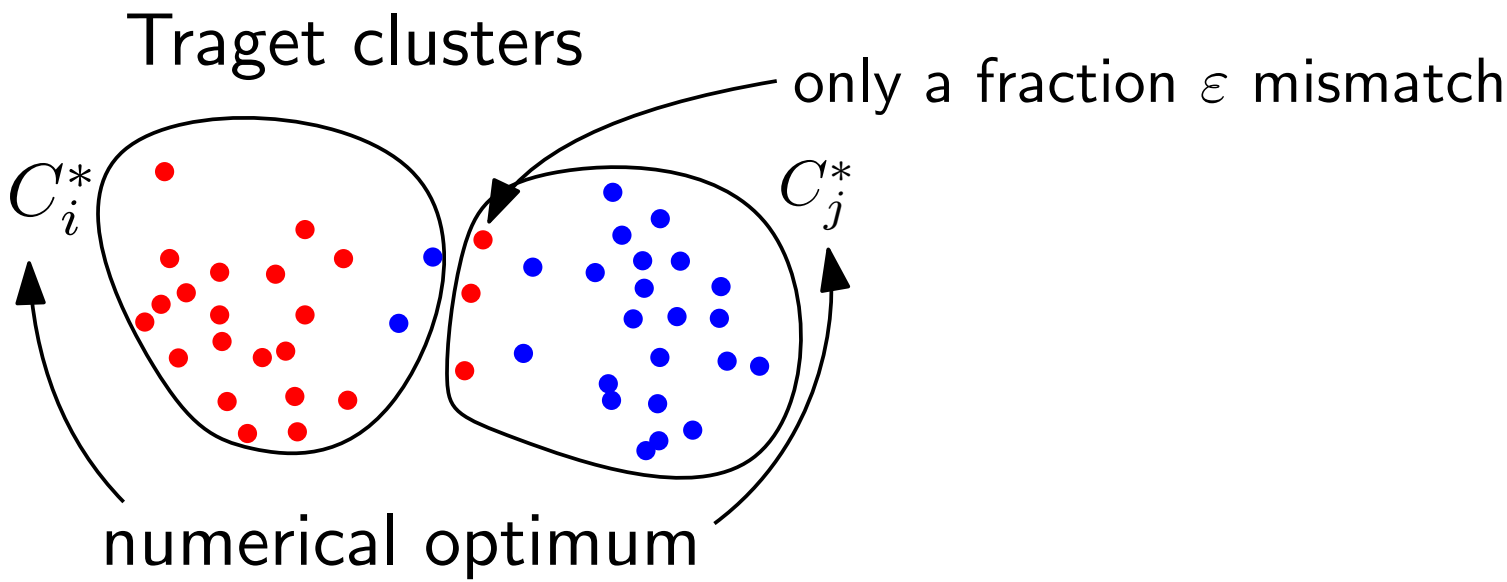
Other Stability Difinitions

Def. (Approximation Stability) The target clustering of a k-median instance is (c, ε)-approximation stable if every c-approximate k-clustering is ε-accurate, meaning that it agrees with the target clustering on at least a $1 - \varepsilon$ fraction of the points.



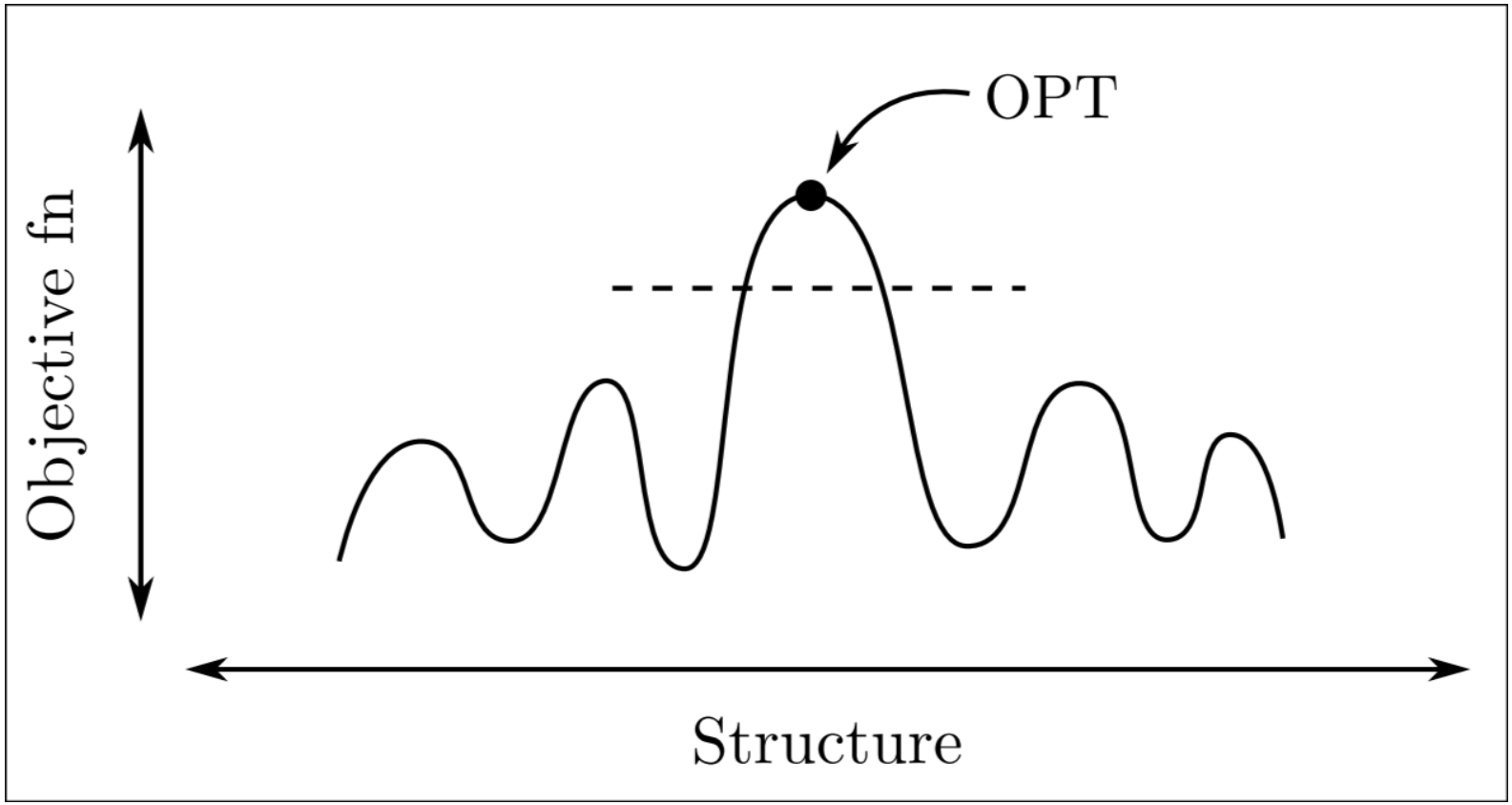
Other Stability Difinitions

Def. (Approximation Stability) The target clustering of a k-median instance is (c, ε)-approximation stable if every c-approximate k-clustering is ε-accurate, meaning that it agrees with the target clustering on at least a $1 - \varepsilon$ fraction of the points.



Theorem. For every pair $\alpha, \varepsilon > 0$ of constants, there is a polynomial-time algorithm that, for every $(1 + \alpha, \varepsilon)$-approximation stable k-median instance, recovers a k-clustering that agrees with the target clustering on all but an $O(\varepsilon/\alpha)$ fraction of the points.

Other Stability Difinitions



Intuition: Numerical near-optimality implies structural near-optimality.