A Constant Approximation for Min-sum k-Clustering

Ruiquan Gao

Stanford University

based on joint work with

Aditya Anand
Max Planck Institute for Informatics

Moses Charikar
Stanford University

Fabrizio Grandoni
IDSIA, USI-SUPSI

Euiwoong Lee
University of Michigan

Amatya Sharma
University of Michigan

Ernest van Wijland
Universite Paris-Cite

Min-sum k-Clustering

- **Problem Formulation**
  - Metric space (D, d)
  - Objective: k clusters $\mathcal{S} = \{S_1, S_2, \dots, S_k\}$ that minimizes

$$\sum_{t \in [k]} \sum_{i,j \in S_t} d(i,j)$$

Min-sum k-Clustering

- **Problem Formulation**
  - Metric space (D, d)
  - Objective: k clusters $\mathcal{S} = \{S_1, S_2, \dots, S_k\}$ that minimizes

$$\sum_{t \in [k]} \sum_{i,j \in S_t} d(i,j)$$

  - Dual version of max k-cut

Min-sum k-Clustering

- **Problem Formulation**
  - Metric space (D, d)
  - Objective: k clusters $\mathcal{S} = \{S_1, S_2, \dots, S_k\}$ that minimizes

$$\sum_{t \in [k]} \sum_{i,j \in S_t} d(i,j)$$

  - Dual version of max k-cut
  - d is not a metric $\Longrightarrow$ no $O(n^{2-\epsilon})$-approximation unless $\mathrm{P} = \mathrm{NP}$

Min-sum k-Clustering

- **Problem Formulation**
  - Metric space (D, d)
  - Objective: k clusters $\mathcal{S} = \{S_1, S_2, \dots, S_k\}$ that minimizes

$$\sum_{t \in [k]} \sum_{i,j \in S_t} d(i,j)$$

  - Dual version of max k-cut
  - d is not a metric $\Longrightarrow$ no $O(n^{2-\epsilon})$-approximation unless $\mathrm{P} = \mathrm{NP}$

- **Kleinberg's consistency axiom** [Kle02]
  - Optimal clustering unchanged if
    - Intra-cluster distances shrunk
    - Inter-cluster distances expanded

Comparison with k-median

Min-sum k-Clustering

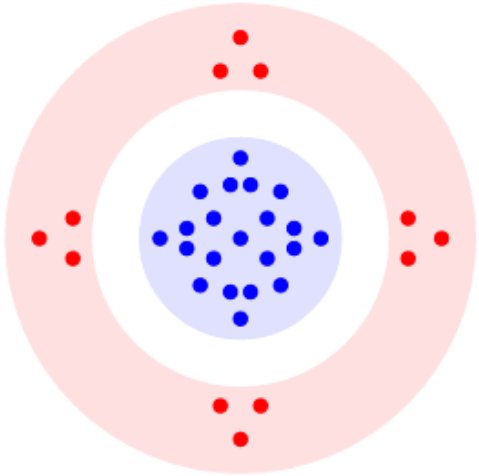
- k clusters $\mathcal{S} = \{S_1, S_2, \dots, S_k\}$

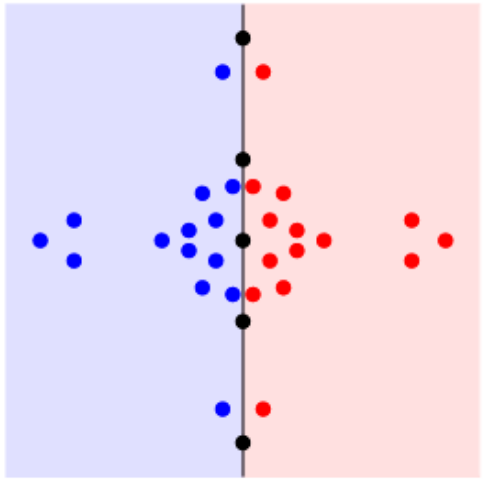
$$\sum_{t\in[k]} \sum_{i,j\in S_t} d(i,j)$$

k-median

- k clusters $\mathcal{S} = \{S_1, S_2, \dots, S_k\}$ and k centers $c_1, c_2, \dots, c_k$

$$\sum_{t\in[k]} \sum_{i\in S_t} d(i,c_t)$$





Figures credited to [KLRSZ25]

Connection with Balanced k-Median

- **Balanced k-median**
  - Metric space (D, d)
  - Objective: k clusters $\mathcal{S} = \{S_1, S_2, \dots, S_k\}$ and k centers $c_1, c_2, \dots, c_k$

$$\sum_{i \in [k]} |S_i| \sum_{j \in S_i} d(j, c_i)$$

Connection with Balanced k-Median

- **Balanced k-median**
  - Metric space (D, d)
  - Objective: k clusters $\mathcal{S} = \{S_1, S_2, \dots, S_k\}$ and k centers $c_1, c_2, \dots, c_k$

$$\sum_{i \in [k]} |S_i| \sum_{j \in S_i} d(j, c_i)$$

- **Relations to min-sum k-clustering**
  - Equivalent up to a factor of 2 in approximation
  - Many previous algorithms are built on this fact

History of the problem

History of the problem

- **Introduction**
  - Sahni and Gonzalez **[SG74;SG76]**
  - Among the "P-Complete" problems (hard unless $\mathrm{P} = \mathrm{NP}$)

History of the problem

- **Introduction**
  - Sahni and Gonzalez **[SG74;SG76]**
  - Among the "P-Complete" problems (hard unless $\mathrm{P} = \mathrm{NP}$)

- **Non-trivial Approximations**
  - $O(\epsilon^{-1}\log^{1+\epsilon} n)$ for any fixed $\epsilon > 0$ **[BCR01]**
  - $O(\log n)$ **[BFSS19]**

History of the problem

- **Introduction**
  - Sahni and Gonzalez [SG74;SG76]
  - Among the "P-Complete" problems (hard unless $\mathrm{P} = \mathrm{NP}$)

- **Non-trivial Approximations**
  - $O(\epsilon^{-1}\log^{1+\epsilon} n)$ for any fixed $\epsilon > 0$ [BCR01]
  - $O(\log n)$ [BFSS19]

- **Hardness**
  - APX-hard [GI03]
  - 1.415 [CKL21]

History of the problem

- **Introduction**
  - Sahni and Gonzalez [SG74;SG76]
  - Among the "P-Complete" problems (hard unless $\mathrm{P} = \mathrm{NP}$)
- **Non-trivial Approximations**
  - $O(\epsilon^{-1}\log^{1+\epsilon} n)$ for any fixed $\epsilon > 0$ [BCR01]
  - $O(\log n)$ [BFSS19]
- **Hardness**
  - APX-hard [GI03]
  - 1.415 [CKL21]

Major open problem
Is there a constant approximation?

Our Result

$O(1)$-**approximation**
for general metrics

- **The first constant-factor approximation**
  - Extend to objective $\sum_{S\in\mathcal{S}}\sum_{i,j\in S} d^z(i,j)$ for $z = O(1)$

Approximation in Other Settings

- **Parameterized complexity (for k)**
  - PTAS for metrics and squared Euclidean costs [FKKR03]
  - Sublinear time algorithm [KLRSZ25,26]
- **Structured metrics**
  - Quasi-PTAS [NRS24]:
    - bounded doubling metrics
    - graph of bounded treewidth / highway dimension
    - minor-free metrics
- **Bicriteria approximations**
  - $O(1)$-approximation with $O(k)$ clusters [BCR01]
  - $O(\epsilon^{-3})$-approximation by discarding $\leq \epsilon n$ points [BOR21]

Technical Overview

Algorithmic Framework

- Lagrange Multiplier Preserving (LMP) approximation

- Bipoint rounding

Algorithmic Framework

- **Lagrange Multiplier Preserving (LMP) approximation**
  - OPT: any optimal minsum solution
  - f: cluster opening cost
  - Our guarantee: ($|\mathrm{ALG}|$: =#clusters)

$$f|\mathrm{ALG}| + \mathrm{cost}(\mathrm{ALG}) \leq fk + \Gamma_1 \cdot \mathrm{cost}(\mathrm{OPT})$$

- **Bipoint rounding**

Algorithmic Framework

- **Lagrange Multiplier Preserving (LMP) approximation**
  - OPT: any optimal minsum solution
  - f: cluster opening cost
  - Our guarantee: ($|\mathrm{ALG}|$: =#clusters)

$$f|\mathrm{ALG}| + \mathrm{cost}(\mathrm{ALG}) \leq fk + \Gamma_1 \cdot \mathrm{cost}(\mathrm{OPT})$$

- **Bipoint rounding**
  - Given two LMP solutions $\mathcal{S}_1, \mathcal{S}_2$ with k_1, k_2 clusters
    - $k_1 < k < k_2$
  - Our guarantee: with $|ALG| = k$ and $\alpha = \frac{k_2 - k}{k_2 - k_1}$

$$\mathrm{cost}(\mathrm{ALG}) \leq \Gamma_2 \cdot (\alpha \cdot \mathrm{cost}(\mathcal{S}_1) + (1 - \alpha) \cdot \mathrm{cost}(\mathcal{S}_2))$$

Algorithmic Framework

- **Lagrange Multiplier Preserving (LMP) approximation**
  - OPT: any optimal minsum solution
  - f: cluster opening cost
  - Our guarantee: ($|\mathrm{ALG}|$: =#clusters)

$$f|\mathrm{ALG}| + \mathrm{cost}(\mathrm{ALG}) \leq fk + \Gamma_1 \cdot \mathrm{cost}(\mathrm{OPT})$$

- **Bipoint rounding**
  - Given two LMP solutions $\mathcal{S}_1, \mathcal{S}_2$ with k_1, k_2 clusters
    - $k_1 < k < k_2$
  - Our guarantee: with $|ALG| = k$ and $\alpha = \frac{k_2 - k}{k_2 - k_1}$

$$\mathrm{cost}(\mathrm{ALG}) \leq \Gamma_2 \cdot (\alpha \cdot \mathrm{cost}(\mathcal{S}_1) + (1-\alpha) \cdot \mathrm{cost}(\mathcal{S}_2))$$

Standard Combination [JV01]

$(\Gamma_1\Gamma_2 + \epsilon)$-approximation

Key Challenge: Size of Clusters

- **Objectives**
  - Min-sum k-clustering: $\sum_{S\in\mathcal{S}}\sum_{i\in S} d(i,S)$, where $d(i,S)=\sum_{j\in S} d(i,j)$
  - Balanced k-median: $\sum_{i\in[k]}\sum_{j\in S_i} |S_i| d(j,c_i)$

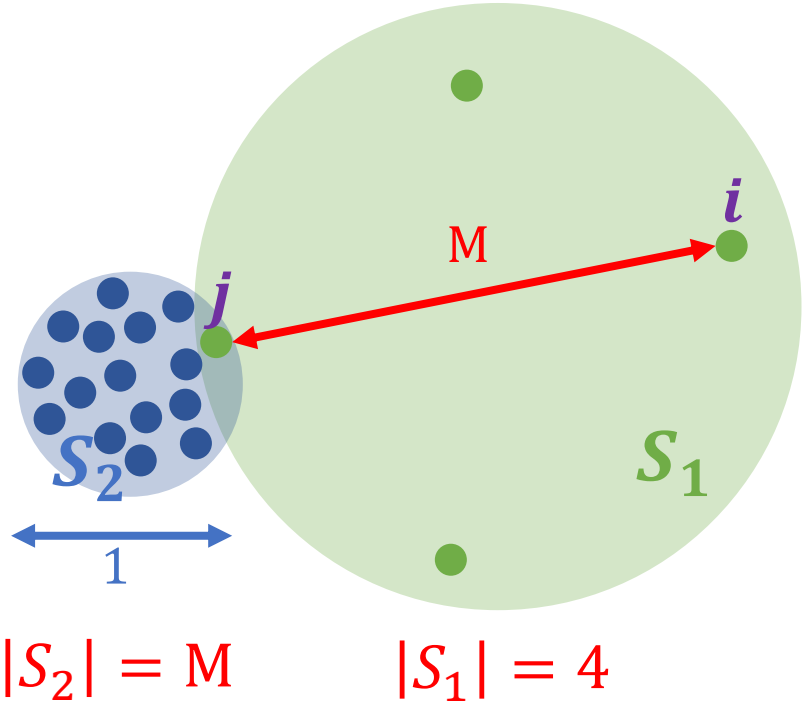
Key Challenge: Size of Clusters

- **Objectives**
  - Min-sum k-clustering: $\sum_{S \in \mathcal{S}} \sum_{i \in S} d(i, S)$, where $d(i, S) = \sum_{j \in S} d(i, j)$
  - Balanced k-median: $\sum_{i \in [k]} \sum_{j \in S_i} |S_i| d(j, c_i)$

Per-client cost increases with the size of the cluster

Key Challenge: Size of Clusters

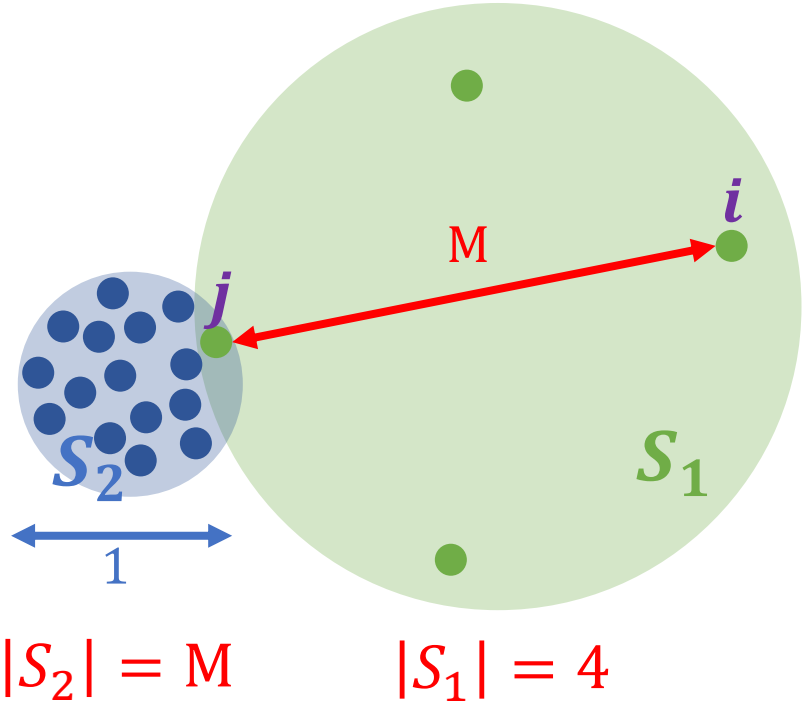
- **Objectives**
  - Min-sum k-clustering: $\sum_{S\in\mathcal{S}}\sum_{i\in S} d(i,S)$, where $d(i,S)=\sum_{j\in S} d(i,j)$
  - Balanced k-median: $\sum_{i\in[k]}\sum_{j\in S_i} |S_i| d(j,c_i)$



Per-client cost increases with the size of the cluster

Key Challenge: Size of Clusters

- **Objectives**
  - Min-sum k-clustering: $\sum_{S\in\mathcal{S}}\sum_{i\in S} d(i,S)$, where $d(i,S)=\sum_{j\in S} d(i,j)$
  - Balanced k-median: $\sum_{i\in[k]}\sum_{j\in S_i} |S_i| d(j,c_i)$

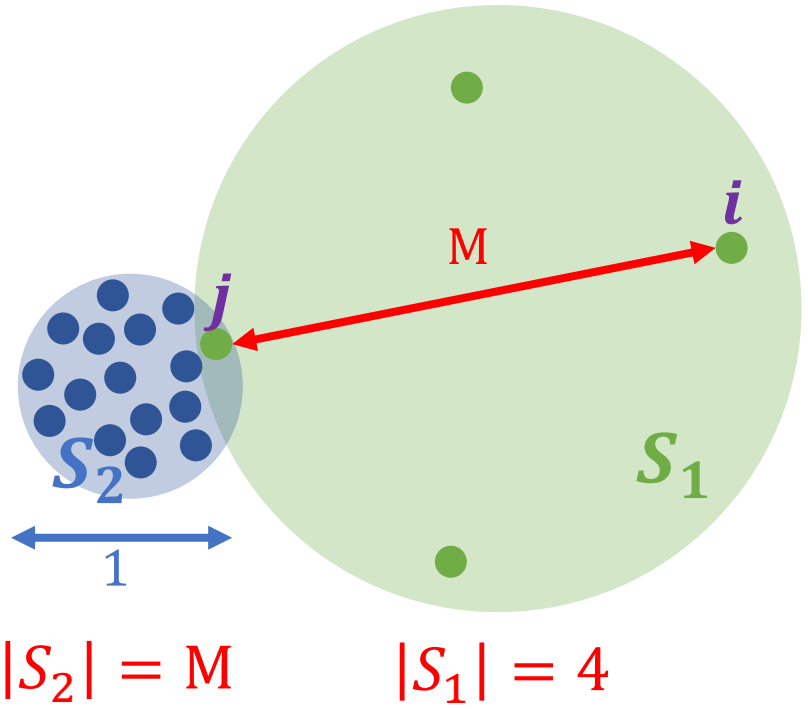


Per-client cost increases with the size of the cluster

$$d(i,S_1)=\Theta(M), \quad d(i,S_2)\approx\Theta(M^2)$$
$$d(j,S_1)=\Theta(M), \quad d(j,S_2)\approx\Theta(M)$$

Key Challenge: Size of Clusters

- **Objectives**
  - Min-sum k-clustering: $\sum_{S\in\mathcal{S}}\sum_{i\in S} d(i,S)$, where $d(i,S)=\sum_{j\in S} d(i,j)$
  - Balanced k-median: $\sum_{i\in[k]}\sum_{j\in S_i} |S_i| d(j,c_i)$

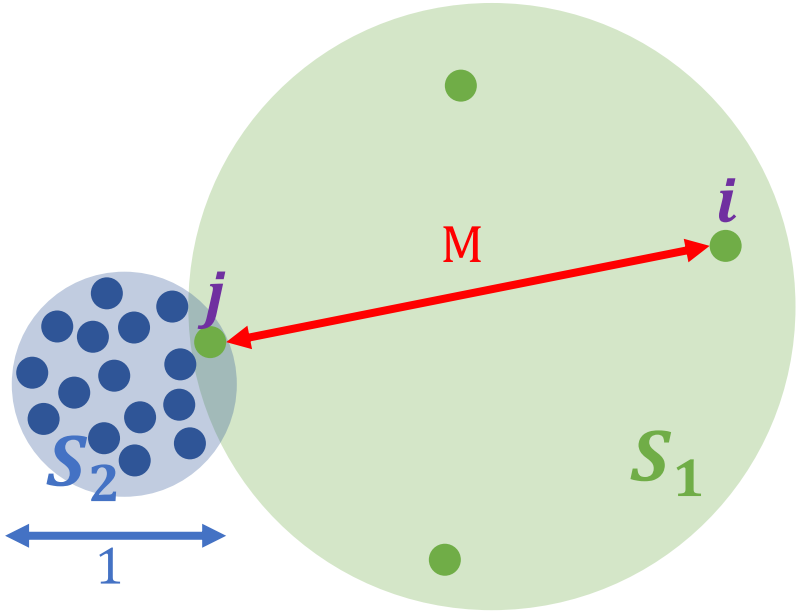


Per-client cost increases with the size of the cluster

$$d(i,S_1)=\Theta(M), \quad d(i,S_2)\approx\Theta(M^2)$$
$$d(j,S_1)=\Theta(M), \quad d(j,S_2)\approx\Theta(M)$$

Key Challenge: Size of Clusters

- **Objectives**
  - Min-sum k-clustering: $\sum_{S\in\mathcal{S}}\sum_{i\in S} d(i,S)$, where $d(i,S)=\sum_{j\in S} d(i,j)$
  - Balanced k-median: $\sum_{i\in[k]}\sum_{j\in S_i} |S_i| d(j,c_i)$



Per-client cost increases with the size of the cluster

$$d(i,S_1)=\Theta(M), \quad \boxed{d(i,S_2)\approx\Theta(M^2)}$$
$$d(j,S_1)=\Theta(M), \quad d(j,S_2)\approx\Theta(M)$$

NO triangle inequality like

$$d(i,S_2)\le O(1)\cdot (d(i,S_1)+d(j,S_2)+d(j,S_1))$$

LMP Approximation

Dual Fitting for LMP Approximations

- **Primal**

$$\min \sum_{S \subseteq D} (\text{cost}(S) + f) y_S$$

$$\text{s.t.} \sum_{S:\, i \in S} y_S \geq 1 \qquad \forall i \in D$$

$$y \geq 0$$

where $\text{cost(S)} = \sum_{i,j \in S} d(i,j)$

- **Dual**

$$\max \sum_{i \in D} \alpha_i$$

$$\text{s.t.} \sum_{i \in D} \alpha_i \leq f + cost(S) \quad \forall S \subseteq D$$

$$\alpha \geq 0$$

Dual Fitting for LMP Approximations

- **Primal**

$$\min \sum_{S \subseteq D} (\text{cost}(S) + f) y_S$$

$$\text{s.t.} \sum_{S:\, i \in S} y_S \geq 1 \qquad \forall i \in D$$

$$y \geq 0$$

where $\text{cost(S)} = \sum_{i,j \in S} d(i,j)$

- **Dual**

$$\max \sum_{i \in D} \alpha_i$$

$$\text{s.t.} \sum_{i \in D} \alpha_i \leq f + cost(S) \quad \forall S \subseteq D$$

$$\alpha \geq 0$$

- **Two desired properties**
  - **Dual payment.** $\sum_{i \in D} \alpha_i \geq f|ALG| + \sum_{S \in \text{ALG}} \text{cost}(S)$
  - **Dual feasibility.** $\sum_{i \in S} \alpha_i \leq f + \Gamma \cdot \text{cost}(S), \quad \forall S \subseteq D$

Dual Fitting for LMP Approximations

- **Primal**

$$\min \sum_{S \subseteq D} (\text{cost}(S) + f) y_S$$

$$\text{s.t.} \sum_{S:\, i \in S} y_S \geq 1 \qquad \forall i \in D$$

$$y \geq 0$$

where $\text{cost}(S) = \sum_{i,j \in S} d(i,j)$

- **Dual**

$$\max \sum_{i \in D} \alpha_i$$

$$\text{s.t.} \sum_{i \in D} \alpha_i \leq f + cost(S) \quad \forall S \subseteq D$$

$$\alpha \geq 0$$

- **Two desired properties**
  - **Dual payment.** $\sum_{i \in D} \alpha_i \geq f|ALG| + \sum_{S \in \text{ALG}} \text{cost}(S)$
  - **Dual feasibility.** $\sum_{i \in S} \alpha_i \leq f + \Gamma \cdot \text{cost}(S), \quad \forall S \subseteq D$



Γ-LMP approximation

A Simple Adaptation of [JV01]

- **Algorithm**

Dual payment. $\sum_{i \in D} \alpha_i \geq f|ALG| + \sum_{S \in \mathrm{ALG}} \mathrm{cost}(S)$

Dual feasibility. $\sum_{i \in S} \alpha_i \leq f + \Gamma \cdot \mathrm{cost}(S), \quad \forall S \subseteq D$

A Simple Adaptation of [JV01]

- **Algorithm**
  - Initialize $\alpha_i \equiv 0$ and all clients as **active**

Dual payment. $\sum_{i \in D} \alpha_i \geq f|ALG| + \sum_{S \in \mathrm{ALG}} \mathrm{cost}(S)$

Dual feasibility. $\sum_{i \in S} \alpha_i \leq f + \Gamma \cdot \mathrm{cost}(S), \quad \forall S \subseteq D$

A Simple Adaptation of [JV01]

- **Algorithm**
  - Initialize $\alpha_i \equiv 0$ and all clients as **active**
  - **(dual-growth)** Increase α_i for **active** clients, until
    - A cluster becomes openable:

      $\sum_{j \in S} \alpha_j = f + \text{cost}(S)$ for some S

      All client in S becomes **directly connected**
    - An **active** client pays for its connection:

      $\alpha_i = 2d(i, S)$ for existing S

      Client i becomes **indirectly connected**

Dual payment. $\sum_{i \in D} \alpha_i \geq f|ALG| + \sum_{S \in \text{ALG}} \text{cost}(S)$

Dual feasibility. $\sum_{i \in S} \alpha_i \leq f + \Gamma \cdot \text{cost}(S), \quad \forall S \subseteq D$

A Simple Adaptation of [JV01]

- **Algorithm**
  - Initialize $\alpha_i \equiv 0$ and all clients as **active**
  - **(dual-growth)** Increase α_i for **active** clients, until
    - A cluster becomes openable:
      $\sum_{j \in S} \alpha_j = f + \text{cost}(S)$ for some S
      All client in S becomes **directly connected**
    - An **active** client pays for its connection:
      $\alpha_i = 2d(i, S)$ for existing S
      Client i becomes **indirectly connected**
  - **(pruning)** After all clients become **inactive**,
    - Choose an independent set of clusters (no intersection)
    - Reroute each client to one of the chosen clusters

Dual payment. $\sum_{i \in D} \alpha_i \geq f|ALG| + \sum_{S \in \text{ALG}} \text{cost}(S)$
Dual feasibility. $\sum_{i \in S} \alpha_i \leq f + \Gamma \cdot \text{cost}(S), \quad \forall S \subseteq D$

A Simple Adaptation of [JV01]

- **Algorithm**
  - Initialize $\alpha_i \equiv 0$ and all clients as **active**
  - **(dual-growth)** Increase α_i for **active** clients, until
    - A cluster becomes openable:
      $\sum_{j \in S} \alpha_j = f + \text{cost}(S)$ for some S
      All client in S becomes **directly connected**
    - An **active** client pays for its connection:
      $\alpha_i = 2d(i, S)$ for existing S
      Client i becomes **indirectly connected**
  - **(pruning)** After all clients become **inactive**,
    - Choose an independent set of clusters (no intersection)
    - Reroute each client to one of the chosen clusters

Dual payment. $\sum_{i \in D} \alpha_i \geq f|ALG| + \sum_{S \in \text{ALG}} \text{cost}(S)$
Dual feasibility. $\sum_{i \in S} \alpha_i \leq f + \Gamma \cdot \text{cost}(S), \quad \forall S \subseteq D$

Issue 1.
$d(i, S)$ changes when S expands

Issue 2.
NO triangle inequality

A Simple Adaptation of [JV01]

- **Algorithm**
  - Initialize $\alpha_i \equiv 0$ and all clients as **active**
  - **(dual-growth)** Increase α_i for **active** clients, until
    - A cluster becomes openable:
      $\sum_{j \in S} \alpha_j = f + \text{cost}(S)$ for some S
      All client in S becomes **directly connected**
    - An **active** client pays for its connection:
      $\alpha_i = 2d(i, S)$ for existing S
      Client i becomes **indirectly connected**
  - **(pruning)** After all clients become **inactive**,
    - Choose an independent set of clusters (no intersection)
    - Reroute each client to one of the chosen clusters

Dual payment. $\sum_{i \in D} \alpha_i \geq f|ALG| + \sum_{S \in \text{ALG}} \text{cost}(S)$
Dual feasibility. $\sum_{i \in S} \alpha_i \leq f + \Gamma \cdot \text{cost}(S), \quad \forall S \subseteq D$

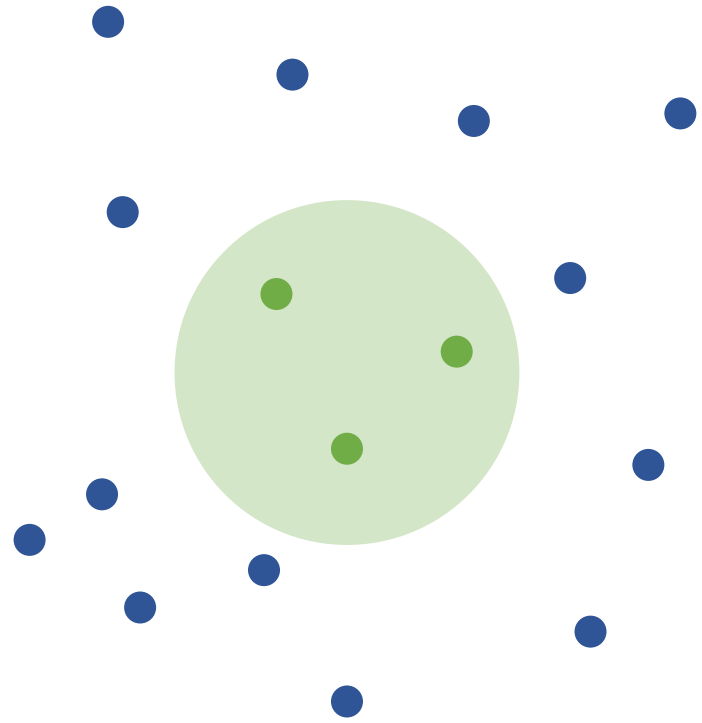
Hard to upper bound each α_i

Issue 1.
$d(i, S)$ changes when S expands

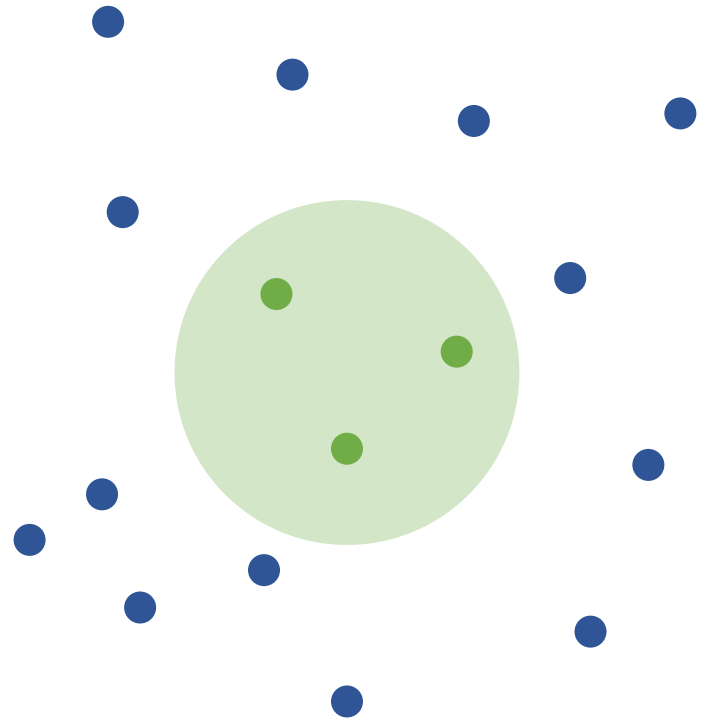
Issue 2.
NO triangle inequality

Resolving Issue 1: Cluster Splitting

Resolving Issue 1: Cluster Splitting



Resolving Issue 1: Cluster Splitting



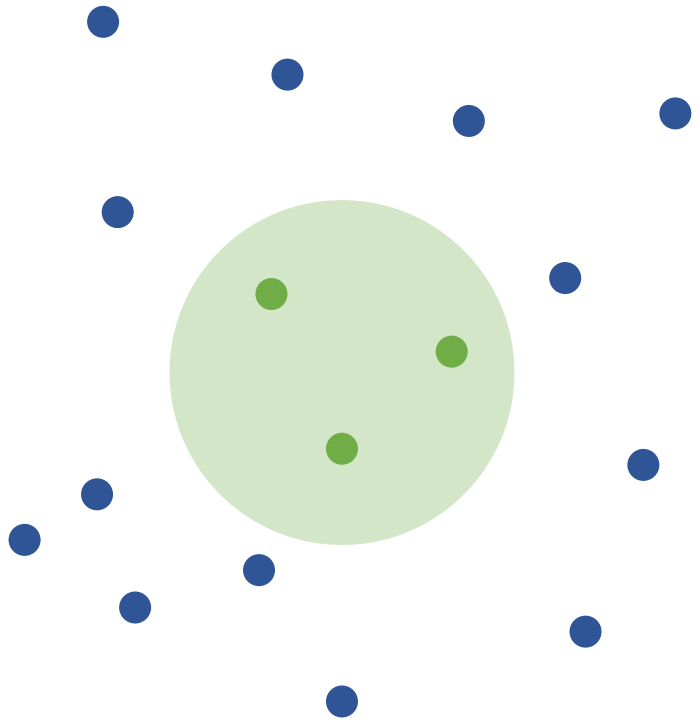
Consider two parts

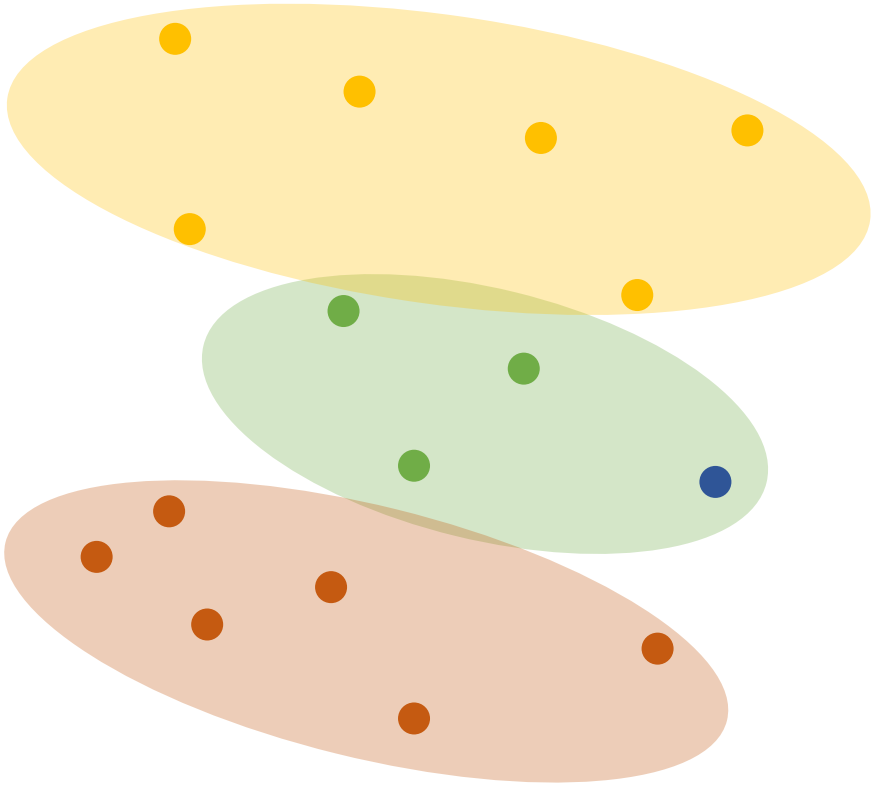
Core: $\sum_{i \in C_{\text{core}}} \alpha_i \geq f + \text{cost}(C_{\text{core}})$

Outer: $\alpha_i \geq \gamma_{IC} \cdot d(i, C_{\text{core}})$

and $\alpha_i \geq \max_{j \in C_{\text{core}}} \alpha_j$

Resolving Issue 1: Cluster Splitting





Consider two parts

Core: $\sum_{i \in C_{\text{core}}} \alpha_i \geq f + \text{cost}(C_{\text{core}})$

Outer: $\alpha_i \geq \gamma_{IC} \cdot d(i, C_{\text{core}})$
and $\alpha_i \geq \max_{j \in C_{\text{core}}} \alpha_j$

Split into three clusters

Remove $2|C_{\text{core}}|$ to form a new cluster
Each satisfies $\sum_{i \in C} \alpha_i \geq f + \text{cost}(C)$

Candidate Algorithm 1

- **Algorithm**
  - Initialize $\alpha_i \equiv 0$ and all clients **active**
  - Maintain set of original clusters: $\mathcal{S}$
  - **(dual-growth)** Increase α_i for **active** clients, until
    - A cluster becomes openable:

      $\sum_{j \in S} \alpha_j = f + \mathrm{cost}(S)$ for some S

      All client in S becomes **directly connected**, include S in $\mathcal{S}$
    - An **active** client pays for its connection:

      $\alpha_i = \gamma_{IC} d(i, S)$ for some $S \in \mathcal{S}$

      Client i becomes **indirectly connected**
  - **(pruning)** After all clients become **inactive**,
    - Choose an independent set of clusters (no intersection)
    - Reroute each client to one of the chosen clusters

Issue 1.
$d(i, S)$ changes when S expands

Issue 2.
NO triangle inequality

Candidate Algorithm 1

- **Algorithm**
  - Initialize $\alpha_i \equiv 0$ and all clients **active**
  - Maintain set of original clusters: $\mathcal{S}$
  - **(dual-growth)** Increase α_i for **active** clients, until
    - A cluster becomes openable:

      $\sum_{j \in S} \alpha_j = f + \mathrm{cost}(S)$ for some S

      All client in S becomes **directly connected**, include S in $\mathcal{S}$
    - An **active** client pays for its connection:

      $\alpha_i = \gamma_{IC} d(i, S)$ for some $S \in \mathcal{S}$

      Client i becomes **indirectly connected**
  - **(pruning)** After all clients become **inactive**,
    - Choose an independent set of clusters (no intersection)
    - Reroute each client to one of the chosen clusters

Issue 2.
NO triangle inequality

Candidate Algorithm 1

- **Algorithm**
  - Initialize $\alpha_i \equiv 0$ and all clients **active**
  - Maintain set of original clusters: $\mathcal{S}$
  - **(dual-growth)** Increase α_i for **active** clients, until
    - A cluster becomes openable:

      $\sum_{j \in S} \alpha_j = f + \mathrm{cost}(S)$ for some S

      All client in S becomes **directly connected**, include S in $\mathcal{S}$
    - An **active** client pays for its connection:

      $\alpha_i = \gamma_{IC} d(i, S)$ for some $S \in \mathcal{S}$

      Client i becomes **indirectly connected**
  - **(pruning)** After all clients become **inactive**,
    - Choose an independent set of clusters (no intersection)
    - Reroute each client to one of the chosen clusters

Issue 2.
NO triangle inequality

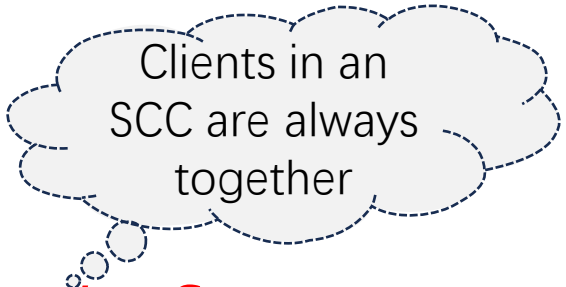
Resolving Issue 2: Dependencies

Resolving Issue 2: Dependencies

- **Dependency graph**
  - Directed graph over clients
  - A path $i \rightarrow j$ implies $i \in S \Longrightarrow j \in S$

Resolving Issue 2: Dependencies

- **Dependency graph**
  - Directed graph over clients
  - A path $i \to j$ implies $i \in S \Longrightarrow j \in S$



Resolving Issue 2: Dependencies

- **Dependency graph**
  - Directed graph over clients
  - A path $i \to j$ implies $i \in S \Longrightarrow j \in S$

Clients in an SCC are always together

- **Update rule**
  - At the time S is opened
  - We add an edge $i \to j$ for any $i \in S$ and any **active** client $j \in S$

Resolving Issue 2: Dependencies

- **Dependency graph**
  - Directed graph over clients
  - A path $i \to j$ implies $i \in S \Longrightarrow j \in S$

Clients in an SCC are always together

- **Update rule**
  - At the time S is opened
  - We add an edge $i \to j$ for any $i \in S$ and any **active** client $j \in S$

Active clients are always together

Resolving Issue 2: Dependencies

- **Dependency graph**
  - Directed graph over clients
  - A path $i \to j$ implies $i \in S \Longrightarrow j \in S$

Clients in an SCC are always together

- **Update rule**
  - At the time S is opened
  - We add an edge $i \to j$ for any $i \in S$ and any **active** client $j \in S$

Active clients are always together

Openable clusters.

1. Satisfies the dependency
2. $\alpha_j \geq \gamma_{DC} \cdot d(j, S), \quad \forall j \in S$
3. $\sum_{j \in S} \alpha_j \geq f + 2\gamma_{\mathrm{DC}} \mathrm{cost}(S)$
4. $|S| \ll$ any intersecting cluster
5. Others for the analysis

Resolving Issue 2: Dependencies

- **Dependency graph**
  - Directed graph over clients
  - A path $i \to j$ implies $i \in S \Longrightarrow j \in S$
- **Update rule**
  - At the time S is opened
  - We add an edge $i \to j$ for any $i \in S$ and any active client $j \in S$

Openable clusters.

1. Satisfies the dependency
2. $\alpha_j \geq \gamma_{DC} \cdot d(j, S), \quad \forall j \in S$
3. $\sum_{j \in S} \alpha_j \geq f + 2\gamma_{\mathrm{DC}} \mathrm{cost}(S)$
4. $|S| \ll$ any intersecting cluster

Resolving Issue 2: Dependencies

- **Dependency graph**
  - Directed graph over clients
  - A path $i \to j$ implies $i \in S \Longrightarrow j \in S$
- **Update rule**
  - At the time S is opened
  - We add an edge $i \to j$ for any $i \in S$ and any active client $j \in S$

Openable clusters.

1. Satisfies the dependency
2. $\alpha_j \geq \gamma_{DC} \cdot d(j, S), \quad \forall j \in S$
3. $\sum_{j \in S} \alpha_j \geq f + 2\gamma_{\mathrm{DC}} \mathrm{cost}(S)$
4. $|S| \ll$ any intersecting cluster

Pruning.

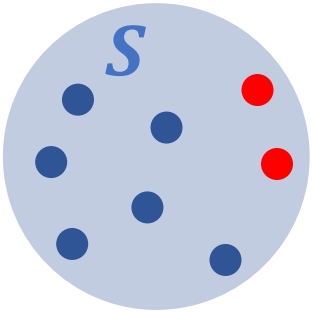
- Choose an independent set of clusters (no intersection), **higher priority for later opened clusters**
- Reroute each client to one of the chosen clusters

Resolving Issue 2: Dependencies

- **Dependency graph**
  - Directed graph over clients
  - A path $i \to j$ implies $i \in S \Longrightarrow j \in S$
- **Update rule**
  - At the time S is opened
  - We add an edge $i \to j$ for any $i \in S$ and any active client $j \in S$

Openable clusters.

1. Satisfies the dependency
2. $\alpha_j \geq \gamma_{DC} \cdot d(j, S), \quad \forall j \in S$
3. $\sum_{j \in S} \alpha_j \geq f + 2\gamma_{\mathrm{DC}} \mathrm{cost}(S)$
4. $|S| \ll$ any intersecting cluster



Pruning.

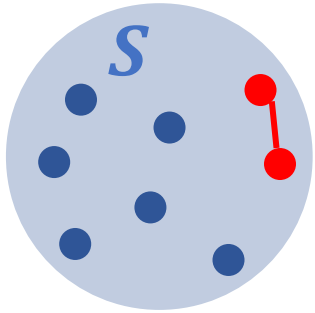
- Choose an independent set of clusters (no intersection), **higher priority for later opened clusters**
- Reroute each client to one of the chosen clusters

Resolving Issue 2: Dependencies

- **Dependency graph**
  - Directed graph over clients
  - A path $i \to j$ implies $i \in S \Longrightarrow j \in S$
- **Update rule**
  - At the time S is opened
  - We add an edge $i \to j$ for any $i \in S$ and any active client $j \in S$

Openable clusters.

1. Satisfies the dependency
2. $\alpha_j \geq \gamma_{DC} \cdot d(j, S), \quad \forall j \in S$
3. $\sum_{j \in S} \alpha_j \geq f + 2\gamma_{\mathrm{DC}} \mathrm{cost}(S)$
4. $|S| \ll$ any intersecting cluster



Pruning.

- Choose an independent set of clusters (no intersection), **higher priority for later opened clusters**
- Reroute each client to one of the chosen clusters

Resolving Issue 2: Dependencies

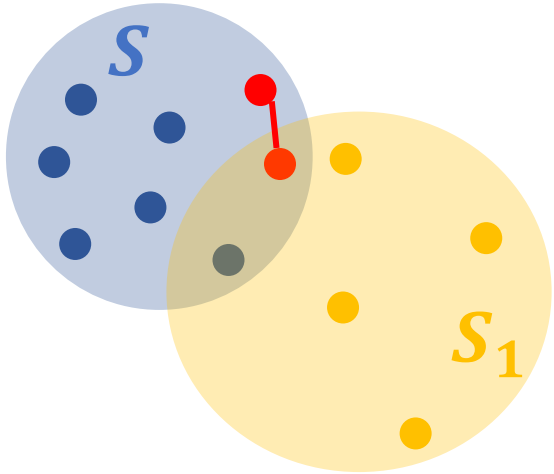
- **Dependency graph**
  - Directed graph over clients
  - A path $i \to j$ implies $i \in S \Longrightarrow j \in S$
- **Update rule**
  - At the time S is opened
  - We add an edge $i \to j$ for any $i \in S$ and any active client $j \in S$

Openable clusters.

1. Satisfies the dependency
2. $\alpha_j \geq \gamma_{DC} \cdot d(j, S), \quad \forall j \in S$
3. $\sum_{j \in S} \alpha_j \geq f + 2\gamma_{\mathrm{DC}} \mathrm{cost}(S)$
4. $|S| \ll$ any intersecting cluster

Pruning.

- Choose an independent set of clusters (no intersection), **higher priority for later opened clusters**
- Reroute each client to one of the chosen clusters

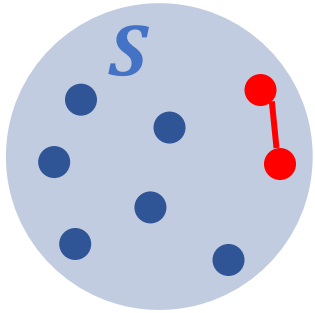


Resolving Issue 2: Dependencies

- **Dependency graph**
  - Directed graph over clients
  - A path $i \to j$ implies $i \in S \Longrightarrow j \in S$
- **Update rule**
  - At the time S is opened
  - We add an edge $i \to j$ for any $i \in S$ and any active client $j \in S$

Openable clusters.

1. Satisfies the dependency
2. $\alpha_j \geq \gamma_{DC} \cdot d(j, S), \quad \forall j \in S$
3. $\sum_{j \in S} \alpha_j \geq f + 2\gamma_{\mathrm{DC}} \mathrm{cost}(S)$
4. $|S| \ll$ any intersecting cluster



Pruning.

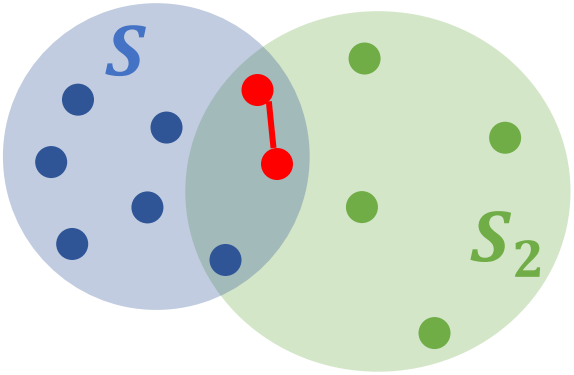
- Choose an independent set of clusters (no intersection), **higher priority for later opened clusters**
- Reroute each client to one of the chosen clusters

Resolving Issue 2: Dependencies

- **Dependency graph**
  - Directed graph over clients
  - A path $i \to j$ implies $i \in S \Longrightarrow j \in S$
- **Update rule**
  - At the time S is opened
  - We add an edge $i \to j$ for any $i \in S$ and any active client $j \in S$

Openable clusters.

1. Satisfies the dependency
2. $\alpha_j \geq \gamma_{DC} \cdot d(j, S), \quad \forall j \in S$
3. $\sum_{j \in S} \alpha_j \geq f + 2\gamma_{\mathrm{DC}} \mathrm{cost}(S)$
4. $|S| \ll$ any intersecting cluster



Pruning.

- Choose an independent set of clusters (no intersection), **higher priority for later opened clusters**
- Reroute each client to one of the chosen clusters

Our algorithm

- **Algorithm**
  - Initialize $\alpha_i \equiv 0$ and all clients **active**
  - Maintain set of original clusters: $\mathcal{S}$
  - **(dual-growth)** Increase α_i for **active** clients, until
    - A cluster becomes openable:

      some S satisfying all conditions

      All client in S becomes **directly connected**, include S in $\mathcal{S}$
    - A client pays for its connection:

      $\alpha_i = \gamma_{IC} d(i, S)$ for some $S \in \mathcal{S}$

      Client i becomes **indirectly connected**
  - **(pruning)** After all clients become inactive,
    - Choose an independent set of clusters (no intersection)
    - Connect each client to one of the chosen clusters

Issue 2.
NO triangle inequality

Our algorithm

- **Algorithm**
  - Initialize $\alpha_i \equiv 0$ and all clients **active**
  - Maintain set of original clusters: $\mathcal{S}$
  - **(dual-growth)** Increase α_i for **active** clients, until
    - A cluster becomes openable:

      some S satisfying all conditions

      All client in S becomes **directly connected**, include S in $\mathcal{S}$
    - A client pays for its connection:

      $\alpha_i = \gamma_{IC} d(i, S)$ for some $S \in \mathcal{S}$

      Client i becomes **indirectly connected**
  - **(pruning)** After all clients become inactive,
    - Choose an independent set of clusters (no intersection)
    - Connect each client to one of the chosen clusters

Our algorithm

- **Algorithm**
  - Initialize $\alpha_i \equiv 0$ and all clients **active**
  - Maintain set of original clusters: $\mathcal{S}$
  - **(dual-growth)** Increase α_i for **active** cli
    - A cluster becomes openable:

      some S satisfying all conditions

      All client in S becomes **directly connected**, include S in $\mathcal{S}$
    - A client pays for its connection:

      $\alpha_i = \gamma_{IC} d(i, S)$ for some $S \in \mathcal{S}$

      Client i becomes **indirectly connected**
  - **(pruning)** After all clients become inactive,
    - Choose an independent set of clusters (no intersection)
    - Connect each client to one of the chosen clusters

Using the connection with balanced k-median, we can efficiently implement this approximately

More Challenges

Dual feasibility. $\forall S$,
$\sum_{i \in S} \alpha_i \leq f + \Gamma \text{cost}(S)$

Some new ideas.

More Challenges

Dual feasibility. $\forall S$,
$\sum_{i \in S} \alpha_i \leq f + \Gamma \text{cost}(S)$

No direct guarantee
from our algorithm

Some new ideas.

More Challenges

Dual feasibility. $\forall S$,
$\sum_{i \in S} \alpha_i \leq f + \Gamma \text{cost}(S)$

No direct guarantee from our algorithm

Some new ideas.

Edit S to a "related" cluster that satisfies the dependencies.

More Challenges

Dual feasibility. $\forall S$,
$\sum_{i \in S} \alpha_i \leq f + \Gamma \mathrm{cost}(S)$

No direct guarantee from our algorithm

Some new ideas.

Edit S to a "related" cluster that satisfies the dependencies.

via combining several non-trivial observations from the openable conditions

Bipoint Rounding

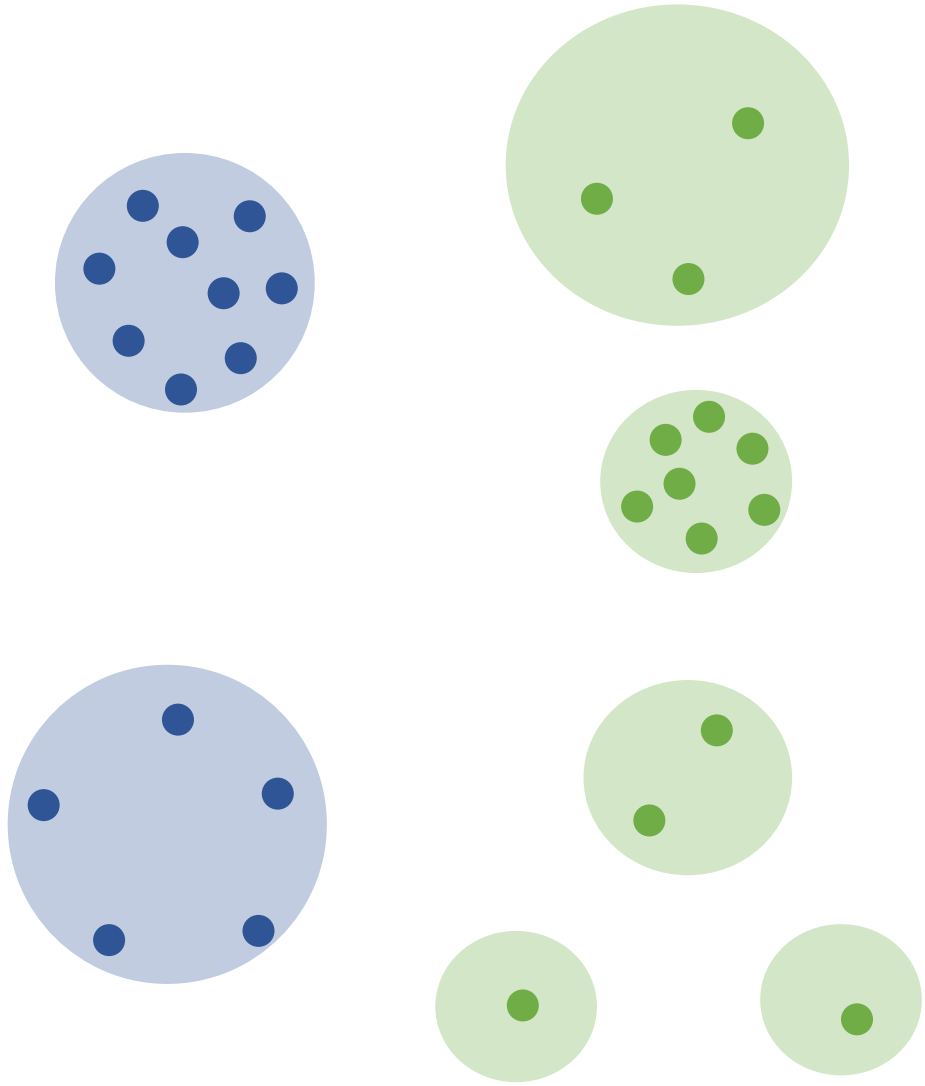
Algorithmic Framework

- **Bipoint rounding**
  - Given two LMP solutions S_1, S_2 with k_1, k_2 clusters
    - $k_1 < k < k_2$
  - Our guarantee: with $|ALG| = k$ and $\alpha = \frac{k_2 - k}{k_2 - k_1}$

$$\mathrm{cost(ALG)} \leq \Gamma_2 \cdot (\alpha \cdot \mathrm{cost(S_1)} + (1 - \alpha) \cdot \mathrm{cost}(S_2))$$

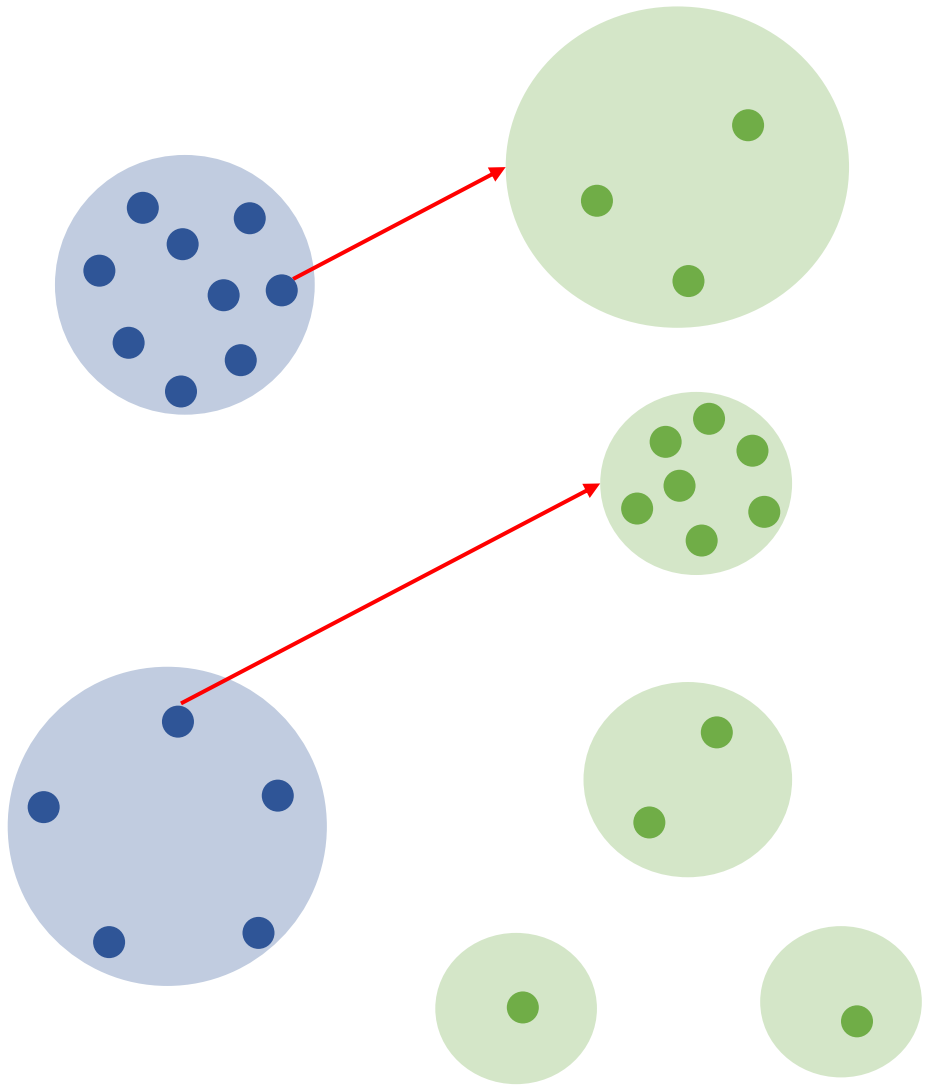
Challenges

- A simple adaptation of [JV01]



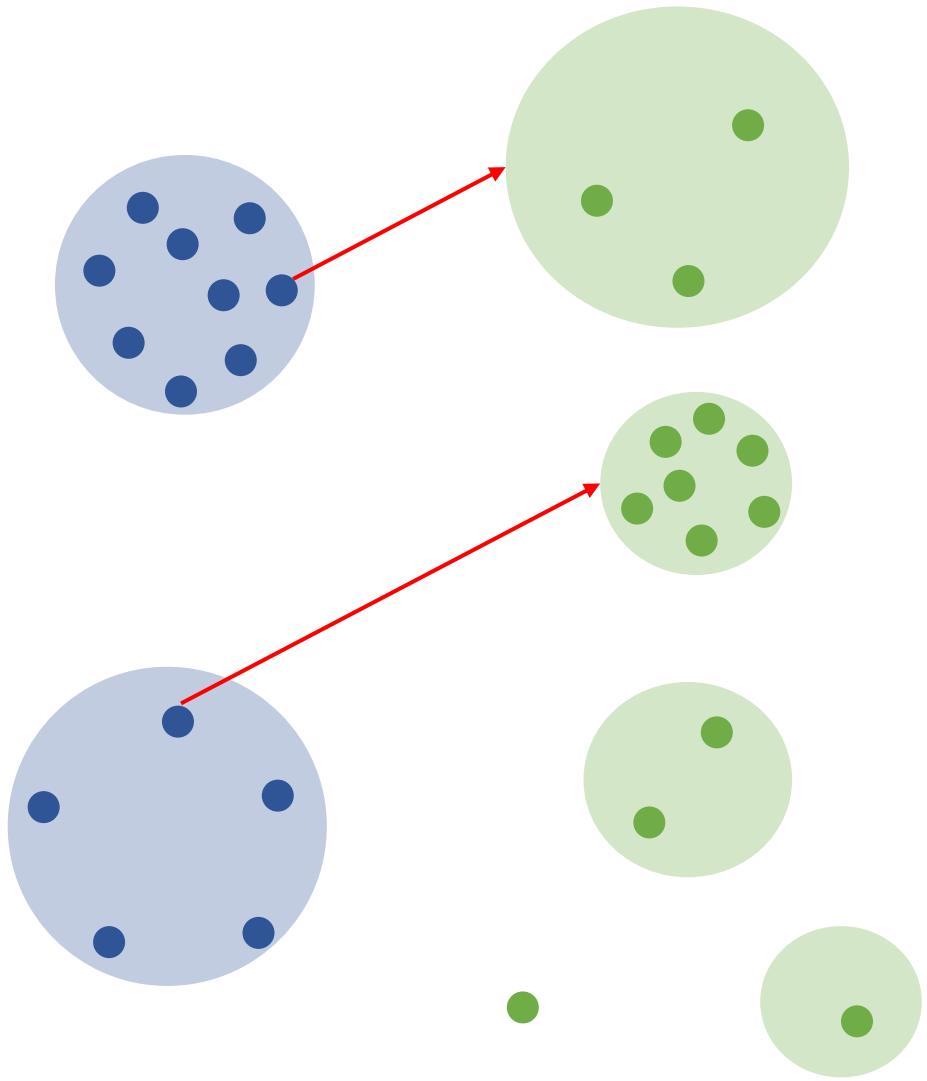
Challenges

- **A simple adaptation of** [JV01]
  1. Label each $\mathcal{S}_1$-cluster by a random intersecting $\mathcal{S}_2$-cluster



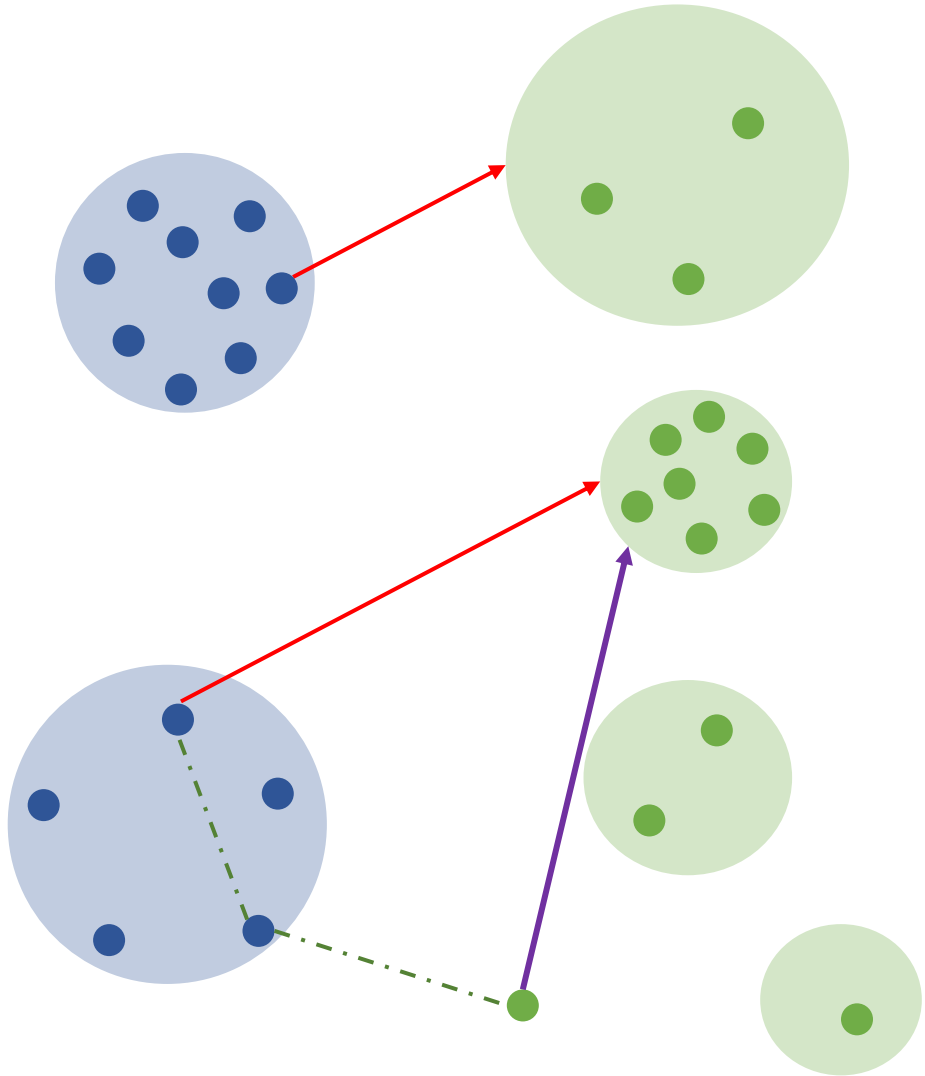
Challenges

- **A simple adaptation of [JV01]**
  1. Label each $\mathcal{S}_1$-cluster by a random intersecting $\mathcal{S}_2$-cluster
  2. Remove $k_2 - k$ remaining $\mathcal{S}_2$- clusters, randomly



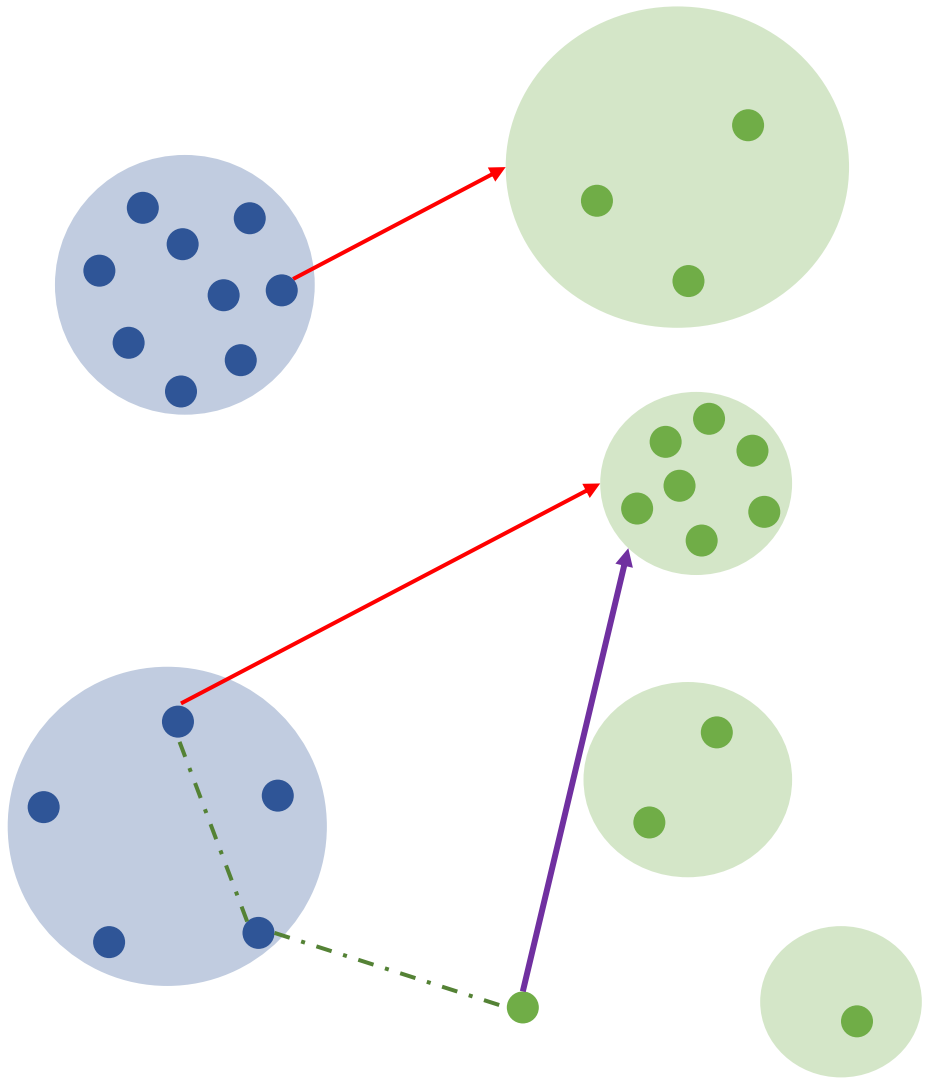
Challenges

- **A simple adaptation of [JV01]**
  1. Label each $\mathcal{S}_1$-cluster by a random intersecting $\mathcal{S}_2$-cluster
  2. Remove $k_2 - k$ remaining $\mathcal{S}_2$- clusters, randomly
  3. For clients whose $\mathcal{S}_2$-cluster is removed, reroute using its $\mathcal{S}_1$-cluster's label



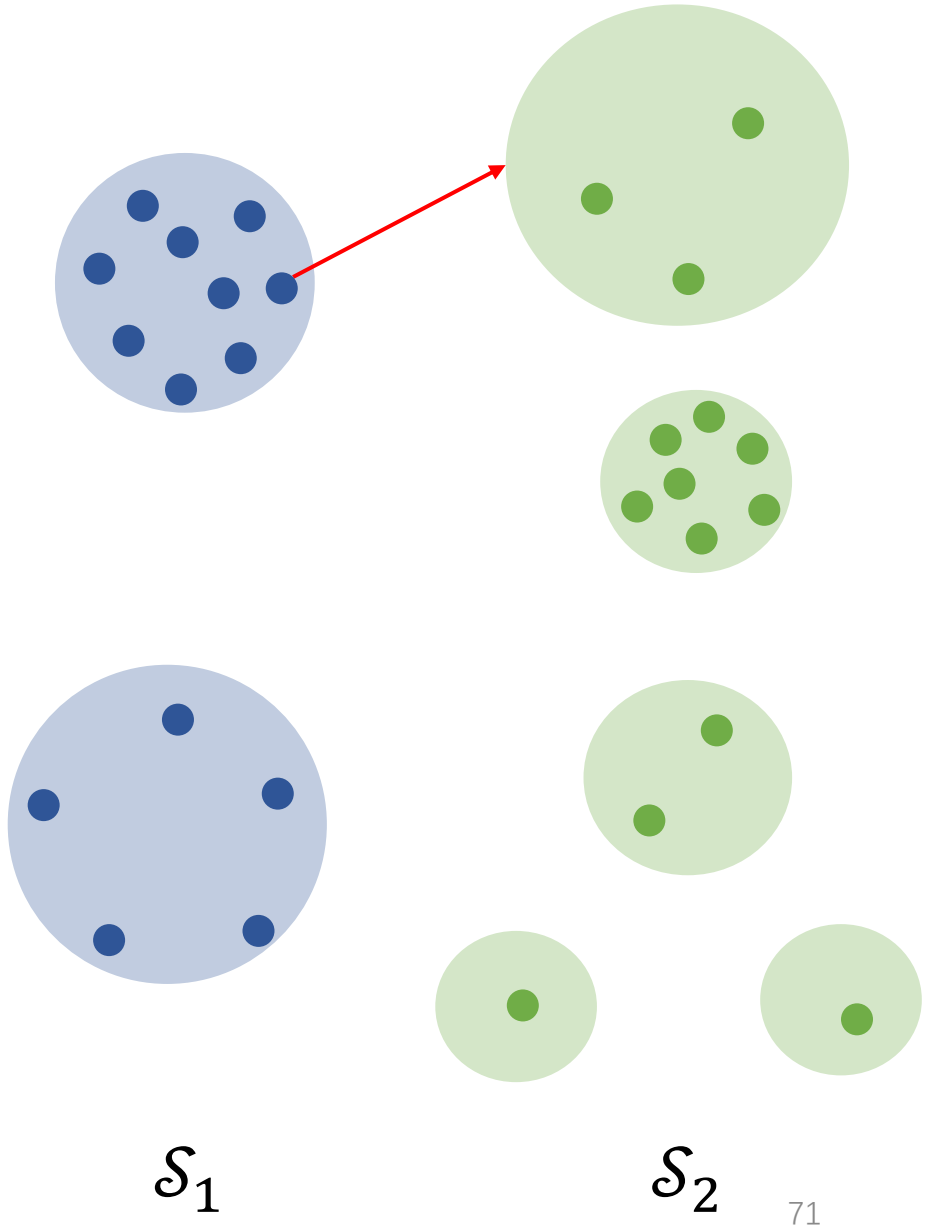
Challenges

- **A simple adaptation of [JV01]**
  1. Label each $\mathcal{S}_1$-cluster by a random intersecting $\mathcal{S}_2$-cluster
  2. Remove $k_2 - k$ remaining $\mathcal{S}_2$- clusters, randomly
  3. For clients whose $\mathcal{S}_2$-cluster is removed, reroute using its $\mathcal{S}_1$-cluster's label
- **Issue**
  - No triangle inequality
  - Clients in Step 3 has no cost bounds



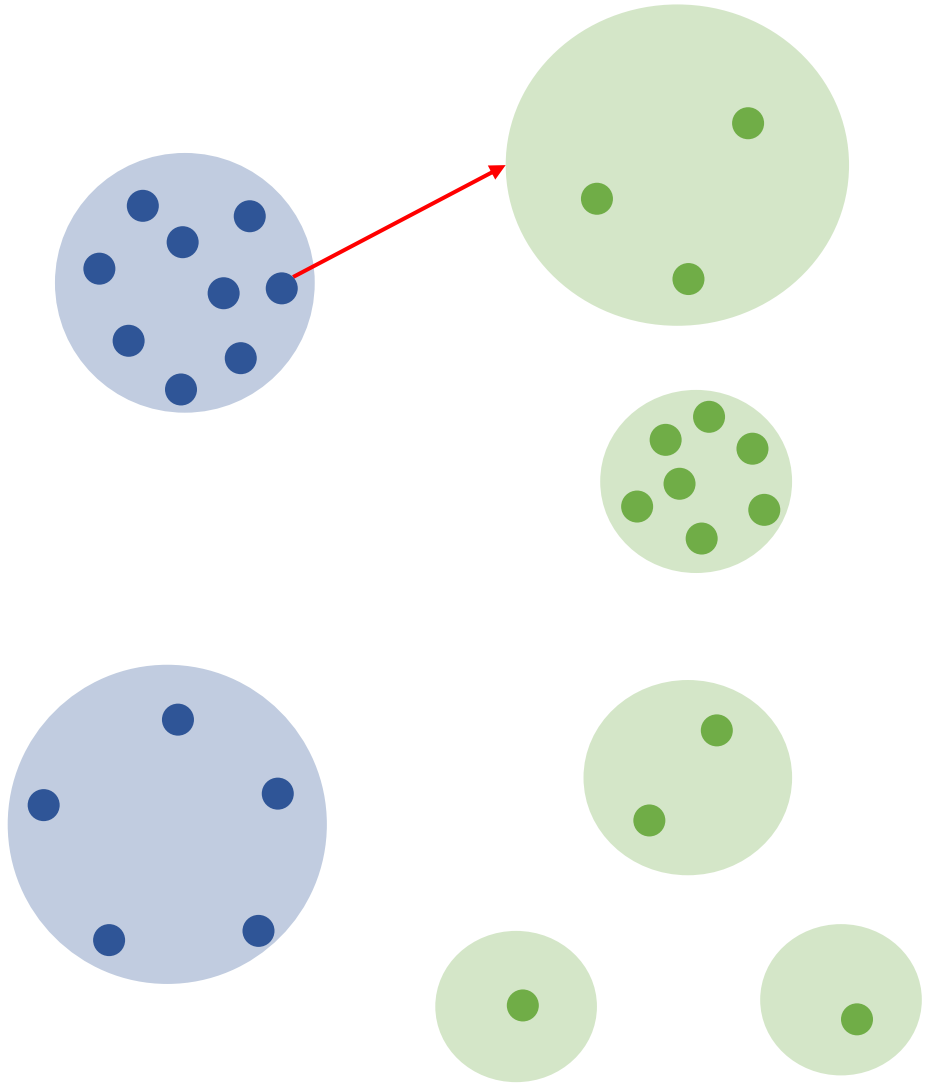
Reverse Labelling

- **Good&Bad clients**
  - Good clients: whose $\mathcal{S}_1$-cluster $\geq$ $\mathcal{S}_2$-cluster
  - Bad clients: otherwise



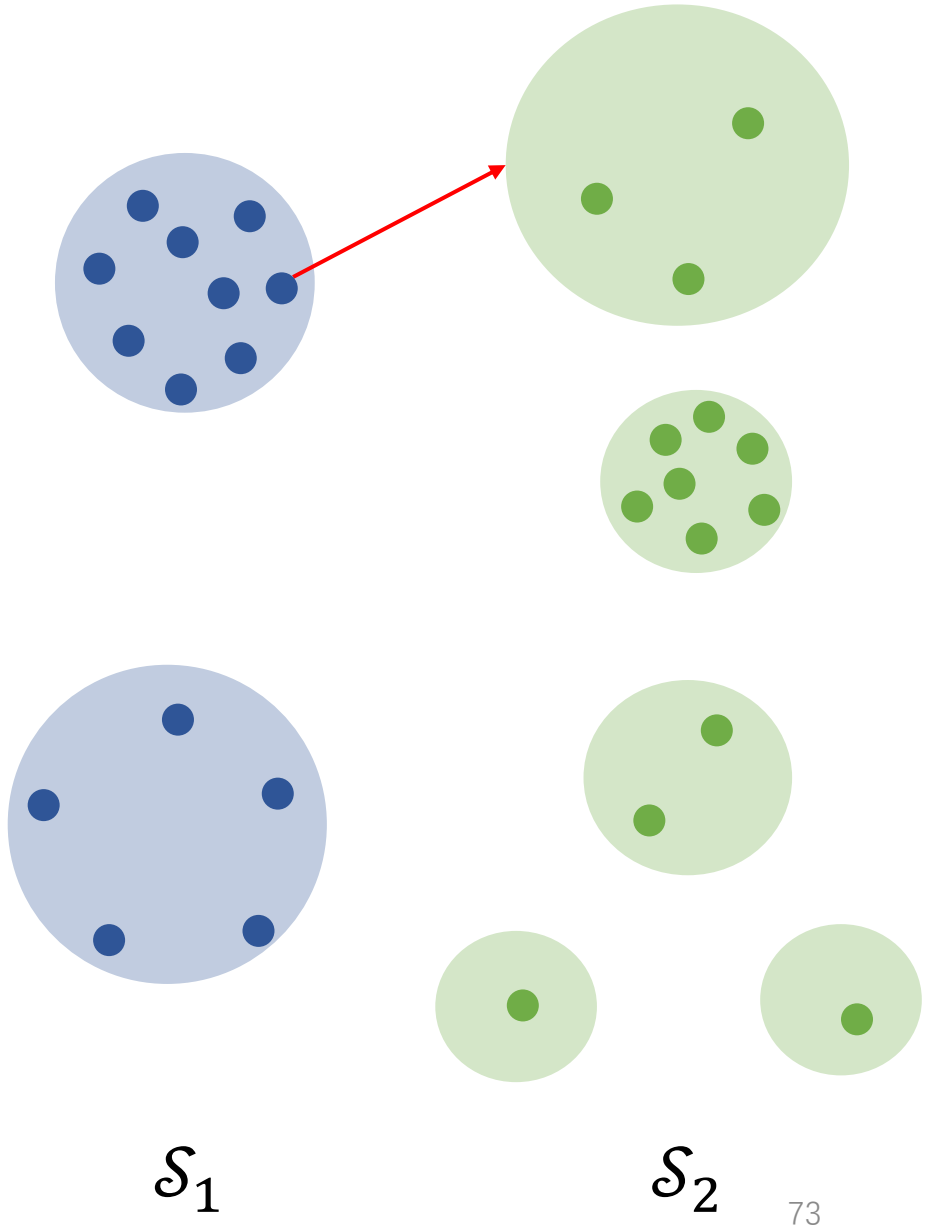
Reverse Labelling

- **Good&Bad clients**
  - Good clients: whose $\mathcal{S}_1$-cluster $\geq$ $\mathcal{S}_2$-cluster
  - Bad clients: otherwise
- **Bad clusters**
  - Have at least one bad client



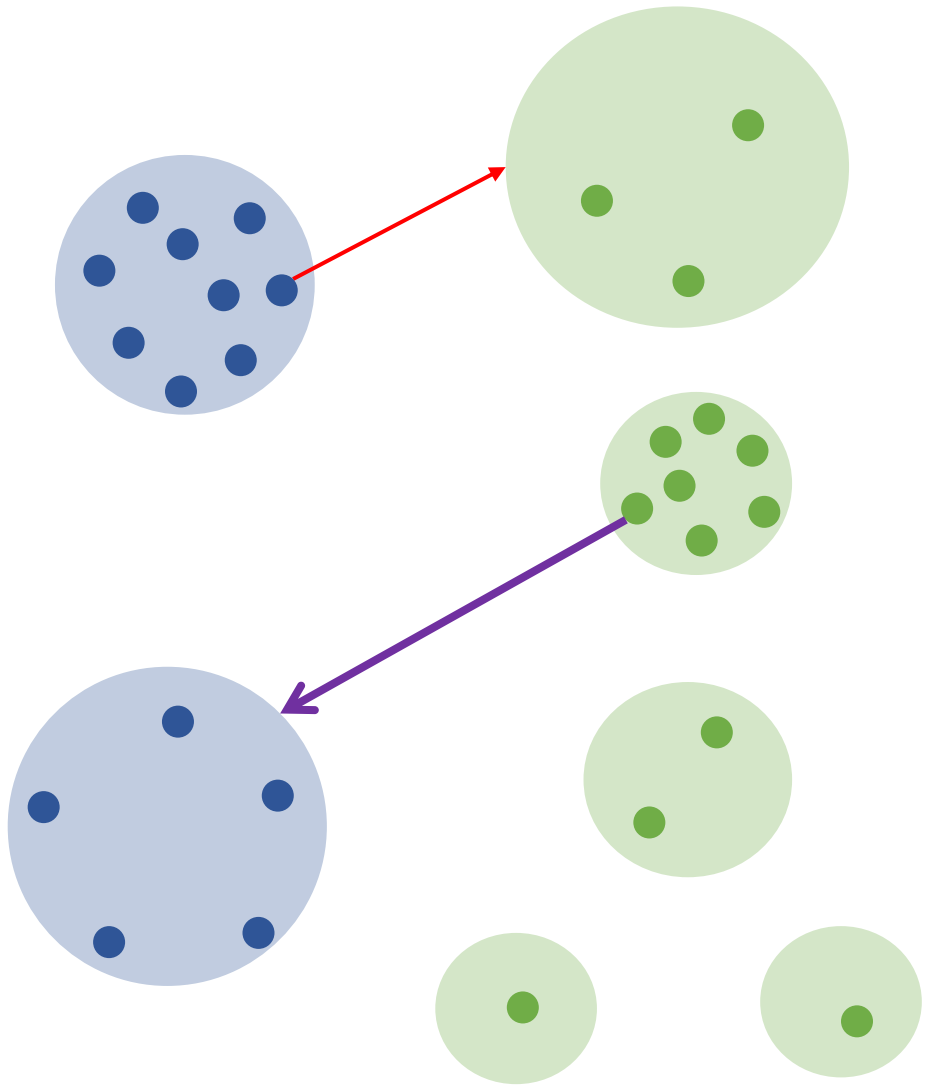
Reverse Labelling

- **Good&Bad clients**
  - Good clients: whose $\mathcal{S}_1$-cluster $\geq$ $\mathcal{S}_2$-cluster
  - Bad clients: otherwise
- **Bad clusters**
  - Have at least one bad client
- **Reverse labelling**
  - For bad $\mathcal{S}_2$-clusters
  - Label it by a random intersecting $\mathcal{S}_1$-cluster
  - Reroute its bad clients



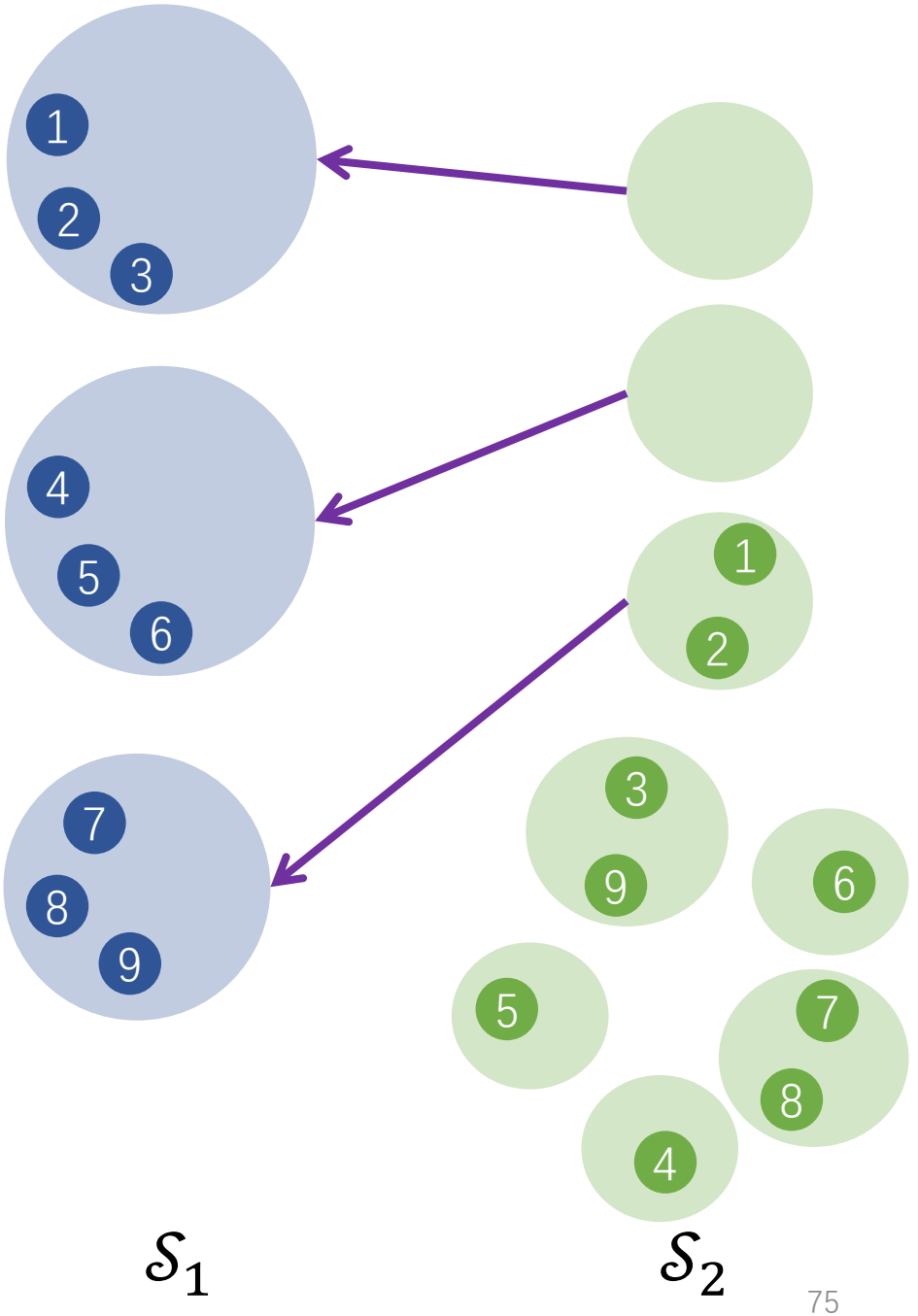
Reverse Labelling

- **Good&Bad clients**
  - Good clients: whose $\mathcal{S}_1$-cluster $\geq$ $\mathcal{S}_2$-cluster
  - Bad clients: otherwise
- **Bad clusters**
  - Have at least one bad client
- **Reverse labelling**
  - For bad $\mathcal{S}_2$-clusters
  - Label it by a random intersecting $\mathcal{S}_1$-cluster
  - Reroute its bad clients



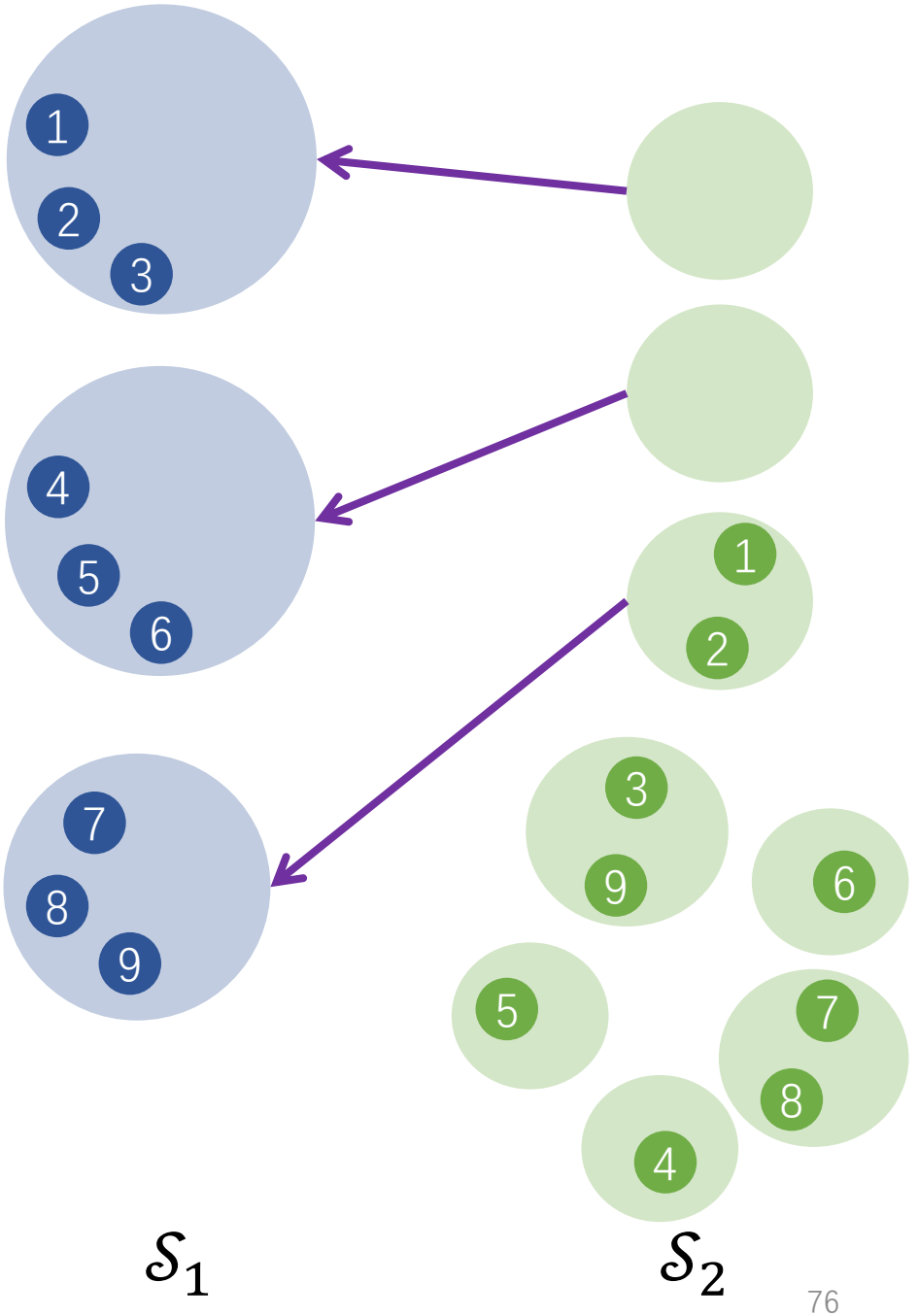
A Simple Case

- **Case**
  - Reverse labels are pairwise different



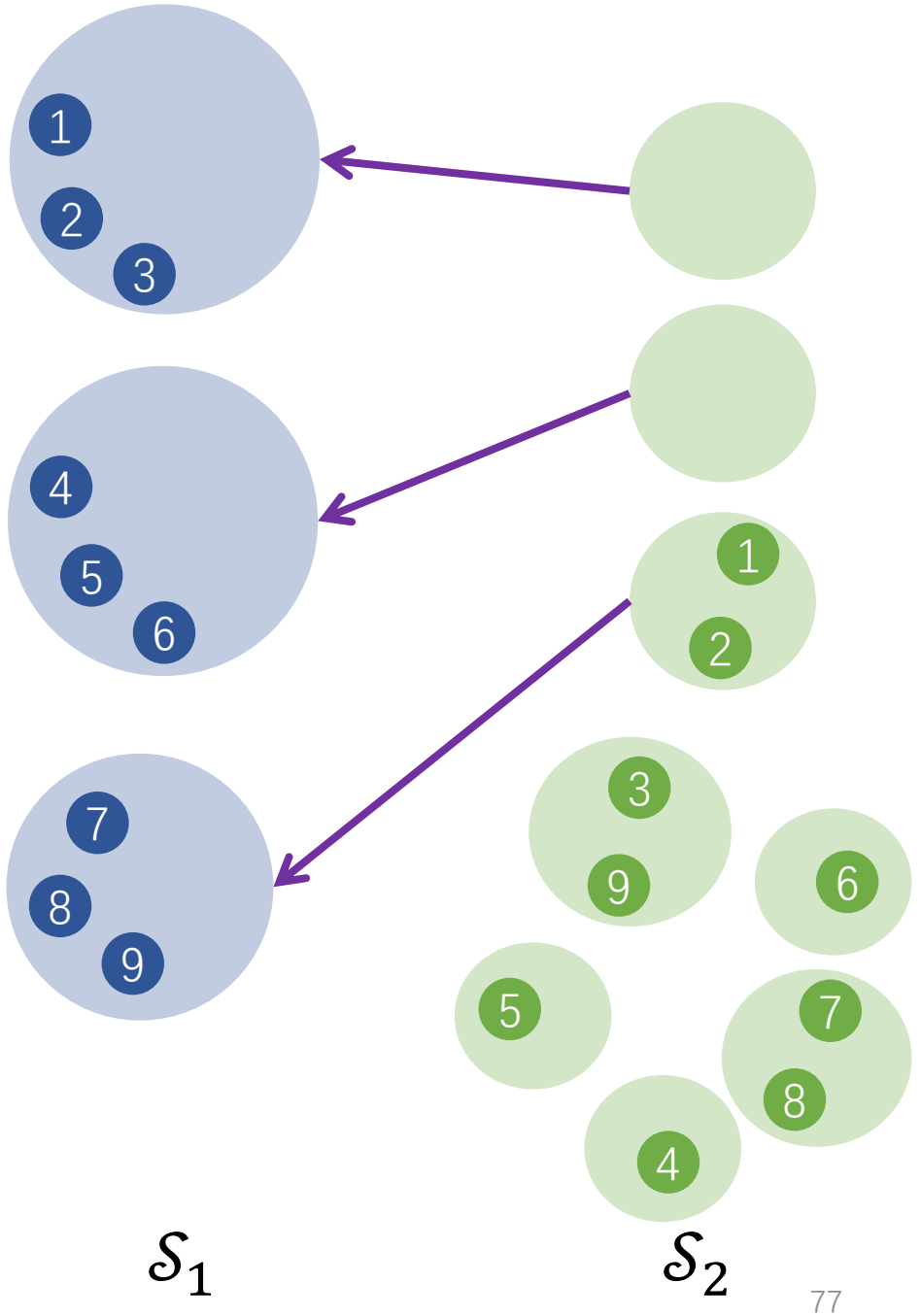
A Simple Case

- **Case**
  - Reverse labels are pairwise different
- **Iterative algorithm**



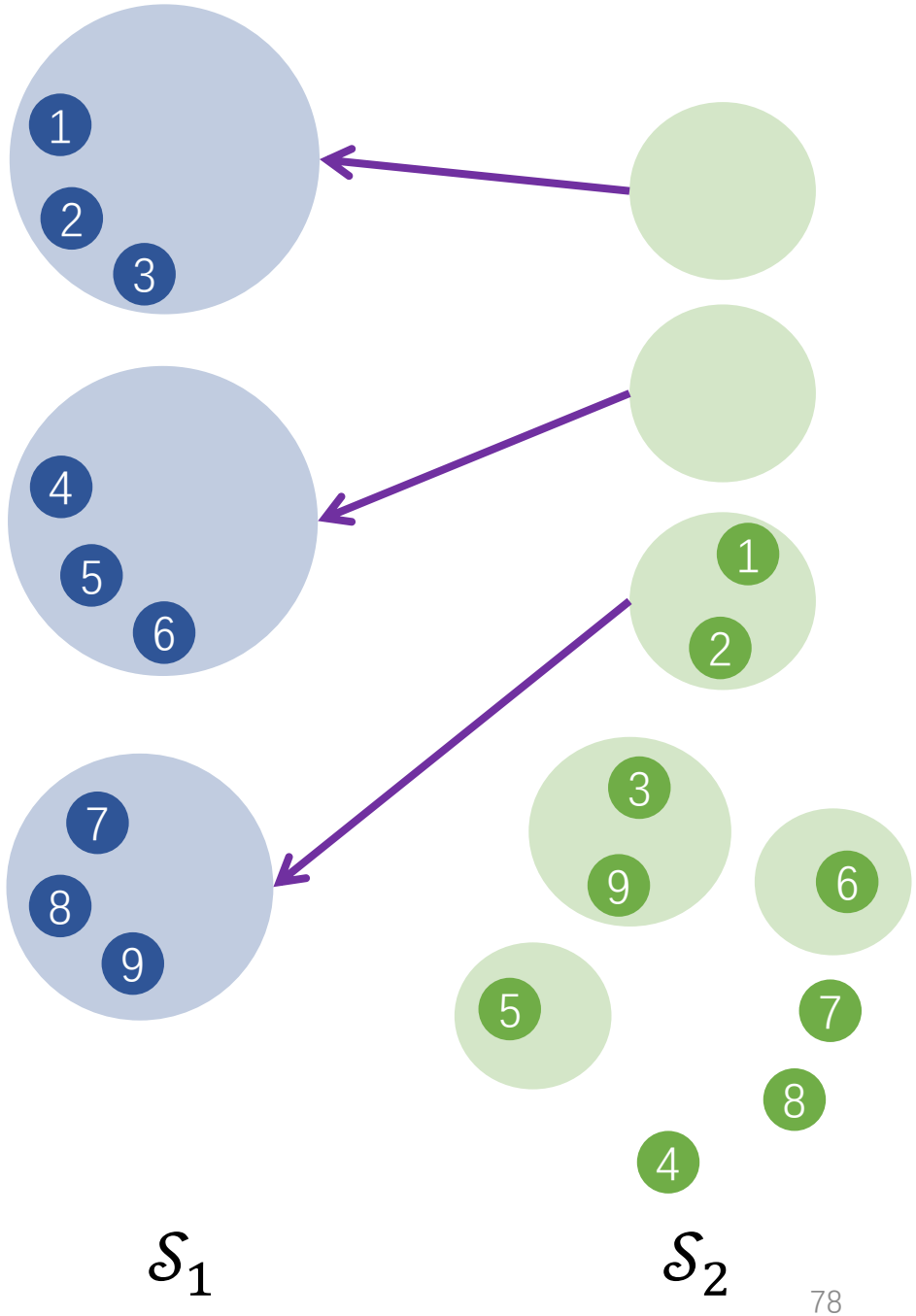
A Simple Case

- **Case**
  - Reverse labels are pairwise different
- **Iterative algorithm**
  1. Remove $k_2 - k$ remaining S_2-clusters



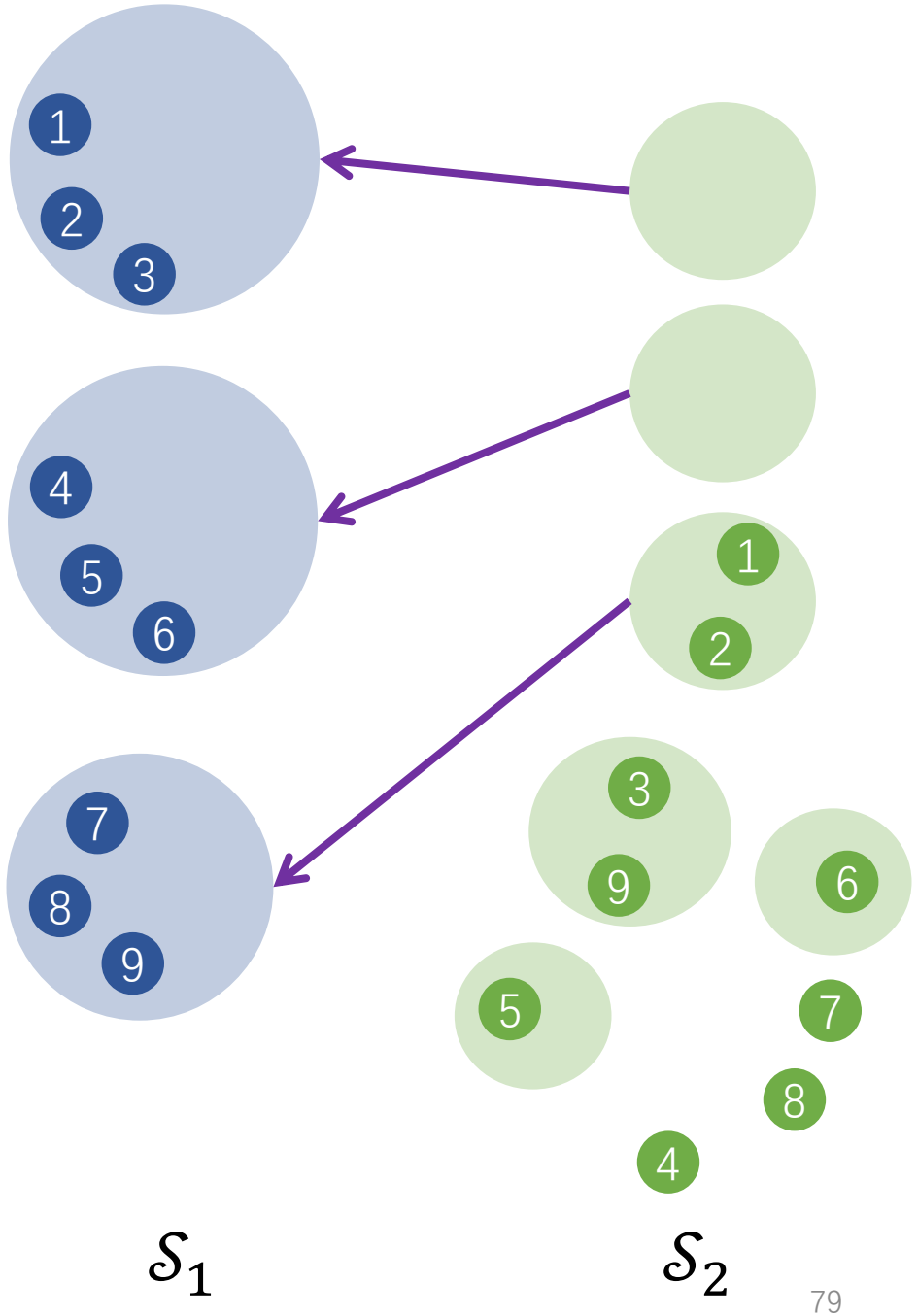
A Simple Case

- **Case**
  - Reverse labels are pairwise different
- **Iterative algorithm**
  1. Remove $k_2 - k$ remaining S_2-clusters



A Simple Case

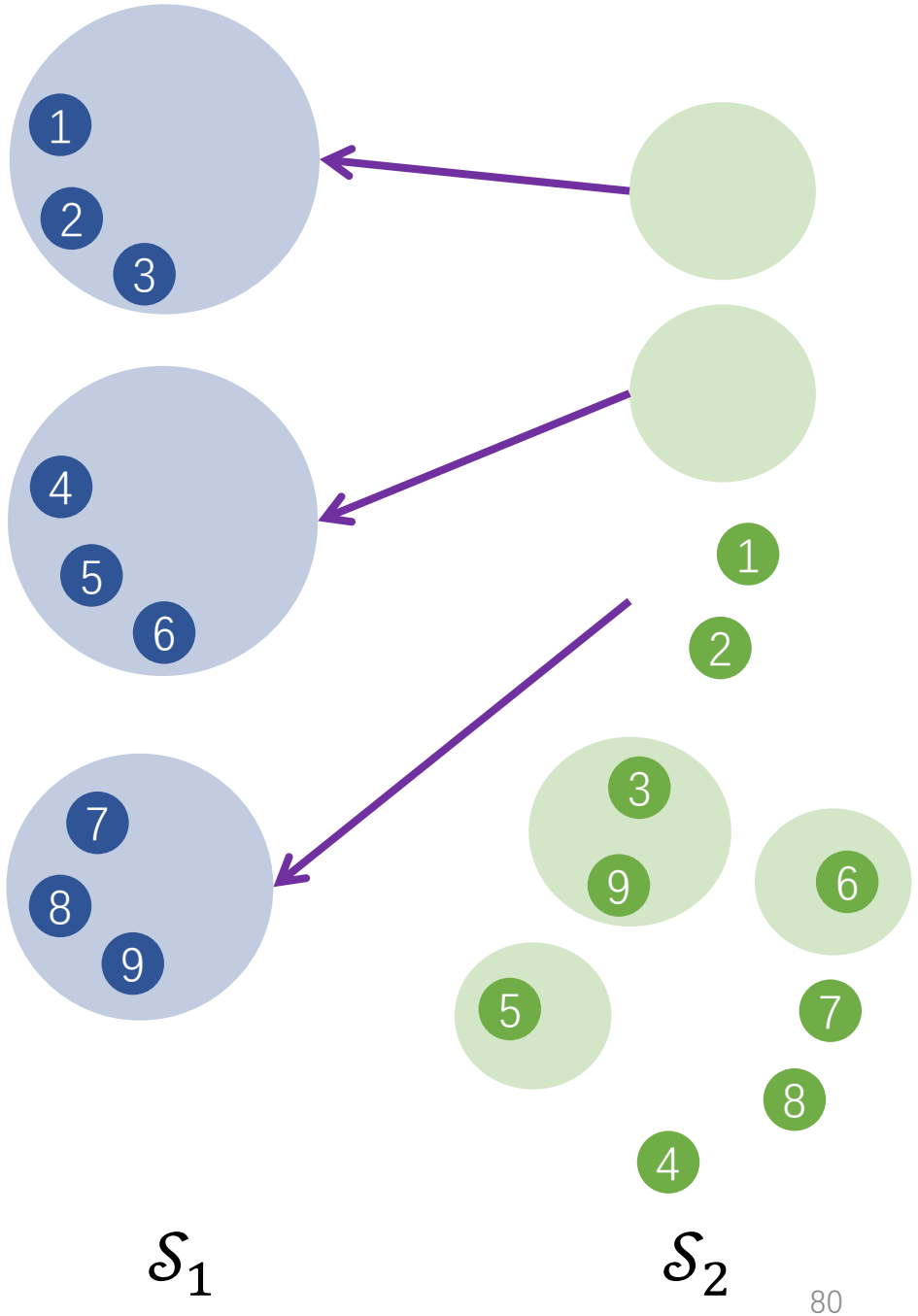
- **Case**
  - Reverse labels are pairwise different
- **Iterative algorithm**
  1. Remove $k_2 - k$ remaining $\mathcal{S}_2$-clusters
  2. For each bad $\mathcal{S}_1$-cluster (increasing size),
     - If majority of good clients' $\mathcal{S}_2$-clusters are not removed, keep the corresponding $\mathcal{S}_2$-cluster
     - Otherwise, remove the corresponding $\mathcal{S}_2$-cluster, but add the $\mathcal{S}_1$-cluster to final solution



Short note: we use 1/3 instead of the majority in the general case. Majority is enough for this simple case.

A Simple Case

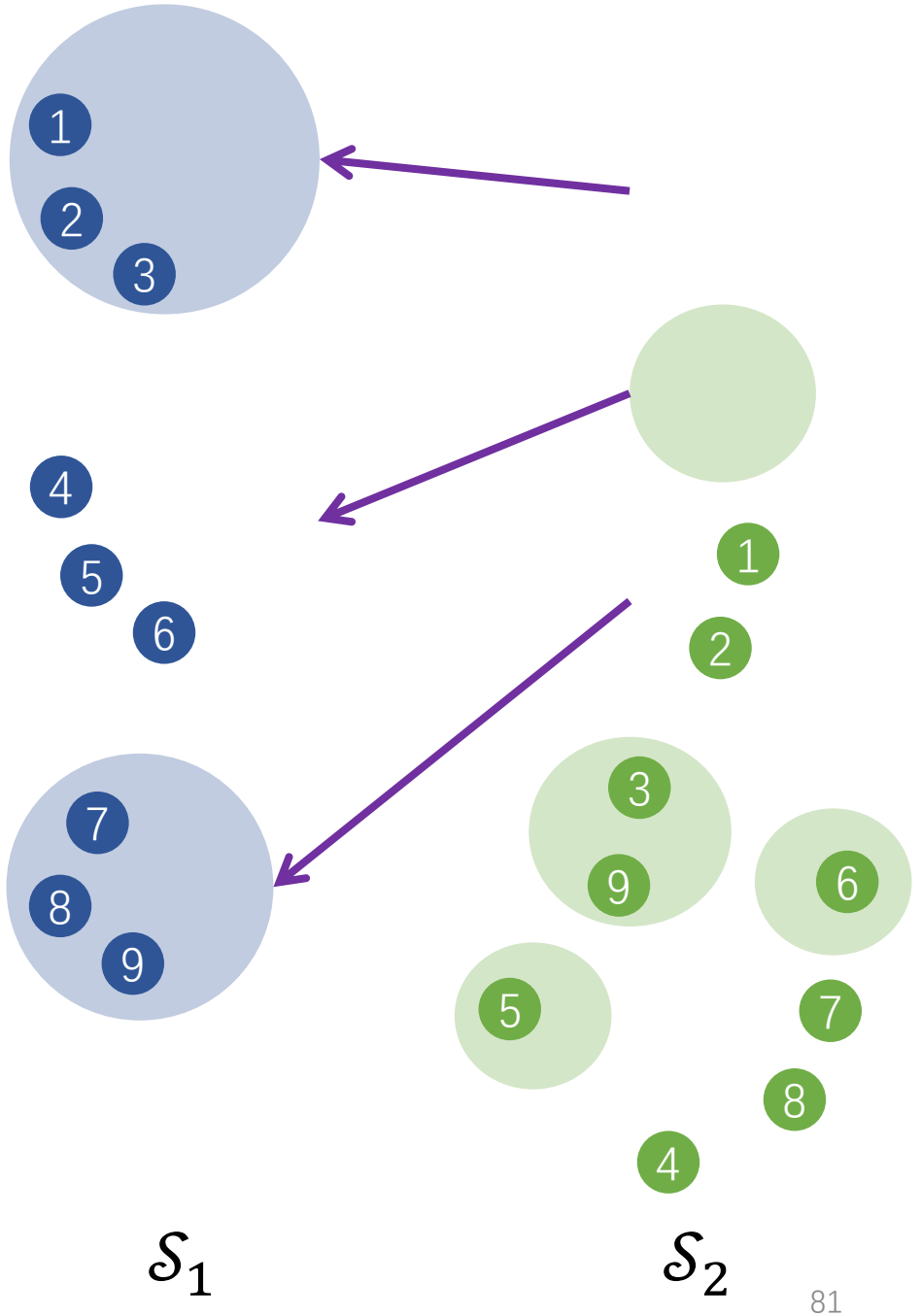
- **Case**
  - Reverse labels are pairwise different
- **Iterative algorithm**
  1. Remove $k_2 - k$ remaining $\mathcal{S}_2$-clusters
  2. For each bad $\mathcal{S}_1$-cluster (increasing size),
     - If majority of good clients' $\mathcal{S}_2$-clusters are not removed, keep the corresponding $\mathcal{S}_2$-cluster
     - Otherwise, remove the corresponding $\mathcal{S}_2$-cluster, but add the $\mathcal{S}_1$-cluster to final solution



Short note: we use 1/3 instead of the majority in the general case. Majority is enough for this simple case.

A Simple Case

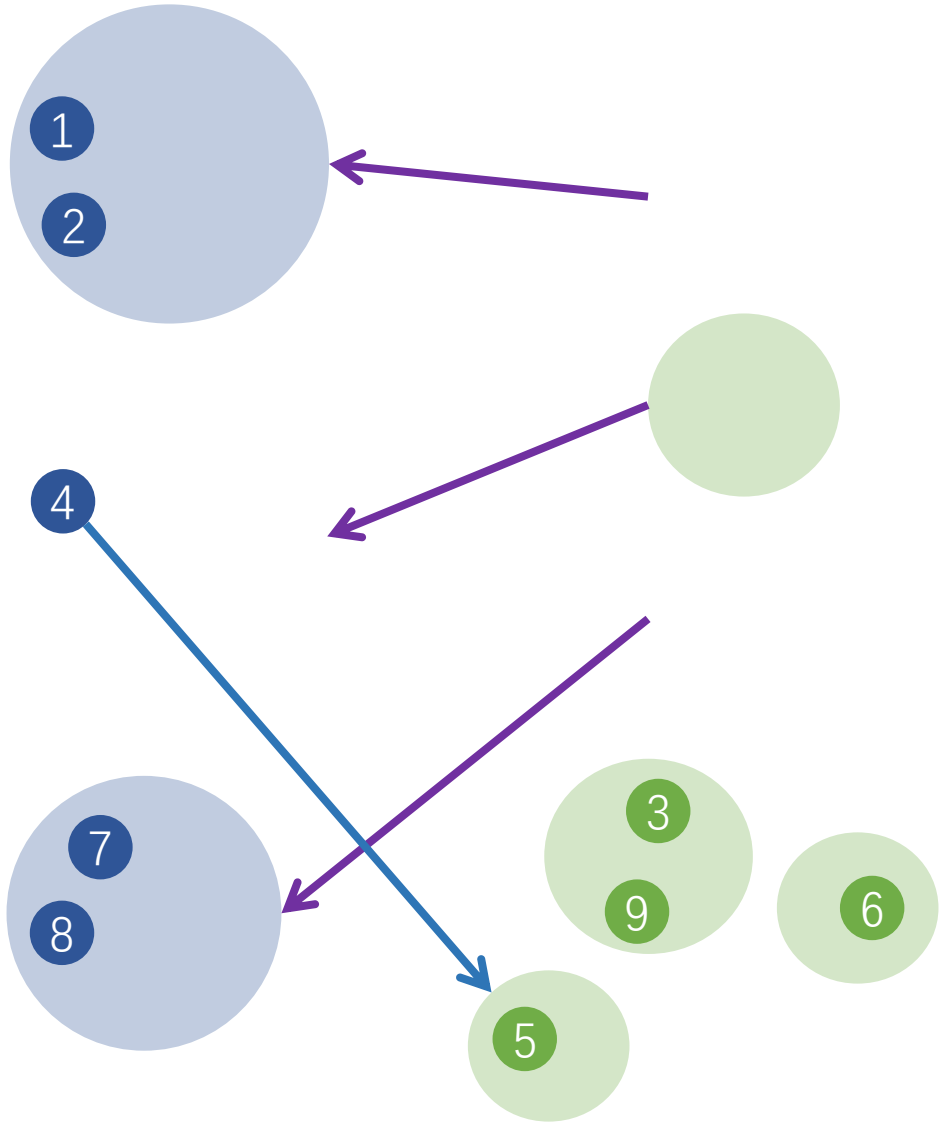
- **Case**
  - Reverse labels are pairwise different
- **Iterative algorithm**
  1. Remove $k_2 - k$ remaining $\mathcal{S}_2$-clusters
  2. For each bad $\mathcal{S}_1$-cluster (increasing size),
     - If majority of good clients' $\mathcal{S}_2$-clusters are not removed, keep the corresponding $\mathcal{S}_2$-cluster
     - Otherwise, remove the corresponding $\mathcal{S}_2$-cluster, but add the $\mathcal{S}_1$-cluster to final solution



Short note: we use 1/3 instead of the majority in the general case. Majority is enough for this simple case.

A Simple Case

- **Case**
  - Reverse labels are pairwise different
- **Iterative algorithm**
  1. Remove $k_2 - k$ remaining $\mathcal{S}_2$-clusters
  2. For each bad $\mathcal{S}_1$-cluster (increasing size),
     - If majority of good clients' $\mathcal{S}_2$-clusters are not removed, keep the corresponding $\mathcal{S}_2$-cluster
     - Otherwise, remove the corresponding $\mathcal{S}_2$-cluster, but add the $\mathcal{S}_1$-cluster to final solution

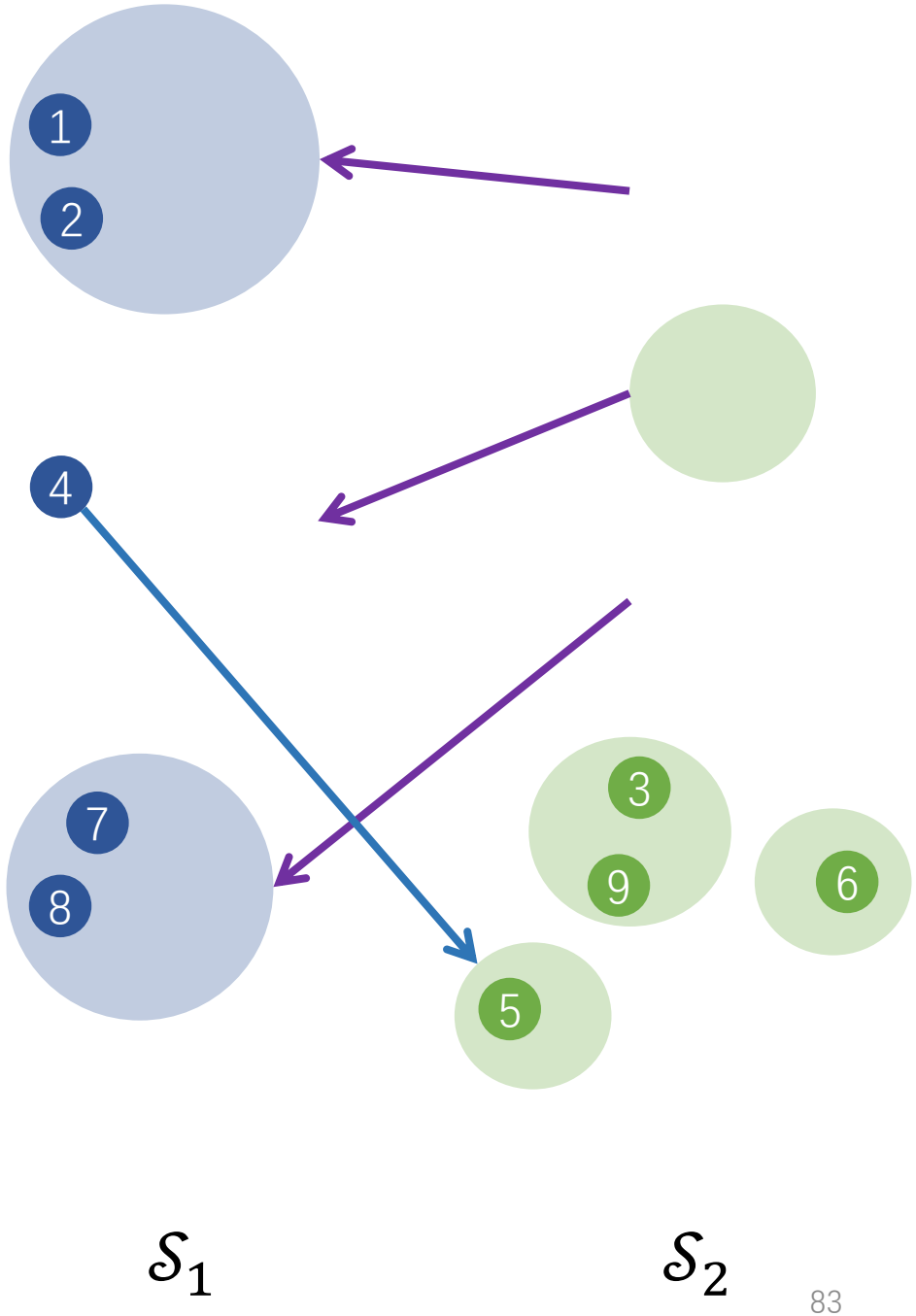


$\mathcal{S}_1$ $\mathcal{S}_2$

Short note: we use 1/3 instead of the majority in the general case. Majority is enough for this simple case.

A Simple Case

- **Case**
  - Reverse labels are pairwise different
- **Iterative algorithm**
  1. Remove $k_2 - k$ remaining $\mathcal{S}_2$-clusters
  2. For each bad $\mathcal{S}_1$-cluster (increasing size),
     - If majority of good clients' $\mathcal{S}_2$-clusters are not removed, keep the corresponding $\mathcal{S}_2$-cluster
     - Otherwise, remove the corresponding $\mathcal{S}_2$-cluster, but add the $\mathcal{S}_1$-cluster to final solution
- **Analysis**
  - Erdos-Ko-Rado theorem ensures for each $S \in \mathcal{S}_2$: $\Pr[S \text{ removed}] = O(\alpha)$



Short note: we use 1/3 instead of the majority in the general case. Majority is enough for this simple case.

Conclusions

Conclusions



- **$O(1)$-approximation for min-sum k-clustering**
  - Cluster splitting
  - Dual fitting with dependencies
  - Reverse labeling for bipoint rounding

  } **Handling the curse of size**

- **Open problems**
  - Small/simpler constant approximations?
    - Our approximation ratio is ~2b
    - Our algorithm is quite complicated
  - $O(1)$-approximation for capacitated k-median?
    - Best-known: $\mathbf{poly}(1/\epsilon)$-approximation that either
      - Opens $(1+\epsilon)k$ centers
      - Violates the capacities by $1+\epsilon$