Approximation Algorithms (ADM III)

6- The primal dual method

Guillaume Sagnol

Outline

Set Cover Problem

Given: A set of elements $E = \{e_1, \ldots, e_n\}$, a family of subsets $\{S_1, \ldots, S_m\} \subseteq 2^E$, and a weight $w_j \geq 0$ for each $j \in \{1, \ldots, m\}$.

Set Cover Problem

Given: A set of elements $E = \{e_1, \ldots, e_n\}$, a family of subsets $\{S_1, \ldots, S_m\} \subseteq 2^E$, and a weight $w_j \geq 0$ for each $j \in \{1, \ldots, m\}$.

Task: Find $I \subseteq \{1, \ldots, m\}$ minimizing $\sum_{j \in I} w_j$ such that $\bigcup_{j \in I} S_j = E$.

Set Cover Problem

Given: A set of elements $E = \{e_1, \ldots, e_n\}$, a family of subsets $\{S_1, \ldots, S_m\} \subseteq 2^E$, and a weight $w_j \geq 0$ for each $j \in \{1, \ldots, m\}$.

Task: Find $I \subseteq \{1, \ldots, m\}$ minimizing $\sum_{j \in I} w_j$ such that $\bigcup_{j \in I} S_j = E$.

LP relaxation:

$$\begin{array}{llll} \min & \displaystyle\sum_{j=1}^{m} w_j \cdot x_j & & \\ \text{s.t.} & \displaystyle\sum_{j:e_i \in S_j} x_j \geq 1 & & \text{for all } i = 1, \ldots, n \\ & x_j \geq 0 & & \text{for all } j = 1, \ldots, m \end{array}$$

Set Cover Problem

Given: A set of elements $E = \{e_1, \dots, e_n\}$, a family of subsets $\{S_1, \dots, S_m\} \subseteq 2^E$, and a weight $w_j \geq 0$ for each $j \in \{1, \dots, m\}$.

Task: Find $I \subseteq \{1, \dots, m\}$ minimizing $\sum_{j \in I} w_j$ such that $\bigcup_{j \in I} S_j = E$.

LP relaxation:

$$\begin{array}{llcll} \min & \sum_{j=1}^{m} w_j \cdot x_j & & & \\ \text{s.t.} & \sum_{j: e_i \in S_j} x_j & \geq & 1 & \text{for all } i = 1, \dots, n \\ & x_j & \geq & 0 & \text{for all } j = 1, \dots, m \end{array}$$

Dual LP:

$$\begin{array}{llcll} \max & \sum_{i=1}^{n} y_i & & & \\ \text{s.t.} & \sum_{e_i \in S_j} y_i & \leq & w_j & \text{for all } j = 1, \dots, m \\ & y_i & \geq & 0 & \text{for all } i = 1, \dots, n \end{array}$$

Primal-Dual Algorithm Set Cover (see Ch. 1)

set $y :\equiv 0$ and $I := \emptyset$;

while $\exists e_k \notin \bigcup_{j \in I} S_j$

    increase y_k until $\exists j$ with $e_k \in S_j$ such that $\sum_{i : e_i \in S_j} y_i = w_j$;

    set $I := I \cup \{j\}$;

Primal-Dual Algorithm Set Cover (see Ch. 1)

1 set $y :\equiv 0$ and $I := \emptyset$;
2 while $\exists e_k \notin \bigcup_{j \in I} S_j$
3   increase y_k until $\exists j$ with $e_k \in S_j$ such that $\sum_{i : e_i \in S_j} y_i = w_j$;
4   set $I := I \cup \{j\}$;

Theorem 6.1 (recap of Theorem 1.10)

The primal-dual algorithm is an f-approximation algorithm for the Set Cover Problem where $f := \max_{i=1,\ldots,n} |\{j \mid e_i \in S_j\}|$.

Primal-Dual Algorithm Set Cover (see Ch. 1)

1. set $y :\equiv 0$ and $I := \emptyset$;
2. while $\exists e_k \notin \bigcup_{j \in I} S_j$
3. increase y_k until $\exists j$ with $e_k \in S_j$ such that $\sum_{i: e_i \in S_j} y_i = w_j$;
4. set $I := I \cup \{j\}$;

Theorem 6.1 (recap of Theorem 1.10)

The primal-dual algorithm is an f-approximation algorithm for the Set Cover Problem where $f := \max_{i=1,\ldots,n} |\{j \mid e_i \in S_j\}|$.

Proof:

$$\sum_{j \in I} w_j = \sum_{j \in I} \sum_{i: e_i \in S_j} y_i = \sum_{i=1}^{n} y_i \cdot |\{j \in I \mid e_i \in S_j\}|$$

$$\leq f \cdot \sum_{i=1}^{n} y_i \leq f \cdot \mathtt{OPT}_{LP} \leq f \cdot \mathtt{OPT}$$

□

Approximate Complementary Slackness

Remark. The pair of feasible solutions (x, y) to the primal and dual LP found by the algorithm satisfies

$$x_j > 0 \quad \Longrightarrow \quad \sum_{e_i \in S_j} y_i = w_j \qquad (\textit{compl. slackness})$$

$$y_i > 0 \quad \Longrightarrow \quad \sum_{j : e_i \in S_j} x_j \leq f \qquad (\textit{approx. compl. slackness})$$

Approximate Complementary Slackness

Remark. The pair of feasible solutions (x, y) to the primal and dual LP found by the algorithm satisfies

$$x_j > 0 \quad \Longrightarrow \quad \sum_{e_i \in S_j} y_i = w_j \qquad (\mathit{compl.\ slackness})$$

$$y_i > 0 \quad \Longrightarrow \quad \sum_{j: e_i \in S_j} x_j \leq f \qquad (\mathit{approx.\ compl.\ slackness})$$

The analysis on the previous slide only relies on these two properties!

Outline

Feedback Vertex Set Problem

Given: Undirected graph $G = (V, E)$ with node weights $w_i \geq 0$, $i \in V$.

Task: Find $S \subseteq V$ minimizing $\sum_{i \in S} w_i$ such that $G[V \setminus S]$ is acyclic.

Feedback Vertex Set Problem

Given: Undirected graph $G = (V, E)$ with node weights $w_i \geq 0$, $i \in V$.

Task: Find $S \subseteq V$ minimizing $\sum_{i \in S} w_i$ such that $G[V \setminus S]$ is acyclic.

Integer programming formulation: Let $\mathcal{C}$ denote the set of all cycles in G.

$$\begin{aligned} \min \quad & \sum_{i \in V} w_i \cdot x_i & \\ \text{s.t.} \quad & \sum_{i \in C} x_i \geq 1 & \text{for all } C \in \mathcal{C}, \\ & x_i \in \{0, 1\} & \text{for all } i \in V. \end{aligned}$$

Feedback Vertex Set Problem

Given: Undirected graph $G = (V, E)$ with node weights $w_i \geq 0$, $i \in V$.

Task: Find $S \subseteq V$ minimizing $\sum_{i \in S} w_i$ such that $G[V \setminus S]$ is acyclic.

Integer programming formulation: Let $\mathcal{C}$ denote the set of all cycles in G.

$$\begin{aligned}
\min \quad & \sum_{i \in V} w_i \cdot x_i & \\
\text{s.t.} \quad & \sum_{i \in C} x_i \geq 1 & \text{for all } C \in \mathcal{C}, \\
& x_i \in \{0, 1\} & \text{for all } i \in V.
\end{aligned}$$

Dual of LP relaxation ($x \geq 0$):

$$\begin{aligned}
\max \quad & \sum_{C \in \mathcal{C}} y_C & \\
\text{s.t.} \quad & \sum_{C \in \mathcal{C}:\, i \in C} y_C \leq w_i & \text{for all } i \in V, \\
& y_C \geq 0 & \text{for all } C \in \mathcal{C}.
\end{aligned}$$

Primal-Dual Algo for Feedback Vertex Set Problem

1. set $y := 0$ and $S := \emptyset$;
2. while there is a cycle C in G
3. increase y_C until there is an $i \in C$ with $\sum_{C' : i \in C'} y_{C'} = w_i$;
4. set $S := S \cup \{i\}$ and delete i from G;
5. repeatedly remove nodes of degree one from G;

Primal-Dual Algo for Feedback Vertex Set Problem

1. set $y := 0$ and $S := \emptyset$;
2. while there is a cycle C in G
3. increase y_C until there is an $i \in C$ with $\sum_{C': i \in C'} y_{C'} = w_i$;
4. set $S := S \cup \{i\}$ and delete i from G;
5. repeatedly remove nodes of degree one from G;

Analysis:

$$\sum_{i \in S} w_i = \sum_{i \in S} \sum_{C:\, i \in C} y_C = \sum_{C \in \mathcal{C}} |S \cap C| \cdot y_C$$

Primal-Dual Algo for Feedback Vertex Set Problem

1. set $y := 0$ and $S := \emptyset$;
2. while there is a cycle C in G
3. increase y_C until there is an $i \in C$ with $\sum_{C' : i \in C'} y_{C'} = w_i$;
4. set $S := S \cup \{i\}$ and delete i from G;
5. repeatedly remove nodes of degree one from G;

Analysis:

$$\sum_{i \in S} w_i = \sum_{i \in S} \sum_{C:\, i \in C} y_C = \sum_{C \in \mathcal{C}} |S \cap C| \cdot y_C$$

- Idea: If $|S \cap C| \leq \alpha$ whenever $y_C > 0$, we get performance ratio α.

Primal-Dual Algo for Feedback Vertex Set Problem

1. set $y := 0$ and $S := \emptyset$;
2. while there is a cycle C in G
3. increase y_C until there is an $i \in C$ with $\sum_{C' : i \in C'} y_{C'} = w_i$;
4. set $S := S \cup \{i\}$ and delete i from G;
5. repeatedly remove nodes of degree one from G;

Analysis:

$$\sum_{i \in S} w_i = \sum_{i \in S} \sum_{C :\, i \in C} y_C = \sum_{C \in \mathcal{C}} |S \cap C| \cdot y_C$$

- Idea: If $|S \cap C| \leq \alpha$ whenever $y_C > 0$, we get performance ratio α.
- But: If we choose arbitrary C in each iteration, $|S \cap C|$ can be large.

Primal-Dual Algo for Feedback Vertex Set Problem

1. set $y := 0$ and $S := \emptyset$;
2. while there is a cycle C in G
3. increase y_C until there is an $i \in C$ with $\sum_{C': i \in C'} y_{C'} = w_i$;
4. set $S := S \cup \{i\}$ and delete i from G;
5. repeatedly remove nodes of degree one from G;

Analysis:

$$\sum_{i \in S} w_i = \sum_{i \in S} \sum_{C:\, i \in C} y_C = \sum_{C \in \mathcal{C}} |S \cap C| \cdot y_C$$

- Idea: If $|S \cap C| \leq \alpha$ whenever $y_C > 0$, we get performance ratio α.
- But: If we choose arbitrary C in each iteration, $|S \cap C|$ can be large.
- Idea: Always choose short cycle C with $|C| \leq \alpha$.

Primal-Dual Algo for Feedback Vertex Set Problem

1. set $y := 0$ and $S := \emptyset$;
2. while there is a cycle C in G
3. increase y_C until there is an $i \in C$ with $\sum_{C': i \in C'} y_{C'} = w_i$;
4. set $S := S \cup \{i\}$ and delete i from G;
5. repeatedly remove nodes of degree one from G;

Analysis:

$$\sum_{i \in S} w_i = \sum_{i \in S} \sum_{C:\, i \in C} y_C = \sum_{C \in \mathcal{C}} |S \cap C| \cdot y_C$$

- Idea: If $|S \cap C| \leq \alpha$ whenever $y_C > 0$, we get performance ratio α.
- But: If we choose arbitrary C in each iteration, $|S \cap C|$ can be large.
- Idea: Always choose short cycle C with $|C| \leq \alpha$.
- But: This is not always possible (e.g., if graph is one large cycle).
- Obs.: From path of nodes of degree two, algorithm chooses ≤ 1 node.

Refined Primal-Dual Algorithm for Feedback Vertex Set

Lemma 6.2

In any graph G that has no nodes of degree one, there is a cycle with $\leq 2\lceil \log_2 n \rceil$ nodes of degree 3 or more, and it can be found in linear time.

Refined Primal-Dual Algorithm for Feedback Vertex Set

Lemma 6.2

In any graph G that has no nodes of degree one, there is a cycle with $\leq 2\lceil \log_2 n \rceil$ nodes of degree 3 or more, and it can be found in linear time.

Proof:… □

Refined Primal-Dual Algorithm for Feedback Vertex Set

Lemma 6.2

In any graph G that has no nodes of degree one, there is a cycle with $\leq 2\lceil \log_2 n \rceil$ nodes of degree 3 or more, and it can be found in linear time.

Proof:… $\square$

Theorem 6.3

If the primal-dual algorithm chooses in each iteration a cycle with at most $\leq 2\lceil \log_2 n \rceil$ nodes of degree 3 or more, it has performance ratio $4\lceil \log_2 n \rceil$.

Refined Primal-Dual Algorithm for Feedback Vertex Set

Lemma 6.2

In any graph G that has no nodes of degree one, there is a cycle with $\leq 2\lceil \log_2 n \rceil$ nodes of degree 3 or more, and it can be found in linear time.

Proof:… $\square$

Theorem 6.3

If the primal-dual algorithm chooses in each iteration a cycle with at most $\leq 2\lceil \log_2 n \rceil$ nodes of degree 3 or more, it has performance ratio $4\lceil \log_2 n \rceil$.

Proof:… $\square$

Refined Primal-Dual Algorithm for Feedback Vertex Set

Lemma 6.2

In any graph G that has no nodes of degree one, there is a cycle with $\leq 2\lceil \log_2 n \rceil$ nodes of degree 3 or more, and it can be found in linear time.

Proof:… □

Theorem 6.3

If the primal-dual algorithm chooses in each iteration a cycle with at most $\leq 2\lceil \log_2 n \rceil$ nodes of degree 3 or more, it has performance ratio $4\lceil \log_2 n \rceil$.

Proof:… □

Remarks.

- The LP relaxation has an integrality gap of $\Omega(\log n)$.
- There is a primal-dual 2-approximation algorithm based on a more sophisticated integer programming formulation

Outline

Shortest s-t-Path Problem

Given: Undir. graph $G = (V, E)$ with edge costs $c_e \geq 0$, $e \in E$; $s, t \in V$

Task: Find minimum-cost s-t-path.

Shortest s-t-Path Problem

Given: Undir. graph $G = (V, E)$ with edge costs $c_e \geq 0$, $e \in E$; $s, t \in V$

Task: Find minimum-cost s-t-path.

IP formulation: (let $\mathcal{S} := \{S \subseteq V \mid s \in S,\ t \in V \setminus S\}$)

$$
\begin{aligned}
\min \quad & \sum_{e \in E} c_e \cdot x_e \\
\text{s.t.} \quad & \sum_{e \in \delta(S)} x_e \geq 1 && \text{for all } S \in \mathcal{S}, \\
& x_e \in \{0, 1\} && \text{for all } e \in E.
\end{aligned}
$$

Shortest s-t-Path Problem

Given: Undir. graph $G = (V, E)$ with edge costs $c_e \geq 0$, $e \in E$; $s, t \in V$

Task: Find minimum-cost s-t-path.

IP formulation: (let $\mathcal{S} := \{S \subseteq V \mid s \in S,\ t \in V \setminus S\}$)

$$\begin{aligned} \min \quad & \sum_{e \in E} c_e \cdot x_e & \\ \text{s.t.} \quad & \sum_{e \in \delta(S)} x_e \geq 1 & \text{for all } S \in \mathcal{S}, \\ & x_e \in \{0, 1\} & \text{for all } e \in E. \end{aligned}$$

Dual of LP relaxation ($x \geq 0$):

$$\begin{aligned} \max \quad & \sum_{S \in \mathcal{S}} y_S & \\ \text{s.t.} \quad & \sum_{S \in \mathcal{S}\colon e \in \delta(S)} y_S \leq c_e & \text{for all } e \in E, \\ & y_S \geq 0 & \text{for all } S \in \mathcal{S}. \end{aligned}$$

Primal-Dual Algorithm for Shortest s-t-Path Problem

```
set y := 0 and F := ∅;
while there is no s-t-path in F
    let C be the connected component of (V, F) containing s;
    increase y_C until there is an e ∈ δ(C) with ∑_{S∈𝒮: e∈δ(S)} y_S = c_e;
    set F := F ∪ {e};
delete edges from F that do not lie on s-t-path in F;
```

Primal-Dual Algorithm for Shortest s-t-Path Problem

```
set y := 0 and F := ∅;
while there is no s-t-path in F
    let C be the connected component of (V, F) containing s;
    increase y_C until there is an e ∈ δ(C) with Σ_{S∈𝒮: e∈δ(S)} y_S = c_e;
    set F := F ∪ {e};
delete edges from F that do not lie on s-t-path in F;
```

Lemma 6.4

Throughout the algorithm, the set of edges in F always forms a tree containing node s. □

Primal-Dual Algorithm for Shortest s-t-Path Problem

1 set $y := 0$ and $F := \emptyset$;
2 while there is no s-t-path in F
3 let C be the connected component of (V, F) containing s;
4 increase y_C until there is an $e \in \delta(C)$ with $\sum_{S \in \mathcal{S}\colon e \in \delta(S)} y_S = c_e$;
5 set $F := F \cup \{e\}$;
6 delete edges from F that do not lie on s-t-path in F;

Lemma 6.4

Throughout the algorithm, the set of edges in F always forms a tree containing node s. □

Theorem 6.5

The algorithm finds a shortest s-t-path.

Outline

Steiner Forest Problem

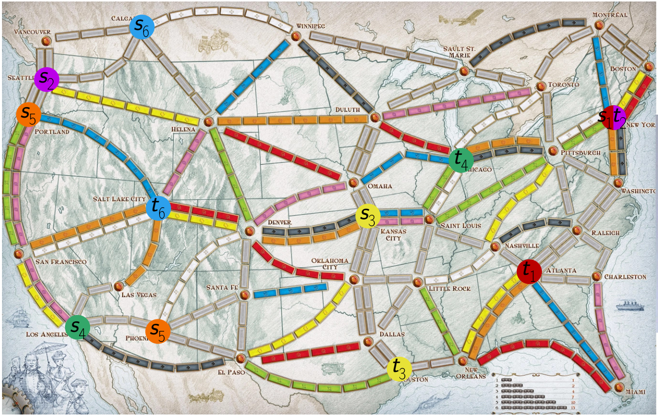
Given: Graph $G = (V, E)$ with costs $c_e \geq 0$, $e \in E$; k pairs $s_i, t_i \in V$.

Task: Find $F \subseteq E$ minimizing $c(F)$ and connecting s_i and t_i, for all i.

Steiner Forest Problem

Given: Graph $G = (V, E)$ with costs $c_e \geq 0$, $e \in E$; k pairs $s_i, t_i \in V$.

Task: Find $F \subseteq E$ minimizing $c(F)$ and connecting s_i and t_i, for all i.



Steiner Forest Problem

Given: Graph $G = (V, E)$ with costs $c_e \geq 0$, $e \in E$; k pairs $s_i, t_i \in V$.

Task: Find $F \subseteq E$ minimizing $c(F)$ and connecting s_i and t_i, for all i.

IP formulation: (let $\mathcal{S}_i := \{S \subseteq V \mid |S \cap \{s_i, t_i\}| = 1\}$ and $\mathcal{S} := \bigcup_{i=1}^{k} \mathcal{S}_i$)

$$
\begin{aligned}
\min \quad & \sum_{e \in E} c_e \cdot x_e & \\
\text{s.t.} \quad & \sum_{e \in \delta(S)} x_e \geq 1 & \text{for all } S \in \mathcal{S}, \\
& x_e \in \{0, 1\} & \text{for all } e \in E.
\end{aligned}
$$

Steiner Forest Problem

Given: Graph $G = (V, E)$ with costs $c_e \geq 0$, $e \in E$; k pairs $s_i, t_i \in V$.

Task: Find $F \subseteq E$ minimizing $c(F)$ and connecting s_i and t_i, for all i.

IP formulation: (let $\mathcal{S}_i := \{S \subseteq V \mid |S \cap \{s_i, t_i\}| = 1\}$ and $\mathcal{S} := \bigcup_{i=1}^{k} \mathcal{S}_i$)

$$\begin{aligned}
\min \quad & \sum_{e \in E} c_e \cdot x_e & \\
\text{s.t.} \quad & \sum_{e \in \delta(S)} x_e \geq 1 & \text{for all } S \in \mathcal{S}, \\
& x_e \in \{0, 1\} & \text{for all } e \in E.
\end{aligned}$$

Dual of LP relaxation ($x \geq 0$):

$$\begin{aligned}
\max \quad & \sum_{S \in \mathcal{S}} y_S & \\
\text{s.t.} \quad & \sum_{S \in \mathcal{S}:\, e \in \delta(S)} y_S \leq c_e & \text{for all } e \in E, \\
& y_S \geq 0 & \text{for all } S \in \mathcal{S}.
\end{aligned}$$

Primal-Dual Algorithm for Steiner Forest Problem

set $y :\equiv 0$ and $F := \emptyset$;
while not all s_i-t_i pairs are connected in (V, F)
    let C connected comp. of (V, F) with $|C \cap \{s_i, t_i\}| = 1$ for some i;
    increase y_C until there is an $e \in \delta(C)$ with $\sum_{S \in \mathcal{S}:\, e \in \delta(S)} y_S = c_e$;
    set $F := F \cup \{e\}$;

Primal-Dual Algorithm for Steiner Forest Problem

1. set $y :\equiv 0$ and $F := \emptyset$;
2. while not all s_i-t_i pairs are connected in (V, F)
3. let C connected comp. of (V, F) with $|C \cap \{s_i, t_i\}| = 1$ for some i;
4. increase y_C until there is an $e \in \delta(C)$ with $\displaystyle\sum_{S \in \mathcal{S}:\, e \in \delta(S)} y_S = c_e$;
5. set $F := F \cup \{e\}$;

Analysis: $$\sum_{e \in F} c_e = \sum_{e \in F} \sum_{S \in \mathcal{S}:\, e \in \delta(S)} y_S = \sum_{S \in \mathcal{S}} |\delta(S) \cap F| \cdot y_S$$

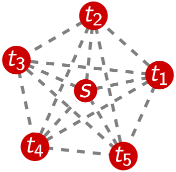
Primal-Dual Algorithm for Steiner Forest Problem

1. set $y :\equiv 0$ and $F := \emptyset$;
2. while not all s_i-t_i pairs are connected in (V, F)
3. let C connected comp. of (V, F) with $|C \cap \{s_i, t_i\}| = 1$ for some i;
4. increase y_C until there is an $e \in \delta(C)$ with $\sum_{S \in \mathcal{S}:\, e \in \delta(S)} y_S = c_e$;
5. set $F := F \cup \{e\}$;

Analysis: $$\sum_{e \in F} c_e = \sum_{e \in F} \sum_{S \in \mathcal{S}:\, e \in \delta(S)} y_S = \sum_{S \in \mathcal{S}} |\delta(S) \cap F| \cdot y_S$$

Problem: It can happen that $|\delta(S) \cap F| = k$ for $y_S > 0$ and $\sum_{S \in \mathcal{S}} y_S \leq \frac{1}{k}\mathtt{OPT}$:

- $G = K_{k+1}$ (complete graph)
- $s_i := s$ for $i = 1, \dots, k$
- $c_e := 1$ for all $e \in E$



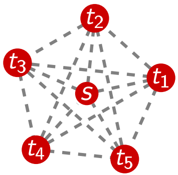
Primal-Dual Algorithm for Steiner Forest Problem

1. set $y :\equiv 0$ and $F := \emptyset$;
2. while not all s_i-t_i pairs are connected in (V, F)
3. let C connected comp. of (V, F) with $|C \cap \{s_i, t_i\}| = 1$ for some i;
4. increase y_C until there is an $e \in \delta(C)$ with $\sum_{S \in \mathcal{S}:\, e \in \delta(S)} y_S = c_e$;
5. set $F := F \cup \{e\}$;

Analysis: $$\sum_{e \in F} c_e = \sum_{e \in F} \sum_{S \in \mathcal{S}:\, e \in \delta(S)} y_S = \sum_{S \in \mathcal{S}} |\delta(S) \cap F| \cdot y_S$$

Problem: It can happen that $|\delta(S) \cap F| = k$ for $y_S > 0$ and $\sum_{S \in \mathcal{S}} y_S \leq \frac{1}{k}\texttt{OPT}$:

- $G = K_{k+1}$ (complete graph)
- $s_i := s$ for $i = 1, \ldots, k$
- $c_e := 1$ for all $e \in E$



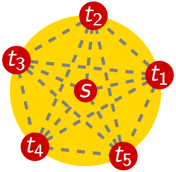
Primal-Dual Algorithm for Steiner Forest Problem

1. set $y :\equiv 0$ and $F := \emptyset$;
2. while not all s_i-t_i pairs are connected in (V, F)
3. let C connected comp. of (V, F) with $|C \cap \{s_i, t_i\}| = 1$ for some i;
4. increase y_C until there is an $e \in \delta(C)$ with $\sum_{S \in \mathcal{S}:\, e \in \delta(S)} y_S = c_e$;
5. set $F := F \cup \{e\}$;

Analysis: $$\sum_{e \in F} c_e = \sum_{e \in F} \sum_{S \in \mathcal{S}:\, e \in \delta(S)} y_S = \sum_{S \in \mathcal{S}} |\delta(S) \cap F| \cdot y_S$$

Problem: It can happen that $|\delta(S) \cap F| = k$ for $y_S > 0$ and $\sum_{S \in \mathcal{S}} y_S \leq \frac{1}{k}\texttt{OPT}$:

- $G = K_{k+1}$ (complete graph)
- $s_i := s$ for $i = 1, \dots, k$
- $c_e := 1$ for all $e \in E$



- $y_{\{s\}} = 1$

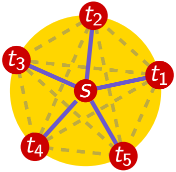
Primal-Dual Algorithm for Steiner Forest Problem

1. set $y :\equiv 0$ and $F := \emptyset$;
2. while not all s_i-t_i pairs are connected in (V, F)
3. let C connected comp. of (V, F) with $|C \cap \{s_i, t_i\}| = 1$ for some i;
4. increase y_C until there is an $e \in \delta(C)$ with $\sum_{S \in \mathcal{S}:\, e \in \delta(S)} y_S = c_e$;
5. set $F := F \cup \{e\}$;

Analysis: $$\sum_{e \in F} c_e = \sum_{e \in F} \sum_{S \in \mathcal{S}:\, e \in \delta(S)} y_S = \sum_{S \in \mathcal{S}} |\delta(S) \cap F| \cdot y_S$$

Problem: It can happen that $|\delta(S) \cap F| = k$ for $y_S > 0$ and $\sum_{S \in \mathcal{S}} y_S \leq \frac{1}{k}\mathtt{OPT}$:

- $G = K_{k+1}$ (complete graph)
- $s_i := s$ for $i = 1, \ldots, k$
- $c_e := 1$ for all $e \in E$



- $y_{\{s\}} = 1$
- $|\delta(\{s\}) \cap F| = k$

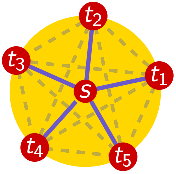
Primal-Dual Algorithm for Steiner Forest Problem

1. set $y :\equiv 0$ and $F := \emptyset$;
2. while not all s_i-t_i pairs are connected in (V, F)
3. let C connected comp. of (V, F) with $|C \cap \{s_i, t_i\}| = 1$ for some i;
4. increase y_C until there is an $e \in \delta(C)$ with $\sum_{S \in \mathcal{S}:\, e \in \delta(S)} y_S = c_e$;
5. set $F := F \cup \{e\}$;

Analysis: $$\sum_{e \in F} c_e = \sum_{e \in F} \sum_{S \in \mathcal{S}:\, e \in \delta(S)} y_S = \sum_{S \in \mathcal{S}} |\delta(S) \cap F| \cdot y_S$$

Problem: It can happen that $|\delta(S) \cap F| = k$ for $y_S > 0$ and $\sum_{S \in \mathcal{S}} y_S \leq \frac{1}{k}\texttt{OPT}$:

- $G = K_{k+1}$ (complete graph)
- $s_i := s$ for $i = 1, \dots, k$
- $c_e := 1$ for all $e \in E$



- $y_{\{s\}} = 1$
- $|\delta(\{s\}) \cap F| = k$
- $\sum_S y_S = 1$, $\texttt{OPT} = k$

Refined Primal-Dual Algo for Steiner Forest Problem

set $y :\equiv 0$ and $F := \emptyset$;

while not all s_i-t_i pairs are connected in (V, F)

  let $\mathcal{C} := \{$conn. comp. C of (V, F): $|C \cap \{s_i, t_i\}| = 1$ for some $i\}$;

  increase y_C for all $C \in \mathcal{C}$ uniformly until for some $e \in \delta(C)$, $C \in \mathcal{C}$

$$\sum_{S \in \mathcal{S}:\, e \in \delta(S)} y_S = c_e \quad ;$$

  set $F := F \cup \{e\}$;

Refined Primal-Dual Algo for Steiner Forest Problem

set $y :\equiv 0$ and $F := \emptyset$;

while not all s_i-t_i pairs are connected in (V, F)

    let $\mathcal{C} := \{\text{conn. comp. } C \text{ of } (V, F)\colon |C \cap \{s_i, t_i\}| = 1 \text{ for some } i\}$;

    increase y_C for all $C \in \mathcal{C}$ uniformly until for some $e \in \delta(C)$, $C \in \mathcal{C}$

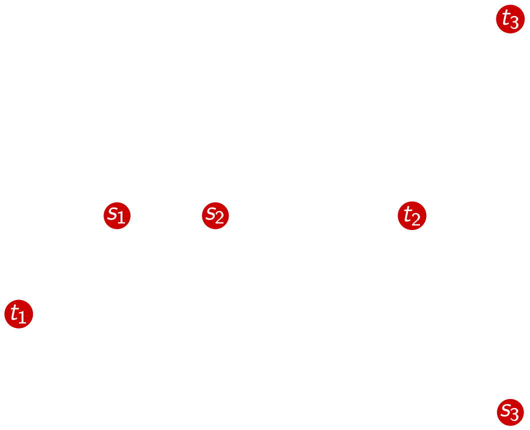
$$\sum_{S \in \mathcal{S}\colon e \in \delta(S)} y_S = c_e \quad ;$$

    set $F := F \cup \{e\}$;

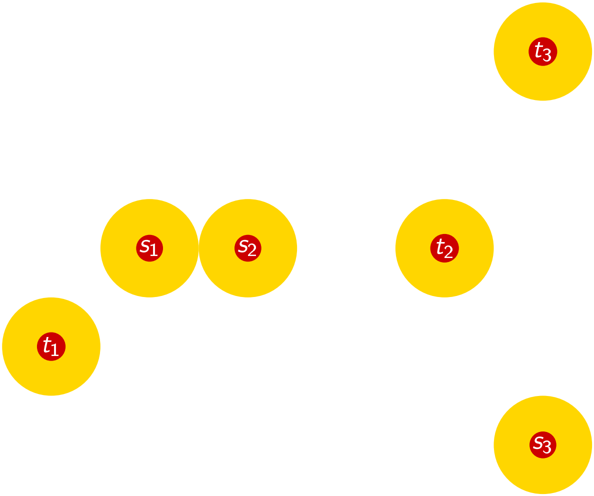
For all $e \in F$ (in reverse of the order in which they were added)

    if $F \setminus \{e\}$ is a feasible solution, then remove e from F;

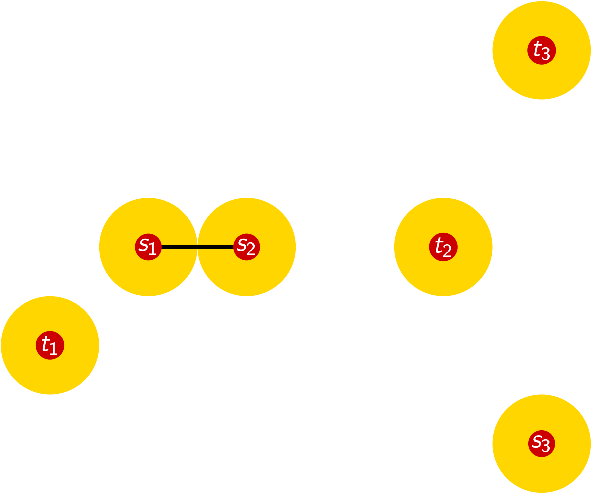
Example



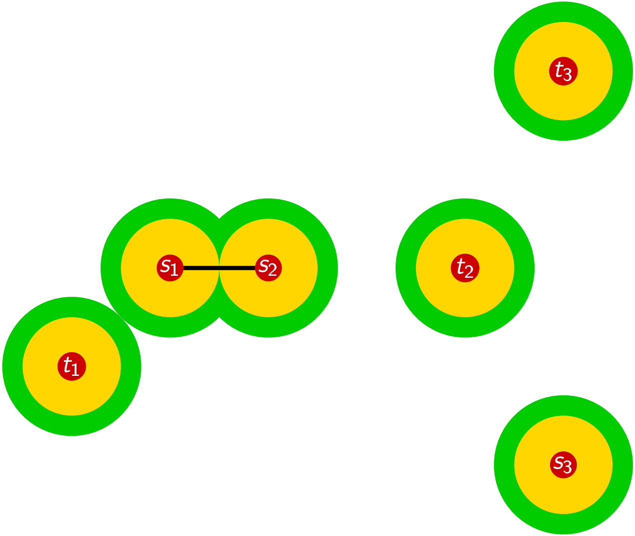
Example



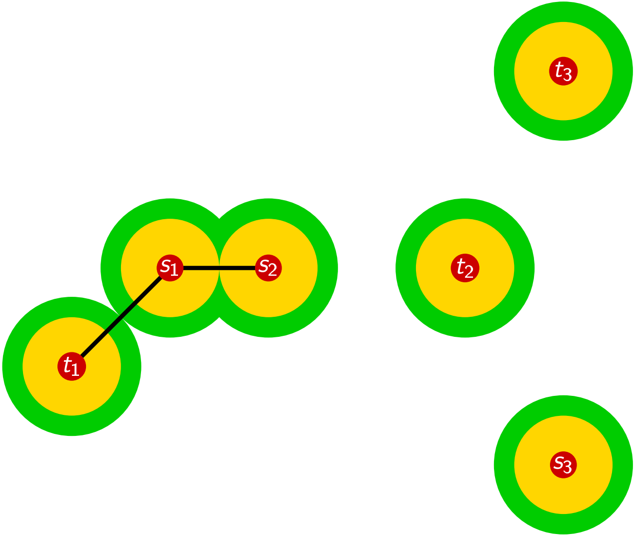
Example



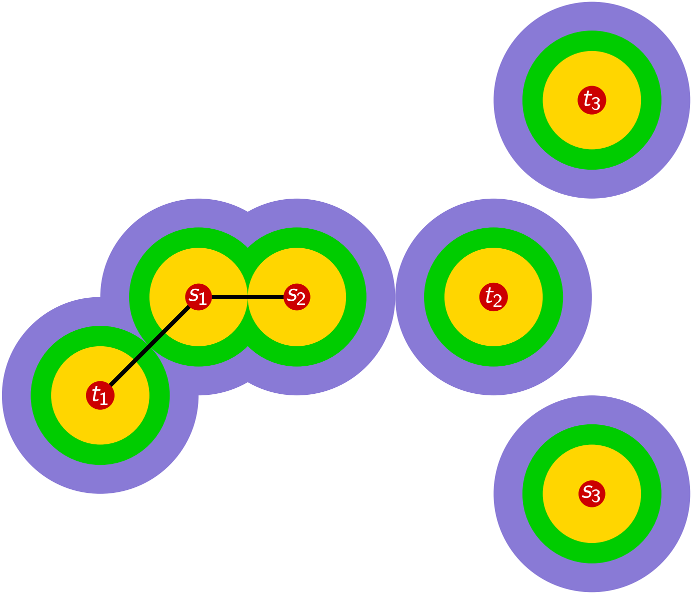
Example



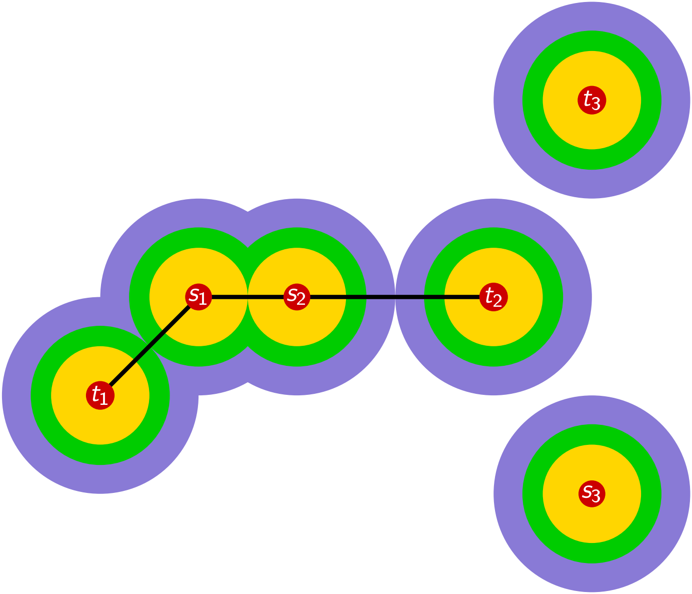
Example



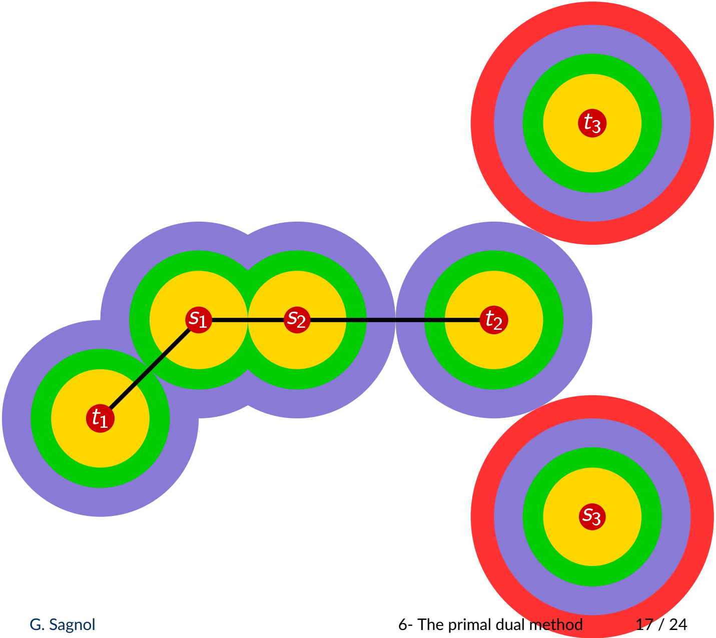
Example



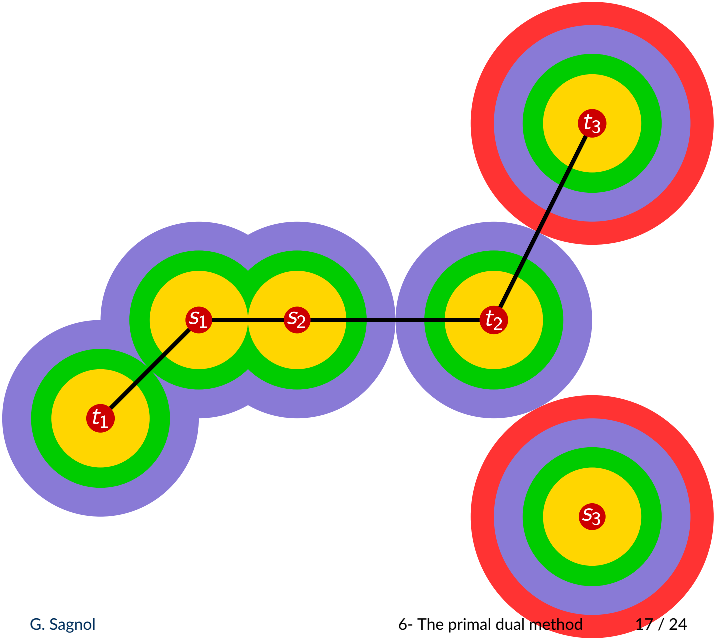
Example



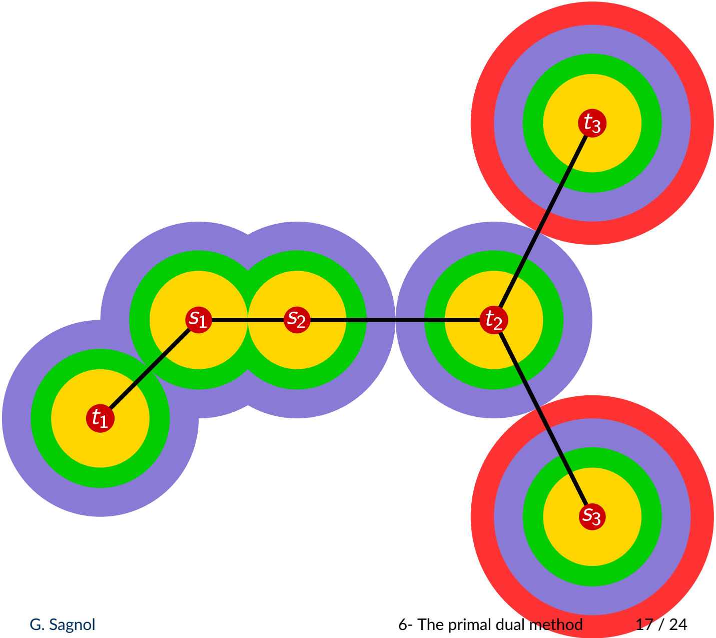
Example



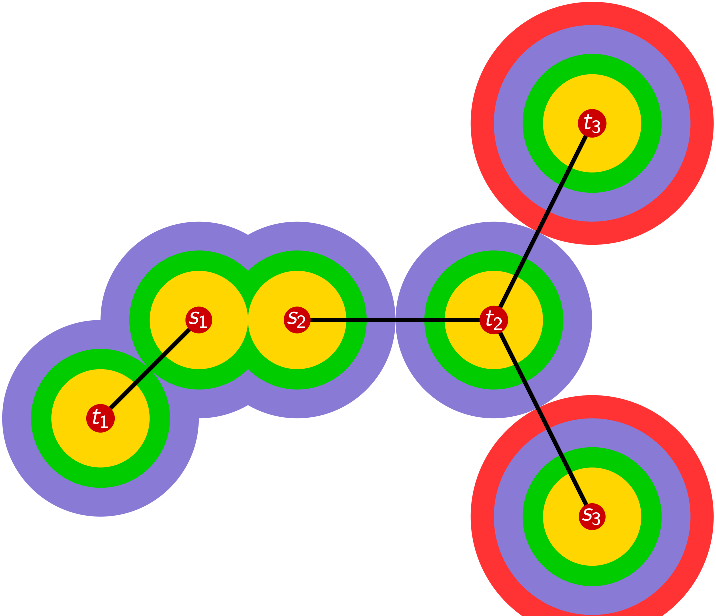
Example



Example



Example



Analysis

Observation. At any point in the algorithm, the set of edges F is a forest.

Analysis

Observation. At any point in the algorithm, the set of edges F is a forest.

Lemma 6.6

Let F' be the final set of edges returned by the algorithm. For any $\mathcal{C}$ in any iteration of the algorithm,

$$\sum_{C \in \mathcal{C}} |\delta(C) \cap F'| \leq 2|\mathcal{C}| \ .$$

Analysis

Observation. At any point in the algorithm, the set of edges F is a forest.

Lemma 6.6

Let F' be the final set of edges returned by the algorithm. For any $\mathcal{C}$ in any iteration of the algorithm,

$$\sum_{C \in \mathcal{C}} |\delta(C) \cap F'| \leq 2|\mathcal{C}| \ .$$

Theorem 6.7

The refined primal-dual algorithm is a 2-approximation algorithm for the Steiner Forest Problem.

Analysis

Observation. At any point in the algorithm, the set of edges F is a forest.

Lemma 6.6

Let F' be the final set of edges returned by the algorithm. For any $\mathcal{C}$ in any iteration of the algorithm,

$$\sum_{C \in \mathcal{C}} |\delta(C) \cap F'| \leq 2|\mathcal{C}| \ .$$

Theorem 6.7

The refined primal-dual algorithm is a 2-approximation algorithm for the Steiner Forest Problem.

Proof:… □

Analysis

Observation. At any point in the algorithm, the set of edges F is a forest.

Lemma 6.6

Let F' be the final set of edges returned by the algorithm. For any $\mathcal{C}$ in any iteration of the algorithm,

$$\sum_{C \in \mathcal{C}} |\delta(C) \cap F'| \leq 2|\mathcal{C}| \ .$$

Theorem 6.7

The refined primal-dual algorithm is a 2-approximation algorithm for the Steiner Forest Problem.

Proof:… □

Corollary 6.8

Integrality gap of the LP relaxation is at most 2. This bound is tight. □

Outline

Uncapacitated Facility Location Problem

Given: Set of facilities F with opening costs $f_i \geq 0$, $i \in F$;
set of clients D with metric connection costs $c_{ij} \geq 0$, $i \in F, j \in D$.

Task: Choose $F' \subseteq F$ and assign each client to nearest facility in F'.

Objective: Minimize $\sum_{i \in F'} f_i + \sum_{j \in D} \min_{i \in F'} c_{ij}$.

IP formulation:

$$\begin{aligned}
\min \quad & \sum_{i \in F} f_i \cdot y_i + \sum_{i \in F, j \in D} c_{ij} \cdot x_{ij} & \\
\text{s.t.} \quad & \sum_{i \in F} x_{ij} = 1 & \text{for all } j \in D, \\
& y_i - x_{ij} \geq 0 & \text{for all } i \in F, j \in D, \\
& x_{ij}, y_i \in \{0, 1\} & \text{for all } i \in F, j \in D.
\end{aligned}$$

LP Relaxation and Dual LP

primal LP:

$$\begin{aligned} \min \quad & \sum_{i \in F} f_i \cdot y_i + \sum_{i \in F, j \in D} c_{ij} \cdot x_{ij} & \\ \text{s.t.} \quad & \sum_{i \in F} x_{ij} = 1 & \text{for all } j \in D, \\ & y_i - x_{ij} \geq 0 & \text{for all } i \in F, j \in D, \\ & x_{ij}, y_i \geq 0 & \text{for all } i \in F, j \in D. \end{aligned}$$

dual LP:

$$\begin{aligned} \max \quad & \textstyle\sum_{j \in D} v_j & \\ \text{s.t.} \quad & \textstyle\sum_{j \in D} w_{ij} \leq f_i & \text{for all } i \in F, \\ & v_j - w_{ij} \leq c_{ij} & \text{for all } i \in F, j \in D, \\ & w_{ij} \geq 0 & \text{for all } i \in F, j \in D. \end{aligned}$$

Interpretation of dual LP:

- v_j is total amount that client j wants to pay for being served.

LP Relaxation and Dual LP

primal LP:

$$\begin{aligned} \min \quad & \sum_{i \in F} f_i \cdot y_i + \sum_{i \in F, j \in D} c_{ij} \cdot x_{ij} & \\ \text{s.t.} \quad & \sum_{i \in F} x_{ij} = 1 & \text{for all } j \in D, \\ & y_i - x_{ij} \geq 0 & \text{for all } i \in F, j \in D, \\ & x_{ij}, y_i \geq 0 & \text{for all } i \in F, j \in D. \end{aligned}$$

dual LP:

$$\begin{aligned} \max \quad & \textstyle\sum_{j \in D} v_j & \\ \text{s.t.} \quad & \textstyle\sum_{j \in D} w_{ij} \leq f_i & \text{for all } i \in F, \\ & v_j - w_{ij} \leq c_{ij} & \text{for all } i \in F, j \in D, \\ & w_{ij} \geq 0 & \text{for all } i \in F, j \in D. \end{aligned}$$

Interpretation of dual LP:

- v_j is total amount that client j wants to pay for being served.
- client j might contribute w_{ij} to facility i for being connected to i.

Primal-Dual Algo for Uncapacitated Facility Location

Notation & High-level idea:

- For the current feasible dual solution (v, w) and $j \in D$ let
$$N(j) := \{i \in F \mid v_j \geq c_{ij}\} \ .$$

Primal-Dual Algo for Uncapacitated Facility Location

Notation & High-level idea:

- For the current feasible dual solution (v, w) and $j \in D$ let
$$N(j) := \{i \in F \mid v_j \geq c_{ij}\} \ .$$
- If $w_{ij} > 0$, we say that client j *contributes* to facility i.

Primal-Dual Algo for Uncapacitated Facility Location

Notation & High-level idea:

- For the current feasible dual solution (v, w) and $j \in D$ let
$$N(j) := \{i \in F \mid v_j \geq c_{ij}\} \ .$$
- If $w_{ij} > 0$, we say that client j *contributes* to facility i.
- During the primal-dual algorithm, we maintain dual feasibility, and we define the set of *tight facilities*:
$$T = \{i \in F : \sum_{j \in D} w_{ij} = f_i\} \ .$$

Primal-Dual Algo for Uncapacitated Facility Location

Notation & High-level idea:

- For the current feasible dual solution (v, w) and $j \in D$ let
$$N(j) := \{i \in F \mid v_j \geq c_{ij}\} \ .$$
- If $w_{ij} > 0$, we say that client j *contributes* to facility i.
- During the primal-dual algorithm, we maintain dual feasibility, and we define the set of *tight facilities*:
$$T = \{i \in F : \sum_{j \in D} w_{ij} = f_i\} \ .$$
- During the algorithm, we increase the v_j's ans w_{ij}'s, in a way that
  - client j contributes only to facilities $i \in N(j)$;
  - we maintain $v_j = w_{ij} + c_{ij}, \ \forall i \in N(j)$.

Primal-Dual Algo for Uncapacitated Facility Location

Notation & High-level idea:

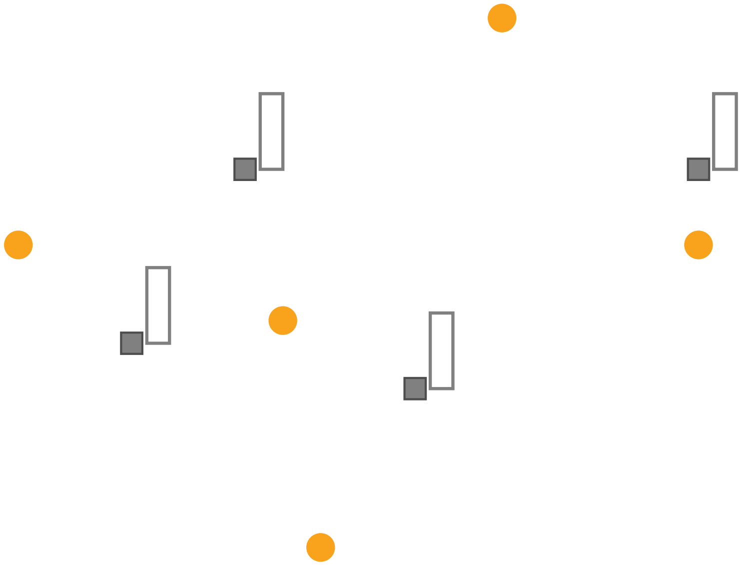
- For the current feasible dual solution (v, w) and $j \in D$ let
$$N(j) := \{i \in F \mid v_j \geq c_{ij}\} \ .$$
- If $w_{ij} > 0$, we say that client j *contributes* to facility i.
- During the primal-dual algorithm, we maintain dual feasibility, and we define the set of *tight facilities*:
$$T = \{i \in F : \sum_{j \in D} w_{ij} = f_i\} \ .$$
- During the algorithm, we increase the v_j's ans w_{ij}'s, in a way that
  - client j contributes only to facilities $i \in N(j)$;
  - we maintain $v_j = w_{ij} + c_{ij}, \ \forall i \in N(j)$.
- The set S is the set of clients not yet in the neighborhood of a tight facility; We'll iterate until $S = \emptyset$.

Primal-Dual Algo for Uncapacitated Facility Location

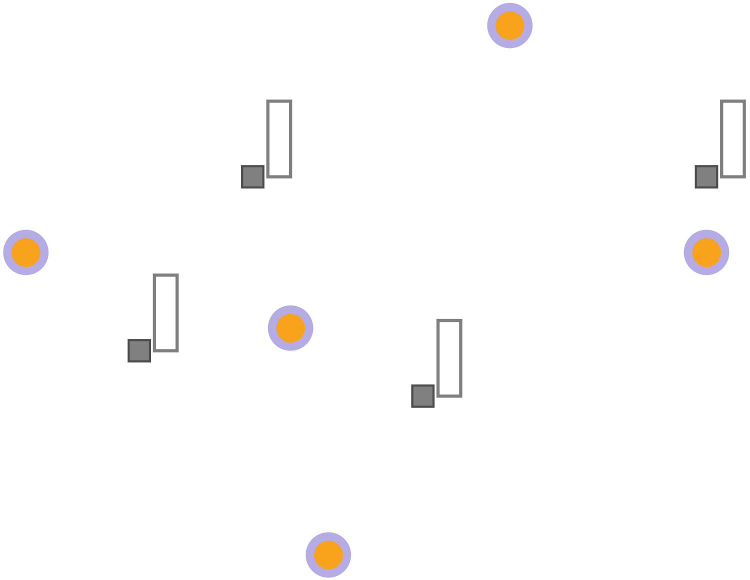
Algorithm:

1. set $S := D$, $T := \emptyset$, $v_j := 0$, $w_{ij} := 0$ for all $i \in F, j \in D$;
2. while $S \neq \emptyset$
3. for all $j \in S$ increase v_j and w_{ij} for all $i \in N(j)$ uniformly until one of the following occurs:
   - i. $\sum_{j \in D} w_{ij} = f_i$ for some $i \notin T$; Then, $T \leftarrow T \cup \{i\}$
   - ii. $v_j \geq c_{ij}$ for some $i \in T$, $j \in S$; Then, $S \leftarrow S \setminus \{j\}$
4. set $F' := \emptyset$
5. while $T \neq \emptyset$ pick $i \in T$;
6. $F' := F' \cup \{i\}$; $T := T \setminus \{h \mid \exists j \in D, \, w_{ij} > 0, \, w_{hj} > 0\}$;
7. open all facilities in F'; connect each $j \in D$ to nearest $i \in F'$.

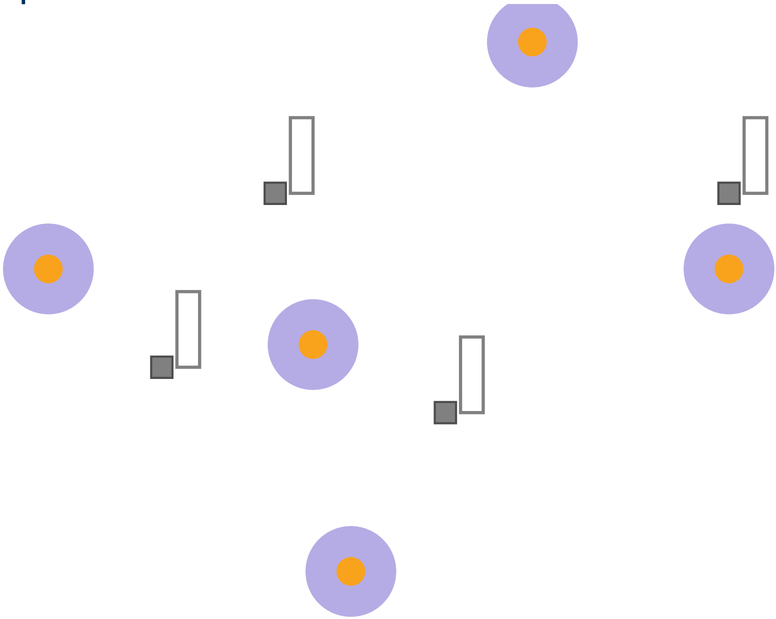
Example



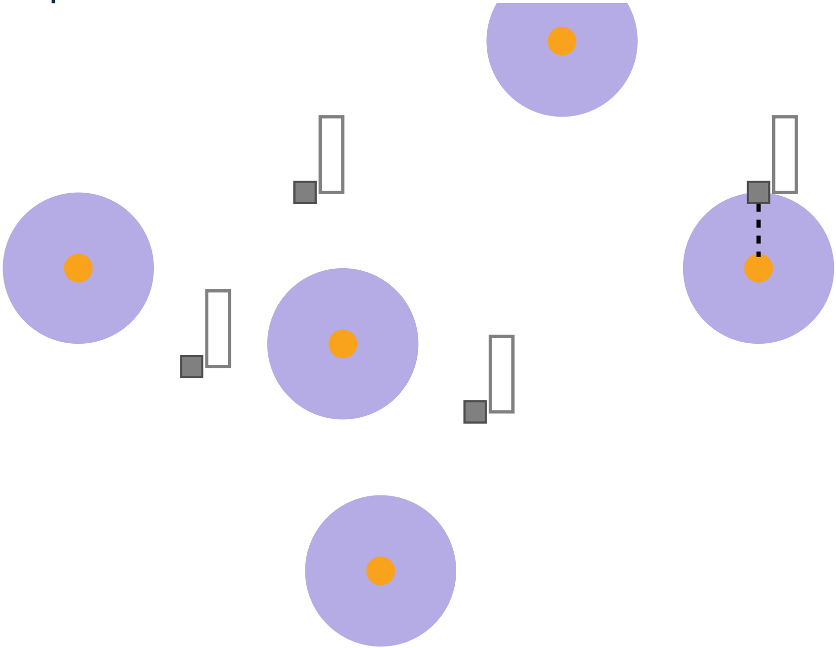
Example



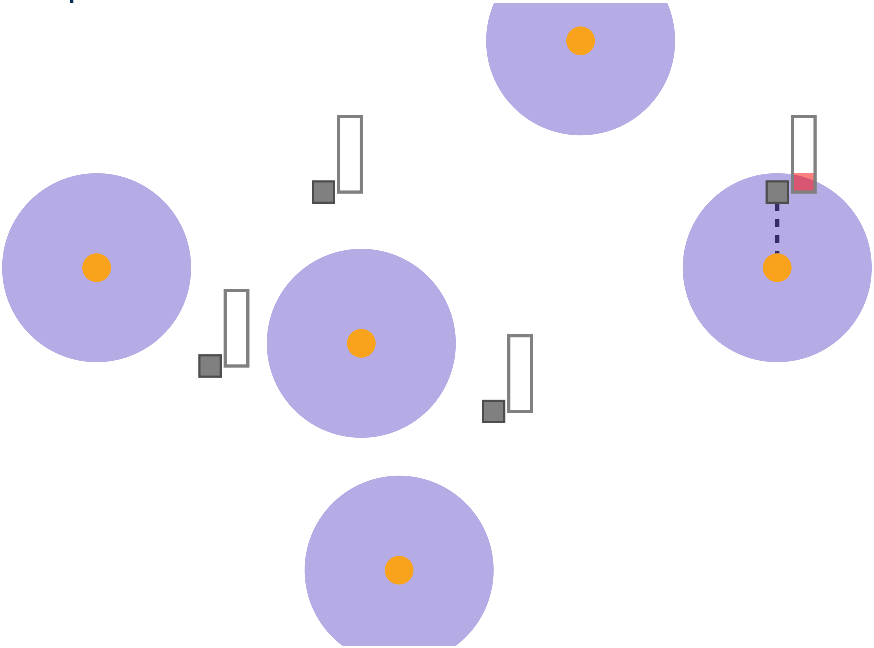
Example



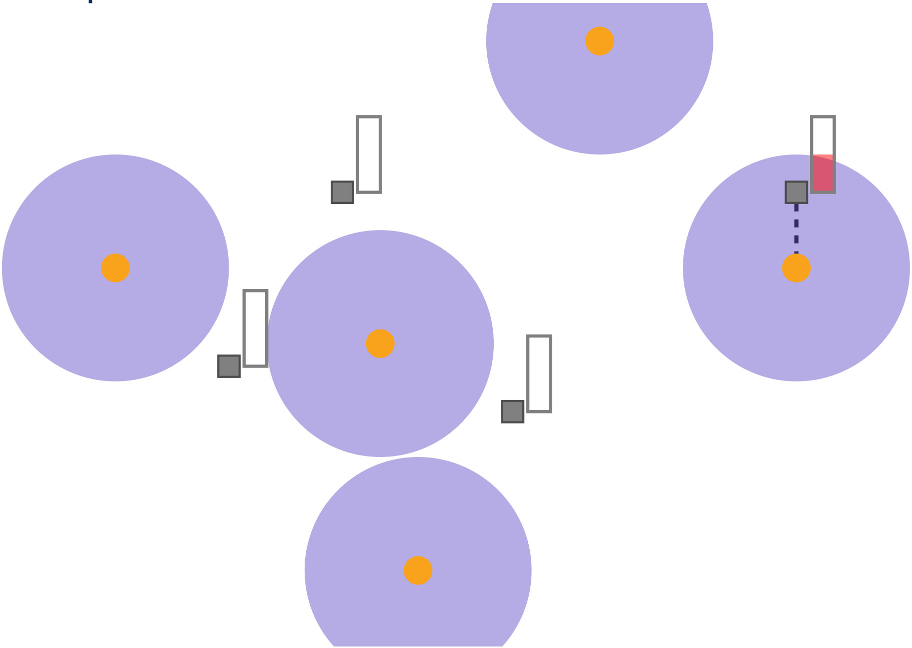
Example



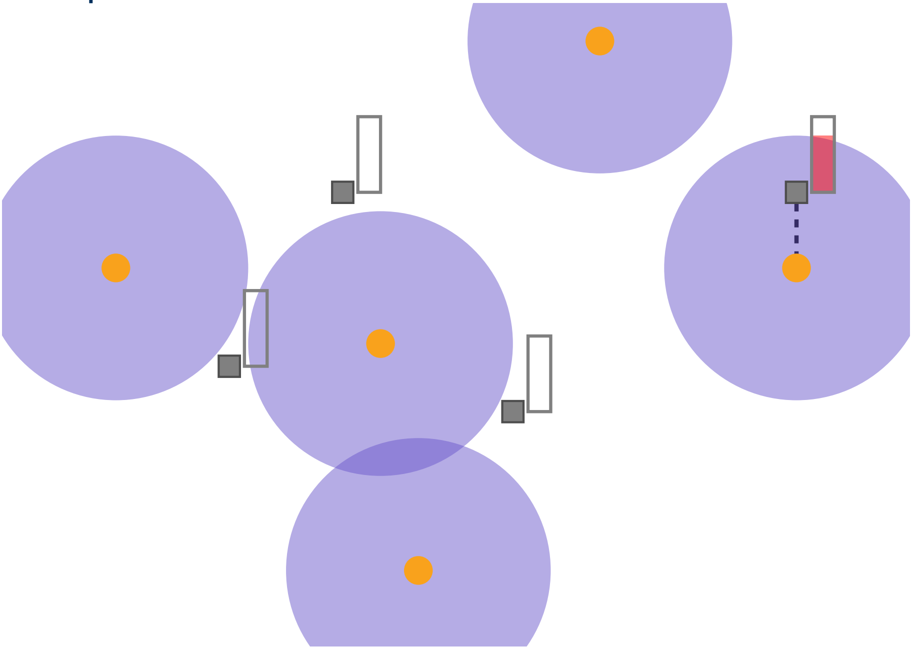
Example



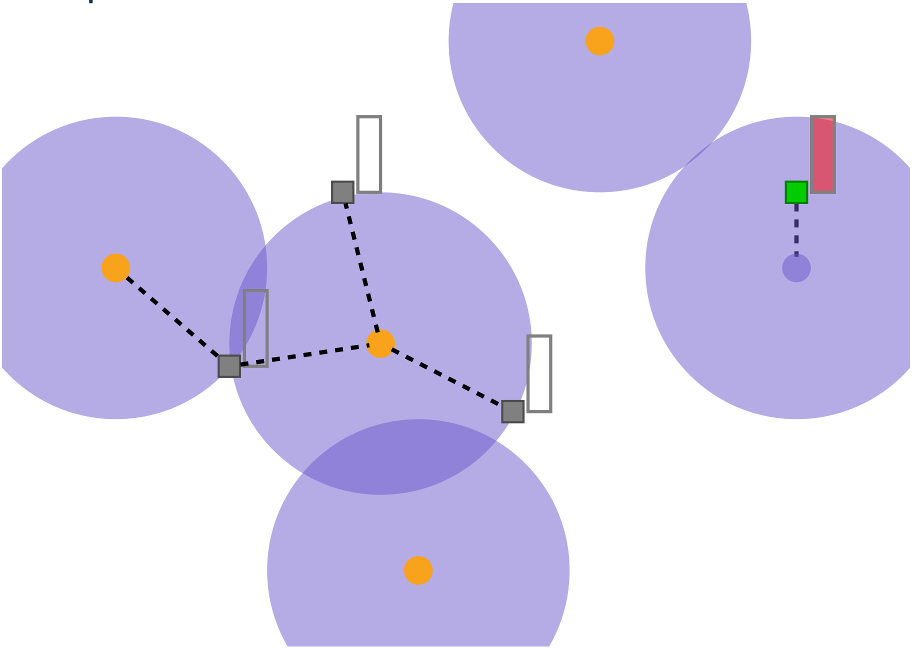
Example



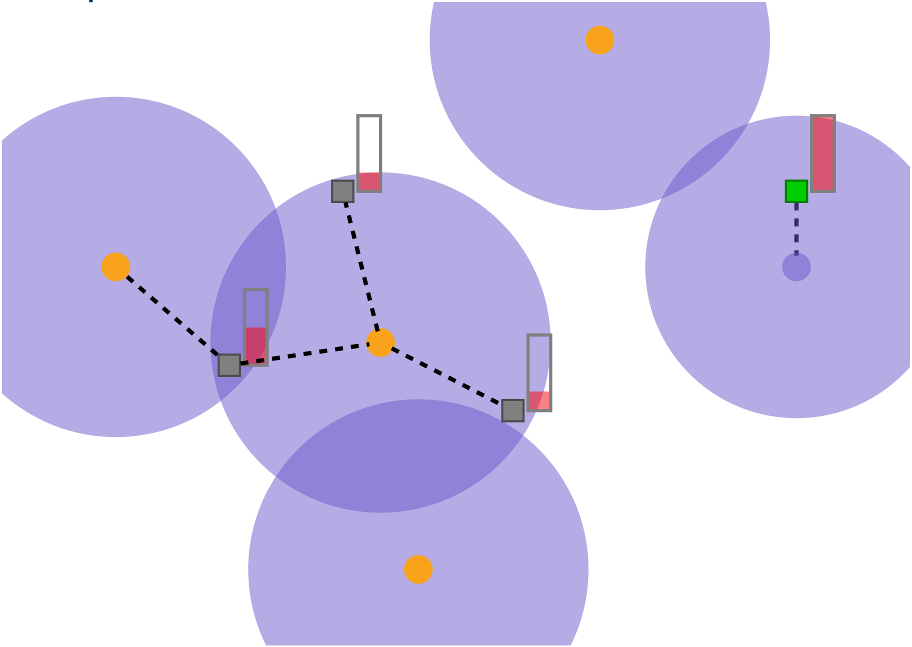
Example



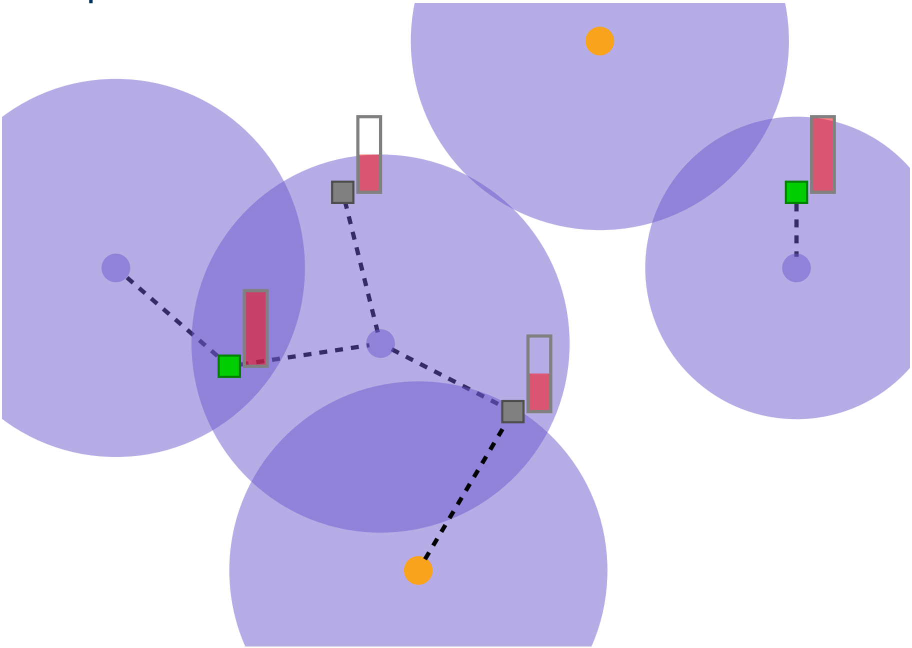
Example



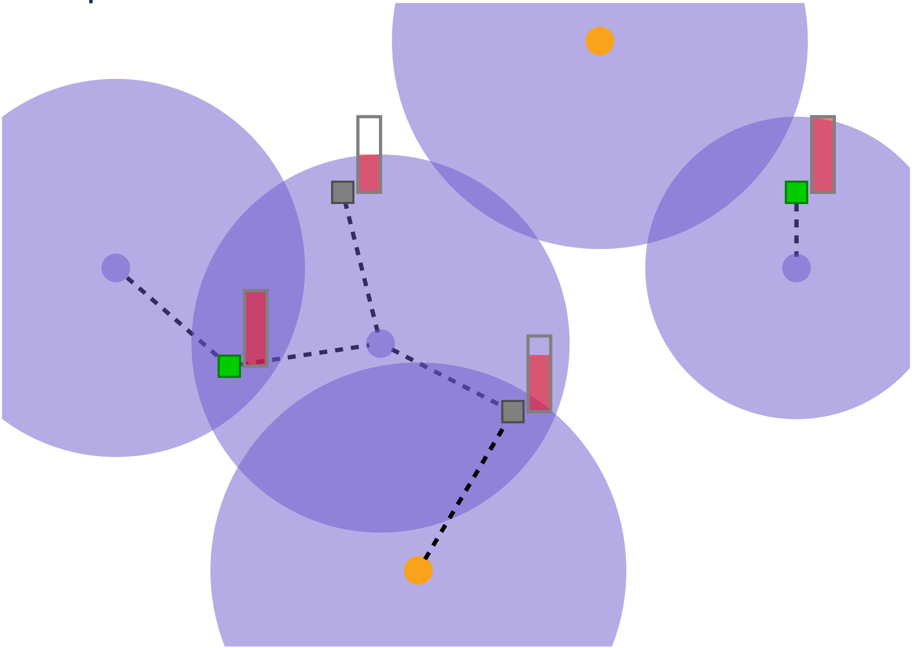
Example



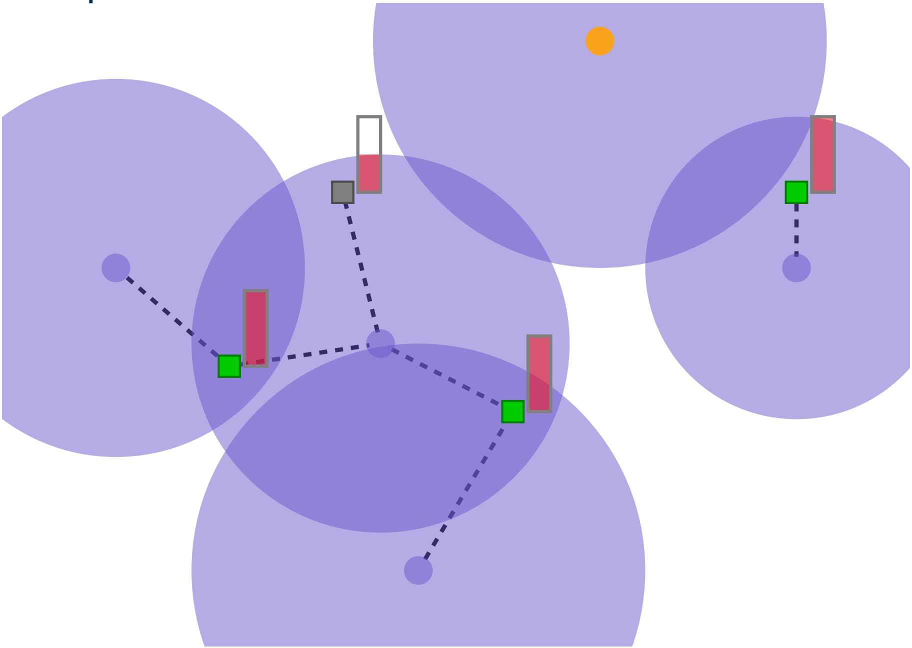
Example



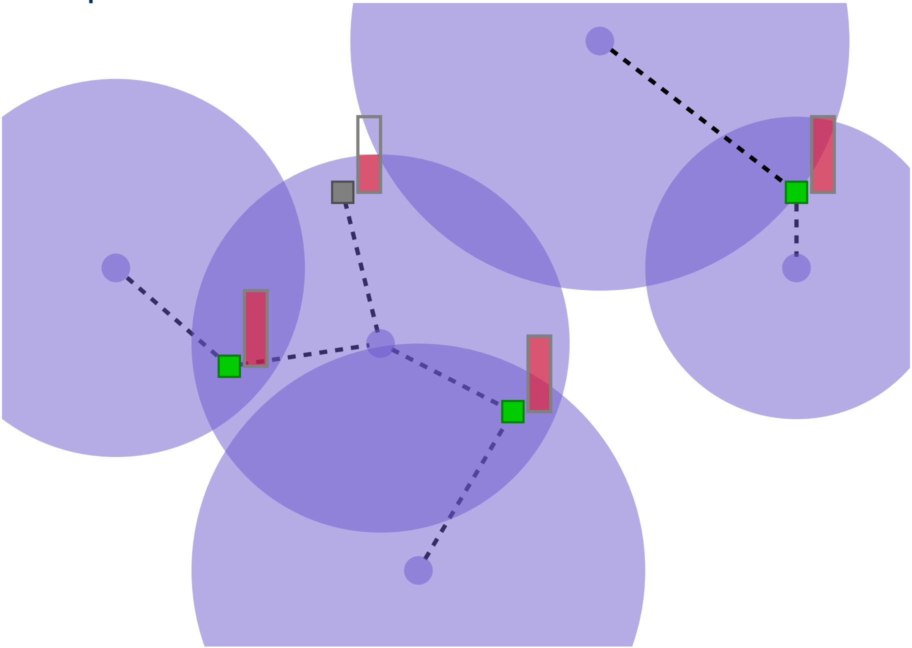
Example



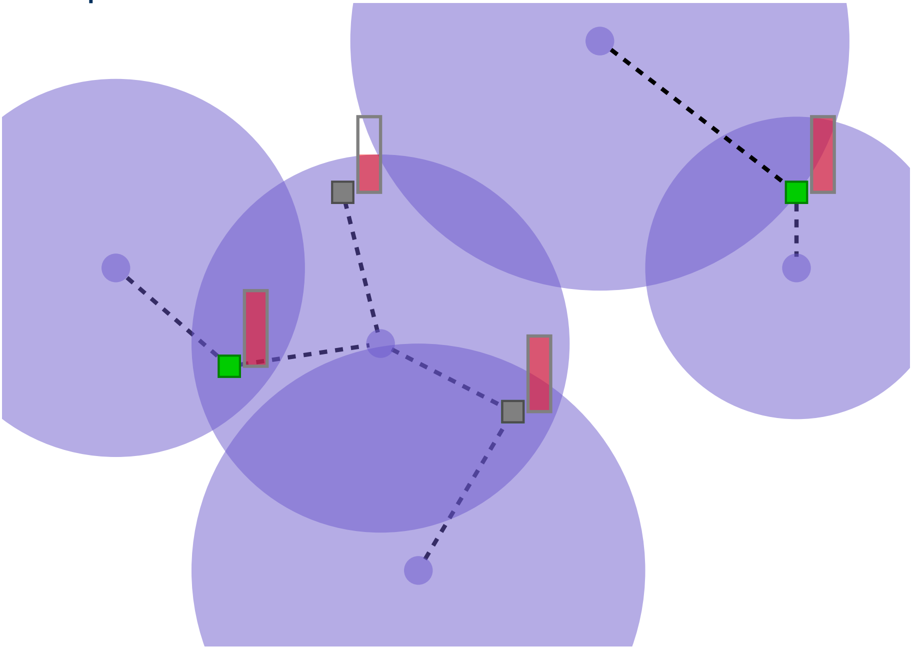
Example



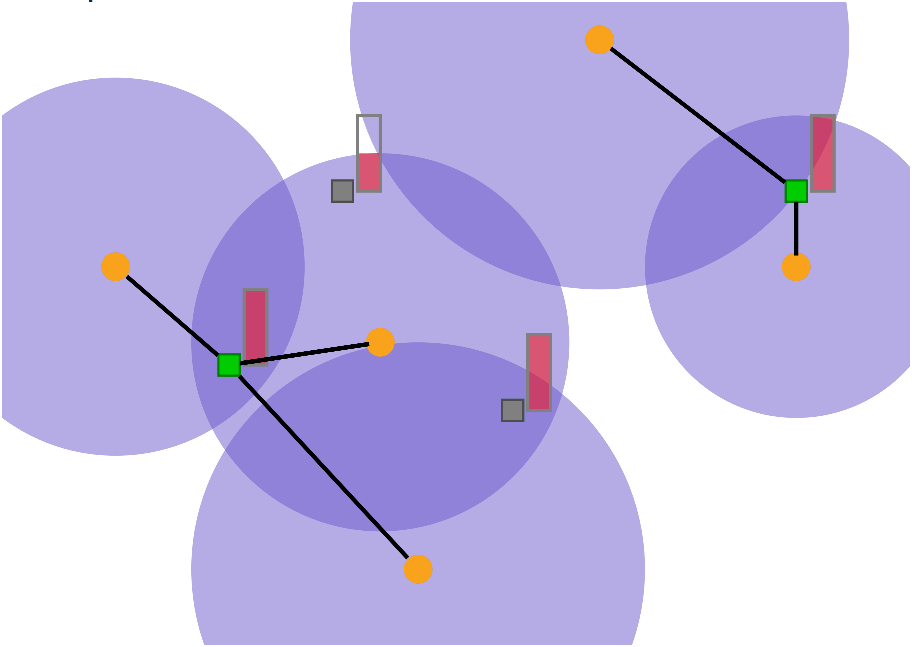
Example



Example



Example



Analysis

Lemma 6.9

If a client j does not have a neighbor in F', then there is a facility $i \in F'$ such that $c_{ij} \leq 3v_j$.

Analysis

Lemma 6.9

If a client j does not have a neighbor in F', then there is a facility $i \in F'$ such that $c_{ij} \leq 3v_j$.

Proof:... □

Analysis

Lemma 6.9

If a client j does not have a neighbor in F', then there is a facility $i \in F'$ such that $c_{ij} \leq 3v_j$.

Proof:... □

Theorem 6.10

The algorithm is a 3-approximation algorithm for the uncapacitated facility location problem.

Analysis

Lemma 6.9

If a client j does not have a neighbor in F', then there is a facility $i \in F'$ such that $c_{ij} \leq 3v_j$.

Proof:… □

Theorem 6.10

The algorithm is a 3-approximation algorithm for the uncapacitated facility location problem.

Proof:… □