

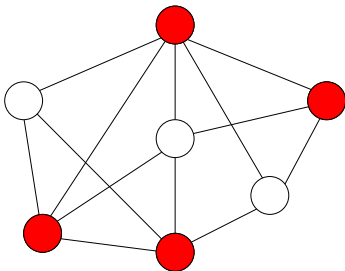
Advanced Algorithms (Fall 2025)

Primal-Dual Algorithms

Lecturers: 尹一通, 刘景铖, 栗师

Nanjing University

- 1 2-Approximation Algorithm for Weighted Vertex Cover Using Primal-Dual
- 2 3-Approximation Algorithm for Uncapacitated Facility Location Problem Using Primal Dual



Weighted Vertex Cover Problem

Input: graph $G = (V, E)$, **vertex weights** $w \in \mathbb{Z}_{>0}^V$

Output: vertex cover S of G , to minimize $\sum_{v \in S} w_v$

LP Relaxation

$$\min \sum_{v \in V} w_v x_v$$

$$x_u + x_v \geq 1 \quad \forall (u, v) \in E$$

$$x_v \geq 0 \quad \forall v \in V$$

Dual LP

$$\max \sum_{e \in E} y_e$$

$$\sum_{e \in \delta(v)} y_e \leq w_v \quad \forall v \in V$$

$$y_e \geq 0 \quad \forall e \in E$$

LP Relaxation

$$\min \sum_{v \in V} w_v x_v$$

$$x_u + x_v \geq 1 \quad \forall (u, v) \in E$$

$$x_v \geq 0 \quad \forall v \in V$$

Dual LP

$$\max \sum_{e \in E} y_e$$

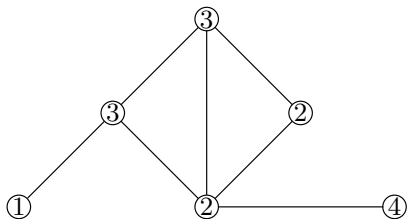
$$\sum_{e \in \delta(v)} y_e \leq w_v \quad \forall v \in V$$

$$y_e \geq 0 \quad \forall e \in E$$

- Algorithm constructs **integral primal solution** x and dual solution y simultaneously.

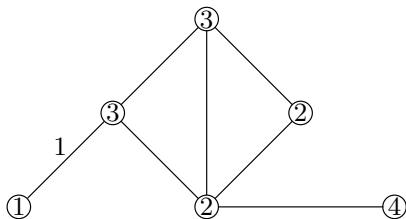
Primal-Dual Algorithm for Weighted Vertex Cover Problem

- 1: $x \leftarrow 0, y \leftarrow 0$, all edges said to be **uncovered**
- 2: **while** there exists at least one uncovered edge **do**
- 3: take such an edge e arbitrarily
- 4: increasing y_e until the dual constraint for one end-vertex v of e becomes tight
- 5: $x_v \leftarrow 1$, claim all edges incident to v are **covered**
- 6: **return** x



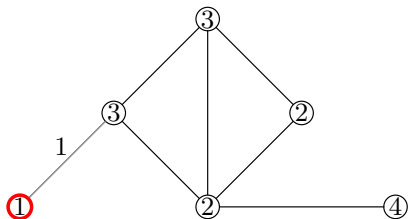
Primal-Dual Algorithm for Weighted Vertex Cover Problem

- 1: $x \leftarrow 0, y \leftarrow 0$, all edges said to be **uncovered**
- 2: **while** there exists at least one uncovered edge **do**
- 3: take such an edge e arbitrarily
- 4: increasing y_e until the dual constraint for one end-vertex v of e becomes tight
- 5: $x_v \leftarrow 1$, claim all edges incident to v are **covered**
- 6: **return** x



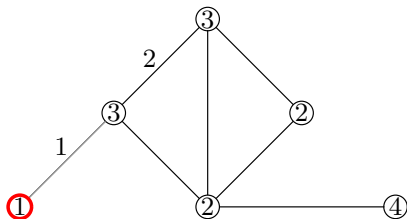
Primal-Dual Algorithm for Weighted Vertex Cover Problem

- 1: $x \leftarrow 0, y \leftarrow 0$, all edges said to be **uncovered**
- 2: **while** there exists at least one uncovered edge **do**
- 3: take such an edge e arbitrarily
- 4: increasing y_e until the dual constraint for one end-vertex v of e becomes tight
- 5: $x_v \leftarrow 1$, claim all edges incident to v are **covered**
- 6: **return** x



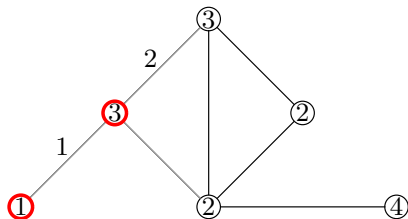
Primal-Dual Algorithm for Weighted Vertex Cover Problem

- 1: $x \leftarrow 0, y \leftarrow 0$, all edges said to be **uncovered**
- 2: **while** there exists at least one uncovered edge **do**
- 3: take such an edge e arbitrarily
- 4: increasing y_e until the dual constraint for one end-vertex v of e becomes tight
- 5: $x_v \leftarrow 1$, claim all edges incident to v are **covered**
- 6: **return** x



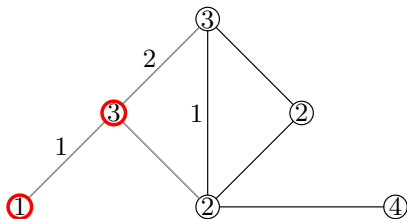
Primal-Dual Algorithm for Weighted Vertex Cover Problem

- 1: $x \leftarrow 0, y \leftarrow 0$, all edges said to be **uncovered**
- 2: **while** there exists at least one uncovered edge **do**
- 3: take such an edge e arbitrarily
- 4: increasing y_e until the dual constraint for one end-vertex v of e becomes tight
- 5: $x_v \leftarrow 1$, claim all edges incident to v are **covered**
- 6: **return** x



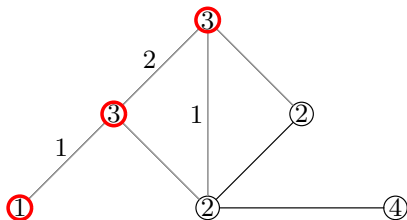
Primal-Dual Algorithm for Weighted Vertex Cover Problem

- 1: $x \leftarrow 0, y \leftarrow 0$, all edges said to be **uncovered**
- 2: **while** there exists at least one uncovered edge **do**
- 3: take such an edge e arbitrarily
- 4: increasing y_e until the dual constraint for one end-vertex v of e becomes tight
- 5: $x_v \leftarrow 1$, claim all edges incident to v are **covered**
- 6: **return** x



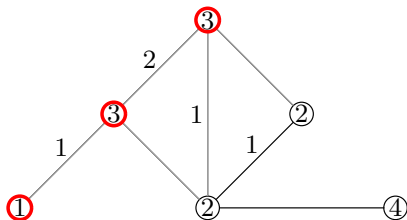
Primal-Dual Algorithm for Weighted Vertex Cover Problem

- 1: $x \leftarrow 0, y \leftarrow 0$, all edges said to be **uncovered**
- 2: **while** there exists at least one uncovered edge **do**
- 3: take such an edge e arbitrarily
- 4: increasing y_e until the dual constraint for one end-vertex v of e becomes tight
- 5: $x_v \leftarrow 1$, claim all edges incident to v are **covered**
- 6: **return** x



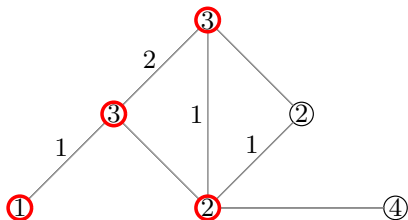
Primal-Dual Algorithm for Weighted Vertex Cover Problem

- 1: $x \leftarrow 0, y \leftarrow 0$, all edges said to be **uncovered**
- 2: **while** there exists at least one uncovered edge **do**
- 3: take such an edge e arbitrarily
- 4: increasing y_e until the dual constraint for one end-vertex v of e becomes tight
- 5: $x_v \leftarrow 1$, claim all edges incident to v are **covered**
- 6: **return** x



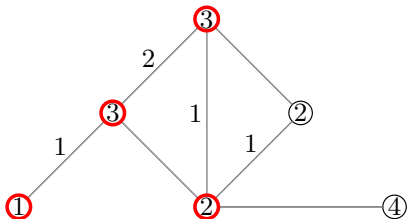
Primal-Dual Algorithm for Weighted Vertex Cover Problem

- 1: $x \leftarrow 0, y \leftarrow 0$, all edges said to be **uncovered**
- 2: **while** there exists at least one uncovered edge **do**
- 3: take such an edge e arbitrarily
- 4: increasing y_e until the dual constraint for one end-vertex v of e becomes tight
- 5: $x_v \leftarrow 1$, claim all edges incident to v are **covered**
- 6: **return** x



Primal-Dual Algorithm for Weighted Vertex Cover Problem

- 1: $x \leftarrow 0, y \leftarrow 0$, all edges said to be **uncovered**
- 2: **while** there exists at least one uncovered edge **do**
- 3: take such an edge e arbitrarily
- 4: increasing y_e until the dual constraint for one end-vertex v of e becomes tight
- 5: $x_v \leftarrow 1$, claim all edges incident to v are **covered**
- 6: **return** x



Lemma

- ① x satisfies all primal constraints
- ② y satisfies all dual constraints
- ③ $P \leq 2D \leq 2D^* \leq 2 \cdot \text{opt}$
 $P := \sum_{v \in V} x_v$: value of x
 $D := \sum_{e \in E} y_e$: value of y
 D^* : dual LP value

Proof of $P \leq 2D$.

$$\begin{aligned} P &= \sum_{v \in V} w_v x_v \leq \sum_{v \in V} x_v \sum_{e \in \delta(v)} y_e = \sum_{(u,v) \in E} y_{(u,v)} (x_u + x_v) \\ &\leq 2 \sum_{e \in E} y_e = 2D. \end{aligned}$$

□

Proof of $P \leq 2D$.

$$\begin{aligned} P &= \sum_{v \in V} w_v x_v \leq \sum_{v \in V} x_v \sum_{e \in \delta(v)} y_e = \sum_{(u,v) \in E} y_{(u,v)} (x_u + x_v) \\ &\leq 2 \sum_{e \in E} y_e = 2D. \end{aligned}$$

□

- a more general framework: construct an arbitrary **maximal** dual solution y ; choose the vertices whose dual constraints are tight
- y is maximal: increasing any coordinate y_e makes y infeasible

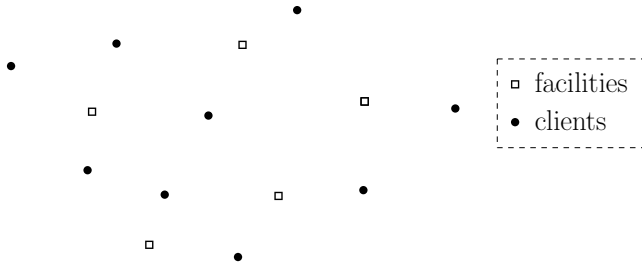
Proof of $P \leq 2D$.

$$\begin{aligned} P &= \sum_{v \in V} w_v x_v \leq \sum_{v \in V} x_v \sum_{e \in \delta(v)} y_e = \sum_{(u,v) \in E} y_{(u,v)} (x_u + x_v) \\ &\leq 2 \sum_{e \in E} y_e = 2D. \end{aligned}$$

□

- a more general framework: construct an arbitrary **maximal** dual solution y ; choose the vertices whose dual constraints are tight
- y is maximal: increasing any coordinate y_e makes y infeasible
- primal-dual algorithms do not need to solve LPs
- LPs are used in analysis only
- faster than LP-rounding algorithm in general

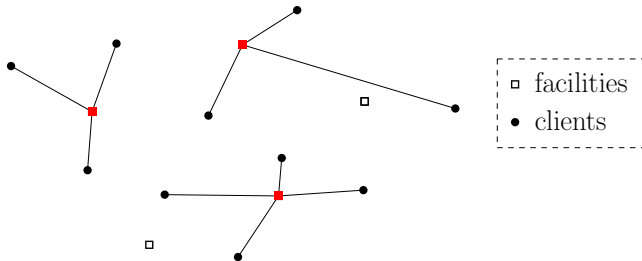
- 1 2-Approximation Algorithm for Weighted Vertex Cover Using Primal-Dual
- 2 3-Approximation Algorithm for Uncapacitated Facility Location Problem Using Primal Dual



Uncapacitated Facility Location Problem

Input: F : potential facilities C : clients

d : (symmetric) metric over $F \cup C$ $(f_i)_{i \in F}$: facility opening costs

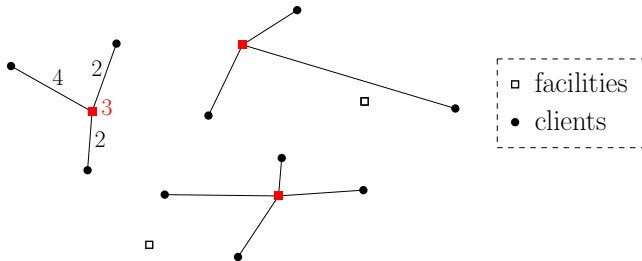


Uncapacitated Facility Location Problem

Input: F : potential facilities C : clients

d : (symmetric) metric over $F \cup C$ $(f_i)_{i \in F}$: facility opening costs

Output: $S \subseteq F$, so as to minimize $\sum_{i \in S} f_i + \sum_{j \in C} d(j, S)$



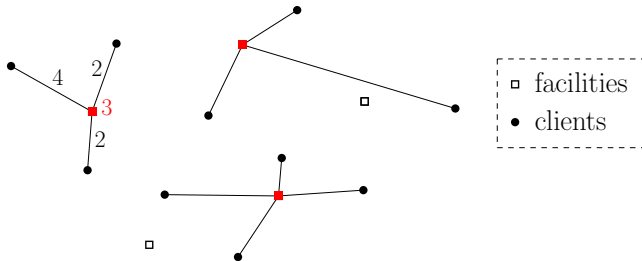
Uncapacitated Facility Location Problem

Input: F : potential facilities C : clients

d : (symmetric) metric over $F \cup C$ $(f_i)_{i \in F}$: facility opening costs

Output: $S \subseteq F$, so as to minimize $\sum_{i \in S} f_i + \sum_{j \in C} d(j, S)$

- 1.488-approximation [Li, 2011]



Uncapacitated Facility Location Problem

Input: F : potential facilities C : clients

d : (symmetric) metric over $F \cup C$ $(f_i)_{i \in F}$: facility opening costs

Output: $S \subseteq F$, so as to minimize $\sum_{i \in S} f_i + \sum_{j \in C} d(j, S)$

- 1.488-approximation [Li, 2011]
- 1.463-hardness of approximation, $1.463 \approx \text{root of } x = 1 + 2e^{-x}$

- y_i : open facility i ?
- $x_{i,j}$: connect client j to facility i ?

Basic LP Relaxation

$$\min \sum_{i \in F} f_i y_i + \sum_{i \in F, j \in C} d(i, j) x_{i,j}$$

$$\sum_{i \in F} x_{i,j} \geq 1 \quad \forall j \in C$$

$$x_{i,j} \leq y_i \quad \forall i \in F, j \in C$$

$$x_{i,j} \geq 0 \quad \forall i \in F, j \in C$$

$$y_i \geq 0 \quad \forall i \in F$$

- y_i : open facility i ?
- $x_{i,j}$: connect client j to facility i ?

Basic LP Relaxation

$$\min \sum_{i \in F} f_i y_i + \sum_{i \in F, j \in C} d(i, j) x_{i,j}$$

$$\sum_{i \in F} x_{i,j} \geq 1 \quad \forall j \in C$$

$$x_{i,j} \leq y_i \quad \forall i \in F, j \in C$$

$$x_{i,j} \geq 0 \quad \forall i \in F, j \in C$$

$$y_i \geq 0 \quad \forall i \in F$$

Obs. When $(y_i)_{i \in F}$ is determined, $(x_{i,j})_{i \in F, j \in C}$ can be determined automatically.

- y_i : open facility i ?
- $x_{i,j}$: connect client j to facility i ?

Basic LP Relaxation

$$\min \sum_{i \in F} f_i y_i + \sum_{i \in F, j \in C} d(i, j) x_{i, j}$$

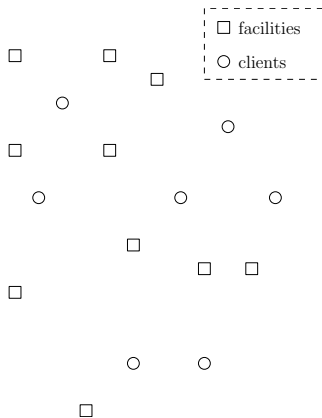
$$\sum_{i \in F} x_{i, j} \geq 1 \quad \forall j \in C$$

$$x_{i, j} \leq y_i \quad \forall i \in F, j \in C$$

$$x_{i, j} \geq 0 \quad \forall i \in F, j \in C$$

$$y_i \geq 0 \quad \forall i \in F$$

Obs. When $(y_i)_{i \in F}$ is determined, $(x_{i, j})_{i \in F, j \in C}$ can be determined automatically.



- y_i : open facility i ?
- $x_{i,j}$: connect client j to facility i ?

Basic LP Relaxation

$$\min \sum_{i \in F} f_i y_i + \sum_{i \in F, j \in C} d(i, j) x_{i, j}$$

$$\sum_{i \in F} x_{i, j} \geq 1 \quad \forall j \in C$$

$$x_{i, j} \leq y_i \quad \forall i \in F, j \in C$$

$$x_{i, j} \geq 0 \quad \forall i \in F, j \in C$$

$$y_i \geq 0 \quad \forall i \in F$$



Obs. When $(y_i)_{i \in F}$ is determined, $(x_{i,j})_{i \in F, j \in C}$ can be determined automatically.

- y_i : open facility i ?
- $x_{i,j}$: connect client j to facility i ?

Basic LP Relaxation

$$\min \sum_{i \in F} f_i y_i + \sum_{i \in F, j \in C} d(i, j) x_{i, j}$$

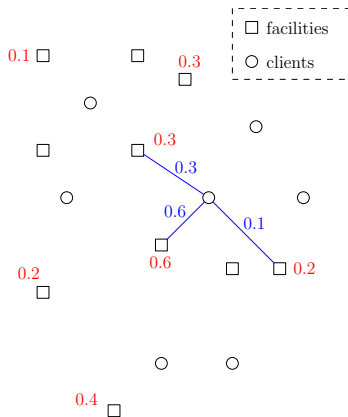
$$\sum_{i \in F} x_{i, j} \geq 1 \quad \forall j \in C$$

$$x_{i, j} \leq y_i \quad \forall i \in F, j \in C$$

$$x_{i, j} \geq 0 \quad \forall i \in F, j \in C$$

$$y_i \geq 0 \quad \forall i \in F$$

Obs. When $(y_i)_{i \in F}$ is determined, $(x_{i, j})_{i \in F, j \in C}$ can be determined automatically.



Basic LP Relaxation

$$\min \sum_{i \in F} f_i y_i + \sum_{i \in F, j \in C} d(i, j) x_{i,j}$$

$$\sum_{i \in F} x_{i,j} \geq 1 \quad \forall j \in C$$

$$x_{i,j} \leq y_i \quad \forall i \in F, j \in C$$

$$x_{i,j} \geq 0 \quad \forall i \in F, j \in C$$

$$y_i \geq 0 \quad \forall i \in F$$

- LP is not of covering type
- harder to understand the dual

Basic LP Relaxation

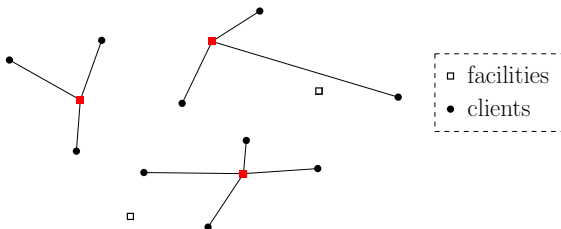
$$\begin{aligned} \min \quad & \sum_{i \in F} f_i y_i + \sum_{i \in F, j \in C} d(i, j) x_{i,j} \\ & \sum_{i \in F} x_{i,j} \geq 1 \quad \forall j \in C \\ & x_{i,j} \leq y_i \quad \forall i \in F, j \in C \\ & x_{i,j} \geq 0 \quad \forall i \in F, j \in C \\ & y_i \geq 0 \quad \forall i \in F \end{aligned}$$

- LP is not of covering type
- harder to understand the dual
- consider an equivalent covering LP
- idea: treat a solution as a set of **stars**

Basic LP Relaxation

$$\begin{aligned} \min \quad & \sum_{i \in F} f_i y_i + \sum_{i \in F, j \in C} d(i, j) x_{i,j} \\ \sum_{i \in F} x_{i,j} & \geq 1 \quad \forall j \in C \\ x_{i,j} & \leq y_i \quad \forall i \in F, j \in C \\ x_{i,j} & \geq 0 \quad \forall i \in F, j \in C \\ y_i & \geq 0 \quad \forall i \in F \end{aligned}$$

- LP is not of covering type
- harder to understand the dual
- consider an equivalent covering LP
- idea: treat a solution as a set of **stars**



- $(i, J), i \in F, J \subseteq C$: star with center i and leaves J

- $(i, J), i \in F, J \subseteq C$: star with center i and leaves J
- $\text{cost}(i, J) := f_i + \sum_{j \in J} d(i, j)$: cost of star (i, J)

- $(i, J), i \in F, J \subseteq C$: star with center i and leaves J
- $\text{cost}(i, J) := f_i + \sum_{j \in J} d(i, j)$: cost of star (i, J)
- $x_{i,J} \in \{0, 1\}$: if star (i, J) is chosen

- $(i, J), i \in F, J \subseteq C$: star with center i and leaves J
- $\text{cost}(i, J) := f_i + \sum_{j \in J} d(i, j)$: cost of star (i, J)
- $x_{i,J} \in \{0, 1\}$: if star (i, J) is chosen

Equivalent LP

$$\begin{aligned} \min \quad & \sum_{(i,J)} \text{cost}(i, J) \cdot x_{i,J} \\ \sum_{(i,J): j \in J} x_{i,J} & \geq 1 \quad \forall j \in C \\ x_{i,J} & \geq 0 \quad \forall (i, J) \end{aligned}$$

- $(i, J), i \in F, J \subseteq C$: star with center i and leaves J
- $\text{cost}(i, J) := f_i + \sum_{j \in J} d(i, j)$: cost of star (i, J)
- $x_{i,J} \in \{0, 1\}$: if star (i, J) is chosen

Equivalent LP

$$\begin{aligned}
 \min \quad & \sum_{(i,J)} \text{cost}(i, J) \cdot x_{i,J} \\
 \sum_{(i,J): j \in J} x_{i,J} & \geq 1 \quad \forall j \in C \\
 x_{i,J} & \geq 0 \quad \forall (i, J)
 \end{aligned}$$

Dual LP

$$\begin{aligned}
 \max \quad & \sum_{j \in C} \alpha_j \\
 \sum_{j \in J} \alpha_j & \leq \text{cost}(j, J) \quad \forall (i, J) \\
 \alpha_j & \geq 0 \quad \forall j \in C
 \end{aligned}$$

- $(i, J), i \in F, J \subseteq C$: star with center i and leaves J
- $\text{cost}(i, J) := f_i + \sum_{j \in J} d(i, j)$: cost of star (i, J)
- $x_{i,J} \in \{0, 1\}$: if star (i, J) is chosen

Equivalent LP

$$\begin{aligned}
 \min \quad & \sum_{(i,J)} \text{cost}(i, J) \cdot x_{i,J} \\
 \sum_{(i,J): j \in J} x_{i,J} & \geq 1 \quad \forall j \in C \\
 x_{i,J} & \geq 0 \quad \forall (i, J)
 \end{aligned}$$

Dual LP

$$\begin{aligned}
 \max \quad & \sum_{j \in C} \alpha_j \\
 \sum_{j \in J} \alpha_j & \leq \text{cost}(j, J) \quad \forall (i, J) \\
 \alpha_j & \geq 0 \quad \forall j \in C
 \end{aligned}$$

- both LPs have exponential size, but the final algorithm can run in polynomial time

$$\min \sum_{(i,J)} \text{cost}(i, J) \cdot x_{i,J}$$

$$\sum_{(i,J):j \in J} x_{i,J} \geq 1 \quad \forall j \in C$$

$$x_{i,J} \geq 0 \quad \forall (i, J)$$

$$\max \sum_{j \in C} \alpha_j$$

$$\sum_{j \in J} \alpha_j \leq \text{cost}(j, J) \quad \forall (i, J)$$

$$\alpha_j \geq 0 \quad \forall j \in C$$

- α_j : budget of j

$$\min \sum_{(i,J)} \text{cost}(i, J) \cdot x_{i,J}$$

$$\sum_{(i,J): j \in J} x_{i,J} \geq 1 \quad \forall j \in C$$

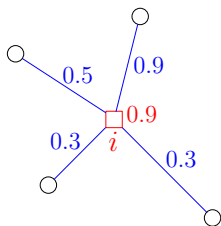
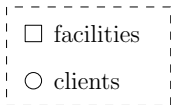
$$x_{i,J} \geq 0 \quad \forall (i, J)$$

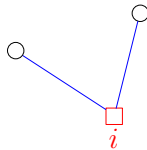
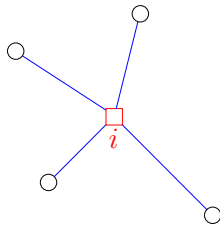
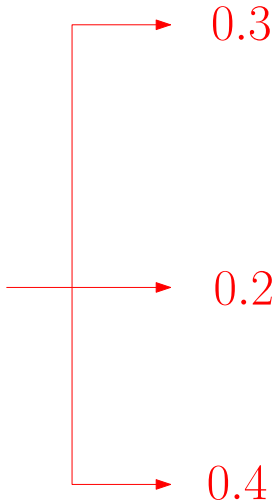
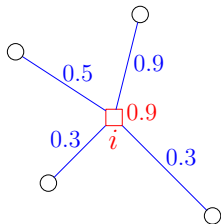
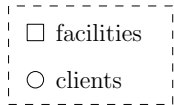
$$\max \sum_{j \in C} \alpha_j$$

$$\sum_{j \in J} \alpha_j \leq \text{cost}(j, J) \quad \forall (i, J)$$

$$\alpha_j \geq 0 \quad \forall j \in C$$

- α_j : budget of j
- dual constraints: total budget in any star is \leq its cost
- $\implies \text{opt} \geq \text{total budget} = \text{dual value}$





Construction of Dual Solution α

- α_j 's can only increase
- α is always feasible

Construction of Dual Solution α

- α_j 's can only increase
- α is always feasible
- if a dual constraint becomes tight, **freeze** all clients in star
- unfrozen clients are called **active** clients

Construction of Dual Solution α

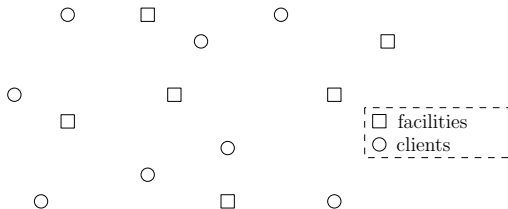
- α_j 's can only increase
- α is always feasible
- if a dual constraint becomes tight, **freeze** all clients in star
- unfrozen clients are called **active** clients

Construction of Dual Solution α

- 1: $\alpha_j \leftarrow 0, \forall j \in C$
- 2: **while** exists at least one active client **do**
- 3: increase the budgets α_j for all active clients j at uniform rate, until (at least) one new client is frozen

Construction of Dual Solution α

- α_j 's can only increase
- α is always feasible
- if a dual constraint becomes tight, **freeze** all clients in star
- unfrozen clients are called **active** clients

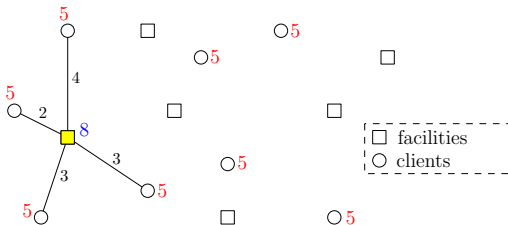


Construction of Dual Solution α

- 1: $\alpha_j \leftarrow 0, \forall j \in C$
- 2: **while** exists at least one active client **do**
- 3: increase the budgets α_j for all active clients j at uniform rate, until (at least) one new client is frozen

Construction of Dual Solution α

- α_j 's can only increase
- α is always feasible
- if a dual constraint becomes tight, **freeze** all clients in star
- unfrozen clients are called **active** clients

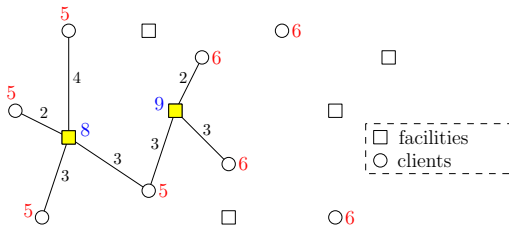


Construction of Dual Solution α

- 1: $\alpha_j \leftarrow 0, \forall j \in C$
- 2: **while** exists at least one active client **do**
- 3: increase the budgets α_j for all active clients j at uniform rate, until (at least) one new client is frozen

Construction of Dual Solution α

- α_j 's can only increase
- α is always feasible
- if a dual constraint becomes tight, **freeze** all clients in star
- unfrozen clients are called **active** clients

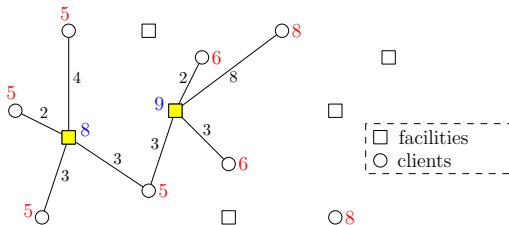


Construction of Dual Solution α

- 1: $\alpha_j \leftarrow 0, \forall j \in C$
- 2: **while** exists at least one active client **do**
- 3: increase the budgets α_j for all active clients j at uniform rate, until (at least) one new client is frozen

Construction of Dual Solution α

- α_j 's can only increase
- α is always feasible
- if a dual constraint becomes tight, **freeze** all clients in star
- unfrozen clients are called **active** clients

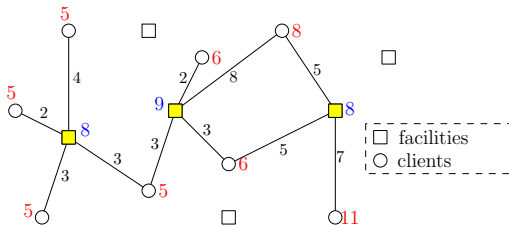


Construction of Dual Solution α

- 1: $\alpha_j \leftarrow 0, \forall j \in C$
- 2: **while** exists at least one active client **do**
- 3: increase the budgets α_j for all active clients j at uniform rate, until (at least) one new client is frozen

Construction of Dual Solution α

- α_j 's can only increase
- α is always feasible
- if a dual constraint becomes tight, **freeze** all clients in star
- unfrozen clients are called **active** clients

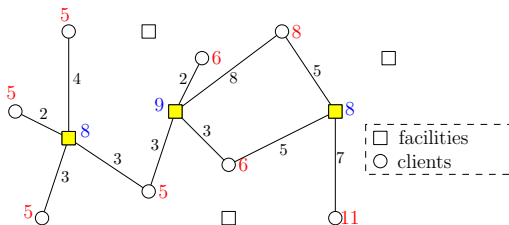


Construction of Dual Solution α

- 1: $\alpha_j \leftarrow 0, \forall j \in C$
- 2: **while** exists at least one active client **do**
- 3: increase the budgets α_j for all active clients j at uniform rate, until (at least) one new client is frozen

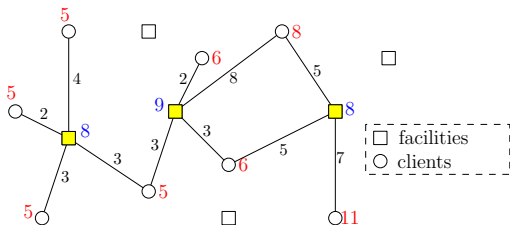
Construction of Dual Solution α

- □: tight facilities; they are temporarily open



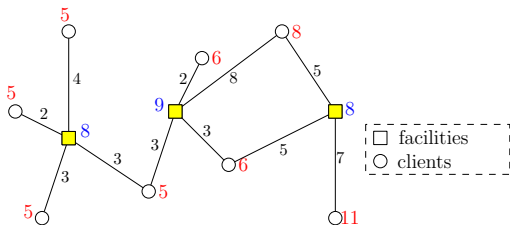
Construction of Dual Solution α

- ■: tight facilities; they are temporarily open
- : permanently closed



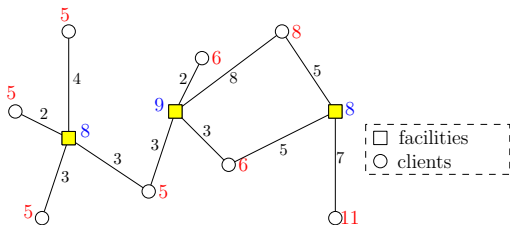
Construction of Dual Solution α

- ■: tight facilities; they are temporarily open
- □: permanently closed



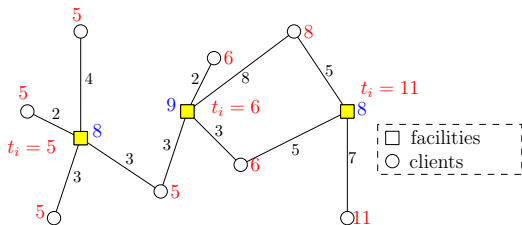
Construction of Dual Solution α

- \blacksquare : tight facilities; they are temporarily open
- \square : permanently closed
- t_i : time when facility i becomes tight



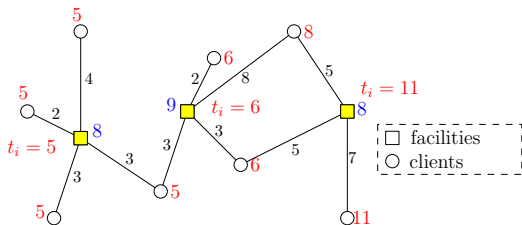
Construction of Dual Solution α

- \blacksquare : tight facilities; they are temporarily open
- \square : permanently closed
- t_i : time when facility i becomes tight



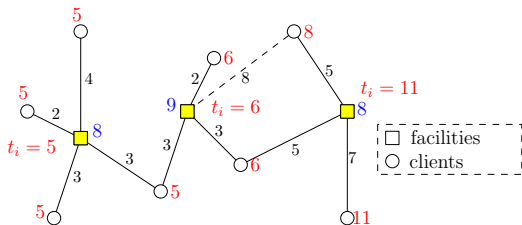
Construction of Dual Solution α

- \blacksquare : tight facilities; they are temporarily open
- \square : permanently closed
- t_i : time when facility i becomes tight
- construct a bipartite graph: (i, j) exists $\iff \alpha_j > d(i, j)$,



Construction of Dual Solution α

- \blacksquare : tight facilities; they are temporarily open
- \square : permanently closed
- t_i : time when facility i becomes tight
- construct a bipartite graph: (i, j) exists $\iff \alpha_j > d(i, j)$,



Construction of Integral Primal Solution

Construction of Integral Primal Solution

1: $S \leftarrow \emptyset$, all clients are **unowned**

Construction of Integral Primal Solution

Construction of Integral Primal Solution

- 1: $S \leftarrow \emptyset$, all clients are **unowned**
- 2: **for** every temporarily open facility i , in increasing order of t_i **do**

Construction of Integral Primal Solution

Construction of Integral Primal Solution

- 1: $S \leftarrow \emptyset$, all clients are **unowned**
- 2: **for** every temporarily open facility i , in increasing order of t_i **do**
- 3: **if** all (solid-line) neighbors of i are unowned **then**
- 4: $S \leftarrow S \cup \{i\}$, open facility i
- 5: connect to all its neighbors to i
- 6: let i own them

Construction of Integral Primal Solution

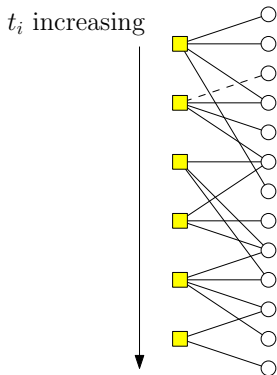
Construction of Integral Primal Solution

- 1: $S \leftarrow \emptyset$, all clients are **unowned**
- 2: **for** every temporarily open facility i , in increasing order of t_i **do**
- 3: **if** all (solid-line) neighbors of i are unowned **then**
- 4: $S \leftarrow S \cup \{i\}$, open facility i
- 5: connect to all its neighbors to i
- 6: let i own them
- 7: connect unconnected clients to their nearest facilities in S

Construction of Integral Primal Solution

Construction of Integral Primal Solution

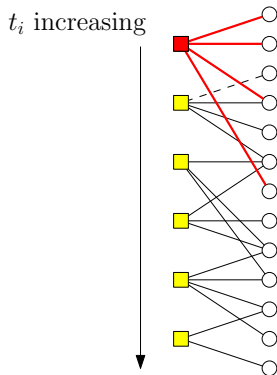
- 1: $S \leftarrow \emptyset$, all clients are **unowned**
- 2: **for** every temporarily open facility i , in increasing order of t_i **do**
- 3: **if** all (solid-line) neighbors of i are unowned **then**
- 4: $S \leftarrow S \cup \{i\}$, open facility i
- 5: connect to all its neighbors to i
- 6: let i own them
- 7: connect unconnected clients to their nearest facilities in S



Construction of Integral Primal Solution

Construction of Integral Primal Solution

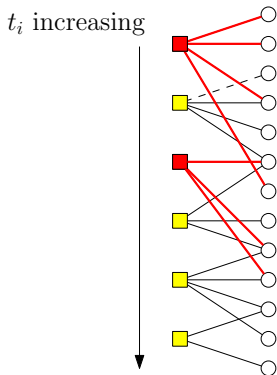
- 1: $S \leftarrow \emptyset$, all clients are **unowned**
- 2: **for** every temporarily open facility i , in increasing order of t_i **do**
- 3: **if** all (solid-line) neighbors of i are unowned **then**
- 4: $S \leftarrow S \cup \{i\}$, open facility i
- 5: connect to all its neighbors to i
- 6: let i own them
- 7: connect unconnected clients to their nearest facilities in S



Construction of Integral Primal Solution

Construction of Integral Primal Solution

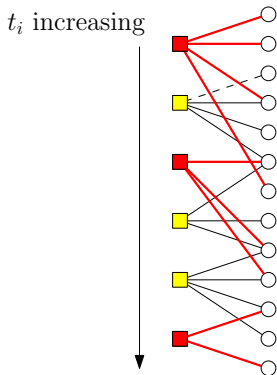
- 1: $S \leftarrow \emptyset$, all clients are **unowned**
- 2: **for** every temporarily open facility i , in increasing order of t_i **do**
- 3: **if** all (solid-line) neighbors of i are unowned **then**
- 4: $S \leftarrow S \cup \{i\}$, open facility i
- 5: connect to all its neighbors to i
- 6: let i own them
- 7: connect unconnected clients to their nearest facilities in S



Construction of Integral Primal Solution

Construction of Integral Primal Solution

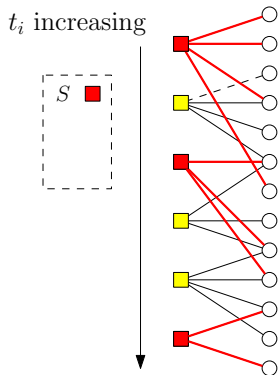
- 1: $S \leftarrow \emptyset$, all clients are **unowned**
- 2: **for** every temporarily open facility i , in increasing order of t_i **do**
- 3: **if** all (solid-line) neighbors of i are unowned **then**
- 4: $S \leftarrow S \cup \{i\}$, open facility i
- 5: connect to all its neighbors to i
- 6: let i own them
- 7: connect unconnected clients to their nearest facilities in S



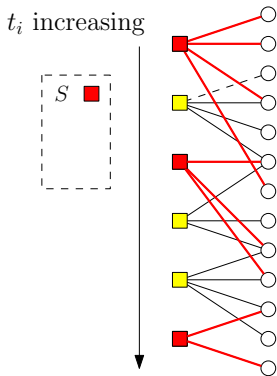
Construction of Integral Primal Solution

Construction of Integral Primal Solution

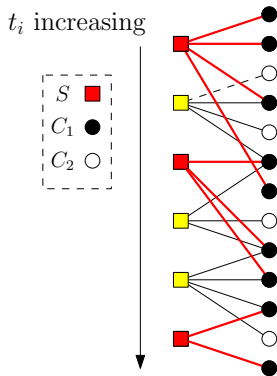
- 1: $S \leftarrow \emptyset$, all clients are **unowned**
- 2: **for** every temporarily open facility i , in increasing order of t_i **do**
- 3: **if** all (solid-line) neighbors of i are unowned **then**
- 4: $S \leftarrow S \cup \{i\}$, open facility i
- 5: connect to all its neighbors to i
- 6: let i own them
- 7: connect unconnected clients to their nearest facilities in S



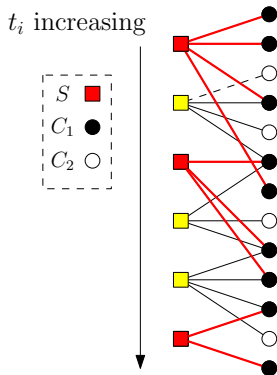
- S : set of open facilities
- C_1 : clients that make contributions
- C_2 : clients that do not make contributions



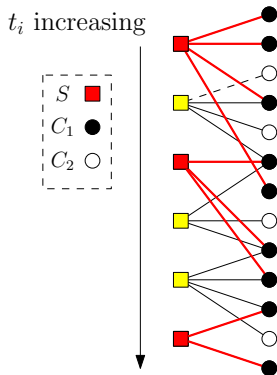
- S : set of open facilities
- C_1 : clients that make contributions
- C_2 : clients that do not make contributions



- S : set of open facilities
- C_1 : clients that make contributions
- C_2 : clients that do not make contributions
- f : total facility cost
- c_j : connection cost of client j
- $c = \sum_{j \in C} c_j$: total connection cost



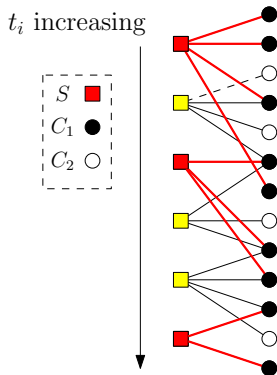
- S : set of open facilities
- C_1 : clients that make contributions
- C_2 : clients that do not make contributions
- f : total facility cost
- c_j : connection cost of client j
- $c = \sum_{j \in C} c_j$: total connection cost
- $D = \sum_{j \in C} \alpha_j$: value of α



- S : set of open facilities
- C_1 : clients that make contributions
- C_2 : clients that do not make contributions
- f : total facility cost
- c_j : connection cost of client j
- $c = \sum_{j \in C} c_j$: total connection cost
- $D = \sum_{j \in C} \alpha_j$: value of α

Lemma

- $f + \sum_{j \in C_1} c_j \leq \sum_{j \in C_1} \alpha_j$
- for any client $j \in C_2$, we have $c_j \leq 3\alpha_j$



Lemma

- $f + \sum_{j \in C_1} c_j \leq \sum_{j \in C_1} \alpha_j$
- for any client $j \in C_2$, we have $c_j \leq 3\alpha_j$

Lemma

- $f + \sum_{j \in C_1} c_j \leq \sum_{j \in C_1} \alpha_j$
- for any client $j \in C_2$, we have $c_j \leq 3\alpha_j$

• So, $f + c = f + \sum_{j \in C} c_j \leq 3 \sum_{j \in C} \alpha_j = 3D \leq 3 \cdot \text{opt}.$

Lemma

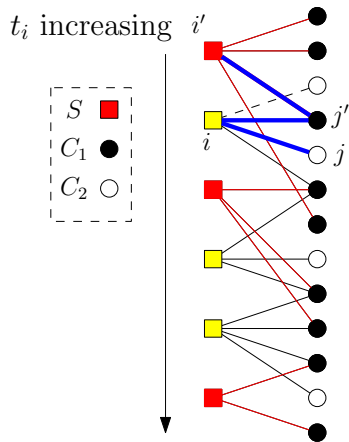
- $f + \sum_{j \in C_1} c_j \leq \sum_{j \in C_1} \alpha_j$
- for any client $j \in C_2$, we have $c_j \leq 3\alpha_j$

- So, $f + c = f + \sum_{j \in C} c_j \leq 3 \sum_{j \in C} \alpha_j = 3D \leq 3 \cdot \text{opt.}$

- stronger statement:

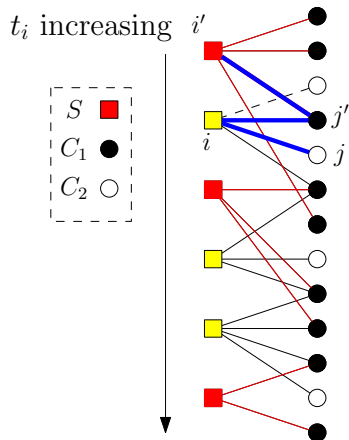
$$3f + c = 3f + \sum_{j \in C} c_j \leq 3 \sum_{j \in C} \alpha_j = 3D \leq 3 \cdot \text{opt.}$$

Proof of $\forall j \in C_2, c_j \leq 3\alpha_j$



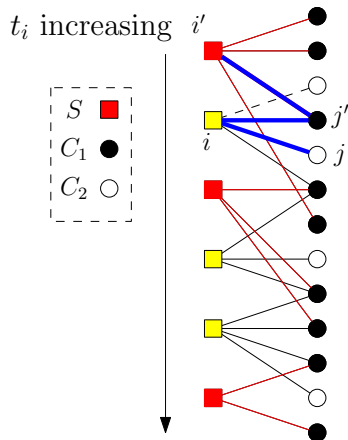
Proof of $\forall j \in C_2, c_j \leq 3\alpha_j$

- at time α_j , j is frozen.



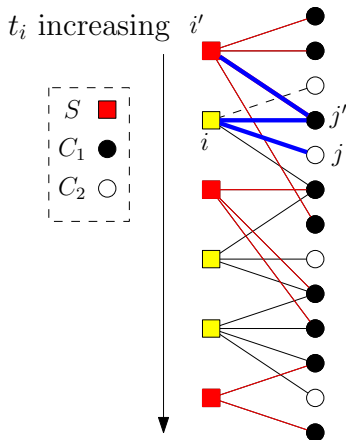
Proof of $\forall j \in C_2, c_j \leq 3\alpha_j$

- at time α_j , j is frozen.
- let i be the temporarily open facility it connects to



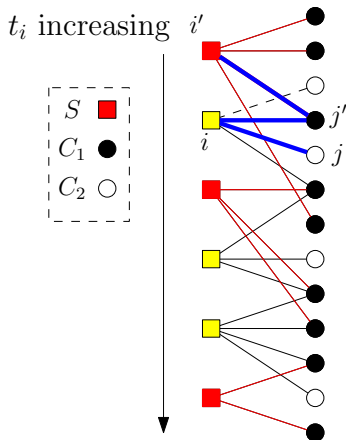
Proof of $\forall j \in C_2, c_j \leq 3\alpha_j$

- at time α_j , j is frozen.
- let i be the temporarily open facility it connects to
- $i \in S$: then $c_j \leq \alpha_j$. assume $i \notin S$.



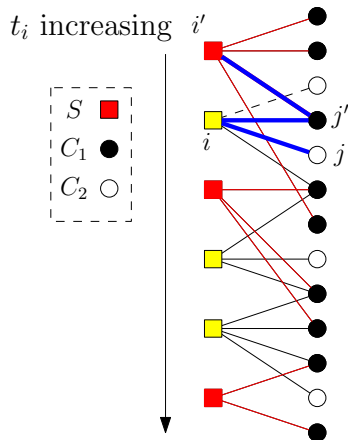
Proof of $\forall j \in C_2, c_j \leq 3\alpha_j$

- at time α_j , j is frozen.
- let i be the temporarily open facility it connects to
- $i \in S$: then $c_j \leq \alpha_j$. assume $i \notin S$.
- there exists a client j' , which made contribution to i , and owned by another facility $i' \in S$



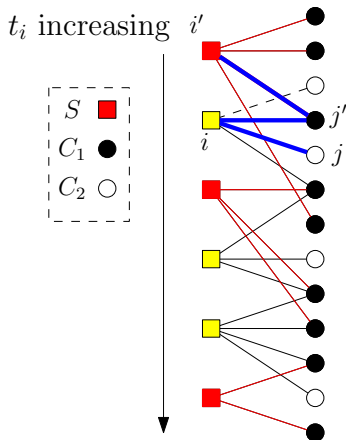
Proof of $\forall j \in C_2, c_j \leq 3\alpha_j$

- at time α_j , j is frozen.
- let i be the temporarily open facility it connects to
- $i \in S$: then $c_j \leq \alpha_j$. assume $i \notin S$.
- there exists a client j' , which made contribution to i , and owned by another facility $i' \in S$
- $d(j, i) \leq \alpha_j$



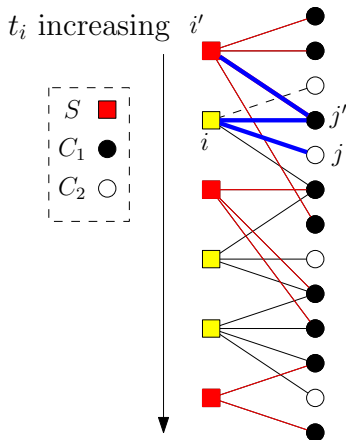
Proof of $\forall j \in C_2, c_j \leq 3\alpha_j$

- at time α_j , j is frozen.
- let i be the temporarily open facility it connects to
- $i \in S$: then $c_j \leq \alpha_j$. assume $i \notin S$.
- there exists a client j' , which made contribution to i , and owned by another facility $i' \in S$
- $d(j, i) \leq \alpha_j$
- $d(j', i) < \alpha_{j'}, d(j', i') < \alpha_{j'}$



Proof of $\forall j \in C_2, c_j \leq 3\alpha_j$

- at time α_j , j is frozen.
- let i be the temporarily open facility it connects to
- $i \in S$: then $c_j \leq \alpha_j$. assume $i \notin S$.
- there exists a client j' , which made contribution to i , and owned by another facility $i' \in S$
- $d(j, i) \leq \alpha_j$
- $d(j', i) < \alpha_{j'}, d(j', i') < \alpha_{j'}$
- $\alpha_{j'} = t'_i \leq t_i \leq \alpha_j$



Proof of $\forall j \in C_2, c_j \leq 3\alpha_j$

- at time α_j , j is frozen.
- let i be the temporarily open facility it connects to
- $i \in S$: then $c_j \leq \alpha_j$. assume $i \notin S$.
- there exists a client j' , which made contribution to i , and owned by another facility $i' \in S$
- $d(j, i) \leq \alpha_j$
- $d(j', i) < \alpha_{j'}, d(j', i') < \alpha_{j'}$
- $\alpha_{j'} = t'_i \leq t_i \leq \alpha_j$
- $d(j, i') \leq d(j, i) + d(i, j') + d(j', i') \leq \alpha_j + \alpha_j + \alpha_j = 3\alpha_j$

