Combinatorial Algorithms

Local search

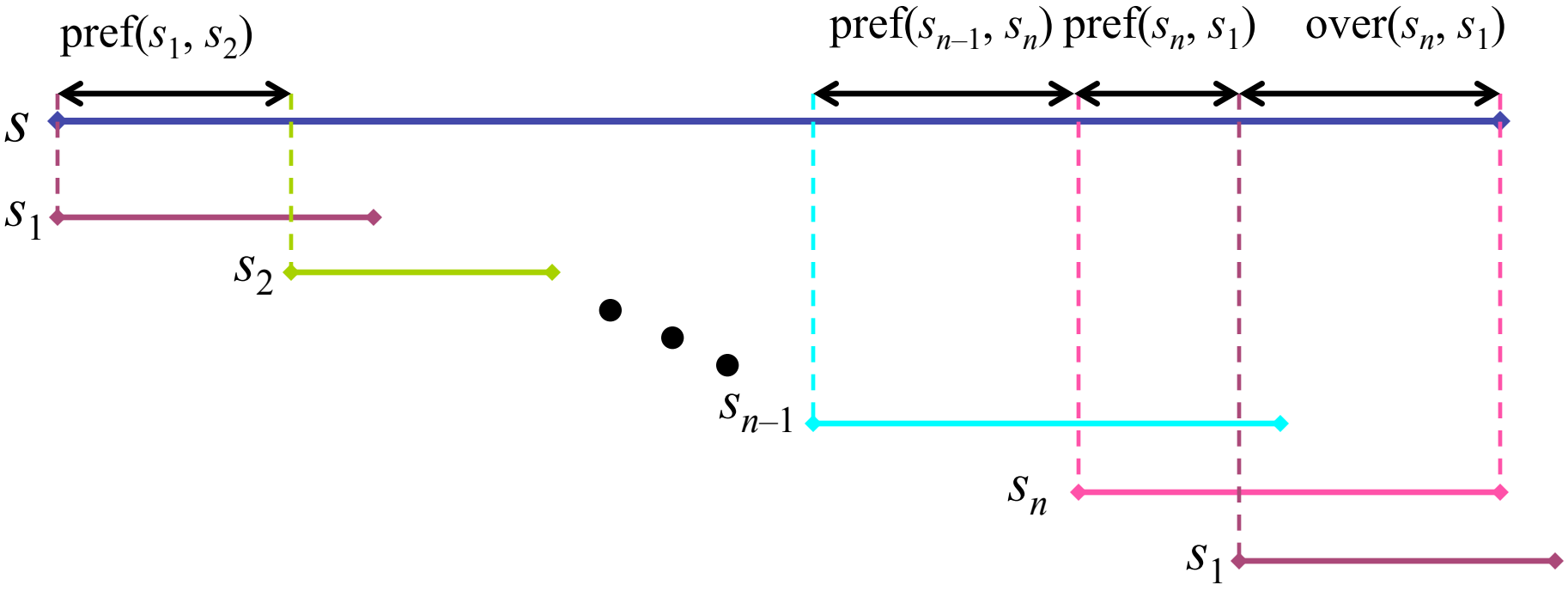
Shortest superstring

- *Given* a finite alphabet Σ, and a set of n strings $S = \{s_1, \ldots, s_n\} \subseteq \Sigma^+$.
- *Find* a shortest string s that contains each s_i as a substring.
- Without lost of generality, we may assume that no string s_i is a substring of another string s_j, $i \neq j$.

Overlap, prefix

- We begin by developing a good lower bound on OPT.
- Let us assume that $s_1, s_2, \ldots, s_n$ are numbered in order of leftmost occurrence in the shortest superstring, s.
- Let overlap(s_i, s_j) denote the maximum overlap between s_i and s_j i.e., the longest suffix of s_i that is a prefix of s_j.
- Let prefix(s_i, s_j) be the prefix of s_i obtained removing its overlap with s_j.

Prefix



$$\mathrm{OPT} = |\mathrm{prefix}(s_1, s_2)| + |\mathrm{prefix}(s_2, s_3)| + \cdots + |\mathrm{prefix}(s_n, s_1)| + |\mathrm{overlap}(s_n, s_1)|.$$

$$\mathrm{OPT} = |\mathrm{prefix}(s_1, s_2)| + |\mathrm{prefix}(s_2, s_3)| + \cdots + |\mathrm{prefix}(s_n, s_1)| + |\mathrm{overlap}(s_n, s_1)|$$

- Define the **prefix graph of** S as the directed graph G_{pref} on vertex set $V=\{1,\ldots,n\}$ that contains an edge $i \rightarrow j$ of weight $prefix(s_i,s_j)$ for each i, j.
- $|\,prefix(s_1,s_2)| + |\,prefix(s_2,s_3)| + \ldots + |\,prefix(s_n,s_1)|$ represents the weight of the tour $1 \rightarrow 2 \rightarrow \ldots \rightarrow n \rightarrow 1$.
- Hence the minimum weight of a travelling salesman tour of the prefix graph gives a lower bound on OPT.
- Unfortunately, this lower bound is not very useful. TSP is NP-hard.

Lower Bound

- We will use the minimum weight of a cycle cover of the prefix graph.
- **A cycle cover** is a collection of disjoint cycles covering all vertices.
- A Hamiltonian cycle is a cycle cover.
- We get that the minimum weight of a cycle cover lower-bounds OPT.
- Unlike minimum TSP, a minimum weight cycle cover can be computed in polynomial time.

Cycle → prefix

- If $c = (i_1 \rightarrow i_2 \rightarrow \ldots \rightarrow i_l \rightarrow i_1)$ is a cycle in the prefix graph, let $\alpha(c) = prefix(s_{i_1}, s_{i_2}) \circ \ldots \circ prefix(s_{i_{l-1}}, s_{i_l}) \circ prefix(s_{i_l}, s_{i_1})$.
- Let $w(c)$ be the weight of c, $w(c) = |\alpha(c)|$.
- Notice that each string $s_{i_1}, s_{i_2}, \ldots, s_{i_l}$ is a substring of $(\alpha(c))^{\infty}$.
- Next, let $\sigma(c) = \alpha(c) \circ s_{i_1}$.
- Then $\sigma(c)$ is a superstring of $s_{i_1}, s_{i_2}, \ldots, s_{i_l}$.
- In the above construction, we "opened" cycle c at an arbitrary string s_{i_1}. For the rest of the algorithm, we will call s_{i_1} the representative string for c.

Example

abcdeabcdeabcde
bcdeabcdeabcdea
cdeabcdeabcdeabc
deabcdeabcdeabcd
abcdeabcdeabcde

$\alpha(c) =$ abcde , $|\alpha(c)|=5$, $(\alpha(c))^2 =$ abcdeabcde ,
bcdeabcdeabcdea is a substring of $(\alpha(c))^4$.

$\sigma(c) = \alpha(c) \circ s_{i_1} =$ abcdeabcdeabcdeabcde

Algorithm Superstring

Input ($S = \{s_1, \ldots, s_n\}$)

1) Construct the prefix graph G_{pref} corresponding to strings in S.
2) Find a minimum weight cycle cover of G_{pref}, $C = \{c_1, \ldots, c_k\}$

Output ($\sigma(c_1) \circ \ldots \circ \sigma(c_k)$).

Remark

- Clearly, the output $\sigma(c_1) \circ \ldots \circ \sigma(c_k)$ is a superstring of the strings in S.
- Notice that if in each of the cycles we can find a representative string of length at most the weight of the cycle, then the string output is within 2OPT.
- Thus, the hard case is when all strings of some cycle c are long.

Example

abcde|abcde|abcde
bcde|abcde|abcde|a
cde|abcde|abcde|abc
de|abcde|abcde|abcd
abcde|abcde|abcde

$\alpha(c)$ = abcde , $|\alpha(c)|=5$, $(\alpha(c))^2$ = abcdeabcde , bcdeabcdeabcdea is a substring of $(\alpha(c))^4$.

$\sigma(c) = \alpha(c) \circ s_{i_1}$ = abcde|abcde|abcde|abcde

New lower bound

- **Lemma 4.6**

  If each string in $S' \subseteq S$ is a substring of t^{∞} for a string t, then there is a cycle of weight at most $|t|$ in the prefix graph covering all the vertices corresponding to string in S'.

Proof of Lemma 4.6

- For each string in S', locate the starting point of its first occurrence in t^{∞} .
- All these starting points will be distinct and will lie in the first copy of t.
- Consider the cycle in the prefix graph visiting the corresponding vertices in this order.
- Clearly, the weight of this cycle is at most $|t|$.

Lower bound on overlap

- **Lemma 4.7**

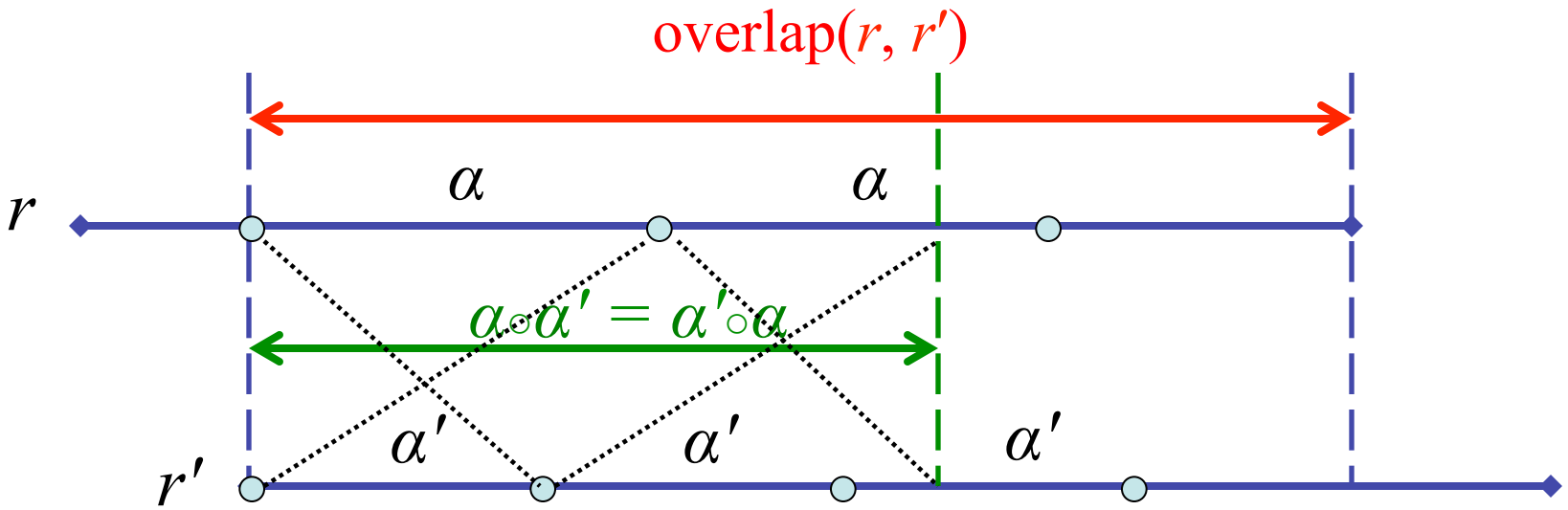
  Let c and c' be two cycles in C (cyclic cover of the minimal weight), and let r, r' be representative strings from these cycles. Then

  $$|\text{overlap}(r, r')| < w(c) + w(c').$$

$|\text{overlap}(r, r')| \geq w(c) + w(c')$

α is a prefix of length $w(c)$ of overlap (r, r').

α' is a prefix of length $w(c')$ of overlap (r, r').

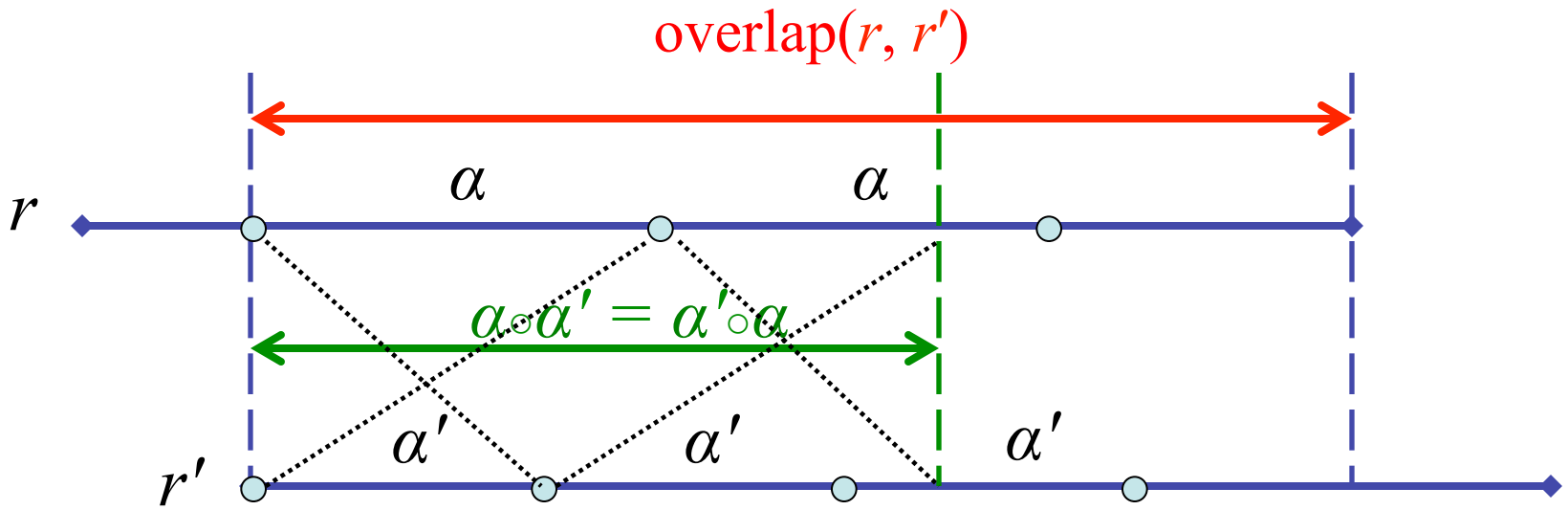


Since $|\text{overlap}(r, r')| \geq w(c) + w(c')$, it is follows that α and α' commute.

$$|\text{overlap}(r, r')| \geq w(c) + w(c').$$

α is a prefix of length $w(c)$ of overlap (r, r').

α' is a prefix of length $w(c')$ of overlap (r, r').



$(\alpha)^\infty = (\alpha')^\infty$

For any N > 0, the prefix of length N of $(\alpha)^\infty$ is the same as that of $(\alpha')^\infty$.

Proof of Lemma 4.7

- Now, by Lemma 4.6, there is a cycle of weight at most $w(c)$ in the prefix graph covering all strings in c and c', contradicting the fact that C is a minimum weight cycle cover.
- So, we have $|\text{overlap}(r, r')| < w(c) + w(c')$.

Approximation ratio of Algorithm Superstring

Theorem 4.8

Algorithm Superstring achieves an approximation factor of 4 for the shortest superstring problem.

Algorithm Superstring

Input ($S = \{s_1, \ldots, s_n\}$)

1) Construct the prefix graph G_{pref} corresponding to strings in S.
2) Find a minimum weight cycle cover of G_{pref}, $C = \{c_1, \ldots, c_k\}$

Output ($\sigma(c_1) \circ \ldots \circ \sigma(c_k)$).

Proof

$$w(C) = \sum_{i=1}^{k} w(c_i) \le \mathrm{OPT}$$

$$A = \sum_{i=1}^{k} \sigma(c_i) = w(C) + \sum_{i=1}^{k} |r_i|$$

r_i is a representative string for c_i.

$string^*: \ldots r_1, \ldots, r_2, \ldots, r_k, \ldots$

$$\mathrm{OPT} \ge \sum_{i=1}^{k} |r_i| - \sum_{i=1}^{k-1} |\mathrm{overlap}(r_i, r_{i+1})| \ge$$

$$\overset{L\,4.7}{\ge} \sum_{i=1}^{k} |r_i| - 2\sum_{i=1}^{k} w(c_i)$$

$$\sum_{i=1}^{k} |r_i| \le 3\,\mathrm{OPT}$$

$$A \le 4\,\mathrm{OPT}$$

Local Search

- A local search algorithm starts with an arbitrary feasible solution to the problem, and then check if some small, local change to the solution results in an improved objective function.
- If so, the change is made.
- When no further change can be made, we have a *locally optimal solution*, and it is sometimes possible to prove that such locally optimal solutions have value close to that of optimal solution.

Uncapacitated Facility Location Problem

- *Given* a set of clients D and a set of facilities F, with a facility cost f_i for each facility $i \in F$, and an assignment cost c_{ij} for each facility $i \in F$ and each client $j \in D$.
- *Find* a subset $H \subseteq F$ and an assignment φ: $D \rightarrow H$ so as to minimize the total cost of the open facilities plus the assignment costs.

$$\sum_{i \in H} f_i + \sum_{j \in D} c_{j\varphi(j)} \rightarrow \min$$

Metric UFLP

- We assume that the set of clients and potential facility locations are in a metric space.
- That is, for each $i, j \in F \cup D$, we have a value c_{ij}, and for each $i, j, k \in F \cup D$, we have that $c_{i,k} \leq c_{i,j} + c_{j,k}$.
- Note that whenever we consider a distance between $i \in F$ and $j \in D$, we will maintain the convention that it is referred to as c_{ij}.

An assignment of clients

- Let the set of open facilities be given.
- Assign each client to its nearest open facility.
- We obtain an optimal assignment for the given set of open facilities.

$$X_H = \sum_{i \in H} f_i \qquad X^* = X_{H^*}$$

$$Y_H = \sum_{j \in D} c_{j\varphi(j)} \qquad Y^* = Y_{H^*}$$

Neighborhood $N(H)$

H, φ_H:

1. Open one additional facility $H := H \cup \{i\}$, $i \in F \setminus H$,
2. Close one facility that is currently open $H := H \setminus \{i\}$, $i \in H$.
3. Open a new facility and close an open facility
   $H := H \cup \{i\} \setminus \{j\}$, $i \in F \setminus H$, $j \in H$.

Update the current assignment of clients to open facilities. The algorithm will always maintain that each client is assigned to its nearest open facility.

Local Search Algorithm

Input $(G, f: F \rightarrow \mathbf{Q}^+, c: E \rightarrow \mathbf{Q}^+)$

1) Choose an arbitrary current solution H.

2) **While** there exists a solution $H' \in N(H)$ such that $X_H + Y_H > X_{H'} + Y_{H'}$ **do** $H := H'$.

Output (H, φ_H)

Locally optimal solution

- We want to analyze the quality of the solution found by the Local Search Algorithm.
- A solution obtained by this algorithm is said to be a **locally optimal solution**.
- We will focus not on an algorithmic statement but instead on proving that **any** locally optimal solution is near-optimal.

Total assignment cost

Lemma 5.1

Let H and φ_H be a locally optimal solution. Then $Y_H \leq X^* + Y^* = \mathrm{OPT}$.

.

Proof of Lemma 5.1(1)

- Since H is a locally optimal solution, we know that adding any facility to H does not improve the solution (with respect to the best possible updated assignment).
- In this way we will focus on a few potential changes to the current solution, and analyze their change in cost.
- Note that we consider the changes only for the sake of analysis, and we do not actually change the solution.

Proof of Lemma 5.1(2)

Case 1. Consider some facility $i^* \in H^* - H$.

Suppose we open the additional facility i^*, and reassign to that facility all of the clients that were assigned to i^* in the optimal solution: that is we reassign all clients j such that $\varphi^*(j) = i^*$. Since our current solution H and $\varphi(H)$ is locally optimal, we know that the additional facility cost of i^* is at least as much as the improvement in cost that would result from reassigning each client optimally to its nearest open facility; hence, f_{i^*} must also be more than the improvement resulting from our specific reassignment; that is,

$$f_{i^*} \geq \sum_{j:\varphi^*(j)=i^*} \left(c_{j\varphi(j)} - c_{j\varphi^*(j)} \right)$$

Proof of Lemma 5.1(3)

Case 2. Consider a facility $i^* \in H^* \cap H$.

The local optimality of H and $\varphi(H)$ implies that each client j is currently assigned to its closest open facility, and so each term in the summation below must be nonpositive,

$$\sum_{j:\varphi^*(j)=i^*} \left(c_{j\varphi(j)} - c_{j\varphi^*(j)} \right) \leq 0 \leq f_{i^*}.$$

- Summing over all facilities in the optimal solution, we obtain

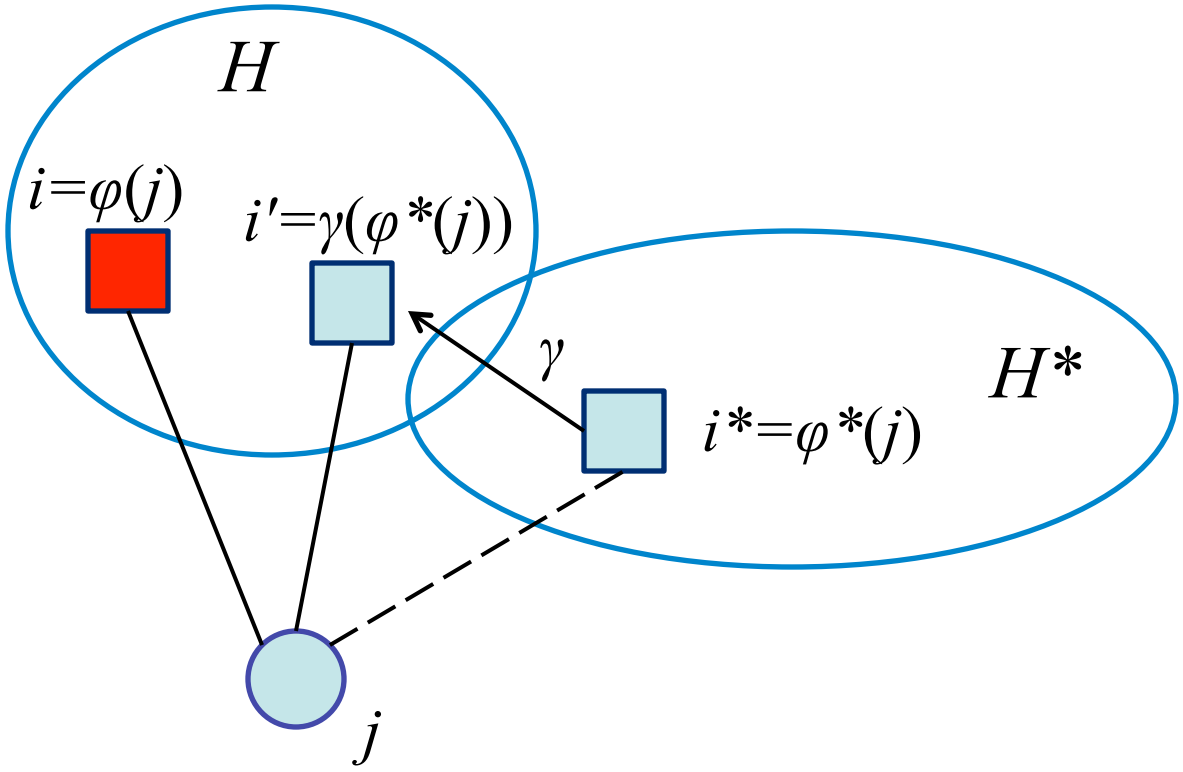
$$\sum_{i^* \in H^*} f_{i^*} \geq \sum_{i^* \in H^*} \sum_{j:\varphi^*(j)=i^*} \left(c_{j\varphi(j)} - c_{j\varphi^*(j)} \right) = \sum_{j \in D} \left(c_{j\varphi(j)} - c_{j\varphi^*(j)} \right).$$

$$X^* \geq Y_H - Y^*$$

Total facility cost

- The argument to show that a local optimum has a small total facility cost is somewhat more complicated. As in the proof of the previous lemma, we will consider a set of changes to the solution H, each of which will generate a corresponding inequality.
- For any move that deletes a facility $i \in H$, we must reassign each of the clients that are assigned to i. If we were simply deleting i, then each such client must be reassigned to a facility in $H - \{i\}$.

Reassignment of client j to facility $i' = \gamma(\varphi^*(j))$.



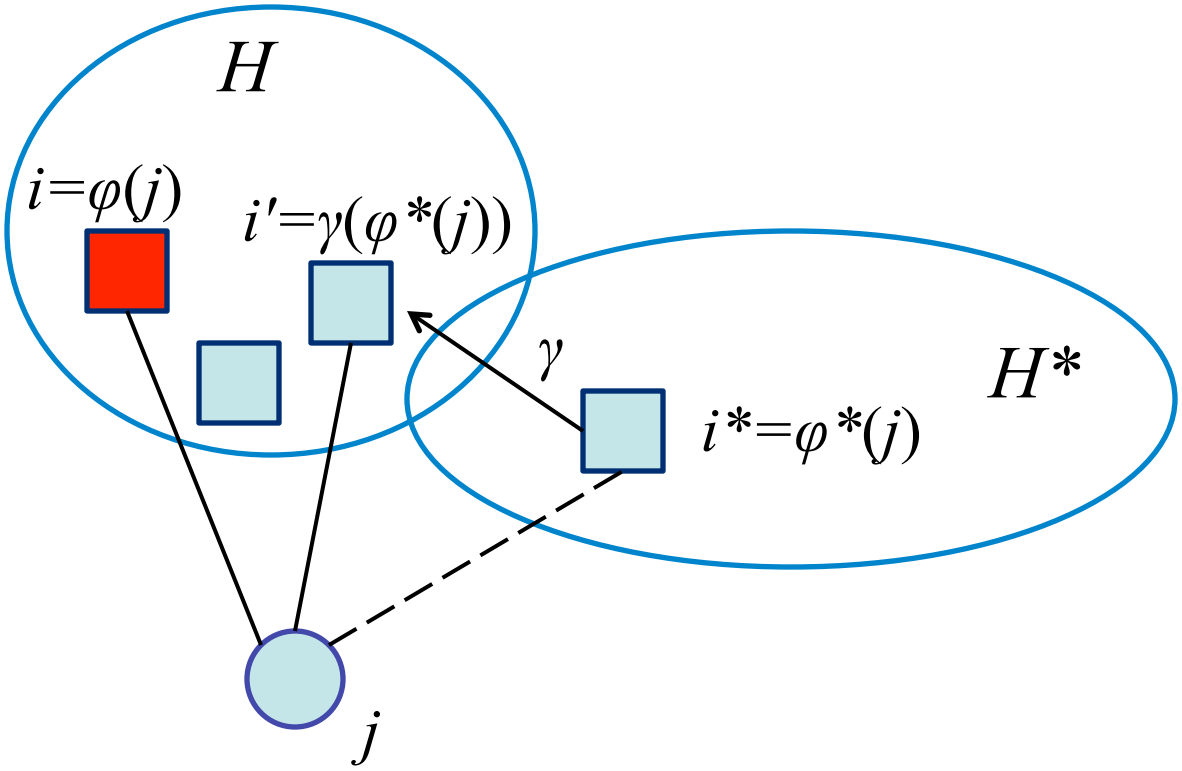
$\gamma(\varphi^*(j))$ is the facility in H closest to $\varphi^*(j)$.

Cost of Reassigning

Lemma 5.2

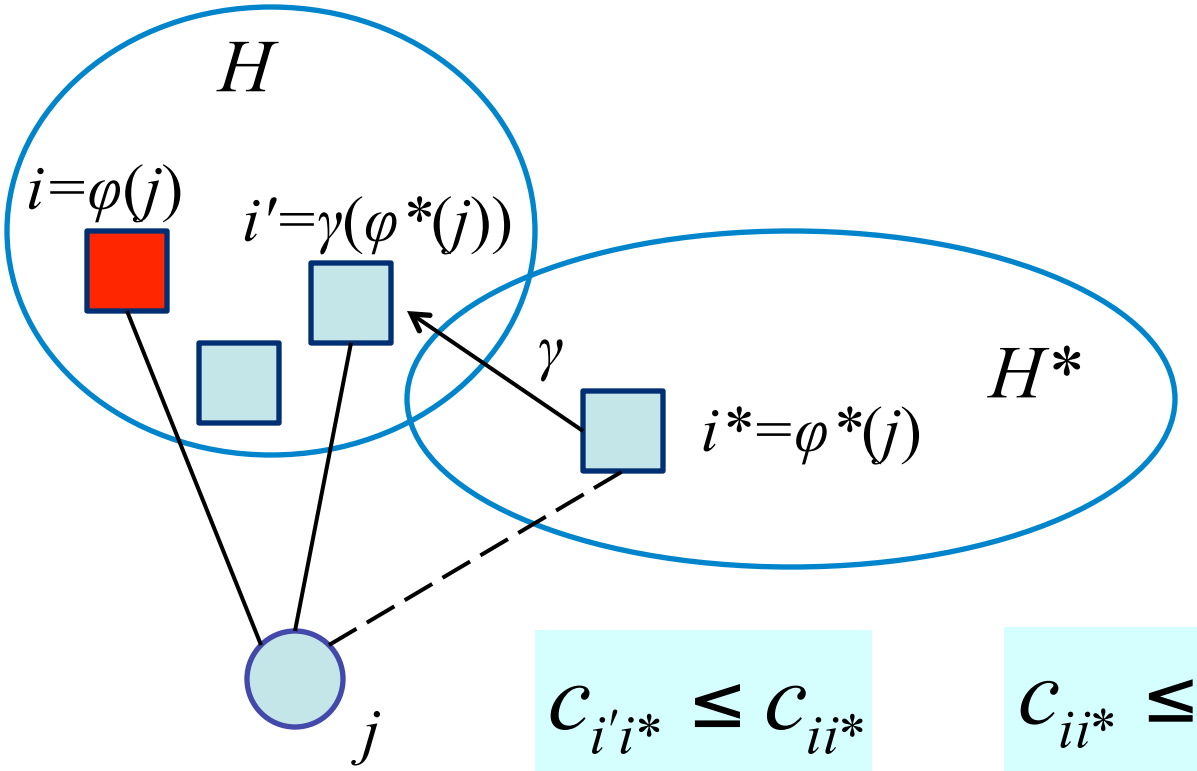
Consider any client j for which $\varphi(j) = i$ is not equal to $i' = \gamma(\varphi^*(j))$. Then the increase in cost of reassigning client j to i' (instead of to i) is at most $2c_{j,\varphi^*(j)}$.

Proof of Lemma 5.2



Consider a client j currently being served by i, where its facility in H^*, $i^* = \varphi^*(j)$, is such that i^*'s nearest facility in H, $\gamma(i^*)$, is not the facility i. Let $i' = \gamma(\varphi^*(j))$.

Proof of Lemma 5.2



$$c_{i'i^*} \le c_{ii^*} \qquad c_{ii^*} \le c_{ji} + c_{ji^*}$$

$$c_{ji'} \le c_{i'i^*} + c_{ji^*} \le c_{ii^*} + c_{ji^*} \le c_{ji} + 2c_{ji^*}$$

$$c_{ji'} - c_{ji} \le 2c_{ji^*}$$

Cost of Reassigning

Lemma 5.2

Consider any client j for which $\varphi(j) = i$ is not equal to $i' = \gamma(\varphi^*(j))$. Then the increase in cost of reassigning client j to i' (instead of to i) is at most $2c_{j,\varphi^*(j)}$.

We will apply this lemma both when i is deleted and i is swapped out of the solution.

An upper bound on X_H

Lemma 5.3

Let H and φ_H be a locally optimal solution. Then $X_H \leq X^* + 2Y^*$.

.

Proof of Lemma 5.3(2)

- In our proof, we will give a set of moves that either deletes or swaps out every facility in H (once each) and either adds or swaps in every facility in H^* (again once each).
- Since the change in cost for each of these local moves in nonnegative, this will allow us to bound the facility cost H in terms of the facility H^* and additional terms that we will bound by twice the optimal assignment cost.

Proof of Lemma 5.3(1)

- Suppose that we want to delete a facility $i \in H$.
- Each client j, that is currently served by i must be reassigned to one of the remaining open facilities in $H - \{i\}$.
- We shall call a facility i **safe**, if for every facility $i^* \in H^*$, the facility $\gamma(i^*) \in H$ closest to i^* is different from i.
- For any safe facility i, we can consider the local move of closing facility i, since we can safely reassign each of its client j to $\gamma(\varphi^*(j))$, and apply Lemma 5.2 to bound the resulting increase in the assignment cost for reassigned client j by $2c_{j,\varphi^*(j)}$.

Bound on “safe” facilities

- Since H is locally optimal, we know that this local change cannot decrease the overall cost, and hence the savings obtained by closing the safe facility i must be no more than the increase in assignment costs incurred by reassigning all of the clients assigned to i. That is,

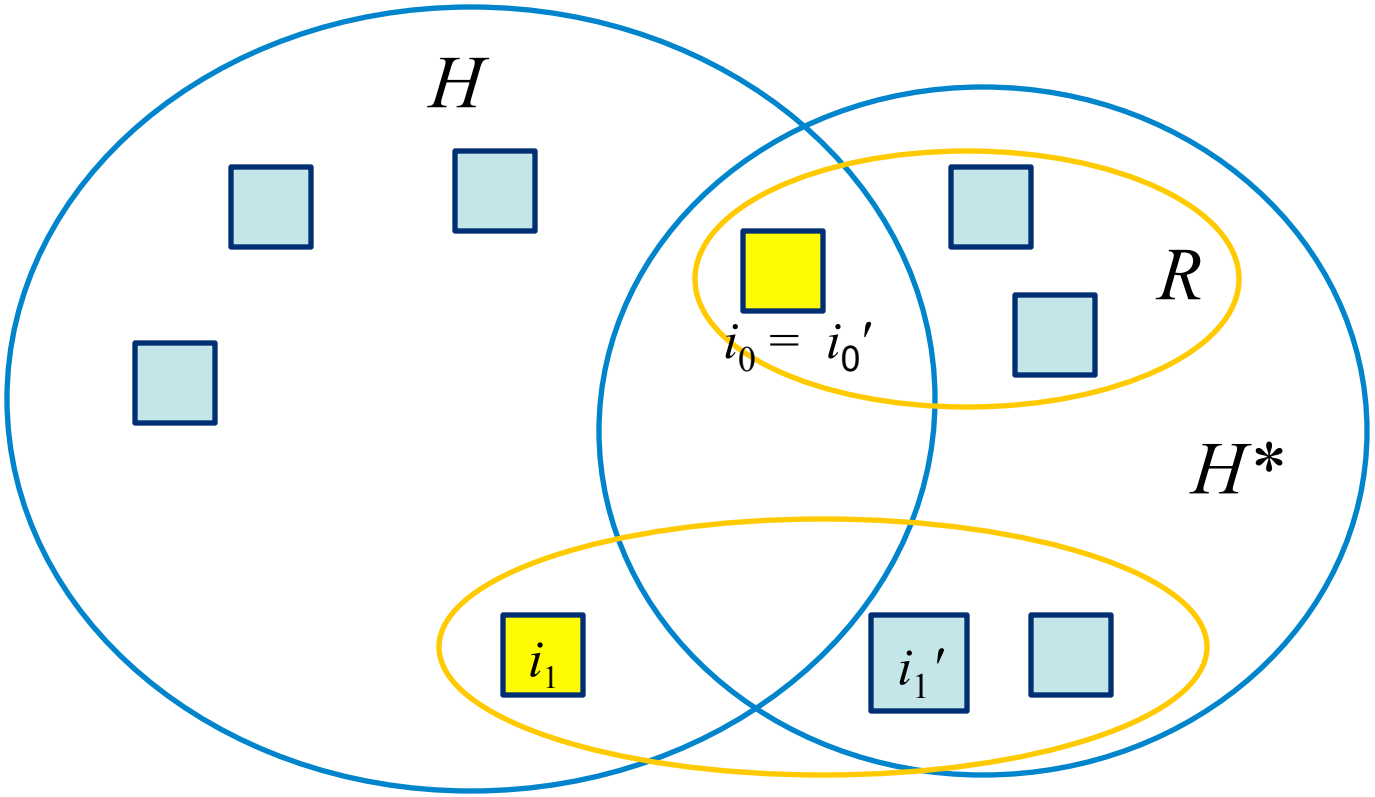
$$f_i \leq \sum_{j:\varphi(j)=i} 2c_{j\varphi^*(j)},$$

$$-f_i + \sum_{j:\varphi(j)=i} 2c_{j\varphi^*(j)} \geq 0.$$

Unsafe facilities

- Consider a facility i that is not safe.
- Let $R_i \subseteq H^*$ be the (nonempty) set of facilities $i^* \in H^*$ such that $\gamma(i^*) = i$, i.e. $R_i = \{i^* \in H^* | \gamma(i^*) = i\}$.
- Among those facilities in R_i, let i' be the one closest to i.
- We will derive one inequality for each member of R_i, based on
  - an add move for each member of $R_i - \{i'\}$,
  - one swap move closing the facility at i, while opening a facility at i'.

Unsafe facilities



– unsafe facilities

Each facility $i^* \in H^*$ occurs in exactly one corresponding set R.

Add move for $i^* \in R_i - \{i'\}$

- We open a facility at i^*, and for each client j that is assigned to i in the locally optimal solution ($\varphi(j) = i$) and is assigned to i^* in the optimal solution ($\varphi^*(j) = i^*$), we reassign client j to i^*.

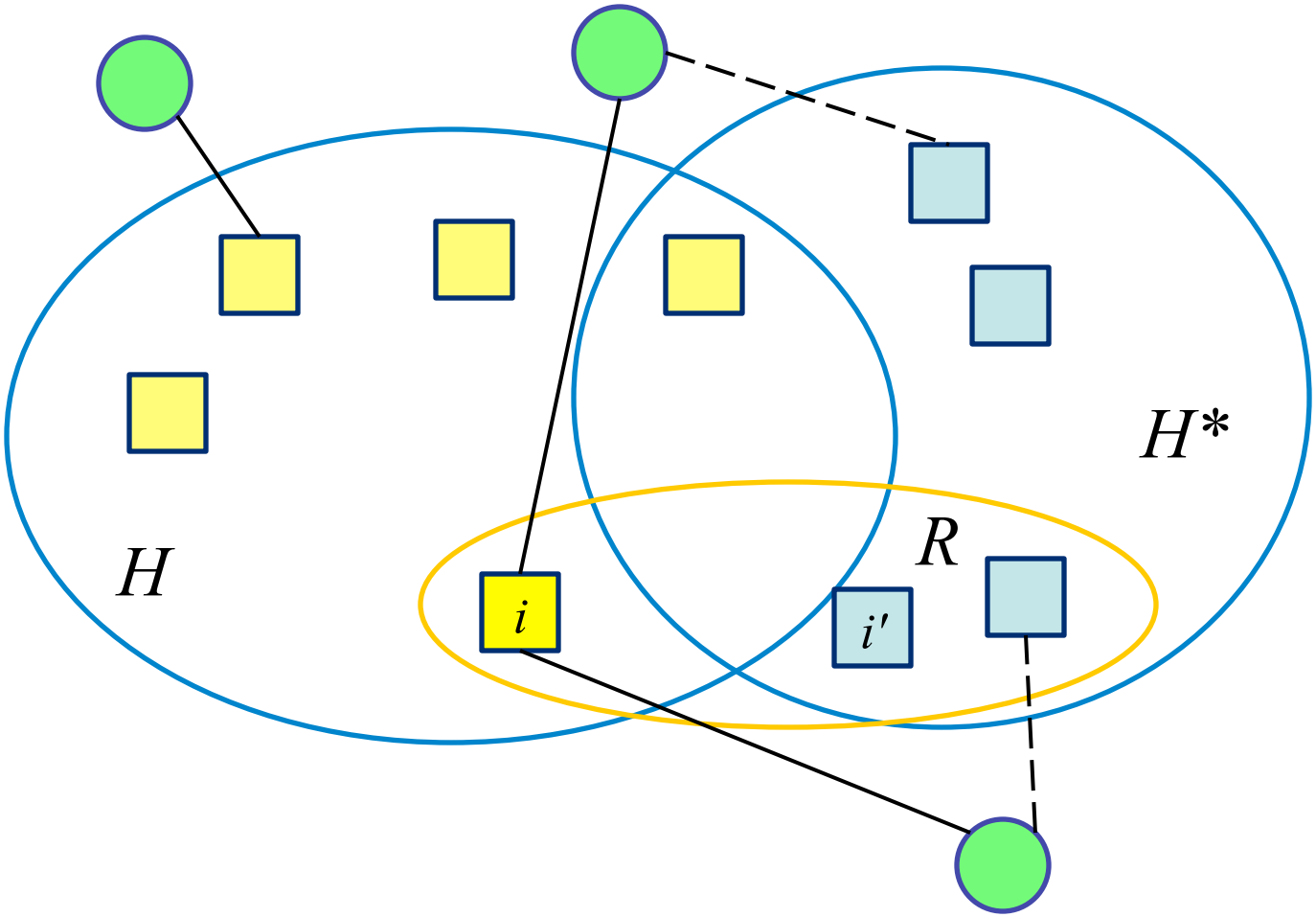
- The change in cost caused by this move must also be nonnegative, and we derive the inequality

$$f_{i^*} + \sum_{j:\varphi(j)=i \,\&\, \varphi^*(j)=i^*} \left(c_{j,\varphi^*(j)} - c_{j,\varphi(j)} \right) \geq 0.$$

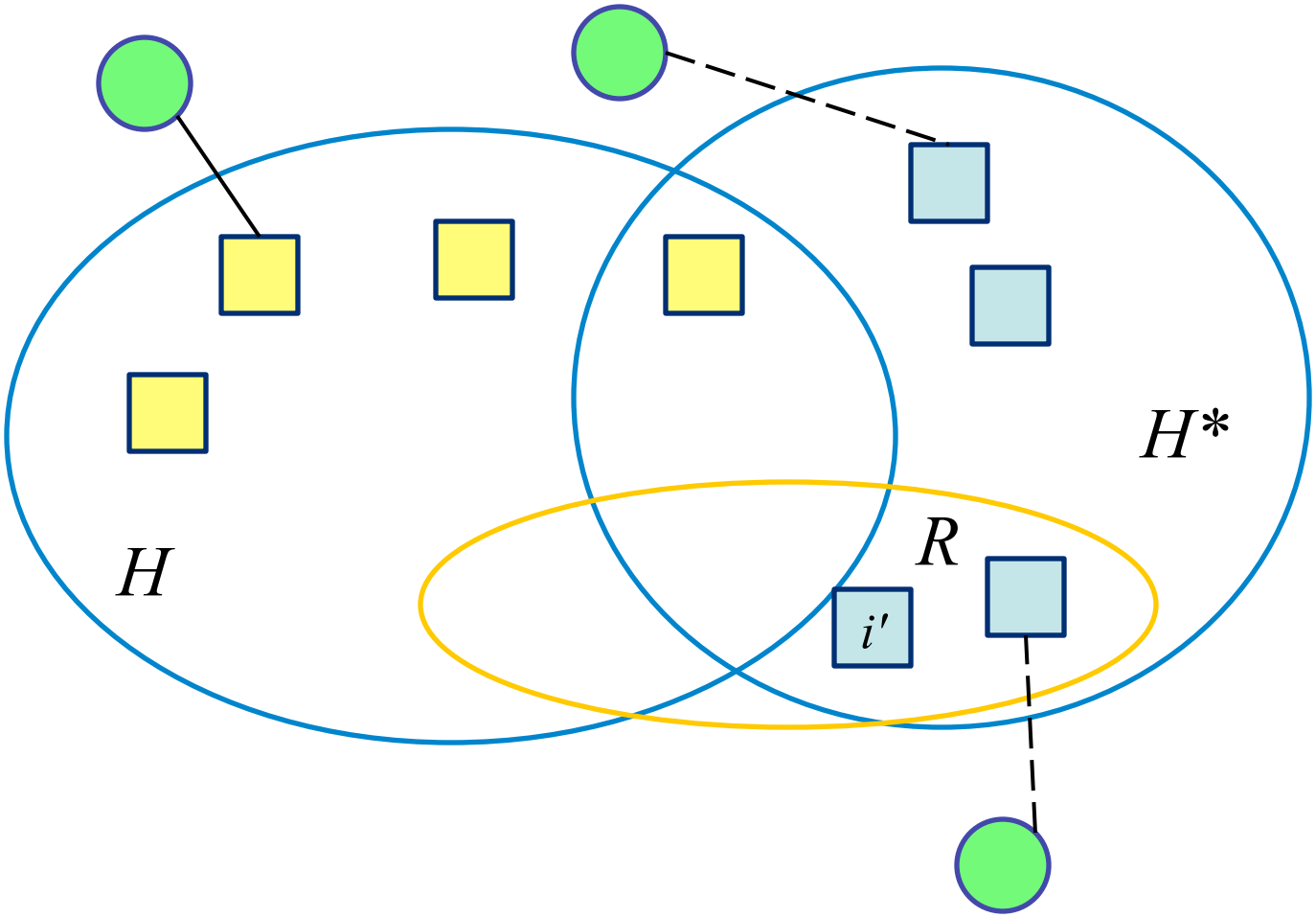
Swap move that closes the facility at i but opens a facility i' ($i \neq i'$).

- To make this swap move precise, we will also specify a reassignment of the clients assigned to i by φ. Each client j such that $\varphi(j) = i$
  - for which $\varphi^*(j) \notin R$ is reassigned to $\gamma(\varphi^*(j))$,
  - for which $\varphi^*(j) \in R$ is reassigned to i'.

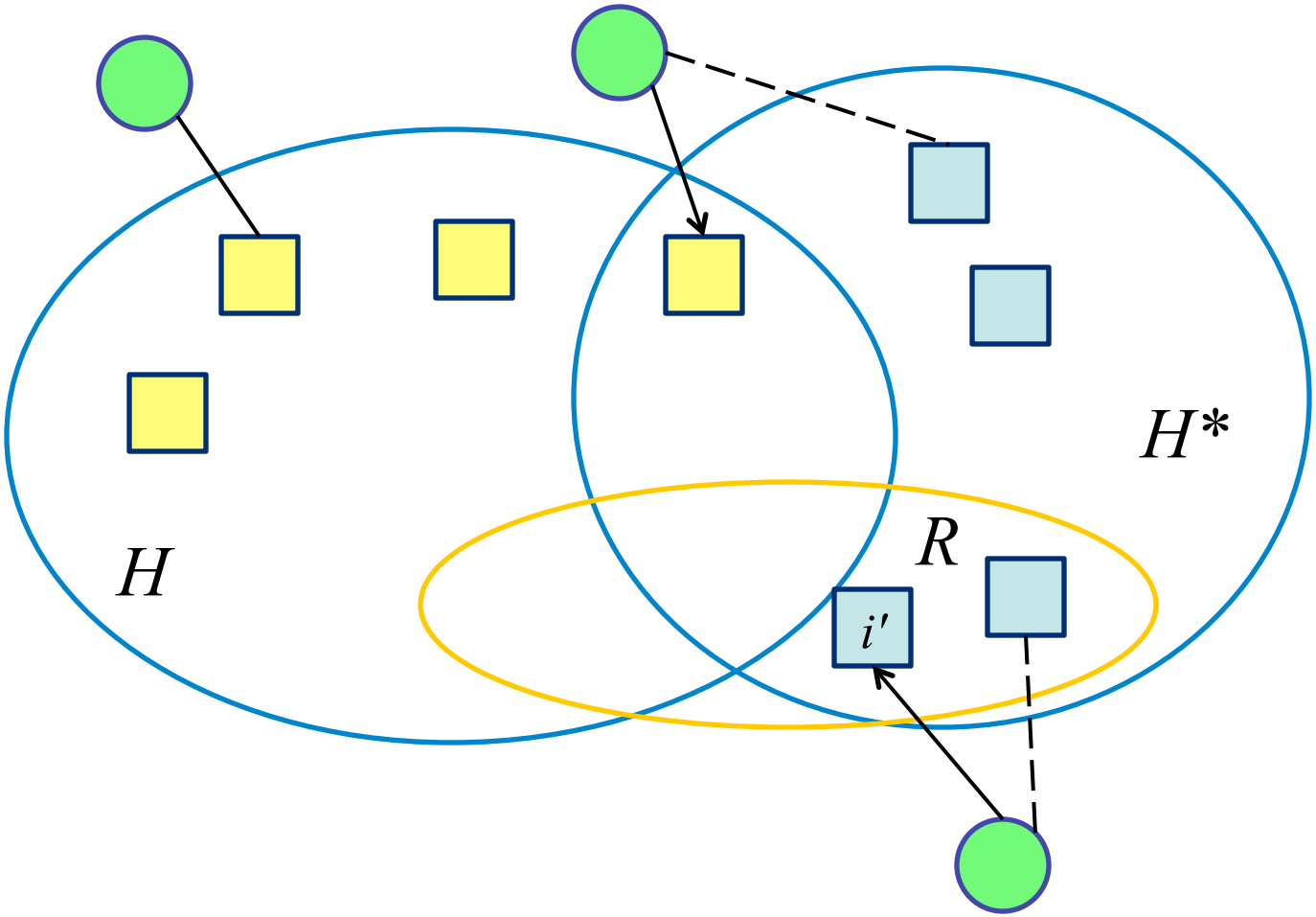
Swap i to i', $i \neq i'$.



Swap i to i', $i \neq i'$.



Swap i to i', $i \neq i'$.



Inequalities based on the swap move

- Close the facility at i and open a facility i': $f_{i'} - f_i$.
- Consider client j such that $\varphi(j) = i$.
  - If $\varphi^*(j) \notin R$ then j is reassigned to $\gamma(\varphi^*(j))$:

    Lemma 5.2 $\Rightarrow$ the increase in the cost $\leq 2c_{j,\varphi^*(j)}$.
  - If $\varphi^*(j) \in R$ then j is reassigned to i'. It follows that the change in the assignment cost is exactly $c_{ji'} - c_{ji}$.

  Combining all of these pieces, we obtain an upper bound on the total change in cost of this swap move. Again, we know that the true change in cost is nonnegative, and hence

$$f_{i'} - f_i + \sum_{j:\varphi(j)=i \,\&\, \varphi^*(j) \notin R} 2c_{j\varphi(j)} + \sum_{j:\varphi(j)=i \,\&\, \varphi^*(j) \in R} \left(c_{ji'} - c_{ji}\right) \geq 0$$

How about if $i = i'$

$$f_{i'} - f_i + \sum_{j:\varphi(j)=i\&\varphi^*(j)\notin R} 2c_{j\varphi(j)} + \sum_{j:\varphi(j)=i\&\varphi^*(j)\in R} \left(c_{ji'} - c_{ji}\right) \geq 0$$

Suppose that $i = i'$; the above inequality reduces to the essentially trivial inequality that

$$\sum_{j:\varphi(j)=i\&\varphi^*(j)\notin R} 2c_{j\varphi(j)} \geq 0$$

Net effect

$$f_{i'} - f_i + \sum_{j:\varphi(j)=i \& \varphi^*(j) \notin R} 2c_{j\varphi(j)} + \sum_{j:\varphi(j)=i \& \varphi^*(j) \in R} \left(c_{ji'} - c_{ji} \right) \geq 0$$

$$f_{i^*} + \sum_{j:\varphi(j)=i \& \varphi^*(j)=i^*} \left(c_{j,\varphi^*(j)} - c_{j,\varphi(j)} \right) \geq 0 \qquad i^* \in R - \{i'\}$$

For unsafe facility i, let us consider the net effect of combining all of these inequalities. Adding these, we get that

$$-f_i + \sum_{i^* \in R} f_{i^*} + \sum_{j:\varphi(j)=i \& \varphi^*(j) \notin R} 2c_{j\varphi^*(j)} +$$

$$\sum_{j:\varphi(j)=i \& \varphi^*(j) \in R} \left(c_{ji'} - c_{ji} \right) + \sum_{j:\varphi(j)=i \& \varphi^*(j) \in R - \{i'\}} \left(c_{j\varphi^*(j)} - c_{j\varphi(j)} \right) \geq 0$$

Simplification

$$-f_i + \sum_{i^* \in R} f_{i^*} + \sum_{j:\varphi(j)=i \,\&\, \varphi^*(j) \notin R} 2c_{j\varphi^*(j)} +$$

$$\sum_{j:\varphi(j)=i \,\&\, \varphi^*(j) \in R} \left(c_{ji'} - c_{ji}\right) + \sum_{j:\varphi(j)=i \,\&\, \varphi^*(j) \in R - \{i'\}} \left(c_{j\varphi^*(j)} - c_{j\varphi(j)}\right) \geq 0$$

We will simplify the above expression by combining the final two summations, and by showing that for each client j that appears in either summation, we can upper bound its total contribution by $2c_{j\varphi^*(j)}$.

- $\varphi(j)=i$ & $\varphi^*(j)=i'$: $c_{ji'} - c_{ji} \leq 2c_{ji'} = 2c_{j\varphi^*(j)}$.
- $\varphi(j)=i$ & $\varphi^*(j) \in R - \{i'\}$: $c_{ji'} + c_{j\varphi^*(j)} - 2c_{ji} \leq$ $c_{ii'} + c_{j\varphi^*(j)} - c_{ji} \leq c_{i\varphi^*(j)} + c_{j\varphi^*(j)} - c_{ji} \leq 2c_{j\varphi^*(j)}$.

$$-f_i + \sum_{i^* \in R} f_{i^*} + \sum_{j:\varphi(j)=i} 2c_{j\varphi^*(j)} \geq 0$$

Proof of Lemma 5.3(2)

- Safe facility i: $-f_i + \sum_{j:\varphi(j)=i} 2c_{j\varphi^*(j)} \geq 0$
- Unsafe facility i: $-f_i + \sum_{i^* \in R} f_{i^*} + \sum_{j:\varphi(j)=i} 2c_{j\varphi^*(j)} \geq 0$

$$\sum_{i^* \in H^*} f_{i^*} - \sum_{i \in H} f_i + \sum_{j \in D} 2c_{j\varphi^*(j)} \geq 0$$

$$X^* - X_H + 2Y^* \geq 0$$

Total cost of a locally optimal solution

- **Theorem 5.4**

  Let H and φ_H be a locally optimal solution for the uncapacitated facility location problem. Then this solution has a total cost that is at most 3OPT.

- Proof.
- $Y_H \leq X^* + Y^*$ (Lemma 5.1)
- $X_H \leq X^* + 2Y^*$ (Lemma 5.3)
- $X_H + Y_H \leq 2X^* + 3Y^* \leq 3\mathrm{OPT}$.

Exercise

- Consider the following algorithm:

Input ($G, f: F \to \mathbf{Q}^+, c: E \to \mathbf{Q}^+$)

1) Increase the cost of each facility by a factor 2.
1) Choose an arbitrary current solution H.
2) **While** there exists a solution $H' \in N(H)$ such that $X_H + Y_H > X_{H'} + Y_{H'}$ **do** $H := H'$.

Output (H, φ_H)

- Find an upper bound on the cost a solution obtained by the above algorithm.