L5: Auction algorithm for Min-Cost Flow

FEL3250: Network Optimization

Chathuranga Weeraddana

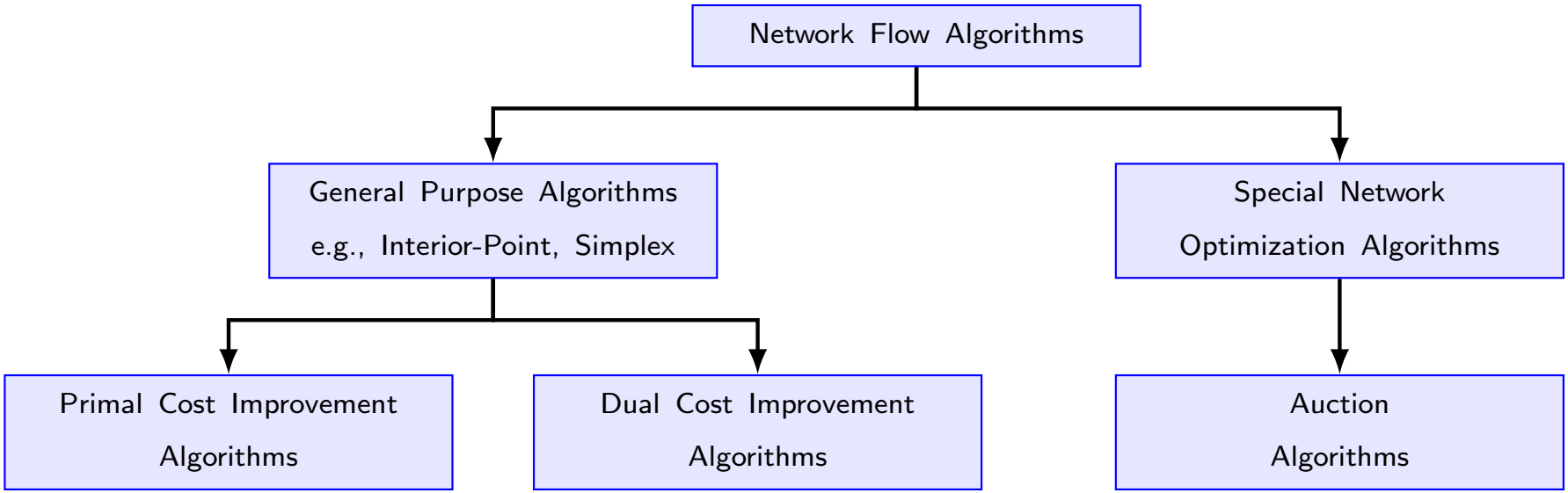
20 February 2014

Outline

- Introduction (big picture)
- Assignment problem (brief outline)
- Naive auction algorithm for the assignment problem
- Auction algorithm for the assignment problem
- Auction algorithms for the Min-Cost-Flow

Introduction
(Big Picture)

Solve Network Flow Problems



- *primal cost improvement*: construct negative cost simple cycles along which flow can be pushed while being capacity feasible
- *dual cost improvement*: solve the dual problem to find optimal prices, which are connected to the optimal primal flows via CS
- *auction*: reminiscent of real-life auctions which generate a sequence of prices, which are connected to primal flows via $\epsilon-$CS

Why Auction is Preferable to Others

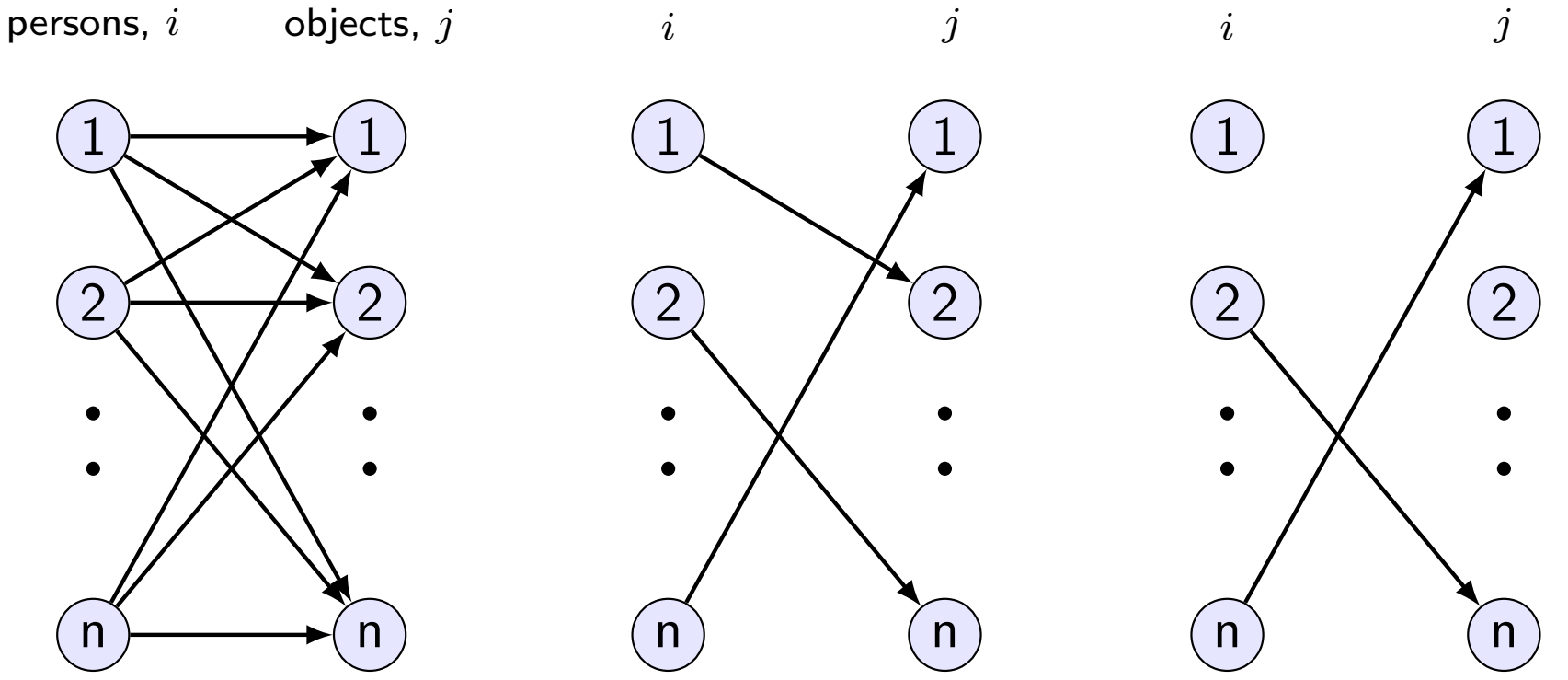
- admits a totally asynchronous implementation
- well suited for implementation on parallel machines
- even in the asynchronous case optimality is guaranteed under mild assumptions

Bertsekas, D. P., and Castañon, D. A., 1991. "Parallel Synchronous and Asynchronous Implementations of the Auction Algorithm," Parallel Computing, Vol. 17, pp. 707-732.

Assignment problem
(Brief outline)

Assignment Problem

- recall: Min-Cost Flow $\Leftrightarrow$ assignment problem



Assignment Problem

- recall: Min-Cost Flow $\Leftrightarrow$ assignment problem
- assignment problem (primal):

$$\begin{aligned} \text{maximize} \quad & \sum_{(i,j)\in\mathcal{A}} a_{ij}x_{ij} \\ \text{subject to} \quad & \sum_{\{j|(i,j)\in\mathcal{A}\}} x_{ij} = 1, \ \forall\, i = 1,\ldots,n \\ & \sum_{\{i|(i,j)\in\mathcal{A}\}} x_{ij} = 1, \ \forall\, j = 1,\ldots,n \\ & 0 \le x_{ij} \le 1, \ \forall\, (i,j) \in \mathcal{A}\ . \end{aligned}$$

- dual problem:

$$\text{minimize} \quad \sum_{i=1:n} q_i(p) + \sum_{j=1:n} p_j \ , \ \text{where}$$

$$q_i(p) = \max_{j\in A(i)} \{a_{ij} - p_j\}$$

- CS condition:

$$a_{ij_i} = \max_{j\in A(i)} \{a_{ij} - p_j\}, \quad \forall\, i = 1,\ldots,n\ .$$

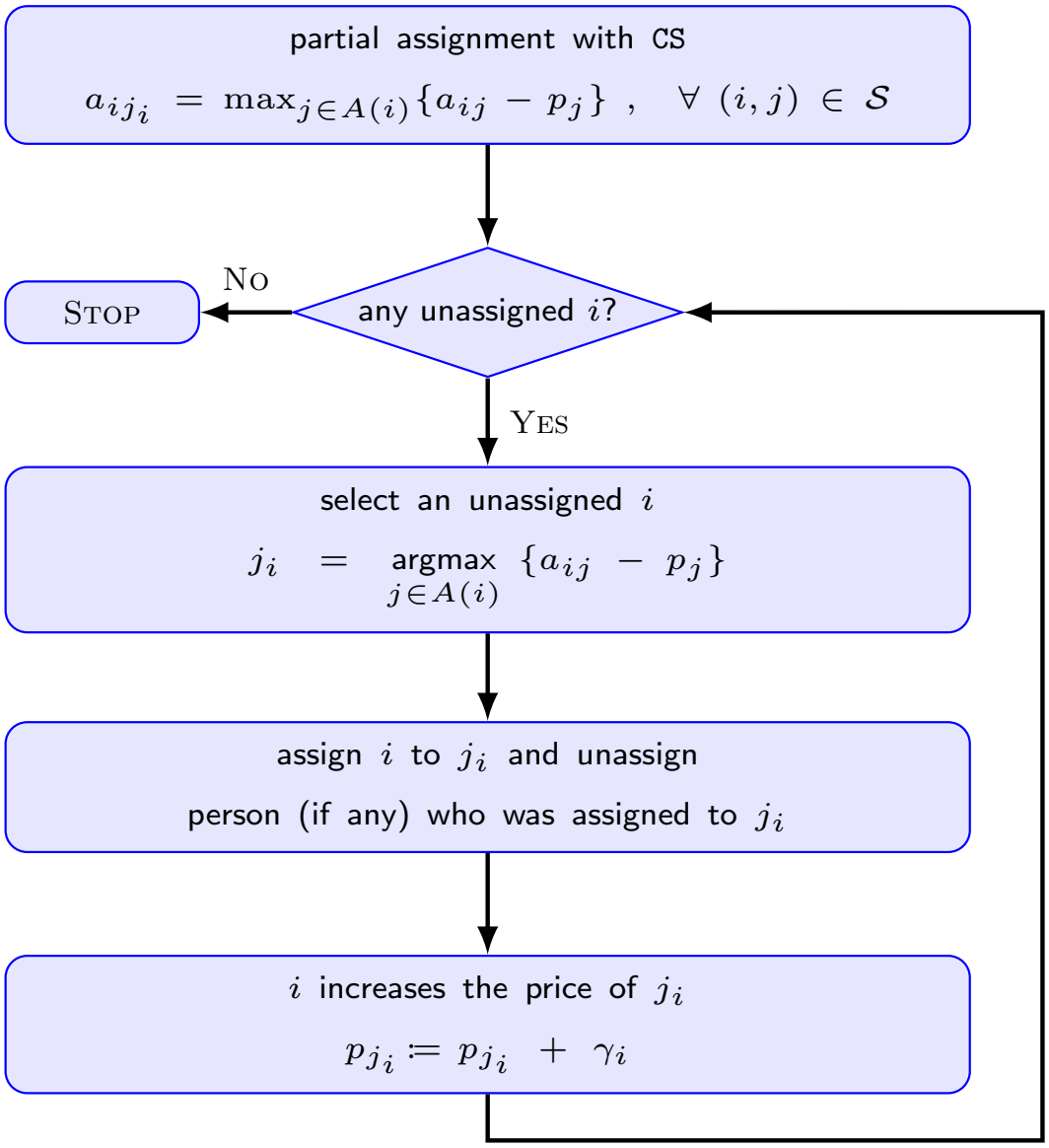
Naive Auction for Assignment Problem

Naive Auction

- it resembles the real-life auctions
- bidder i increases the price of its best object (say j_i) so that he/she is indifference between j_i and the second best object
- the algorithm relies on the CS condition:

$$a_{ij_i} - p_{j_i} = \max_{j \in A(i)} \{a_{ij} - p_j\}, \quad \forall\, i = 1, \ldots, n\ .$$

Naive Auction



initialization

check if any unassigned persons (potential bidders)

select a bidder and find the best object for it

$a_{ij_i} - p_{j_i} = \max_{j \in A(i)}\{a_{ij} - p_j\}$

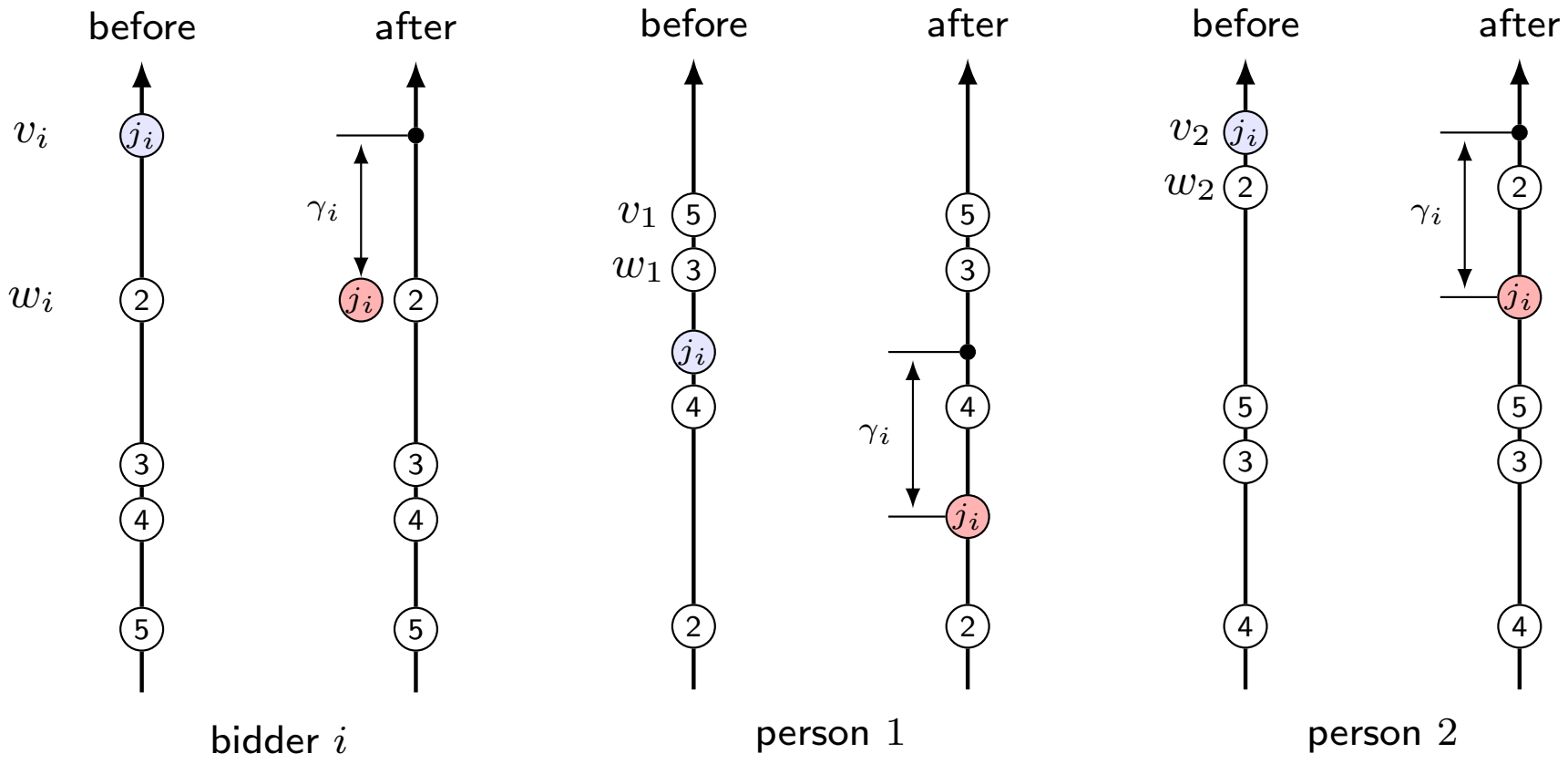
assignment/unassignment

price increase by the bidder

Naive Auction

price Increase by the Bidder i

$$\gamma_i = \underbrace{\max_{j \in A(i)} \{a_{ij} - p_j\}}_{v_i} - \underbrace{\max_{j \in A(i), j \neq j_i} \{a_{ij} - p_j\}}_{w_i}$$

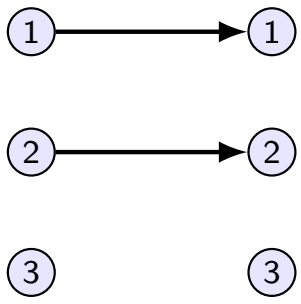


Naive Auction

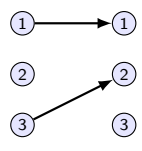
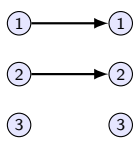
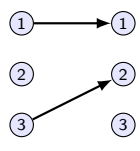
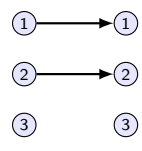
- drawback: naive auction does not always work
- if bidding increment is $0 \Rightarrow$ problems, algorithm goes in cycles

Naive Auction

example: any person can be assigned to any object



Here $a_{ij} = C > 0$ for all (i, j) with $i = 1, 2, 3$ and $j = 1, 2$ and $a_{ij} = 0$ for all (i, j) with $i = 1, 2, 3$ and $j = 3$

p_1	0	0	0	0
p_2	0	0	0	0
p_3	0	0	0	0
bidder	3	2	3	2
object	2	2	2	2
increment (γ_i)	0	0	0	0
assignment				

never ending cycles are created

Auction Algorithm for Assignment Problem

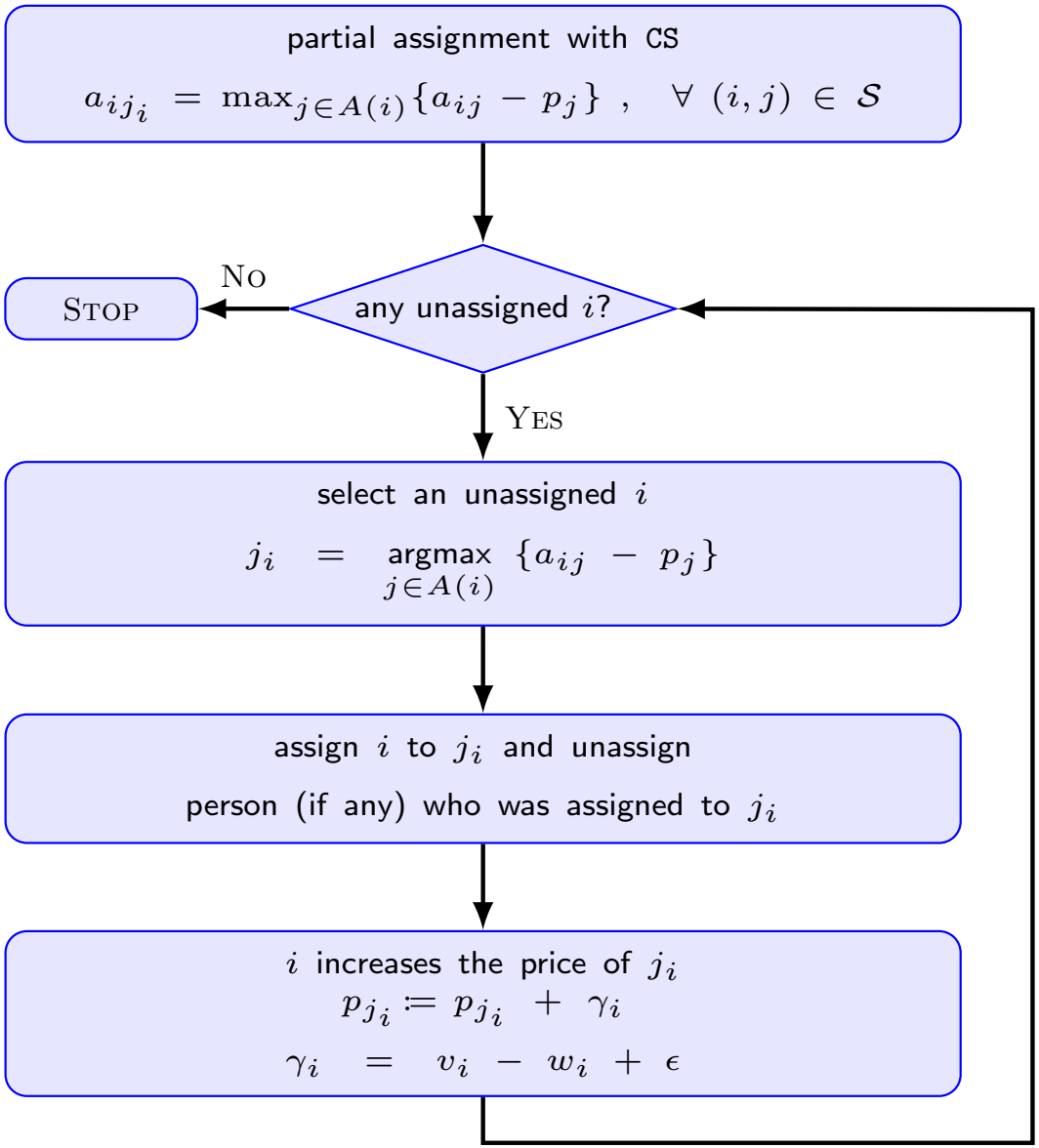
Auction Algorithm

- it resembles the real-life auctions
- bidder i increases the price of its best object (say j_i) at least by ϵ
- the algorithm relies on the $\epsilon-$CS condition:

$$a_{ij} - p_j \geq \max_{j \in A(i)} \{a_{ij} - p_j\} - \epsilon, \quad \forall\ i = 1, \ldots, n\ .$$

a perturbation mechanism is introduced

Auction Algorithm



initialization

check if any unassigned persons (potential bidders)

select a bidder and find the best object for it

$a_{ij_i} - p_{j_i} = \max_{j \in A(i)}\{a_{ij} - p_j\}$

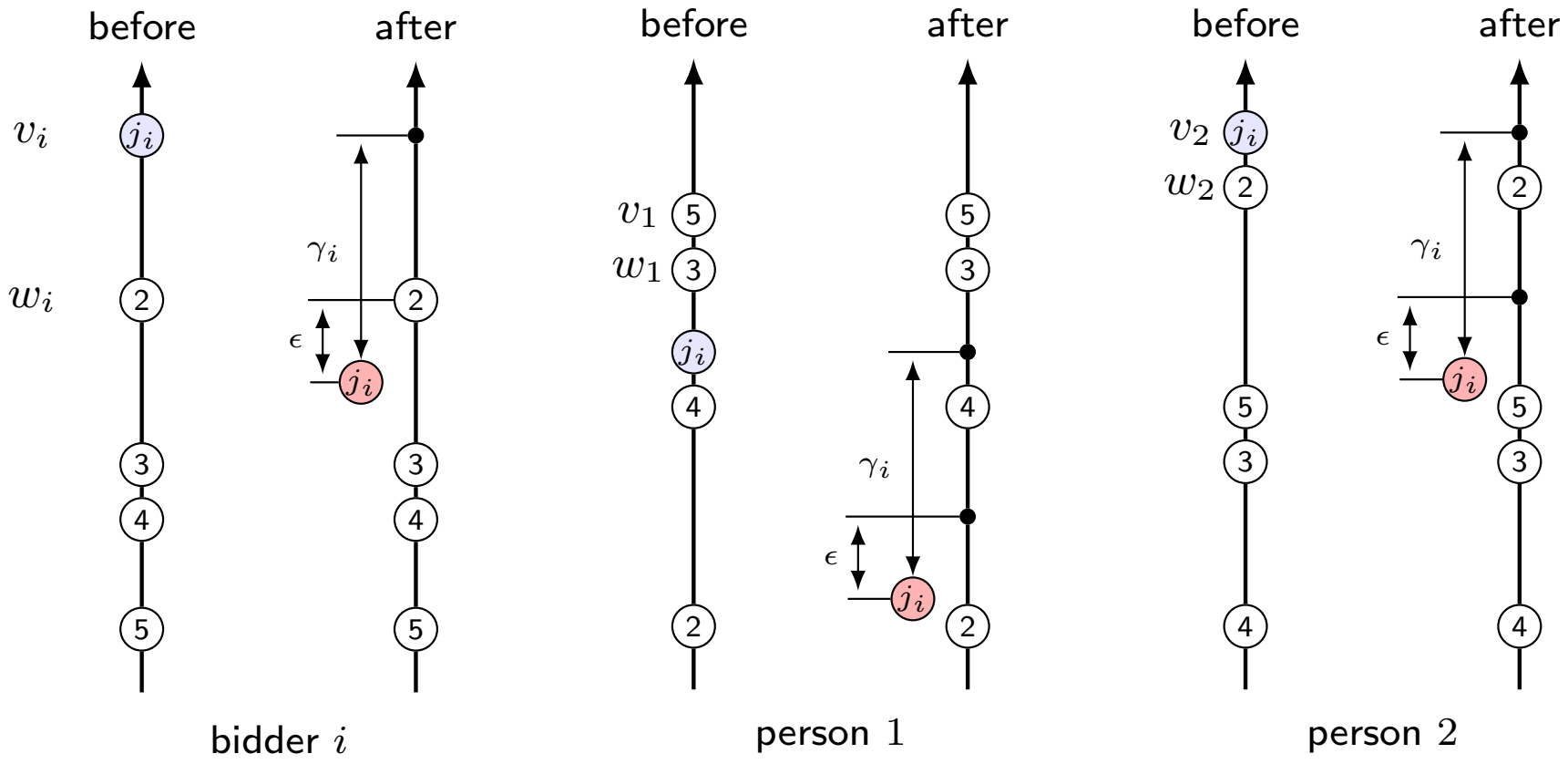
assignment/unassignment

price increase by the bidder

Auction Algorithm

price Increase by the Bidder i

$$\gamma_i = \underbrace{\max_{j \in A(i)} \{a_{ij} - p_j\}}_{v_i} - \underbrace{\max_{j \in A(i), j \neq j_i} \{a_{ij} - p_j\}}_{w_i} + \epsilon$$



Auction Algorithm

- algorithm terminates
- solution upon termination satisfies $\epsilon-\texttt{CS}$
- suboptimality is bounded

Proposition

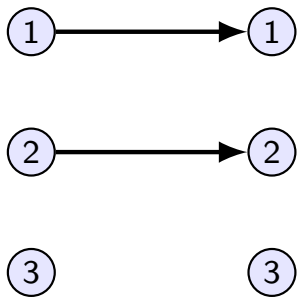
If at least one feasible assignment exists, the auction algorithm terminates with a feasible assignment that is within $n\epsilon$ of being optimal (and is optimal if the problem data are integer and $\epsilon < 1/n$).

Proof:

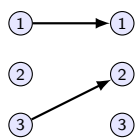
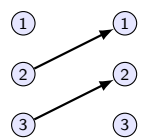
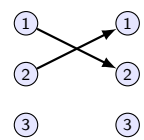
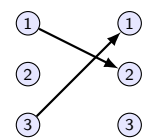
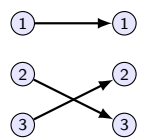
See pp 256-257 of the textbox.

Auction Algorithm

example: any person can be assigned to any object



Here $a_{ij} = C = 3\epsilon > 0$ for all (i, j) with $i = 1, 2, 3$ and $j = 1, 2$ and $a_{ij} = 0$ for all (i, j) with $i = 1, 2, 3$ and $j = 3$

p_1	0	0	2ϵ	2ϵ	4ϵ	4ϵ
p_2	0	ϵ	ϵ	3ϵ	3ϵ	3ϵ
p_3	0	0	0	0	0	ϵ
bidder	3	2	1	3	2	
object	2	1	2	1	3	
increment (γ_i)	$3\epsilon-3\epsilon+\epsilon=\epsilon$	$3\epsilon-2\epsilon+\epsilon=2\epsilon$	$2\epsilon-\epsilon+\epsilon=2\epsilon$	$\epsilon-0+\epsilon=2\epsilon$	$0-0+\epsilon=\epsilon$	
assignment						

Auction Algorithm

- *approximate coordinate descent interpretation*
  - will be discussed in an exercise
- *dealing with infeasibility*
  - will be discussed in an exercise

Auction: Variants and Extensions

- **variants:**

  1. *Gauss-Seidel version*: bidding is done by one unassign person and the object is assigned to the bidder (good for serial computing environments)

  2. *Jacobi version*: bidding is done by many unassign person and the object is assigned to the highest bidder (good for parallel computing environments)

- **extensions:** reverse auction algorithm

  - identical to normal algorithm with the roles of persons and objects are reversed

Auction Algorithms for Min-Cost Flow

Auction Algorithm, Min-Cost Flow

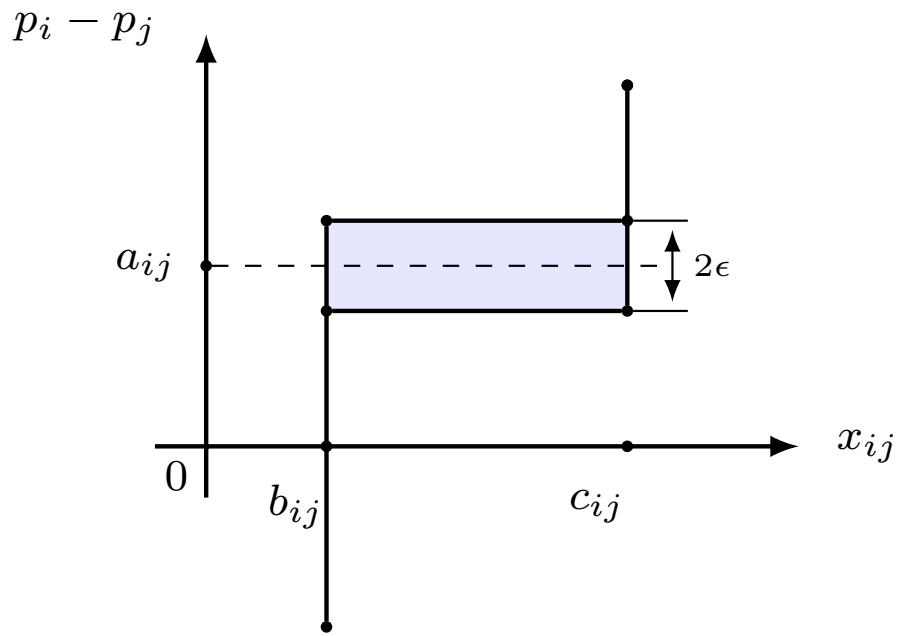
- $\epsilon-$relaxation method (our main focus)
- auction/sequential shortest path algorithm (ASSP)

Auction Algorithm, Min-Cost Flow

- the algorithm relies on the $\epsilon-$CS condition:

$$p_i - p_j \leq a_{ij} + \epsilon, \quad \forall\ (i,j) \in \mathcal{A} \ \text{ with } \ x_{ij} < c_{ij}$$

$$p_j - p_i \leq a_{ji} - \epsilon, \quad \forall\ (j,i) \in \mathcal{A} \ \text{ with } \ b_{ji} < x_{ji}$$



Auction Algorithm, Min-Cost Flow

- some terminologies ...
- given any flow x the surplus g_i of node i is

$$g_i = s_i - \underbrace{\left(\sum_{\{j|(i,j)\in\mathcal{A}\}} x_{ij} - \sum_{\{j|(j,i)\in\mathcal{A}\}} x_{ji} \right)}_{\text{divergence}}$$

- given (x, p) with $\epsilon-$CS, arc $(i.j)$ is

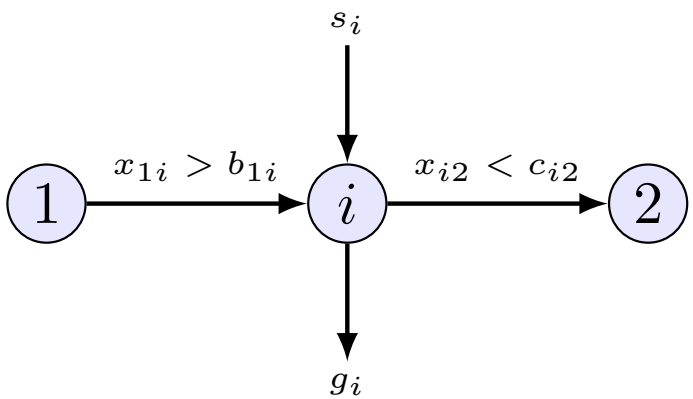
$$\epsilon^+\texttt{-unblocked} \text{ if } p_i = p_j + a_{ij} + \epsilon \text{ and } x_{ij} < c_{ij}$$

$$\epsilon^-\texttt{-unblocked} \text{ if } p_i = p_j - a_{ji} + \epsilon \text{ and } b_{ji} < x_{ji}$$

- `candidate list` of node i: outgoing arcs (i, j) that are ϵ^+`-unblocked` and incoming arcs (j, i) that are ϵ^-`-unblocked`

Auction Algorithm, Min-Cost Flow

- key idea ...
- given a pair (x, p) satisfying ϵ-CS
- $\exists\ i$ such that $g_i > 0$
- associated with this node i, only two possibilities:
  - candidate list (CL) is nonempty $\Rightarrow$ push up/down x_{ij}/x_{ji}
  - CL is empty $\Rightarrow$ make a price change $\Rightarrow$ CL become nonempty



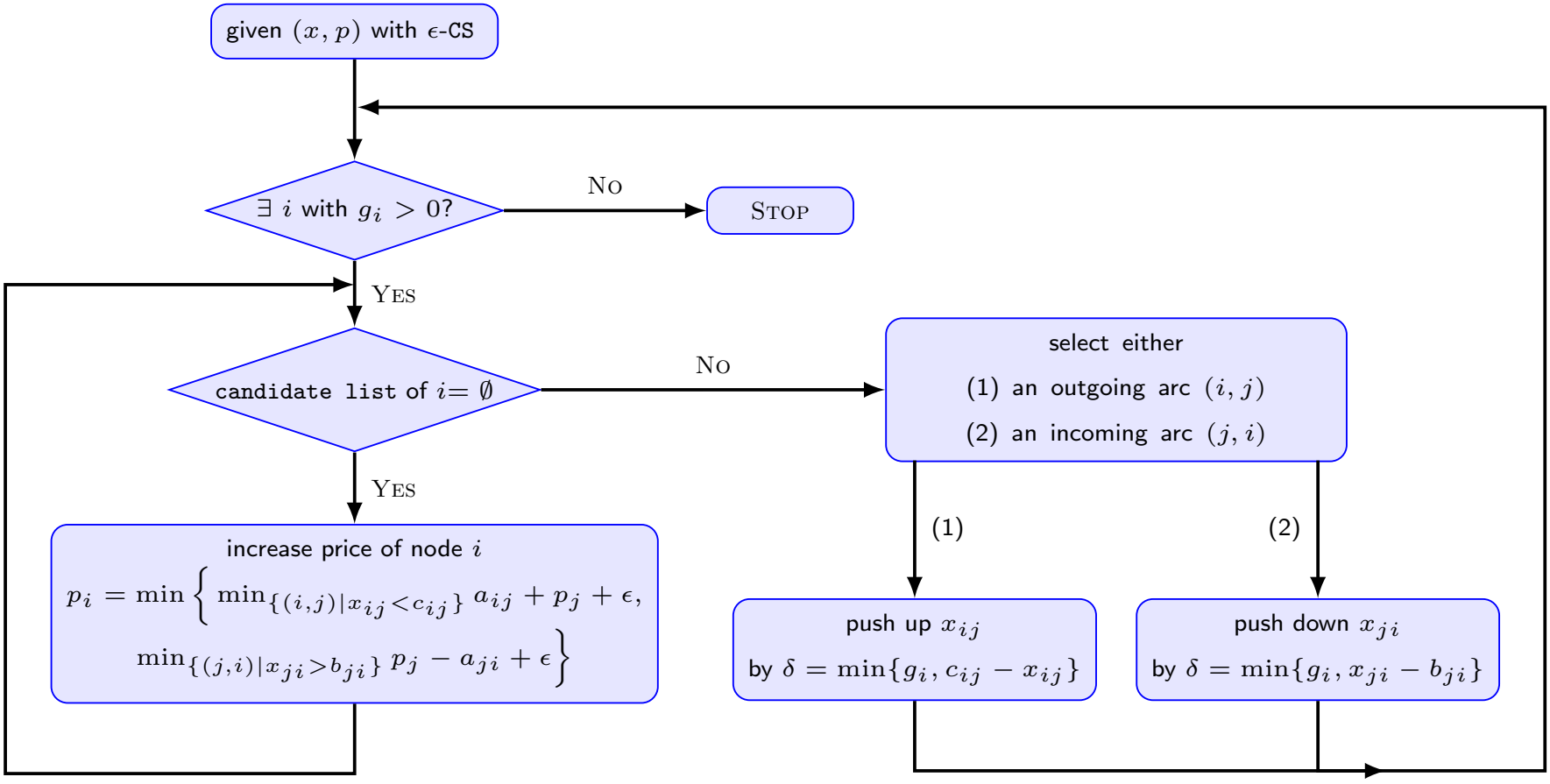
$$g_i = s_i - (x_{i2} - x_{1i})$$

push up x_{i2}, i.e., an ϵ^+-unblocked arc

push down x_{1i}, i.e., an ϵ^--unblocked arc

result = reduce g_i to zero as desired

Auction Algorithm, Min-Cost Flow



Auction Algorithm, Min-Cost Flow

- algorithm terminates
- solution upon termination satisfies $\epsilon-$CS
- suboptimality is bounded

Proposition

Assume that the minimum cost flow problem is feasible. Then the ϵ-relaxation method terminates with a pair (x, p) satisfying ϵ-CS. The flow vector x is feasible, and is optimal if $\epsilon < 1/N$.

Proof:

See pp 308-310 of the textbox.

Auction Algorithm, Min-Cost Flow

example: will be discussed in an exercise

Thank You!

Questions?

For more questions:

chatw@kth.se
themisc@kth.se
carlofi@kth.se
michael.rabbat@mcgill.ca