Lecture 8: Assignment Algorithms

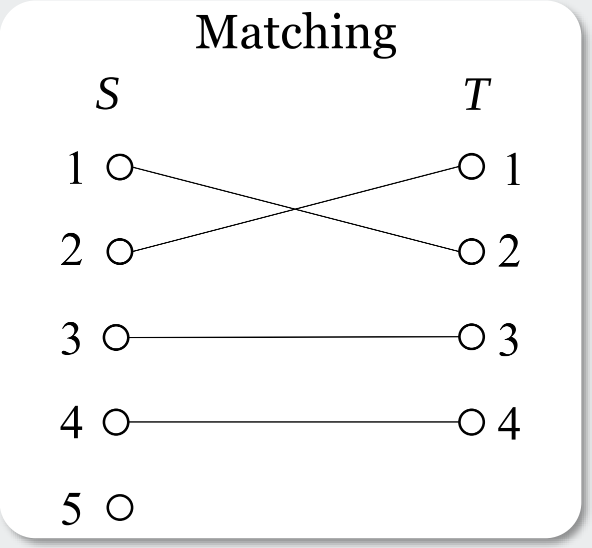
Prof. Krishna R. Pattipati
Dept. of Electrical and Computer Engineering
University of Connecticut
Contact: krishna@engr.uconn.edu; (860) 486-2890

Outline

- Examples of assignment problems
- Assignment Algorithms
  - Auction and variants
  - Hungarian Algorithm (also called Kuhn-Munkres Algorithm)
    - Easy to understand, but not for practical applications
  - Successive shortest path algorithm (Hungarian; Jonker, Volgenant and Castanon (JVC))
  - Signature … Not efficient computationally
- Special cases
- M-Best Assignment Algorithms
  - Murty (1968)
  - Stone & Cox (1995)
  - Popp, Pattipati & Bar-Shalom (1999)

Examples of assignment problems

- Assignment problem
  - Also known as weighted bipartite matching problem
- Bipartite graph
  - Has two sets of nodes $S, T \Rightarrow V = S \cup T$
  - And a set of edges E connecting them
- A *matching* on a bipartite graph $G = (S, T, E)$ is a subset of edges $X \in E \ni$ no two edges in X are incident to the same node



  - Nodes 1, 2, 3, 4 of S are matched or covered
  - Node 5 is uncovered or exposed

Matching problems

- Two types of matching problems: $m = |S|$; $n = |T|$
  - Cardinality matching problem
    - Find a matching with maximum number of arcs

$$\begin{aligned}
\max \ & \sum_{(i,j)\in E} x_{ij} \\
\text{s.t.} \ & \sum_i x_{ij} \le 1 \\
& \sum_j x_{ij} \le 1 \\
& x_{ij} \in \{0,1\}
\end{aligned}$$

  - Weighted matching problem or the assignment problem

$$\begin{aligned}
\max \ & \sum_{(i,j)\in E} w_{ij} x_{ij} && \text{or} \quad \min \sum_{(i,j)\in E} c_{ij} x_{ij},\ c_{ij} = -w_{ij} \\
\text{s.t.} \ & \sum_{(i,j)\in E} x_{ij} \le 1 \quad \forall i = 1,\ldots,n && (\text{or } =) \\
& \sum_{(i,j)\in E} x_{ij} \le 1 \quad \forall j = 1,\ldots,m && (\text{or } =) \\
& x_{ij} \in \{0,1\}
\end{aligned}$$

    - Problem can also be written as

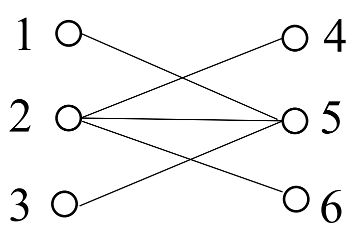
$\max_{\alpha} \sum_i w_{i\alpha_i}$ $\quad \alpha$ ~ permutation of columns of W

(= assignment of object α_i to person i)

Examples of assignment problems

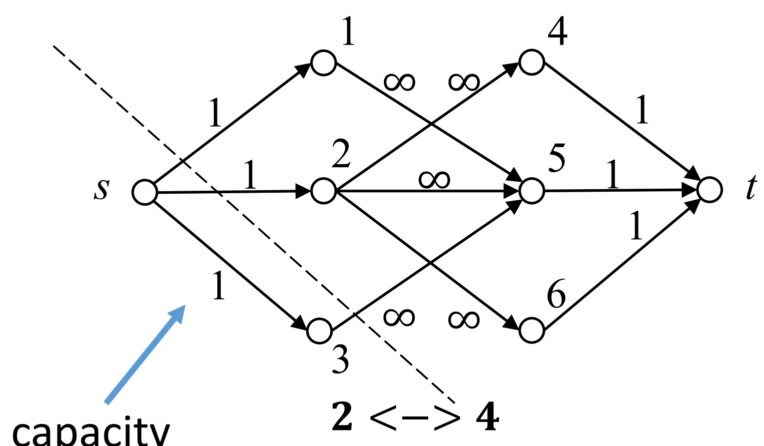
- Examples
  - Assigning people to machines
  - Correlating reports from two sensors
  - System of distinct representatives (cardinality matching problem)
  - Cardinality matching ~ maximum flow (~ ≡ analogous to)

cardinality matching



$m + n$ nodes

maximum flow



2 <−> 4

1 <−> 5

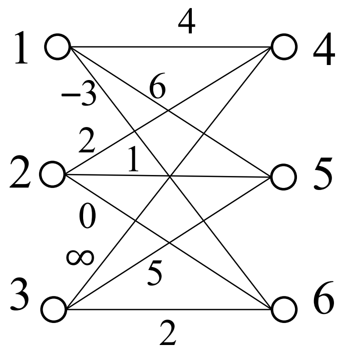
$m + n + 2$ nodes

(Lecture 9)

Examples of assignment problems

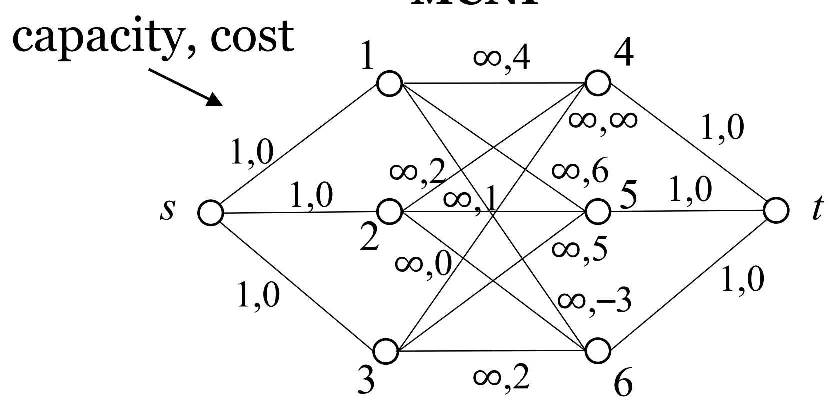
- $n \times n$ assignment or bipartite matching ~ minimum cost network flow problem … (Lecture 10)

bipartite matching



$2n$ nodes

MCNF



$2n + 2$ nodes

$$
\left[c_{ij}\right] = \begin{bmatrix} 4 & 6 & -3 \\ 2 & 1 & 0 \\ \infty & 5 & 2 \end{bmatrix}
$$

⇒ Can use RELAX to solve assignment problem (Lecture 10)

⇒ In this particular case, even ϵ – relax works as well as RELAX even on sequential computers (Lecture 10)

Optimality conditions

- Dual of the assignment problem
  - Assume equality constraints & $m = n$ w/o loss of generality

Primal

$$\max \sum_{(i,j)\in E} w_{ij} x_{ij}$$

$$\text{s.t.} \quad \sum_{(i,j)\in E} x_{ij} = 1 \quad \forall i = 1,\dots,n$$

$$\sum_{(i,j)\in E} x_{ij} = 1 \quad \forall j = 1,\dots,n$$

$$x_{ij} \in \{0,1\}, \forall (i,j) \in E$$

Dual

$$\min \left\{ \sum_{i=1}^{n} \pi_i + \sum_{j=1}^{n} p_j \right\}$$

$$\text{s.t.} \quad \pi_i + p_j \geq w_{ij}, \forall (i,j) \in E$$

CS conditions imply: $\left.\begin{array}{l} x_{ij} > 0 \Rightarrow \pi_i + p_j = w_{ij} \\ x_{ij} = 0 \Rightarrow \pi_i + p_j > w_{ij} \end{array}\right\}$ at optimum: $\pi_i = \max_{k \in \text{out}(i)} \{w_{ik} - p_k\} = w_{ij} - p_j$

- To provide physical interpretation, let i = person and j = object
  - p_j ~ price of object j = amount of money that a person is willing to pay when assigned to j
  - $(w_{ij} - p_j)$ ~ profit margin if person i is assigned to object j = benefit person i associates with being assigned to object j
  - So, if (i, j) is an optimal pair (i.e., part of an optimal solution), then the profit margin π_i^* is

$$\pi_i^* = w_{ij} - p_j^* = \max_{k \in \text{out}(i)} \left\{ w_{ik} - p_k^* \right\}$$

Optimality conditions

- Using this fact, we can simplify the dual problem considerably

$$\min q(\underline{\pi}, \underline{p}) = \min \left\{ \sum_{i=1}^{n} \pi_i + \sum_{j=1}^{n} p_j \right\}$$

$$\text{s.t.} \quad \pi_i + p_j \geq w_{ij}, \forall (i, j) \in E$$

$$\text{(or)} \quad \pi_i + p_j \geq w_{ij}, \forall j \in \text{out}(i); i = 1, 2, .., n$$

Note that every π_i can be *decreased* by a constant δ and every p_j *increased* by δ without affecting q.

  - Suppose that somebody gave us prices of objects $\{p_j\}$
  - Then, for a given set of $\{p_j\}$, the optimal

$$\pi_i = \max_{j \in \text{out}(i)} \{w_{ij} - p_j\}$$

  - Dual problem is equivalent to

$$\min q(\underline{p}) = \min_{\underline{p}} \left\{ \sum_{j=1}^{n} p_j + \sum_{i=1}^{n} \max_{j \in \text{out}(i)} \{w_{ij} - p_j\} \right\}$$

  - Note: no constraints on p_j

Auction algorithm for the assignment problem

- Start with an initial set of object prices, $\{p_j : 1 \le j \le n\}$ (e.g., $p_j = 0$ or $\max_i w_{ij}$)
- Initially, either no objects are assigned or else have ϵ-complementary slackness satisfied

$$\pi_i - \epsilon = \max_k \{w_{ik} - p_k\} - \epsilon \le w_{ij} - p_j, j = \arg\max_k \{w_{ik} - p_k\}$$

$$\pi_i - \epsilon \;\le\; w_{ij} - p_j = \pi_i, \forall\, i, j \in \text{out}(i) \ni j \text{ is assigned to } i$$

$$\Rightarrow \pi_i = w_{ij} - p_j \ge \max_k \{w_{ik} - p_k\} - \epsilon \Rightarrow \text{CS Conditions}$$

$\Rightarrow$ Partial ϵ-optimal assignment

- Auction algorithm consists of two phases
  - *Bidding phase:*
    - Each unassigned person i computes the "current value" of object j (i.e., potential profit if i is assigned to j) & bids for the object of his choice
  - *Assignment phase:*
    - Each object (temporarily) offers itself to the highest bidder
- Bidding phase
  - Each unassigned person i computes the value of object $j \in \text{out}(i)$

$$v_{ij} = w_{ij} - p_j; \quad j \in \text{out}(i)$$

Auction has Two phases: Bidding & Assignment

- Let
$$\pi_i = \text{max. value} = \max_{j \in \text{out}(i)} v_{ij} = v_{ij}^*$$
- Find the <u>next</u> best value (second best profit)
$$\phi_i = \max_{\substack{j \in \text{out}(i) \\ j \neq j^*}} v_{ij}$$
- Person i then bids for object j^* at a price of $b_{ij}^* = p_j^* + \pi_i - \phi_i + \epsilon = w_{ij}^* - \phi_i + \epsilon$
- Note: actually, can have $b_{ij}^* \in \{p_j^* + \epsilon,\ p_j^* + \pi_i - \phi_i + \epsilon\}$ ... but the above choice provides best performance (more later, from the dual cost structure)
- **Q**: what if j^* is the only object $\Rightarrow$ set $\phi_i = -\infty$ or a number $\ll \pi_i$
- This process is *highly parallelizable*

• Assignment phase
- For each object j, if $P(j)$ is the set of persons from whom j received a bid, then
$$p_j = \max_{i \in P(j)} b_{ij} = b_{i^* j}$$
- Announce the new price to all persons
- Assign person i^* to object $j \Rightarrow x_{i^* j} = 1$
- De-assign any previous assignment i' to $j \Rightarrow x_{i' j} = 0$
- If there was no bid, don't do anything
- This process is also *highly parallelizable*

Properties of Auction algorithm

- Properties
  - If p_j is price of object j before assignment & p'_j after assignment, we have $p'_j \geq p_j, \quad \forall j \Rightarrow p_j \uparrow$

$$\Rightarrow p'_{j^*} = p_{j^*} + \pi_i - \phi_i + \epsilon \Rightarrow \; p'_{j^*} \geq p_{j^*}$$

  - Maintains ϵ-complementary slackness for assigned objects
  - Suppose object j^* accepts bid of person i

$$\pi'_i = w_{ij^*} - p'_{j^*} = \phi_i - \epsilon = \max_{\substack{j \in \text{out}(i) \\ j \neq j^*}} \{w_{ij} - p_j\} - \epsilon \geq \max_{\substack{j \in \text{out}(i) \\ j \neq j^*}} \{w_{ij} - p'_j\} - \epsilon$$

$$w_{ij^*} - p'_{j^*} \geq \max_{j \in \text{out}(i)} \{w_{ij} - p'_j\} - \epsilon$$

$\Rightarrow$ ϵ-*CS* conditions continue to hold

  - Profit margin of i after assignment

$$\pi'_i = \max_{j \in \text{out}(i)} \{w_{ij} - p'_j\} = \max \left\{ w_{ij^*} - p'_{j^*}, \max_{\substack{j \in \text{out}(i) \\ j \neq j^*}} \{w_{ij} - p'_j\} \right\}$$

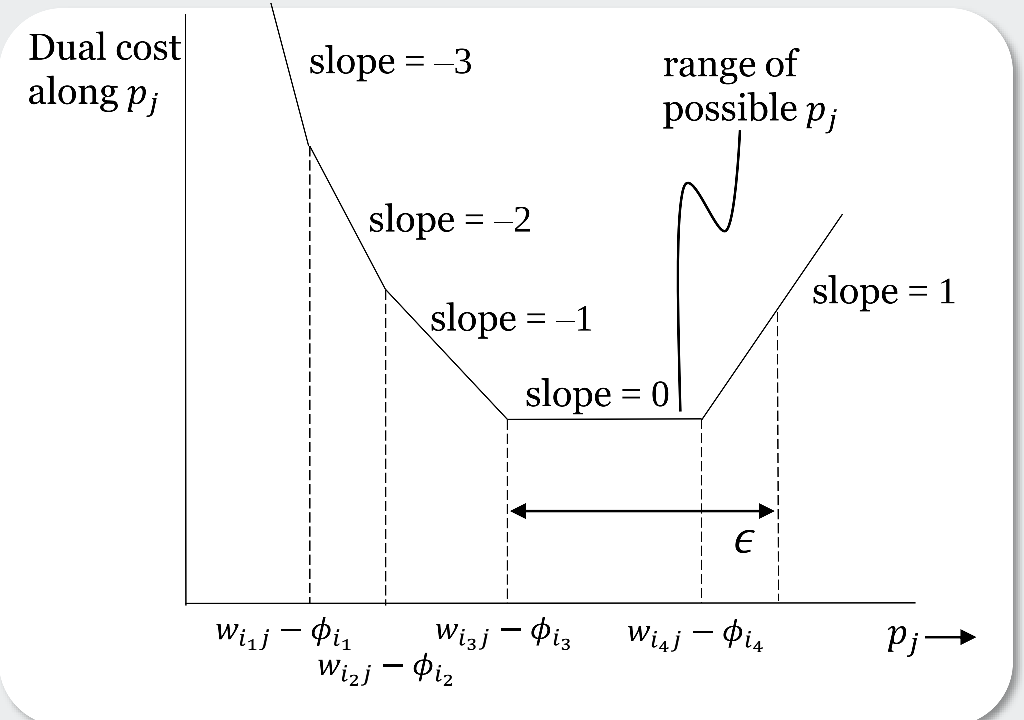
$$\leq \max \left\{ w_{ij^*} - p'{}_{j^*}, \max_{\substack{j \in \text{out}(i) \\ j \neq j^*}} \{w_{ij} - p_j\} \right\} = \max\{\phi_i - \epsilon, \phi_i\} = \phi_i$$

$p'_{j^*} = p_j + \pi_i - \phi_i + \epsilon \Rightarrow$ price increases by least ϵ

$\pi'_i = w_{ij^*} - p'_{j^*} = \phi_i - \epsilon \geq \pi'_i - \epsilon \Rightarrow$ profit goes down by at least ϵ

Coordinate descent interpretation

- Coordinate descent interpretation of bidding and assignment phases
  - Jacobi-like relaxation step for minimizing the dual function $q(\underline{p})$
    - In each bidding and assignment phase, the price p_j of each object j bidded upon is increased to either a value that minimizes $q(\underline{p})$ when all other prices are kept constant, or else exceeds the largest such value by no more than ϵ



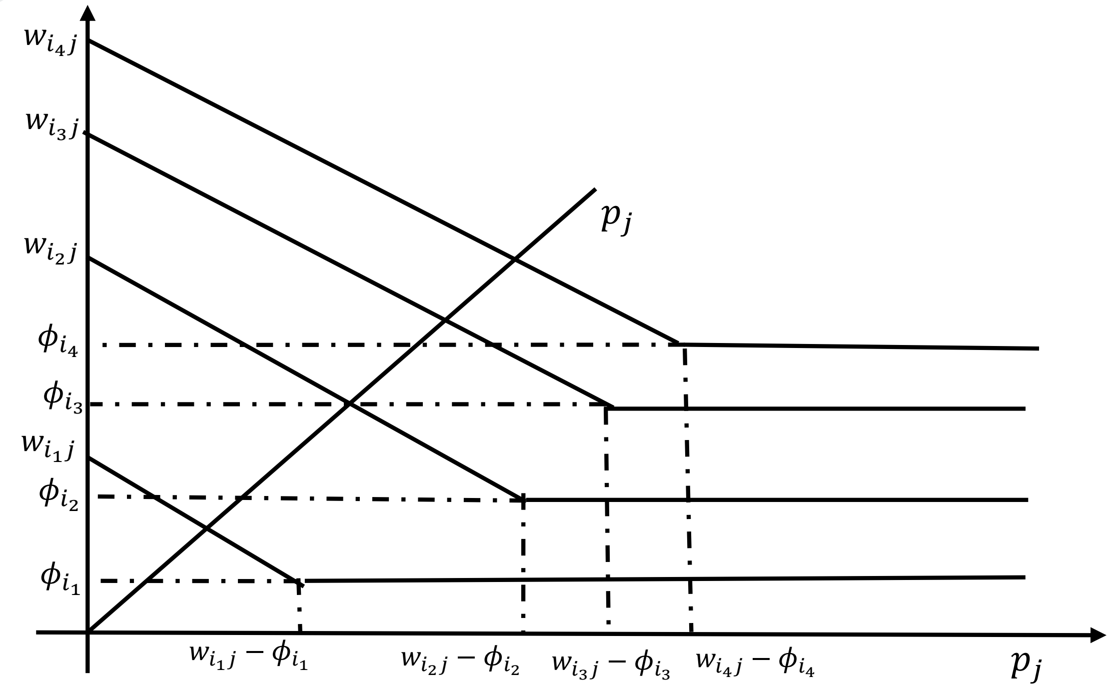
Recall: $q\left(\underline{p}\right) = \sum_{j=1}^{n} p_j + \sum_{i=1}^{n} \max_{j \in \text{out}(i)} \{w_{ij} - p_j\}$

$$\Rightarrow \max_{j \in \text{out}(i)} \{w_{ij} - p_j\} = \max\left\{(w_{ij} - p_j), \max_{j' \in \text{out}(i), j' \neq j} (w_{ij'} - p_{j'})\right\}$$

$$= \max\{(w_{ij} - p_j), \phi_i\}$$

Coordinate descent interpretation

- If $p_j < w_{ij} - \phi_i$, i bids for object j the amount $w_{ij} - \phi_i + \epsilon$
- Accepted bid by j: $\max_i \{w_{ij} - \phi_i\} + \epsilon$
- Also, note that the right directional derivative of $q(\underline{p})$ along p_j $(= \underline{e}_j)$ is
$$d_j^+ = 1 - (\#\text{ of persons } i \text{ with } j \in \text{out}(i) \ni p_j < w_{ij} - \phi_i)$$
- This is because we can increase p_j only if $\phi_i = w_{ij'} - p_{j'} \le w_{ij} - p_j$ where j' is second best
  - This interpretation leads to two other variations of auction
- Each assigned person i also bids for his own assigned object j, $b_{ij} = w_{ij} - \phi_i + \epsilon$



Gauss-Seidel method

- Gauss-Seidel
  - Only one person bids at a time
  - Price rise is taken into account before the next person bids
  - Problem: can't parallelize, but has much faster convergence on sequential implementations
  - $\exists$ *variations* in between Jacobi and Gauss-Seidel (e.g. bids by a subset of persons)… Research Problem
  - Q: Can we maintain optimality even in the presence of ϵ?
  - A: Yes!!
- Suppose i is assigned object $j_i, \ \forall i = 1,2,\ldots,n$
- Each step of the algorithm maintains

$$w_{ij_i} \geq \pi_i + p_{j_i} - \epsilon$$
$$\Rightarrow \sum_i w_{ij_i} \geq \sum_i \pi_i + \sum_i p_{j_i} - n\epsilon$$

- If f^* is the optimal primal value (note: maximization)

$$f^* \geq \sum_i w_{ij_i} \geq \sum_i \pi_i + \sum_i p_{j_i} - n\epsilon \geq f^* - n\epsilon$$

- If a_{ij} are integer & $\epsilon < \frac{1}{n} \Rightarrow n\epsilon < 1 \Rightarrow$ optimality
  - Does the algorithm terminate?
- Yes if $\exists$ at least one feasible solution

Illustration of Gauss-Seidel Auction algorithm

$$W = \begin{bmatrix} -14 & -5 & -8 & -7 \\ -2 & -12 & -6 & -5 \\ -7 & -8 & -3 & -9 \\ -2 & -4 & -6 & -10 \end{bmatrix}$$

- Initialize prices to zero. $\varepsilon = 0.2$

Iter	Prices	X	Bidder	Object	Bid
1	(0,0,0,0)	ϕ	1	2	2.2
2	(0,2.2,0,0)	(1,2)	2	1	3.2
3	(3.2,2.2,0,0)	(1,2),(2,1)	3	3	6.2
4	(3.2,2.2,6.2,0)	(1,2),(2,1), (3,3)	4	1	4.4
5	(4.4,2.2,6.2,0)	(1,2),(3,3), (4,1)	2	4	1.6
6	(4.4,2.2,6.2,1.6)	(1,2),(3,3), (4,1),(2,4)	ϕ	ϕ	ϕ

- Assignments: $x_{12} = 1; x_{33} = 1; x_{41} = 1; x_{24} = 1$

$$\underline{p}^T = \begin{bmatrix} 4.4 & 2.2 & 6.2 & 1.6 \end{bmatrix}$$

- $f^* = w_{12} + w_{33} + w_{41} + w_{24} = -15$

Auction with a different price initialization

$$W = \begin{bmatrix} -14 & -5 & -8 & -7 \\ -2 & -12 & -6 & -5 \\ -7 & -8 & -3 & -9 \\ -2 & -4 & -6 & -10 \end{bmatrix}$$

- Initialize prices. $p_j = \max_i(w_{ij})$ ε = 0.2

$$\underline{p}^T = \begin{bmatrix} -2 & -4 & -3 & -5 \end{bmatrix}$$

Iter	Prices	X	Bidder	Object	Bid
1	(-2,-4,-3,-5)	(2,1),(4,2),(3, 3)	1	2	-1.8
2	(-2,-1.8,-3, -5)	(2,1),(3,3),(1, 2)	4	1	0.2
3	(0.2,-1.8,-3, --5)	(3,3),(1,2),(4, 1)	2	4	-1.8
4	(0.2, -1.8, -3, -1.8)	(3,3),(1,2), (4,1),(2,4)	ϕ	ϕ	ϕ

- Assignments: $x_{12} = 1; x_{33} = 1; x_{41} = 1; x_{24} = 1$

$$\underline{p}^T = \begin{bmatrix} 0.2 & -1.8 & -3 & -1.8 \end{bmatrix}$$

- $f^* = w_{12} + w_{33} + w_{41} + w_{24} = -15$

Proof of convergence

- Proof relies on the following facts
  - If an object is assigned, it remains assigned throughout
  - Each time an object is bidded upon, its price increases by at least ϵ
  - If an object is bidded upon infinite number of times, its price $\rightarrow \infty$
  - Recall: unbounded dual $\rightarrow$ infeasible primal
- If a person bids for at most out(i) times, $\pi_i \downarrow$ by at least ϵ
  - Each time a person bids, $\pi_i \downarrow$ or π_i remains same
  - If exceed out(i) times, $\pi_i \downarrow$ at least ϵ
  - If a person bids infinite # of times, $\pi_i \downarrow -\infty$
- Can show (see Bertsekas's book) that, if the problem is feasible

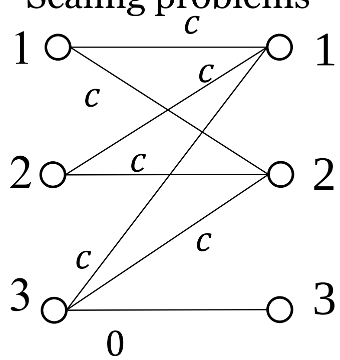
$$\pi_i \geq -(2n-1)C - (n-1)\epsilon - \max_j \left\{ p_j^0 \right\}, \text{ where } C = \max_{i,j} w_{ij}$$

If $\pi_i <$ this bound during auction, declare primal as infeasible. Unfortunately, it may take many iterations before the bound is crossed. Alternately, add artificial link (i, j)

$$w_{ij} < -(2n-1)C$$

Algorithm complexity

- Scaling problems



p_j

0 ϵ 2ϵ

0 ϵ 2ϵ

# of bidding phases = c/ϵ

- For some examples, # of bidding phases proportional to $\frac{nC}{\epsilon}$ $\Rightarrow$ performance is very sensitive to C
- Scale all costs w_{ij} by $(n + 1)$ & apply the algorithm with progressively lower value of ϵ until $\epsilon \approx 1$ or $\epsilon < 1$
  - Use
    $$\epsilon(k) = \max\left\{1, \frac{\Delta}{\tau_k}\right\}, k = 0,1,2,\ldots$$
  - $\Delta = nC$ & $\tau = 2$ is suggested
  - Alternately, $\epsilon = \frac{nC}{2}$ initially & reduce $\epsilon \leftarrow \max\left\{\frac{\epsilon}{6}, 1\right\}$
  - With these values, the complexity of the algorithm is $O(n|E|\log(nC))$

Proof of convergence for asynchronous version is similar to shortest path algorithm

- Computational results
  - Comparable to any existing assignment algorithm… average complexity $O(n^{1.8})$
  - Parallelizable
  - Especially fast for sparse problems

Auction algorithm variants

- Recent variants:
  - Reverse auction ⇒ objects bid for persons
  - Forward/reverse auction ⇒ alternate cycles of person-object biddings
  - Look-back auction
  - Extensions to inequality constraints
  - Asymmetric assignment problem ⇒ $m \neq n$
  - Multi-assignment problem ⇒ a person may be assigned to more than one object
- Reverse auction
  - Objects compete for persons by offering discounts
  - Duals: $q(\pi) = \sum_{i=1}^{n} \pi_i + \sum_{i=1}^{n} \max_{i \in \mathrm{In}(j)} (w_{ij} - \pi_i)$
  - Reverse auction iteration
    - Find the "best" person $i^* = \arg\max_{i \in \mathrm{In}(j)} (w_{ij} - \pi_i)$

      Let $\beta_j = w_{i^*j} - \pi_{i^*}$

      $$\delta_j = \max_{\substack{i \in \mathrm{In}(j) \\ i \neq i^*}} \{w_{ij} - \pi_i\}$$

      If i^* is the only person, set $\delta_j = -\infty$ or $\delta_j \ll \beta_j$
    - Bid for person i^*

      $$b_{i^*j} = w_{i^*j} - \delta_j + \epsilon$$
    - For each person i receiving at least one bid,

      $$\pi_i = \max_{j \in B(i)} b_{ij} = b_{ij^*}$$

$B(i)$ = set of objects from which i received a bid. De-assign any previous assignment for i, set $x_{ij^*} = 1$

Auction algorithm variants

- Combined forward and reverse auction
  - Maintain $\epsilon - CS$ conditions on both $(\underline{\pi}, \underline{p})$

$$\Rightarrow \quad \pi_i + p_j \geq w_{ij} - \epsilon \qquad \forall (i,j)$$

$$\pi_i + p_j = w_{ij} \qquad \forall (i,j) \in X \quad = \text{Solution}$$

  - **Execute alternately the forward auction a few iterations and reverse auction a few iterations**
    - **Run forward auction**: persons bid on objects. At the end of each iteration, set

$$\pi_i = w_{ij^*} - p_{j^*} \quad \text{if} \quad x_{ij^*} = 1$$

      stop if all are assigned, else go to Reverse Auction

      (do until number of assignments increases by at least one.)
    - **Run reverse auction**: objects bid on persons. At the end of each iteration, set

$$p_j = w_{i^*j} - \pi_{i^*} \quad \text{if} \quad x_{i^*j} = 1$$

      stop if all are assigned, else go to Forward Auction.

      (do until number of assignments increases by at least one.)
  - No need to do ϵ –scaling
- Look-back auction
  - Number of objects a person bids to is typically small (≈3)
  - Keep track of biddings of each person i in a small list
- Asymmetric assignment
  - Number of objects $n >$ number of persons, m
  - All persons should be assigned, but allow objects to remain unassigned

Asymmetric Auction algorithm

Primal

$$\max \sum_{(i,j)\in E} w_{ij} x_{ij}$$

$$\text{s.t.} \quad \sum_{(i,j)\in E} x_{ij} = 1 \quad \forall i = 1,2,\dots,m$$

$$\sum_{(i,j)\in E} x_{ij} \le 1 \quad \forall j = 1,2,\dots,n$$

$$x_{ij} \in \{0,1\}, \forall (i,j) \in E$$

Equivalent primal

$$\max \sum_{(i,j)\in E} w_{ij} x_{ij}$$

$$\text{s.t.} \quad \sum_{(i,j)\in E} x_{ij} = 1 \quad \forall i = 1,2,\dots,m \Rightarrow w_{sj} = 0$$

$$\sum_{(i,j)\in E} x_{ij} + x_{sj} = 1 \quad \forall j = 1,2,\dots,n$$

$$\sum_{j=1}^{n} x_{sj} = n - m$$

$$0 \le x_{ij} \ \forall (i,j) \in E$$

$$0 \le x_{sj} \ \forall j = 1,2,\dots,n$$

Dual

$$\min \sum_{i=1}^{m} \pi_i + \sum_{j=1}^{n} p_j - (n-m)\lambda$$

$$\text{s.t.} \quad \pi_i + p_j \ge w_{ij} \quad \forall (i,j) \in E$$

$$\lambda \le p_j \quad j = 1,2,\dots,n$$

$$\Rightarrow \min \sum_{i=1}^{m} \pi_i + \sum_{j=1}^{n} \max(w_{ij} - \pi_i) - (n-m) \min_j \max_{i \in \mathrm{In}(j)} (w_{ij} - \pi_i)$$

Reverse Auction algorithm

- Use a modified reverse auction
- Select an object j which is unassigned and satisfies $p_j > \lambda$. If no such object is found, terminate the algorithm.
- Find best person i^*

$$i^* = \arg\max_{i \in \mathrm{In}(j)} \left\{ w_{ij} - \pi_i \right\}$$

$$\beta_j = w_{i^* j} - \pi_{i^*}$$

$$\delta_j = \max_{i \in \mathrm{In}(j), i \neq i^*} \left\{ w_{ij} - \pi_i \right\}$$

If $\lambda \geq \beta_j - \epsilon$, set $p_j = \lambda$ and go to next iteration. Otherwise, let

$$\alpha = \min\left\{ \beta_j - \lambda, \beta_j - \delta_j + \epsilon \right\}$$

$$p_j = \beta_j - \alpha = \max\left\{ \lambda, \delta_j - \epsilon \right\} \Rightarrow p_j$$

$$\pi_{i^*} = \pi_{i^*} + \alpha = \pi_{i^*} \uparrow$$

Remove any assignment of i^*

Set $x_{i^* j} = 1$

- The algorithm terminates with an assignment that is within $m\epsilon$ of being optimal ($\epsilon \leq 1/m$). See Bertsekas's book.
- Can use forward-reverse auction. Initialize forward auction with object prices equal to zero.
- Multi-assignment $\Rightarrow$ it is possible to assign more than one object to a single person (e.g., tracking clusters of objects)… See Exercise 7.10 of Bertsekas's book.
- Asymmetric Assignment where there is no need for every person, as well as for every object, to be assigned. See Exercise 7.11 of Bertsekas's book.

Hungarian algorithm for the assignment problem

- Typically done as minimization. Fact: Adding a constant to a row or a column does not change solution

$$\min \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij}; c_{ij} = -w_{ij}$$
$$\text{s.t.} \sum_{j=1}^{n} x_{ij} = 1 \quad \forall i = 1,\ldots,n$$
$$\sum_{i=1}^{n} x_{ij} = 1 \quad \forall j = 1,\ldots,n$$
$$x_{ij} \in \{0,1\}, \forall (i,j)$$

$$\min \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} - \delta \sum_{j=1}^{n} x_{ij} = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} - \delta$$
$$\text{s.t.} \sum_{j=1}^{n} x_{ij} = 1 \quad \forall i = 1,\ldots,n$$
$$\sum_{i=1}^{n} x_{ij} = 1 \quad \forall j = 1,\ldots,n$$
$$x_{ij} \in \{0,1\}, \forall (i,j)$$

- Konig-Egervary Theorem: If have *n* zeros in different rows and columns of matrix *C*, then we can construct a "perfect" matching. This is called "ideal" ("perfect") cost matrix.
- Kuhn-Munkres algorithm systematically converts any *C* matrix into an "ideal" cost matrix
- Algorithm is called Hungarian in view of Konig-Egervary Theorem

Hungarian algorithm steps for minimization problem

- **Step 1:** For each row, subtract the minimum number in that row from all numbers in that row.
- **Step 2:** For each column, subtract the minimum number in that column from all numbers in that column.
- **Step 3:** Draw the minimum number of lines to cover all zeroes. If this number = n, Done — an assignment can be made.
- **Step 4:** Determine the minimum uncovered number (call it δ).
  - Subtract δ from uncovered numbers.
  - Add δ to numbers covered by two lines.
  - Numbers covered by one line remain the same.
  - Then, Go to **Step 3**.

	J	$\bar{J}$
I	↑	x
$\bar{I}$	x	↓

- Note: Essentially, we came to step 4 because we only partially matched *a subset of rows I and the corresponding subset of columns, J*. What if we subtract $\delta > 0$ from every row that is **not** in I , and we add δ to every column in J? Then, the only entries that get decreased are the entries that are **not** covered. The total decrease = $(n\text{-}|I|\text{-}|J|)\ \delta{>}0$.

Illustration of Hungarian algorithm

$$\begin{bmatrix} 14 & 5 & 8 & 7 \\ 2 & 12 & 6 & 5 \\ 7 & 8 & 3 & 9 \\ 2 & 4 & 6 & 10 \end{bmatrix} \xrightarrow{\text{Step 1}} \begin{bmatrix} 9 & 0 & 3 & 2 \\ 0 & 10 & 4 & 3 \\ 4 & 5 & 0 & 6 \\ 0 & 2 & 4 & 8 \end{bmatrix} \xrightarrow{\text{Step 2}} \begin{bmatrix} 9 & 0 & 3 & 0 \\ 0 & 10 & 4 & 1 \\ 4 & 5 & 0 & 4 \\ 0 & 2 & 4 & 6 \end{bmatrix}$$

$$\xrightarrow{\text{Step 4}} \begin{bmatrix} 10 & 0 & 4 & 0 \\ 0 & 9 & 4 & 0 \\ 4 & 4 & 0 & 3 \\ 0 & 1 & 4 & 5 \end{bmatrix} \xrightarrow{\text{Step 3}}$$

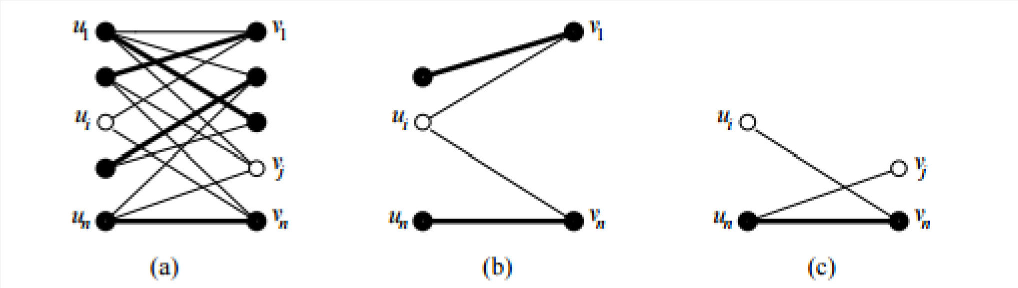
- Assignments
  - $x_{33} = 1$
  - $x_{41} = 1$
  - $x_{24} = 1$
  - $x_{12} = 1$

- Cost = $c_{12} + c_{24} + c_{33} + c_{41} = 15$

- **Row-column heuristic**: at each step, pick min element, assign, and remove corresponding row and column.
  - x_{21}=1; x_{33}=1; x_{42}=1; x_{14}=1; cost= 2+3+4+7=16

Graph theoretic ideas behind Hungarian algorithm

- An **alternating path** in a bipartite graph $G= (S, T, E)$ with respect to a matching M is a path with its edges alternately in M and not in M. See (a).
- **Alternating tree** Q (forest, F if the graph is disconnected) with respect to a matching M has a root node r in S and all other nodes except r are matched. See (b).
- An **augmenting path** P is a simple alternating path joining two free vertices $u \in S$ and $v \in T$. See (c).
- A Matching M is **perfect** (**maximal**) if there is *no* augmenting path with respect to M. see (a). $\{(u_1, v_3), (u_2, v_j), (u_i, v_1), (u_n, v_n)\}$



(a) (b) (c)

$u_i \to v_1 \to u_2 \to v_j$ (or) $u_i \to v_n \to u_n \to v_j$

multipliers

- The Hungarian algorithm successively builds maximum matchings on a equality graph G' that satisfies CS conditions, i.e., $r_{ij} = u_i + v_j - c_{ij} = 0$

Connection to Duality

- Recall dual:

$$\max q(\underline{u}, v) = \max \left\{ \sum_{i=1}^{n} u_i + \sum_{j=1}^{n} v_j \right\}$$
$$\text{s.t.} \quad u_i + v_j \le c_{ij}, \forall (i, j) \in E$$

- Hungarian algorithm is a primal-dual algorithm that starts with

$$u_i = \min_{j \in out(i)} c_{ij}; i = 1, 2, .., n$$
$$v_j = \min_{i \in In(j)} (c_{ij} - u_i); j = 1, 2, .., n$$

- Maintains dual feasibility $r_{ij} = c_{ij} - u_i - v_j \ge 0$.
- Works with equality graph, *G'* for which $r_{ij} = c_{ij} - u_i - v_j = 0$, i.e., preserves CS conditions. Solves the maximum cardinality bipartite matching problem on the graph *G'* that either finds a perfect matching, or we get a vertex cover of size < *n*.
- If (I, J) is not perfect cover, it finds the direction of **dual increase**

$$\delta = \min_{i \notin I; j \notin J} r_{ij}$$
$$\pi_i = \pi_i - \delta; i \in I$$
$$p_j = p_j + \delta; j \notin J$$

Illustration of Hungarian algorithm

$$C = \begin{bmatrix} 14 & 5 & 8 & 7 \\ 2 & 12 & 6 & 5 \\ 7 & 8 & 3 & 9 \\ 2 & 4 & 6 & 10 \end{bmatrix} \Rightarrow \underline{u} = \begin{bmatrix} 5 \\ 2 \\ 3 \\ 2 \end{bmatrix}; \underline{v}^T = \begin{bmatrix} 0 & 0 & 0 & 2 \end{bmatrix}; R = \begin{bmatrix} 9 & 0 & 3 & 0 \\ 0 & 10 & 4 & 1 \\ 4 & 5 & 0 & 4 \\ 0 & 2 & 4 & 6 \end{bmatrix};$$

$$Dual\ cost = 14$$

$$G' = (\{1,2,3,4\}, \{1,2,3,4\}, \{(1,2),(1,4),(2,1),(3,3),(4,1)\})$$

$$Matching, M' = \{(1,2),(2,1),(3,3)\} \neq perfect.\ I = \{1\};\ J = \{1,3\}.\ Node\ cover = 3 < 4$$

$$\delta = \min_{i \notin I,\, j \notin J} c'_{ij} = 1 \Rightarrow \underline{u} = \begin{bmatrix} 4 \\ 2 \\ 3 \\ 2 \end{bmatrix}; \underline{v}^T = \begin{bmatrix} 0 & 1 & 0 & 3 \end{bmatrix}; R = \begin{bmatrix} 10 & 0 & 4 & 0 \\ 0 & 9 & 4 & 0 \\ 4 & 4 & 0 & 3 \\ 0 & 1 & 4 & 5 \end{bmatrix}$$

- Assignments: $x_{33} = 1$; $x_{41} = 1$; $x_{24} = 1$; $x_{12} = 1$
- Cost = $c_{12} + c_{24} + c_{33} + c_{41} = 15$

$$\text{Dual cost} = \sum_{i=1}^{n} u_i + \sum_{j=1}^{n} v_j = 15$$

Class of Shortest Augmenting Path Algorithms

- Fact: Assignment is a special case of transportation problem and the more general minimum cost network flow (MCNF) problem (Lecture 10)
- In assignment, each node in S transmits one unit and each unit in T must receive one unit $\Rightarrow$ multi-source, multi-sink problem
- Indeed, an optimal solution can be found by considering one source in S at a time and finding the shortest path emanating from it to an unassigned node in T.
- While Hungarian algorithm finds any **feasible** augmenting path, Jonker, Volgenant and Castanon (JVC) and a number of other algorithms find the **shortest** augmenting paths
  - JVC $\equiv$ a clever shortest augmenting path implementation of Hungarian and a number of pre-processing techniques, including column reduction, reduction transfer, reduction of unassigned rows, which is basically auction
- Basic idea
  - Select an unassigned node in S
  - Construct the residual (auxiliary, incremental) graph, G' with costs $\{r_{ij}\}$
  - Find the shortest augmenting path via Dijkstra (recall $r_{ij} \geq 0$)
  - Augment the solution $\Rightarrow$ improve the match
  - Update the dual variables so that CS conditions hold

Jonker, Volgenant and Castanon (JVC) algorithm

- JVC algorithm for primal minimization ($c_{ij} = -w_{ij}$, Code is available on the net)

Primal

$$\min \sum_i \sum_j x_{ij} c_{ij}$$

$$\text{s.t.} \quad \sum_i x_{ij} = 1$$

$$\sum_j x_{ij} = 1$$

$$x_{ij} \geq 0, \forall (i, j,) \in E$$

Dual

$$\max \left\{ \sum_i u_i + \sum_j v_j \right\}$$

$$\text{s.t.} \quad c_{ij} - u_i - v_j \geq 0$$

  - Indices i and j refer to rows and columns, respectively
  - $x_i(y_j)$ is the column (row) index assigned to row i (column j), with $x_i = 0$ ($y_j = 0$) for an unassigned row i (column j)
  - The dual variables (prices) u_i and v_j correspond to row i and column j, respectively, with $r_{ij} = c_{ij} - u_i - v_j$ denoting the reduced costs

- Basic outline of JVC algorithm

  Step 1: *initialization* … column reduction

  Step 2: *termination*, if all rows are assigned

  Step 3: *augmentation* … construct auxiliary network and determine from unassigned row i to unassigned column j an alternating path of minimal total reduced cost … use to augment the solution

  Step 4: *update dual solution* … restore complementary slackness and go to step 2

JVC algorithm procedure – initialization

- Initialization
  - Column reduction

    for $j = n \dots 1$ do

    $c_j = j;\ h = c_{1j};\ i_1 = 1$

    for $i = 2 \dots n$ do

    if $c_{ij} < h$ then $h = c_{ij};\ i_1 = i$

    $v_j = h$

    if $x_{i_1} = 0$ then $x_{i_1} = j; y_j = i_1$

    else $x_i = -x_i; y_j = 0$

    end do

    end do

    $$\Rightarrow v_j = \min_i (c_{ij})$$

    .. each column is assigned to minimal row element

    .. some rows may not be assigned
  - Reduction transfer from unassigned to assigned rows

    for each assigned row i do

    $j_1 = x_i;\ \mu = \min\{c_{ij} - v_j\colon j = 1, \dots, n;\ \ j \neq j_1\}$

    $v_{j1} = v_{j1} - (\mu - u_i);\ \ u_i = \mu$

    end do

  ... similar to 2nd best profit in auction

Column Reduction:

$$C = \begin{bmatrix} 14 & 5 & 8 & 7 \\ 2 & 12 & 6 & 5 \\ 7 & 8 & 3 & 9 \\ 2 & 4 & 6 & 10 \end{bmatrix}$$

$$\Rightarrow \underline{v}^T = \begin{bmatrix} 2 & 4 & 3 & 5 \end{bmatrix}; \underline{u} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}; x_{21} = x_{42} = x_{33} = 1$$

$Dual\ \cos t = 14$

$Row 1\ and\ column\ 4\ unassigned.$

Reduction Transfer:

Row 2: μ=min(8,3,0), $v_1 = 2; u_1 = 0$

Row 3: μ=min(5,4,4),$v_3 = -1; u_3 = 4$

Row 4: μ=min(0,3,5),$v_2 = 4; u_4 = 0$

$$\Rightarrow \underline{v}^T = \begin{bmatrix} 2 & 4 & -1 & 5 \end{bmatrix}; \underline{u} = \begin{bmatrix} 0 \\ 0 \\ 4 \\ 0 \end{bmatrix}$$

$Dual\ \cos t = 14$

JVC algorithm procedure – pre-processing

- Augmenting reduction of unassigned rows (…auction)

  $LIST = \{\text{all unassigned rows}\}$

  $\forall i \in LIST$ do

  repeat

  $k_1 = \min\{c_{ij} - v_j:\ \ j = 1, \ldots, n\}$

  select j_1 with $c_{ij_1} - v_{j_1} = k_1$

  $k_2 = \min\{c_{ij} - v_j:\ \ j = 1, \ldots, n; j \neq j_1\}$

  select j_2 with $w = c_{ij_2} - v_{j_2} = k_2;\ j_2 \neq j_1$

  $u_i = k_2$

  if $k_1 < k_2$ then

  $v_{j_1} = v_{j_1} - (k_2 - k_1)$

  else if j_1 is assigned, then

  $j_1 = j_2$

  $r = y_{j_1};$

  if $r > 0$ then

  $x_r = 0;\ x_i = j_1;\ y_{j_1} = i; i = r$

  until $k_1 = k_2$ or $r = 0$

  end do

Augmentation Reduction:

Row 1: k_1=min(12,1,9,2)=1;$k_2 = 2$

$v_2 = 3; u_1 = 2$

$x_{12} = 1$

$x_{42} = 0$

$$\Rightarrow \underline{v}^T = \begin{bmatrix} 2 & 3 & -1 & 5 \end{bmatrix}; \underline{u} = \begin{bmatrix} 2 \\ 0 \\ 4 \\ 0 \end{bmatrix}$$

$Dual\ \cos t = 15$

Row 4: k_1=min(0,1,7,5)=0;$k_2 = 1$

$v_1 = 1; u_4 = 1$

$x_{41} = 1$

$x_{21} = 0$

$$\Rightarrow \underline{v}^T = \begin{bmatrix} 1 & 3 & -1 & 5 \end{bmatrix}; \underline{u} = \begin{bmatrix} 2 \\ 0 \\ 4 \\ 1 \end{bmatrix}$$

Row 2: k_1=min(1,9,7,0)=0;$k_2 = 1$

$v_4 = 4; u_2 = 1$

$x_{24} = 1$

$r = 0$

$$\Rightarrow \underline{v}^T = \begin{bmatrix} 0 & 3 & -1 & 4 \end{bmatrix}; \underline{u} = \begin{bmatrix} 2 \\ 1 \\ 4 \\ 2 \end{bmatrix}$$

Primal and dual feasible. Done!

- Complexity of first two initialization procedures is $O(n^2)$, and it can be shown that the augmenting reduction procedure has complexity $O(Rn^2)$, with R the range of cost coefficients

JVC algorithm procedure – augmentation + update

- Augmentation
  - Modified version of Dijkstra's shortest augmenting path method

$\forall i^*$ unassigned do
    $TOSCAN = \{1 \dots n\}$
    for $j = 1 \dots n$ do
        $d_j = \infty$
    end do
    $i = i^*;\ d_{i^*} = 0;\ \mu = 0$
    repeat
        $\forall j \in (OUT(i) \cap TOSCAN)$ do
            if $\mu + c_{ij} - u_i - v_j < d_j$ then
                $d_j = \mu + c_{ij} - u_i - v_j;\quad \text{pred}[j] = i$
        end do
        $\mu = \infty$
        $\forall j \in TOSCAN$ do
            if $d_j < \mu$ then
                $\mu = d_j;\ \mu_j = j$
        end do
        $i = y_{\mu_j}; TOSCAN = TOSCAN - \{\mu_j\}$
    until $y_{\mu_j} = 0$
end do

Don't need to execute this too many times

  - Complexity for augmentation phase is $O(n^3)$, so this holds for the entire JVC algorithm
- Update of the dual solution
  - After augmentation of the partial assignment, the values of dual variables must be updated to restore complementary slackness, that is,
    - $c_{ik} - u_i - v_k = 0$, if $x_i = k$ for assigned column $k, i = 1, \dots, n$ and
    - $c_{ik} - u_i - v_k \geq 0$

Murty's Clever Partitioning Idea to get *m*-best solutions

- Suppose you have three binary variables, x_1 , x_2 , x_3

x_1	x_2	x_3
1	1	1
1	1	0
1	0	1
1	0	0
0	1	1
0	1	0
0	0	1
0	0	0

Best solution: (1,1,0)

2nd Best solution: (0,1,1)

- To get the 3rd best, do the partitioning on 2nd best solution
  - Fix x_1= 0 and x_2=0 and search over(0,0,1),(0,0,0)
  - Fix x_1= 0, x_2=1 and x_3=0
  - Pick the 3rd best from these and others in the list.

- To get 2nd best solution
  - Fix x_1= 0
    - Now, you are searching over (0,1,1), (0,1,0),(0,0,1),(0,0,0)
  - Fix x_1= 1 and x_2=0
    - Now, you are searching over (1,0,1), (1,0,0)
  - Fix x_1= 1, x_2=1 and x_3=1
    - Now, you evaluate (1,1,1)

  $\Rightarrow$ You searched over all (x_1 , x_2 , x_3), except the best (1,1,0)
  - Pick the 2nd best solution (cost and variables) from the solutions from these. Keep the other two in a list.
- To get the 3rd best, do the partitioning on 2nd best solution

Murty's Partitioning for Assignment

- Suppose we express the assignment problem P of size n as a bipartite graph, represented as a list of triples $\langle y, z, l\rangle$, with an assignment A^k from the (feasible) solution space A denoted as a set of triples in which each y and z appear exactly once, that is,

$$A = \bigcup_{k=1}^{n!} A^k, \qquad \text{where } A^k = \left\{\left\langle y_{k_j}, z_{k_j}, l_{k_j}\right\rangle\right\}, \qquad j = 1, \dots, n$$

- The cost $C(A^k)$ of the assignment A^k is simply the sum of l's in the triples, that is,

$$C\left(A^k\right) = \sum_{j=1}^{n} l_{k_j}$$

- The single best assignment A^* to P can be found using any of the classical methods of solving the assignment problem (e.g., auction, JVC)
- Subsequent assignments to P are found by solving a succession of assignment problems, where P is *partitioned* into a series of sub-problems, $P_1, P_2, \dots, P_n$, having solution spaces $A_1, A_2, \dots, A_n \ni$

$$\bigcup_{i=1}^{n} A_i = A - A^*$$

$$A_i \cap A_j = \emptyset, \qquad \text{for } i, j = 1, \dots, n, \qquad i \neq j$$

- Subproblem P_1 is P less the 1st triple $\langle y_1, z_1, l_1\rangle$ in the best assignment A^* of P

$$\Rightarrow A_1 = \left\{A^i \in A \colon \left\langle y_{i_j}, z_{i_j}, l_{i_j}\right\rangle \neq \langle y_1, z_1, l_1\rangle, j = 1, \dots, n\right\}$$

- In subproblems $P_2, \dots, P_n$, we want to force $\langle y_1, z_1, l_1\rangle$ to be in all assignments of $A_2, \dots, A_n$

Murty's M-Best assignment algorithm

- Hence, P_2 is P_1, plus the triple $\langle y_1, z_1, l_1 \rangle$, less all triples $\langle y_1, z_j, l_j \rangle \& \langle y_j, z_1, l_j \rangle$, $j = 2, \dots, n$, and less the 2nd triple $\langle y_2, z_2, l_2 \rangle$ in the best assignment A^* of P

$$\Rightarrow A_2 = \left\{ A^i \in A : \left\langle y_{i_j}, z_{i_j}, l_{i_j} \right\rangle \neq \langle y_2, z_2, l_2 \rangle,\ j = 1, \dots, n\ \ \& \langle y_1, z_1, l_1 \rangle \in A^i \right\}$$

- In the construction of subproblems $P_k, \dots, P_n$, we force $\langle y_l, z_l, l_l \rangle, l = 1, \dots, k-1$ to be in all assignments of $A_k, \dots, A_n$
- In general, subproblem P_k, $1 < k \le n$, is P_{k-1} plus the triple $\langle y_{k-1}, z_{k-1}, l_{k-1} \rangle$, less all triples $\langle y_{k-1}, z_j, l_j \rangle \& \langle y_j, z_{k-1}, l_j \rangle, j \neq k-1$, and less the kth triple $\langle y_k, z_k, l_k \rangle$ in the best assignment A^* of P

$$\Rightarrow A_k = \left\{ A^i \in A : \left\langle y_{i_j}, z_{i_j}, l_{i_j} \right\rangle \neq \langle y_k, z_k, l_k \rangle,\ j = 1, \dots, n\ \ \& \langle y_l, z_l, l_l \rangle \in A^i, l = 1, \dots, k-1 \right\}$$

- Note that solution spaces A_i for subproblems P_i, $i = 1, \dots, n$ are disjoint and their union will be exactly the solution space to P less its optimal assignment (i.e., $A - A^*$)
- Once we partition P according to its optimal assignment A^*, we place the resulting subproblems $P_1, \dots, P_n$ together with their optimal assignments $A_1^*, \dots, A_n^*$ on a priority queue of (problem, assignment) pairs (e.g., $Queue \leftarrow (P_i, A_i^*), i = 1, \dots, n$)
- We then find the problem P' in the queue having the best assignment … the best assignment A'^* to this problem is the 2nd best assignment to P
- We then remove P' from the queue and replace it by its partitioning (according to optimal assignment A'^*) $\Rightarrow$ the best assignment found in the queue will be the 3rd best assignment to P
- Complexity: since we perform one partitioning for each of the M-Best assignments, each partitioning (worst case) creating $O(n)$ new problems, $\Rightarrow O(Mn)$ assignment problems and queue insertions . . . each assignment takes $O(n^3)$, each insertion takes at most $O(Mn)$, hence, Murty's algorithm takes $O(Mn^4)$

Optimization of Murty's algorithm

- (Stone & Cox)
  - Recall Murty's algorithm is independent of the assignment algorithm chosen for solving the assignment problem
  - Suppose we use the JVC algorithm to find the optimal assignment
    - 2 important properties of the JVC algorithm
      - ❖ The optimal assignment will be found as long as the partial binary variable assignment, X, and dual variables u and v satisfy the following two criteria:

$$\forall i, j \ni x_{ij} = 1, c_{ij} = u_i + v_j$$

$$\forall i, j \ni x_{ij} = 0, c_{ij} \geq u_i + v_j$$

      - ❖ The slack or reduced cost (i.e., $c_{ij} - u_i - v_j$) between any z and y is useful in estimating the cost of the best assignment in which they are assigned to each other
        - ➢ If the current lower bound on the cost of the partial assignment, with either z_i or y_j unassigned, is C, then the cost of the optimal assignment that assigns z_i to y_j will be at least $C + c_{ij} - u_i - v_j$
        - ➢ A new lower bound can be computed by finding the minimum slack of a partial assignment

    When used in Murty's algorithm, Stone & Cox noted that the JVC assignment algorithm allows for a # of optimizations that dramatically improve both average-case and worst-case complexity
  - 3 optimizations proposed
    - Dual variables & partial solution inheritance during partitioning
    - Sorting subproblems by lower cost bounds before solving
    - Partitioning in an optimized order
  - Only inheritance reduces worst-case complexity, so we'll discuss it

Dual variables

- Dual variables & partial solution inheritance during partitioning
  - When a problem is partitioned, considerable work has been expended in finding its best assignment . . . we exploit this computation when finding best assignments in subproblems resulting from partitioning
  - Consider a problem P being partitioned, with dual variables $u^{(P)}$ and $v^{(P)}$, binary variable assignment $x^{(P)}$, and cost matrix $c^{(P)}$
    - Subproblem P_1 can inherit part of P's computation by assigning $c^{(P_1)} = c^{(P)}$, setting $c_{i_1 j_1}^{(P_1)} = \infty$ and setting $x_{i_1 j_1}^{(P_1)} = 0$ corresponding to arc $\langle y_{i_1}, z_{i_1}, l \rangle \in A^*$, and initializing $x^{(P_1)} = x^{(P)}$, $u^{(P_1)} = u^{(P)}$, and $v^{(P_1)} = v^{(P)}$
    - In general, subproblem P_k can inherit part of the computation at P_{k-1} by assigning $c^{(P_k)} = c^{(P_{k-1})}$, setting $c_{i_k j_k}^{(P_k)} = \infty$ and setting $x_{i_k j_k}^{(P_k)} = 0$ corresponding to $\langle y_{i_k}, z_{i_k}, l \rangle \in A^*$, and initializing $x^{(P_k)} = x^{(P_{k-1})}$, $u^{(P_k)} = u^{(P_{k-1})}$, and $v^{(P_k)} = v^{(P_{k-1})}$

    The value of this optimization is that, after the first assignment to the original assignment problem is found, no more than one augmentation needs to be performed for each new subproblem
  - Complexity: the augmentation step of JVC takes $O(n^2)$, so Murty's algorithm with this optimization is $O(Mn^3)$

Correlation problem

- A solvable case of assignment problem: correlation problem

$$w_{ij} = \int_{\alpha_i}^{\beta_j} f(y)\,dy, \text{ if } \beta_j \geq \alpha_i$$

$$w_{ij} = \int_{\beta_j}^{\alpha_i} g(y)\,dy, \text{ if } \beta_j < \alpha_i$$

  - $f(y) = g(y) = 1 \Rightarrow w_{ij} = |\alpha_i - \beta_j|$
  - Matching
$$\begin{array}{ccccc} \alpha_1 \geq & \alpha_2 & \dots \geq & \alpha_n \\ \updownarrow & \updownarrow & & \updownarrow \\ \beta_1 \geq & \beta_2 & \dots \geq & \beta_n \end{array}$$
  - This can be interpreted as the following assignment problem
    - Suppose have n in S and n in T such that
      - Attractiveness of $i \in S = \alpha_i$
      - Attractiveness of $j \in T = \beta_j$
      - Define $w_{ij} = |\alpha_i - \beta_j|$

$$\begin{aligned} &\min \textstyle\sum_i \sum_j x_{ij} w_{ij} \\ &\text{s.t. } \textstyle\sum_i x_{ij} = 1, \forall j \\ &\qquad \textstyle\sum_j x_{ij} = 1, \forall i \\ &\qquad x_{ij} \geq 0 \end{aligned} \quad \Rightarrow \quad \begin{aligned} &\text{sort } \alpha_1 \geq \alpha_2 \geq \ldots \geq \alpha_n \\ &\text{sort } \beta_1 \geq \beta_2 \geq \ldots \geq \beta_n \\ &\text{assign}(i, j) \end{aligned}$$

This is also min-max optimal matching … ∃ several other solvable cases…see Lawler

Summary

- Examples of assignment problems
- Auction algorithm
  - Coordinate dual descent
  - Scaling in critical to the success of the algorithm
- Hungarian method
  - $O(n^3)$ algorithm
  - Not as fast as auction or JVC in practice
- Successive shortest path method
  - JVC algorithm
  - Fastest algorithm in practice to date (?)
- M-Best assignment algorithms

Recent Book: R. Burkard, M. Dell'Amico and S. Martello, Assignment Problems, SIAM, 2009.