7.13 Assignment Problem

Assignment Problem

Assignment problem.

- Input: weighted, complete bipartite graph $G = (L \cup R, E)$ with $|L| = |R|$.
- Goal: find a perfect matching of min weight.

	1'	2'	3'	4'	5'
1	3	8	9	15	10
2	4	10	7	16	14
3	9	13	11	19	10
4	8	13	12	20	13
5	1	7	5	11	9

<u>Min cost perfect matching</u>

M = { 1-2', 2-3', 3-5', 4-1', 5-4' }

cost(M) = 8 + 7 + 10 + 8 + 11 = 44

Applications

Natural applications.

- Match jobs to machines.
- Match personnel to tasks.
- Match PU students to writing seminars.

Non-obvious applications.

- Vehicle routing.
- Signal processing.
- Virtual output queueing.
- Multiple object tracking.
- Approximate string matching.
- Enhance accuracy of solving linear systems of equations.

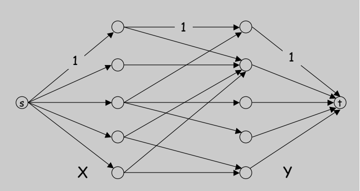
Bipartite Matching

Bipartite matching. Can solve via reduction to max flow.

Flow. During Ford-Fulkerson, all capacities and flows are 0/1. Flow corresponds to edges in a matching M.

Residual graph G_M simplifies to:

- If $(x, y) \notin M$, then (x, y) is in G_M.
- If $(x, y) \in M$, the (y, x) is in G_M.

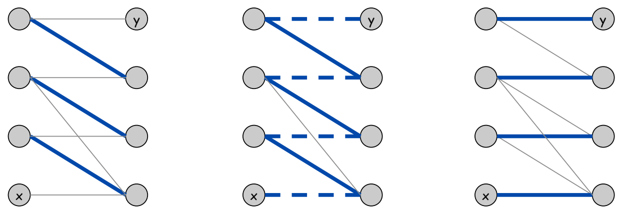


Augmenting path simplifies to:

- Edge from s to an unmatched node $x \in X$.
- Alternating sequence of unmatched and matched edges.
- Edge from unmatched node $y \in Y$ to t.

Alternating Path

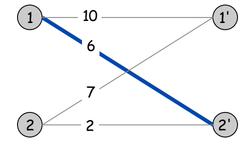
Alternating path. Alternating sequence of unmatched and matched edges, from unmatched node $x \in X$ to unmatched node $y \in Y$.



matching M   alternating path   matching M'

Assignment Problem: Successive Shortest Path Algorithm

Cost of an alternating path. Pay c(x, y) to match x-y; receive c(x, y) to unmatch x-y.



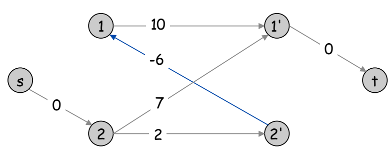
cost(2 - 1') = 7
cost(2 - 2'- 1 - 1') = 2 - 6 + 10 = 6

Shortest alternating path. Alternating path from any unmatched node $x \in X$ to any unmatched node $y \in Y$ with smallest cost.

Successive shortest path algorithm.
- Start with empty matching.
- Repeatedly augment along a shortest alternating path.

Finding The Shortest Alternating Path

Shortest alternating path. Corresponds to shortest s-t path in G_M.



Concern. Edge costs can be negative.

Fact. If always choose shortest alternating path, then G_M contains no negative cycles $\Rightarrow$ compute using Bellman-Ford.

Our plan. Use duality to avoid negative edge costs (and negative cost cycles) $\Rightarrow$ compute using Dijkstra.

Equivalent Assignment Problem

Duality intuition. Adding (or subtracting) a constant to every entry in row x or column y does not change the min cost perfect matching(s).

c(x, y)

3	8	9	15	10
4	10	7	16	14
9	13	11	19	10
8	13	12	20	13
1	7	5	11	9

11

subtract 11 from column 4 $\longrightarrow$

$c^p(x, y)$

3	8	9	4	10
4	10	7	2	14
9	13	11	8	10
8	13	12	9	13
1	7	5	0	9

Equivalent Assignment Problem

Duality intuition. Adding $p(x)$ to row x and subtracting $p(y)$ from row y does not change the min cost perfect matching(s).

$c(x, y)$

3	**8**	9	15	10	5
4	10	**7**	16	14	4
9	13	11	19	**10**	3
8	13	12	20	13	0
1	7	5	**11**	9	8
8	13	11	19	13	

$\longrightarrow$

$c^p(x, y)$

0	**0**	3	1	2
0	1	**0**	1	5
4	3	3	3	**0**
0	0	1	1	0
1	2	2	**0**	4

↑ 9 + 8 - 13

Reduced Costs

Reduced costs. For $x \in X$, $y \in Y$, define $c^p(x, y) = p(x) + c(x, y) - p(y)$.

Observation 1. Finding a min cost perfect matching with reduced costs is equivalent to finding a min cost perfect matching with original costs.

$c(x, y)$

3	**8**	9	15	10	5
4	10	**7**	16	14	4
9	13	11	19	**10**	3
8	13	12	20	13	0
1	7	5	**11**	9	8
8	13	11	19	13	

$\longrightarrow$

$c^p(x, y)$

0	**0**	3	1	2
0	1	**0**	1	5
4	3	3	3	**0**
0	0	1	1	0
1	2	2	**0**	4

↑ 9 + 8 - 13

Compatible Prices

Compatible prices. For each node v, maintain prices $p(v)$ such that:

- (i) $c^p(x, y) \geq 0$ for for all $(x, y) \notin M$.
- (ii) $c^p(x, y) = 0$ for for all $(x, y) \in M$.

Observation 2. If p are compatible prices for a perfect matching M, then M is a min cost perfect matching.

$c(x, y)$

3	**8**	9	15	10	5
4	10	**7**	16	14	4
9	13	11	19	**10**	3
8	13	12	20	13	0
1	7	5	**11**	9	8
8	13	11	19	13	

$\longrightarrow$

$c^p(x, y)$

0	**0**	3	1	2
0	1	**0**	1	5
4	3	3	3	**0**
0	0	1	1	0
1	2	2	**0**	4

$\text{cost}(M) = \Sigma_{(x,y) \in M}\, c(x, y) = (8+7+10+8+11) = 44$

$\text{cost}(M) = \Sigma_{y \in Y}\, p(y) \; - \; \Sigma_{x \in X}\, p(x) = (8+13+11+19+13) - (5+4+3+0+8) = 44$

Maintaining Compatible Prices

Lemma 1. Let p be compatible prices for matching M. Let d be shortest path distances in G_M with costs c^p. All edges (x, y) on shortest path have $c^{p+d}(x, y) = 0$.

↑ forward or reverse edges

Pf.

- If $(x, y) \in M$, then (y, x) on shortest path and $d(x) = d(y) - c^p(x, y)$. If $(x, y) \notin M$, then (x, y) on shortest path and $d(y) = d(x) + c^p(x, y)$.
- In either case, $d(x) + c^p(x, y) - d(y) = 0$.
- By definition, $c^p(x, y) = p(x) + c(x, y) - p(y)$.
- Substituting for $c^p(x, y)$ yields: $(p(x) + d(x)) + c(x, y) - (p(y) + d(y)) = 0$.
- In other words, $c^{p+d}(x, y) = 0$. ▪

Reduced costs: $c^p(x, y) = p(x) + c(x, y) - p(y)$.

Maintaining Compatible Prices

Lemma 2. Let p be compatible prices for matching M. Let d be shortest path distances in G_M with costs c^p. Then p' = p + d are also compatible prices for M.

Pf. $(x, y) \in M$

- (y, x) is the only edge entering x in G_M. Thus, (y, x) on shortest path.
- By Lemma 1, $c^{p+d}(x, y) = 0$.

Pf. $(x, y) \notin M$

- (x, y) is an edge in G_M $\Rightarrow$ $d(y) \le d(x) + c^p(x, y)$.
- Substituting $c^p(x, y) = p(x) + c(x, y) - p(y) \ge 0$ yields $(p(x) + d(x)) + c(x, y) - (p(y) + d(y)) \ge 0$.
- In other words, $c^{p+d}(x, y) \ge 0$. ▪

Compatible prices. For each node v:
(i) $c^p(x, y) \ge 0$ for for all $(x, y) \notin M$.
(ii) $c^p(x, y) = 0$ for for all $(x, y) \in M$.

Maintaining Compatible Prices

Lemma 3. Let M' be matching obtained by augmenting along a min cost path with respect to c^{p+d}. Then p' = p + d is compatible with M'.

Pf.

- By Lemma 2, the prices p + d are compatible for M.
- Since we augment along a min cost path, the only edges (x, y) that swap into or out of the matching are on the shortest path.
- By Lemma 1, these edges satisfy $c^{p+d}(x, y) = 0$.
- Thus, compatibility is maintained. ▪

Compatible prices. For each node v:
(i) $c^p(x, y) \ge 0$ for for all $(x, y) \notin M$.
(ii) $c^p(x, y) = 0$ for for all $(x, y) \in M$.

Successive Shortest Path Algorithm

Successive shortest path.

```
Successive-Shortest-Path(X, Y, c) {
   M ← φ
   foreach x ∈ X:  p(x) ← 0                p is compatible
   foreach y ∈ Y:  p(y) ← min e into y c(e)  with M = φ

   while (M is not a perfect matching) {
      Compute shortest path distances d
      P ← shortest alternating path using costs cᵖ
      M ← updated matching after augmenting along P
      foreach v ∈ X ∪ Y:  p(v) ← p(v) + d(v)
   }
   return M
}
```


Successive Shortest Path: Analysis

Invariant. The algorithm maintains a matching M and compatible prices p.
Pf. Follows from Lemmas 2 and 3 and initial choice of prices. ▪

Theorem. The algorithm returns a min cost perfect matching.
Pf. Upon termination M is a perfect matching, and p are compatible prices. Optimality follows from Observation 2. ▪

Theorem. The algorithm can be implemented in $O(n^3)$ time.
Pf.

- Each iteration increases the cardinality of M by 1 $\Rightarrow$ n iterations.
- Bottleneck operation is computing shortest path distances d. Since all costs are nonnegative, each iteration takes $O(n^2)$ time using (dense) Dijkstra. ▪

Weighted Bipartite Matching

Weighted bipartite matching. Given weighted bipartite graph, find maximum cardinality matching of minimum weight. m edges, n nodes

Successive shortest path algorithm. $O(mn \log n)$ time using heap-based version of Dijkstra's algorithm.

Best known bounds. $O(mn^{1/2})$ deterministic; $O(n^{2.376})$ randomized.

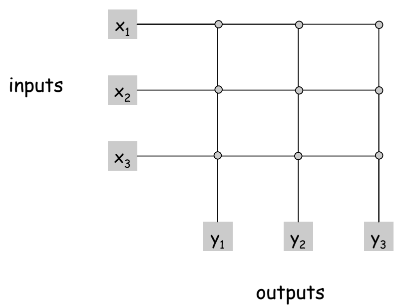
Planar weighted bipartite matching. $O(n^{3/2} \log^5 n)$.

Input Queued Switching

Input-Queued Switching

Input-queued switch.

- n inputs and n outputs in an n-by-n crossbar layout.
- At most one cell can depart an input at a time.
- At most one cell can arrive at an output at a time.
- Cell arrives at input x and must be routed to output y.

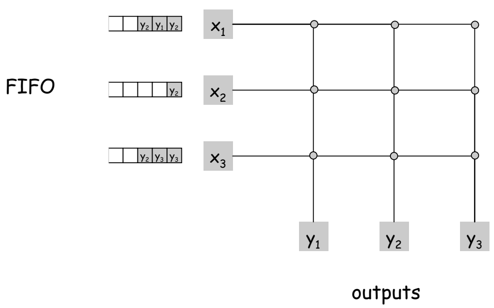


Input-Queued Switching

FIFO queueing. Each input x maintains one queue of cells to be routed.

Head-of-line blocking (HOL).

- A cell can be blocked by a cell queued ahead of it that is destined for a different output.
- Can limit throughput to 58%, even when arrivals are uniform.

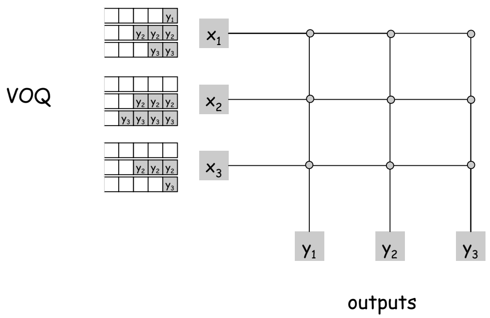


Input-Queued Switching

Virtual output queueing (VOQ). Each input x maintains n queue of cells, one for each output y.

Maximum size matching. Find a max cardinality matching.

- Achieves 100% when arrivals are uniform.
- Can starve input-queues when arrivals are non-uniform.



Input-Queued Switching

Max weight matching. Find a min cost perfect matching between inputs x and outputs y, where c(x, y) equals:

- [LQF] The number of cells waiting to go from input x to output y.
- [OCF] The waiting time of the cell at the head of VOQ from x to y.

Theorem. LQF and OCF achieve 100% throughput if arrivals are independent.

Practice.

- Too slow in practice for this application, difficult to implement in hardware. Provides theoretical framework.
- Use maximal (weighted) matching. ⇒ 2-approximation.

Reference: http://robotics.eecs.berkeley.edu/~wlr/Papers/AMMW.pdf