Algorithm II

7. Network Flow III

WU Xiaokun 吴晓堃

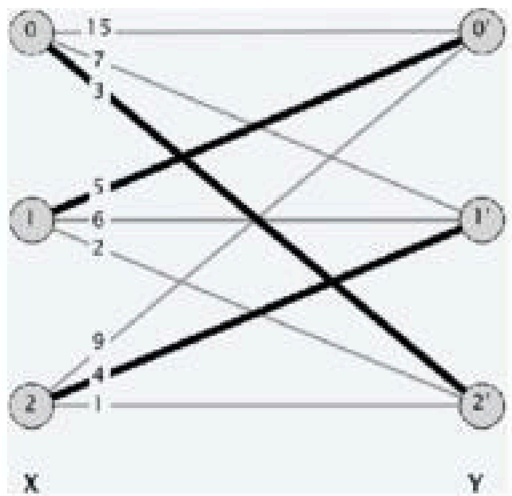
xkun.wu [at] gmail

Assignment problem

Assignment problem

Input. Weighted, complete bipartite graph $G = (X \cup Y, E)$ with $|X| = |Y|$.

Goal. Find a perfect matching of min weight.



min-cost perfect matching M = { 0-2', 1-0', 2-1' }

$cost(M) = 3 + 5 + 4 = 12$

Seminar assignment

Goal. Given m seminars and $n = 12m$ students who rank their top 8 choices, assign each student to one seminar so that:

- Each seminar is assigned exactly 12 students.
- Students tend to be "happy" with their assigned seminar.

Solution.

- Create one node for each student i and 12 nodes for each seminar j.
- Solve assignment problem where c_{ij} is some function f of the ranks:

$$c_{ij} = \begin{cases} f(rank(i,j)) & \text{if } i \text{ ranks } j \\ \infty & \text{otherwise} \end{cases}$$

Applications

Natural applications.

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

Non-obvious applications.

- Vehicle routing.
- Signal processing.
- Earth-mover's distance.
- Multiple object tracking.
- Virtual output queueing.
- Handwriting recognition.
- Locating objects in space.
- Approximate string matching.
- Enhance accuracy of solving linear systems of equations.

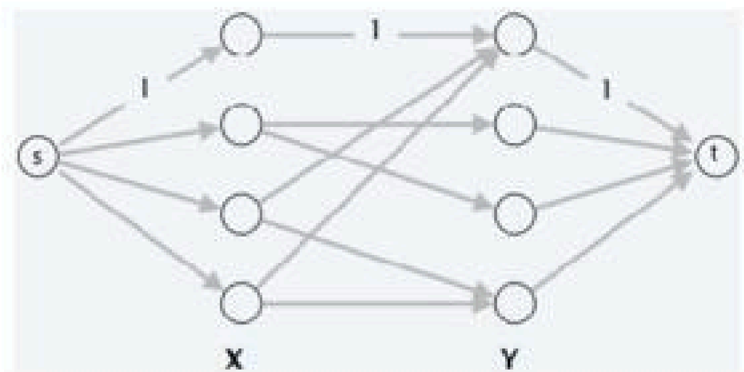
Bipartite matching

Bipartite matching. Can solve via reduction to maximum flow.

Flow. During Ford-Fulkerson, all residual 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 $(x, y) \in M$, then (y, x) is in G_M.



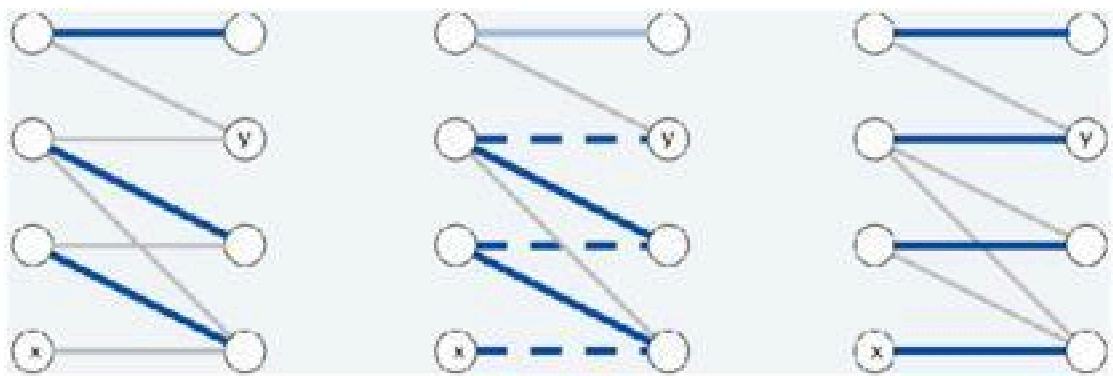
Augmenting path simplifies to:

- 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

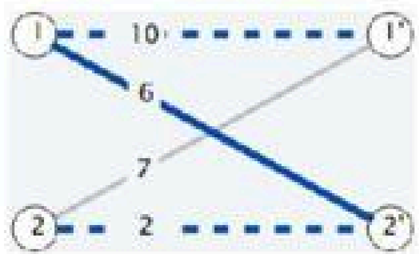
Def. An **alternating path** P with respect to a matching M is an alternating sequence of unmatched and matched edges, starting from an unmatched node $x \in X$ and going to an unmatched node $y \in Y$.

Key property. Can use P to increase by one the cardinality of the matching.
Pf. Set $M' = M \ominus P$.



Successive shortest path

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



$$P = 2 \to 2' \to 1 \to 1'$$

$$cost(P) = 2 - 6 + 10 = 6$$

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

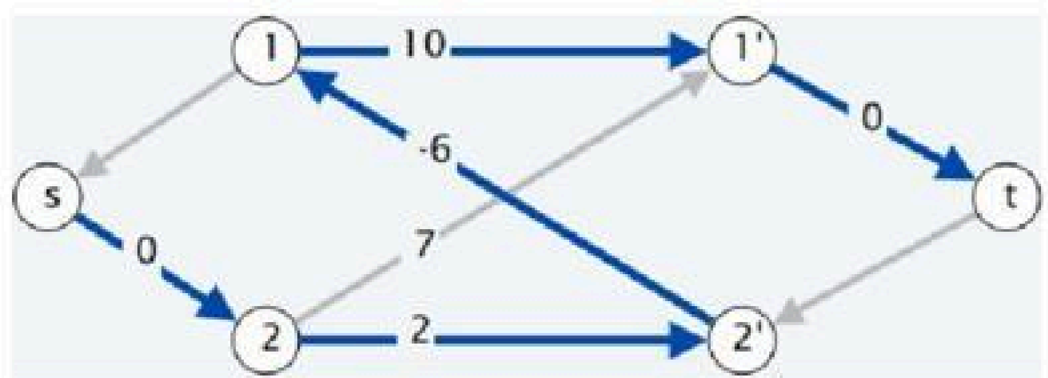
Successive shortest path algorithm.

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

Demo: Successive shortest path algorithm

Finding the shortest alternating path

Shortest alternating path. Corresponds to minimum cost $s \rightsquigarrow t$ path in G_M.



Concern. Edge costs can be negative.

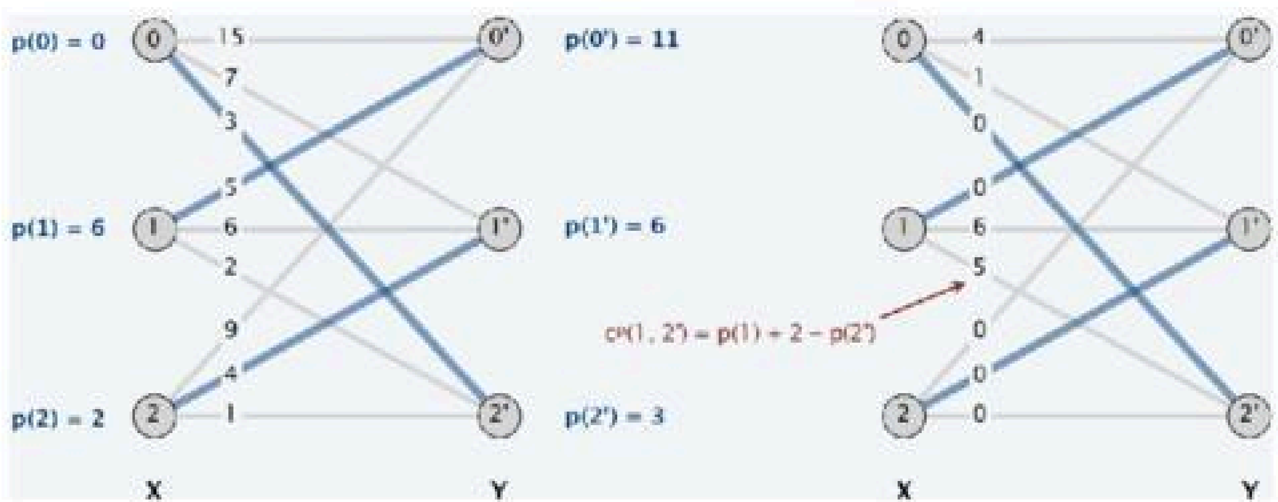
Fact. If always choose shortest alternating path, then G_M contains no negative cycles $\Rightarrow$ can compute using Bellman-Ford.

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

Equivalent assignment problem

Duality intuition. Adding a constant $p(x)$ to the cost of every edge incident to node $x \in X$ does not change the min-cost perfect matching(s).
Pf. Every perfect matching uses exactly one edge incident to node x.

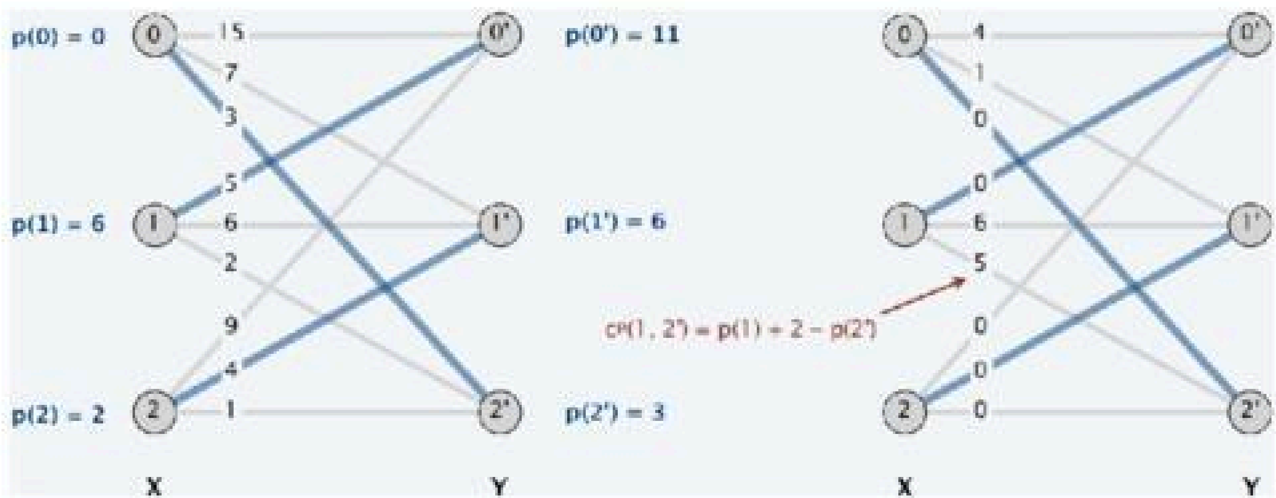
Duality intuition. Adding a constant $p(y)$ to the cost of every edge incident to node $y \in Y$ does not change the min-cost perfect matching(s).
Pf. Every perfect matching uses exactly one edge incident to node y.



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.

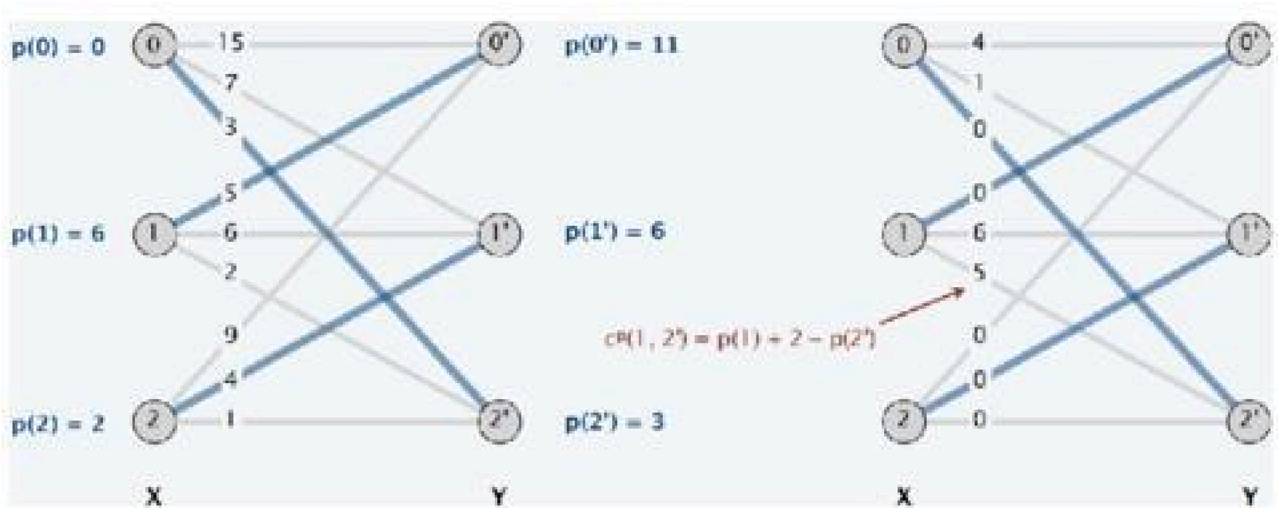


Compatible prices

Compatible prices. For each node $v \in X \cup Y$, maintain prices $p(v)$ such that:

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

Observation 2. If prices p are compatible with a perfect matching M, then M is a min-cost perfect matching.
Pf. Matching M has 0 cost.



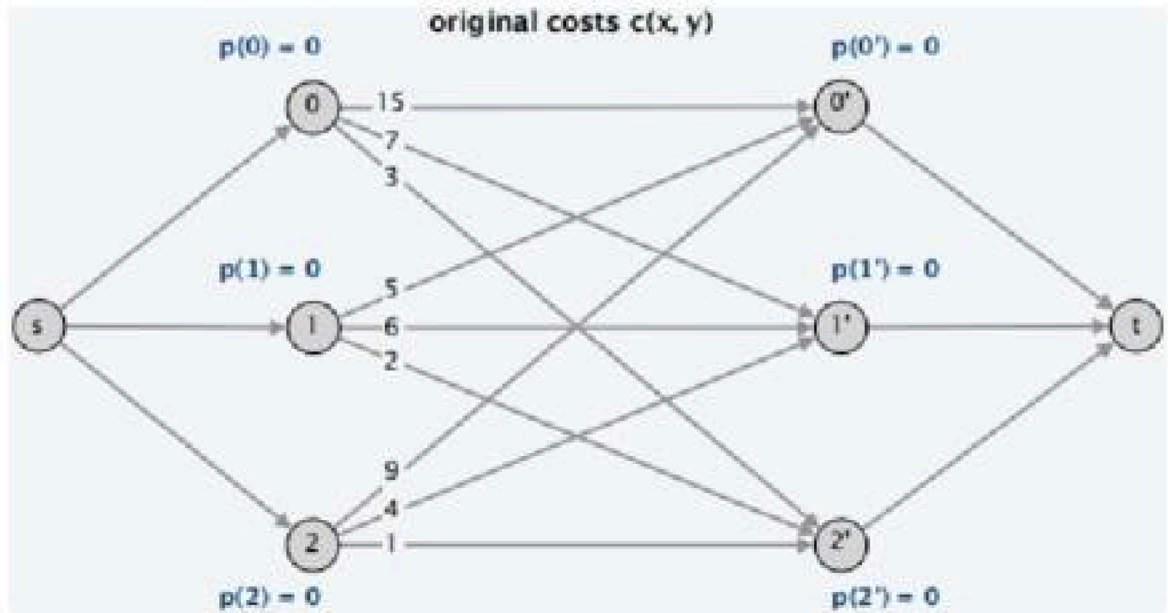
Successive shortest path: algorithm

SUCCESSIVE-SHORTEST-PATH (X, Y, c)

1. $M = \emptyset$;
2. FOREACH $v \in X \cup Y : p(v) = 0$;
3. WHILE (M is not a perfect matching)
    1. d = shortest path distances using costs c^p;
    2. P = shortest alternating path using costs c^p;
    3. M = updated matching after augmenting along P;
    4. FOREACH $v \in X \cup Y : p(v) = p(v) + d(v)$; RETURN M;

≡

Successive shortest path: demo



Maintaining compatible prices 1

Lemma 1. Let p be compatible prices for 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.

Pf. Let (x, y) be some edge on shortest path.

- 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, $c^{p+d}(x, y) = 0$.

≡

Maintaining compatible prices 2

Lemma 2. Let p be compatible prices for 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 $G_M \Rightarrow d(y) \leq d(x) + c^p(x, y)$.
- Substituting $c^p(x, y) = p(x) + c(x, y) - p(y) \geq 0$ yields $(p(x) + d(x)) + c(x, y) - (p(y) + d(y)) \geq 0$.
  - In other words, $c^{p+d}(x, y) \geq 0$.

≡

Maintaining compatible prices 3

Lemma 3. Let p be compatible prices for M and let M' be matching obtained by augmenting along a min cost path with respect to c^{p+d}. Then $p' = p + d$ are compatible prices for 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 min-cost path.
- By LEMMA 1, these edges satisfy $c^{p+d}(x, y) = 0$.
- Thus, compatibility is maintained.

Successive shortest path: analysis

Invariant. The algorithm maintains a matching M and compatible prices p.
Pf. Follows from LEMMA 2 and LEMMA 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 a weighted bipartite graph with n nodes and m edges, find a maximum cardinality matching of minimum weight.

Theorem. [Fredman-Tarjan 1987] The successive shortest path algorithm solves the problem in $O(n^2 + mn \log n)$ time using Fibonacci heaps.

Theorem. [Gabow-Tarjan 1989] There exists an $O(mn^{1/2} \log(nC))$ time algorithm for the problem when the costs are integers between 0 and C.

Input-queued switching

Input-queued switching Problem

Input-queued switch.

- n input ports and n output ports 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.

Application. High-bandwidth switches.