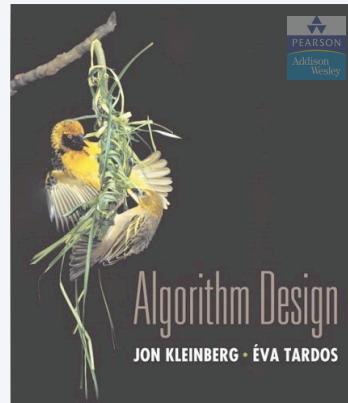


7. NETWORK FLOW III

- ▶ assignment problem
- ▶ input-queued switching

Lecture slides by Kevin Wayne
Copyright © 2005 Pearson-Addison Wesley
Copyright © 2013 Kevin Wayne
<http://www.cs.princeton.edu/~wayne/kleinberg-tardos>

Last updated on Sep 8, 2013 6:40 AM



7. NETWORK FLOW III

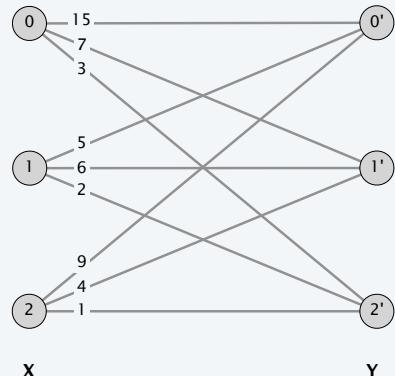
- ▶ assignment problem
- ▶ input-queued switching

SECTION 7.13

Assignment problem

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

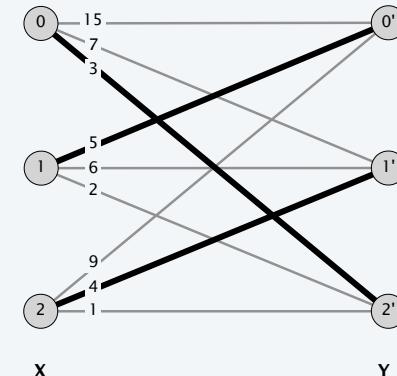
Goal. Find a perfect matching of min weight.



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
 $M = \{ 0-2', 1-0', 2-1' \}$
 $\text{cost}(M) = 3 + 5 + 4 = 12$

Princeton writing seminars

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 of the ranks:

$$c_{ij} = \begin{cases} f(\text{rank}(i, j)) & \text{if } i \text{ ranks } j \\ \infty & \text{if } i \text{ does not rank } j \end{cases}$$

Title	Course #	Professor	Day/Time	Location
1980s, The	WRI 168	Scott, Andrea	M/W 1:30pm-2:50pm	Hargadon G002
America and the Melting Pot	WRI 157	Skinazi, Karen	T/TH 8:30am-9:50am	Butler 026
America and the Melting Pot	WRI 158	Skinazi, Karen	T/TH 11:00am-12:20pm	Hargadon G004
American Mysticism	WRI 191	Laufenberg, George	T/TH 7:30pm-8:50pm	99 Alexander 101
American Revolutions	WRI 184	Grosghal, Dov	M/W 8:30am-9:50am	Butler 026
Animal Mind, The	WRI 101	Gould, James	M/W 8:30am-9:50am	Blair T3
Art of Adventure, The	WRI 151	Moffitt, Anne	T/TH 11:00am-12:20pm	Butler 027

5

Kidney exchange

If a donor and recipient have a different blood type, they can exchange their kidneys with another donor and recipient pair in a similar situation.

Can also be done among multiple pairs (or starting with an altruistic donor).



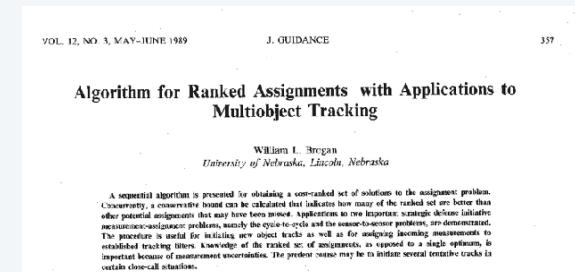
7

Locating objects in space

Goal. Given n objects in 3d space, locate them with 2 sensors.

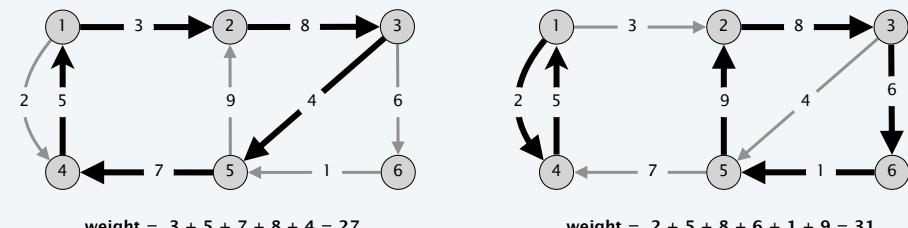
Solution.

- Each sensor computes line from it to each particle.
- Let c_{ij} = distance between line i from censor 1 and line j from sensor 2.
- Due to measurement errors, we might have $c_{ij} > 0$.
- Solve assignment problem to locate n objects.



6

Kidney exchange



8

Applications

Natural applications.

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

Non-obvious applications.

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

9

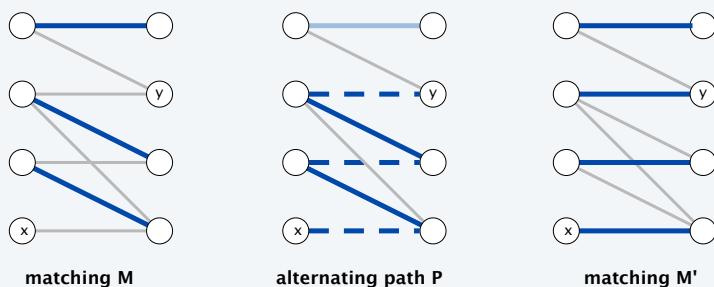
Alternating path

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 $M' = M \oplus P$.

symmetric difference



11

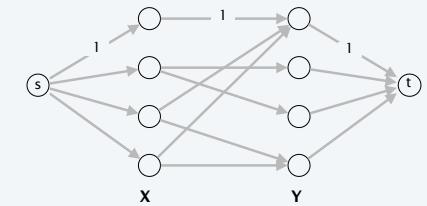
Bipartite matching

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 .



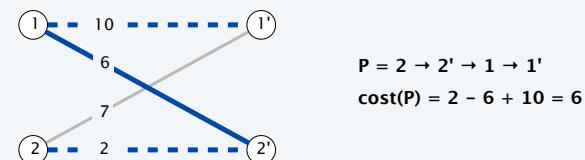
Augmenting path simplifies to:

- 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 .

10

Assignment problem: successive shortest path algorithm

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



$$P = 2 \rightarrow 2' \rightarrow 1 \rightarrow 1' \\ \text{cost}(P) = 2 - 6 + 10 = 6$$

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

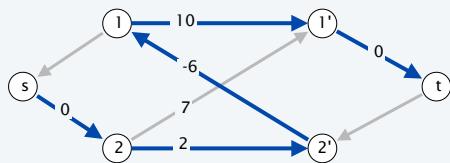
Successive shortest path algorithm.

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

12

Finding the shortest alternating path

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



Concern. Edge costs can be negative.

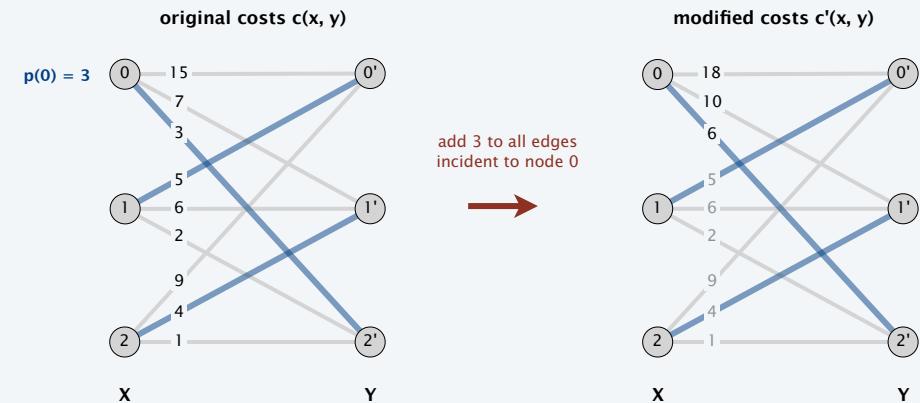
Fact. If always choose shortest alternating path, then G_M contains no negative cycles \Rightarrow can compute using Bellman-Ford.

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 . ■



13

14

Equivalent assignment problem

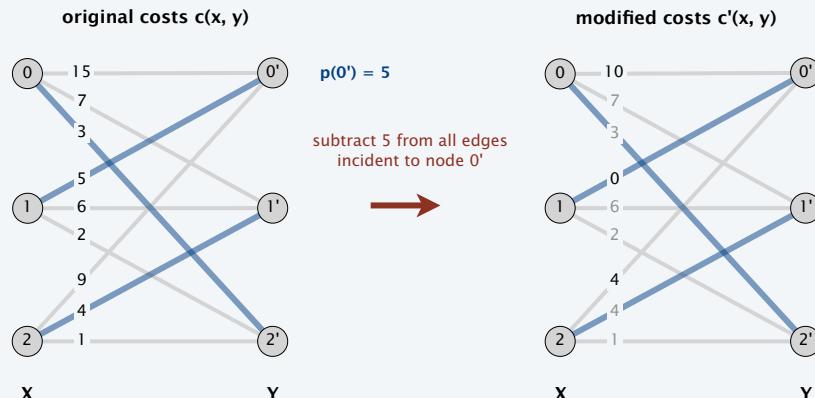
Duality intuition. Subtracting a constant $p(y)$ to the cost of every edge incident to node $y \in Y$ does not change the min-cost perfect matching(s).

Pf. Every perfect matching uses exactly one edge incident to node y . ■

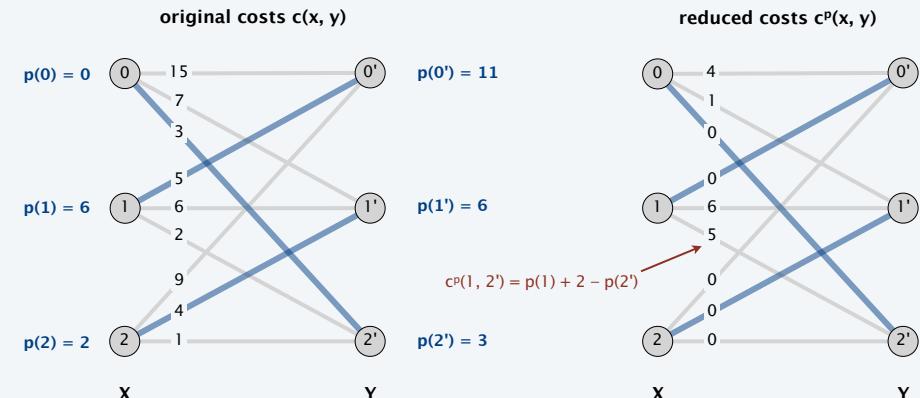
Reduced costs

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.



15



16

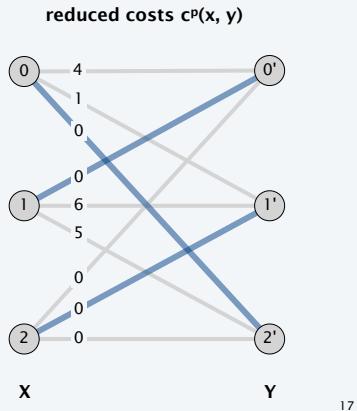
Compatible prices

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. ■

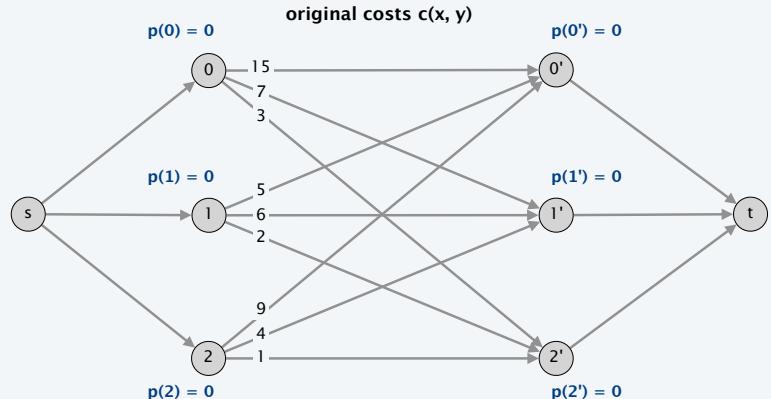


17

Successive shortest path algorithm

Initialization.

- $M = \emptyset$.
- For each $v \in X \cup Y$: $p(v) \leftarrow 0$.



19

Successive shortest path algorithm

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

$M \leftarrow \emptyset$.

FOREACH $v \in X \cup Y$: $p(v) \leftarrow 0$.

prices p are compatible with M
 $c^p(x, y) = c(x, y) \geq 0$

WHILE (M is not a perfect matching)

$d \leftarrow$ shortest path distances using costs c^p .

$P \leftarrow$ shortest alternating path using costs c^p .

$M \leftarrow$ updated matching after augmenting along P .

FOREACH $v \in X \cup Y$: $p(v) \leftarrow p(v) + d(v)$.

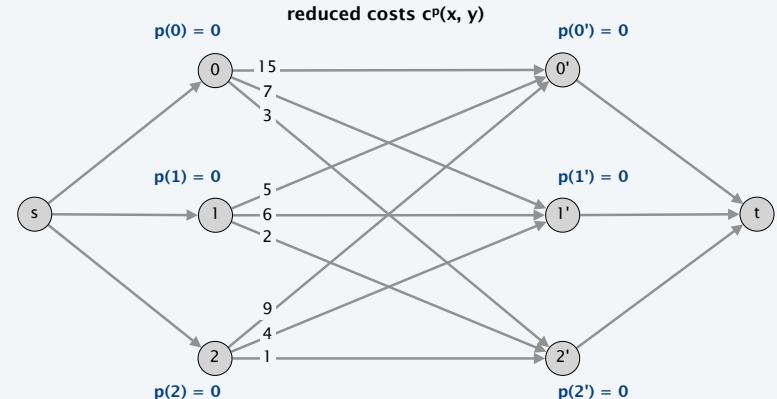
RETURN M .

18

Successive shortest path algorithm

Initialization.

- $M = \emptyset$.
- For each $v \in X \cup Y$: $p(v) \leftarrow 0$.

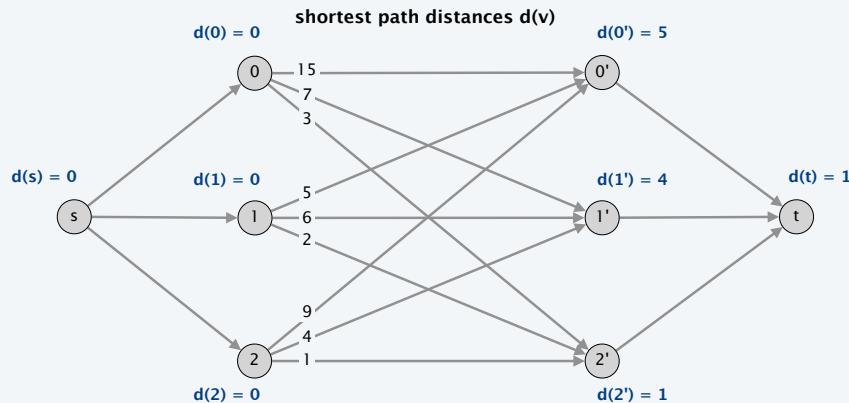


20

Successive shortest path algorithm

Step 1.

- Compute shortest path distances $d(v)$ from s to v using $c^p(x, y)$.
- Update matching M via shortest path from s to t .
- For each $v \in X \cup Y$: $p(v) \leftarrow p(v) + d(v)$.

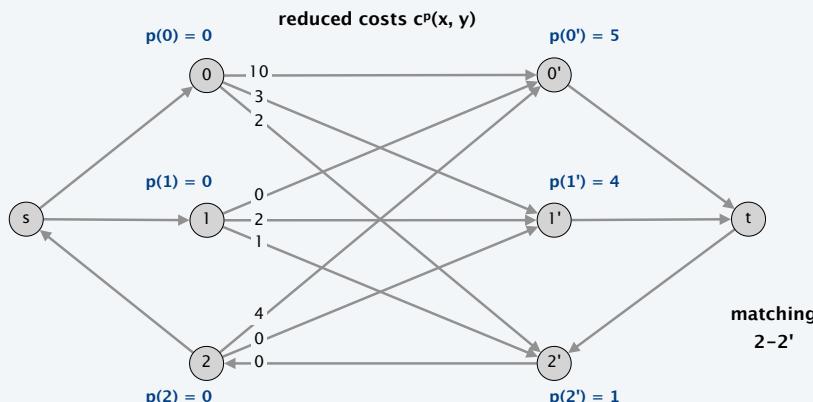


21

Successive shortest path algorithm

Step 1.

- Compute shortest path distances $d(v)$ from s to v using $c^p(x, y)$.
- Update matching M via shortest path from s to t .
- For each $v \in X \cup Y$: $p(v) \leftarrow p(v) + d(v)$.

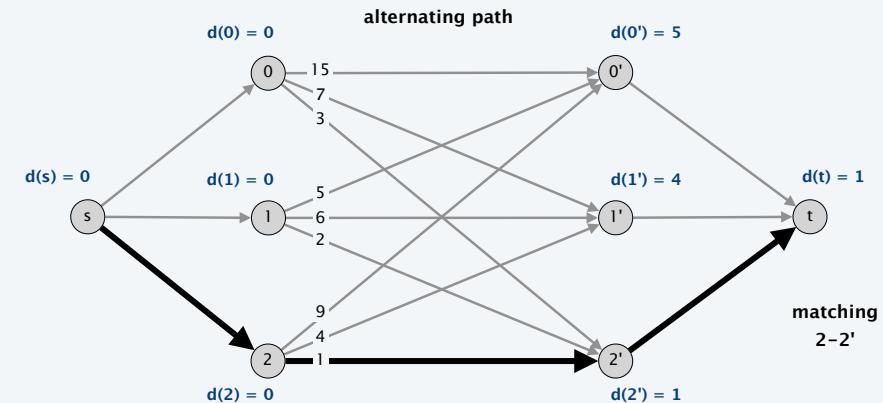


23

Successive shortest path algorithm

Step 1.

- Compute shortest path distances $d(v)$ from s to v using $c^p(x, y)$.
- Update matching M via shortest path from s to t .
- For each $v \in X \cup Y$: $p(v) \leftarrow p(v) + d(v)$.

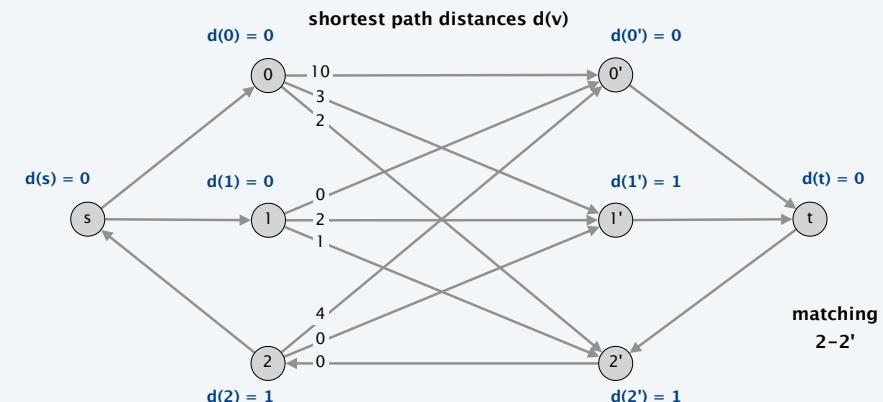


22

Successive shortest path algorithm

Step 2.

- Compute shortest path distances $d(v)$ from s to v using $c^p(x, y)$.
- Update matching M via shortest path from s to t .
- For each $v \in X \cup Y$: $p(v) \leftarrow p(v) + d(v)$.

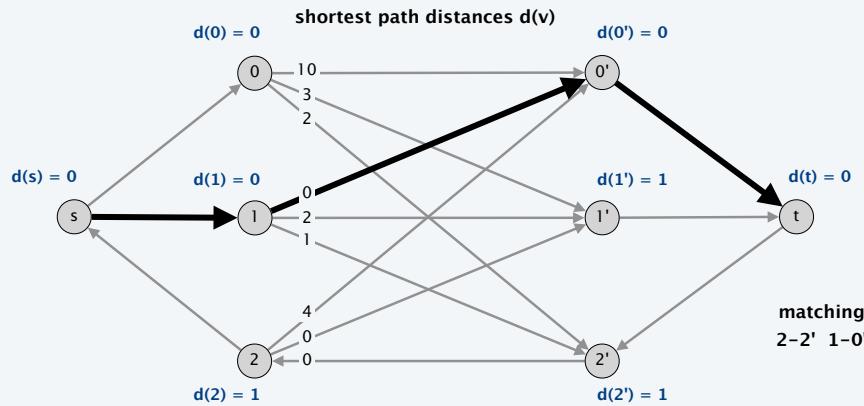


24

Successive shortest path algorithm

Step 2.

- Compute shortest path distances $d(v)$ from s to v using $c^p(x, y)$.
- Update matching M via shortest path from s to t .
- For each $v \in X \cup Y$: $p(v) \leftarrow p(v) + d(v)$.

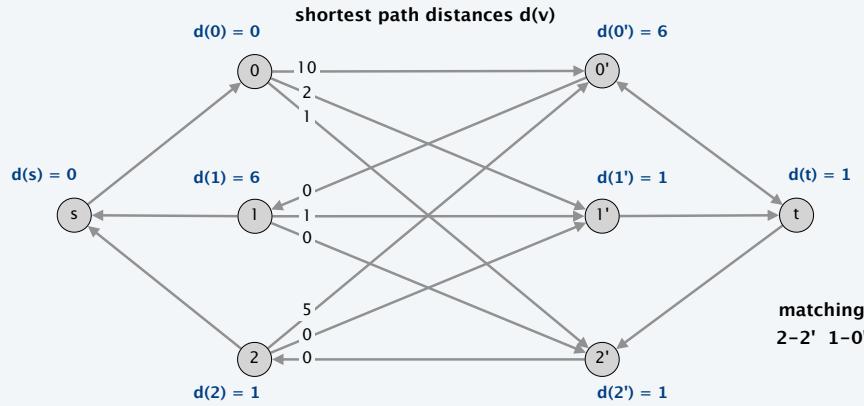


25

Successive shortest path algorithm

Step 3.

- Compute shortest path distances $d(v)$ from s to v using $c^p(x, y)$.
- Update matching M via shortest path from s to t .
- For each $v \in X \cup Y$: $p(v) \leftarrow p(v) + d(v)$.

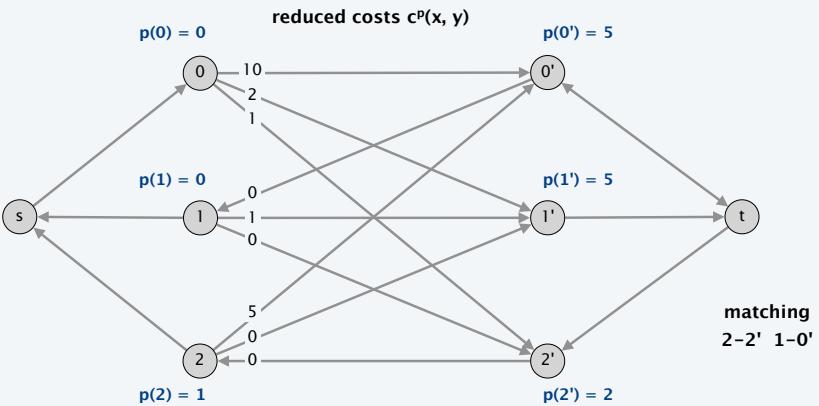


27

Successive shortest path algorithm

Step 2.

- Compute shortest path distances $d(v)$ from s to v using $c^p(x, y)$.
- Update matching M via shortest path from s to t .
- For each $v \in X \cup Y$: $p(v) \leftarrow p(v) + d(v)$.

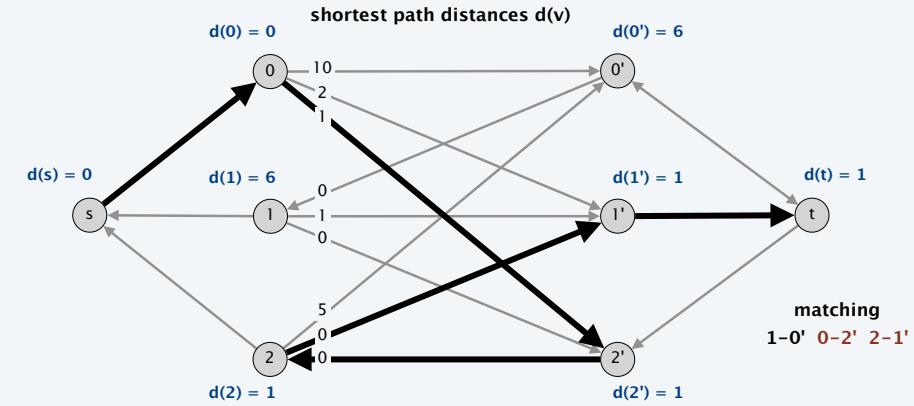


26

Successive shortest path algorithm

Step 3.

- Compute shortest path distances $d(v)$ from s to v using $c^p(x, y)$.
- Update matching M via shortest path from s to t .
- For each $v \in X \cup Y$: $p(v) \leftarrow p(v) + d(v)$.

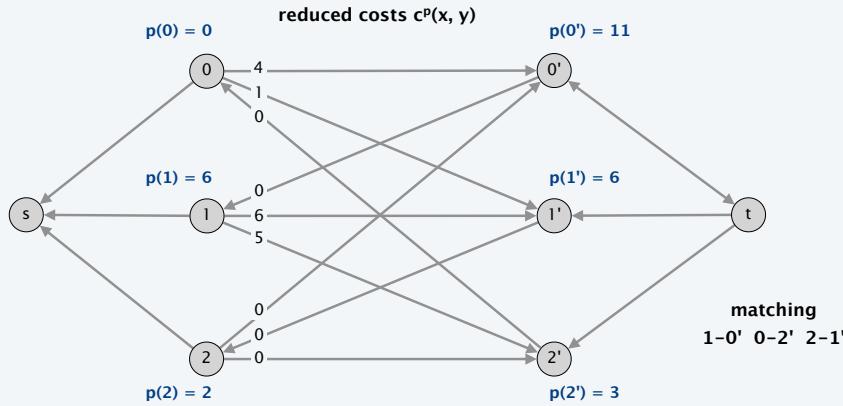


28

Successive shortest path algorithm

Step 3.

- Compute shortest path distances $d(v)$ from s to v using $c^p(x, y)$.
- Update matching M via shortest path from s to t .
- For each $v \in X \cup Y$: $p(v) \leftarrow p(v) + d(v)$.



29

Maintaining compatible prices

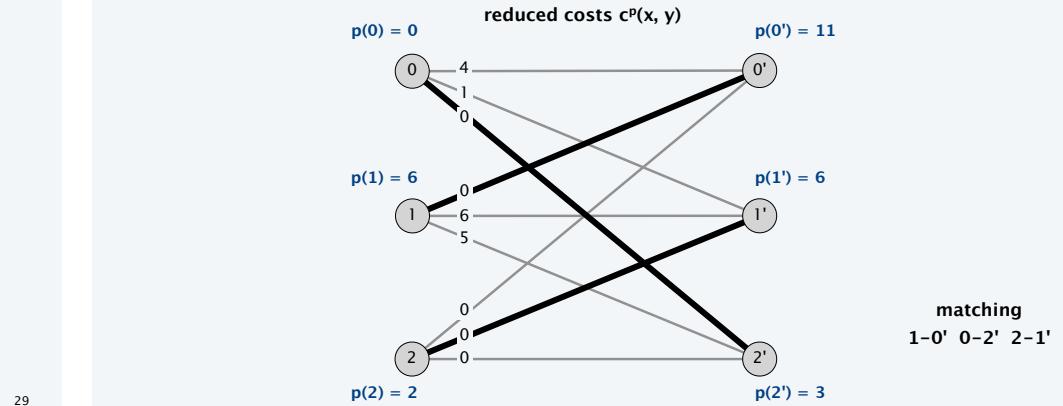
Lemma 1. Let p be compatible prices for M . Let d be shortest path distances in G_M with costs c^p . All edges (x, y) on shortest path have $c^{p+d}(x, y) = 0$.

↗
forward or reverse edges

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, $c^{p+d}(x, y) = 0$. ■

Given prices p , the reduced cost of edge (x, y) is
 $c^p(x, y) = p(x) + c(x, y) - p(y)$.



30

Maintaining compatible prices

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, $c^{p+d}(x, y) \geq 0$. ■

Prices p are compatible with matching M :

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

31

32

Maintaining compatible prices

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. ▀

Prices p are compatible with matching M :

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

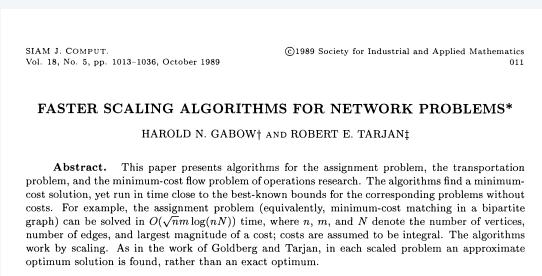
33

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 .



35

Successive shortest path algorithm: 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. ▀

34

History

Thorndike 1950. Formulated in a modern way by a psychologist.

PSYCHOMETRIKA—VOL. 15, NO. 3
SEPTEMBER, 1950

THE PROBLEM OF CLASSIFICATION OF PERSONNEL*

ROBERT L. THORNDIKE
TEACHERS COLLEGE, COLUMBIA UNIVERSITY

The personnel classification problem arises in its pure form when all job applicants must be used, being divided among a number of job categories. The use of tests for classification involves problems of two types: (1) problems concerning the design, choice, and weighting of tests into a battery, and (2) problems of establishing the optimum administrative procedure of using test results for assignment. A consideration of the first problem emphasizes the desirability of using simple, factorially pure tests which may be expected to have a wide range of validities for different job categories. In the use of test results for assignment, an initial problem is that of expressing predictions of success in different jobs in comparable score units. These units should take account of predictor validity and of job importance. Procedures are described for handling assignment either in terms of daily quotas or in terms of a stable predicted yield.

Assign individuals to jobs to maximize average success of all individuals.

36

History

Thorndike 1950. Formulated in a modern way by a psychologist.

There are, as has been indicated, a finite number of permutations in the assignment of men to jobs. When the classification problem as formulated above was presented to a mathematician, he pointed to this fact and said that from the point of view of the mathematician there was no problem. Since the number of permutations was finite, one had only to try them all and choose the best. He dismissed the problem at that point. This is rather cold comfort to the psychologist, however, when one considers that only ten men and ten jobs mean over three and a half million permutations. Trying out all the permutations may be a mathematical solution to the problem, it is not a practical solution.

anticipated theory of computational complexity!

37

History

Jacobi (1804-1851). Introduces a bound on the order of a system of m ordinary differential equations in m unknowns and reduces it to....

De investigando ordine systematis aequationum differentialium vulgarium cuiuscunq;e.
(Ex. ill. C. G. J. Jacobi manuscriptis posthumis in medium protulit*) C. W. Borchardi.)

Investigatio ad solvendum problema inaequalitatum reducitur.
Systema aequationum differentialium vulgarium est non canonicum**, si aequationes altissima variabilium dependentia differentialia tali modo continent, ut horum valores ex iis petere non licent. Id quod fit, quod aequationes nonnullas altissimis illis differentialibus carentes in systemate proposito vel ipse inveniuntur vel eliminatione ex eo obtinuerint. *Eo casu numerus Constantium Arbitriariorum, quem integratio completa inducit, sive ordo systematis semper minor est summa altissimorum ordinum, ad quos differentialia singularium variabilium in aequationibus differentialibus propositis accident.* Qui ordo systematis cognoscitur, si per differentiationes et eliminationes contingit systema propositum redigere in aliud forma canonica gaudens eique nequivaleat, ita ut de systemate canonico etiam ad propositum reditus patet. Nam summa altissimorum ordinum, ad quos in systemate canonico differentialia singularium variabilium dependentia ascendent, etiam systematis propositio non canonici ordo erit. Ad quem ordinem investigandum non tamen opus est ea ad formam canonican reductione, sed res per considerationes sequentes absolvit potest.

Ponamus inter variabiles independentem t atque n variabiles dependentes x_1, x_2, \dots, x_n , haberi n aequationes differentiales:
(1.) $u_1 = 0, u_2 = 0, \dots, u_n = 0,$
sicut
$$h^{(1)} \\ h^{(2)} \\ \vdots \\ h^{(n)}$$
 altissimus ordo, ad quem in aequatione $u_i = 0$ differentialia variabilis x_i ascen-

Looking for the order of a system of arbitrary ordinary differential equations

History

Kuhn 1955. First poly-time algorithm; named "Hungarian" algorithm to honor two Hungarian mathematicians (Kőnig and Egerváry).

Munkres 1957. Reviewed algorithm; observed $O(n^4)$ implementation.

Edmonds-Karp, Tomizawa 1971. Improved to $O(n^3)$.

THE HUNGARIAN METHOD FOR THE ASSIGNMENT PROBLEM¹

H. W. Kuhn
Bryn Mawr College

Assuming that numerical scores are available for the performance of each of n persons on each of n jobs, the "assignment problem" is the quest for an assignment of persons to jobs so that the sum of the n scores so obtained is as large as possible. It is shown that ideas latent in the work of two Hungarian mathematicians may be exploited to yield a new method of solving this problem.

anticipated development of combinatorial optimization

38

History

Jacobi (1804-1851). The assignment problem! Moreover, he provides a polynomial-time algorithm.

Problem.

Diapontantur n quantitates $h_k^{(i)}$ quaecunque in schema Quadrati, ita ut habeantur n series horizontales et n series verticulares quarum queaque est n terminorum. Ex illis quantitatibus eligantur n transversales i. e. in series horizontibus simul atque verticalibus diversis positae, quod fieri potest $1 \cdot 2 \dots n$ modis; ex omnibus illis modis querendus est qui summam n numerorum electorum suppediet maximam.

Dispositis quantitatibus $h_k^{(i)}$ in figuram quadratam

h'_1	h'_2	\dots	h'_n
h''_1	h''_2	\dots	h''_n
\dots	\dots	\dots	\dots
$h^{(n)}_1$	$h^{(n)}_2$	\dots	$h^{(n)}_n$

carum systema appellabo schema propositum; omne schema inde ortum addendo singulis ejusdem serierum horizontalium terminis eandem quantitatem appellabo scheme derreatum. Sit

ℓ^0

quantitas addenda terminis ℓ^0 seriei horizontali, quo facto singula $1 \cdot 2 \dots n$ aggregata transversala, inter quae maximum eligendum est, eadem augebuntur quantitate

$$\ell^0 + \ell^1 + \dots + \ell^n = L,$$

quippe ad singula aggregata formanda e quaque serie horizontali unus eligendum est terminus. Quia de re si statur

$$h_k^{(1)} + h_k^{(2)} + \dots + h_k^{(n)} = p_k^{(i)}$$

aliquae aggregatus transversalis maximum et terminis $h_k^{(i)}$ formatum

$$h_1^{(i)} + h_2^{(i)} + \dots + h_n^{(i)} = H,$$

fit valor aggregati transversalis maximus e terminis $p_k^{(i)}$ formati

$$p_1^{(i)} + p_2^{(i)} + \dots + p_n^{(i)} = H + L$$

Problem.

We dispose nn arbitrary quantities $h_k^{(i)}$ in a square table in such a way that we have n horizontal series and n vertical series having each one n terms. Among these quantities, to chose n being transversal, that is all disposed in different horizontal and vertical series, which may be done in $1 \cdot 2 \dots n$ ways; and among these ways, to research one that gives the maximum of the sum of the n chosen numbers.

h'_1	h'_2	\dots	h'_n
h''_1	h''_2	\dots	h''_n
\dots	\dots	\dots	\dots
$h^{(n)}_1$	$h^{(n)}_2$	\dots	$h^{(n)}_n$

we can add to each term of the same horizontal series a same quantity, and we call $\ell^{(i)}$ the quantity added to the terms of the i^{th} horizontal series. This being done, each of the $1 \cdot 2 \dots n$ transversal sums among which we need to find a maximum is increased by the same quantity

$$\ell^0 + \ell^1 + \dots + \ell^{(n)} = L,$$

because, in order to form these sums, we need to pick a term in each horizontal series. Hence, if we use

$$h_k^{(i)} + \ell^{(i)} = p_k^{(i)}$$

and that the maximal transversal sum of the terms $h_k^{(i)}$ is

$$h_1^{(i_1)} + h_2^{(i_2)} + \dots + h_n^{(i_n)} = H,$$

this makes that the value of the maximal sum formed with the $p_k^{(i)}$ is

$$p_1^{(i_1)} + p_2^{(i_2)} + \dots + p_n^{(i_n)} = H + L,$$

Jacobi formulated the assignment problem; proposed and analyzed the Hungarian algorithm

40

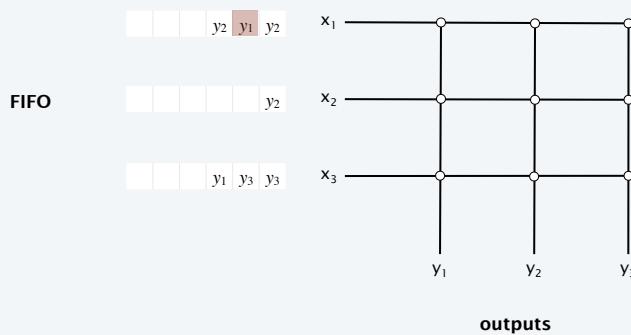
7. NETWORK FLOW III

- ▶ assignment problem
- ▶ input-queued switching

FIFO queuing

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.

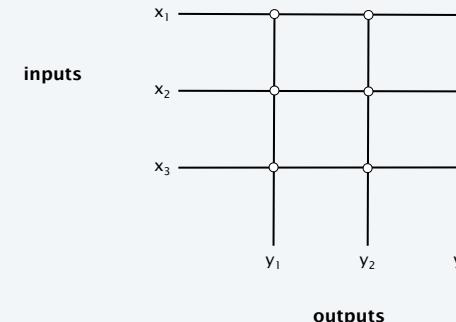


Input-queued switching

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.



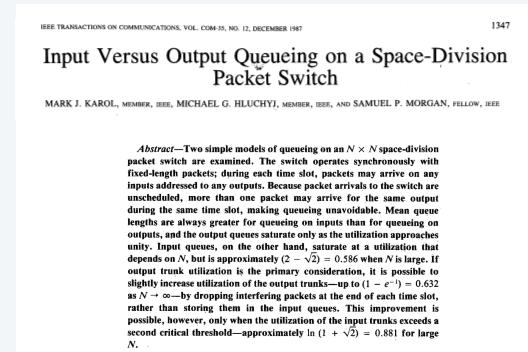
42

FIFO queuing

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.

Fact. FIFO can limit throughput to 58% even when arrivals are uniform i.i.d.

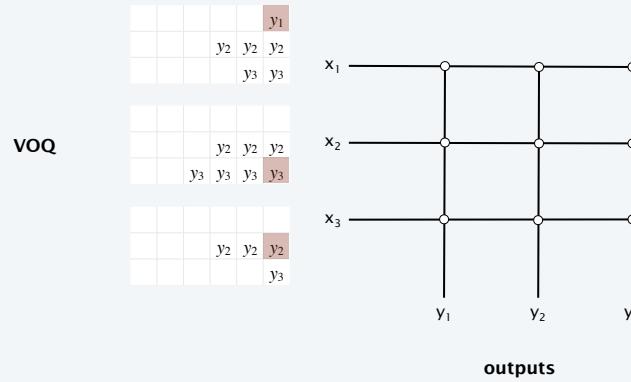


Virtual output queueing

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

Maximum size matching. Find a max cardinality matching.

Fact. VOQ achieves 100% throughput when arrivals are uniform i.i.d. but can starve input-queues when arrivals are nonuniform.



45

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 (even if not uniform).

Achieving 100% Throughput in an Input-Queued Switch

Nick McKeown, Senior Member, IEEE; Adisak Mekkittikul, Member, IEEE; Venkat Anantharam, Fellow, IEEE; and Jean Walrand, Fellow, IEEE

Abstract — It is well known that head-of-line blocking limits the throughput of an input-queued switch with first-in-first-out (FIFO) queues. Under certain conditions, the throughput can be increased by using non-FIFO queuing policies. It is shown here that if non-FIFO queuing policies are used, the throughput can be increased to 100% for all independent traffic patterns. If a suitable queuing policy and scheduling algorithm are used, then it is possible to achieve 100% throughput for all independent traffic patterns. The proof is based on a linear programming argument involving a simple linear programming argument and quadratic Lyapunov functions. The proof also shows that the switch can be implemented using a separate FIFO queue for each output and that the switch is scheduled using a maximum weight bipartite matching algorithm. Two algorithms for achieving 100% throughput are presented: longest queue first (LQF) and oldest cell first (OCF). Both algorithms favor queues with larger occupancy, ensuring that larger queues receive more bandwidth. The OCF algorithm, however, can lead to the permanent starvation of short queues. OCF overcomes this limitation by favoring cells with large waiting times.

Index Terms — Arbitration, ATM, input-queued switch, input-queued packet switch, queuing networks, scheduling algorithm.

46