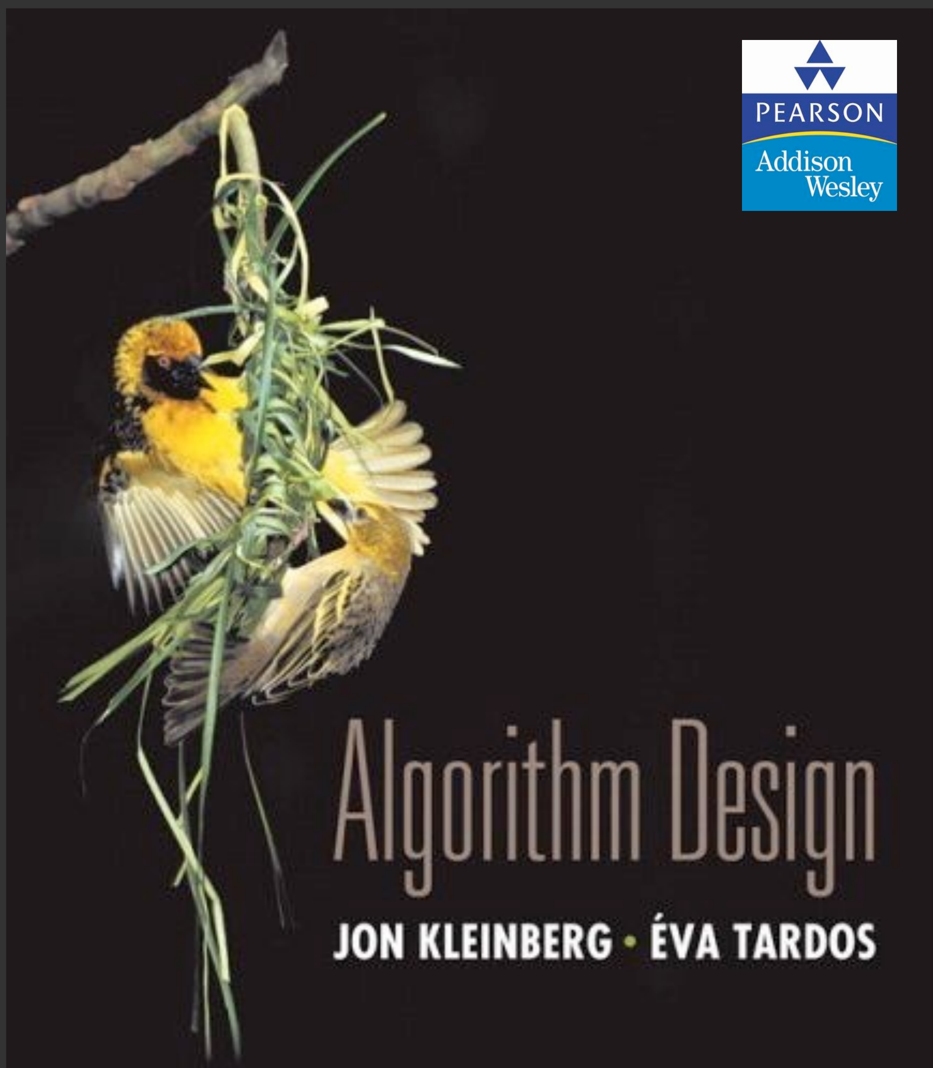


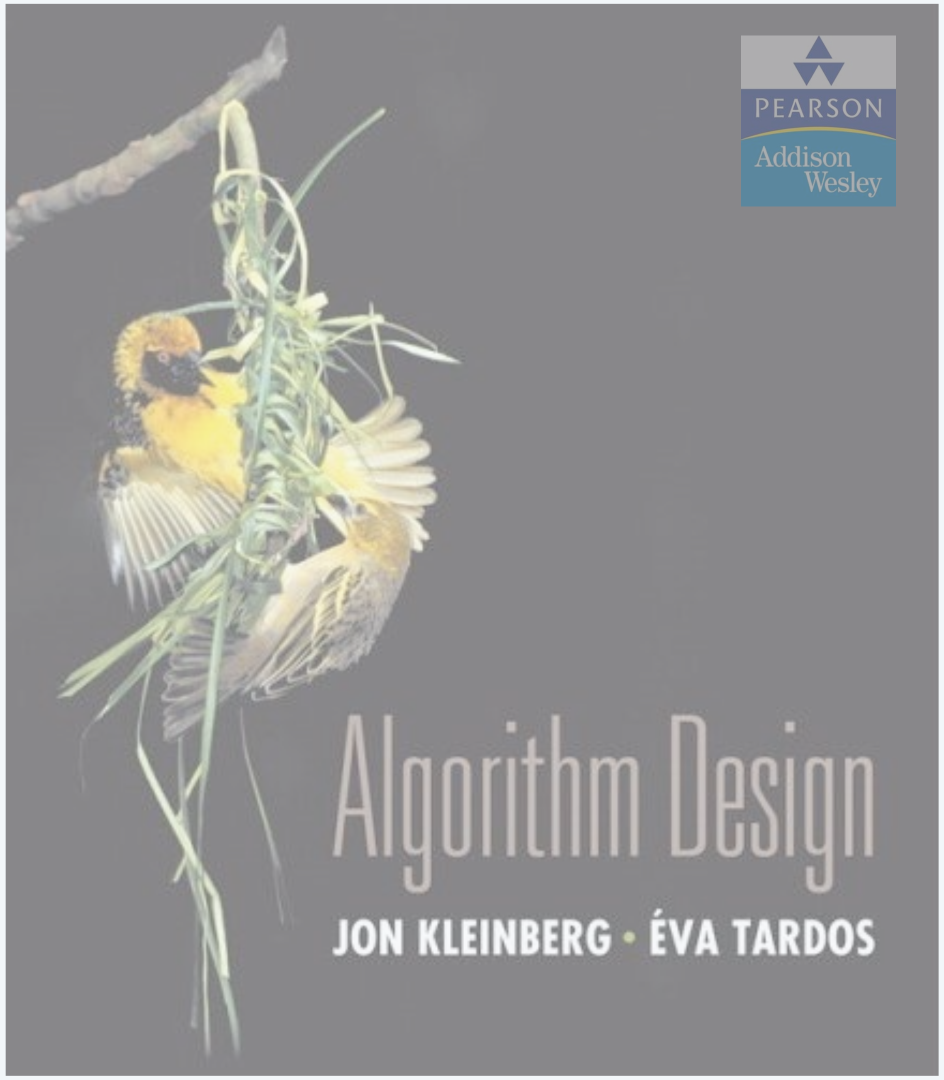
7. Network Flow III

- *assignment problem*
- *input-queued switching*

Lecture slides by Kevin Wayne

http://www.cs.princeton.edu/~wayne/kleinberg-tardos

Last updated on 7/25/17 11:04 AM



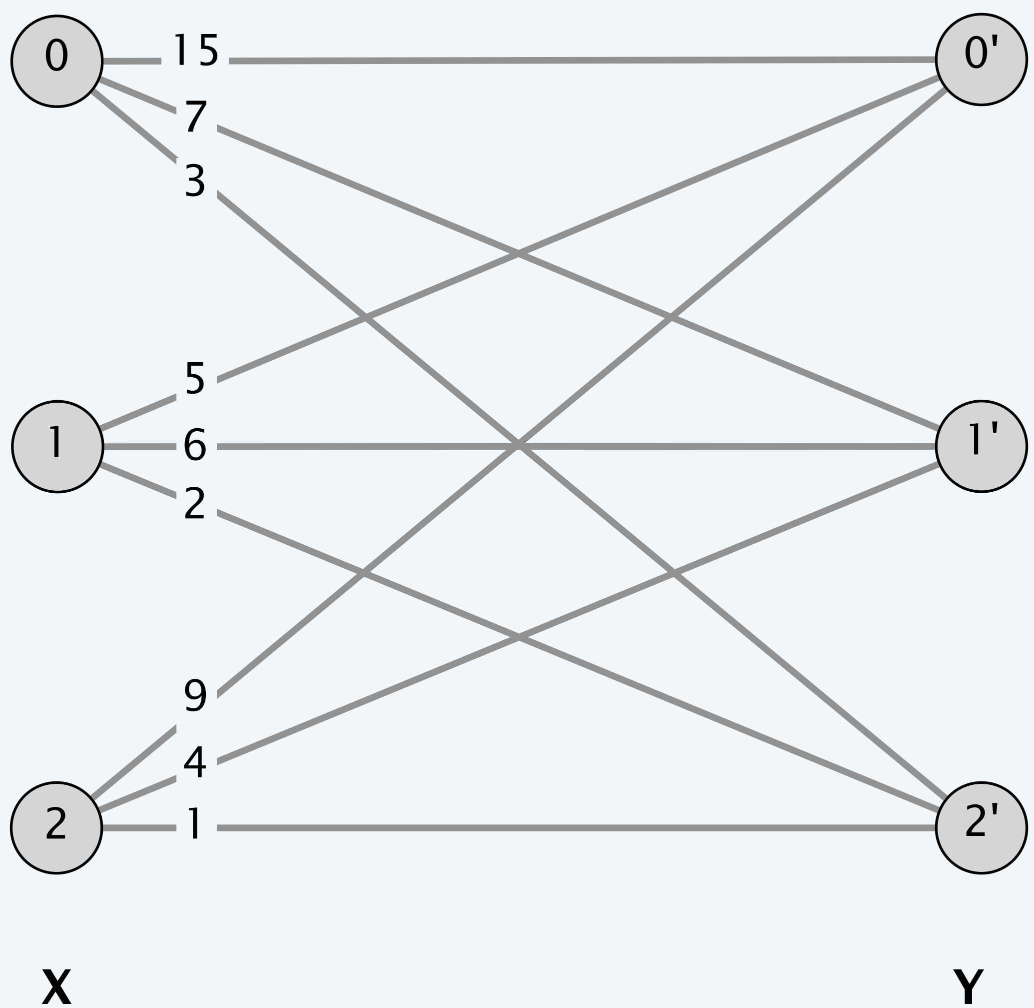
7. Network Flow III

- *assignment problem*
- *input-queued switching*

Assignment problem

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

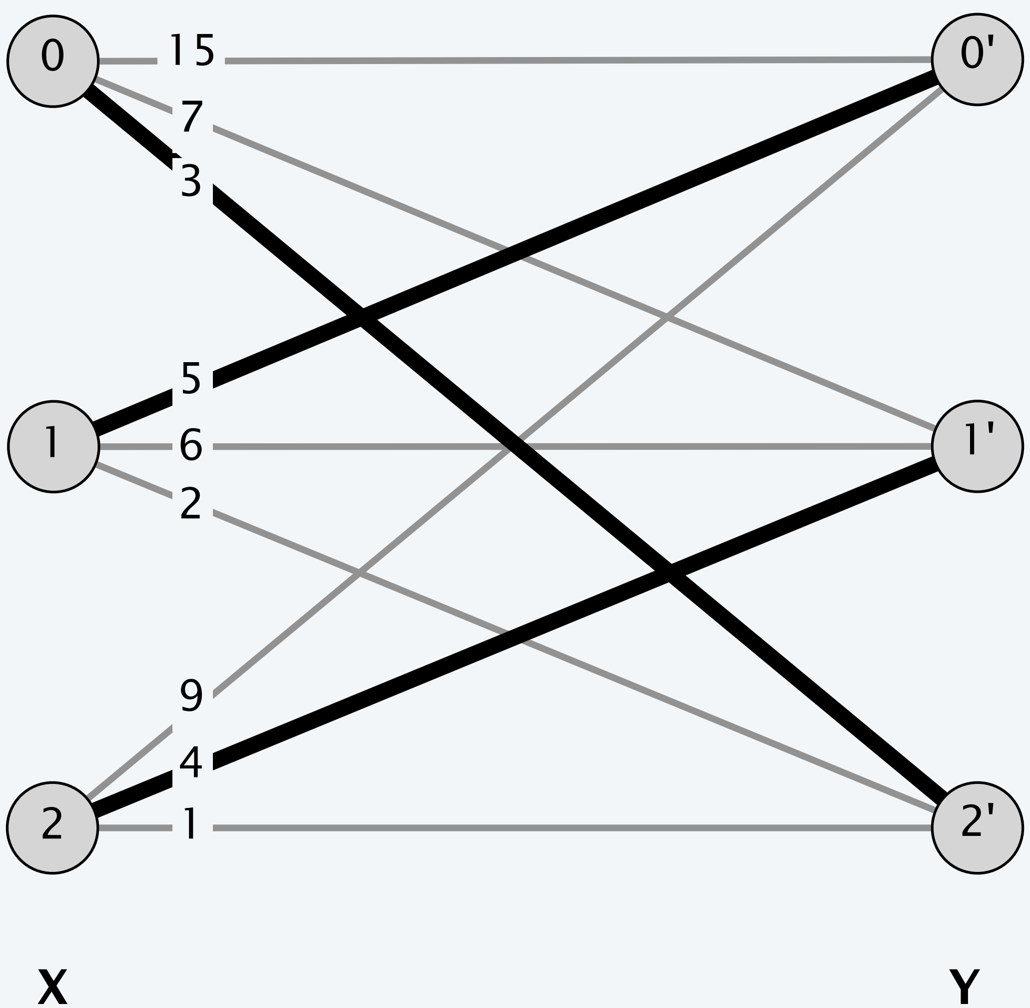
Goal. Find a perfect matching of min weight.



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' }

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:

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

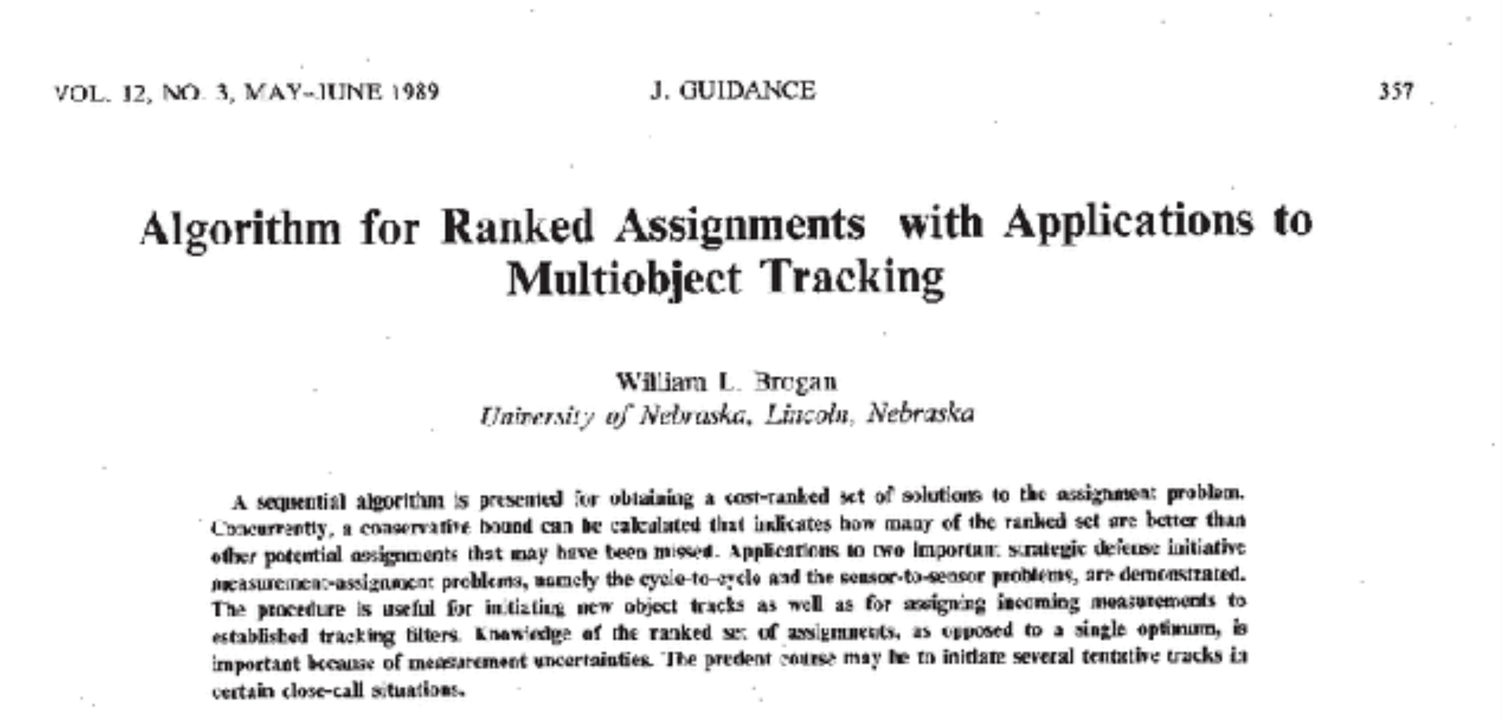
Title	Course #	Professor	Day/Time	Location
1980s, The	WRI 168	Scott, Andrea	M/W 1:30pm-2:50pm	Hargadon G002
America and the Melting Pot	WRI 157	Skinazi, Karen	T/TH 8:30am-9:50am	Butler 026
America and the Melting Pot	WRI 158	Skinazi, Karen	T/TH 11:00am-12:20pm	Hargadon G004
American Mysticism	WRI 191	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

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.



VOL. 12, NO. 3, MAY-JUNE 1989 J. GUIDANCE 357

Algorithm for Ranked Assignments with Applications to Multiobject Tracking

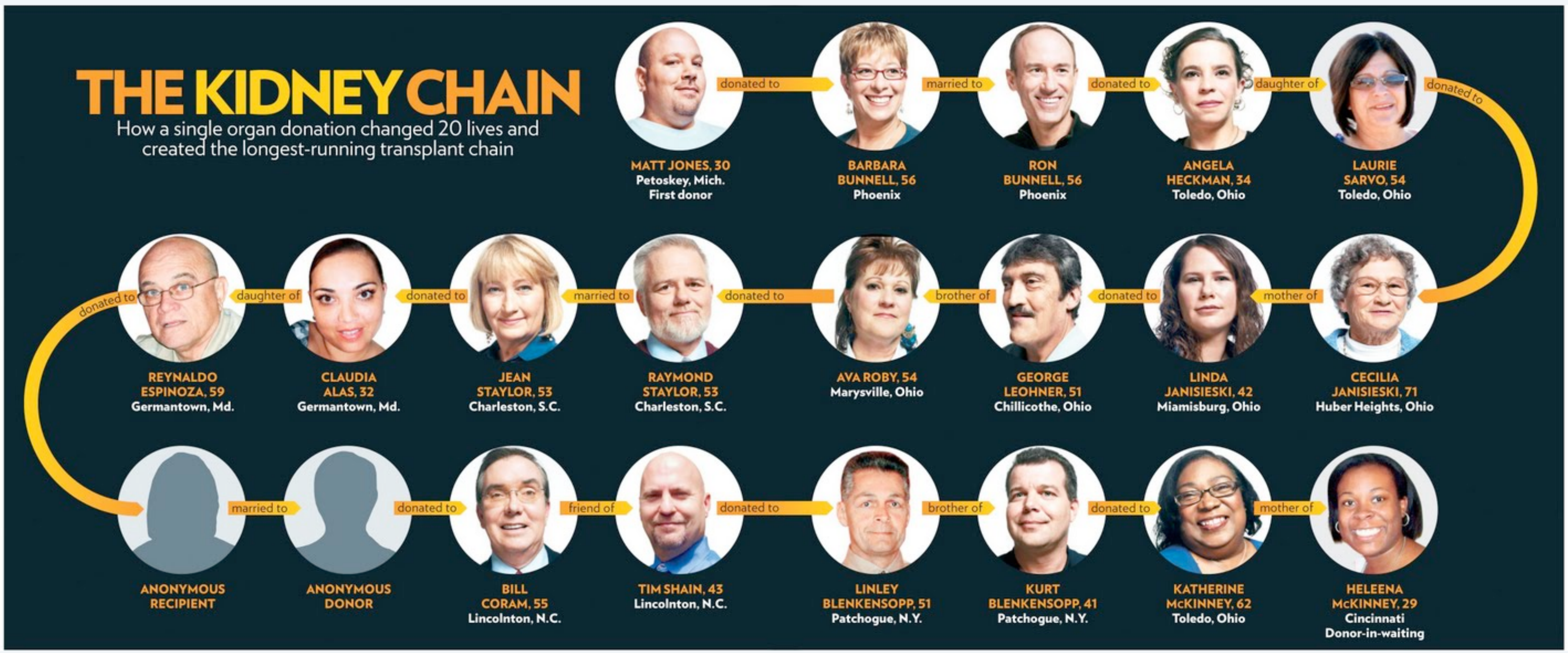
William L. Brogan
University of Nebraska, Lincoln, Nebraska

A sequential algorithm is presented for obtaining a cost-ranked set of solutions to the assignment problem. Concurrently, a conservative bound can be calculated that indicates how many of the ranked set are better than other potential assignments that may have been missed. Applications to two important strategic defense initiative measurement-assignment problems, namely the cycle-to-cycle and the sensor-to-sensor problems, are demonstrated. The procedure is useful for initiating new object tracks as well as for assigning incoming measurements to established tracking filters. Knowledge of the ranked set of assignments, as opposed to a single optimum, is important because of measurement uncertainties. The prudent course may be to initiate several tentative tracks in certain close-call situations.

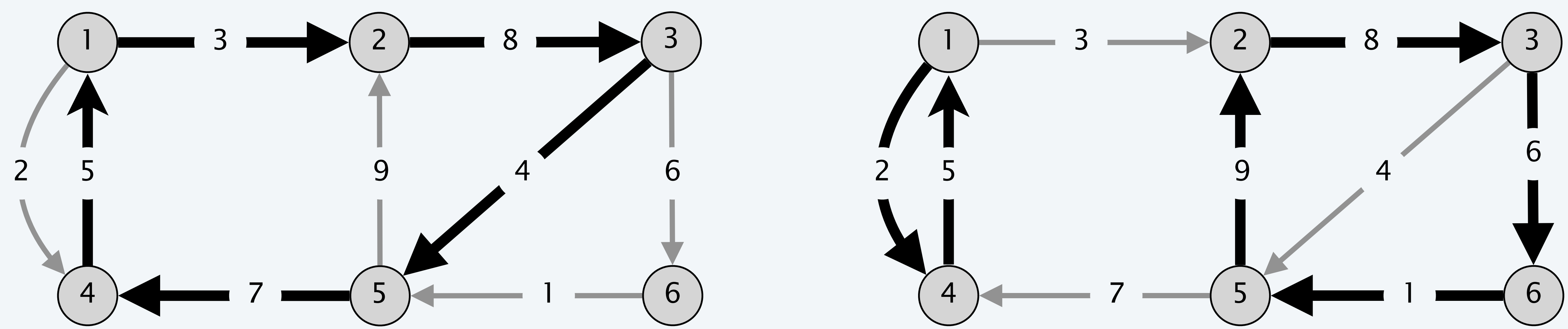
Kidney exchange

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).



Kidney exchange



weight = 3 + 5 + 7 + 8 + 4 = 27

weight = 2 + 5 + 8 + 6 + 1 + 9 = 31

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.
- 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.

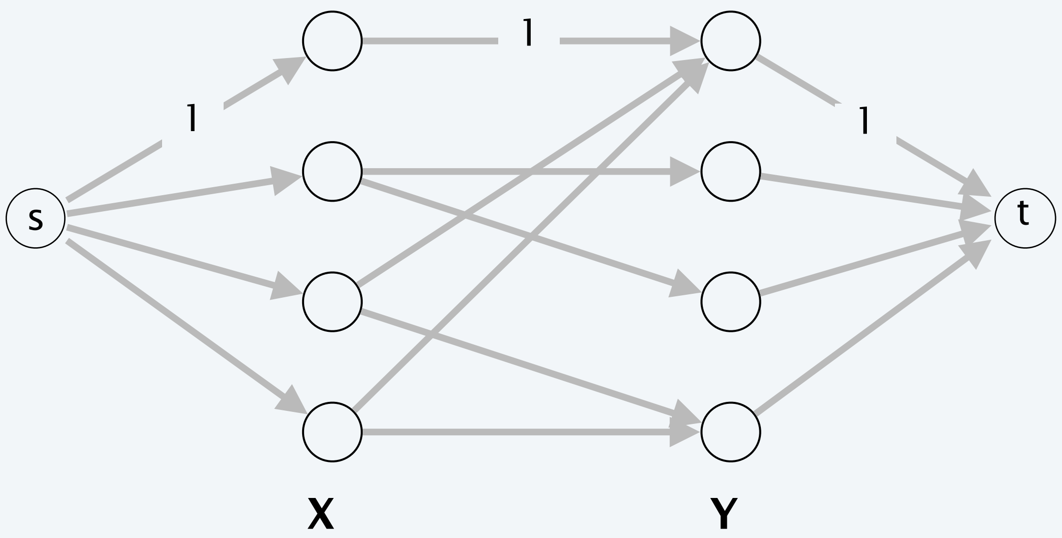
Bipartite matching

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.

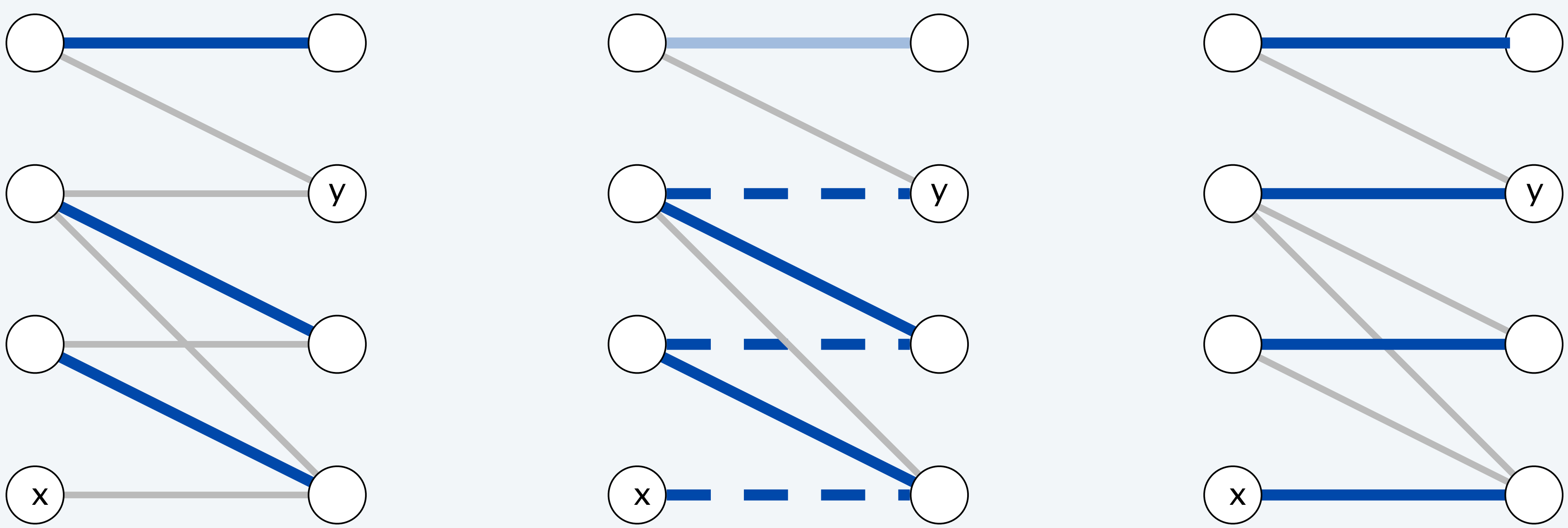
Alternating path

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 $M' = M \oplus P$.

symmetric difference



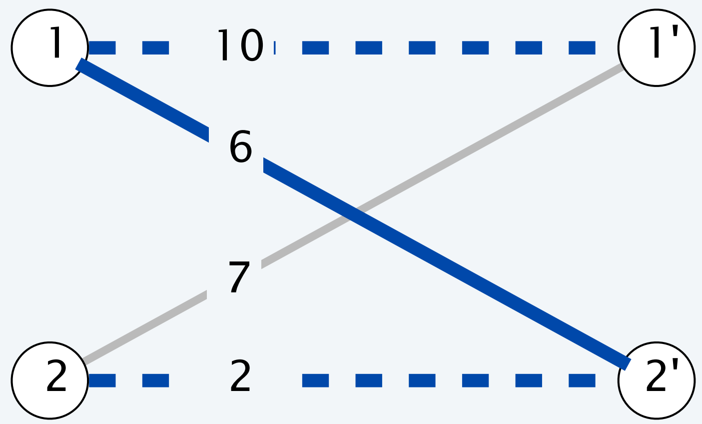
matching M

alternating path P

matching M′

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 → 2' → 1 → 1'

cost(P) = 2 – 6 + 10 = 6

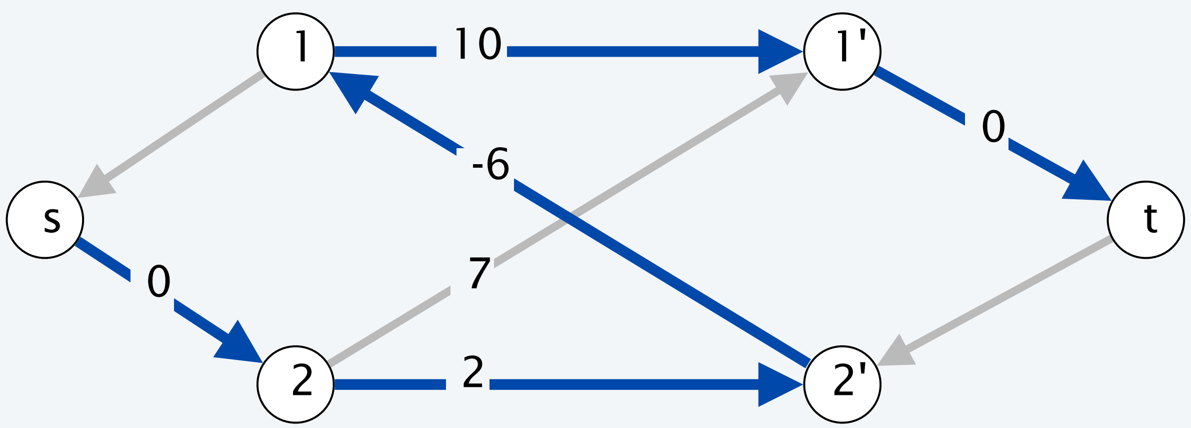
Shortest alternating path. Alternating path from any unmatched node $x \in X$ to any unmatched node $y \in Y$ with smallest cost.

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 minimum cost $s \rightsquigarrow t$ path in G_M.



Concern. Edge costs can be negative.

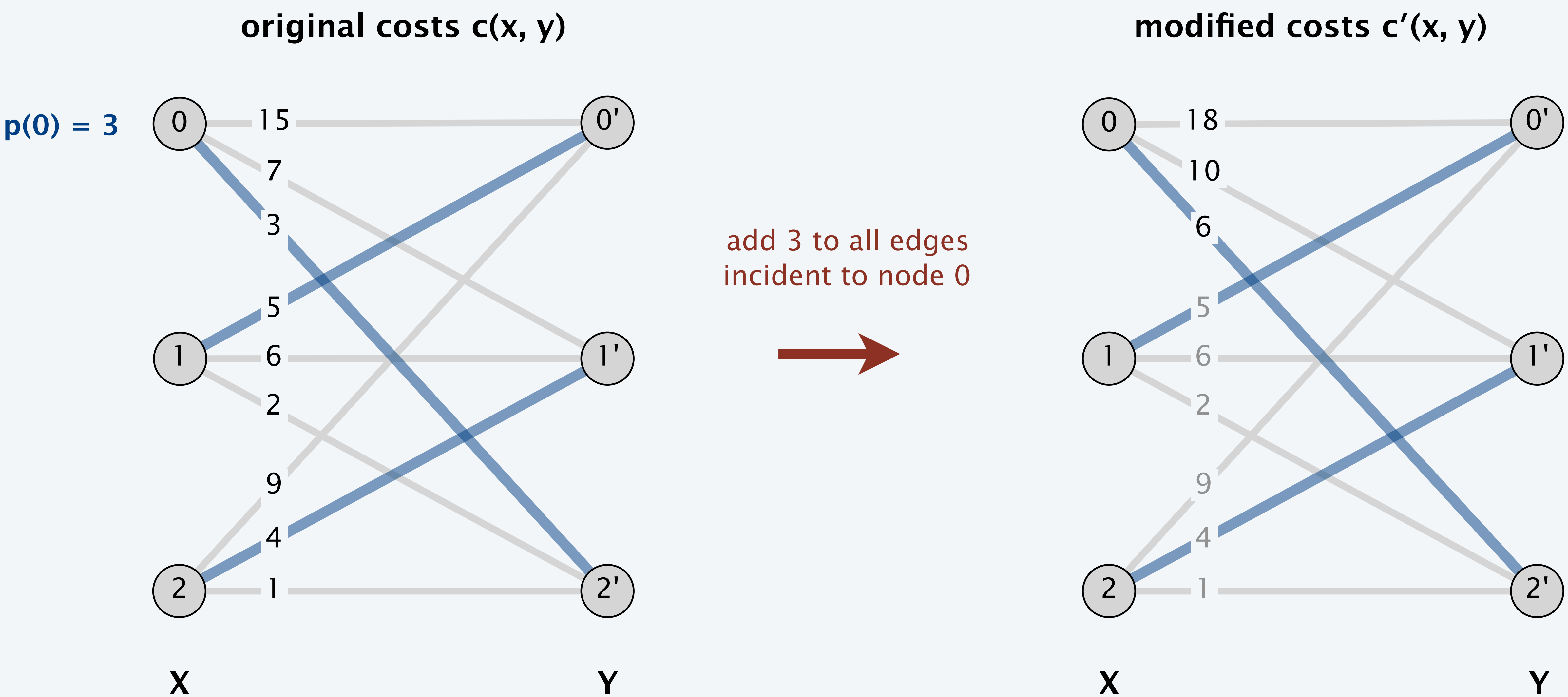
Fact. If always choose shortest alternating path, then G_M contains no negative cycles $\Rightarrow$ can compute using Bellman-Ford.

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).

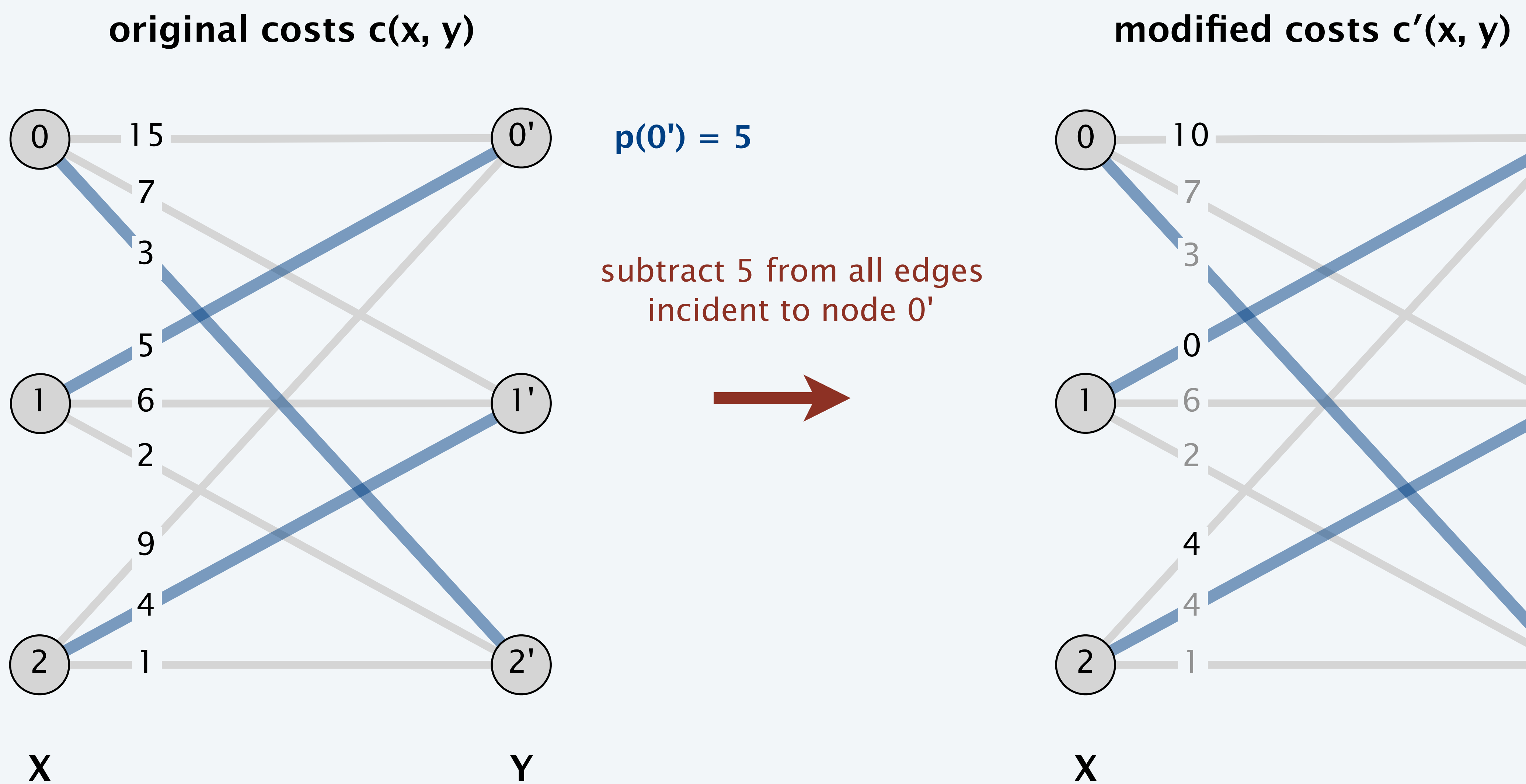
Pf. Every perfect matching uses exactly one edge incident to node x. ▪



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).

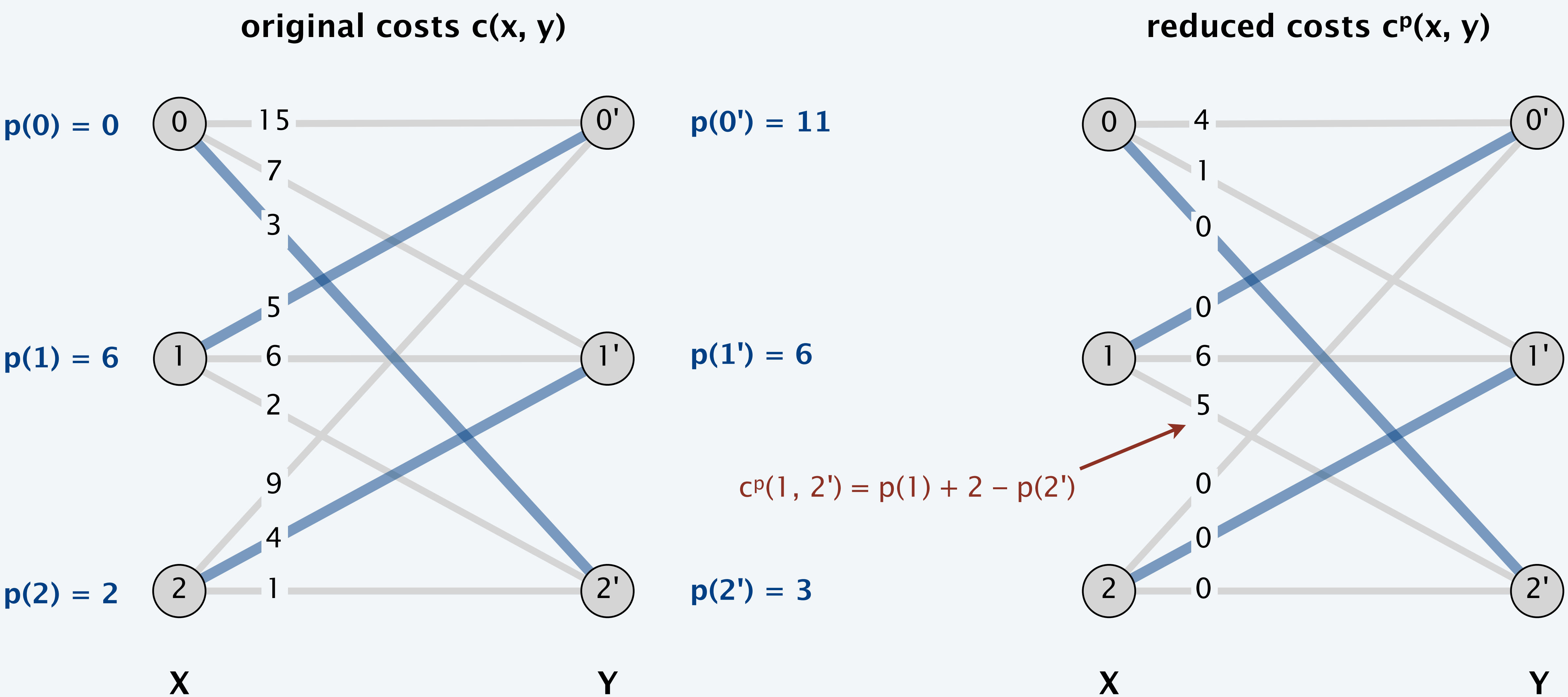
Pf. Every perfect matching uses exactly one edge incident to node y. ▪



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.



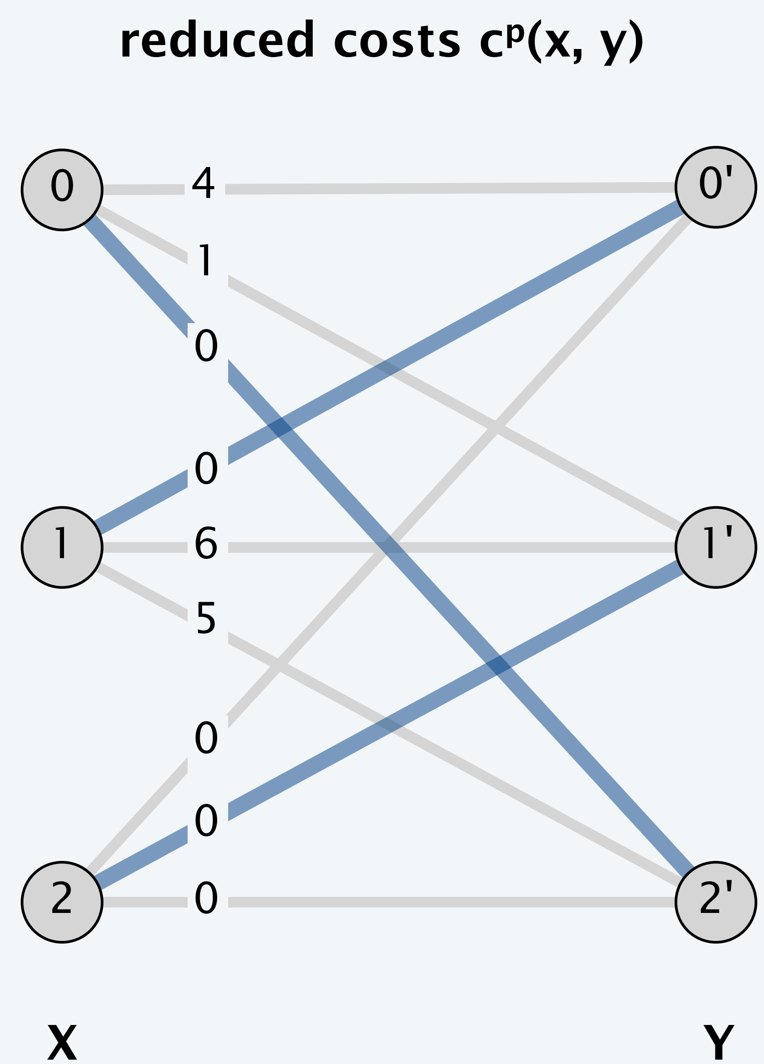
Compatible prices

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)

$M \leftarrow \varnothing$.

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.

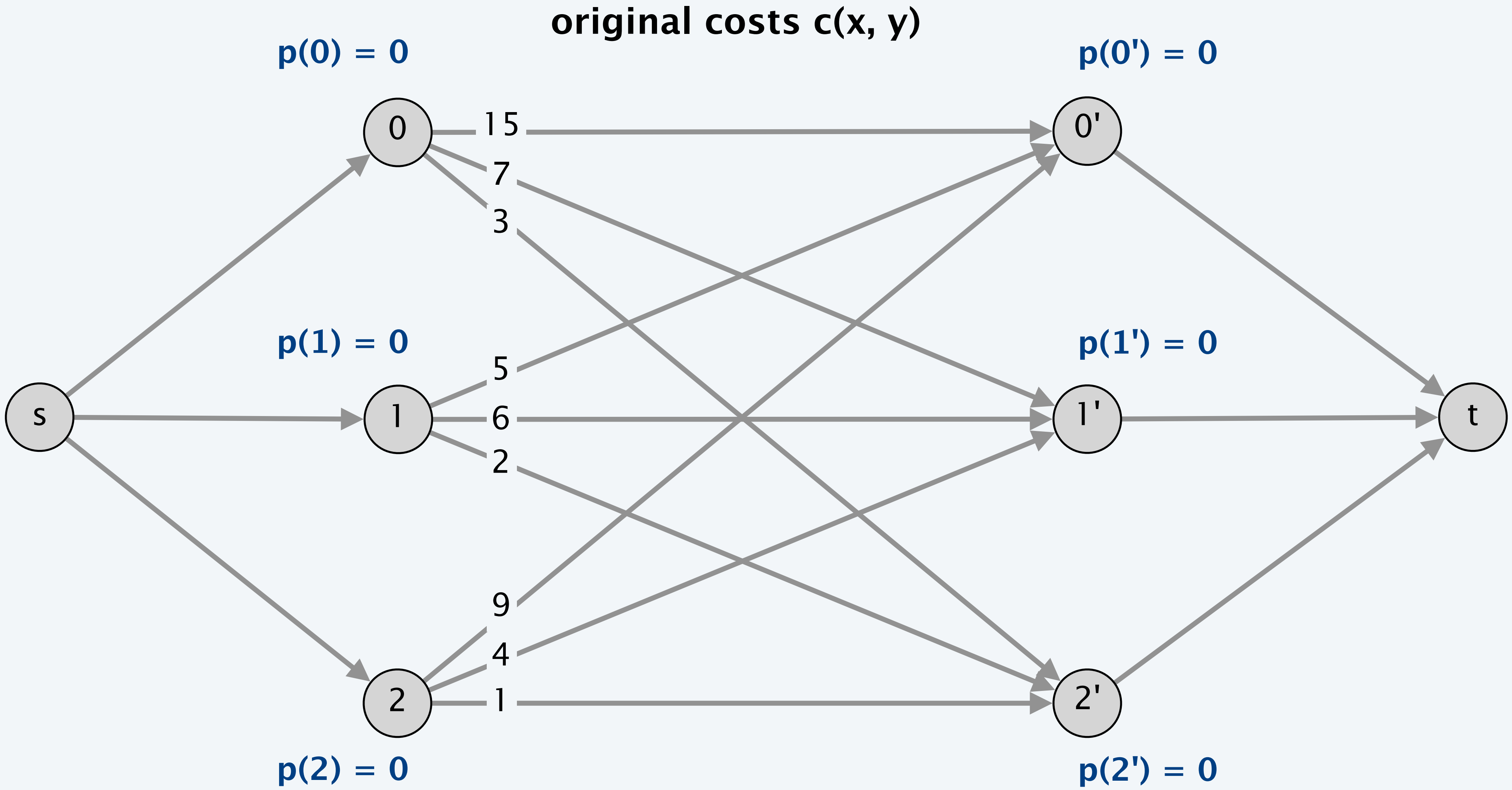
FOREACH $v \in X \cup Y : p(v) \leftarrow p(v) + d(v)$.

RETURN M.

Successive shortest path algorithm

Initialization.

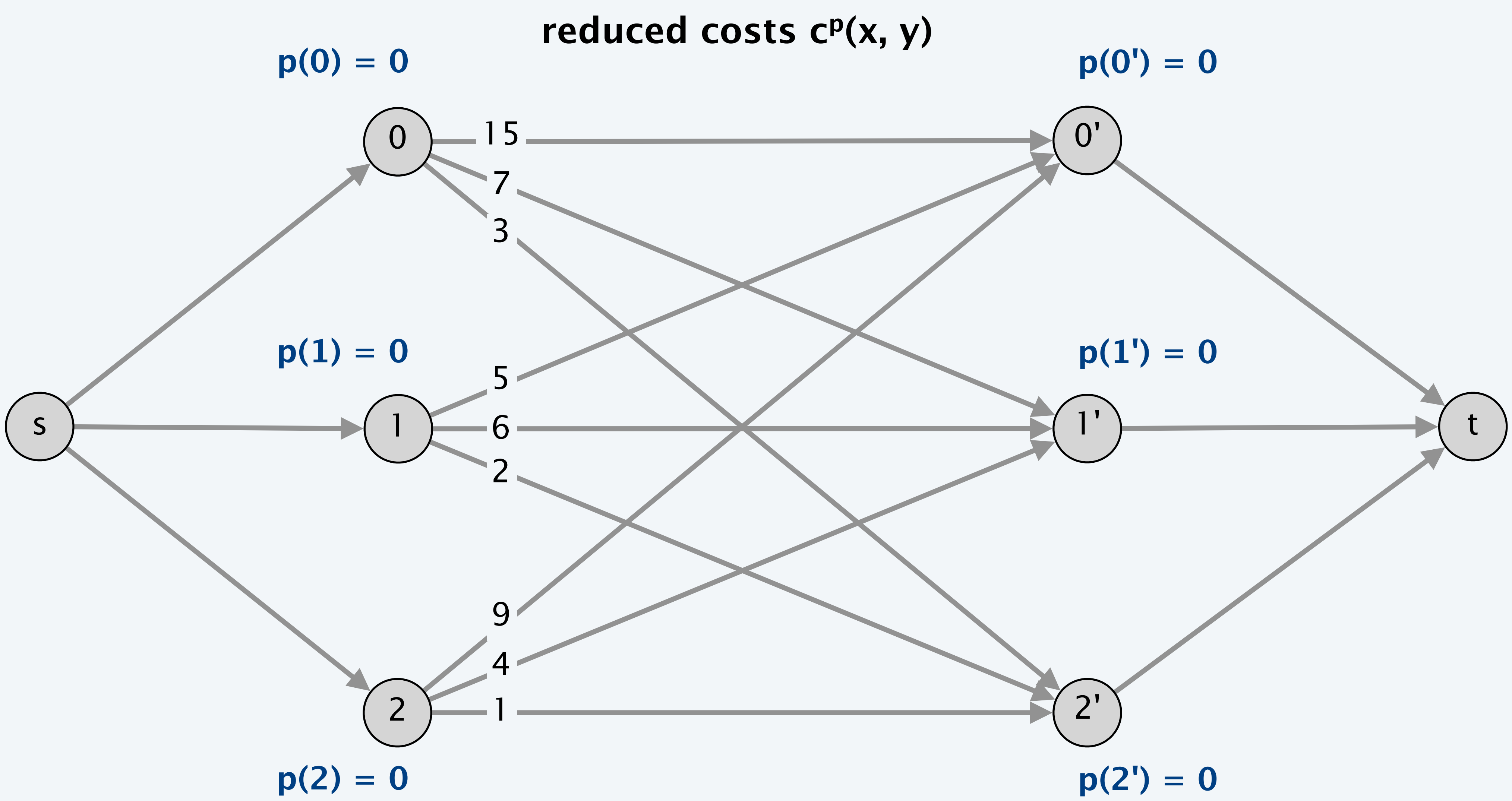
- $M = \varnothing$.
- For each $v \in X \cup Y : p(v) \leftarrow 0$.



Successive shortest path algorithm

Initialization.

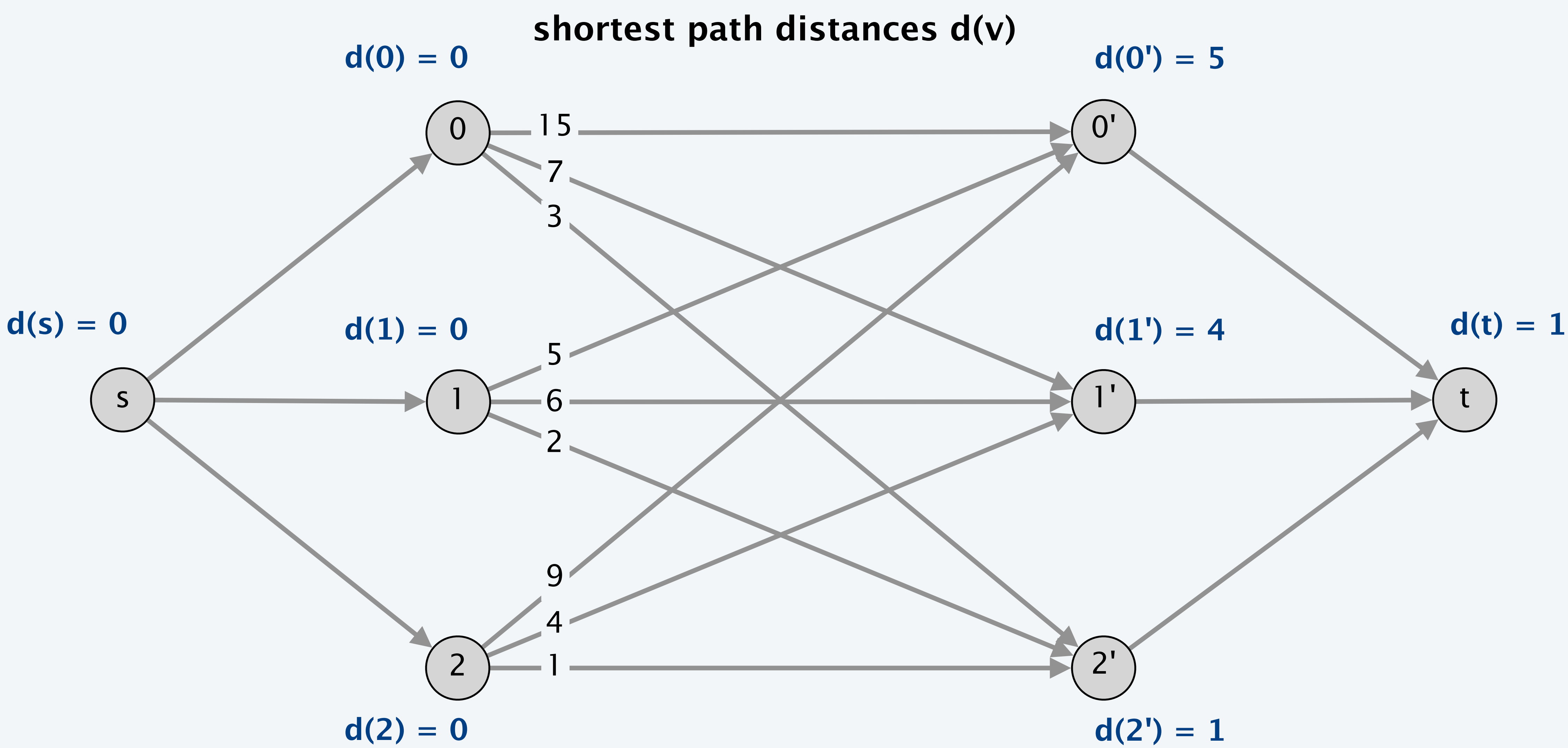
- $M = \varnothing$.
- For each $v \in X \cup Y : p(v) \leftarrow 0$.



Successive shortest path algorithm

Step 1.

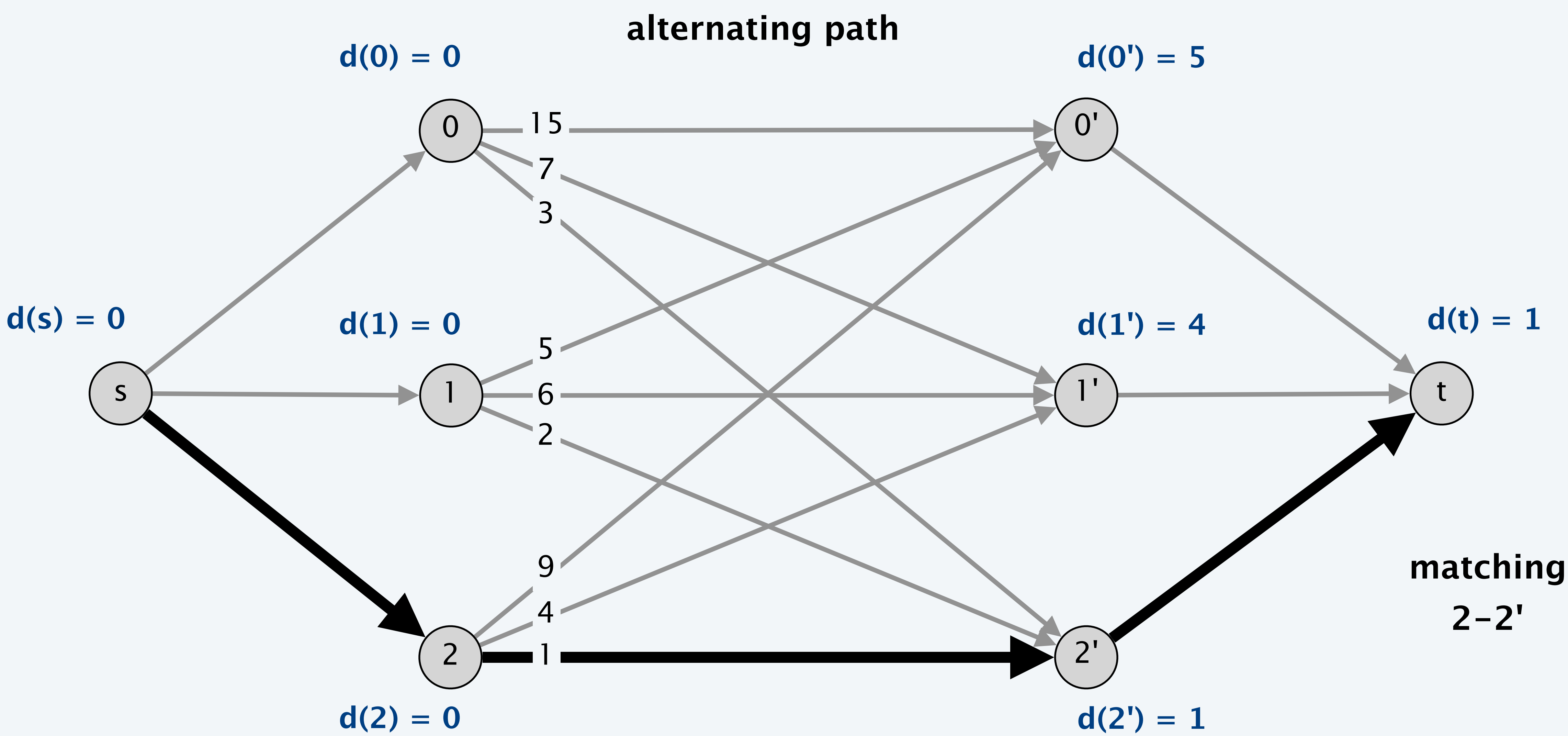
- Compute shortest path distances $d(v)$ from s to v using $c^p(x, y)$.
- Update matching M via shortest path from s to t.
- For each $v \in X \cup Y$: $p(v) \leftarrow p(v) + d(v)$.



Successive shortest path algorithm

Step 1.

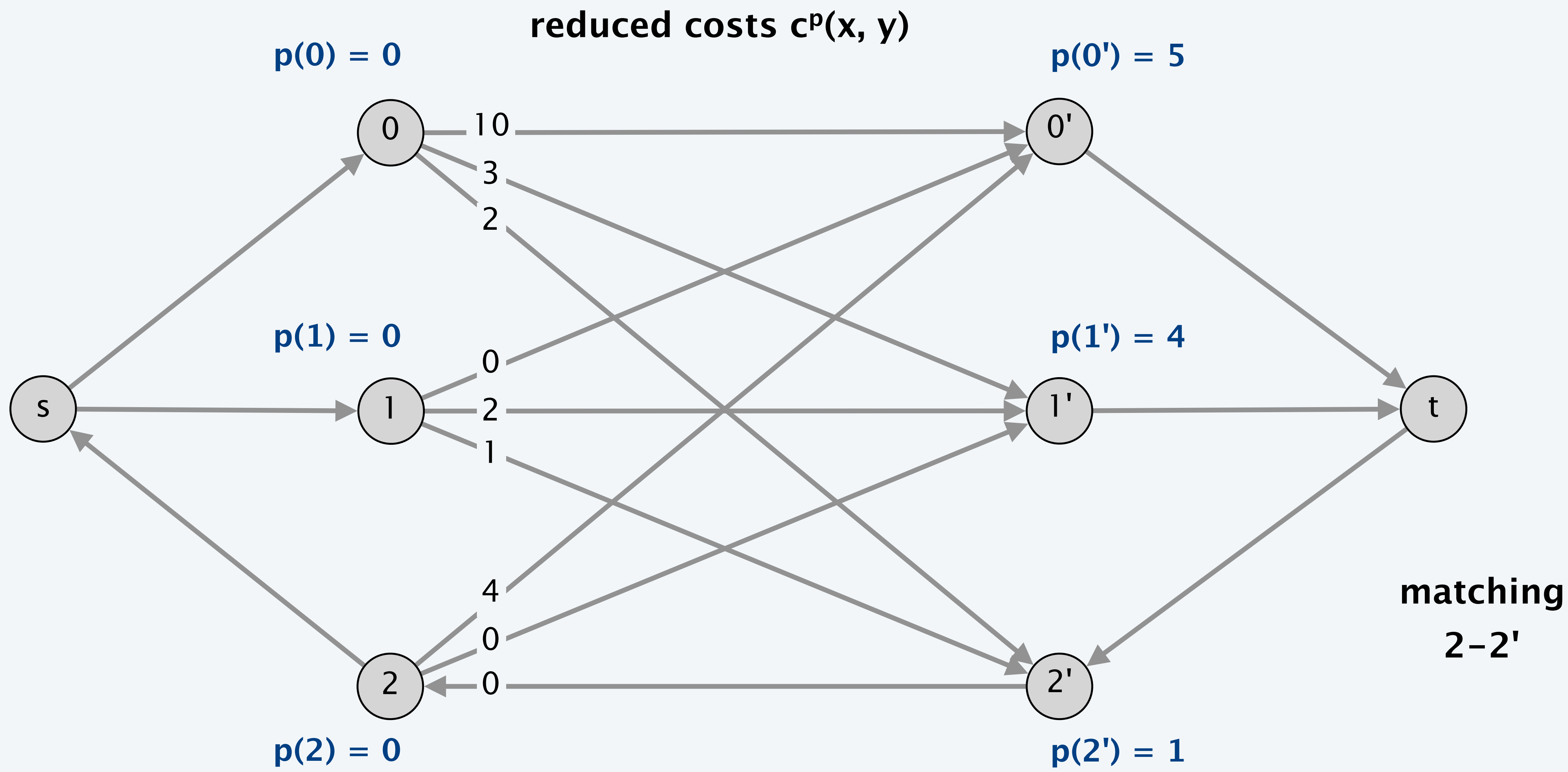
- Compute shortest path distances $d(v)$ from s to v using $c^p(x, y)$.
- Update matching M via shortest path from s to t.
- For each $v \in X \cup Y$: $p(v) \leftarrow p(v) + d(v)$.



Successive shortest path algorithm

Step 1.

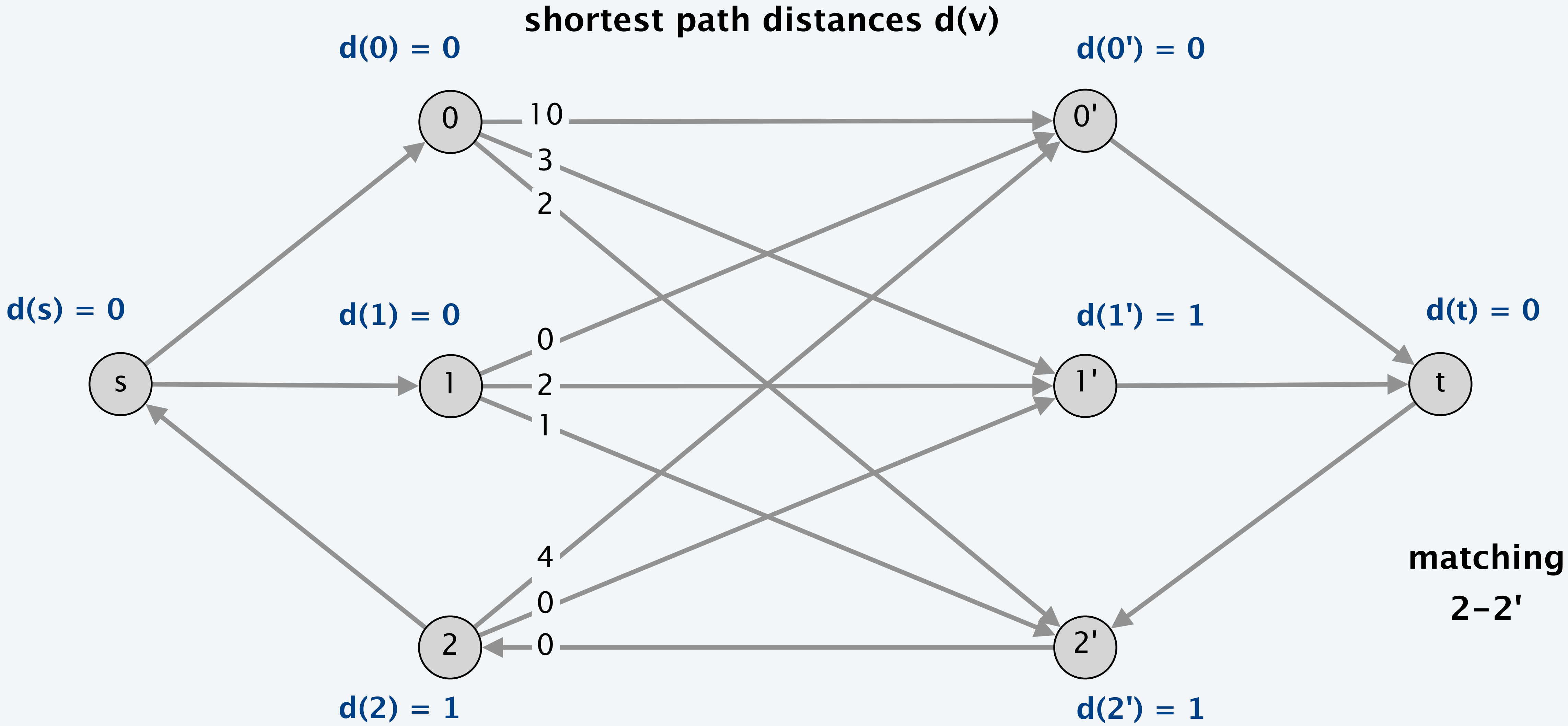
- Compute shortest path distances $d(v)$ from s to v using $c^p(x, y)$.
- Update matching M via shortest path from s to t.
- For each $v \in X \cup Y$: $p(v) \leftarrow p(v) + d(v)$.



Successive shortest path algorithm

Step 2.

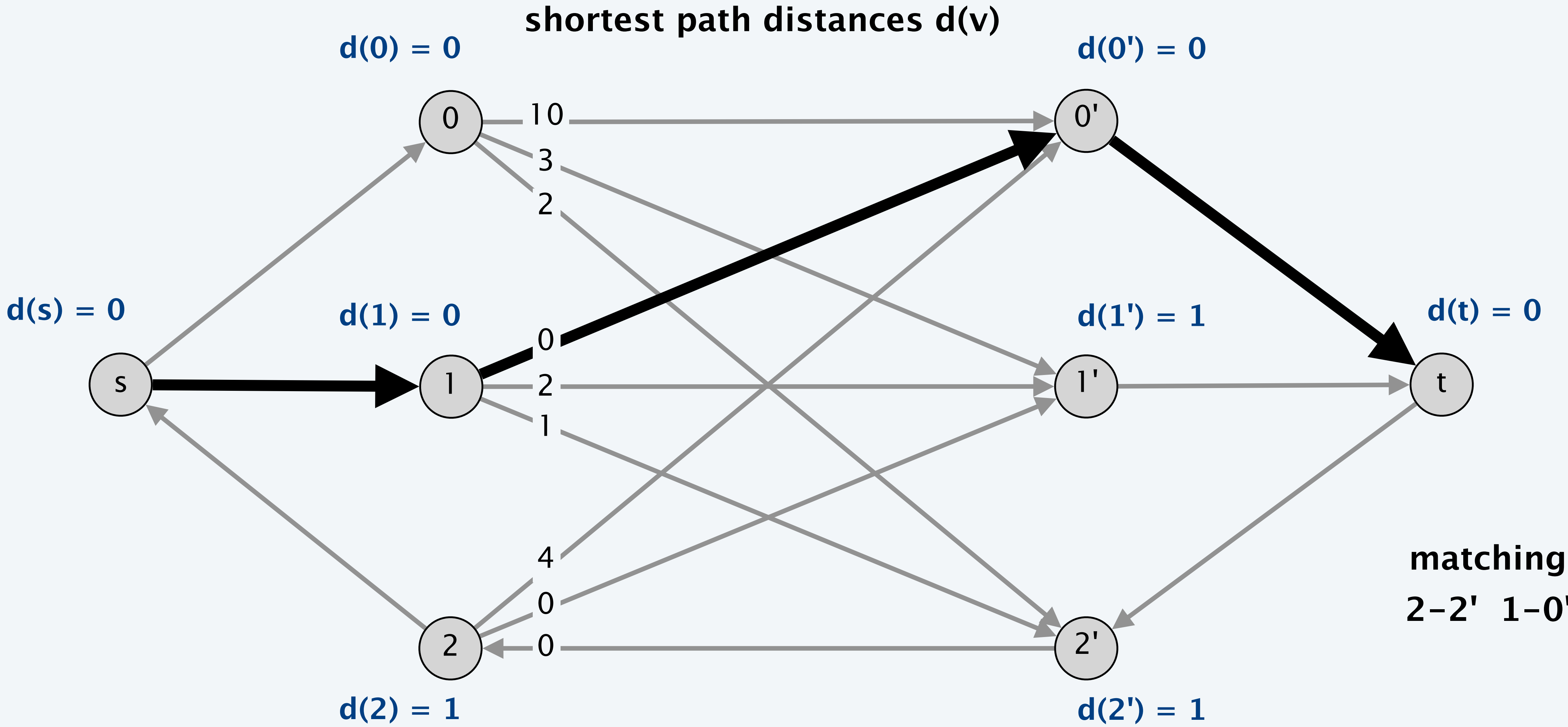
- Compute shortest path distances $d(v)$ from s to v using $c^p(x, y)$.
- Update matching M via shortest path from s to t.
- For each $v \in X \cup Y$: $p(v) \leftarrow p(v) + d(v)$.



Successive shortest path algorithm

Step 2.

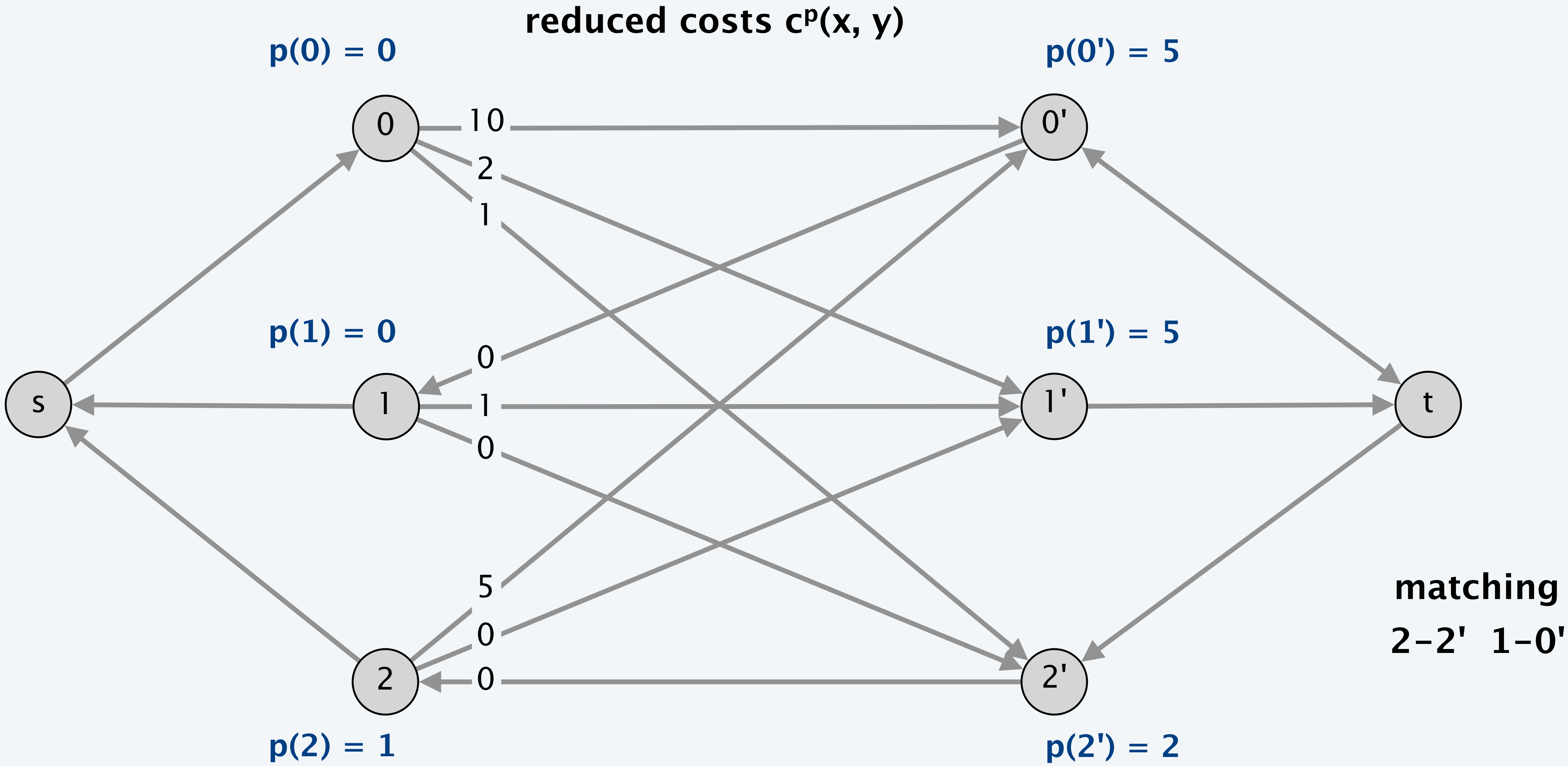
- Compute shortest path distances $d(v)$ from s to v using $c^p(x, y)$.
- Update matching M via shortest path from s to t.
- For each $v \in X \cup Y$: $p(v) \leftarrow p(v) + d(v)$.



Successive shortest path algorithm

Step 2.

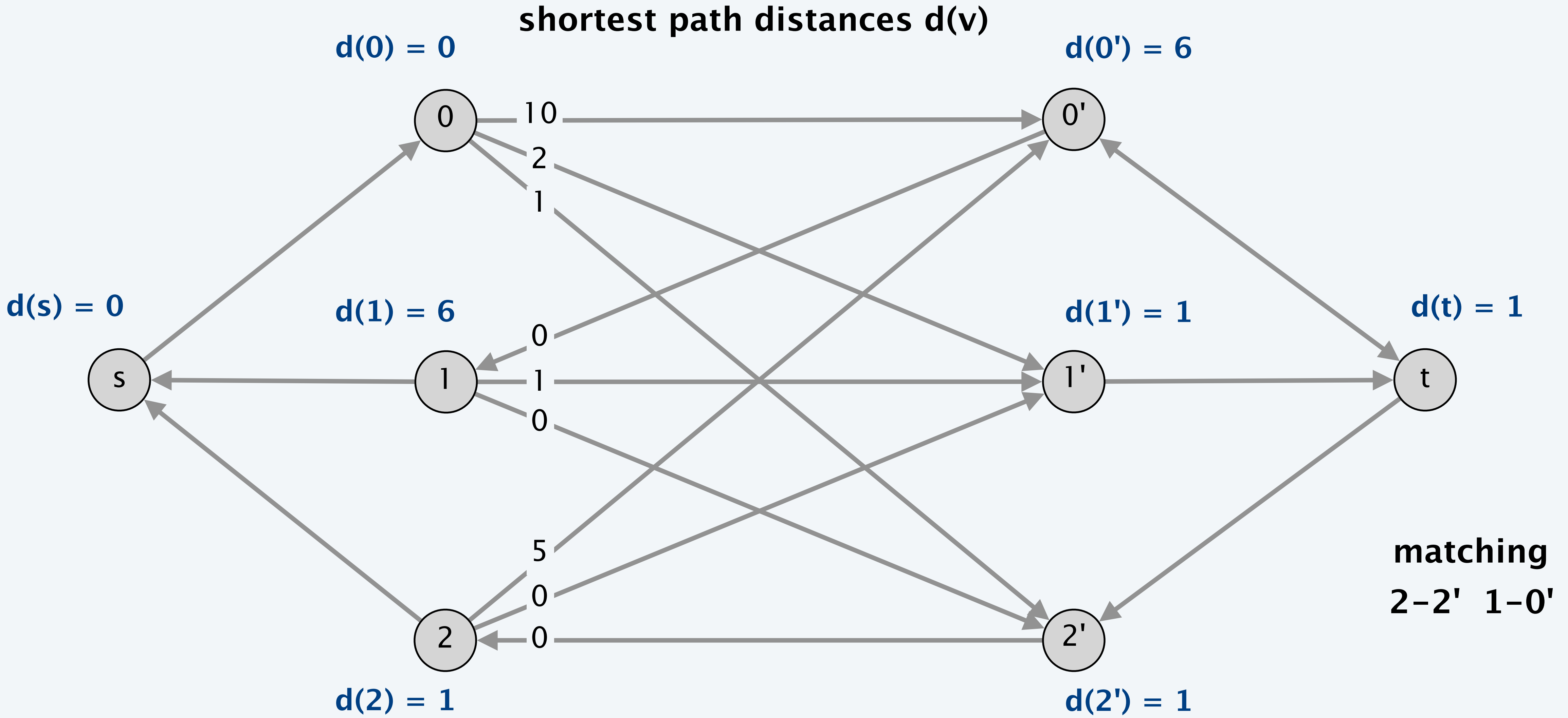
- Compute shortest path distances $d(v)$ from s to v using $c^p(x, y)$.
- Update matching M via shortest path from s to t.
- For each $v \in X \cup Y$: $p(v) \leftarrow p(v) + d(v)$.



Successive shortest path algorithm

Step 3.

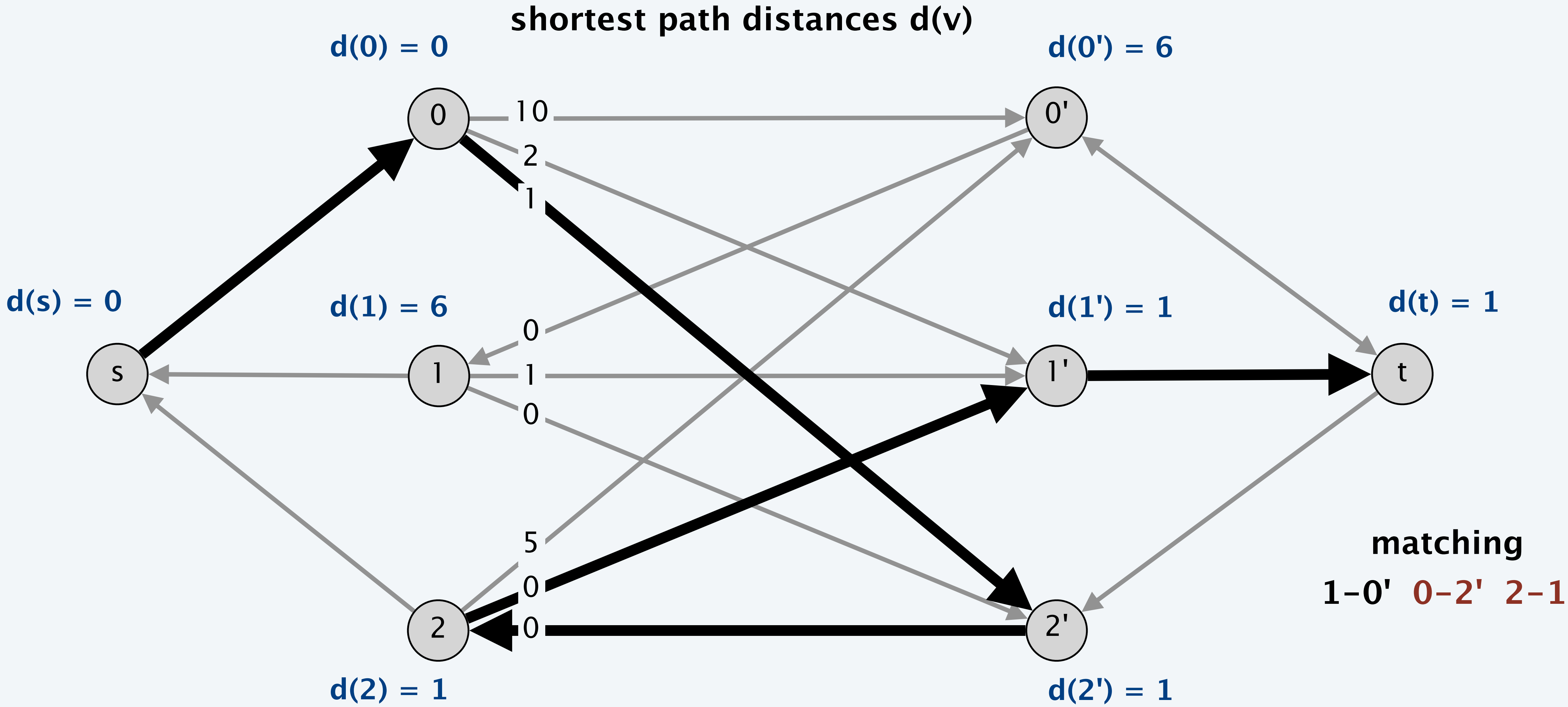
- Compute shortest path distances $d(v)$ from s to v using $c^p(x, y)$.
- Update matching M via shortest path from s to t.
- For each $v \in X \cup Y$: $p(v) \leftarrow p(v) + d(v)$.



Successive shortest path algorithm

Step 3.

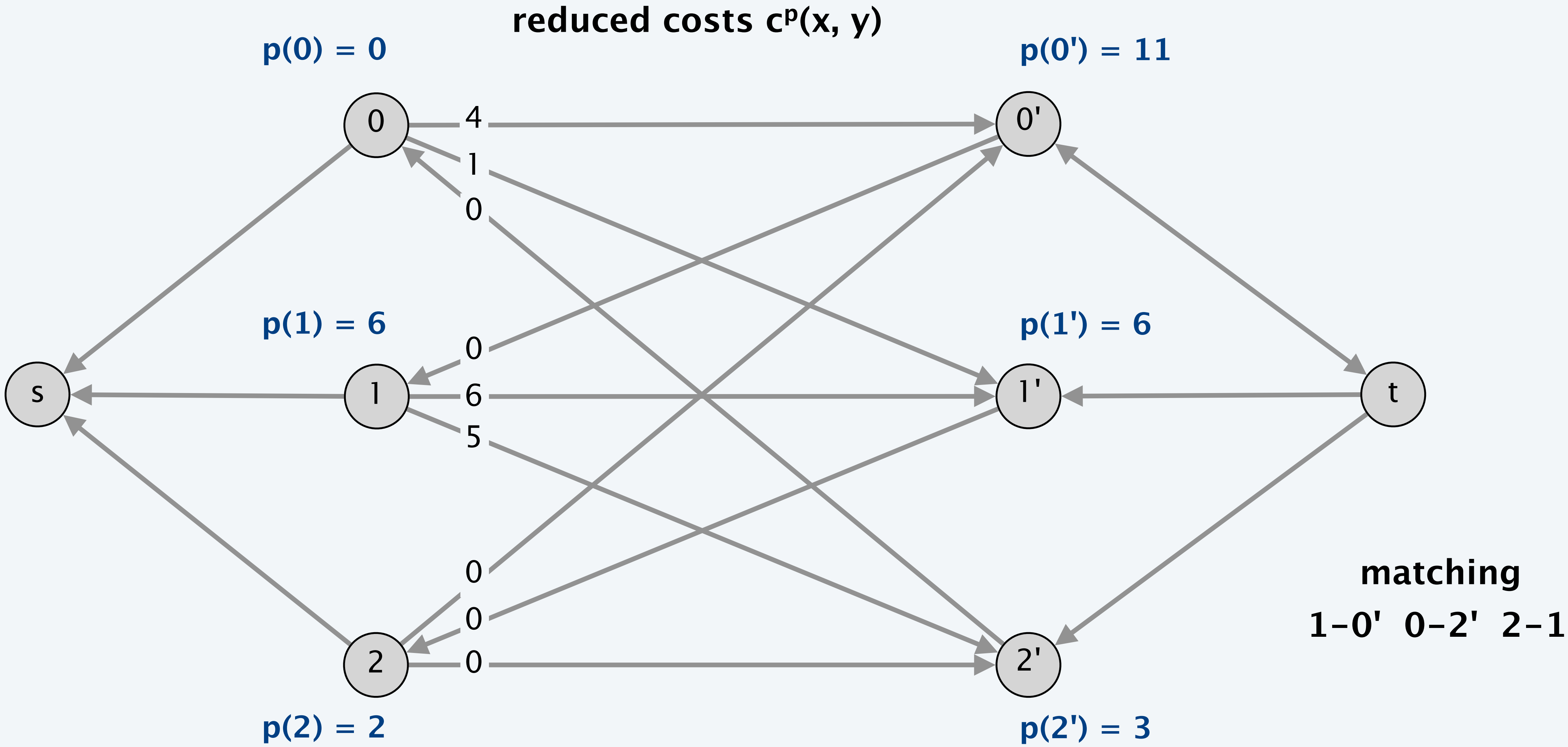
- Compute shortest path distances $d(v)$ from s to v using $c^p(x, y)$.
- Update matching M via shortest path from s to t.
- For each $v \in X \cup Y$: $p(v) \leftarrow p(v) + d(v)$.



Successive shortest path algorithm

Step 3.

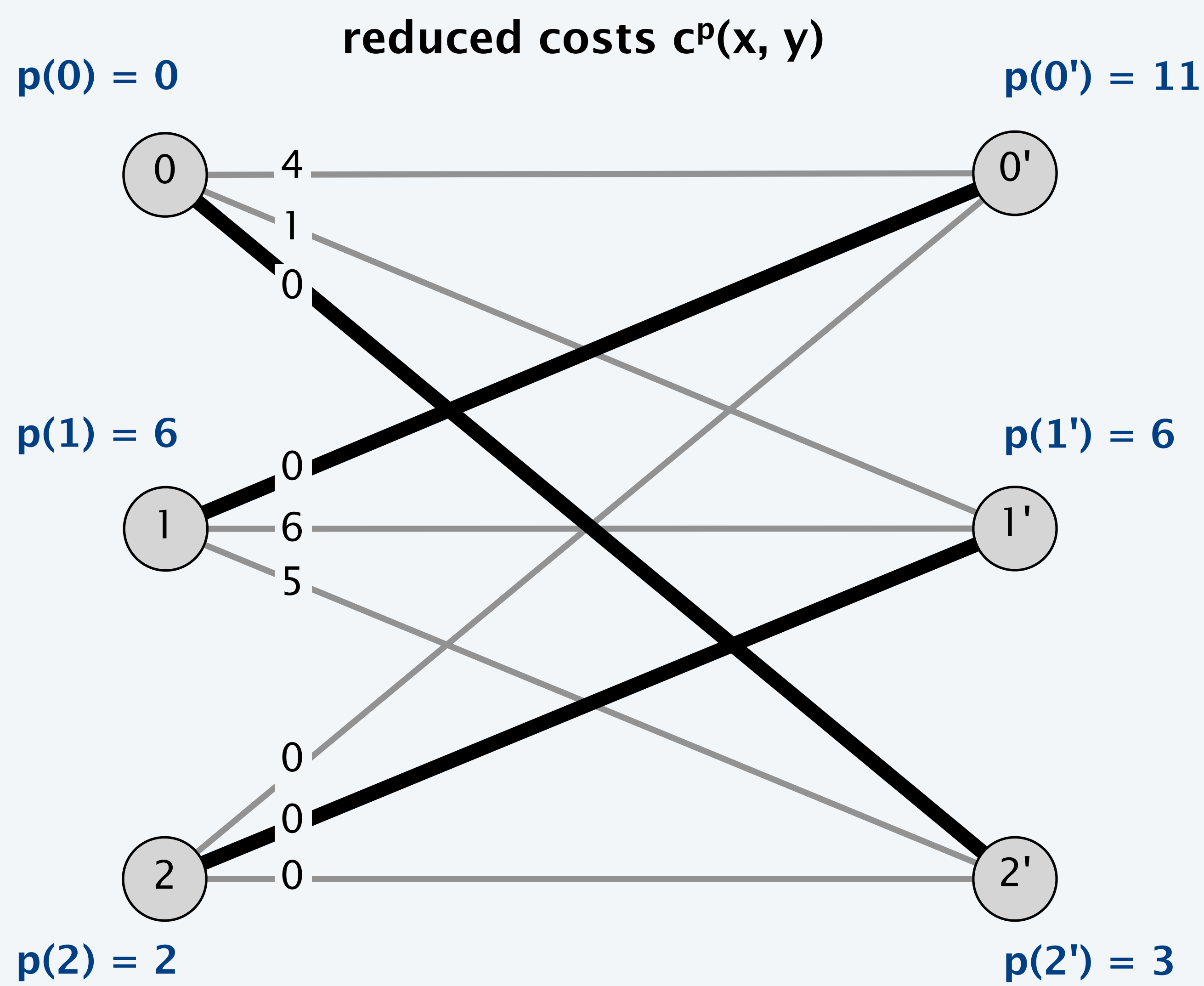
- Compute shortest path distances $d(v)$ from s to v using $c^p(x, y)$.
- Update matching M via shortest path from s to t.
- For each $v \in X \cup Y$: $p(v) \leftarrow p(v) + d(v)$.



Successive shortest path algorithm

Termination.

- M is a perfect matching.
- Prices p are compatible with M.



matching
1–0' 0–2' 2–1'

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

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 $(x, y) \in M$.

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 $(x, y) \in M$.

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. ▪

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 $O(n^2 + m\,n \log n)$ time using Fibonacci heaps.

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

SIAM J. COMPUT.
Vol. 18, No. 5, pp. 1013–1036, October 1989

FASTER SCALING ALGORITHMS FOR NETWORK PROBLEMS*

HAROLD N. GABOW† AND ROBERT E. TARJAN‡

Abstract. This paper presents algorithms for the assignment problem, the transportation problem, and the minimum-cost flow problem of operations research. The algorithms find a minimum-cost solution, yet run in time close to the best-known bounds for the corresponding problems without costs. For example, the assignment problem (equivalently, minimum-cost matching in a bipartite graph) can be solved in $O(\sqrt{n}m \log(nN))$ time, where n, m, and N denote the number of vertices, number of edges, and largest magnitude of a cost; costs are assumed to be integral. The algorithms work by scaling. As in the work of Goldberg and Tarjan, in each scaled problem an approximate optimum solution is found, rather than an exact optimum.

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.

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!

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 PROBLEM[1]

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

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 cujuscunque.

(Ex. ill. *C. G. J. Jacobi* manuscriptis posthumis in medium protulit*) *C. W. Borchardt.*)

1.

Investigatio ad solvendum problema inaequalitatum reducitur.

Systema aequationum differentialium vulgarium est *non canonicum***), si aequationes altissima variabilium dependentium differentialia tali modo continent, ut horum valores ex iis petere non liceat. Id quod fit, quoties aequationes nonnullae altissimis illis differentialibus carentes in systemate proposito vel ipsae inveniuntur vel eliminatione ex eo obtinentur. *Eo casu numerus Constantium Arbitrariarum, quas integratio completa inducit, sive ordo systematis semper minor est summa altissimorum ordinum, ad quos differentialia singularum variabilium in aequationibus differentialibus propositis ascendunt.* Qui ordo systematis cognoscitur, si per differentiationes et eliminationes contingit systema propositum redigere in aliud forma canonica gaudens eique aequivalens, ita ut de systemate canonico etiam ad propositum reditus pateat. Nam summa altissimorum ordinum, ad quos in systemate canonico differentialia singularum variabilium dependentium ascendunt, etiam systematis propositi non canonici ordo erit. Ad quem ordinem investigandum non tamen opus est ea ad formam canonicam reductione, sed res per considerationes sequentes absolvi potest.

Ponamus inter variabilem independentem t atque n variabiles dependentes $x_1, x_2, \ldots x_n$ haberi n aequationes differentiales:

$$(1.) \quad u_1 = 0, \quad u_2 = 0, \quad \ldots \quad u_n = 0,$$

sitque

$$h_k^{(i)}$$

altissimus ordo, ad quem in aequatione $u_i = 0$ differentialia variabilis x_k ascen-

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

History

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

Problema.

Disponantur nn quantitates $h_k^{(i)}$ quaecunque in schema Quadrati, ita ut habeantur n series horizontales et n series verticales quarum quaeque est n terminorum. Ex illis quantitatibus eligantur n transversales i. e. in seriebus horizontalibus simul atque verticalibus diversis positae, quod fieri potest 1.2...n modis; ex omnibus illis modis quaerendus est is qui summam n numerorum electorum suppeditet maximam.

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

$$\begin{array}{cccc} h_1' & h_2' & \dots & h_n' \\ h_1'' & h_2'' & \dots & h_n'' \\ \dots & \dots & \dots & \dots \\ h_1^{(n)} & h_2^{(n)} & \dots & h_n^{(n)} \end{array}$$

earum systema appellabo *schema propositum;* omne schema inde ortum addendo singulis ejusdem seriei horizontalis terminis eandem quantitatem apellabo *schema derivatum*. Sit

$$l^{(i)}$$

quantitas addenda terminis i^{tae} seriei horizontalis, quo facto singula 1.2...n aggregata transversalia, inter quae maximum eligendum est, eadem augebuntur quantitate

$$l'+l''+\dots+l^{(n)} = L,$$

quippe ad singula aggregata formanda e quaque serie horizontali unus eligendus est terminus. Qua de re si statuitur

$$h_k^{(i)}+l^{(i)} = p_k^{(i)}$$

atque aggregatum transversale maximum e terminis $h_k^{(i)}$ formatum

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

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

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

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 1.2...n ways; and among these ways, to research one that gives the maximum of the sum of the n chosen numbers.

$$\begin{array}{cccc} h_1' & h_2' & \dots & h_n' \\ h_1'' & h_2'' & \dots & h_n'' \\ \dots & \dots & \dots & \dots \\ h_1^{(n)} & h_2^{(n)} & \dots & h_n^{(n)}, \end{array}$$

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 $1.2\dots n$ transversal sums among which we need to find a maximum is increased by the same quantity

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

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

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

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

7. Network Flow III

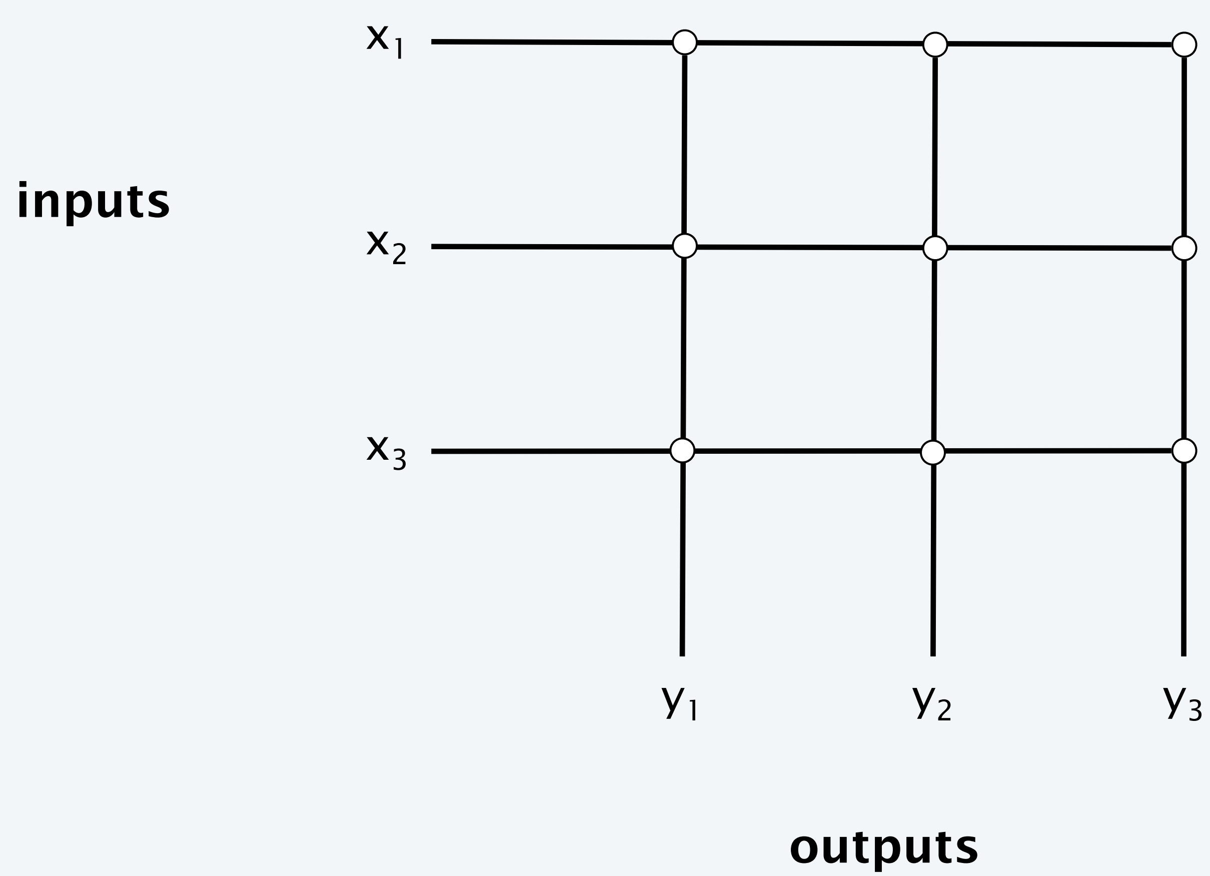
- *assignment problem*
- *input-queued switching*

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.

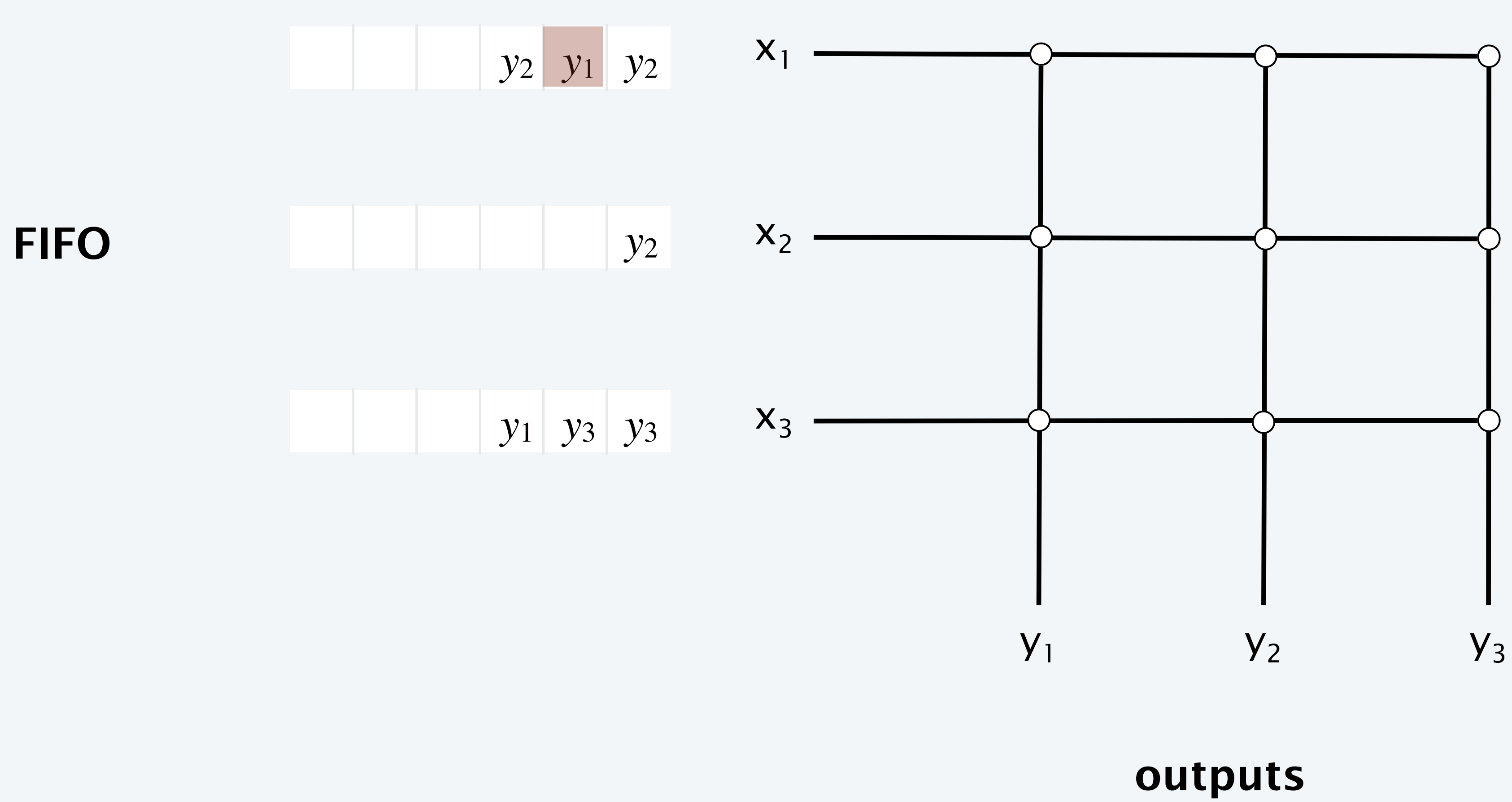
Application. High-bandwidth switches.



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.



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.

IEEE TRANSACTIONS ON COMMUNICATIONS, VOL. COM-35, NO. 12, DECEMBER 1987 1347

Input Versus Output Queueing on a Space-Division Packet Switch

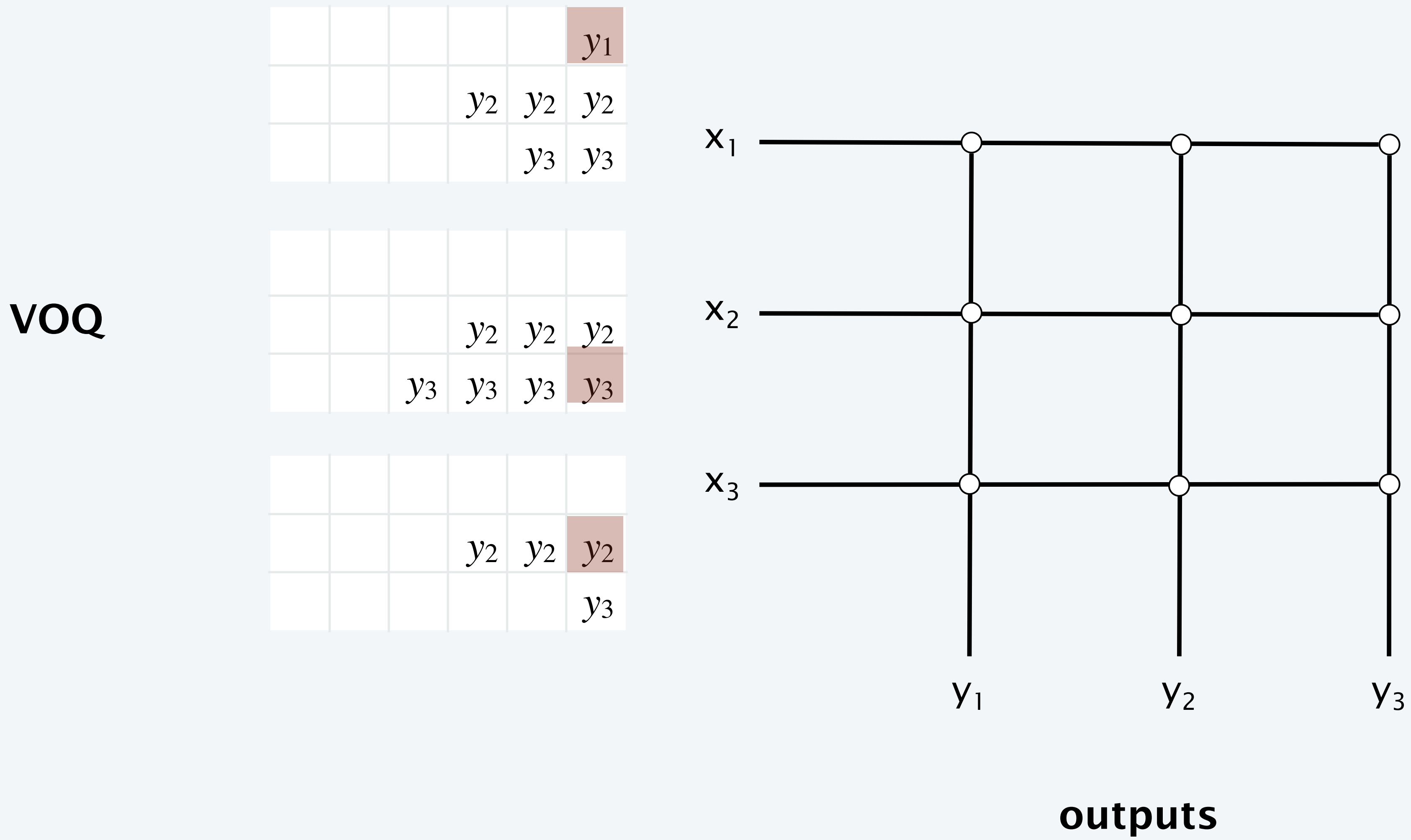
MARK J. KAROL, MEMBER, IEEE, MICHAEL G. HLUCHYJ, MEMBER, IEEE, AND SAMUEL P. MORGAN, FELLOW, IEEE

Abstract—Two simple models of queueing on an $N \times N$ space-division packet switch are examined. The switch operates synchronously with fixed-length packets; during each time slot, packets may arrive on any inputs addressed to any outputs. Because packet arrivals to the switch are unscheduled, more than one packet may arrive for the same output during the same time slot, making queueing unavoidable. Mean queue lengths are always greater for queueing on inputs than for queueing on outputs, and the output queues saturate only as the utilization approaches unity. Input queues, on the other hand, saturate at a utilization that depends on N, but is approximately $(2 - \sqrt{2}) = 0.586$ when N is large. If output trunk utilization is the primary consideration, it is possible to slightly increase utilization of the output trunks—up to $(1 - e^{-1}) = 0.632$ as $N \to \infty$—by dropping interfering packets at the end of each time slot, rather than storing them in the input queues. This improvement is possible, however, only when the utilization of the input trunks exceeds a second critical threshold—approximately $\ln(1 + \sqrt{2}) = 0.881$ for large N.

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.



Input-queued switching

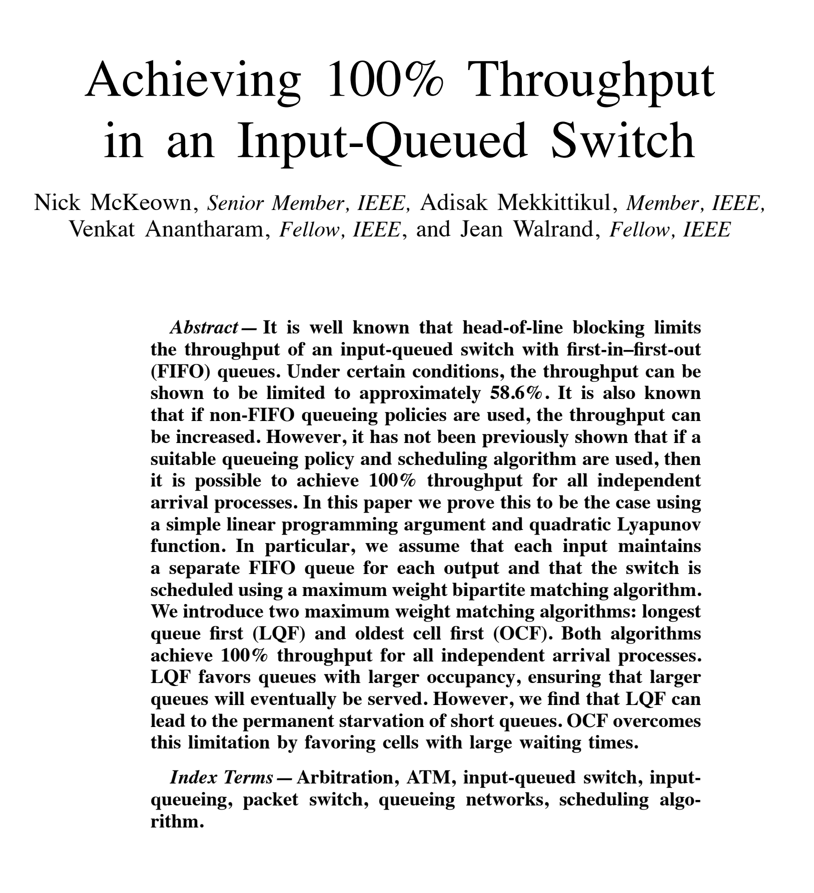
Maximum-weight matching. Find a min cost perfect matching between inputs x and outputs y, where $c(x, y)$ equals:

- [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).

Practice.

- Assignment problem too slow in practice.
- Difficult to implement in hardware.
- Provides theoretical framework: use maximal (weighted) matching.



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 shown to be limited to approximately 58.6%. It is also known that if non-FIFO queueing policies are used, the throughput can be increased. However, it has not been previously shown that if a suitable queueing policy and scheduling algorithm are used, then it is possible to achieve 100% throughput for all independent arrival processes. In this paper we prove this to be the case using a simple linear programming argument and quadratic Lyapunov function. In particular, we assume that each input maintains a separate FIFO queue for each output and that the switch is scheduled using a maximum weight bipartite matching algorithm. We introduce two maximum weight matching algorithms: longest queue first (LQF) and oldest cell first (OCF). Both algorithms achieve 100% throughput for all independent arrival processes. LQF favors queues with larger occupancy, ensuring that larger queues will eventually be served. However, we find that LQF 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-queueing, packet switch, queueing networks, scheduling algorithm.**