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.

\[
\begin{align*}
0 &\quad 15 & 7 & 3 \\
1 &\quad 5 & 6 & 2 \\
2 &\quad 4 & 9 & 1 \\
\end{align*}
\]

\[
\begin{align*}
0' &\quad 7 & 4 \\
1' &\quad 5 & 6 \\
2' &\quad 9 & 1 \\
\end{align*}
\]

\[
X\quad \quad \quad \quad \quad Y
\]

\[
\begin{align*}
0 &\quad 15 & 7 & 3 \\
1 &\quad 5 & 6 & 2 \\
2 &\quad 4 & 9 & 1 \\
\end{align*}
\]

\[
\begin{align*}
0' &\quad 7 & 4 \\
1' &\quad 5 & 6 \\
2' &\quad 9 & 1 \\
\end{align*}
\]

\[
\begin{align*}
0 &\quad 15 & 7 & 3 \\
1 &\quad 5 & 6 & 2 \\
2 &\quad 4 & 9 & 1 \\
\end{align*}
\]

\[
\begin{align*}
0' &\quad 7 & 4 \\
1' &\quad 5 & 6 \\
2' &\quad 9 & 1 \\
\end{align*}
\]

\[
X\quad \quad \quad \quad \quad Y
\]

\[
\begin{align*}
0 &\quad 15 & 7 & 3 \\
1 &\quad 5 & 6 & 2 \\
2 &\quad 4 & 9 & 1 \\
\end{align*}
\]

\[
\begin{align*}
0' &\quad 7 & 4 \\
1' &\quad 5 & 6 \\
2' &\quad 9 & 1 \\
\end{align*}
\]

\[
\begin{align*}
0 &\quad 15 & 7 & 3 \\
1 &\quad 5 & 6 & 2 \\
2 &\quad 4 & 9 & 1 \\
\end{align*}
\]

\[
\begin{align*}
0' &\quad 7 & 4 \\
1' &\quad 5 & 6 \\
2' &\quad 9 & 1 \\
\end{align*}
\]

\[
X\quad \quad \quad \quad \quad Y
\]

\[
\begin{align*}
0 &\quad 15 & 7 & 3 \\
1 &\quad 5 & 6 & 2 \\
2 &\quad 4 & 9 & 1 \\
\end{align*}
\]

\[
\begin{align*}
0' &\quad 7 & 4 \\
1' &\quad 5 & 6 \\
2' &\quad 9 & 1 \\
\end{align*}
\]

\[
X\quad \quad \quad \quad \quad Y
\]

\[
\begin{align*}
0 &\quad 15 & 7 & 3 \\
1 &\quad 5 & 6 & 2 \\
2 &\quad 4 & 9 & 1 \\
\end{align*}
\]

\[
\begin{align*}
0' &\quad 7 & 4 \\
1' &\quad 5 & 6 \\
2' &\quad 9 & 1 \\
\end{align*}
\]

\[
X\quad \quad \quad \quad \quad Y
\]

\[
\begin{align*}
0 &\quad 15 & 7 & 3 \\
1 &\quad 5 & 6 & 2 \\
2 &\quad 4 & 9 & 1 \\
\end{align*}
\]

\[
\begin{align*}
0' &\quad 7 & 4 \\
1' &\quad 5 & 6 \\
2' &\quad 9 & 1 \\
\end{align*}
\]

\[
X\quad \quad \quad \quad \quad Y
\]

\[
\begin{align*}
0 &\quad 15 & 7 & 3 \\
1 &\quad 5 & 6 & 2 \\
2 &\quad 4 & 9 & 1 \\
\end{align*}
\]

\[
\begin{align*}
0' &\quad 7 & 4 \\
1' &\quad 5 & 6 \\
2' &\quad 9 & 1 \\
\end{align*}
\]

\[
X\quad \quad \quad \quad \quad Y
\]

\[
\begin{align*}
0 &\quad 15 & 7 & 3 \\
1 &\quad 5 & 6 & 2 \\
2 &\quad 4 & 9 & 1 \\
\end{align*}
\]

\[
\begin{align*}
0' &\quad 7 & 4 \\
1' &\quad 5 & 6 \\
2' &\quad 9 & 1 \\
\end{align*}
\]

\[
X\quad \quad \quad \quad \quad Y
\]

\[
\begin{align*}
0 &\quad 15 & 7 & 3 \\
1 &\quad 5 & 6 & 2 \\
2 &\quad 4 & 9 & 1 \\
\end{align*}
\]

\[
\begin{align*}
0' &\quad 7 & 4 \\
1' &\quad 5 & 6 \\
2' &\quad 9 & 1 \\
\end{align*}
\]

\[
X\quad \quad \quad \quad \quad Y
\]

\[
\begin{align*}
0 &\quad 15 & 7 & 3 \\
1 &\quad 5 & 6 & 2 \\
2 &\quad 4 & 9 & 1 \\
\end{align*}
\]

\[
\begin{align*}
0' &\quad 7 & 4 \\
1' &\quad 5 & 6 \\
2' &\quad 9 & 1 \\
\end{align*}
\]

\[
X\quad \quad \quad \quad \quad Y
\]

\[
\begin{align*}
0 &\quad 15 & 7 & 3 \\
1 &\quad 5 & 6 & 2 \\
2 &\quad 4 & 9 & 1 \\
\end{align*}
\]

\[
\begin{align*}
0' &\quad 7 & 4 \\
1' &\quad 5 & 6 \\
2' &\quad 9 & 1 \\
\end{align*}
\]

\[
X\quad \quad \quad \quad \quad Y
\]

\[
\begin{align*}
0 &\quad 15 & 7 & 3 \\
1 &\quad 5 & 6 & 2 \\
2 &\quad 4 & 9 & 1 \\
\end{align*}
\]

\[
\begin{align*}
0' &\quad 7 & 4 \\
1' &\quad 5 & 6 \\
2' &\quad 9 & 1 \\
\end{align*}
\]

\[
X\quad \quad \quad \quad \quad Y
\]

\[
\begin{align*}
0 &\quad 15 & 7 & 3 \\
1 &\quad 5 & 6 & 2 \\
2 &\quad 4 & 9 & 1 \\
\end{align*}
\]

\[
\begin{align*}
0' &\quad 7 & 4 \\
1' &\quad 5 & 6 \\
2' &\quad 9 & 1 \\
\end{align*}
\]

\[
X\quad \quad \quad \quad \quad Y
\]

\[
\begin{align*}
0 &\quad 15 & 7 & 3 \\
1 &\quad 5 & 6 & 2 \\
2 &\quad 4 & 9 & 1 \\
\end{align*}
\]

\[
\begin{align*}
0' &\quad 7 & 4 \\
1' &\quad 5 & 6 \\
2' &\quad 9 & 1 \\
\end{align*}
\]

\[
X\quad \quad \quad \quad \quad Y
\]

\[
\begin{align*}
0 &\quad 15 & 7 & 3 \\
1 &\quad 5 & 6 & 2 \\
2 &\quad 4 & 9 & 1 \\
\end{align*}
\]

\[
\begin{align*}
0' &\quad 7 & 4 \\
1' &\quad 5 & 6 \\
2' &\quad 9 & 1 \\
\end{align*}
\]

\[
X\quad \quad \quad \quad \quad Y
\]

\[
\begin{align*}
0 &\quad 15 & 7 & 3 \\
1 &\quad 5 & 6 & 2 \\
2 &\quad 4 & 9 & 1 \\
\end{align*}
\]

\[
\begin{align*}
0' &\quad 7 & 4 \\
1' &\quad 5 & 6 \\
2' &\quad 9 & 1 \\
\end{align*}
\]

\[
X\quad \quad \quad \quad \quad Y
\]

\[
\begin{align*}
0 &\quad 15 & 7 & 3 \\
1 &\quad 5 & 6 & 2 \\
2 &\quad 4 & 9 & 1 \\
\end{align*}
\]

\[
\begin{align*}
0' &\quad 7 & 4 \\
1' &\quad 5 & 6 \\
2' &\quad 9 & 1 \\
\end{align*}
\]

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

<table>
<thead>
<tr>
<th>Title</th>
<th>Course #</th>
<th>Professor</th>
<th>Day/Time</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>YAWA, The</td>
<td>W65166</td>
<td>Scott, Andrew</td>
<td>M/W 11:30-2:30 pm</td>
<td>Hodtman G302</td>
</tr>
<tr>
<td>America and the Potting Pot</td>
<td>W65157</td>
<td>Shaked, Karen</td>
<td>T/TH 9:00-10:30 am</td>
<td>Building 026</td>
</tr>
<tr>
<td>America and the Potting Pot</td>
<td>W65158</td>
<td>Shaked, Karen</td>
<td>T/TH 11:00-12:30 pm</td>
<td>Hodtman G304</td>
</tr>
<tr>
<td>American Mythology</td>
<td>W65191</td>
<td>Laufeberg, George</td>
<td>T/TH 12:30-1:30 pm</td>
<td>99 Alexander 101</td>
</tr>
<tr>
<td>American Revolution</td>
<td>W65194</td>
<td>Gross, Dan</td>
<td>W/Th 11:00-12:30 am</td>
<td>Building 026</td>
</tr>
<tr>
<td>Animal Mind, The</td>
<td>W65193</td>
<td>Gould, James</td>
<td>W/Th 8:00-9:30 am</td>
<td>Building 026</td>
</tr>
<tr>
<td>Art of Adventure, The</td>
<td>W65153</td>
<td>Waff, Anne</td>
<td>T/TH 11:00-12:30 am</td>
<td>Building 027</td>
</tr>
</tbody>
</table>

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 sensor 1 and line $j$ from sensor 2.
- Due to measurement errors, we might have $c_{ij} > 0$.
- Solve assignment problem to locate $n$ objects.

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).

**Algorithm for Ranked Assignments with Applications to Multitarget Tracking**

William E. Jones  
Department of Mathematics, California Institute of Technology  

A compact algorithm is provided for solving a general class of optimization problems that arise in the context of multi-target tracking. The algorithm is applicable to a wide range of problems where the goal is to assign a fixed number of targets to a fixed number of sensors while minimizing a cost function of the assignment. The algorithm is based on a linear programming formulation and is shown to be computationally efficient in practice. The algorithm is illustrated through a set of example problems and is shown to be applicable to a variety of real-world scenarios.
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.

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 $t$.

Assignment problem: successive shortest path algorithm

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

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 $s \rightarrow t$ 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 $x$. •

Reduced costs

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

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**

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

**Successive shortest path algorithm**

**Successive shortest path algorithm**

**Initialization.**
- $M = \emptyset$.
- 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 \( v \in X \cup Y \): \( p(v) \leftarrow p(v) + d(v) \).

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

![Diagram showing the computation of shortest path distances and matching](image1)

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

![Diagram showing updated matching and shortest path distances](image2)

---

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

![Diagram showing the computation of shortest path distances and matching](image3)

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

![Diagram showing updated matching and shortest path distances](image4)
**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 $v \in X \cup Y$: $p(v) \leftarrow p(v) + d(v)$.

![Diagram of a graph with nodes labeled $s$, $1$, $2$, $3$, $0$, $0'$, $2'$, and edges labeled with distances and reduced costs.](image)

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

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

**Successive shortest path algorithm**

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

![Diagram of a graph with nodes labeled $s$, $1$, $2$, $3$, $0$, $0'$, $2'$, and edges labeled with distances and reduced costs.](image)

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

$(x, y) \notin M$
- $(x, y)$ 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 $(x, y) \in M$.

![Diagram of a graph with nodes labeled $s$, $1$, $2$, $3$, $0$, $0'$, $2'$, and edges labeled with distances and reduced costs.](image)
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 $O(n^2)$ 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 $O(m n^{1/2} \log(nC))$ time algorithm for the problem when the costs are integers between 0 and $C$.

---

History

**Thurndike 1950.** Formulated in a modern way by a psychologist.
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

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

The Hungarian method for the assignment problem

H. W. Kuhn
Brandeis University

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 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

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

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

De inventando ordine systematico sequationum differentialium vulgarii usitacionis. (Ex H. C. J. Jacobi. Tiberio principis in nostris praebat C. W. Borchard.)

History

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

Munkres 1957. Reviewed algorithm; observed \( O(n^2) \) implementation.

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

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....

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

\[
\begin{align*}
\text{Problemas} &
\text{m} \times \text{m} \text{es quinquies enunschum que su sume horizontale e seri seri verticalis omnes quaeque ad terminum. Es ille quinque valores alineas e transversales s. e. o. alineas horizontalibus simil atque verticalibus divers posse, quod furt vitat 3...5 moedis, or omnibus ille modus generis ad ipsum commove distributum aliquem est susceptum maximum.}
\end{align*}
\]
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.

![Diagram of input-queued switching](image)

**FIFO queuing.** 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.

![Diagram of FIFO queuing](image)
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.

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).

Practice.

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

Input-queued switching