Today

Today

Maximum Weight Matching

Today

Maximum Weight Matching
Undergraduate: saw maximum matching!

Today

Maximum Weight Matching
Undergraduate: saw maximum matching! (hopefully.)

Today

Maximum Weight Matching
Undergraduate: saw maximum matching! (hopefully.) Will review.

Matching.

Given a bipartite graph, $G = (U, V, E)$, with edge weights $w : E \to R$, find a maximum weight matching.

Matching.

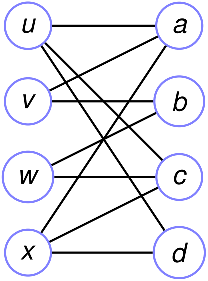
Given a bipartite graph, $G = (U, V, E)$, with edge weights $w : E \to R$, find a maximum weight matching.

A matching is a set of edges where no two share an endpoint.

Matching.

Given a bipartite graph, $G = (U, V, E)$, with edge weights $w : E \to R$, find a maximum weight matching.

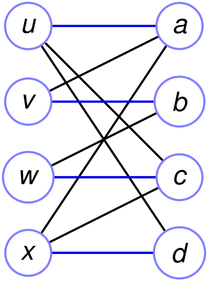
A matching is a set of edges where no two share an endpoint.



Matching.

Given a bipartite graph, $G = (U, V, E)$, with edge weights $w : E \to R$, find a maximum weight matching.

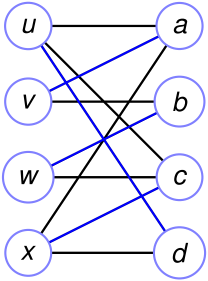
A matching is a set of edges where no two share an endpoint.



Matching.

Given a bipartite graph, $G = (U, V, E)$, with edge weights $w : E \to R$, find a maximum weight matching.

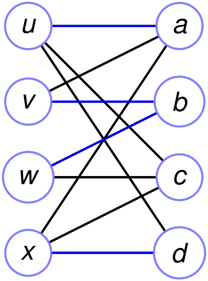
A matching is a set of edges where no two share an endpoint.



Matching.

Given a bipartite graph, $G = (U, V, E)$, with edge weights $w : E \rightarrow R$, find a maximum weight matching.

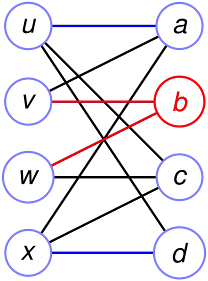
A matching is a set of edges where no two share an endpoint.



Matching.

Given a bipartite graph, $G = (U, V, E)$, with edge weights $w : E \to R$, find a maximum weight matching.

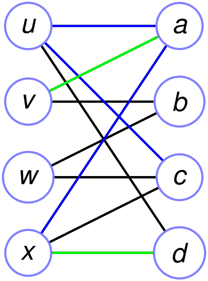
A matching is a set of edges where no two share an endpoint.



Matching.

Given a bipartite graph, $G = (U, V, E)$, with edge weights $w : E \to R$, find a maximum weight matching.

A matching is a set of edges where no two share an endpoint.

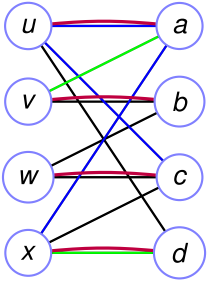


Blue – 3. Green - 2,
Black - 1, Non-edges - 0.

Matching.

Given a bipartite graph, $G = (U, V, E)$, with edge weights $w : E \to R$, find a maximum weight matching.

A matching is a set of edges where no two share an endpoint.



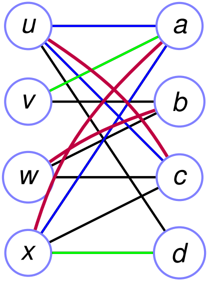
Blue – 3. Green - 2,
Black - 1, Non-edges - 0.

Solution Value: 7.

Matching.

Given a bipartite graph, $G = (U, V, E)$, with edge weights $w : E \to R$, find a maximum weight matching.

A matching is a set of edges where no two share an endpoint.



Blue – 3. Green - 2,
Black - 1, Non-edges - 0.

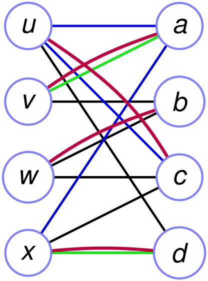
Solution Value: 7.

Solution Value: 7.

Matching.

Given a bipartite graph, $G = (U, V, E)$, with edge weights $w : E \to R$, find a maximum weight matching.

A matching is a set of edges where no two share an endpoint.



Blue – 3. Green - 2,
Black - 1, Non-edges - 0.

Solution Value: 7.

Solution Value: 7.

Solution Value: 8.

Applications

Jobs to workers.

Applications

Jobs to workers.

Teachers to classes.

Applications

Jobs to workers.

Teachers to classes.

Classes to classrooms.

Applications

Jobs to workers.

Teachers to classes.

Classes to classrooms.

“The assignment problem”

Applications

Jobs to workers.

Teachers to classes.

Classes to classrooms.

"The assignment problem"

Min Weight Matching.

Applications

Jobs to workers.

Teachers to classes.

Classes to classrooms.

"The assignment problem"

Min Weight Matching.
    Negate values and find maximum weight matching.

Vertex Cover

Given a bipartite graph, $G = (U, V, E)$, with edge weights $w : E \rightarrow R$, find an vertex cover function of minimum total value.

Vertex Cover

Given a bipartite graph, $G = (U, V, E)$, with edge weights $w : E \rightarrow R$, find an vertex cover function of minimum total value.

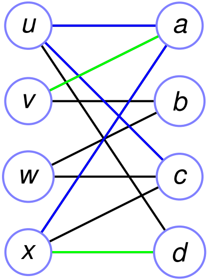
A function $p : V \rightarrow R$, where for all edges, $e = (u, v)$, $p(u) + p(v) \geq w(e)$.

Vertex Cover

Given a bipartite graph, $G = (U, V, E)$, with edge weights $w : E \rightarrow R$, find an vertex cover function of minimum total value.

A function $p : V \rightarrow R$, where for all edges, $e = (u, v)$, $p(u) + p(v) \geq w(e)$.

Minimize $\sum_{v \in U \cup V} p(u)$.

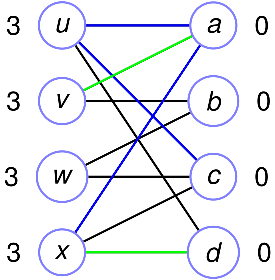


Vertex Cover

Given a bipartite graph, $G = (U, V, E)$, with edge weights $w : E \to R$, find an vertex cover function of minimum total value.

A function $p : V \to R$, where for all edges, $e = (u, v)$, $p(u) + p(v) \geq w(e)$.

Minimize $\sum_{v \in U \cup V} p(u)$.



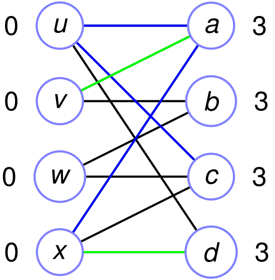
Solution Value: 12.

Vertex Cover

Given a bipartite graph, $G = (U, V, E)$, with edge weights $w : E \to R$, find an vertex cover function of minimum total value.

A function $p : V \to R$, where for all edges, $e = (u, v)$, $p(u) + p(v) \geq w(e)$.

Minimize $\sum_{v \in U \cup V} p(u)$.



Solution Value: 12.

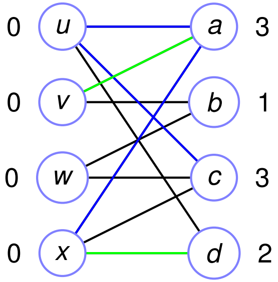
Solution Value: 12.

Vertex Cover

Given a bipartite graph, $G = (U, V, E)$, with edge weights $w : E \to R$, find an vertex cover function of minimum total value.

A function $p : V \to R$, where for all edges, $e = (u, v)$, $p(u) + p(v) \geq w(e)$.

Minimize $\sum_{v \in U \cup V} p(u)$.



Solution Value: 12.

Solution Value: 12.

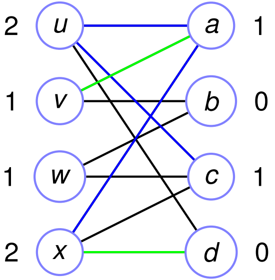
Solution Value: 9.

Vertex Cover

Given a bipartite graph, $G = (U, V, E)$, with edge weights $w : E \rightarrow R$, find an vertex cover function of minimum total value.

A function $p : V \rightarrow R$, where for all edges, $e = (u, v)$, $p(u) + p(v) \geq w(e)$.

Minimize $\sum_{v \in U \cup V} p(u)$.



Solution Value: 12.

Solution Value: 12.

Solution Value: 9.

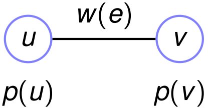
Solution Value: 8.

Cover is upper bound.

Feasible $p(\cdot)$,

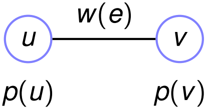
Cover is upper bound.

Feasible $p(\cdot)$, for edge $e = (u, v)$, $p(u) + p(v) \geq w(e)$.

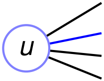


Cover is upper bound.

Feasible $p(\cdot)$, for edge $e = (u,v)$, $p(u) + p(v) \geq w(e)$.

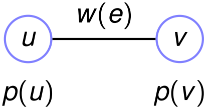


For a matching M, each u is the endpoint of at most one edge in M.

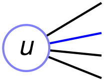


Cover is upper bound.

Feasible $p(\cdot)$, for edge $e = (u,v)$, $p(u) + p(v) \geq w(e)$.



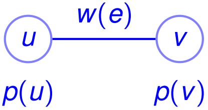
For a matching M, each u is the endpoint of at most one edge in M.



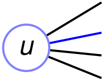
$$\sum_{e=(u,v)\in M} w(e)$$

Cover is upper bound.

Feasible $p(\cdot)$, for edge $e = (u, v)$, $p(u) + p(v) \geq w(e)$.



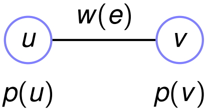
For a matching M, each u is the endpoint of at most one edge in M.



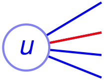
$$\sum_{e=(u,v)\in M} w(e) \leq \sum_{e=(u,v)\in M} (p(u) + p(v))$$

Cover is upper bound.

Feasible $p(\cdot)$, for edge $e = (u,v)$, $p(u) + p(v) \geq w(e)$.



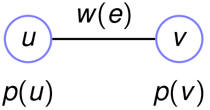
For a matching M, each u is the endpoint of at most one edge in M.



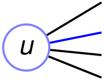
$$\sum_{e=(u,v)\in M} w(e) \leq \sum_{e=(u,v)\in M} (p(u) + p(v)) \leq \sum_{u \in U} p(u) + \sum_{v \in V} p(v)$$

Cover is upper bound.

Feasible $p(\cdot)$, for edge $e = (u,v)$, $p(u) + p(v) \geq w(e)$.



For a matching M, each u is the endpoint of at most one edge in M.

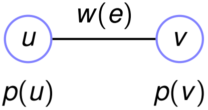


$$\sum_{e=(u,v)\in M} w(e) \leq \sum_{e=(u,v)\in M} (p(u) + p(v)) \leq \sum_{u \in U} p(u) + \sum_{v \in V} p(v)$$

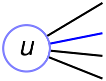
Holds with equality if

Cover is upper bound.

Feasible $p(\cdot)$, for edge $e = (u,v)$, $p(u) + p(v) \geq w(e)$.



For a matching M, each u is the endpoint of at most one edge in M.



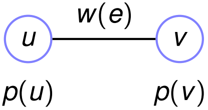
$$\sum_{e=(u,v)\in M} w(e) \leq \sum_{e=(u,v)\in M} (p(u) + p(v)) \leq \sum_{u \in U} p(u) + \sum_{v \in V} p(v)$$

Holds with equality if

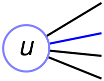
for $e \in M$, $w(e) = p(u) + p(v)$ (Defn: tight edge.) and

Cover is upper bound.

Feasible $p(\cdot)$, for edge $e = (u, v)$, $p(u) + p(v) \geq w(e)$.



For a matching M, each u is the endpoint of at most one edge in M.

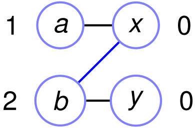


$$\sum_{e=(u,v)\in M} w(e) \leq \sum_{e=(u,v)\in M} (p(u) + p(v)) \leq \sum_{u \in U} p(u) + \sum_{v \in V} p(v)$$

Holds with equality if

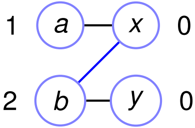
for $e \in M$, $w(e) = p(u) + p(v)$ (Defn: tight edge.) and
perfect matching.

Simple example.



Blue edge – 2, Others – 1.

Simple example.

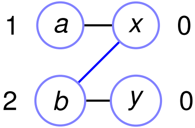


Blue edge – 2, Others – 1.

Using max incident edge.

Value: 3.

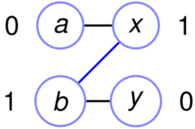
Simple example.



Blue edge – 2, Others – 1.

Using max incident edge.

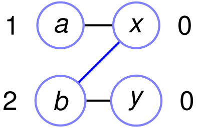
Value: 3.
Using max incident edge.



Value: 2.
Same as optimal matching!

Proof of optimality.

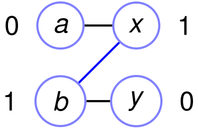
Simple example.



Blue edge – 2, Others – 1.

Using max incident edge.

Value: 3.
Using max incident edge.

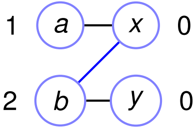


Value: 2.
Same as optimal matching!

Proof of optimality.

Matching and cover are optimal,

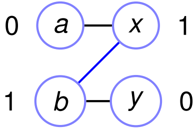
Simple example.



Blue edge – 2, Others – 1.

Using max incident edge.

Value: 3.
Using max incident edge.

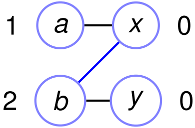


Value: 2.
Same as optimal matching!

Proof of optimality.

Matching and cover are optimal,
edges in matching have $w(e) = p(u) + p(v)$. Tight edge.

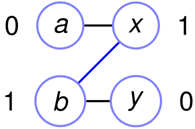
Simple example.



Blue edge – 2, Others – 1.

Using max incident edge.

Value: 3.
Using max incident edge.



Value: 2.
  Same as optimal matching!

Proof of optimality.

Matching and cover are optimal,
  edges in matching have $w(e) = p(u) + p(v)$. Tight edge.
  all nodes are matched.

Maximum Matching

Given a bipartite graph, $G = (U, V, E)$, find a maximum sized matching.

Maximum Matching

Given a bipartite graph, $G = (U, V, E)$, find a maximum sized matching.

Size of maximum matching of a path on length n?

Maximum Matching

Given a bipartite graph, $G = (U, V, E)$, find a maximum sized matching.

Size of maximum matching of a path on length n?
(A) $n/2$

Maximum Matching

Given a bipartite graph, $G = (U, V, E)$, find a maximum sized matching.

Size of maximum matching of a path on length n?

(A) $n/2$
(B) $\lfloor n/2 \rfloor$

Maximum Matching

Given a bipartite graph, $G = (U, V, E)$, find a maximum sized matching.

Size of maximum matching of a path on length n?
(A) $n/2$
(B) $\lfloor n/2 \rfloor$

(Rao would have said (A), don't worry.)

Maximum Matching

Given a bipartite graph, $G = (U, V, E)$, find a maximum sized matching.

Size of maximum matching of a path on length n?

(A) $n/2$

(B) $\lfloor n/2 \rfloor$

(Rao would have said (A), don't worry.)

Why?

Maximum Matching

Given a bipartite graph, $G = (U, V, E)$, find a maximum sized matching.

Size of maximum matching of a path on length n?

(A) $n/2$
(B) $\lfloor n/2 \rfloor$

(Rao would have said (A), don't worry.)

Why?

Alg: Start at end, and alternately put in edge or not.

Maximum Matching

Given a bipartite graph, $G = (U, V, E)$, find a maximum sized matching.

Size of maximum matching of a path on length n?
(A) $n/2$
(B) $\lfloor n/2 \rfloor$

(Rao would have said (A), don't worry.)

Why?
Alg: Start at end, and alternately put in edge or not.

What if one has a partial matching.

Maximum Matching

Given a bipartite graph, $G = (U, V, E)$, find a maximum sized matching.

Size of maximum matching of a path on length n?

(A) $n/2$

(B) $\lfloor n/2 \rfloor$

(Rao would have said (A), don't worry.)

Why?

Alg: Start at end, and alternately put in edge or not.

What if one has a partial matching.

How do you make it bigger?

Maximum Matching

Given a bipartite graph, $G = (U, V, E)$, find a maximum sized matching.

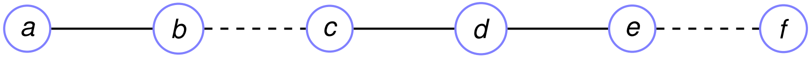
Size of maximum matching of a path on length n?
(A) $n/2$
(B) $\lfloor n/2 \rfloor$

(Rao would have said (A), don't worry.)

Why?
Alg: Start at end, and alternately put in edge or not.

What if one has a partial matching.

How do you make it bigger?



Maximum Matching

Given a bipartite graph, $G = (U, V, E)$, find a maximum sized matching.

Size of maximum matching of a path on length n?

(A) $n/2$
(B) $\lfloor n/2 \rfloor$

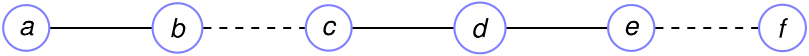
(Rao would have said (A), don't worry.)

Why?

Alg: Start at end, and alternately put in edge or not.

What if one has a partial matching.

How do you make it bigger?



Greedily adding fails.

Maximum Matching

Given a bipartite graph, $G = (U, V, E)$, find a maximum sized matching.

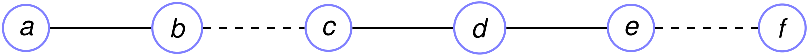
Size of maximum matching of a path on length n?
(A) $n/2$
(B) $\lfloor n/2 \rfloor$

(Rao would have said (A), don't worry.)

Why?
Alg: Start at end, and alternately put in edge or not.

What if one has a partial matching.

How do you make it bigger?



Greedily adding fails. So how?

Maximum Matching

Given a bipartite graph, $G = (U, V, E)$, find a maximum sized matching.

Size of maximum matching of a path on length n?

(A) $n/2$

(B) $\lfloor n/2 \rfloor$

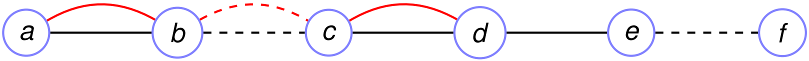
(Rao would have said (A), don't worry.)

Why?

Alg: Start at end, and alternately put in edge or not.

What if one has a partial matching.

How do you make it bigger?



Greedily adding fails. So how? Augmenting Alternating Path.

Maximum Matching

Given a bipartite graph, $G = (U, V, E)$, find a maximum sized matching.

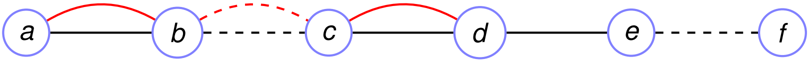
Size of maximum matching of a path on length n?
(A) $n/2$
(B) $\lfloor n/2 \rfloor$

(Rao would have said (A), don't worry.)

Why?
Alg: Start at end, and alternately put in edge or not.

What if one has a partial matching.

How do you make it bigger?



Greedily adding fails. So how? Augmenting Alternating Path. Switch!

Maximum Matching

Given a bipartite graph, $G = (U, V, E)$, find a maximum sized matching.

Size of maximum matching of a path on length n?

(A) $n/2$

(B) $\lfloor n/2 \rfloor$

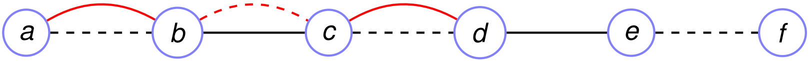
(Rao would have said (A), don't worry.)

Why?

Alg: Start at end, and alternately put in edge or not.

What if one has a partial matching.

How do you make it bigger?



Greedily adding fails. So how? Augmenting Alternating Path. Switch!

Maximum Matching

Given a bipartite graph, $G = (U, V, E)$, find a maximum sized matching.

Maximum Matching

Given a bipartite graph, $G = (U, V, E)$, find a maximum sized matching.

Key Idea: Augmenting Alternating Paths.

Maximum Matching

Given a bipartite graph, $G = (U, V, E)$, find a maximum sized matching.

Key Idea: Augmenting Alternating Paths.

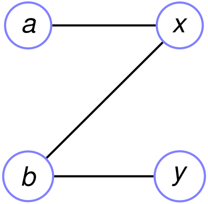
Example:

Maximum Matching

Given a bipartite graph, $G = (U, V, E)$, find a maximum sized matching.

Key Idea: Augmenting Alternating Paths.

Example:

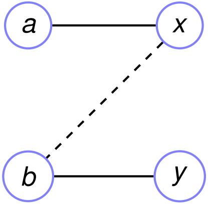


Maximum Matching

Given a bipartite graph, $G = (U, V, E)$, find a maximum sized matching.

Key Idea: Augmenting Alternating Paths.

Example:

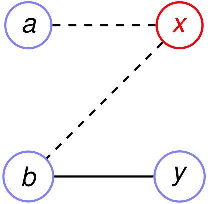


Maximum Matching

Given a bipartite graph, $G = (U, V, E)$, find a maximum sized matching.

Key Idea: Augmenting Alternating Paths.

Example:

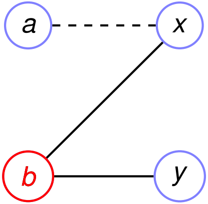


Maximum Matching

Given a bipartite graph, $G = (U, V, E)$, find a maximum sized matching.

Key Idea: Augmenting Alternating Paths.

Example:



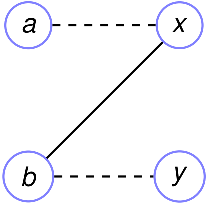
Start at unmatched node(s),
follow unmatched edge(s),
follow matched.
Repeat until an unmatched node.

Maximum Matching

Given a bipartite graph, $G = (U, V, E)$, find a maximum sized matching.

Key Idea: Augmenting Alternating Paths.

Example:

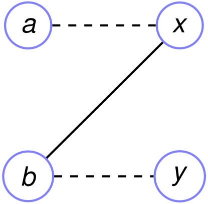


Maximum Matching

Given a bipartite graph, $G = (U, V, E)$, find a maximum sized matching.

Key Idea: Augmenting Alternating Paths.

Example:



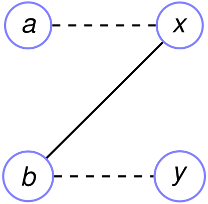
Start at unmatched node(s),

Maximum Matching

Given a bipartite graph, $G = (U, V, E)$, find a maximum sized matching.

Key Idea: Augmenting Alternating Paths.

Example:



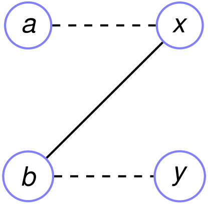
Start at unmatched node(s),
  follow unmatched edge(s),

Maximum Matching

Given a bipartite graph, $G = (U, V, E)$, find a maximum sized matching.

Key Idea: Augmenting Alternating Paths.

Example:



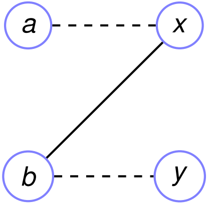
Start at unmatched node(s),
follow unmatched edge(s),
follow matched.

Maximum Matching

Given a bipartite graph, $G = (U, V, E)$, find a maximum sized matching.

Key Idea: Augmenting Alternating Paths.

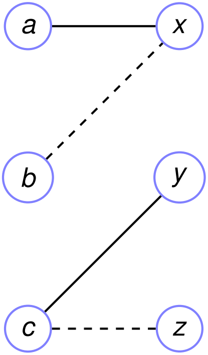
Example:



Start at unmatched node(s),
follow unmatched edge(s),
follow matched.
Repeat until an unmatched node.

No perfect matching

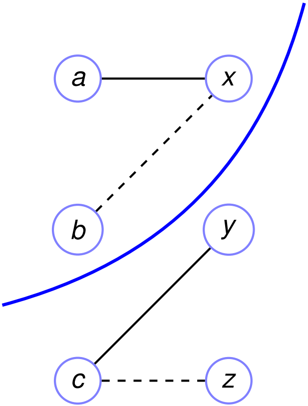
No perfect matching



Can't increase matching size.

No alternating path from (a) to (y).

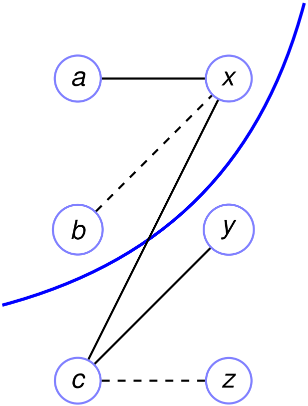
No perfect matching



Can’t increase matching size.
  No alternating path from (a) to (y).

Cut!

No perfect matching

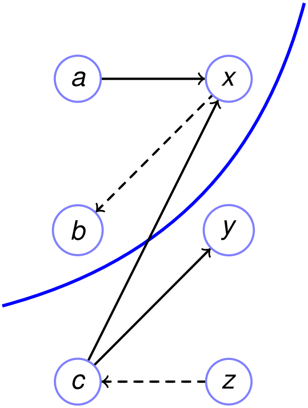


Can’t increase matching size.
No alternating path from (a) to (y).

Cut!

Still no augmenting path.
Still Cut?

No perfect matching



Can’t increase matching size.
No alternating path from (a) to (y).

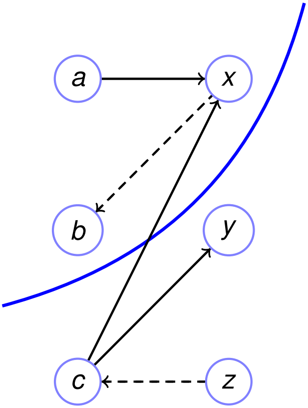
Cut!

Still no augmenting path.
Still Cut?

Use directed graph!
Cut in this graph.

Algorithm:

No perfect matching



Can't increase matching size.
No alternating path from (a) to (y).

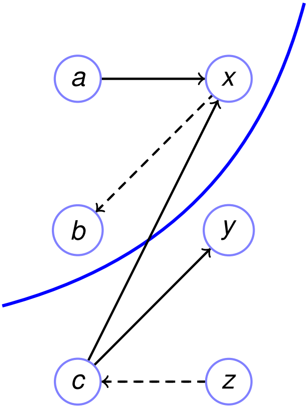
Cut!

Still no augmenting path.
Still Cut?

Use directed graph!
Cut in this graph.

Algorithm:
Given matching.

No perfect matching



Can't increase matching size.
No alternating path from (a) to (y).

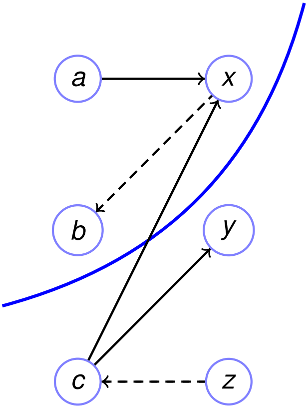
Cut!

Still no augmenting path.
Still Cut?

Use directed graph!
Cut in this graph.

Algorithm:
Given matching.
Direct unmatched edges U to V, matched V to U.

No perfect matching



Can't increase matching size.
No alternating path from (a) to (y).

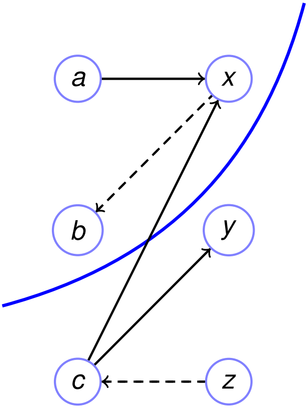
Cut!

Still no augmenting path.
Still Cut?

Use directed graph!
Cut in this graph.

Algorithm:
Given matching.
Direct unmatched edges U to V, matched V to U.
Find path between unmatched nodes on left to right. (BFS, DFS).

No perfect matching



Can't increase matching size.
No alternating path from (a) to (y).

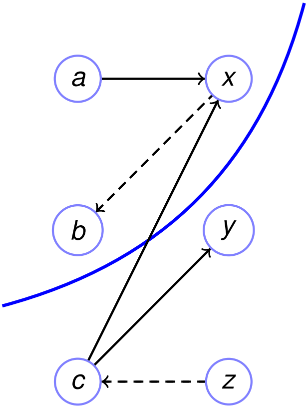
Cut!

Still no augmenting path.
Still Cut?

Use directed graph!
Cut in this graph.

Algorithm:
Given matching.
Direct unmatched edges U to V, matched V to U.
Find path between unmatched nodes on left to right. (BFS, DFS).
Until everything matched

No perfect matching



Can’t increase matching size.
No alternating path from (a) to (y).

Cut!

Still no augmenting path.
Still Cut?

Use directed graph!
Cut in this graph.

Algorithm:
Given matching.
Direct unmatched edges U to V, matched V to U.
Find path between unmatched nodes on left to right. (BFS, DFS).
Until everything matched ... or output a cut.

Back to Maximum Weight Matching.

Want vertex cover (price function) $p(\cdot)$ and matching where.

Back to Maximum Weight Matching.

Want vertex cover (price function) $p(\cdot)$ and matching where.

Optimal solutions to *both* if

Back to Maximum Weight Matching.

Want vertex cover (price function) $p(\cdot)$ and matching where.

Optimal solutions to *both* if
for $e \in M$, $w(e) = p(u) + p(v)$ (Defn: tight edge.) and

Back to Maximum Weight Matching.

Want vertex cover (price function) $p(\cdot)$ and matching where.

Optimal solutions to *both* if

for $e \in M$, $w(e) = p(u) + p(v)$ (Defn: tight edge.) and

perfect matching.

Maximum Weight Matching

Goal: perfect matching on tight edges.

Maximum Weight Matching

Goal: perfect matching on tight edges.

Algorithm

Maximum Weight Matching

Goal: perfect matching on tight edges.

Algorithm

Init: empty matching, feasible cover function ($p(\cdot)$)

Maximum Weight Matching

Goal: perfect matching on tight edges.

Algorithm

Init: empty matching, feasible cover function ($p(\cdot)$)

Add tight edges to matching.

Maximum Weight Matching

Goal: perfect matching on tight edges.

Algorithm

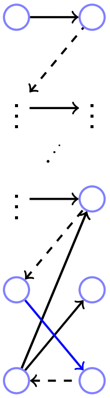
Init: empty matching, feasible cover function ($p(\cdot)$)

Add tight edges to matching.

Use alt./aug. paths of tight edges.

Maximum Weight Matching

Goal: perfect matching on tight edges.



Algorithm

Init: empty matching, feasible cover function ($p(\cdot)$)

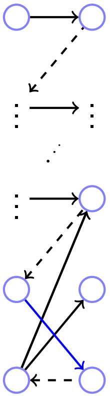
Add tight edges to matching.

Use alt./aug. paths of tight edges.

"maximum matching algorithm."

Maximum Weight Matching

Goal: perfect matching on tight edges.



Algorithm

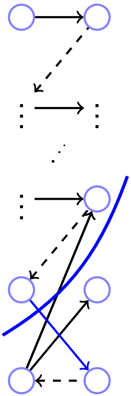
Init: empty matching, feasible cover function ($p(\cdot)$)

Add tight edges to matching.
    Use alt./aug. paths of tight edges.
        "maximum matching algorithm."

No augmenting path.

Maximum Weight Matching

Goal: perfect matching on tight edges.



Algorithm

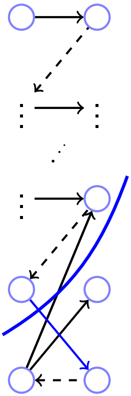
Init: empty matching, feasible cover function ($p(\cdot)$)

Add tight edges to matching.
  Use alt./aug. paths of tight edges.
    "maximum matching algorithm."

No augmenting path.
  Cut, (S, T), in directed graph of tight edges!

Maximum Weight Matching

Goal: perfect matching on tight edges.



Algorithm

Init: empty matching, feasible cover function ($p(\cdot)$)

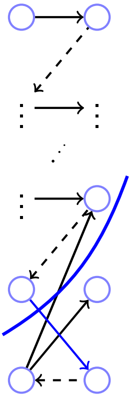
Add tight edges to matching.
 Use alt./aug. paths of tight edges.
  "maximum matching algorithm."

No augmenting path.
 Cut, (S, T), in directed graph of tight edges!

All edges across cut are not tight. (loose?)

Maximum Weight Matching

Goal: perfect matching on tight edges.



Algorithm

Init: empty matching, feasible cover function ($p(\cdot)$)

Add tight edges to matching.
   Use alt./aug. paths of tight edges.
      "maximum matching algorithm."

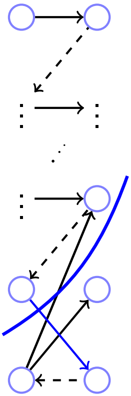
No augmenting path.
  Cut, (S, T), in directed graph of tight edges!

All edges across cut are not tight. (loose?)

Nontight edges leaving cut, go from S_U, T_V.

Maximum Weight Matching

Goal: perfect matching on tight edges.



Algorithm

Init: empty matching, feasible cover function ($p(\cdot)$)

Add tight edges to matching.
  Use alt./aug. paths of tight edges.
    "maximum matching algorithm."

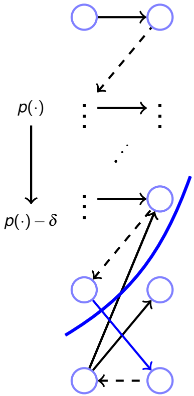
No augmenting path.
  Cut, (S, T), in directed graph of tight edges!

All edges across cut are not tight. (loose?)

Nontight edges leaving cut, go from S_U, T_V.

Maximum Weight Matching

Goal: perfect matching on tight edges.



Algorithm

Init: empty matching, feasible cover function ($p(\cdot)$)

Add tight edges to matching.
  Use alt./aug. paths of tight edges.
    "maximum matching algorithm."

No augmenting path.
 Cut, (S, T), in directed graph of tight edges!

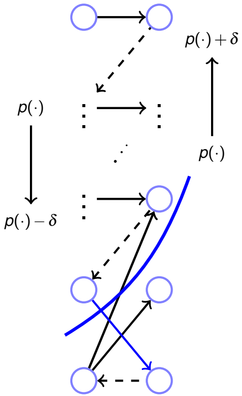
All edges across cut are not tight. (loose?)

Nontight edges leaving cut, go from S_U, T_V.

Lower prices in S_U,

Maximum Weight Matching

Goal: perfect matching on tight edges.



Algorithm

Init: empty matching, feasible cover function ($p(\cdot)$)

Add tight edges to matching.
Use alt./aug. paths of tight edges.
"maximum matching algorithm."

No augmenting path.
Cut, (S, T), in directed graph of tight edges!

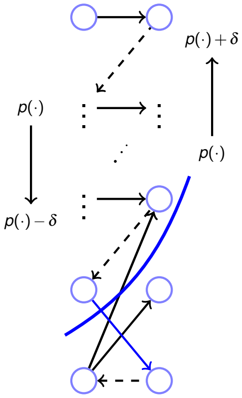
All edges across cut are not tight. (loose?)

Nontight edges leaving cut, go from S_U, T_V.

Lower prices in S_U, raise prices in S_V,

Maximum Weight Matching

Goal: perfect matching on tight edges.



Algorithm

Init: empty matching, feasible cover function ($p(\cdot)$)

Add tight edges to matching.
Use alt./aug. paths of tight edges.
"maximum matching algorithm."

No augmenting path.
Cut, (S,T), in directed graph of tight edges!

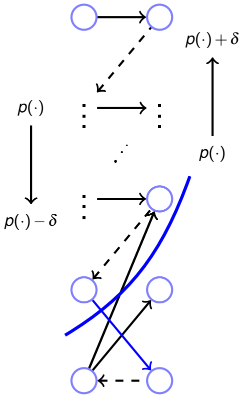
All edges across cut are not tight. (loose?)

Nontight edges leaving cut, go from S_U, T_V.

Lower prices in S_U, raise prices in S_V,
all explored edges still tight,
matched edges still tight

Maximum Weight Matching

Goal: perfect matching on tight edges.



Algorithm

Init: empty matching, feasible cover function ($p(\cdot)$)

Add tight edges to matching.
Use alt./aug. paths of tight edges.
"maximum matching algorithm."

No augmenting path.
Cut, (S, T), in directed graph of tight edges!

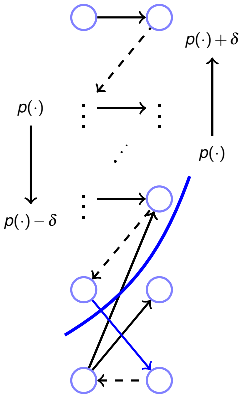
All edges across cut are not tight. (loose?)

Nontight edges leaving cut, go from S_U, T_V.

Lower prices in S_U, raise prices in S_V,
all explored edges still tight,
matched edges still tight
... and get new tight edge!

Maximum Weight Matching

Goal: perfect matching on tight edges.



Algorithm

Init: empty matching, feasible cover function ($p(\cdot)$)

Add tight edges to matching.
  Use alt./aug. paths of tight edges.
    "maximum matching algorithm."

No augmenting path.
  Cut, (S, T), in directed graph of tight edges!

All edges across cut are not tight. (loose?)

Nontight edges leaving cut, go from S_U, T_V.

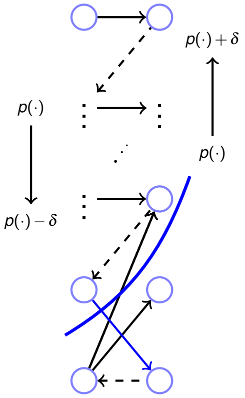
Lower prices in S_U, raise prices in S_V,
  all explored edges still tight,
  matched edges still tight

... and get new tight edge!

What's delta?

Maximum Weight Matching

Goal: perfect matching on tight edges.



Algorithm

Init: empty matching, feasible cover function ($p(\cdot)$)

Add tight edges to matching.
  Use alt./aug. paths of tight edges.
    "maximum matching algorithm."

No augmenting path.
 Cut, (S, T), in directed graph of tight edges!

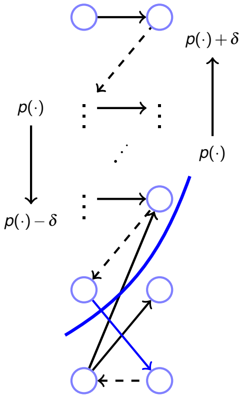
All edges across cut are not tight. (loose?)

Nontight edges leaving cut, go from S_U, T_V.

Lower prices in S_U, raise prices in S_V,
  all explored edges still tight,
  matched edges still tight
... and get new tight edge!
What's delta? $w(e) < p(u) + p(v) \rightarrow$

Maximum Weight Matching

Goal: perfect matching on tight edges.



Algorithm

Init: empty matching, feasible cover function ($p(\cdot)$)

Add tight edges to matching.
Use alt./aug. paths of tight edges.
"maximum matching algorithm."

No augmenting path.
Cut, (S, T), in directed graph of tight edges!

All edges across cut are not tight. (loose?)

Nontight edges leaving cut, go from S_U, T_V.

Lower prices in S_U, raise prices in S_V,
all explored edges still tight,
matched edges still tight
... and get new tight edge!
What's delta? $w(e) < p(u) + p(v) \rightarrow$
$\delta = min_{e \in (S_U \times T_V)} p(u) + p(v) - w(e)$.

Some details

Add 0 value edges, so that optimal solution contains perfect matching.

Some details

Add 0 value edges, so that optimal solution contains perfect matching.

Beginning “Matcher” Solution: $M = \{\}$.

Some details

Add 0 value edges, so that optimal solution contains perfect matching.

Beginning “Matcher” Solution: $M = \{\}$.

Feasible!

Some details

Add 0 value edges, so that optimal solution contains perfect matching.

Beginning "Matcher" Solution: $M = \{\}$.

Feasible! Value = 0.

Some details

Add 0 value edges, so that optimal solution contains perfect matching.

Beginning "Matcher" Solution: $M = \{\}$.

Feasible! Value = 0.

Beginning "Coverer" Solution:
$p(u) =$ maximum incident edge for $u \in U$,

Some details

Add 0 value edges, so that optimal solution contains perfect matching.

Beginning "Matcher" Solution: $M = \{\}$.

Feasible! Value = 0.

Beginning "Coverer" Solution:
$p(u) =$ maximum incident edge for $u \in U$, 0 otherwise.

Some details

Add 0 value edges, so that optimal solution contains perfect matching.

Beginning "Matcher" Solution: $M = \{\}$.

Feasible! Value = 0.

Beginning "Coverer" Solution:
$p(u) =$ maximum incident edge for $u \in U$, 0 otherwise.

Main Work:

Some details

Add 0 value edges, so that optimal solution contains perfect matching.

Beginning "Matcher" Solution: $M = \{\}$.

Feasible! Value = 0.

Beginning "Coverer" Solution:
$p(u) =$ maximum incident edge for $u \in U$, 0 otherwise.

Main Work:
breadth first search from unmatched nodes finds cut.

Some details

Add 0 value edges, so that optimal solution contains perfect matching.

Beginning "Matcher" Solution: $M = \{\}$.

Feasible! Value = 0.

Beginning "Coverer" Solution:
$p(u) =$ maximum incident edge for $u \in U$, 0 otherwise.

Main Work:
breadth first search from unmatched nodes finds cut.
Update prices (find minimum delta.)

Some details

Add 0 value edges, so that optimal solution contains perfect matching.

Beginning "Matcher" Solution: $M = \{\}$.

Feasible! Value = 0.

Beginning "Coverer" Solution:
$p(u) =$ maximum incident edge for $u \in U$, 0 otherwise.

Main Work:
breadth first search from unmatched nodes finds cut.
Update prices (find minimum delta.)

Some details

Add 0 value edges, so that optimal solution contains perfect matching.

Beginning "Matcher" Solution: $M = \{\}$.

Feasible! Value = 0.

Beginning "Coverer" Solution:
$p(u) =$ maximum incident edge for $u \in U$, 0 otherwise.

Main Work:
breadth first search from unmatched nodes finds cut.
Update prices (find minimum delta.)

Simple Implementation:

Some details

Add 0 value edges, so that optimal solution contains perfect matching.

Beginning "Matcher" Solution: $M = \{\}$.

Feasible! Value = 0.

Beginning "Coverer" Solution:
$p(u) =$ maximum incident edge for $u \in U$, 0 otherwise.

Main Work:
breadth first search from unmatched nodes finds cut.
Update prices (find minimum delta.)

Simple Implementation:
Each bfs either augments or adds node to S in next cut.

Some details

Add 0 value edges, so that optimal solution contains perfect matching.

Beginning "Matcher" Solution: $M = \{\}$.

Feasible! Value = 0.

Beginning "Coverer" Solution:
$p(u) =$ maximum incident edge for $u \in U$, 0 otherwise.

Main Work:
breadth first search from unmatched nodes finds cut.
Update prices (find minimum delta.)

Simple Implementation:
Each bfs either augments or adds node to S in next cut.
$O(n)$ iterations per augmentation.

Some details

Add 0 value edges, so that optimal solution contains perfect matching.

Beginning "Matcher" Solution: $M = \{\}$.

Feasible! Value = 0.

Beginning "Coverer" Solution:
$p(u) =$ maximum incident edge for $u \in U$, 0 otherwise.

Main Work:
breadth first search from unmatched nodes finds cut.
Update prices (find minimum delta.)

Simple Implementation:
Each bfs either augments or adds node to S in next cut.
$O(n)$ iterations per augmentation.
$O(n)$ augmentations.

Some details

Add 0 value edges, so that optimal solution contains perfect matching.

Beginning "Matcher" Solution: $M = \{\}$.

Feasible! Value = 0.

Beginning "Coverer" Solution:
$p(u) =$ maximum incident edge for $u \in U$, 0 otherwise.

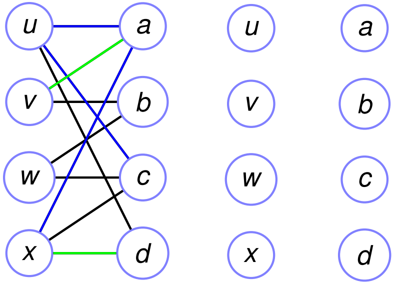
Main Work:
breadth first search from unmatched nodes finds cut.
Update prices (find minimum delta.)

Simple Implementation:
Each bfs either augments or adds node to S in next cut.
$O(n)$ iterations per augmentation.
$O(n)$ augmentations.

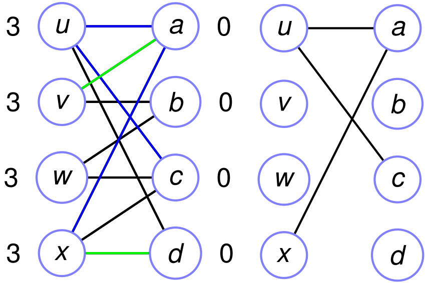
$O(n^2 m)$ time.

Example

Weight legend:
black 1, green 2, blue 3

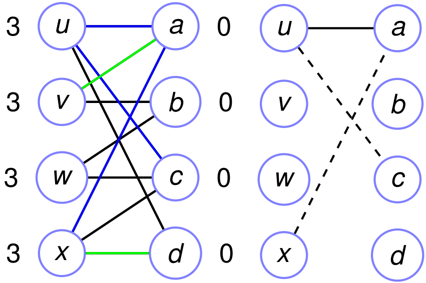


Example



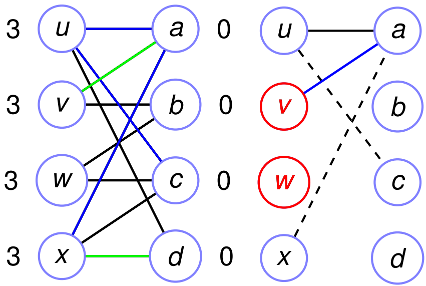
Weight legend:
black 1, green 2, blue 3
Tight edges for inital prices.

Example



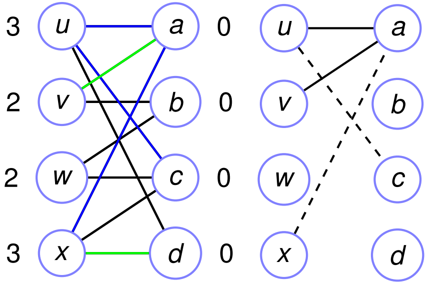
Weight legend:
black 1, green 2, blue 3
Tight edges for inital prices.
Max matching in tight edges.
dashed means matched.

Example



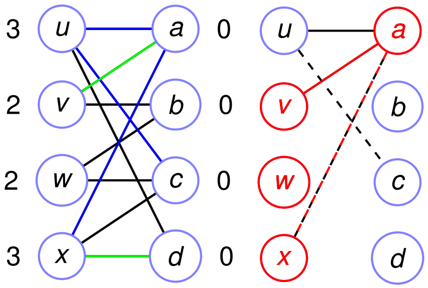
Weight legend:
black 1, green 2, blue 3
Tight edges for inital prices.
Max matching in tight edges.
dashed means matched.
No augmenting path $\rightarrow$
reachable: $S = \{u, w\}$
Blue edge on right soon
to be tight!

Example



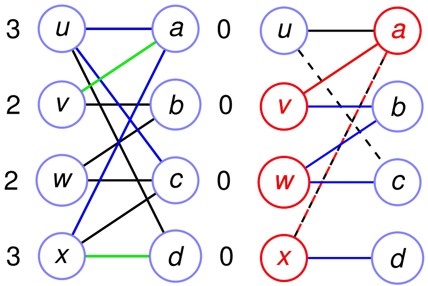
Weight legend:
black 1, green 2, blue 3
Tight edges for inital prices.
Max matching in tight edges.
dashed means matched.
No augmenting path $\rightarrow$
reachable: $S = \{u, w\}$
Blue edge on right soon
to be tight!
Adjust prices...$\delta = 1$
new tight edges.

Example



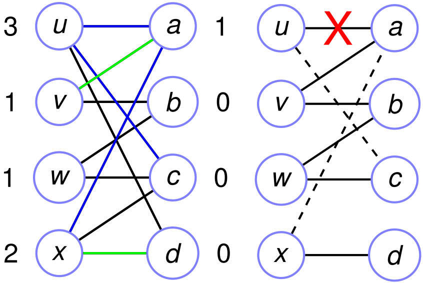
Weight legend:
black 1, green 2, blue 3
Tight edges for inital prices.
Max matching in tight edges.
dashed means matched.
No augmenting path $\rightarrow$
reachable: $S = \{u, w\}$
Blue edge on right soon
to be tight!
Adjust prices...$\delta = 1$
new tight edges.
Still no augmenting path.
Reachable $S = \{v, w, x, a\}$

Example



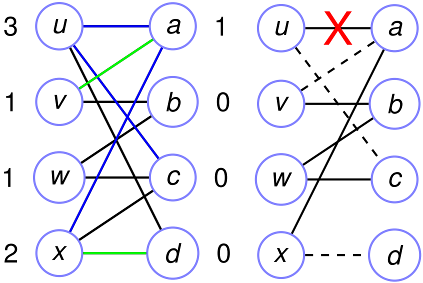
Weight legend:
black 1, green 2, blue 3
Tight edges for inital prices.
Max matching in tight edges.
dashed means matched.
No augmenting path $\rightarrow$
reachable: $S = \{u, w\}$
Blue edge on right soon
to be tight!
Adjust prices...$\delta = 1$
new tight edges.
Still no augmenting path.
Reachable $S = \{v, w, x, a\}$
Blue edges minimally non-tight.

Example



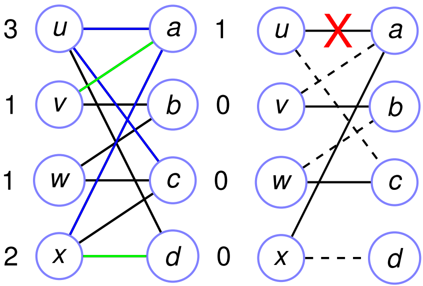
Weight legend:
black 1, green 2, blue 3
Tight edges for inital prices.
Max matching in tight edges.
dashed means matched.
No augmenting path $\rightarrow$
reachable: $S = \{u, w\}$
Blue edge on right soon
to be tight!
Adjust prices...$\delta = 1$
new tight edges.
Still no augmenting path.
Reachable $S = \{v, w, x, a\}$
Blue edges minimally non-tight.
Adjust prices.
Some more tight edges.
And X shows
a “new” nontight edge.

Example



Weight legend:
black 1, green 2, blue 3
Tight edges for inital prices.
Max matching in tight edges.
dashed means matched.
No augmenting path $\rightarrow$
reachable: $S = \{u, w\}$
Blue edge on right soon
to be tight!
Adjust prices...$\delta = 1$
new tight edges.
Still no augmenting path.
Reachable $S = \{v, w, x, a\}$
Blue edges minimally non-tight.
Adjust prices.
Some more tight edges.
And X shows
a "new" nontight edge.
..and another augmentation...

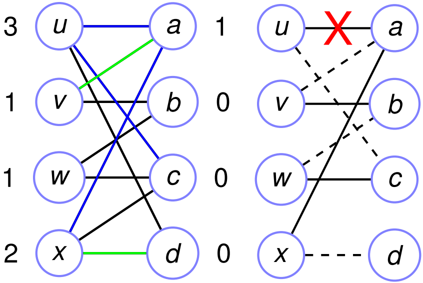
Example



Weight legend:
black 1, green 2, blue 3
Tight edges for inital prices.
Max matching in tight edges.
dashed means matched.
No augmenting path $\rightarrow$
reachable: $S = \{u, w\}$
Blue edge on right soon
to be tight!
Adjust prices...$\delta = 1$
new tight edges.
Still no augmenting path.
Reachable $S = \{v, w, x, a\}$
Blue edges minimally non-tight.
Adjust prices.
Some more tight edges.
And X shows
a "new" nontight edge.
..and another augmentation...

..and finally: a perfect matching.

Example

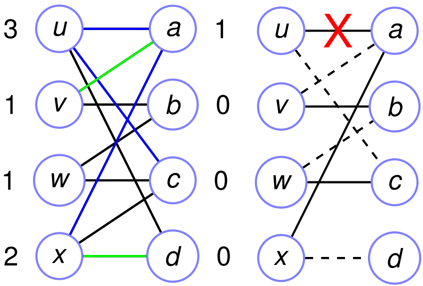


All matched edges tight.

Weight legend:
black 1, green 2, blue 3
Tight edges for inital prices.
Max matching in tight edges.
dashed means matched.
No augmenting path $\rightarrow$
reachable: $S = \{u, w\}$
Blue edge on right soon
to be tight!
Adjust prices...$\delta = 1$
new tight edges.
Still no augmenting path.
Reachable $S = \{v, w, x, a\}$
Blue edges minimally non-tight.
Adjust prices.
Some more tight edges.
And X shows
a “new” nontight edge.
..and another augmentation...

..and finally: a perfect matching.

Example

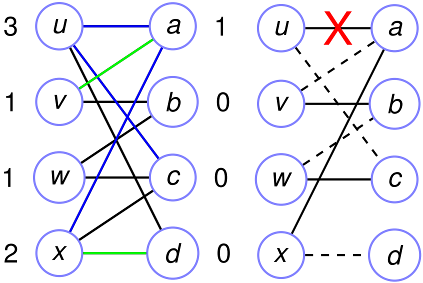


Weight legend:
black 1, green 2, blue 3
Tight edges for inital prices.
Max matching in tight edges.
dashed means matched.
No augmenting path $\rightarrow$
reachable: $S = \{u, w\}$
Blue edge on right soon
to be tight!
Adjust prices...$\delta = 1$
new tight edges.
Still no augmenting path.
Reachable $S = \{v, w, x, a\}$
Blue edges minimally non-tight.
Adjust prices.
Some more tight edges.
And X shows
a “new” nontight edge.
..and another augmentation...

..and finally: a perfect matching.

All matched edges tight.
Perfect matching.

Example

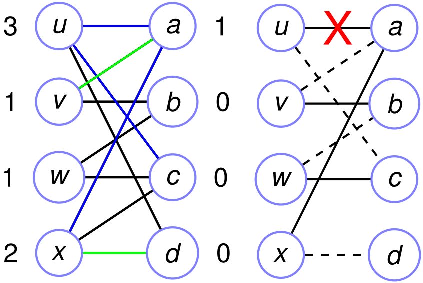


Weight legend:
black 1, green 2, blue 3
Tight edges for inital prices.
Max matching in tight edges.
dashed means matched.
No augmenting path $\rightarrow$
reachable: $S = \{u, w\}$
Blue edge on right soon
to be tight!
Adjust prices...$\delta = 1$
new tight edges.
Still no augmenting path.
Reachable $S = \{v, w, x, a\}$
Blue edges minimally non-tight.
Adjust prices.
Some more tight edges.
And X shows
a "new" nontight edge.
..and another augmentation...

..and finally: a perfect matching.

All matched edges tight.
Perfect matching. Feasible price function.

Example

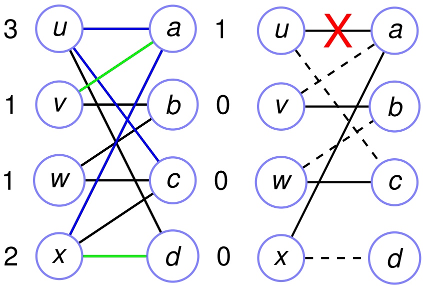


Weight legend:
black 1, green 2, blue 3
Tight edges for inital prices.
Max matching in tight edges.
dashed means matched.
No augmenting path $\rightarrow$
reachable: $S = \{u, w\}$
Blue edge on right soon
to be tight!
Adjust prices...$\delta = 1$
new tight edges.
Still no augmenting path.
Reachable $S = \{v, w, x, a\}$
Blue edges minimally non-tight.
Adjust prices.
Some more tight edges.
And X shows
a "new" nontight edge.
..and another augmentation...

..and finally: a perfect matching.

All matched edges tight.
Perfect matching. Feasible price function. Values the same.

Example

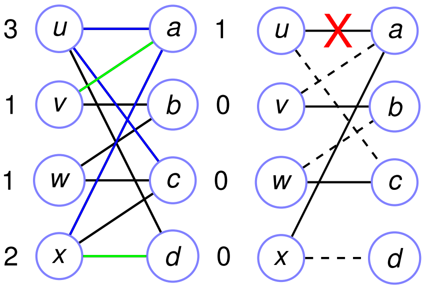


Weight legend:
black 1, green 2, blue 3
Tight edges for inital prices.
Max matching in tight edges.
dashed means matched.
No augmenting path $\rightarrow$
reachable: $S = \{u, w\}$
Blue edge on right soon
to be tight!
Adjust prices...$\delta = 1$
new tight edges.
Still no augmenting path.
Reachable $S = \{v, w, x, a\}$
Blue edges minimally non-tight.
Adjust prices.
Some more tight edges.
And X shows
a "new" nontight edge.
..and another augmentation...

..and finally: a perfect matching.

All matched edges tight.
Perfect matching. Feasible price function. Values the same. Optimal!

Example



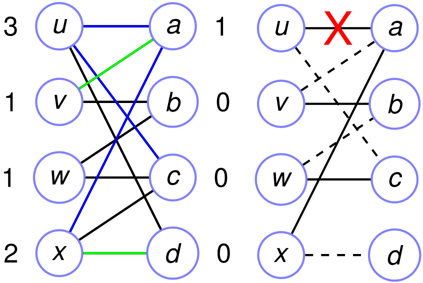
Weight legend:
black 1, green 2, blue 3
Tight edges for inital prices.
Max matching in tight edges.
dashed means matched.
No augmenting path $\rightarrow$
reachable: $S = \{u, w\}$
Blue edge on right soon
to be tight!
Adjust prices...$\delta = 1$
new tight edges.
Still no augmenting path.
Reachable $S = \{v, w, x, a\}$
Blue edges minimally non-tight.
Adjust prices.
Some more tight edges.
And X shows
a "new" nontight edge.
..and another augmentation...

..and finally: a perfect matching.

All matched edges tight.
Perfect matching. Feasible price function. Values the same. Optimal!

Notice:

Example



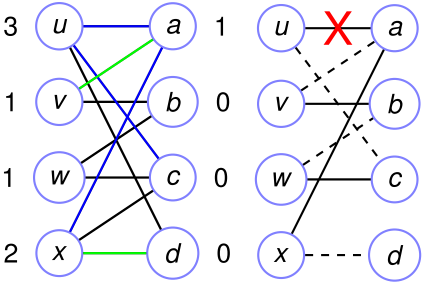
Weight legend:
black 1, green 2, blue 3
Tight edges for inital prices.
Max matching in tight edges.
dashed means matched.
No augmenting path $\rightarrow$
reachable: $S = \{u, w\}$
Blue edge on right soon
to be tight!
Adjust prices...$\delta = 1$
new tight edges.
Still no augmenting path.
Reachable $S = \{v, w, x, a\}$
Blue edges minimally non-tight.
Adjust prices.
Some more tight edges.
And X shows
a "new" nontight edge.
..and another augmentation...

..and finally: a perfect matching.

All matched edges tight.
Perfect matching. Feasible price function. Values the same. Optimal!

Notice:
no weights on the right problem.

Example



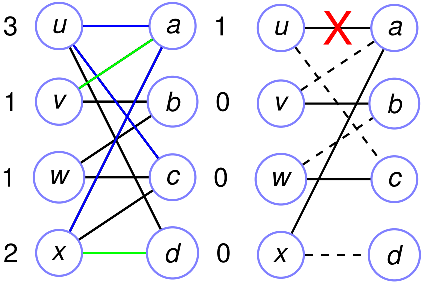
Weight legend:
black 1, green 2, blue 3
Tight edges for inital prices.
Max matching in tight edges.
dashed means matched.
No augmenting path $\rightarrow$
reachable: $S = \{u, w\}$
Blue edge on right soon
to be tight!
Adjust prices...$\delta = 1$
new tight edges.
Still no augmenting path.
Reachable $S = \{v, w, x, a\}$
Blue edges minimally non-tight.
Adjust prices.
Some more tight edges.
And X shows
a "new" nontight edge.
..and another augmentation...

..and finally: a perfect matching.

All matched edges tight.
Perfect matching. Feasible price function. Values the same. Optimal!

Notice:
no weights on the right problem.
retain previous matching through price changes.

Example



Weight legend:
black 1, green 2, blue 3
Tight edges for inital prices.
Max matching in tight edges.
dashed means matched.
No augmenting path $\rightarrow$
reachable: $S = \{u, w\}$
Blue edge on right soon
to be tight!
Adjust prices...$\delta = 1$
new tight edges.
Still no augmenting path.
Reachable $S = \{v, w, x, a\}$
Blue edges minimally non-tight.
Adjust prices.
Some more tight edges.
And X shows
a "new" nontight edge.
..and another augmentation...

..and finally: a perfect matching.

All matched edges tight.
Perfect matching. Feasible price function. Values the same. Optimal!

Notice:
no weights on the right problem.
retain previous matching through price changes.
retains edges in failed search through price changes.

Linear Program.

How?

Linear Program.

How? From lecture warmup.

Linear Program.

How? From lecture warmup.

Linear program: $\max cx, Ax \leq b, x \geq 0$
Dual: $\min y^T b, y^T A \geq c, y \geq 0$

Linear Program.

How? From lecture warmup.

Linear program: $\max cx, Ax \leq b, x \geq 0$
Dual: $\min y^T b, y^T A \geq c, y \geq 0$

Note: Dual variables correspond to primal equations and vice versa.

Linear Program.

How? From lecture warmup.

Linear program: $\max cx, Ax \leq b, x \geq 0$
Dual: $\min y^T b, y^T A \geq c, y \geq 0$

Note: Dual variables correspond to primal equations and vice versa.

Weak Duality:

Linear Program.

How? From lecture warmup.

Linear program: $\max cx, Ax \leq b, x \geq 0$
Dual: $\min y^T b, y^T A \geq c, y \geq 0$

Note: Dual variables correspond to primal equations and vice versa.

Weak Duality:

$$y^T b \geq y^T Ax \geq cx$$

Linear Program.

How? From lecture warmup.

Linear program: $\max cx, Ax \leq b, x \geq 0$
Dual: $\min y^T b, y^T A \geq c, y \geq 0$

Note: Dual variables correspond to primal equations and vice versa.

Weak Duality:

$$y^T b \geq y^T Ax \geq cx$$

First inequality from $b \geq Ax$

Linear Program.

How? From lecture warmup.

Linear program: $\max cx, Ax \leq b, x \geq 0$
Dual: $\min y^T b, y^T A \geq c, y \geq 0$

Note: Dual variables correspond to primal equations and vice versa.

Weak Duality:

$$y^T b \geq y^T Ax \geq cx$$

First inequality from $b \geq Ax$ and second from $y^A \geq c$.

Linear Program.

How? From lecture warmup.

Linear program: $\max cx, Ax \leq b, x \geq 0$
Dual: $\min y^T b, y^T A \geq c, y \geq 0$

Note: Dual variables correspond to primal equations and vice versa.

Weak Duality:

$$y^T b \geq y^T Ax \geq cx$$

First inequality from $b \geq Ax$ and second from $y^A \geq c$.

Complementary slackness:
(1) $x_j > 0 \implies a^{(j)} y = c_j$
(2) $y_i > 0 \implies a_i x = b_i$

Linear Program.

How? From lecture warmup.

Linear program: $\max cx, Ax \leq b, x \geq 0$
Dual: $\min y^T b, y^T A \geq c, y \geq 0$

Note: Dual variables correspond to primal equations and vice versa.

Weak Duality:

$$y^T b \geq y^T Ax \geq cx$$

First inequality from $b \geq Ax$ and second from $y^A \geq c$.

Complementary slackness:
(1) $x_j > 0 \implies a^{(j)} y = c_j$
(2) $y_i > 0 \implies a_i x = b_i$

What does multiplying by 0 do?

Linear Program.

How? From lecture warmup.

Linear program: $\max cx, Ax \le b, x \ge 0$
Dual: $\min y^T b, y^T A \ge c, y \ge 0$

Note: Dual variables correspond to primal equations and vice versa.

Weak Duality:

$$y^T b \ge y^T Ax \ge cx$$

First inequality from $b \ge Ax$ and second from $y^A \ge c$.

Complementary slackness:
(1) $x_j > 0 \implies a^{(j)} y = c_j$
(2) $y_i > 0 \implies a_i x = b_i$

What does multiplying by 0 do?
Zero and one. My love is won. Nothing and nothing done.

Linear Program.

How? From lecture warmup.

Linear program: $\max cx, Ax \leq b, x \geq 0$
Dual: $\min y^T b, y^T A \geq c, y \geq 0$

Note: Dual variables correspond to primal equations and vice versa.

Weak Duality:

$$y^T b \geq y^T Ax \geq cx$$

First inequality from $b \geq Ax$ and second from $y^A \geq c$.

Complementary slackness:
(1) $x_j > 0 \implies a^{(j)}y = c_j$
(2) $y_i > 0 \implies a_i x = b_i$

What does multiplying by 0 do?
Zero and one. My love is won. Nothing and nothing done.

(2) $\implies y^T b = \sum_i y_i b_i$

Linear Program.

How? From lecture warmup.

Linear program: $\max cx, Ax \leq b, x \geq 0$
Dual: $\min y^T b, y^T A \geq c, y \geq 0$

Note: Dual variables correspond to primal equations and vice versa.

Weak Duality:

$$y^T b \geq y^T Ax \geq cx$$

First inequality from $b \geq Ax$ and second from $y^A \geq c$.

Complementary slackness:
(1) $x_j > 0 \implies a^{(j)} y = c_j$
(2) $y_i > 0 \implies a_i x = b_i$

What does multiplying by 0 do?
Zero and one. My love is won. Nothing and nothing done.

(2) $\implies y^T b = \sum_i y_i b_i = \sum_i y_i (a_i x)$

Linear Program.

How? From lecture warmup.

Linear program: $\max cx, Ax \leq b, x \geq 0$
Dual: $\min y^T b, y^T A \geq c, y \geq 0$

Note: Dual variables correspond to primal equations and vice versa.

Weak Duality:

$$y^T b \geq y^T Ax \geq cx$$

First inequality from $b \geq Ax$ and second from $y^A \geq c$.

Complementary slackness:
(1) $x_j > 0 \implies a^{(j)} y = c_j$
(2) $y_i > 0 \implies a_i x = b_i$

What does multiplying by 0 do?
Zero and one. My love is won. Nothing and nothing done.

(2) $\implies y^T b = \sum_i y_i b_i = \sum_i y_i (a_i x) = y^T Ax$.

Linear Program.

How? From lecture warmup.

Linear program: $\max cx, Ax \leq b, x \geq 0$
Dual: $\min y^T b, y^T A \geq c, y \geq 0$

Note: Dual variables correspond to primal equations and vice versa.

Weak Duality:

$$y^T b \geq y^T Ax \geq cx$$

First inequality from $b \geq Ax$ and second from $y^A \geq c$.

Complementary slackness:
(1) $x_j > 0 \implies a^{(j)} y = c_j$
(2) $y_i > 0 \implies a_i x = b_i$

What does multiplying by 0 do?
Zero and one. My love is won. Nothing and nothing done.

(2) $\implies y^T b = \sum_i y_i b_i = \sum_i y_i (a_i x) = y^T Ax$.

Similarly: (1) $\implies y^T Ax = cx$.

Linear Program.

How? From lecture warmup.

Linear program: $\max cx, Ax \leq b, x \geq 0$
Dual: $\min y^T b, y^T A \geq c, y \geq 0$

Note: Dual variables correspond to primal equations and vice versa.

Weak Duality:

$$y^T b \geq y^T Ax \geq cx$$

First inequality from $b \geq Ax$ and second from $y^A \geq c$.

Complementary slackness:
(1) $x_j > 0 \implies a^{(j)}y = c_j$
(2) $y_i > 0 \implies a_i x = b_i$

What does multiplying by 0 do?
Zero and one. My love is won. Nothing and nothing done.

(2) $\implies y^T b = \sum_i y_i b_i = \sum_i y_i (a_i x) = y^T Ax$.

Similarly: (1) $\implies y^T Ax = cx$.

Complementary slackness conditions imply optimality.

nnnn

Perfect Matching

Linear program: $\max \sum_e w_e x_e$, $\forall v : \sum_{e=(u,v)} x_e \leq 1$, $x_e \geq 0$

Perfect Matching

Linear program: $\max \sum_e w_e x_e$, $\forall v : \sum_{e=(u,v)} x_e \leq 1$, $x_e \geq 0$

Dual: $\min \sum_v p_v$, $\forall e = (u,v) : p_u + p_v \geq w_e$, $p_u \geq 0$.

Perfect Matching

Linear program: $\max \sum_e w_e x_e$, $\forall v : \sum_{e=(u,v)} x_e \leq 1$, $x_e \geq 0$

Dual: $\min \sum_v p_v$, $\forall e = (u,v) : p_u + p_v \geq w_e$, $p_u \geq 0$.

Perfect Matching

Linear program: $\max \sum_e w_e x_e$, $\forall v : \sum_{e=(u,v)} x_e \leq 1$, $x_e \geq 0$

Dual: $\min \sum_v p_v$, $\forall e = (u,v) : p_u + p_v \geq w_e$, $p_u \geq 0$.

In this case:

Perfect Matching

Linear program: $\max \sum_e w_e x_e$, $\forall v : \sum_{e=(u,v)} x_e \leq 1$, $x_e \geq 0$

Dual: $\min \sum_v p_v$, $\forall e = (u,v) : p_u + p_v \geq w_e$, $p_u \geq 0$.

In this case:

Dual feasible at start: $p_u \geq \max_{e=(u,v)} w_e$

Perfect Matching

Linear program: $\max \sum_e w_e x_e$, $\forall v : \sum_{e=(u,v)} x_e \leq 1$, $x_e \geq 0$

Dual: $\min \sum_v p_v$, $\forall e = (u,v) : p_u + p_v \geq w_e$, $p_u \geq 0$.

In this case:

- Dual feasible at start: $p_u \geq \max_{e=(u,v)} w_e$
  - Maintain feasibility: adjust prices by δ.

Perfect Matching

Linear program: $\max \sum_e w_e x_e$, $\forall v : \sum_{e=(u,v)} x_e \leq 1$, $x_e \geq 0$

Dual: $\min \sum_v p_v$, $\forall e = (u,v) : p_u + p_v \geq w_e$, $p_u \geq 0$.

In this case:

- Dual feasible at start: $p_u \geq \max_{e=(u,v)} w_e$
  - Maintain feasibility: adjust prices by δ.
- Maintain Primal feasibility.

Perfect Matching

Linear program: $\max \sum_e w_e x_e$, $\forall v : \sum_{e=(u,v)} x_e \leq 1$, $x_e \geq 0$

Dual: $\min \sum_v p_v$, $\forall e = (u,v) : p_u + p_v \geq w_e$, $p_u \geq 0$.

In this case:

- Dual feasible at start: $p_u \geq \max_{e=(u,v)} w_e$
  - Maintain feasibility: adjust prices by δ.
- Maintain Primal feasibility.
- Maintain complementary slackness (2).

Perfect Matching

Linear program: $\max \sum_e w_e x_e$, $\forall v : \sum_{e=(u,v)} x_e \leq 1$, $x_e \geq 0$

Dual: $\min \sum_v p_v$, $\forall e = (u,v) : p_u + p_v \geq w_e$, $p_u \geq 0$.

In this case:

- Dual feasible at start: $p_u \geq \max_{e=(u,v)} w_e$
  - Maintain feasibility: adjust prices by δ.
- Maintain Primal feasibility.
- Maintain complementary slackness (2).
  - $x_e > 0$ only if $p_u + p_v = w_e$.

Perfect Matching

Linear program: $\max \sum_e w_e x_e$, $\forall v : \sum_{e=(u,v)} x_e \leq 1$, $x_e \geq 0$

Dual: $\min \sum_v p_v$, $\forall e = (u,v) : p_u + p_v \geq w_e$, $p_u \geq 0$.

In this case:
- Dual feasible at start: $p_u \geq \max_{e=(u,v)} w_e$
  - Maintain feasibility: adjust prices by δ.
- Maintain Primal feasibility.
- Maintain complementary slackness (2).
  - $x_e > 0$ only if $p_u + p_v = w_e$.
  - Eventually match all vertices.

Perfect Matching

Linear program: $\max \sum_e w_e x_e$, $\forall v : \sum_{e=(u,v)} x_e \leq 1$, $x_e \geq 0$

Dual: $\min \sum_v p_v$, $\forall e = (u,v) : p_u + p_v \geq w_e$, $p_u \geq 0$.

In this case:
- Dual feasible at start: $p_u \geq \max_{e=(u,v)} w_e$
  - Maintain feasibility: adjust prices by δ.
- Maintain Primal feasibility.
- Maintain complementary slackness (2).
  - $x_e > 0$ only if $p_u + p_v = w_e$.
  - Eventually match all vertices.

Perfect Matching

Linear program: $\max \sum_e w_e x_e$, $\forall v : \sum_{e=(u,v)} x_e \leq 1$, $x_e \geq 0$

Dual: $\min \sum_v p_v$, $\forall e = (u,v) : p_u + p_v \geq w_e$, $p_u \geq 0$.

In this case:
- Dual feasible at start: $p_u \geq \max_{e=(u,v)} w_e$
  - Maintain feasibility: adjust prices by δ.
- Maintain Primal feasibility.
- Maintain complementary slackness (2).
  - $x_e > 0$ only if $p_u + p_v = w_e$.
  - Eventually match all vertices.

The Engine that pulls the train:

Perfect Matching

Linear program: $\max \sum_e w_e x_e$, $\forall v : \sum_{e=(u,v)} x_e \leq 1$, $x_e \geq 0$

Dual: $\min \sum_v p_v$, $\forall e = (u,v) : p_u + p_v \geq w_e$, $p_u \geq 0$.

In this case:

- Dual feasible at start: $p_u \geq \max_{e=(u,v)} w_e$
  - Maintain feasibility: adjust prices by δ.
- Maintain Primal feasibility.
- Maintain complementary slackness (2).
  - $x_e > 0$ only if $p_u + p_v = w_e$.
  - Eventually match all vertices.

The Engine that pulls the train: Find a path of tight edges.

Perfect Matching

Linear program: $\max \sum_e w_e x_e$, $\forall v : \sum_{e=(u,v)} x_e \leq 1$, $x_e \geq 0$

Dual: $\min \sum_v p_v$, $\forall e = (u,v) : p_u + p_v \geq w_e$, $p_u \geq 0$.

In this case:

Dual feasible at start: $p_u \geq \max_{e=(u,v)} w_e$

Maintain feasibility: adjust prices by δ.

Maintain Primal feasibility.

Maintain complementary slackness (2).

$x_e > 0$ only if $p_u + p_v = w_e$.

Eventually match all vertices.

The Engine that pulls the train: Find a path of tight edges.

Perfect Matching

Linear program: $\max \sum_e w_e x_e$, $\forall v : \sum_{e=(u,v)} x_e \leq 1$, $x_e \geq 0$

Dual: $\min \sum_v p_v$, $\forall e = (u,v) : p_u + p_v \geq w_e$, $p_u \geq 0$.

In this case:
  Dual feasible at start: $p_u \geq \max_{e=(u,v)} w_e$
    Maintain feasibility: adjust prices by δ.
  Maintain Primal feasibility.
  Maintain complementary slackness (2).
    $x_e > 0$ only if $p_u + p_v = w_e$.
    Eventually match all vertices.

The Engine that pulls the train: Find a path of tight edges.

Complementary slackness (1): Terminate when perfect matching.

Perfect Matching

Linear program: $\max \sum_e w_e x_e$, $\forall v : \sum_{e=(u,v)} x_e \leq 1$, $x_e \geq 0$

Dual: $\min \sum_v p_v$, $\forall e = (u,v) : p_u + p_v \geq w_e$, $p_u \geq 0$.

In this case:

- Dual feasible at start: $p_u \geq \max_{e=(u,v)} w_e$
  - Maintain feasibility: adjust prices by δ.
- Maintain Primal feasibility.
- Maintain complementary slackness (2).
  - $x_e > 0$ only if $p_u + p_v = w_e$.
  - Eventually match all vertices.

The Engine that pulls the train: Find a path of tight edges.

Complementary slackness (1): Terminate when perfect matching.

$\forall v : \sum_{e=(u,v)} x_e$.

Perfect Matching

Linear program: $\max \sum_e w_e x_e$, $\forall v : \sum_{e=(u,v)} x_e \leq 1$, $x_e \geq 0$

Dual: $\min \sum_v p_v$, $\forall e = (u,v) : p_u + p_v \geq w_e$, $p_u \geq 0$.

In this case:
  Dual feasible at start: $p_u \geq \max_{e=(u,v)} w_e$
    Maintain feasibility: adjust prices by δ.
  Maintain Primal feasibility.
  Maintain complementary slackness (2).
    $x_e > 0$ only if $p_u + p_v = w_e$.
    Eventually match all vertices.

The Engine that pulls the train: Find a path of tight edges.

Complementary slackness (1): Terminate when perfect matching.
    $\forall v : \sum_{e=(u,v)} x_e$. So any p_u can be non-zero.

Perfect Matching

Linear program: $\max \sum_e w_e x_e$, $\forall v : \sum_{e=(u,v)} x_e \leq 1$, $x_e \geq 0$

Dual: $\min \sum_v p_v$, $\forall e = (u,v) : p_u + p_v \geq w_e$, $p_u \geq 0$.

In this case:

Dual feasible at start: $p_u \geq \max_{e=(u,v)} w_e$

Maintain feasibility: adjust prices by δ.

Maintain Primal feasibility.

Maintain complementary slackness (2).

$x_e > 0$ only if $p_u + p_v = w_e$.

Eventually match all vertices.

The Engine that pulls the train: Find a path of tight edges.

Complementary slackness (1): Terminate when perfect matching.

$\forall v : \sum_{e=(u,v)} x_e$. So any p_u can be non-zero.

The "play" indicates game playing.

Perfect Matching

Linear program: $\max \sum_e w_e x_e$, $\forall v : \sum_{e=(u,v)} x_e \leq 1$, $x_e \geq 0$

Dual: $\min \sum_v p_v$, $\forall e = (u,v) : p_u + p_v \geq w_e$, $p_u \geq 0$.

In this case:

Dual feasible at start: $p_u \geq \max_{e=(u,v)} w_e$

Maintain feasibility: adjust prices by δ.

Maintain Primal feasibility.

Maintain complementary slackness (2).

$x_e > 0$ only if $p_u + p_v = w_e$.

Eventually match all vertices.

The Engine that pulls the train: Find a path of tight edges.

Complementary slackness (1): Terminate when perfect matching.

$\forall v : \sum_{e=(u,v)} x_e$. So any p_u can be non-zero.

The “play” indicates game playing.

Algorithm plays lower bound against upper bound.

Perfect Matching

Linear program: $\max \sum_e w_e x_e$, $\forall v : \sum_{e=(u,v)} x_e \leq 1$, $x_e \geq 0$

Dual: $\min \sum_v p_v$, $\forall e = (u,v) : p_u + p_v \geq w_e$, $p_u \geq 0$.

In this case:

Dual feasible at start: $p_u \geq \max_{e=(u,v)} w_e$

Maintain feasibility: adjust prices by δ.

Maintain Primal feasibility.

Maintain complementary slackness (2).

$x_e > 0$ only if $p_u + p_v = w_e$.

Eventually match all vertices.

The Engine that pulls the train: Find a path of tight edges.

Complementary slackness (1): Terminate when perfect matching.

$\forall v : \sum_{e=(u,v)} x_e$. So any p_u can be non-zero.

The “play” indicates game playing.

Algorithm plays lower bound against upper bound.

Two person games: von Neuman.

Perfect Matching

Linear program: $\max \sum_e w_e x_e$, $\forall v : \sum_{e=(u,v)} x_e \leq 1$, $x_e \geq 0$

Dual: $\min \sum_v p_v$, $\forall e = (u,v) : p_u + p_v \geq w_e$, $p_u \geq 0$.

In this case:
 Dual feasible at start: $p_u \geq \max_{e=(u,v)} w_e$
  Maintain feasibility: adjust prices by δ.
 Maintain Primal feasibility.
 Maintain complementary slackness (2).
  $x_e > 0$ only if $p_u + p_v = w_e$.
  Eventually match all vertices.

The Engine that pulls the train: Find a path of tight edges.

Complementary slackness (1): Terminate when perfect matching.
 $\forall v : \sum_{e=(u,v)} x_e$. So any p_u can be non-zero.

The "play" indicates game playing.
Algorithm plays lower bound against upper bound.
Two person games: von Neuman. Equilibrium: Nash.

Perfect Matching

Linear program: $\max \sum_e w_e x_e$, $\forall v : \sum_{e=(u,v)} x_e \leq 1$, $x_e \geq 0$

Dual: $\min \sum_v p_v$, $\forall e = (u,v) : p_u + p_v \geq w_e$, $p_u \geq 0$.

In this case:
- Dual feasible at start: $p_u \geq \max_{e=(u,v)} w_e$
  - Maintain feasibility: adjust prices by δ.
- Maintain Primal feasibility.
- Maintain complementary slackness (2).
  - $x_e > 0$ only if $p_u + p_v = w_e$.
  - Eventually match all vertices.

The Engine that pulls the train: Find a path of tight edges.

Complementary slackness (1): Terminate when perfect matching.
$\forall v : \sum_{e=(u,v)} x_e$. So any p_u can be non-zero.

The "play" indicates game playing.
Algorithm plays lower bound against upper bound.
Two person games: von Neuman. Equilibrium: Nash.

Is the path fundamental?

Perfect Matching

Linear program: $\max \sum_e w_e x_e$, $\forall v : \sum_{e=(u,v)} x_e \leq 1$, $x_e \geq 0$

Dual: $\min \sum_v p_v$, $\forall e = (u,v) : p_u + p_v \geq w_e$, $p_u \geq 0$.

In this case:

Dual feasible at start: $p_u \geq \max_{e=(u,v)} w_e$

Maintain feasibility: adjust prices by δ.

Maintain Primal feasibility.

Maintain complementary slackness (2).

$x_e > 0$ only if $p_u + p_v = w_e$.

Eventually match all vertices.

The Engine that pulls the train: Find a path of tight edges.

Complementary slackness (1): Terminate when perfect matching.

$\forall v : \sum_{e=(u,v)} x_e$. So any p_u can be non-zero.

The "play" indicates game playing.
Algorithm plays lower bound against upper bound.
Two person games: von Neuman. Equilibrium: Nash.

Is the path fundamental?

Are things as easy or as hard as $0, 1, 2, \ldots$...?

Perfect Matching

Linear program: $\max \sum_e w_e x_e$, $\forall v : \sum_{e=(u,v)} x_e \leq 1$, $x_e \geq 0$

Dual: $\min \sum_v p_v$, $\forall e = (u,v) : p_u + p_v \geq w_e$, $p_u \geq 0$.

In this case:
Dual feasible at start: $p_u \geq \max_{e=(u,v)} w_e$
Maintain feasibility: adjust prices by δ.
Maintain Primal feasibility.
Maintain complementary slackness (2).
$x_e > 0$ only if $p_u + p_v = w_e$.
Eventually match all vertices.

The Engine that pulls the train: Find a path of tight edges.

Complementary slackness (1): Terminate when perfect matching.
$\forall v : \sum_{e=(u,v)} x_e$. So any p_u can be non-zero.

The "play" indicates game playing.
Algorithm plays lower bound against upper bound.
Two person games: von Neuman. Equilibrium: Nash.

Is the path fundamental?
Are things as easy or as hard as $0, 1, 2, \ldots$...?

...see you on Tuesday