Minimum cost bipartite matching

Giovanni Righini

Università degli Studi di Milano

Definitions

Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ a matching is an edge subset $\mathcal{M} \subseteq \mathcal{E}$ such that it is not incident to any vertex more than once.

A matching is maximal if and only if there is no other matching containing it.

A matching has maximum cardinality if and only if it contains the maximum number of edges of $\mathcal{E}$.

A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is bipartite when $\mathcal{V}$ is formed by two disjoint subsets $\mathcal{S}$ and $\mathcal{T}$ and all edges $[i, j] \in \mathcal{E}$ have an endpoint in $\mathcal{S}$ and the other in $\mathcal{T}$.

A bipartite graph $\mathcal{G} = (\mathcal{S}, \mathcal{T}, \mathcal{E})$ is complete when there are all possible edges between $\mathcal{S}$ and $\mathcal{T}$, i.e. $\mathcal{E} = \mathcal{S} \times \mathcal{T}$.

The problem

We consider the problem of finding the *matching* of maximum cardinality and minimum cost between two vertex subsets $\mathcal{S}$ and $\mathcal{T}$ defining a weighted bipartite graph.

Data:

- a bipartite graph $\mathcal{G} = (\mathcal{S}, \mathcal{T}, \mathcal{E})$,
- a cost function $c : \mathcal{E} \mapsto \Re$.

Problem (Minimum Cost Bipartite Matching Problem). Find a minimum cost *matching* between $\mathcal{S}$ e $\mathcal{T}$ among all those with maximum cardinality.

Graph pre-processing

We assume:

- the two partitions are balanced: $|\mathcal{S}| = |\mathcal{T}| = n$
- the graph is complete: $\mathcal{E} = \mathcal{S} \times \mathcal{T}$.

Observation. If these conditions do not hold, it is always possible to reformulate the problem in an equivalent way on a complete balanced bipartite graph.

Graph pre-processing

Balancing the graph.

If the given bipartite graph is not balanced, we insert dummy vertices in the partition of smaller cardinality, to make it balanced.

No matching is affected by this operation.

Completing the graph.

If the given graph is not complete, we insert dummy edges with a very large cost ("Big-*M*") to make it complete.

Infeasible matchings are now feasible but they have very large cost.
Maximum cardinality feasible matchings in the original graph correspond to matchings with the smallest number of dummy edges in the new graph.
Among them, optimality only depends on the costs of the original edges.

The reformulated problem

After pre-processing, we can reformulate the problem as follows.

Minimum Cost Bipartite Matching Problem (reformulated). Find a minimum cost complete *matching* between the two vertex subsets of a given weighted bipartite graph.

Every solution is represented by an *assignment matrix* where $\mathcal{S}$ is the row set and $\mathcal{T}$ is the column set.

An assignment matrix is a binary square matrix with exactly one entry equal to 1 for each row and each column.

Linear Assignment Problem. Find a minimum cost assignment in a given square matrix.

Reformulation as a flow problem

A “trivial” way of solving the problem is to transform it into a min cost max flow problem.

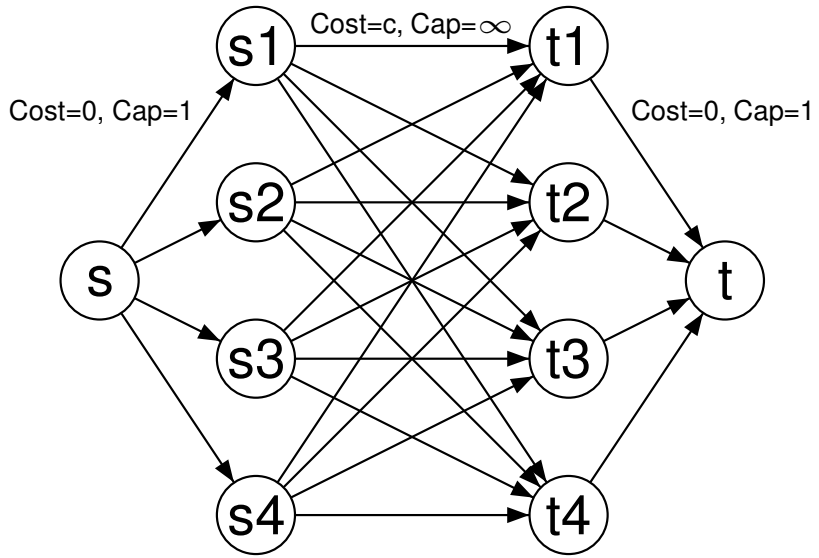


Figure: Network flow reformulation.

A mathematical model (ILP)

We use a binary variable x_{ij} for each edge $[i, j] \in \mathcal{E}$, to indicate whether the edge is in the solution or not.

$$\text{minimize } z = \sum_{i \in \mathcal{S}} \sum_{j \in \mathcal{T}} c_{ij} x_{ij}$$

$$\text{s.t.} \sum_{j \in \mathcal{T}} x_{ij} = 1 \qquad \forall i \in \mathcal{S}$$

$$\sum_{i \in \mathcal{S}} x_{ij} = 1 \qquad \forall j \in \mathcal{T}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i \in \mathcal{S},\ \forall j \in \mathcal{T}.$$

Does it have the integrality property?

A mathematical model (ILP)

We use a binary variable x_{ij} for each edge $[i,j] \in \mathcal{E}$, to indicate whether the edge is in the solution or not.

$$\begin{aligned}
\text{minimize } z &= \sum_{i \in \mathcal{S}} \sum_{j \in \mathcal{T}} c_{ij} x_{ij} \\
\text{s.t. } & \sum_{j \in \mathcal{T}} x_{ij} = 1 && \forall i \in \mathcal{S} \\
& \sum_{i \in \mathcal{S}} x_{ij} = 1 && \forall j \in \mathcal{T} \\
& x_{ij} \in \{0,1\} && \forall i \in \mathcal{S},\ \forall j \in \mathcal{T}.
\end{aligned}$$

- The constraint matrix is *totally unimodular*.
- The right-hand sides are integer numbers.

Hence all solutions of the linear relaxation have integer coordinates.

A mathematical model (LP)

Relaxing the integrality constraints, the following model is obtained:

$$\begin{aligned}
\text{minimize } z &= \sum_{i \in \mathcal{S}} \sum_{j \in \mathcal{T}} c_{ij} x_{ij} \\
\text{s.t. } & \sum_{j \in \mathcal{T}} x_{ij} = 1 && \forall i \in \mathcal{S} \\
& \sum_{i \in \mathcal{S}} x_{ij} = 1 && \forall j \in \mathcal{T} \\
& 0 \leq x_{ij} \leq 1 && \forall i \in \mathcal{S},\ \forall j \in \mathcal{T}.
\end{aligned}$$

Upper bounds $x_{ij} \leq 1$ are redundant because assignment constraints and non-negativity constraints imply them.

A mathematical model (LP)

So we are left with the following model:

$$\text{minimize } z = \sum_{i \in \mathcal{S}} \sum_{j \in \mathcal{T}} c_{ij} x_{ij}$$

$$\text{s.t.} \sum_{j \in \mathcal{T}} x_{ij} = 1 \qquad \forall i \in \mathcal{S}$$

$$\sum_{i \in \mathcal{S}} x_{ij} = 1 \qquad \forall j \in \mathcal{T}$$

$$x_{ij} \geq 0 \qquad \forall i \in \mathcal{S},\ \forall j \in \mathcal{T}.$$

This is a LP problem, hence it has a dual problem and it forms a *strong dual pair* with it.

The primal-dual pair

Primal problem:

$$\text{minimize } z = \sum_{i \in \mathcal{S}} \sum_{j \in \mathcal{T}} c_{ij} x_{ij}$$

$$\text{s.t.} \sum_{j \in \mathcal{T}} x_{ij} = 1 \qquad \forall i \in \mathcal{S}$$

$$\sum_{i \in \mathcal{S}} x_{ij} = 1 \qquad \forall j \in \mathcal{T}$$

$$x_{ij} \geq 0 \qquad \forall i \in \mathcal{S},\ \forall j \in \mathcal{T}.$$

Write the dual.

The primal-dual pair

Primal problem:

$$\text{minimize } z = \sum_{i \in \mathcal{S}} \sum_{j \in \mathcal{T}} c_{ij} x_{ij}$$

$$\text{s.t.} \sum_{j \in \mathcal{T}} x_{ij} = 1 \qquad \forall i \in \mathcal{S}$$

$$\sum_{i \in \mathcal{S}} x_{ij} = 1 \qquad \forall j \in \mathcal{T}$$

$$x_{ij} \geq 0 \qquad \forall i \in \mathcal{S},\ \forall j \in \mathcal{T}.$$

Dual problem:

$$\text{maximize } w = \sum_{i \in \mathcal{S}} u_i + \sum_{j \in \mathcal{T}} v_j$$

$$\text{s.t. } u_i + v_j \leq c_{ij} \qquad \forall i \in \mathcal{S}\ \forall j \in \mathcal{T}.$$

The dual problem

$$\text{maximize } w = \sum_{i \in \mathcal{S}} u_i + \sum_{j \in \mathcal{T}} v_j$$

$$\text{s.t. } u_i + v_j \leq c_{ij} \qquad \forall i \in \mathcal{S}\ \forall j \in \mathcal{T}.$$

Dual variables u and v are unrestricted in sign.

The dual slack variables (primal reduced costs) are:

$$\overline{c}_{ij} = c_{ij} - u_i - v_j.$$

For optimality, complementary slackness conditions impose that:

$$\overline{c}_{ij} x_{ij} = 0 \quad \forall i \in \mathcal{S}\ \forall j \in \mathcal{T}.$$

Partial assignments and primal feasibility

We call partial assignment an assignment satisfying

$$\sum_{j \in \mathcal{T}} x_{ij} \leq 1 \quad \forall i \in \mathcal{S}$$

$$\sum_{i \in \mathcal{S}} x_{ij} \leq 1 \quad \forall j \in \mathcal{T}.$$

Primal infeasibility is measured by the number of missing assignments.

CSCs impose that in each primal/dual pair of base solutions we may have $x_{ij} > 0$ only for edges $[i, j]$ for which $\overline{c}_{ij} = 0$.

We call admissible cells of the assignment matrix those where $\overline{c}_{ij} = 0$.

Primal-dual algorithms

A primal-dual algorithm solves linear programming problems exploiting duality theory and in particular the CSCs.

The algorithm is initialized with a dual feasible solution and a corresponding primal solution (in general, infeasible) satisfying the CSCs.

After every iteration the algorithm keeps a pair of primal (infeasible) and dual (feasible) solutions, satisfying the CSCs.

The algorithm alternates two types of iterations, and it monotonically decreases primal infeasibility until it achieves primal feasibility.

- Primal iteration: keeping the current dual feasible solution fixed, find a primal solution minimizing primal infeasibility among those satisfying the CSCs;
- Dual iteration: keeping the current primal solution fixed, modify the dual solution, keeping it feasible and the CSCs satisfied.

Hungarian algorithm (Kuhn 1955)

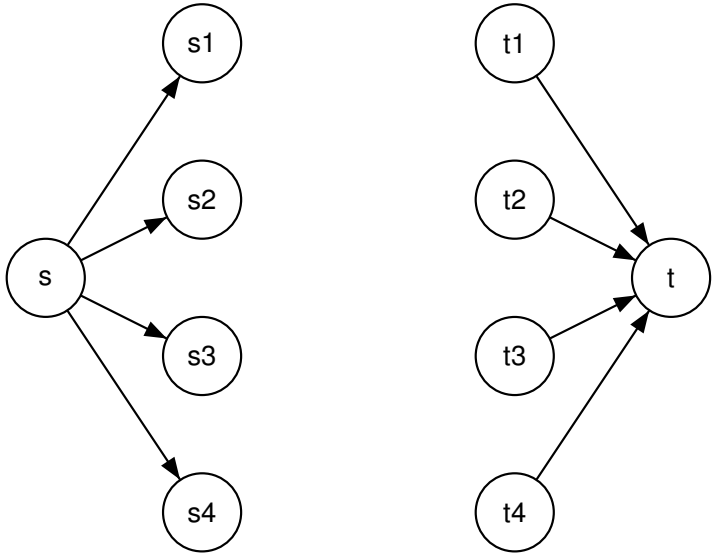
The hungarian algorithm is a primal-dual algorithm.

- Primal iteration: keeping u_i and v_j fixed, and hence $\overline{c}_{ij}$ fixed, determine x maximizing the number of assignments ($x_{ij} = 1$), using only admissible cells;
- Dual iteration: update u_i and v_j, keeping $\overline{c}_{ij} = 0$ where $x_{ij} = 1$ and making some inadmissible cells admissible.

Hungarian algorithm: pseudo-code

Begin
Step 1: Dual initialization of u and v;
Step 2: Primal initialization of x;
while (x is infeasible) **do**
  Step 3.1: Path initialization
  $Path$:=nil;
  while ($Path = nil$) **do**
    while ($Path = nil$) $\wedge$ ($L \neq \emptyset$) **do**
      Step 3.2: Labeling procedure
    end while
    if $Path = nil$ **then**
      Step 4: Dual iteration: Modify u and v;
    end if
  end while
  Step 5: Primal iteration: Modify x;
end while
End

Hungarian algorithm: visualization



$\overline{c}$	t1	t2	t3	t4	u
s1	15	22	13	4	0
s2	12	21	15	7	0
s3	16	20	22	6	0
s4	6	11	8	5	0
v	0	0	0	0	**0**

The values in the bottom-right corners are:

$$z = \sum_{[i,j] \in \mathcal{E}} c_{ij} x_{ij}.$$

$$w = \sum_{i \in \mathcal{S}} u_i + \sum_{j \in \mathcal{T}} v_j.$$

Initially: $u = v = 0$, $\overline{c} = c$, $w = 0$.

	t1	t2	t3	t4	*Mate*
s1	15	22	13	4	nil
s2	12	21	15	7	nil
s3	16	20	22	6	nil
s4	6	11	8	5	nil
Mate	nil	nil	nil	nil	**0**

Initially: $x = 0$, $z = 0$, $Card = 0$.

Step 1: Dual initialization

This can be done with a dual ascent procedure.

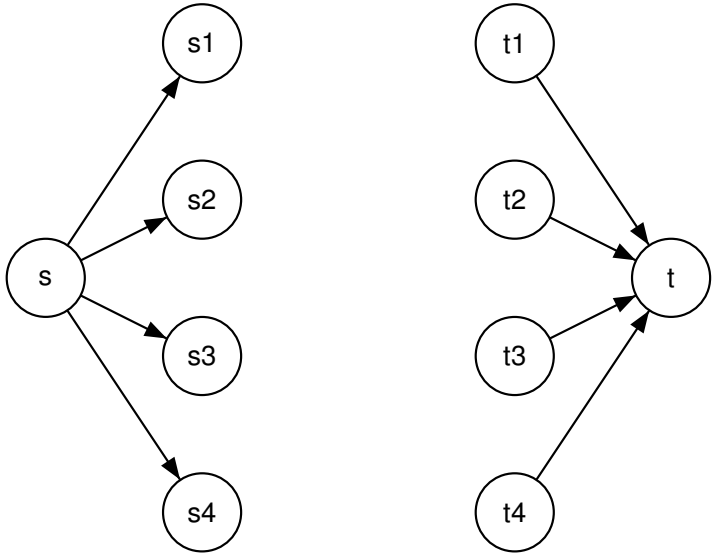
Begin Step 1
for $i \in \mathcal{S}$ **do**
  $u_i := \min_{j \in \mathcal{T}} \{c_{ij}\}$;
end for
for $j \in \mathcal{T}$ **do**
  $v_j := \min_{i \in \mathcal{S}} \{c_{ij} - u_i\}$;
end for
End Step 1

The dual variables are raised one at a time from 0 up to the minimum value that makes a dual constraint active.

This guarantees that the dual solution remains feasible.

Complexity: $O(n^2)$**.**

Visualization of Step 1

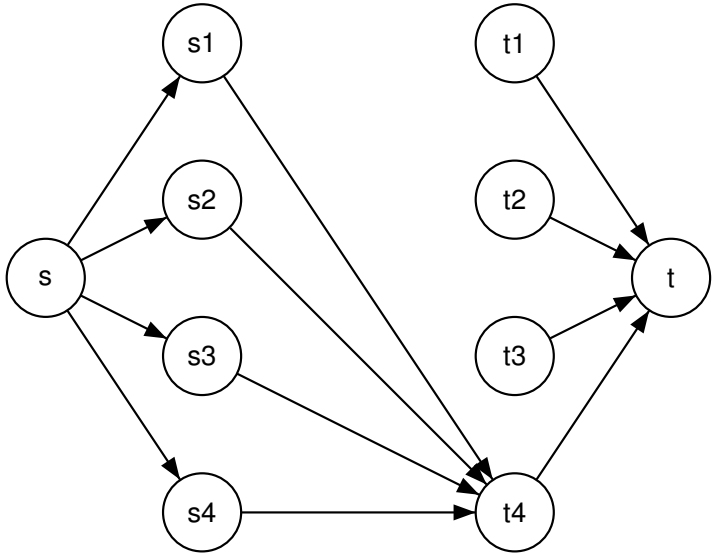


$\overline{c}$	t1	t2	t3	t4	u
s1	15	22	13	**4**	0
s2	12	21	15	**7**	0
s3	16	20	22	**6**	0
s4	6	11	8	**5**	0
v	0	0	0	0	**0**

	t1	t2	t3	t4	*Mate*
s1	15	22	13	4	nil
s2	12	21	15	7	nil
s3	16	20	22	6	nil
s4	6	11	8	5	nil
Mate	nil	nil	nil	nil	**0**

$x = 0$, $Card = 0$.

Visualization of Step 1

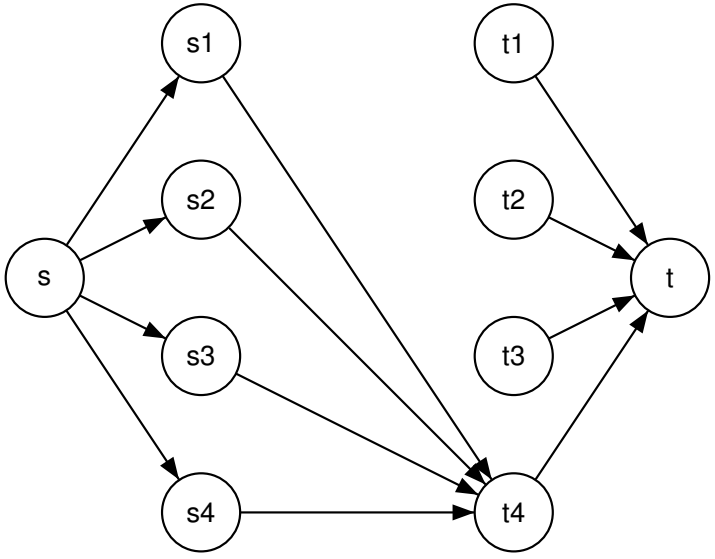


$\overline{c}$	t1	t2	t3	t4	u
s1	11	18	9	0	4
s2	5	14	8	0	7
s3	10	14	16	0	6
s4	1	6	3	0	5
v	0	0	0	0	**22**

	t1	t2	t3	t4	*Mate*
s1	15	22	13	4	nil
s2	12	21	15	7	nil
s3	16	20	22	6	nil
s4	6	11	8	5	nil
Mate	nil	nil	nil	nil	**0**

$x = 0$, $Card = 0$.

Visualization of Step 1

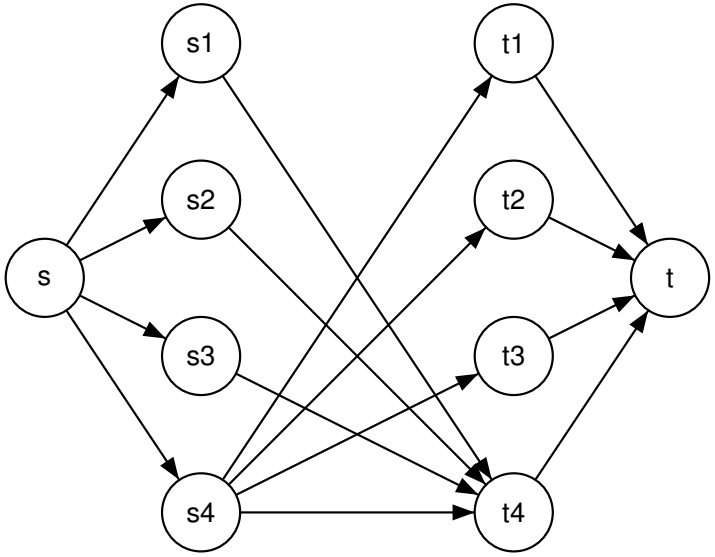


$\overline{c}$	t1	t2	t3	t4	u
s1	11	18	9	**0**	4
s2	5	14	8	0	7
s3	10	14	16	0	6
s4	**1**	**6**	**3**	0	5
v	0	0	0	0	**22**

	t1	t2	t3	t4	*Mate*
s1	15	22	13	4	nil
s2	12	21	15	7	nil
s3	16	20	22	6	nil
s4	6	11	8	5	nil
Mate	nil	nil	nil	nil	**0**

$x = 0$, $Card = 0$.

Visualization of Step 1



$\overline{c}$	t1	t2	t3	t4	u
s1	10	12	6	0	4
s2	4	8	5	0	7
s3	9	8	13	0	6
s4	0	0	0	0	5
v	1	6	3	0	**32**

	t1	t2	t3	t4	*Mate*
s1	15	22	13	4	nil
s2	12	21	15	7	nil
s3	16	20	22	6	nil
s4	6	11	8	5	nil
Mate	nil	nil	nil	nil	**0**

$x = 0$, $Card = 0$.

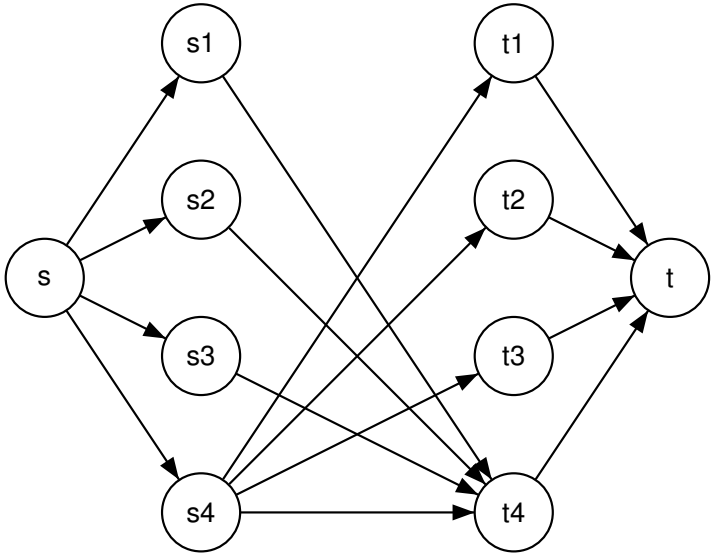
Step 2: Primal initialization

Begin Step 2
for $k \in \mathcal{S} \cup \mathcal{T}$ **do**
  $Mate(k)$:=*nil*;
end for
$Card := 0$;
while $(\exists [i,j] : (c_{ij} - u(i) - v(j) = 0) \wedge (Mate(i) = nil) \wedge (Mate(j) = nil))$
do
  $x_{ij} := 1$;
  $Card := Card + 1$;
  $Mate(i) := j$; $Mate(j) := i$;
end while
End Step 2

A maximal partial matching is computed, using only admissible cells. This requires scanning a square $(n \times n)$ matrix.

Complexity: $O(n^2)$**.**

Visualization of Step 2



$\overline{c}$	t1	t2	t3	t4	u
s1	10	12	6	**0**	4
s2	4	8	5	**0**	7
s3	9	8	13	**0**	6
s4	**0**	**0**	**0**	**0**	5
v	1	6	3	0	**32**

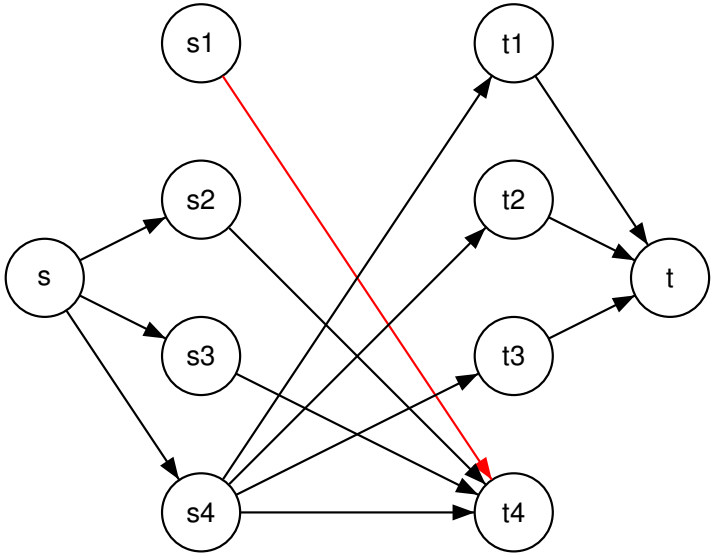
There are 7 admissible cells.

	t1	t2	t3	t4	*Mate*
s1	15	22	13	4	nil
s2	12	21	15	7	nil
s3	16	20	22	6	nil
s4	6	11	8	5	nil
Mate	nil	nil	nil	nil	**0**

Scanning them in lexicographic order by rows and columns, edge $[1, 4]$ is chosen first.

$x = 0$, *Card* $= 0$.

Visualization of Step 2



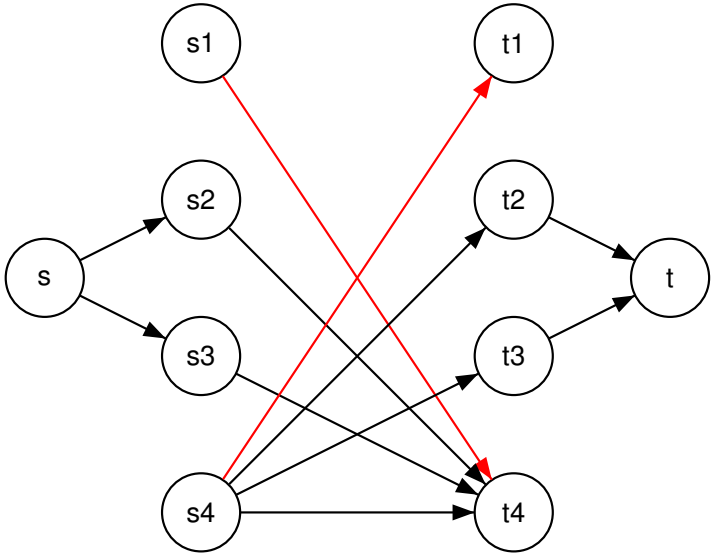
$\overline{c}$	t1	t2	t3	t4	u
s1	10	12	6	0	4
s2	4	8	5	0	7
s3	9	8	13	0	6
s4	**0**	**0**	**0**	0	5
v	1	6	3	0	**32**

We still have three admissible cells: edge $[4, 1]$ is chosen next.

	t1	t2	t3	t4	*Mate*
s1	15	22	13	4	4
s2	12	21	15	7	nil
s3	16	20	22	6	nil
s4	6	11	8	5	nil
Mate	nil	nil	nil	1	**4**

$x_{14} = 1$, $Card = 1$.

Visualization of Step 2



$\overline{c}$	t1	t2	t3	t4	u
s1	10	12	6	0	4
s2	4	8	5	0	7
s3	9	8	13	0	6
s4	0	0	0	0	5
v	1	6	3	0	**32**

	t1	t2	t3	t4	*Mate*
s1	15	22	13	4	4
s2	12	21	15	7	nil
s3	16	20	22	6	nil
s4	6	11	8	5	1
Mate	4	nil	nil	1	**10**

No admissible cells are left on unmatched rows and columns. The current partial matching is maximal.

$x_{14} = x_{41} = 1$, $Card = 2$.

Primal feasibility test

It consists of counting how many edges have been inserted into the primal solution (partial matching).

Primal feasibility test: ??.

Primal feasibility test

It consists of counting how many edges have been inserted into the primal solution (partial matching).

Primal feasibility test: $Card = n$.

Step 3: Search for an augmenting path

Step 3 consists of searching for an *augmenting path*, which is also an *alternating path* since the graph is bipartite. This is a path a unit of flow can follow to go from s to t.

Step 3: Search for an augmenting path

Step 3 consists of searching for an *augmenting path*, which is also an *alternating path* since the graph is bipartite. This is a path a unit of flow can follow to go from s to t.

Every time an s-t path is found, the cardinality of the current partial matching can be increased by 1 (primal iteration). To find the s-t path it may be necessary to execute at most $O(n)$ dual iterations, because each of them allows to reach one more node in $\mathcal{T}$.

Step 3: Search for an augmenting path

Step 3 consists of searching for an *augmenting path*, which is also an *alternating path* since the graph is bipartite. This is a path a unit of flow can follow to go from s to t.

Every time an s-t path is found, the cardinality of the current partial matching can be increased by 1 (primal iteration). To find the s-t path it may be necessary to execute at most $O(n)$ dual iterations, because each of them allows to reach one more node in $\mathcal{T}$.

The path starts from s; every node in $\mathcal{S}$ and $\mathcal{T}$ that can be reached is labeled. *Label* of a node is its predecessor.
L is the set of labels to be used to generate others.

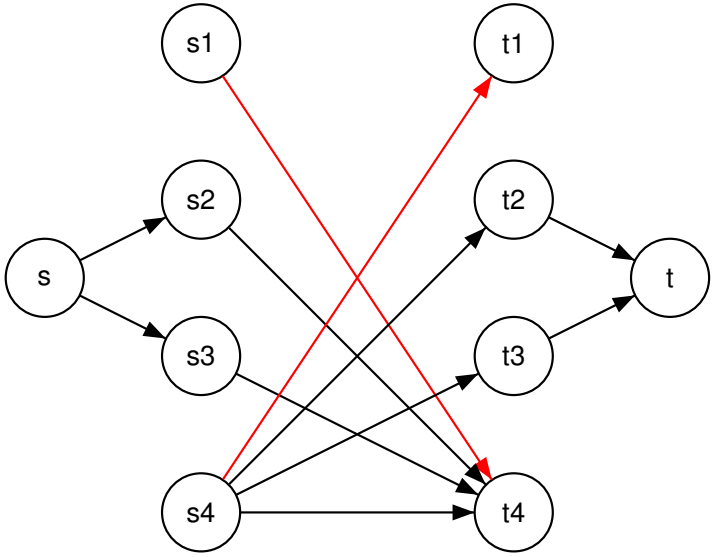
Vector p stores the minimum reduced cost value for each unlabeled column among those in labeled rows.
Vector π stores the corresponding row.

Step 3.1: Path initialization

Begin Step 3.1
L:=$\emptyset$;
for $k \in \mathcal{S} \cup \mathcal{T}$ **do**
  $Label(k)$:=nil;
end for
for $j \in \mathcal{T}$ **do**
  $p(j)$:=∞; $\pi(j)$:=nil;
end for
for $i \in \mathcal{S} : (Mate(i) = nil)$ **do**
  $Label(i)$:=s;
  L:=$L \cup \{i\}$;
  for $j \in \mathcal{T} : (Label(j) = nil)$ **do**
    if $c(i,j) - u(i) - v(j) < p(j)$ **then**
      $p(j)$:=$c(i,j) - u(i) - v(j)$; $\pi(j)$:=i;
    end if
  end for
end for
End Step 3.1

Complexity: $O(n^2)$**.**

Visualization of Step 3.1



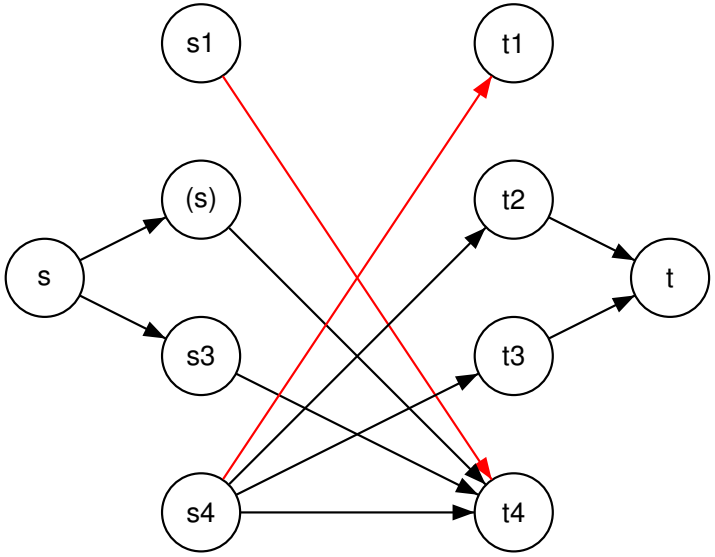
$\overline{c}$	t1	t2	t3	t4	u
s1	10	12	6	0	4
s2	4	8	5	0	7
s3	9	8	13	0	6
s4	0	0	0	0	5
v	1	6	3	0	**32**
p	∞	∞	∞	∞	
π	nil	nil	nil	nil	

	t1	t2	t3	t4	*Mate*
s1	15	22	13	4	4
s2	12	21	15	7	nil
s3	16	20	22	6	nil
s4	6	11	8	5	1
Mate	4	nil	nil	1	**10**

There are two unmatched nodes in $\mathcal{S}$.

$x_{14} = x_{41} = 1$, $Card = 2$.

Visualization of Step 3.1



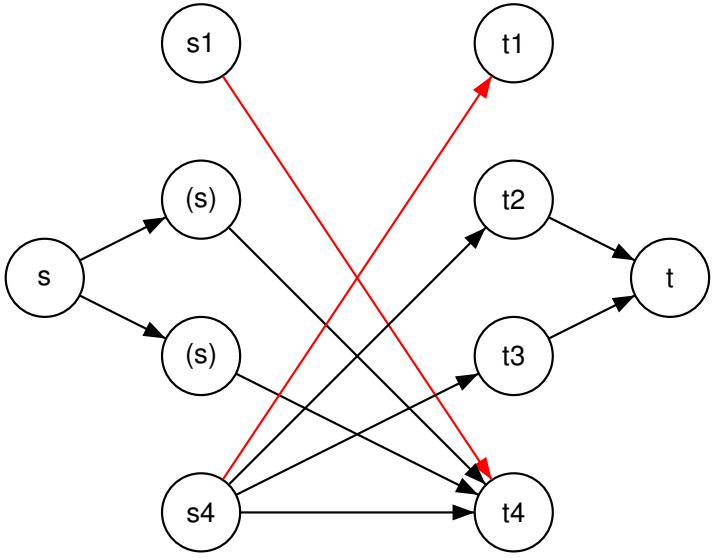
$\overline{c}$	t1	t2	t3	t4	u
s1	10	12	6	0	4
s2	**4**	**8**	**5**	**0**	7
s3	9	8	13	0	6
s4	0	0	0	0	5
v	1	6	3	0	**32**
p	4	8	5	0	
π	s2	s2	s2	s2	

	t1	t2	t3	t4	*Mate*
s1	15	22	13	4	4
s2	12	21	15	7	nil
s3	16	20	22	6	nil
s4	6	11	8	5	1
Mate	4	nil	nil	1	**10**

Insert $L = \{s2\}$.

$x_{14} = x_{41} = 1$, $Card = 2$.

Visualization of Step 3.1



$\overline{c}$	t1	t2	t3	t4	u
s1	10	12	6	0	4
s2	4	8	5	0	7
s3	**9**	**8**	**13**	**0**	6
s4	0	0	0	0	5
v	1	6	3	0	**32**
p	4	8	5	0	
π	s2	s2	s2	s2	

	t1	t2	t3	t4	*Mate*
s1	15	22	13	4	4
s2	12	21	15	7	nil
s3	16	20	22	6	nil
s4	6	11	8	5	1
Mate	4	nil	nil	1	**10**

Insert: $L = \{s2, s3\}$.

$x_{14} = x_{41} = 1$, $Card = 2$.

Step 3.2: Label propagation

Begin Step 3.2
Extract k from L;
if $k \in \mathcal{S}$ **then**
  Step 3.2.A: Propagation from $k \in \mathcal{S}$ **to** $\mathcal{T}$
else
  if ($Mate(k) \neq nil$) **then**
    Step 3.2.B: Propagation from $k \in \mathcal{T}$ **to** $\mathcal{S}$
  else
    $Path := k$;
  end if
end if
End Step 3.2

Propagation stops when an unmatched node $k \in \mathcal{T}$ is labeled.

Each node is inserted/extracted in/from L at most once.

Step 3.2.A: Label propagation from $\mathcal{S}$ to $\mathcal{T}$

Begin Step 3.2.A
for $j \in \mathcal{T} : (Label(j) = nil) \wedge (c(k,j) - u(k) - v(j) = 0)$ **do**
  $Label(j)$:=k;
  $L := L \cup \{j\}$;
end for
End Step 3.2.A

Propagation from $k \in \mathcal{S}$ to $\mathcal{T}$ occurs along edges that:

- correspond to admissible cells;
- do not belong to the current partial matching.

Complexity: $O(n)$**.**

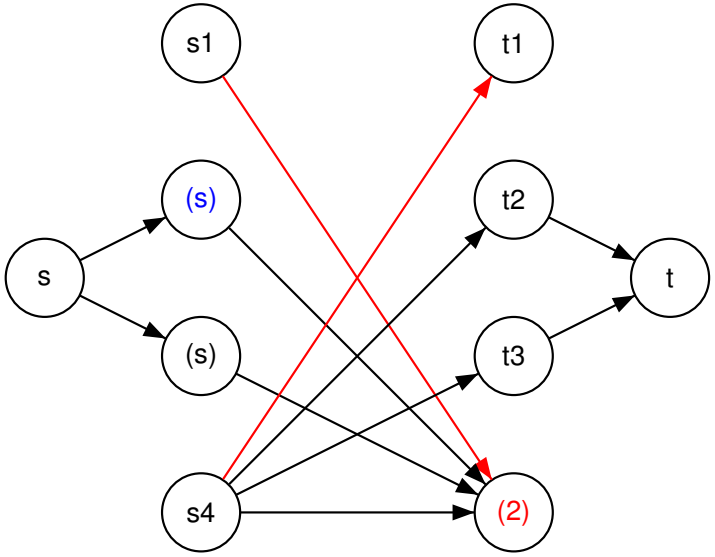
Step 3.2.B: Label propagation from $\mathcal{T}$ to $\mathcal{S}$

Begin Step 3.2.B
if $(Label(Mate(k)) = nil)$ **then**
  $Label(Mate(k))$:=k;
  $L := L \cup \{Mate(k)\}$;
  for $j \in \mathcal{T} : (Label(j) = nil)$ **do**
    if $c(Mate(k), j) - u(Mate(k)) - v(j) < p(j)$ **then**
      $p(j) := c(Mate(k), j) - u(Mate(k)) - v(j)$;
      $\pi(j) := Mate(k)$;
    end if
  end for
end if
End Step 3.2.B

Propagation from $k \in \mathcal{T}$ to $\mathcal{S}$ occurs along edges of the partial matching.

Complexity: $O(n)$**.**

Visualization of Step 3.2



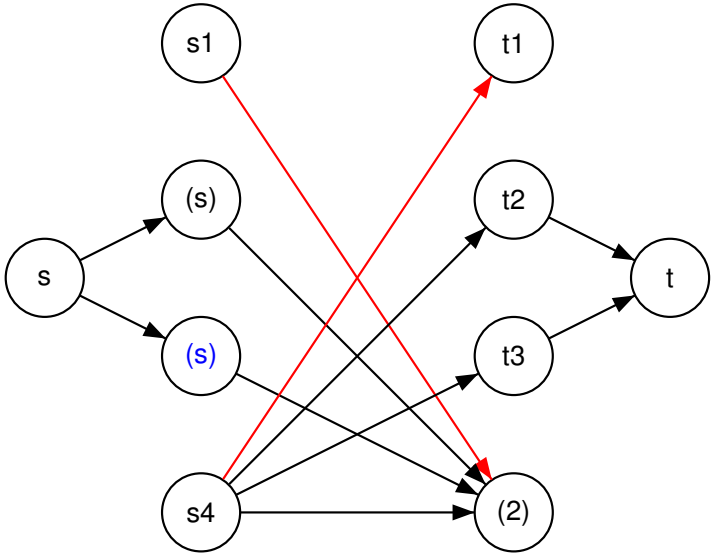
$\overline{c}$	t1	t2	t3	t4	u
s1	10	12	6	0	4
s2	4	8	5	0	7
s3	9	8	13	0	6
s4	0	0	0	0	5
v	1	6	3	0	**32**
p	4	8	5	0	
π	s2	s2	s2	s2	

	t1	t2	t3	t4	*Mate*
s1	15	22	13	4	4
s2	12	21	15	7	nil
s3	16	20	22	6	nil
s4	6	11	8	5	1
Mate	4	nil	nil	1	**10**

Extract: $L = \{s2, s3\}$.
Insert: $L = \{s3, t4\}$.

$x_{14} = x_{41} = 1$, $Card = 2$.

Visualization of Step 3.2



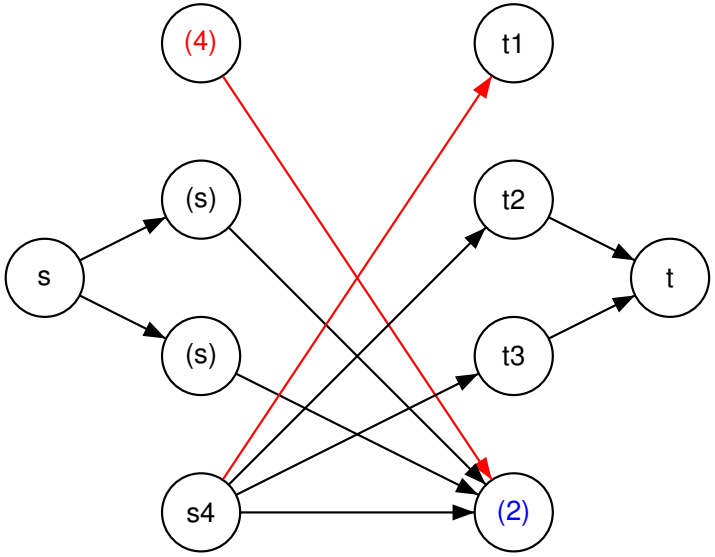
$\overline{c}$	t1	t2	t3	t4	u
s1	10	12	6	0	4
s2	4	8	5	0	7
s3	9	8	13	0	6
s4	0	0	0	0	5
v	1	6	3	0	**32**
p	4	8	5	0	
π	s2	s2	s2	s2	

	t1	t2	t3	t4	*Mate*
s1	15	22	13	4	4
s2	12	21	15	7	nil
s3	16	20	22	6	nil
s4	6	11	8	5	1
Mate	4	nil	nil	1	**10**

Extract: $L = \{s3, t4\}$.
Insert: $L = \{t4\}$.

$x_{14} = x_{41} = 1$, $Card = 2$.

Visualization of Step 3.2



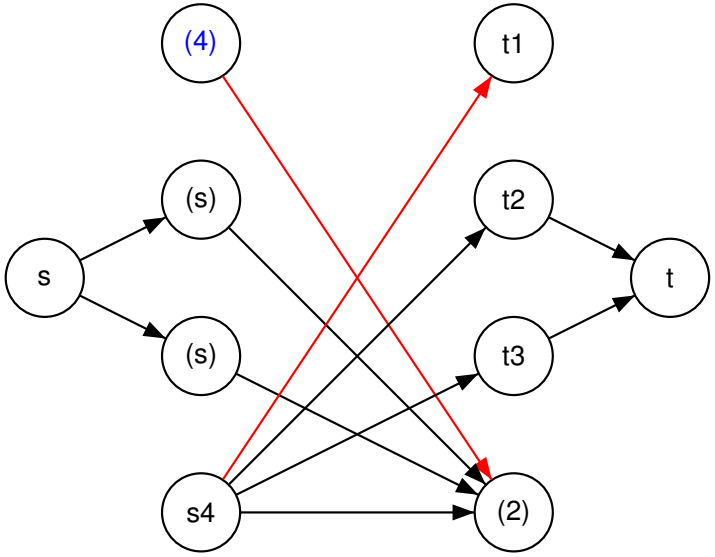
$\overline{c}$	t1	t2	t3	t4	u
s1	**10**	**12**	**6**	**0**	4
s2	4	8	5	0	7
s3	9	8	13	0	6
s4	0	0	0	0	5
v	1	6	3	0	**32**
p	4	8	5	0	
π	s2	s2	s2	s2	

	t1	t2	t3	t4	*Mate*
s1	15	22	13	4	4
s2	12	21	15	7	nil
s3	16	20	22	6	nil
s4	6	11	8	5	1
Mate	4	nil	nil	1	**10**

Extract: $L = \{t4\}$.
Insert: $L = \{s1\}$.

$x_{14} = x_{41} = 1$, $Card = 2$.

Visualization of Step 3.2



$\overline{c}$	t1	t2	t3	t4	u
s1	10	12	6	0	4
s2	4	8	5	0	7
s3	9	8	13	0	6
s4	0	0	0	0	5
v	1	6	3	0	**32**
p	4	8	5	0	
π	s2	s2	s2	s2	

	t1	t2	t3	t4	*Mate*
s1	15	22	13	4	4
s2	12	21	15	7	nil
s3	16	20	22	6	nil
s4	6	11	8	5	1
Mate	4	nil	nil	1	**10**

$x_{14} = x_{41} = 1$, $Card = 2$.

Extract: $L = \{s1\}$.
Insert: $L = \{\}$.

No s-t path has been found.
Nodes $s1$, $s2$, $s3$, $t4$ are labeled.
Nodes $s4$, $t1$, $t2$, $t3$ are not.

Step 4: Dual iteration

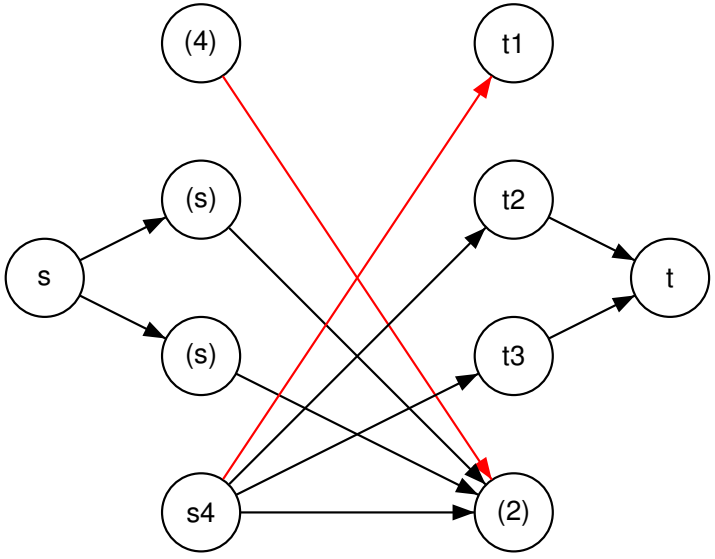
Begin Step 4
$\delta := \min_{j \in \mathcal{T}} \{p(j) : Label(j) = nil\}$;
for $i \in \mathcal{S} : Label(i) \neq nil$ **do**
  $u(i) := u(i) + \delta$;
end for
for $j \in \mathcal{T} : Label(j) \neq nil$ **do**
  $v(j) := v(j) - \delta$;
end for
for $j \in \mathcal{T} : Label(j) = nil$ **do**
  $p(j) := p(j) - \delta$;
end for
for $j \in \mathcal{T} : (Label(j) = nil) \wedge (p(j) = 0)$ **do**
  $Label(j) := (\pi(j))$;
  $L := L \cup \{j\}$;
end for
End Step 4

The value δ is the minimum reduced cost in the sub-matrix of labeled rows and unlabeled columns. However, owing to the vector p, finding δ takes $O(n)$ instead of $O(n^2)$.

Updating u takes $O(n)$.
Updating v takes $O(n)$.
Updating p takes $O(n)$.
Re-initializing L takes $O(n)$.

At least one more cell becomes admissible and it is used to label one more node in $\mathcal{T}$.

Visualization of Step 4



$\overline{c}$	t1	t2	t3	t4	u
s1	**10**	**12**	**6**	0	4
s2	**4**	**8**	**5**	0	7
s3	**9**	**8**	**13**	0	6
s4	0	0	0	0	5
v	1	6	3	0	**32**
p	**4**	**8**	**5**	0	
π	**s2**	s2	s2	s2	

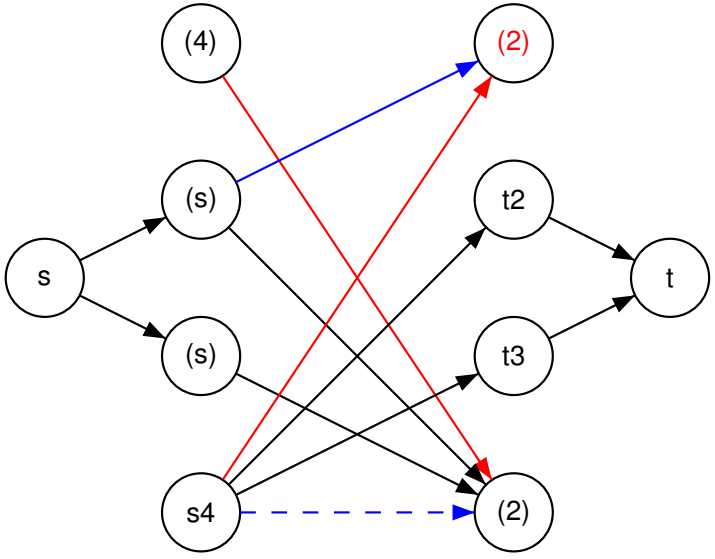
	t1	t2	t3	t4	*Mate*
s1	15	22	13	4	4
s2	12	21	15	7	nil
s3	16	20	22	6	nil
s4	6	11	8	5	1
Mate	4	nil	nil	1	**10**

$x_{14} = x_{41} = 1$, $Card = 2$.

We consider edges joining labeled nodes in $\mathcal{S}$ with unlabeled nodes in $\mathcal{T}$.

We find $\delta = 4$.

Visualization of Step 4



$\overline{c}$	t1	t2	t3	t4	u
s1	**6**	**8**	**2**	0	8
s2	**0**	**4**	**1**	0	11
s3	**5**	**4**	**9**	0	10
s4	0	0	0	**4**	5
v	1	6	3	-4	**40**
p	0	4	1	0	
π	s2	s2	s2	s2	

	t1	t2	t3	t4	*Mate*
s1	15	22	13	4	4
s2	12	21	15	7	nil
s3	16	20	22	6	nil
s4	6	11	8	5	1
Mate	4	nil	nil	1	**10**

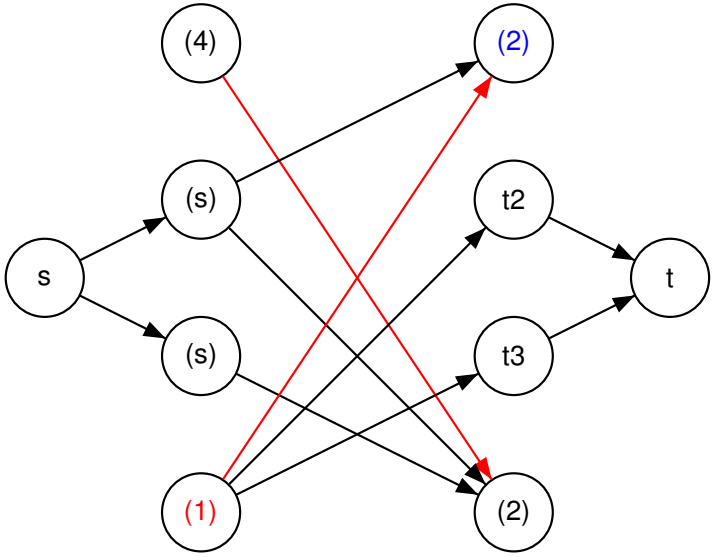
$x_{14} = x_{41} = 1$, $Card = 2$.

Increase u_1, u_2 and u_3 by δ.
Decrease v_4 by δ.
Decrease p_1, p_2 and p_3 by δ.

Cell $[4, 4]$ is no longer admissible.
Cell $[2, 1]$ becomes admissible.

Label $t1$ from $s2$.
Re-initialize: $L = \{t1\}$.

Visualization of Step 3.2



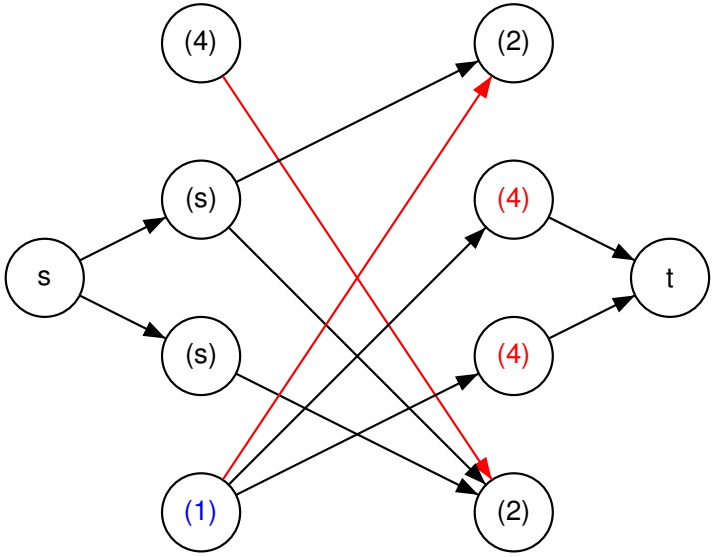
$\overline{c}$	t1	t2	t3	t4	u
s1	6	8	2	0	8
s2	0	4	1	0	11
s3	5	4	9	0	10
s4	0	**0**	**0**	4	5
v	1	6	3	-4	**40**
p	0	0	0	0	
π	s2	s4	s4	s2	

	t1	t2	t3	t4	*Mate*
s1	15	22	13	4	4
s2	12	21	15	7	nil
s3	16	20	22	6	nil
s4	6	11	8	5	1
Mate	4	nil	nil	1	**10**

Extract: $L = \{t1\}$.
Insert: $L = \{s4\}$.

$x_{14} = x_{41} = 1$, $Card = 2$.

Visualization of Step 3.2



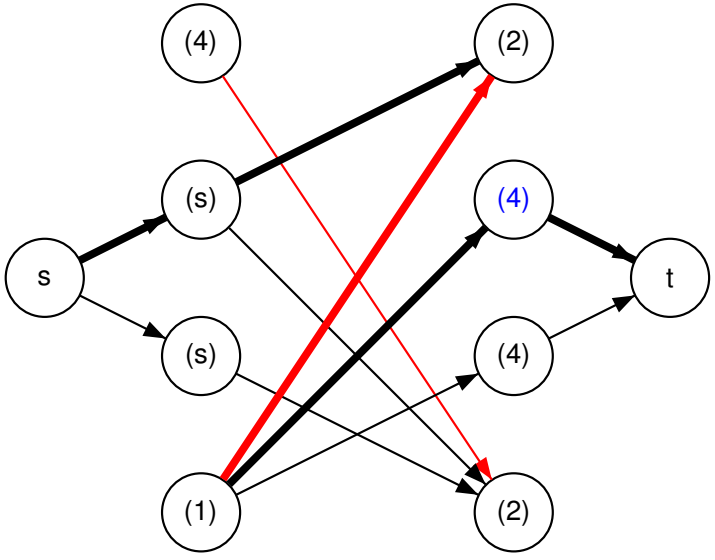
$\overline{c}$	t1	t2	t3	t4	u
s1	6	8	2	0	8
s2	0	4	1	0	11
s3	5	4	9	0	10
s4	0	0	0	4	5
v	1	6	3	-4	**40**
p	0	0	0	0	
π	s2	s4	s4	s2	

	t1	t2	t3	t4	*Mate*
s1	15	22	13	4	4
s2	12	21	15	7	nil
s3	16	20	22	6	nil
s4	6	11	8	5	1
Mate	4	nil	nil	1	**10**

Extract: $L = \{s4\}$.
Insert: $L = \{t2, t3\}$.

$x_{14} = x_{41} = 1$, $Card = 2$.

Visualization of Step 3.2



$\overline{c}$	t1	t2	t3	t4	u
s1	6	8	2	0	8
s2	0	4	1	0	11
s3	5	4	9	0	10
s4	0	0	0	4	5
v	1	6	3	-4	**40**
p	0	0	0	0	
π	s2	s4	s4	s2	

	t1	t2	t3	t4	*Mate*
s1	15	22	13	4	4
s2	12	21	15	7	nil
s3	16	20	22	6	nil
s4	6	11	8	5	1
Mate	4	nil	nil	1	**10**

$x_{14} = x_{41} = 1$, $Card = 2$.

Extract: $L = \{t2, t3\}$.

$t2$ is not matched: an s-t path has been found.

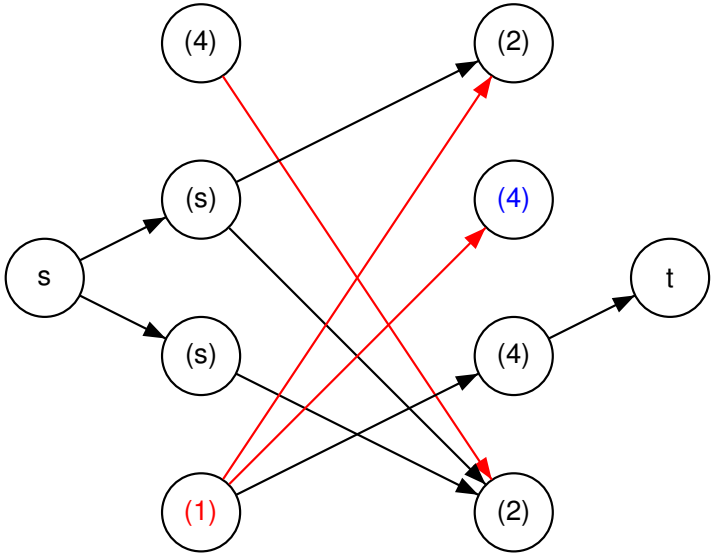
Step 5: primal iteration

Begin Step 5
$j := Path$;
repeat
  $i := Label(j)$;
  $Mate(j) := i$; $Mate(i) := j$;
  $x_{ij} := 1$; $z := z + c_{ij}$;
  $Card := Card + 1$;
  $j := Label(i)$;
  if $Label(i) \neq s$ **then**
    $x_{ij} := 0$; $z := z - c_{ij}$;
    $Card := Card - 1$;
  end if
until $(j = s)$;
End Step 5

The path is reconstructed backward from t to s. It has $O(n)$ edges.

Complexity: $O(n)$.

Visualization of Step 5



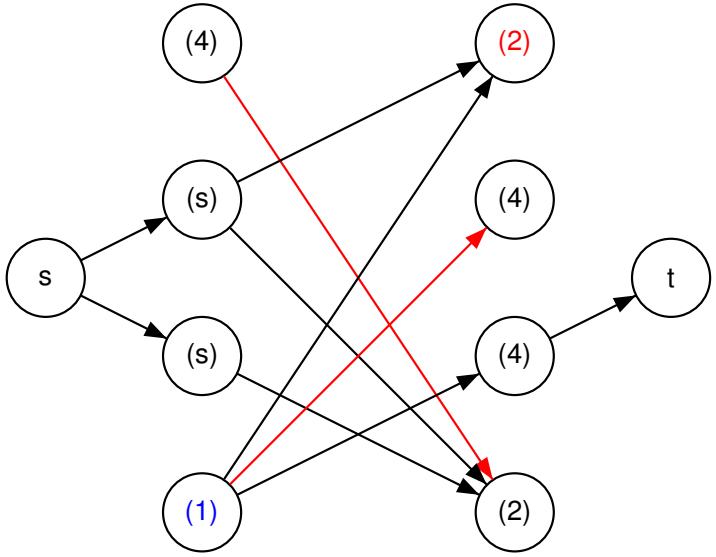
$\overline{c}$	t1	t2	t3	t4	u
s1	6	8	2	0	8
s2	0	4	1	0	11
s3	5	4	9	0	10
s4	0	0	0	4	5
v	1	6	3	-4	**40**
p	0	0	0	0	
π	s2	s4	s4	s2	

	t1	t2	t3	t4	*Mate*
s1	15	22	13	4	4
s2	12	21	15	7	nil
s3	16	20	22	6	nil
s4	6	11	8	5	2
Mate	4	4	nil	1	**21**

The predecessor of $t2$ is $s4$.

$x_{14} = x_{41} = x_{42} = 1$, $Card = 3$.

Visualization of Step 5



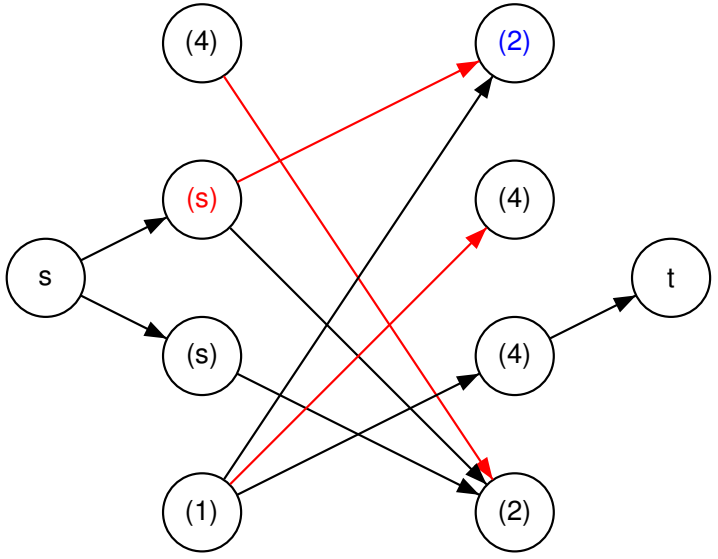
$\overline{c}$	t1	t2	t3	t4	u
s1	6	8	2	0	8
s2	0	4	1	0	11
s3	5	4	9	0	10
s4	0	0	0	4	5
v	1	6	3	-4	**40**
p	0	0	0	0	
π	s2	s4	s4	s2	

	t1	t2	t3	t4	*Mate*
s1	15	22	13	4	4
s2	12	21	15	7	nil
s3	16	20	22	6	nil
s4	6	11	8	5	2
Mate	nil	4	nil	1	**15**

The predecessor of $s4$ is $t1$.

$x_{14} = x_{42} = 1$, $Card = 2$.

Visualization of Step 5



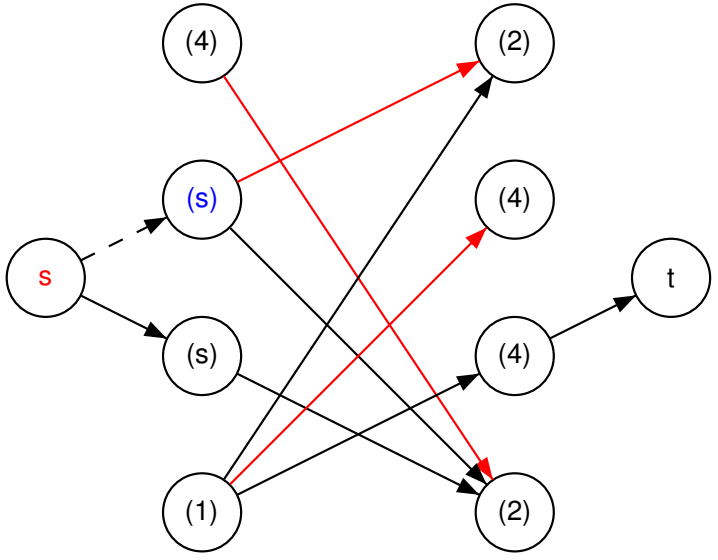
$\overline{c}$	t1	t2	t3	t4	u
s1	6	8	2	0	8
s2	0	4	1	0	11
s3	5	4	9	0	10
s4	0	0	0	4	5
v	1	6	3	-4	**40**
p	0	0	0	0	
π	s2	s4	s4	s2	

	t1	t2	t3	t4	*Mate*
s1	15	22	13	4	4
s2	12	21	15	7	1
s3	16	20	22	6	nil
s4	6	11	8	5	2
Mate	2	4	nil	1	**27**

The predecessor of $t1$ is $s2$.

$x_{14} = x_{42} = x_{21} = 1$, $Card = 3$.

Visualization of Step 5



$\overline{c}$	t1	t2	t3	t4	u
s1	6	8	2	0	8
s2	0	4	1	0	11
s3	5	4	9	0	10
s4	0	0	0	4	5
v	1	6	3	-4	**40**
p	0	0	0	0	
π	s2	s4	s4	s2	

	t1	t2	t3	t4	*Mate*
s1	15	22	13	4	4
s2	12	21	15	7	1
s3	16	20	22	6	nil
s4	6	11	8	5	2
Mate	2	4	nil	1	**27**

$x_{14} = x_{42} = x_{21} = 1$, $Card = 3$.

The predecessor of $s2$ is s.

The primal solution has been updated.

$Card < n$.

We are ready for another stage.

Hungarian algorithm: complexity

Begin
Step 1: Dual initialization;
Step 2: Primal initialization;
while [1] (x is infeasible) **do**
  Step 3.1: Initialization
  $Path$:=nil;
  while [2] ($Path = nil$) **do**
    while [3] ($Path = nil$) $\wedge$ ($L \neq \emptyset$) **do**
      Step 3.2: Labeling procedure
    end while
    if $Path = nil$ **then**
      Step 4: Dual iteration;
    end if
  end while
  Step 5: Primal iteration;
end while
End

Step 1: $O(n^2)$.
Step 2: $O(n^2)$.
Loop 1: $O(n)$ times.
Step 3.1: $O(n)$.
Loop 2 (stage): $O(n)$ times.
Loop 3: $O(n)$ times.
Step 3.2: $O(n)$ $\forall$ node, i.e. $O(n^2)$ $\forall$ stage.
Step 4: $O(n)$.
Step 5: $O(n)$.

Overall complexity: $O(n^3)$.