On a new variant of Murty's ranking assignments algorithm

Marta Pascoal[(1)]

M. Eugénia Captivo[(2)]

João Clímaco[(3)]

marta@mat.uc.pt, maria.captivo@fc.ul.pt, jclimaco@inescc.pt

[(1)] CISUC, Mathematics Department, University of Coimbra, Portugal

[(2)] DEIO–CIO, Faculty of Sciences, University of Lisbon, Portugal

[(3)] INESCC, Faculty of Economics, University of Coimbra, Portugal

The assignment problem

Definitions

- $(\mathcal{N}, \mathcal{A})$ complete bipartite network, $\mathcal{N} = \mathcal{N}_1 \cup \mathcal{N}_2$, $\mathcal{N}_1, \mathcal{N}_2$ disjoint, $|\mathcal{N}_1| = |\mathcal{N}_2| = n$, $\mathcal{A} = \mathcal{N}_1 \times \mathcal{N}_2$.
- $c_{ij} \in \mathbb{N}_0$ cost associated with $(i, j) \in \mathcal{A}$.

The assignment problem

Definitions

- $(\mathcal{N}, \mathcal{A})$ complete bipartite network, $\mathcal{N} = \mathcal{N}_1 \cup \mathcal{N}_2$, $\mathcal{N}_1, \mathcal{N}_2$ disjoint, $|\mathcal{N}_1| = |\mathcal{N}_2| = n$, $\mathcal{A} = \mathcal{N}_1 \times \mathcal{N}_2$.
- $c_{ij} \in \mathbb{N}_0$ cost associated with $(i, j) \in \mathcal{A}$.

Problem

Assign each node in $\mathcal{N}_1$ to one node in $\mathcal{N}_2$ with minimum cost.

$$
\begin{array}{lrcl}
\min & \sum_{(i,j)\in\mathcal{A}} c_{ij} x_{ij} & & \\
\text{s.t.} & \sum_{j\in\mathcal{N}_2} x_{ij} & = & 1,\ i \in \mathcal{N}_1 \\
& \sum_{i\in\mathcal{N}_1} x_{ij} & = & 1,\ j \in \mathcal{N}_2 \\
& x_{ij} & \geq & 0,\ (i,j) \in \mathcal{A} \quad (x_{ij} \in \{0,1\})
\end{array}
$$

Algorithm for the assignment problem

The assignment problem can be seen as a minimum cost flow problem in a bipartite network, where:

- the maximum capacity of arc (i, j) is $u_{ij} = 1$,
- the requirement of node $i \in \mathcal{N}_1$ is $r_i = 1$, and the requirement of node $j \in \mathcal{N}_2$ is $r_j = -1$.

How to solve it:

- consider an initial flow;
- compute a shortest path in $(\mathcal{N}', \mathcal{A}')$, a residual network obtained from $(\mathcal{N}, \mathcal{A})$;
- update the flow in the network.

Algorithm for the assignment problem

The residual network $(\mathcal{N}', \mathcal{A}')$ is such that:

- $\mathcal{N}' = \mathcal{N} \cup \{s, t\}$
- $\mathcal{A}' = \{(i,j) : (i,j) \in \mathcal{A} \wedge x_{ij} = 0\} \cup$

$$\overbrace{\{(j,i) : (i,j) \in \mathcal{A} \wedge x_{ij} = 1\}}^{c_{ji} = -c_{ij}} \cup$$

$$\overbrace{\{(s,i) : i \in \mathcal{N}_1 \text{ not used}\}}^{c_{si}=0} \cup \overbrace{\{(j,t) : j \in \mathcal{N}_2 \text{ not used}\}}^{c_{jt}=0}$$

Algorithm for the assignment problem

The residual network $(\mathcal{N}', \mathcal{A}')$ is such that:

- $\mathcal{N}' = \mathcal{N} \cup \{s, t\}$
- $\mathcal{A}' = \{(i,j) : (i,j) \in \mathcal{A} \wedge x_{ij} = 0\} \cup$

$$\overbrace{\{(j,i) : (i,j) \in \mathcal{A} \wedge x_{ij} = 1\}}^{c_{ji} = -c_{ij}} \cup$$

$$\overbrace{\{(s,i) : i \in \mathcal{N}_1 \text{ not used}\}}^{c_{si} = 0} \cup \overbrace{\{(j,t) : j \in \mathcal{N}_2 \text{ not used}\}}^{c_{jt} = 0}$$

$flow \longleftarrow 0$

Repeat n times

Determine shortest path (p) from s to t in $(\mathcal{N}', \mathcal{A}')$

Update $flow$, in 1 unit, throughout the arcs of p

Shortest path in $(\mathcal{N}', \mathcal{A}')$

Labeling algorithm:

$\pi_i \longleftarrow +\infty, \ \ \forall i \in \mathcal{N} - \{s\}; \ \pi_s \longleftarrow 0$

$L \longleftarrow \{s\}$

`While` ($L \neq \emptyset$) `Do`

  $i \longleftarrow$ node in L; Delete i from L

  `For` ($(i,j) \in \mathcal{A}$ such that $x_{ij} = 0$) `Do`

    `If` (π_j is improved) `Then`

      $\pi_j \longleftarrow \pi_i + c_{ij}$; Add j to L

  `If` (there is a (j,i) preceding i in $flow$) `Then`

    `If` (π_j is improved) `Then`

      $\pi_j \longleftarrow \pi_i - c_{ji}$; Add j to L

Shortest path in $(\mathcal{N}', \mathcal{A}')$

Labeling algorithm:

$\pi_i \longleftarrow +\infty, \ \ \forall i \in \mathcal{N} - \{s\}; \ \pi_s \longleftarrow 0$

$L \longleftarrow \{s\}$

`While` ($L \neq \emptyset$) `Do`

  $i \longleftarrow$ node in L; Delete i from L

  `For` ($(i,j) \in \mathcal{A}$ such that $x_{ij} = 0$) `Do`

    `If` (π_j is improved) `Then`

      $\pi_j \longleftarrow \pi_i + c_{ij}$; Add j to L

  `If` (there is a (j,i) preceding i in $flow$) `Then`

    `If` (π_j is improved) `Then`

      $\pi_j \longleftarrow \pi_i - c_{ji}$; Add j to L

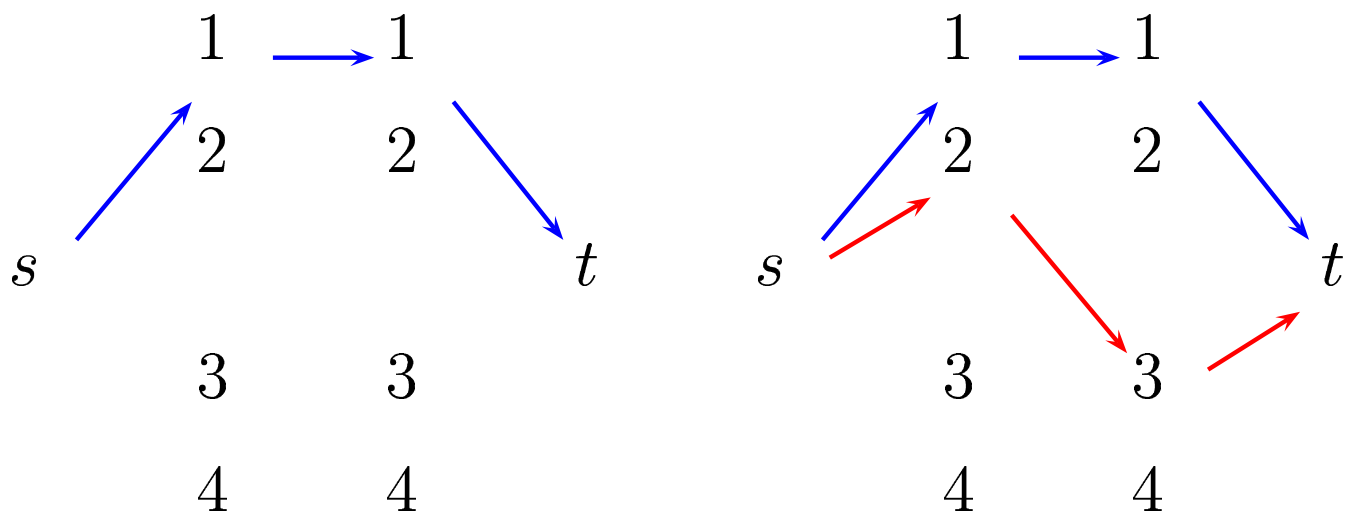
n SP problems in a network with $\mathcal{O}(n)$ nodes $\Rightarrow \mathcal{O}(n\, c(n))$

Example

$$C = \begin{bmatrix} 0 & 0 & 2 & 2 \\ 1 & 3 & 0 & 4 \\ 0 & 6 & 1 & 0 \\ 3 & 3 & 2 & 0 \end{bmatrix}$$

Best assignment:

$a = \{(1,2),(2,3),(3,1),(4,4)\}$

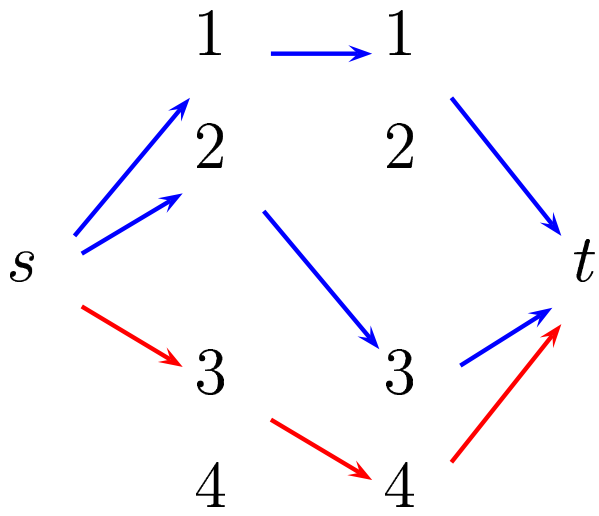


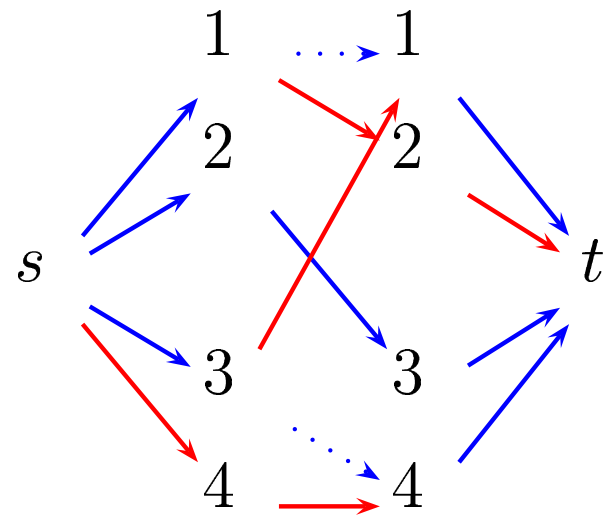
Example

$$C = \begin{bmatrix} 0 & 0 & 2 & 2 \\ 1 & 3 & 0 & 4 \\ 0 & 6 & 1 & 0 \\ 3 & 3 & 2 & 0 \end{bmatrix}$$

Best assignment:

$a = \{(1,2),(2,3),(3,1),(4,4)\}$





The ranking assignments problem

Consider a value $K \in \mathbb{N}$.

Determine $a_1, \ldots, a_K$ by non-decreasing order of cost:

- a_i is determined before a_{i+1}
- $c(a_i) \leq c(a_{i+1})$
- $c(a_i) \leq c(a)$, for any $a \notin \{a_1, \ldots, a_K\}$

Compute the best assignment and store it
While ($k < K$) and (there are assignments left to analyse) Do
    Choose the best assignment stored (a_k) and increase k
    For $(i, j) \in a_k$ Do
        Compute the best assignment coinciding with a_k until (i, j), but not containing (i, j)
        Store it

Murty's algorithm

$a \longleftarrow$ least cost assignment; $k \longleftarrow 0$
$X \longleftarrow \{a\}$

Murty's algorithm

$a \longleftarrow$ least cost assignment; $k \longleftarrow 0$
$X \longleftarrow \{a\}$
`While` (($k < K$) and ($X \neq \emptyset$)) `Do`
    $k \longleftarrow k + 1$
    $a_k \longleftarrow$ best assignment in X; Remove a_k from X
    Restore the a_k computation conditions

Murty's algorithm

$a \longleftarrow$ least cost assignment; $k \longleftarrow 0$
$X \longleftarrow \{a\}$
`While` (($k < K$) and ($X \neq \emptyset$)) `Do`
    $k \longleftarrow k + 1$
    $a_k \longleftarrow$ best assignment in X; Remove a_k from X
    Restore the a_k computation conditions
    `For` ($(i,j) \in a_k$) `Do`
        $a \longleftarrow$ best assignment coinciding with a_k in
            the arcs until (i,j), but not containing (i,j)
        Add a to X

Murty's algorithm

$a \longleftarrow$ least cost assignment; $k \longleftarrow 0$
$X \longleftarrow \{a\}$
`While` $((k < K)$ and $(X \neq \emptyset))$ `Do`
  $k \longleftarrow k + 1$
  $a_k \longleftarrow$ best assignment in X; Remove a_k from X
  Restore the a_k computation conditions
  `For` $((i,j) \in a_k)$ `Do`
    $a \longleftarrow$ best assignment coinciding with a_k in
      the arcs until (i,j), but not containing (i,j)
    Add a to X

  Remove extreme nodes of arcs until the previous to (i,j)
  Remove (i,j)
  Determine the best assignment in the modified network

Murty's algorithm variant

In the previous algorithm the a_k arcs order of analysis is not specified.

Same algorithm using a specific order of analysis (namely reversing it, that is, analysing arcs from the last to the first)

$\Downarrow$

Avoid the resolution of n new assignment problems.

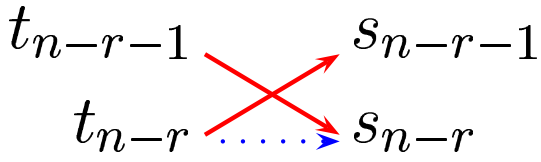
Let a_k be $\{(i_1, j_1), \ldots, (i_r, j_r), (t_1, s_1), \ldots, (t_{n-r}, s_{n-r})\}$, which has been determined with:

- nodes $i_1, \ldots, i_r \in \mathcal{N}_1$ and $j_1, \ldots, j_r \in \mathcal{N}_2$ removed,
- arcs $(m_1, p_1), \ldots, (m_\ell, p_\ell)$ removed.

Murty's algorithm variant

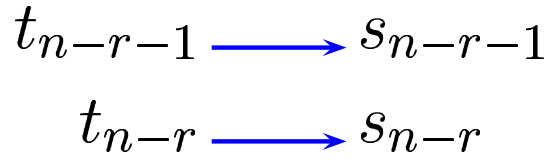
$t_{n-r} \longrightarrow s_{n-r}$

Initialize

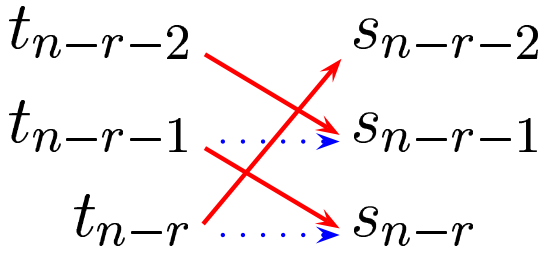


$b \longleftarrow$ Restore t_{n-r-1} and s_{n-r-1}

Add $\{(i_1, j_1), \ldots, (t_{n-r-2}, s_{n-r-2})\} \cup b$

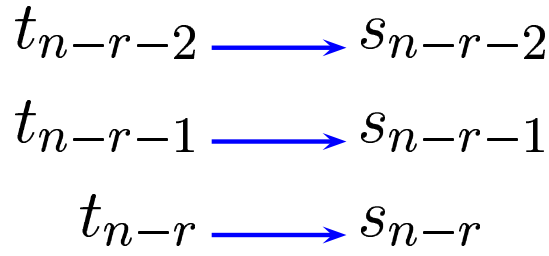


Restore (t_{n-r-1}, s_{n-r-1})



$b \longleftarrow$ Restore t_{n-r-2} and s_{n-r-2}

Add $\{(i_1, j_1), \ldots, (t_{n-r-3}, s_{n-r-3})\} \cup b$



Restore (t_{n-r-2}, s_{n-r-2})

$\ldots$

Inserting nodes in an assignment

Let b be the best assignment in $(\mathcal{N}, \mathcal{A})$ and consider:

- i is added to $\mathcal{N}_1$, i' added to $\mathcal{N}_2$;
- arcs $(i, j) \in \mathcal{A} - \{(i, i')\}$ and $(j, i') \in \mathcal{A} - \{(i, i')\}$ are added to $\mathcal{A}$.

The best assignment b' in the new network can be obtained by:

- computing the shortest path (p) from s to t;
- updating the flow, in 1 unit, throughout the arcs of p.

Inserting arcs in an assignment

Let b be the best assignment in $(\mathcal{N}, \mathcal{A})$ and consider:

- arc (i, i') is added to $\mathcal{A}$.

The best assignment b' in the new network can be obtained by the procedure used when inserting i and i'.

Inserting arcs in an assignment

Let b be the best assignment in $(\mathcal{N}, \mathcal{A})$ and consider:

- arc (i, i') is added to $\mathcal{A}$.

The best assignment b' in the new network can be obtained by the procedure used when inserting i and i'.

If (i, i') is an arc (t_x, s_x) of a_k, then b' is the best assignment in a network without:

- $i_1, \ldots, i_r$, $t_1, \ldots, t_{x-1}$ and $j_1, \ldots, j_r$, $s_1, \ldots, s_{x-1}$;
- $(m_1, p_1), \ldots, (m_\ell, p_\ell)$.

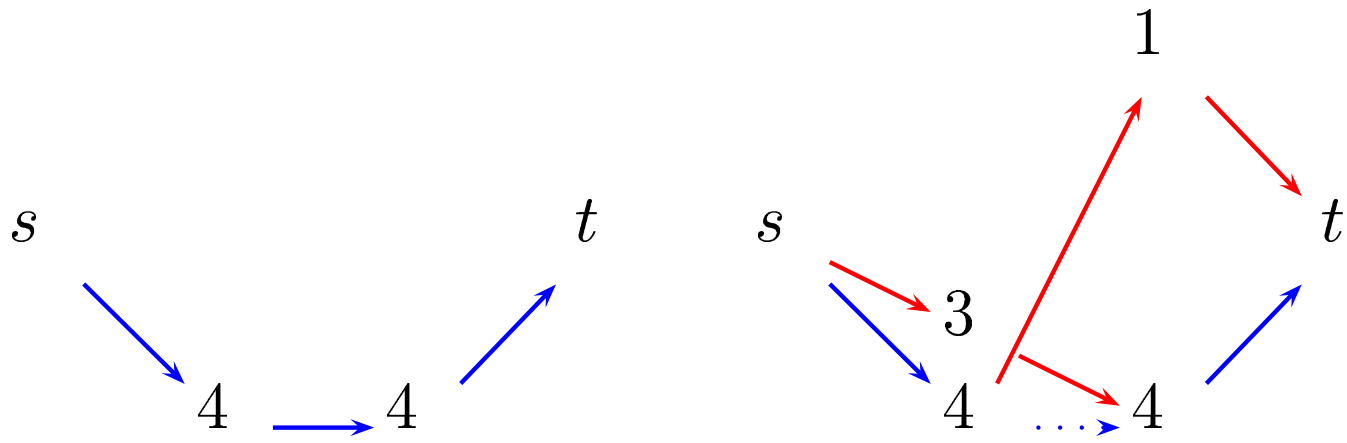
$$\Downarrow$$

Therefore, $b' = \{(t_x, s_x), \ldots, (t_{n-r}, s_{n-r})\}$, which means it is enough to consider a part of a_k.

Example

$$C = \begin{bmatrix} 0 & 0 & 2 & 2 \\ 1 & 3 & 0 & 4 \\ 0 & 6 & 1 & 0 \\ 3 & 3 & 2 & 0 \end{bmatrix}$$

$$a_1 = \{(1,2),(2,3),(3,1),(4,4)\}$$
$$X = \emptyset$$

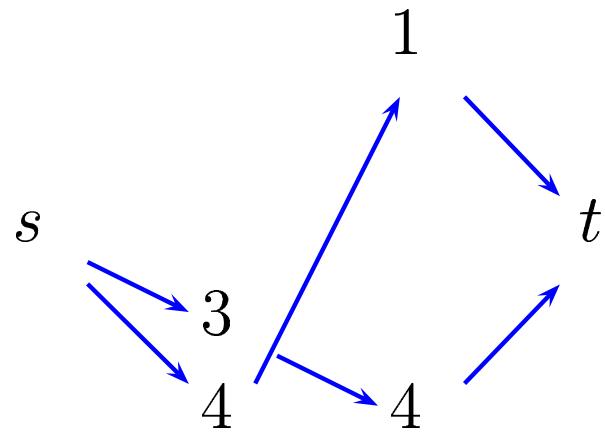


Add $(4,4)$ to $\mathcal{A}$. Add 3 to $\mathcal{N}_1$ and 1 to $\mathcal{N}_2$, with $c_{31} = +\infty$.

Example

$$C = \begin{bmatrix} 0 & 0 & 2 & 2 \\ 1 & 3 & 0 & 4 \\ 0 & 6 & 1 & 0 \\ 3 & 3 & 2 & 0 \end{bmatrix}$$

$$a_1 = \{(1,2),(2,3),(3,1),(4,4)\}$$
$$X = \{\{(1,2),(2,3),(3,4),(4,1)\}\}$$

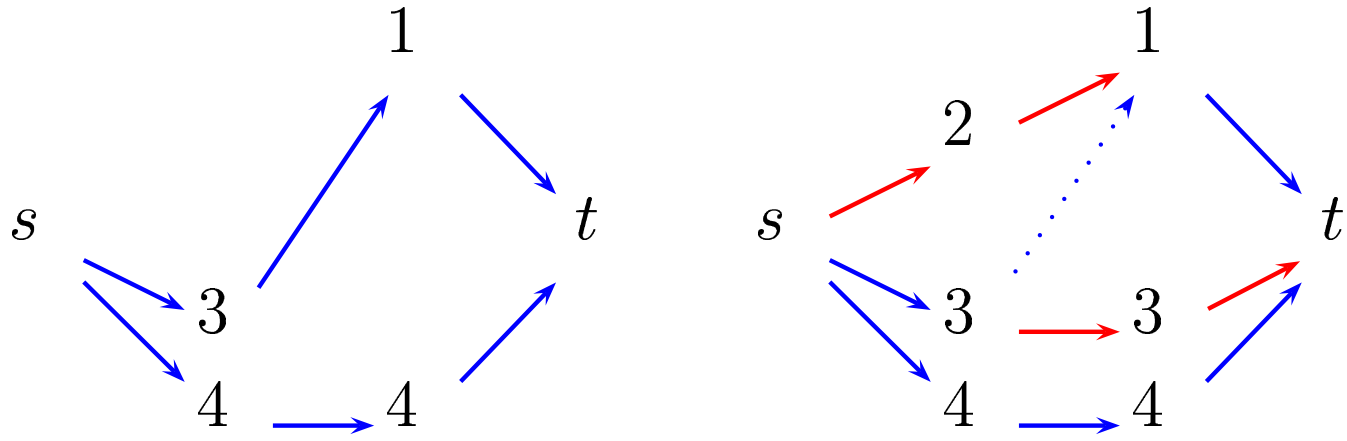


Add $(4,4)$ to $\mathcal{A}$. Add 3 to $\mathcal{N}_1$ and 1 to $\mathcal{N}_2$, with $c_{31} = +\infty$.

Example

$$C = \begin{bmatrix} 0 & 0 & 2 & 2 \\ 1 & 3 & 0 & 4 \\ 0 & 6 & 1 & 0 \\ 3 & 3 & 2 & 0 \end{bmatrix}$$

$$a_1 = \{(1,2),(2,3),(3,1),(4,4)\}$$
$$X = \{\{(1,2),(2,3),(3,4),(4,1)\}\}$$

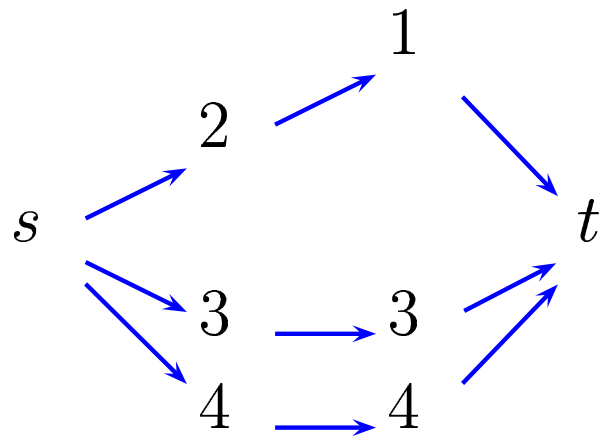


Add $(4,4)$ to $\mathcal{A}$. Add 3 to $\mathcal{N}_1$ and 1 to $\mathcal{N}_2$, with $c_{31} = +\infty$.
Add $(3,1)$ to $\mathcal{A}$. Add 2 to $\mathcal{N}_1$ and 3 to $\mathcal{N}_2$, with $c_{23} = +\infty$.

Example

$$C = \begin{bmatrix} 0 & 0 & 2 & 2 \\ 1 & 3 & 0 & 4 \\ 0 & 6 & 1 & 0 \\ 3 & 3 & 2 & 0 \end{bmatrix}$$

$$a_1 = \{(1,2),(2,3),(3,1),(4,4)\}$$
$$X = \{\{(1,2),(2,3),(3,4),(4,1)\},$$
$$\{(1,2),(2,1),(3,3),(4,4)\}\}$$

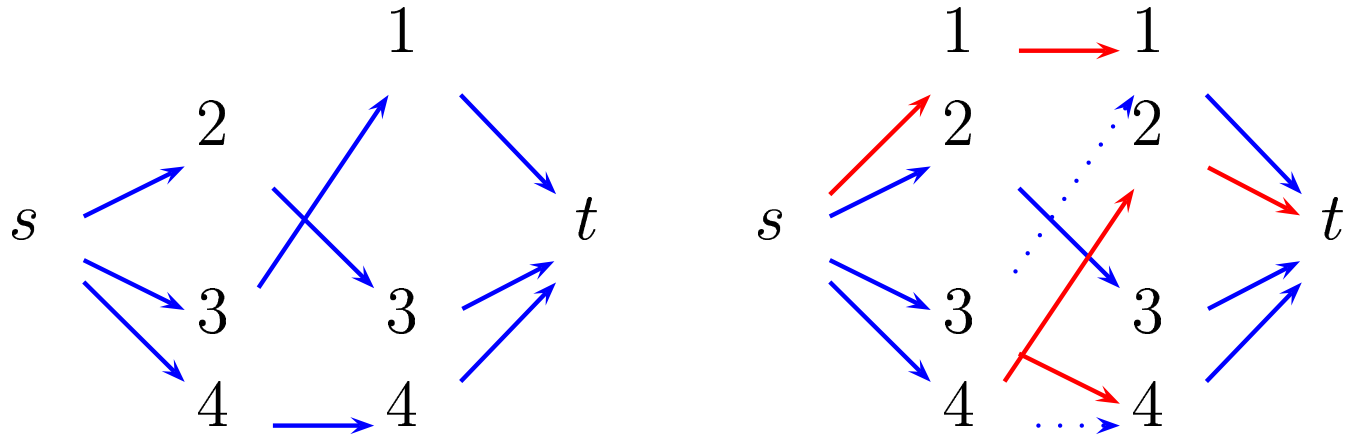


Add $(4,4)$ to $\mathcal{A}$. Add 3 to $\mathcal{N}_1$ and 1 to $\mathcal{N}_2$, with $c_{31} = +\infty$.
Add $(3,1)$ to $\mathcal{A}$. Add 2 to $\mathcal{N}_1$ and 3 to $\mathcal{N}_2$, with $c_{23} = +\infty$.

Example

$$C = \begin{bmatrix} 0 & 0 & 2 & 2 \\ 1 & 3 & 0 & 4 \\ 0 & 6 & 1 & 0 \\ 3 & 3 & 2 & 0 \end{bmatrix}$$

$$a_1 = \{(1,2),(2,3),(3,1),(4,4)\}$$
$$X = \{\{(1,2),(2,3),(3,4),(4,1)\},$$
$$\{(1,2),(2,1),(3,3),(4,4)\}\}$$



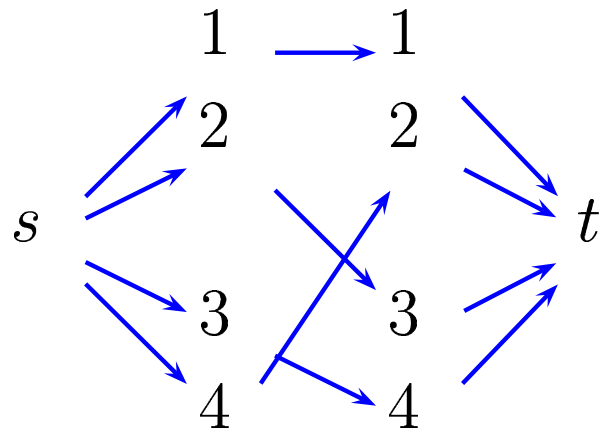
Add $(4,4)$ to $\mathcal{A}$. Add 3 to $\mathcal{N}_1$ and 1 to $\mathcal{N}_2$, with $c_{31} = +\infty$.
Add $(3,1)$ to $\mathcal{A}$. Add 2 to $\mathcal{N}_1$ and 3 to $\mathcal{N}_2$, with $c_{23} = +\infty$.
Add $(2,3)$ to $\mathcal{A}$. Add 1 to $\mathcal{N}_1$ and 2 to $\mathcal{N}_2$, with $c_{12} = +\infty$.

Example

$$C = \begin{bmatrix} 0 & 0 & 2 & 2 \\ 1 & 3 & 0 & 4 \\ 0 & 6 & 1 & 0 \\ 3 & 3 & 2 & 0 \end{bmatrix}$$

$$a_1 = \{(1,2),(2,3),(3,1),(4,4)\}$$

$$X = \{\{(1,2),(2,3),(3,4),(4,1)\},\ \{(1,2),(2,1),(3,3),(4,4)\},\ \{(1,1),(2,3),(3,4),(4,2)\}\}$$



Add $(4,4)$ to $\mathcal{A}$. Add 3 to $\mathcal{N}_1$ and 1 to $\mathcal{N}_2$, with $c_{31} = +\infty$.
Add $(3,1)$ to $\mathcal{A}$. Add 2 to $\mathcal{N}_1$ and 3 to $\mathcal{N}_2$, with $c_{23} = +\infty$.
Add $(2,3)$ to $\mathcal{A}$. Add 1 to $\mathcal{N}_1$ and 2 to $\mathcal{N}_2$, with $c_{12} = +\infty$.

Computational complexity

"Straightforward" implementation:
K assignments are analysed and for each one $\mathcal{O}(n)$ new assignment problems are solved.

$$\Downarrow$$

$$\mathcal{O}(Kn\,(n\,c(n))) = \mathcal{O}(Kn^2c(n))$$

Computational complexity

"Straightforward" implementation:
K assignments are analysed and for each one $\mathcal{O}(n)$ new assignment problems are solved.

$$\Downarrow$$

$$\mathcal{O}(Kn\,(n\,c(n))) = \mathcal{O}(Kn^2c(n))$$

New variant:
K assignments are analysed and for each one $\mathcal{O}(n)$ shortest path problems are solved.

$$\Downarrow$$

$$\mathcal{O}(Kn\,c(n))$$

Computational experiments

Computer characteristics:

- AMD Athlon
- Processor 1.5GHz
- 512 Mbytes of RAM

Complete bipartite networks:

- $n \in \{50, 100, 150, 200\}$ (generated with 10 seeds)
- c_{ij} randomly generated in $\{0, \ldots, 1000\}$

$K = 100$ assignments determined.

Results