Network Simplex Method

Combinatorial Problem Solving (CPS)

Enric Rodríguez-Carbonell

April 26, 2019

Network Programs

- A network program is of the form

$$\begin{aligned} &\min\ c^T x \\ &Ax = b \\ &\ell \leq x \leq u, \end{aligned}$$

where $c \in \mathbb{R}^m$, $b \in \mathbb{R}^n$ and $A \in \{-1, 0, 1\}^{n \times m}$ has the following property:

each column has exactly one 1 and one -1
(and so the remaining coefficients are 0)

- Note that n is the number of constraints and m is the number of variables

Network Programs

- A network program is of the form

$$\begin{aligned} &\min\ c^T x \\ &Ax = b \\ &\ell \le x \le u, \end{aligned}$$

  where $c \in \mathbb{R}^m$, $b \in \mathbb{R}^n$ and $A \in \{-1, 0, 1\}^{n \times m}$ has the following property:

  each column has exactly one 1 and one -1
  (and so the remaining coefficients are 0)

- Example:

$$\min\ x_1 + x_2 + 3x_3 + 10x_4$$

$$\begin{pmatrix} 1 & 0 & 1 & 1 \\ -1 & 1 & 0 & 0 \\ 0 & -1 & -1 & -1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 5 \\ 0 \\ -5 \end{pmatrix}$$

$$0 \le x_1 \le 4 \qquad 0 \le x_3 \le 4$$
$$0 \le x_2 \le 2 \qquad 0 \le x_4 \le 10$$

Minimum Cost Flow Problems

- Network programs can be seen as minimum cost flow problems in a graph
- We associate a digraph $G = (V, E)$ to the matrix of a network program:
  - Vertices V correspond to rows (constraints)
  - Edges E correspond to columns (variables)
  - A column with a 1 at row i and a -1 at row k gives an edge (i, k)

Minimum Cost Flow Problems

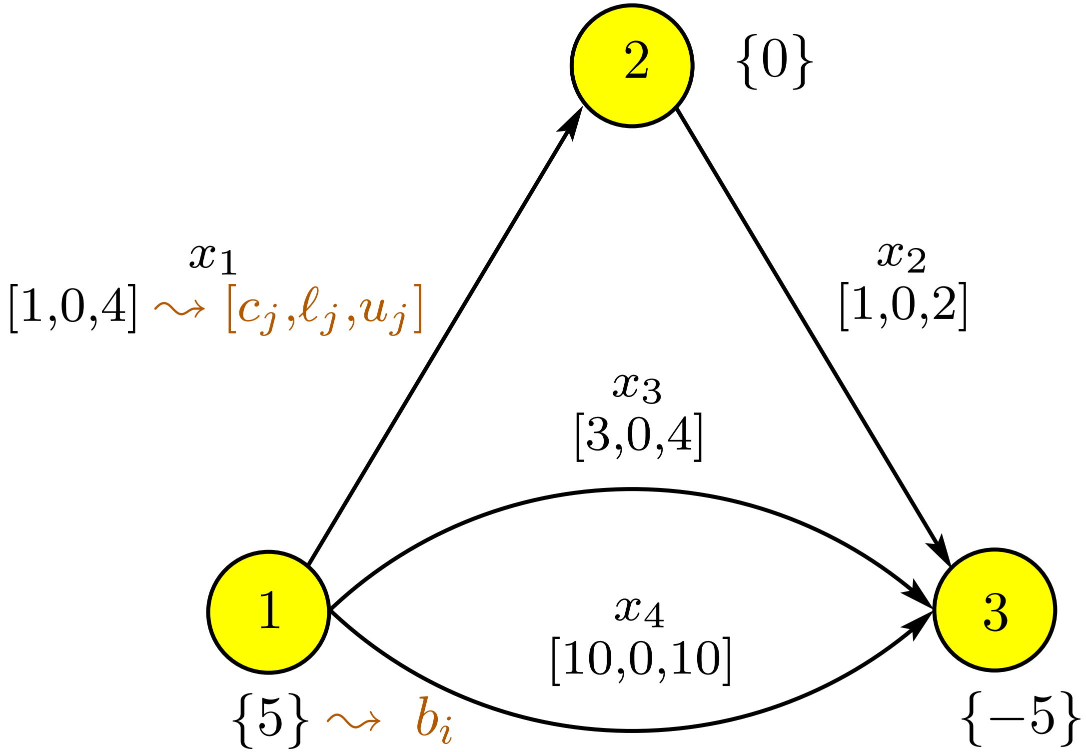
- Network programs can be seen as minimum cost flow problems in a graph
- We associate a digraph $G = (V, E)$ to the matrix of a network program:
  - Vertices V correspond to rows (constraints)
  - Edges E correspond to columns (variables)
  - A column with a 1 at row i and a -1 at row k gives an edge (i, k)
- Then we can reinterpret the other elements of the network program:
  - Each variable x_j is the flow sent along the j-th edge
  - The cost of sending 1 unit of flow is c_j
  - Flow cannot exceed capacity u_j
  - There must be a minimum flow ℓ_j (usually, 0)
  - Total production of flow at vertex i is determined by b_i
- So solving the network program consists in
  finding the feasible flow along the graph that minimizes the cost

Minimum Cost Flow Problems

$$\min \; x_1 + x_2 + 3x_3 + 10x_4$$

$$\begin{pmatrix} 1 & 0 & 1 & 1 \\ -1 & 1 & 0 & 0 \\ 0 & -1 & -1 & -1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 5 \\ 0 \\ -5 \end{pmatrix}$$

$$0 \leq x_1 \leq 4 \qquad 0 \leq x_3 \leq 4$$
$$0 \leq x_2 \leq 2 \qquad 0 \leq x_4 \leq 10$$



Network Simplex Method

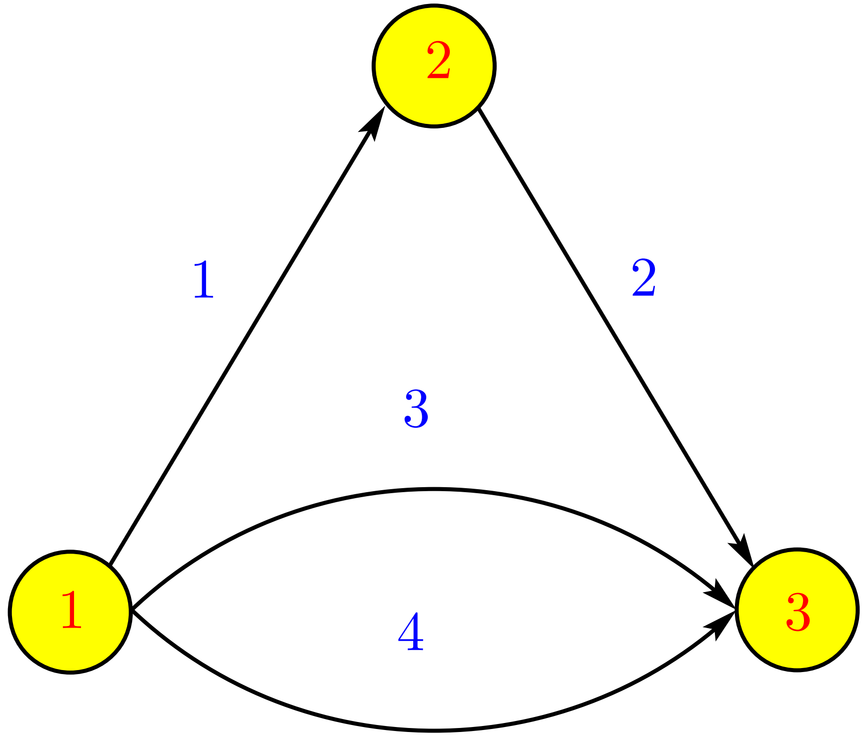
- Network programs satisfy Hoffman & Gale's conditions.
  So simplex method is guaranteed to give integer solutions (if ℓ, u, b in $\mathbb{Z}$)
- Moreover we can specialize the simplex method for network programs
- This lecture is devoted to this specialization: the network simplex method
- In the first place we need to revisit a bit of graph theory

Vertex-Edge Incidence Matrix

- The vertex-edge incidence matrix of digraph $G = (V, E)$ is a matrix A s.t.:
  - Rows are labelled by vertices
  - Columns are labelled by edges
  - For each $v \in V$ and $e \in E$, coefficient $a_{v,e}$ of A is
    - 1 if $e = (v, \cdot)$
    - -1 if $e = (\cdot, v)$
    - 0 otherwise

- Given a network program whose matrix is A,
  the vertex-edge incidence matrix of its associated digraph is precisely A

Vertex-Edge Incidence Matrix



$$
\begin{array}{cc}
 & \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\
\begin{array}{c} 1 \\ 2 \\ 3 \end{array} & \begin{pmatrix} 1 & 0 & 1 & 1 \\ -1 & 1 & 0 & 0 \\ 0 & -1 & -1 & -1 \end{pmatrix}
\end{array}
$$

Paths and Cycles

- A path is a finite sequence $P = (v_1, e_1, v_2, \ldots, v_K, e_K, v_{K+1})$ such that either $e_k = (v_k, v_{k+1})$ or $e_k = (v_{k+1}, v_k)$ for all $1 \leq k \leq K$

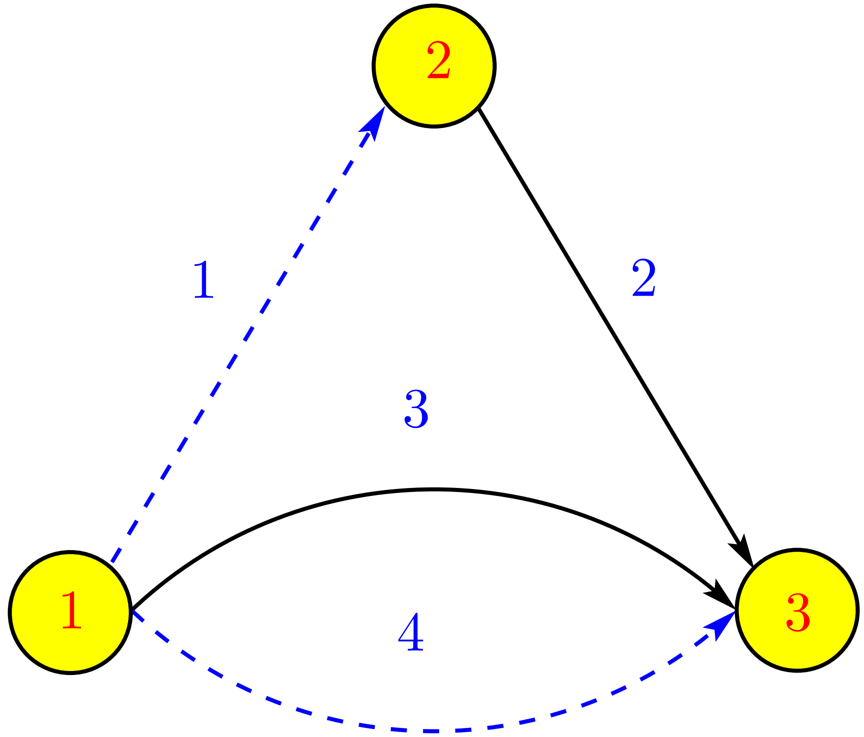
- Note that paths can invert the orientation of edges

- The orientation sequence of a path P is $(O_P(e_1), \ldots, O_P(e_k))$, where

$$O_P(e_k) \begin{cases} +1 & \text{if } e_k = (v_k, v_{k+1}) \\ -1 & \text{if } e_k = (v_{k+1}, v_k) \\ 0 & \text{otherwise} \end{cases}$$

- A cycle is a path such that the initial and the final vertices are the same

Paths and Cycles



$(3, 4, 1, 1, 2)$ is a path with orientation sequence $(-1, 1)$

Paths and Cycles

- **Prop.** Let $P = (v_1, e_1, v_2, \ldots, v_K, e_K, v_{K+1})$ be a path. Then

$$\sum_{k=1}^{K} O_P(e_k) \cdot a_{e_k} = \mathsf{e}_{v_1} - \mathsf{e}_{v_{K+1}},$$

where a_e is the column of e in the vertex-edge incidence matrix A, and e_v is the v-th unit vector, i.e., all zeroes except for a 1 at index v

Paths and Cycles

- **Prop.** Let $P = (v_1, e_1, v_2, \ldots, v_K, e_K, v_{K+1})$ be a path. Then

$$\sum_{k=1}^{K} O_P(e_k) \cdot a_{e_k} = \mathsf{e}_{v_1} - \mathsf{e}_{v_{K+1}},$$

where a_e is the column of e in the vertex-edge incidence matrix A, and e_v is the v-th unit vector, i.e., all zeroes except for a 1 at index v

Proof. Let k be s.t. $1 \leq k \leq K$. There are two cases:

1. If $e_k = (v_k, v_{k+1})$ then $a_{e_k} = \mathsf{e}_{v_k} - \mathsf{e}_{v_{k+1}}$ and $O_P(e_k) = 1$
2. If $e_k = (v_{k+1}, v_k)$ then $a_{e_k} = \mathsf{e}_{v_{k+1}} - \mathsf{e}_{v_k}$ and $O_P(e_k) = -1$

In any case $O_P(e_k) \cdot a_{e_k} = \mathsf{e}_{v_k} - \mathsf{e}_{v_{k+1}}$. So

$$\sum_{k=1}^{K} O_P(e_k) \cdot a_{e_k} = (\mathsf{e}_{v_1} - \mathsf{e}_{v_2}) + (\mathsf{e}_{v_2} - \mathsf{e}_{v_3}) + \ldots + (\mathsf{e}_{v_K} - \mathsf{e}_{v_{K+1}}) = \mathsf{e}_{v_1} - \mathsf{e}_{v_{K+1}}$$

Paths and Cycles

- **Prop.** Let $P = (v_1, e_1, v_2, \ldots, v_K, e_K, v_{K+1})$ be a path. Then

$$\sum_{k=1}^{K} O_P(e_k) \cdot a_{e_k} = \mathsf{e}_{v_1} - \mathsf{e}_{v_{K+1}},$$

  where a_e is the column of e in the vertex-edge incidence matrix A, and e_v is the v-th unit vector, i.e., all zeroes except for a 1 at index v

- **Cor.** If $C = (v_1, e_1, v_2, \ldots, v_K, e_K, v_{K+1})$ is a cycle, the columns $a_{e_1}, a_{e_2}, \ldots, a_{e_K}$ of A are linearly dependent.

Paths and Cycles

- **Prop.** Let $P = (v_1, e_1, v_2, \ldots, v_K, e_K, v_{K+1})$ be a path. Then

$$\sum_{k=1}^{K} O_P(e_k) \cdot a_{e_k} = \mathsf{e}_{v_1} - \mathsf{e}_{v_{K+1}},$$

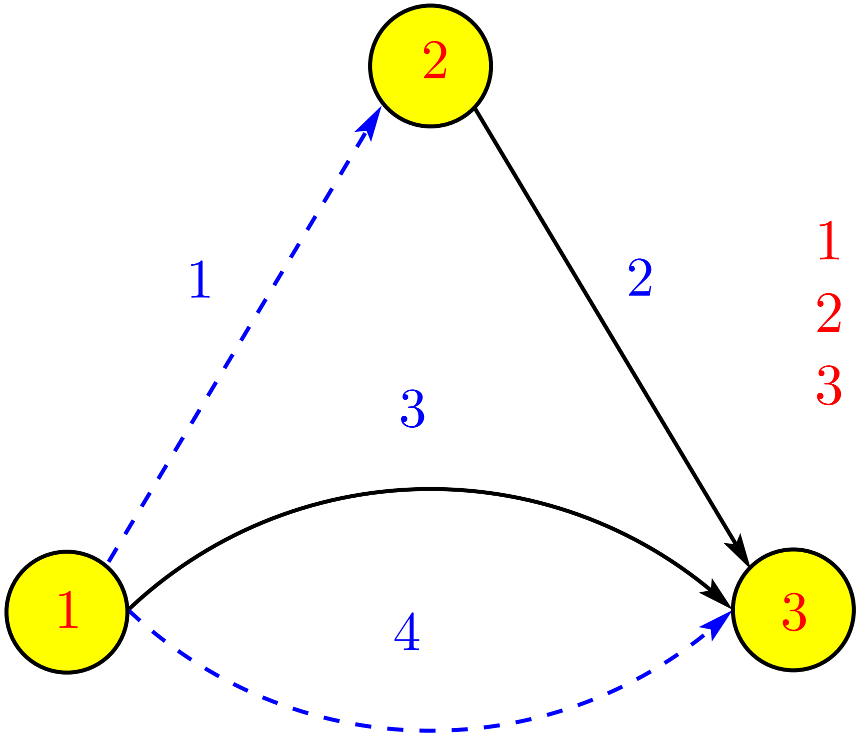
  where a_e is the column of e in the vertex-edge incidence matrix A, and e_v is the v-th unit vector, i.e., all zeroes except for a 1 at index v

- **Cor.** If $C = (v_1, e_1, v_2, \ldots, v_K, e_K, v_{K+1})$ is a cycle, the columns $a_{e_1}, a_{e_2}, \ldots, a_{e_K}$ of A are linearly dependent.

  Proof. If $v_1 = v_{K+1}$ then

$$\sum_{k=1}^{K} O_P(e_k) \cdot a_{e_k} = \mathsf{e}_{v_1} - \mathsf{e}_{v_{K+1}} = 0$$

Paths and Cycles



$$\begin{array}{c c} & \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \end{array} & \begin{pmatrix} 1 & 0 & 1 & 1 \\ -1 & 1 & 0 & 0 \\ 0 & -1 & -1 & -1 \end{pmatrix} \end{array}$$

Path $P = (3, 4, 1, 1, 2)$ has orientation sequence $(-1, 1)$

$$\sum_{k=1}^{K} O_P(e_k) \cdot a_{e_k} = (-1) \begin{pmatrix} 1 \\ 0 \\ -1 \end{pmatrix} + (1) \begin{pmatrix} 1 \\ -1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ -1 \\ 1 \end{pmatrix} = \mathsf{e}_3 - \mathsf{e}_2$$

Trees

- A graph is
  - acyclic if it has no cycles
  - connected if for any pair of vertices u, v there is a path from u to v
  - a tree if it is acyclic and connected
- **Thm.** For a graph T with at least one vertex the following are equivalent:
  - T is a tree
  - For any pair of vertices u, v there is a unique path from u to v
  - T has one less edge than vertices and is connected
  - T has one less edge than vertices and is acyclic
- A subgraph S of G is spanning if it covers all vertices in G
- **Thm.** Every connected graph has a subgraph that is a spanning tree.

Trees

- **Thm.** For any T subgraph of G that is a tree with at least two vertices, the columns $\{a_e \mid e \in T\}$ of A are linearly independent.

Trees

- **Thm.** For any T subgraph of G that is a tree with at least two vertices, the columns $\{a_e \mid e \in T\}$ of A are linearly independent.

  Proof. By contradiction.

  Let T be a tree with the minimum number of vertices N such that $\{a_e \mid e \in T\}$ are linearly dependent, i.e., there are λ_e not all null s.t.

  $$\sum_{e \in T} \lambda_e a_e = 0$$

  If $N = 2$ then T would have one edge, say e, and $a_e \neq 0$

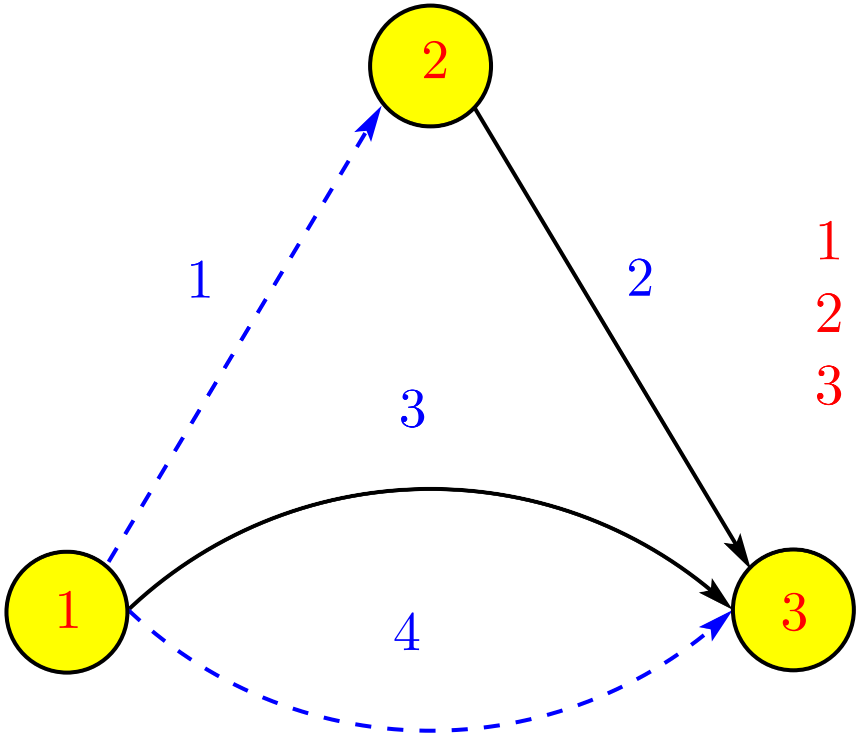
  So $N > 2$. Let v be a leaf of T and let e_v be the only edge in T that has v as an endpoint. Let T' be the tree obtained from T by removing e_v. From

  $$\lambda_{e_v} a_{e_v} + \sum_{e \in T, e \neq e_v} \lambda_e a_e = 0$$

  by projecting onto the row of v we have $\lambda_{e_v} = 0$.

  Hence the tree T' is a subgraph of G with $N - 1 \geq 2$ vertices whose columns are linearly dependent. Contradiction!

Paths and Cycles



$$\begin{array}{c c} & \begin{array}{cccc} \color{blue}1 & \color{blue}2 & \color{blue}3 & \color{blue}4 \end{array} \\ \begin{array}{c} \color{red}1 \\ \color{red}2 \\ \color{red}3 \end{array} & \begin{pmatrix} 1 & 0 & 1 & 1 \\ -1 & 1 & 0 & 0 \\ 0 & -1 & -1 & -1 \end{pmatrix} \end{array}$$

Edges $\{4, 1\}$ induce a subgraph that is a tree, and

$$\operatorname{rank}\begin{pmatrix} 1 & 1 \\ 0 & -1 \\ -1 & 0 \end{pmatrix} = 2$$

Reformulating Network Programs

- **Thm.** If G is a connected graph with $n > 0$ nodes then $\operatorname{rank}(A) = n - 1$

Reformulating Network Programs

- **Thm.** If G is a connected graph with $n > 0$ nodes then $\operatorname{rank}(A) = n - 1$

  Proof. G has a spanning tree T, which has $n - 1$ edges.

  Its columns are linearly independent, so $\operatorname{rank}(A) \geq n - 1$.

  But since adding all rows of A we get 0, finally $\operatorname{rank}(A) = n - 1$.

Reformulating Network Programs

- **Thm.** If G is a connected graph with $n > 0$ nodes then $\mathrm{rank}(A) = n - 1$

- Let us assume graphs of network programs are connected, so $m \geq n - 1$ (otherwise, work independently on the connected components)
- So the matrix of a network program has rank $n - 1$.
  But the simplex method requires to have a full-rank matrix!
- We add an extra variable w with a unit column e_r, where r is taken arbitrarily from $\{1, \ldots, n\}$, and such that it is forced to have value 0:

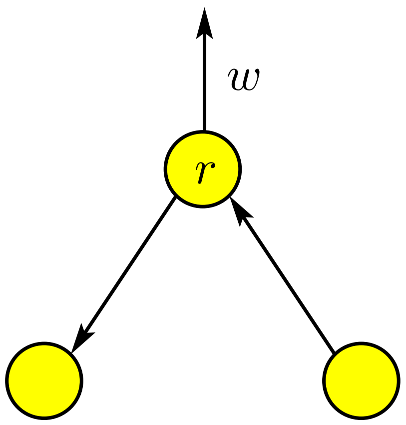
$$\begin{aligned} &\min \ c^T x \\ &Ax + \mathsf{e}_r\, w = b \\ &\ell \leq x \leq u, \\ &0 \leq w \leq 0 \end{aligned}$$

Reformulating Network Programs

- **Thm.** If G is a connected graph with $n > 0$ nodes then $\mathrm{rank}(A) = n - 1$

- Let us assume graphs of network programs are connected, so $m \geq n - 1$ (otherwise, work independently on the connected components)
- So the matrix of a network program has rank $n - 1$.
  But the simplex method requires to have a full-rank matrix!
- We add an extra variable w with a unit column e_r, where r is taken arbitrarily from $\{1, \ldots, n\}$, and such that it is forced to have value 0:

$$
\begin{aligned}
&\min\ c^T x \\
&Ax + \mathsf{e}_r\, w = b \\
&\ell \leq x \leq u, \\
&0 \leq w \leq 0
\end{aligned}
$$



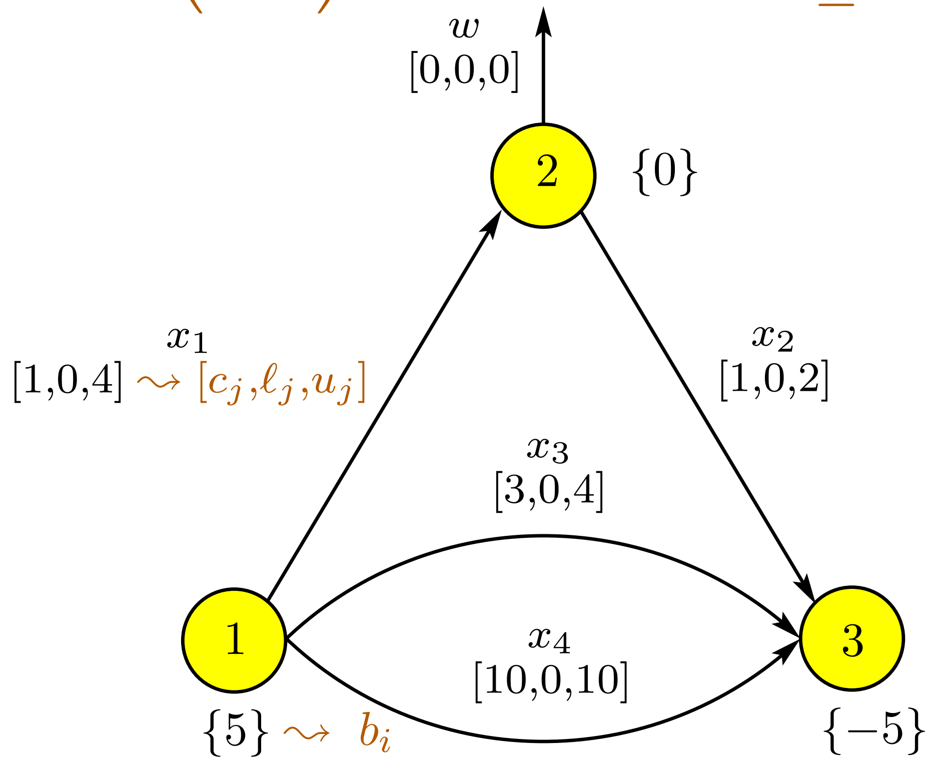
- We associate to such a reformulated network program a rooted graph with root vertex r and root edge w ("going nowhere")

Reformulating Network Programs

Here we choose as a root vertex $r = 2$

$$\min \; x_1 + x_2 + 3x_3 + 10x_4$$

$$\begin{pmatrix} 1 & 0 & 1 & 1 & 0 \\ -1 & 1 & 0 & 0 & 1 \\ 0 & -1 & -1 & -1 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ w \end{pmatrix} = \begin{pmatrix} 5 \\ 0 \\ -5 \end{pmatrix} \qquad \begin{array}{l} 0 \le x_1 \le 4 \\ 0 \le x_2 \le 2 \\ 0 \le x_3 \le 4 \\ 0 \le x_4 \le 10 \\ 0 \le w \le 0 \end{array}$$



Characterization of Bases

- **Thm.** Let A be the matrix of a rooted graph G with root vertex r.
  If T is a spanning tree for G then $B = \mathsf{e}_r \cup \{a_e \,|\, e \in T\}$ is basis of $(A \,|\, \mathsf{e}_r)$

Characterization of Bases

- **Thm.** Let A be the matrix of a rooted graph G with root vertex r. If T is a spanning tree for G then $B = \mathsf{e}_r \cup \{a_e \,|\, e \in T\}$ is basis of $(A \,|\, \mathsf{e}_r)$

  Proof. Let n be the number of vertices of G. As T is a spanning tree, T has $n-1$ edges. Hence $B = \mathsf{e}_r \cup \{a_e \,|\, e \in T\}$ has n columns.

  Let us prove that B spans $\mathbb{R}^n$, i.e., that for any $1 \leq i \leq n$ we can write e_i as linear combination of columns of B

  Two cases:

  - If $i = r$: trivial
  - If $i \neq r$, let $P = (v_1 = i, e_1, v_2, \ldots, v_K, e_K, v_{K+1} = r)$ be a path in T from vertex i to vertex r. As

    $$\sum_{k=1}^{K} O_P(e_k) \cdot a_{e_k} = \mathsf{e}_i - \mathsf{e}_r$$

    we have

    $$\mathsf{e}_r + \sum_{k=1}^{K} O_P(e_k) \cdot a_{e_k} = \mathsf{e}_i$$

  Altogether B is a basis for $(A \,|\, \mathsf{e}_r)$

Characterization of Bases

- **Thm.** Let A be the matrix of a rooted graph G with root vertex r. If T is a spanning tree for G then $B = \mathsf{e}_r \cup \{a_e \,|\, e \in T\}$ is basis of $(A \,|\, \mathsf{e}_r)$

  Proof. Let n be the number of vertices of G. As T is a spanning tree, T has $n-1$ edges. Hence $B = \mathsf{e}_r \cup \{a_e \,|\, e \in T\}$ has n columns.

  Let us prove that B spans $\mathbb{R}^n$, i.e., that for any $1 \leq i \leq n$ we can write e_i as linear combination of columns of B

  Two cases:

  - If $i = r$: trivial
  - If $i \neq r$, let $P = (v_1 = i, e_1, v_2, \ldots, v_K, e_K, v_{K+1} = r)$ be a path in T from vertex i to vertex r. As

    $$\sum_{k=1}^{K} O_P(e_k) \cdot a_{e_k} = \mathsf{e}_i - \mathsf{e}_r$$

    we have

    $$\mathsf{e}_r + \sum_{k=1}^{K} O_P(e_k) \cdot a_{e_k} = \mathsf{e}_i$$

  Altogether B is a basis for $(A \,|\, \mathsf{e}_r)$

- **Cor.** $\operatorname{rank}(A \,|\, \mathsf{e}_r) = n$

Characterization of Bases

- **Thm.** Let A be the matrix of a rooted graph G with root vertex r.
  If B is basis of $(A \,|\, \mathsf{e}_r)$ then $\mathsf{e}_r \in B$ and $\{e \,|\, a_e \in B\}$ is spanning tree of G

Characterization of Bases

- **Thm.** Let A be the matrix of a rooted graph G with root vertex r.
  If B is basis of $(A \,|\, \mathsf{e}_r)$ then $\mathsf{e}_r \in B$ and $\{e \,|\, a_e \in B\}$ is spanning tree of G

  Proof. Let n be the number of vertices of G as usual.

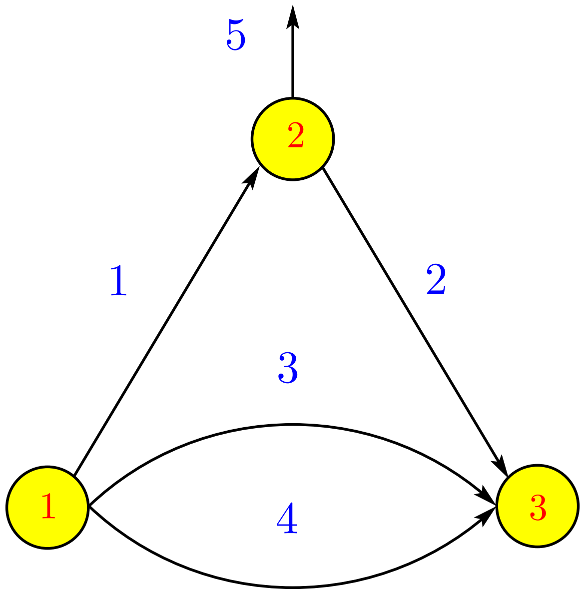
  Since $\mathrm{rank}(A) = n - 1$ and $\mathrm{rank}(A \,|\, \mathsf{e}_r) = n$ we have that $\mathsf{e}_r \in B$.
  So the graph T induced by $\{e \,|\, a_e \in B\}$ has $n - 1$ edges.

  Moreover, by linear independence, T cannot contain cycles.
  Hence T has at least $(n - 1) + 1 = n$ vertices. But G has n vertices.
  Thus T has exactly n vertices, and so is spanning.

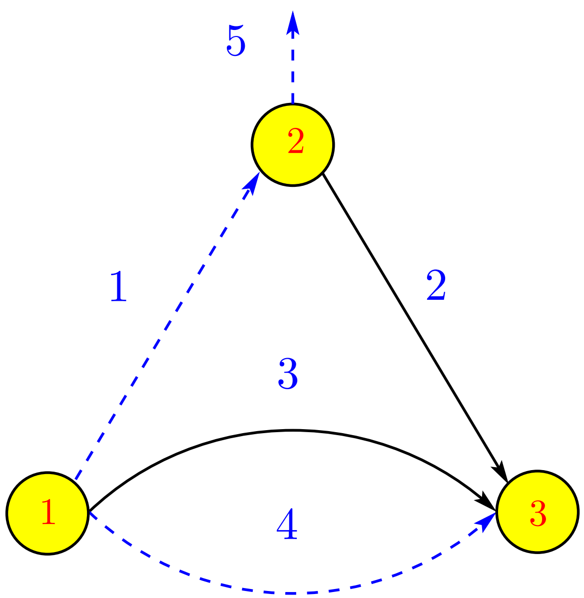
  Since T has one less edge than vertex and is acyclic, it must be a tree.

  All in all, T is a spanning tree.

Characterization of Bases

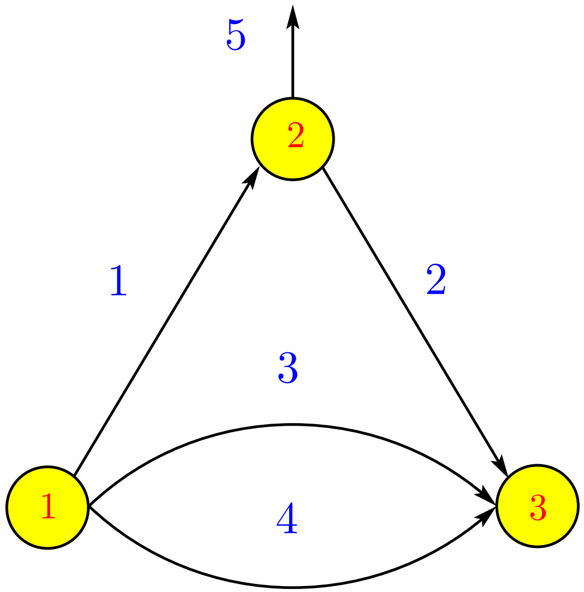


$$\begin{array}{c c} & \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \end{array} & \left(\begin{array}{rrrrr} 1 & 0 & 1 & 1 & 0 \\ -1 & 1 & 0 & 0 & 1 \\ 0 & -1 & -1 & -1 & 0 \end{array} \right) \end{array}$$

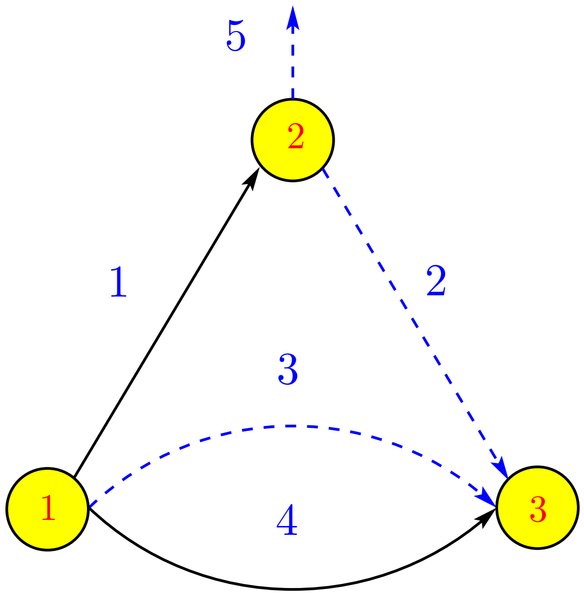


$$B = \left(\begin{array}{rrr} 1 & 1 & 0 \\ -1 & 0 & 1 \\ 0 & -1 & 0 \end{array} \right)$$

Characterization of Bases



$$\begin{array}{c c} & \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \end{array} & \begin{pmatrix} 1 & 0 & 1 & 1 & 0 \\ -1 & 1 & 0 & 0 & 1 \\ 0 & -1 & -1 & -1 & 0 \end{pmatrix} \end{array}$$



$$B = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ -1 & -1 & 0 \end{pmatrix}$$

Specializing the Simplex Method

- Where do we use the basis inverse in the simplex method?

Specializing the Simplex Method

1. `Initialization:` Find an initial feasible basis B
   Compute $B^{-1}, \beta = B^{-1}b, z = c_{\mathcal{B}}^T\beta$

2. `Pricing:` Compute $\pi^T = c_{\mathcal{B}}^T B^{-1}$ and $d_j = c_j - \pi^T a_j$.
   If for all $j \in \mathcal{R}, d_j \geq 0$ then return OPTIMAL
   Else let q be such that $d_q < 0$. Compute $\alpha_q = B^{-1}a_q$

3. `Ratio test:` Compute $\mathcal{I} = \{i \mid 1 \leq i \leq m, \alpha_q^i > 0\}$.
   If $\mathcal{I} = \emptyset$ then return UNBOUNDED
   Else compute $\theta = \min_{i\in\mathcal{I}}(\frac{\beta_i}{\alpha_q^i})$ and p such that $\theta = \frac{\beta_p}{\alpha_q^p}$

4. `Update:`
   $\bar{\mathcal{B}} = \mathcal{B} - \{k_p\} \cup \{q\}$ $\qquad$ $\bar{B} = B + (a_q - a_{k_p})e_p^T$
   $\bar{\beta}_p = \theta$, $\quad \bar{\beta}_i = \beta_i - \theta\alpha_q^i$ if $i \neq p$ $\qquad$ $\bar{z} = z + \theta d_q$
   Go to 2.

Specializing the Simplex Method

- Where do we use the basis inverse in the simplex method?
  - In pricing: we compute the multipliers $\pi^T = c_{\mathcal{B}}^T B^{-1}$
  - In ratio test: we compute the q-th column of the tableau $\alpha_q = B^{-1} a_q$
  - In initialization: we compute the initial basic solution $\beta = B^{-1} b$
- Equivalently:
  - In pricing: we solve the equation $y^T B = c_{\mathcal{B}}^T$ (and then set $\pi = y$)
  - In ratio test: we solve the equation $Bx = a_q$ (and then set $\alpha_q = x$)
  - In initialization: we solve the equation $Bx = b$ (and then set $\beta = x$)
- These equations can be efficiently solved with the graph representation
- So the network simplex method doesn't require to maintain basis inverses

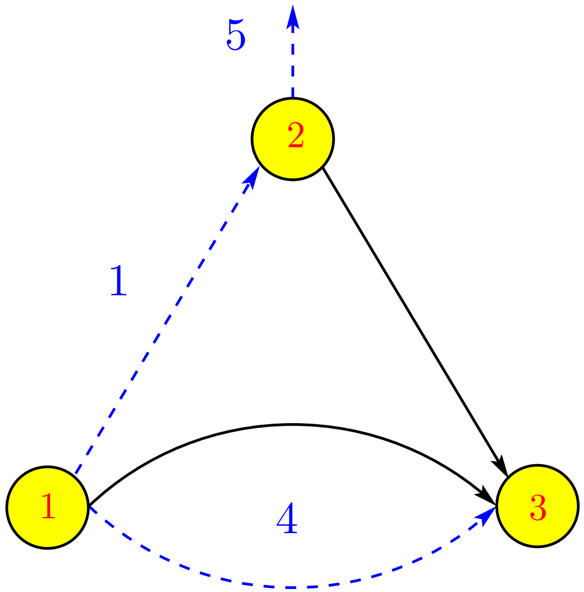
Solving $y^T B = c^T$

- Let A be the matrix of a rooted graph G with root vertex r.
  Let B be a basis for $(A \,|\, \mathsf{e}_r)$.
- We know that $\mathsf{e}_r \in B$ and $T = \{e \,|\, a_e \in B\}$ is a spanning tree for G.

- In the system of equations $y^T B = c^T$:
  - each column (= edge) of B corresponds to one equation
  - each row (= vertex) of B corresponds to one variable

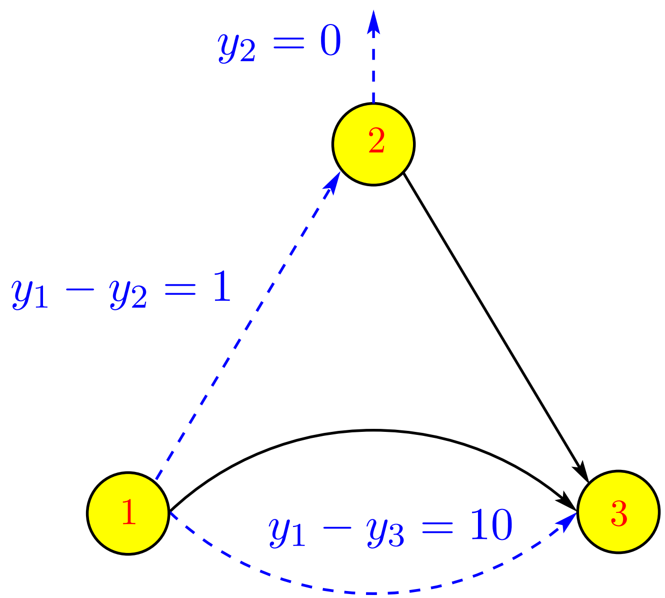
- Each equation either involves 1 variable (column e_r) or 2 (otherwise)

Solving $y^T B = c^T$



$$\begin{array}{cc} & \begin{array}{ccc} \color{blue}1 & \color{blue}4 & \color{blue}5 \end{array} \\ B = & \begin{pmatrix} 1 & 1 & 0 \\ -1 & 0 & 1 \\ 0 & -1 & 0 \end{pmatrix} \begin{array}{c} \color{red}1 \\ \color{red}2 \\ \color{red}3 \end{array} \end{array}$$

Solving $y^T B = c^T$



$$\begin{array}{cc}
 & \begin{array}{ccc} 1 & 4 & 5 \end{array} \\
B = & \begin{pmatrix} 1 & 1 & 0 \\ -1 & 0 & 1 \\ 0 & -1 & 0 \end{pmatrix} \begin{array}{c} 1 \\ 2 \\ 3 \end{array}
\end{array}$$

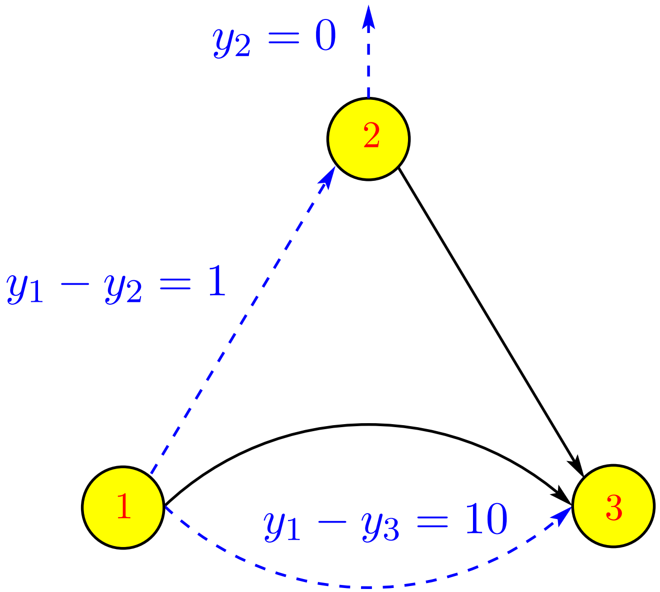
Let us solve $y^T B = c^T$, where $y^T = (y_1 \quad y_2 \quad y_3)$ and $c^T = (1 \quad 10 \quad 0)$

$$\begin{pmatrix} y_1 & y_2 & y_3 \end{pmatrix} \begin{pmatrix} 1 & 1 & 0 \\ -1 & 0 & 1 \\ 0 & -1 & 0 \end{pmatrix} = \begin{pmatrix} y_1 - y_2 & y_1 - y_3 & y_2 \end{pmatrix}$$

$$\left\{ \begin{array}{rcll} y_1 - y_2 & = & 1 & \rightsquigarrow 1 \\ y_1 - y_3 & = & 10 & \rightsquigarrow 4 \\ y_2 & = & 0 & \rightsquigarrow 5 \end{array} \right.$$

Note that by doing
a preorder traversal from root node 2
we can solve the equations

Solving $y^T B = c^T$



$$\begin{array}{cc} & \begin{array}{ccc} 1 & 4 & 5 \end{array} \\ B = & \begin{pmatrix} 1 & 1 & 0 \\ -1 & 0 & 1 \\ 0 & -1 & 0 \end{pmatrix} \end{array} \begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array}$$

Let us solve $y^T B = c^T$, where $y^T = (y_1 \quad y_2 \quad y_3)$ and $c^T = (1 \quad 10 \quad 0)$

$$\begin{pmatrix} y_1 & y_2 & y_3 \end{pmatrix} \begin{pmatrix} 1 & 1 & 0 \\ -1 & 0 & 1 \\ 0 & -1 & 0 \end{pmatrix} = \begin{pmatrix} y_1 - y_2 & y_1 - y_3 & y_2 \end{pmatrix}$$

$$\left\{ \begin{array}{rcll} y_1 - y_2 & = & 1 & \rightsquigarrow 1 \\ y_1 - y_3 & = & 10 & \rightsquigarrow 4 \\ y_2 & = & 0 & \rightsquigarrow 5 \end{array} \right. \qquad \begin{array}{rclcrcl} y_2 & = & 0 & & & & \\ y_1 - y_2 & = & 1 & \Longrightarrow & y_1 & = & y_2 + 1 = 1 \\ y_1 - y_3 & = & 10 & \Longrightarrow & y_3 & = & y_1 - 10 = -9 \end{array}$$

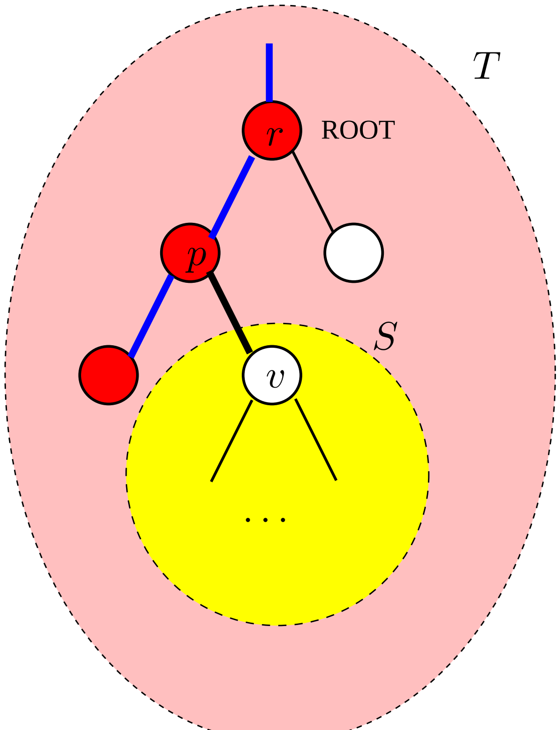
Solving $y^T B = c^T$

- Let us take the root vertex r as the root of T. Let w be the root edge.
- To solve $y^T B = c^T$ call *solve*$(\perp, T)$, where

```
solve(Vertex p, Tree S) { // p is the parent of the root of S
  Vertex v = root(S);
  if (v == r) y[r] = c[w];
  else if ((p, v) ∈ E) y[v] = y[p] − c[(p, v)];
  else                 y[v] = y[p] + c[(v, p)];
  solve(v, S.left());
  solve(v, S.right()); }
```

It is a preorder traversal of T.

At each recursive call (except 1st one) we handle a new equation ($=$ column $=$ edge) with 2 vars y_p and y_v in which one is already assigned (y_p) and the other is not (y_v).



Solving $y^T B = c^T$

- Let us take the root vertex r as the root of T. Let w be the root edge.
- To solve $y^T B = c^T$ call *solve*$(\perp, T)$, where

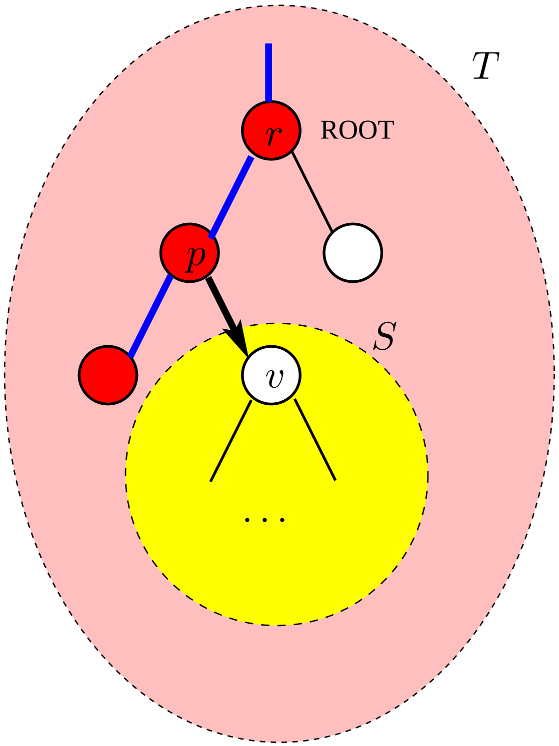
```
solve(Vertex p, Tree S) { // p is the parent of the root of S
  Vertex v = root(S);
  if (v == r) y[r] = c[w];
  else if ((p, v) ∈ E) y[v] = y[p] − c[(p, v)];
  else                 y[v] = y[p] + c[(v, p)];
  solve(v, S.left());
  solve(v, S.right()); }
```

If $v = r$ then the equation is $y^T \mathsf{e}_r = c_w$, i.e., $y_r = c_w$.

Solving $y^T B = c^T$

- Let us take the root vertex r as the root of T. Let w be the root edge.
- To solve $y^T B = c^T$ call *solve*$(\perp, T)$, where

```
solve(Vertex p, Tree S) { // p is the parent of the root of S
  Vertex v = root(S);
  if (v == r) y[r] = c[w];
  else if ((p, v) ∈ E) y[v] = y[p] − c[(p, v)];
  else                 y[v] = y[p] + c[(v, p)];
  solve(v, S.left());
  solve(v, S.right()); }
```



If $e = (p, v) \in E$ then the equation is

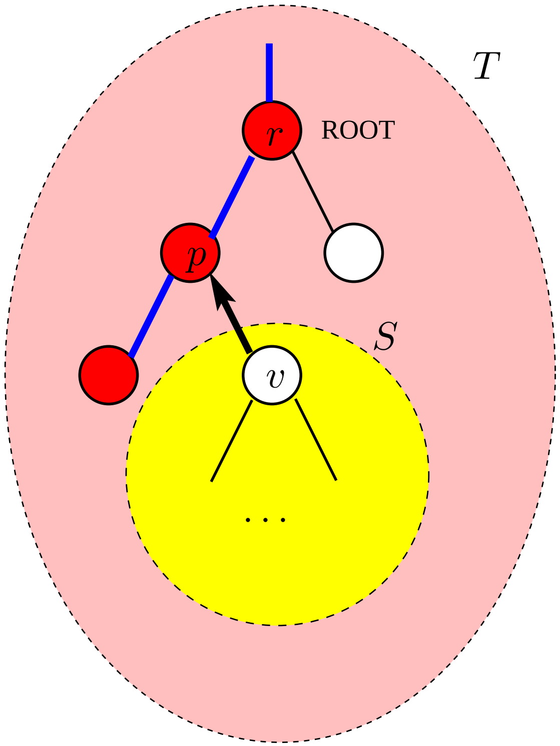
$$y^T(\mathbf{e}_p - \mathbf{e}_v) = y_p - y_v = c_e,$$

i.e., $y_v = y_p - c_e$.

Solving $y^T B = c^T$

- Let us take the root vertex r as the root of T. Let w be the root edge.
- To solve $y^T B = c^T$ call *solve*$(\perp, T)$, where

```
solve(Vertex p, Tree S) { // p is the parent of the root of S
  Vertex v = root(S);
  if (v == r) y[r] = c[w];
  else if ((p, v) ∈ E) y[v] = y[p] − c[(p, v)];
  else                 y[v] = y[p] + c[(v, p)];
  solve(v, S.left ());
  solve(v, S.right ()); }
```



If $e = (v, p) \in E$ then the equation is

$$y^T(\mathsf{e}_v - \mathsf{e}_p) = y_v - y_p = c_e,$$

i.e., $y_v = y_p + c_e$.

Solving $Bx = c$

- Let A be the matrix of a rooted graph G with root vertex r.
  Let B be a basis for $(A \,|\, \mathsf{e}_r)$.
- We know that $\mathsf{e}_r \in B$ and $T = \{e \,|\, a_e \in B\}$ is a spanning tree for G.
- For any $1 \leq i \leq n$ there is a path P_i from i to r, i.e., $P_i = (v_1 = i, e_1, ..., e_K, v_{K+1} = r)$ in T. But recall that

$$\mathsf{e}_i = \mathsf{e}_r + \textstyle\sum_{k=1}^{K} O_{P_i}(e_k) \cdot a_{e_k}$$

- Let us assume B is of the form $(a_{k_1}, a_{k_2}, \ldots, a_{k_{n-1}}, \mathsf{e}_r)$. Then

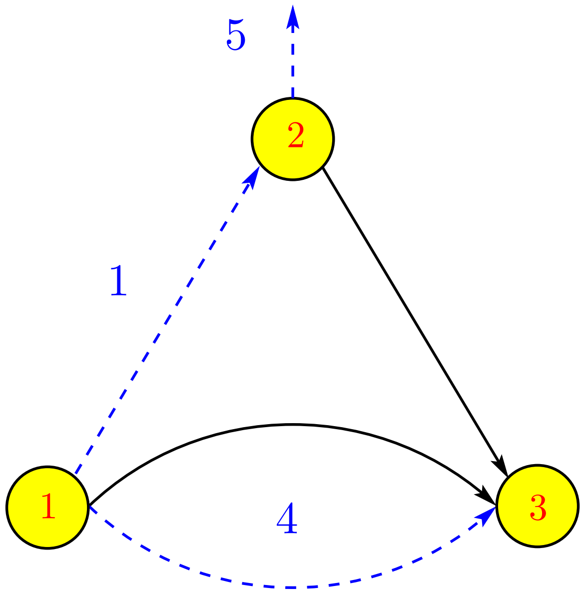
$$\mathsf{e}_i = \mathsf{e}_r + \textstyle\sum_{j=1}^{n-1} O_{P_i}(k_j) \cdot a_{k_j}$$

  as edges k_j not in P_i will have a 0 coefficient by definition of O_{P_i}. So

$$c = \textstyle\sum_{i=1}^{n} c_i \mathsf{e}_i = \left(\sum_{i=1}^{n} c_i\right) \mathsf{e}_r + \sum_{j=1}^{n-1} \left(\sum_{i=1}^{n} c_i \, O_{P_i}(k_j)\right) \cdot a_{k_j}$$

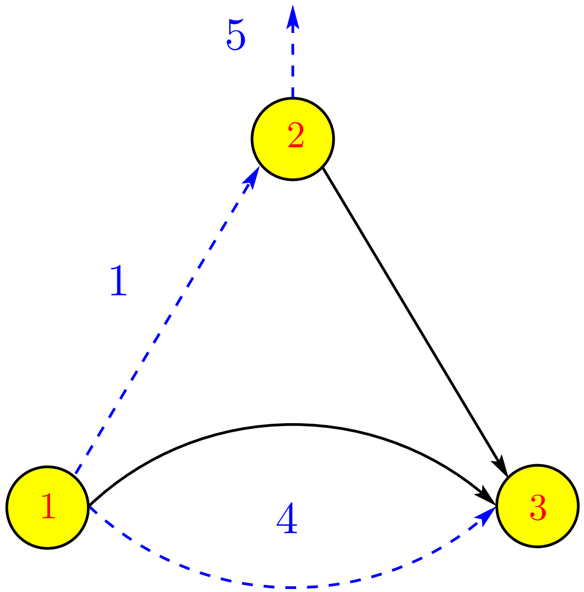
  Let $x_n = \sum_{i=1}^{n} c_i$, $x_j = \sum_{i=1}^{n} c_i \, O_{P_i}(k_j)$ for $1 \leq j < n$. Then $Bx = c$!
- Solving $Bx = c$ amounts to traverse T keeping track of edge orientation

Solving $Bx = c$



$$\begin{array}{cc} & \begin{array}{ccc} 1 & 4 & 5 \end{array} \\ B = & \begin{pmatrix} 1 & 1 & 0 \\ -1 & 0 & 1 \\ 0 & -1 & 0 \end{pmatrix} \end{array} \begin{array}{c} 1 \\ 2 \\ 3 \end{array}$$

Solving $Bx = c$



$$\begin{array}{cc} & \begin{array}{ccc} 1 & 4 & 5 \end{array} \\ B = & \begin{pmatrix} 1 & 1 & 0 \\ -1 & 0 & 1 \\ 0 & -1 & 0 \end{pmatrix} \begin{array}{c} 1 \\ 2 \\ 3 \end{array} \end{array}$$

Let us solve $Bx = c$, where $x^T = (x_1\ x_4\ x_5)^T$, and

$c^T = (c_1\ c_2\ c_3)^T = (0\ 1\ \ -1)^T = \mathsf{e}_2^T - \mathsf{e}_3^T$

There is no need to consider the path P_1 from 1 to 2, as $c_1 = 0$.

Moreover $P_2 = (2)$, and hence $O_{P_3}(\cdot) = 0$.

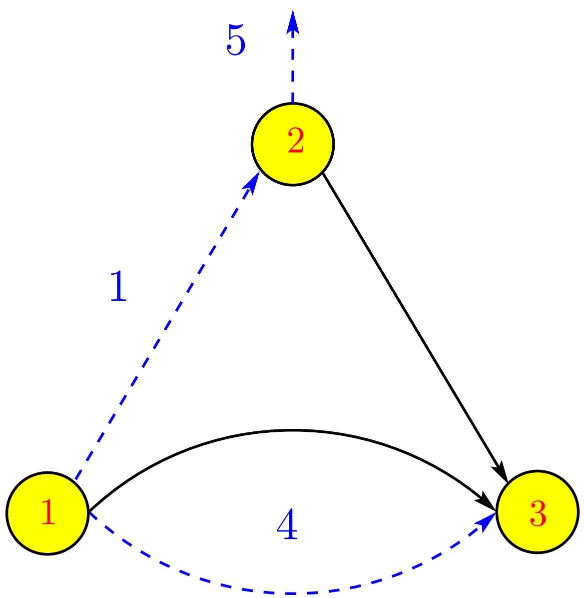
Path from 3 to 2: $P_3 = (3, 4, 1, 1, 2)$ with orientation sequence $(-1, 1)$.

- $x_1 = c_3 \cdot O_{P_3}(1) = (-1) \cdot 1 = -1$
- $x_4 = c_3 \cdot O_{P_3}(4) = (-1) \cdot (-1) = 1$
- $x_5 = c_1 + c_2 + c_3 = 0 + 1 + (-1) = 0$

Solving $Bx = c$

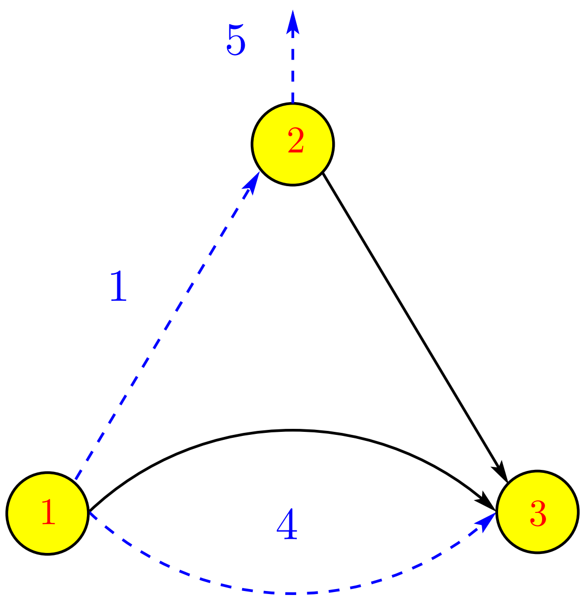
- Let A be the matrix of rooted graph G with root vertex r.
  Let B be a basis for $(A \,|\, \mathsf{e}_r)$.
- We know that $\mathsf{e}_r \in B$ and $T = \{e \,|\, a_e \in B\}$ is a spanning tree for G.
- In the ratio test, c will be one of the columns of A.
- If c is of the form $\mathsf{e}_i - \mathsf{e}_j$,
  let P be the path in T going from vertex i to vertex j.
  Then recall that
  $$\sum_{e \in P} O_P(e) \cdot a_e = \mathsf{e}_i - \mathsf{e}_j$$
- Hence the orientation sequence gives us already the solution.

Solving $Bx = c$



$$
\begin{array}{cc}
 & \begin{array}{ccc} \quad 1 & \quad 4 & 5 \end{array} \\
B = & \begin{pmatrix} 1 & 1 & 0 \\ -1 & 0 & 1 \\ 0 & -1 & 0 \end{pmatrix}
\end{array}
\begin{array}{c} 1 \\ 2 \\ 3 \end{array}
$$

Solving $Bx = c$



$$B = \begin{array}{c} \\ \\ \\ \end{array} \begin{array}{cccc} & 1 & 4 & 5 & \\ & \left(\begin{array}{rrr} 1 & 1 & 0 \\ -1 & 0 & 1 \\ 0 & -1 & 0 \end{array}\right) & & & \begin{array}{c} 1 \\ 2 \\ 3 \end{array} \end{array}$$

Let us solve $Bx = c$, where $x^T = (x_1 \ x_4 \ x_5)$, and
$c^T = (c_1 \ c_2 \ c_3) = (0 \ 1 \ -1) = \mathsf{e}_2^T - \mathsf{e}_3^T$

Path from 2 to 3: $P_3 = (2, 1, 1, 4, 3)$ with orientation sequence $(-1, 1)$. So:

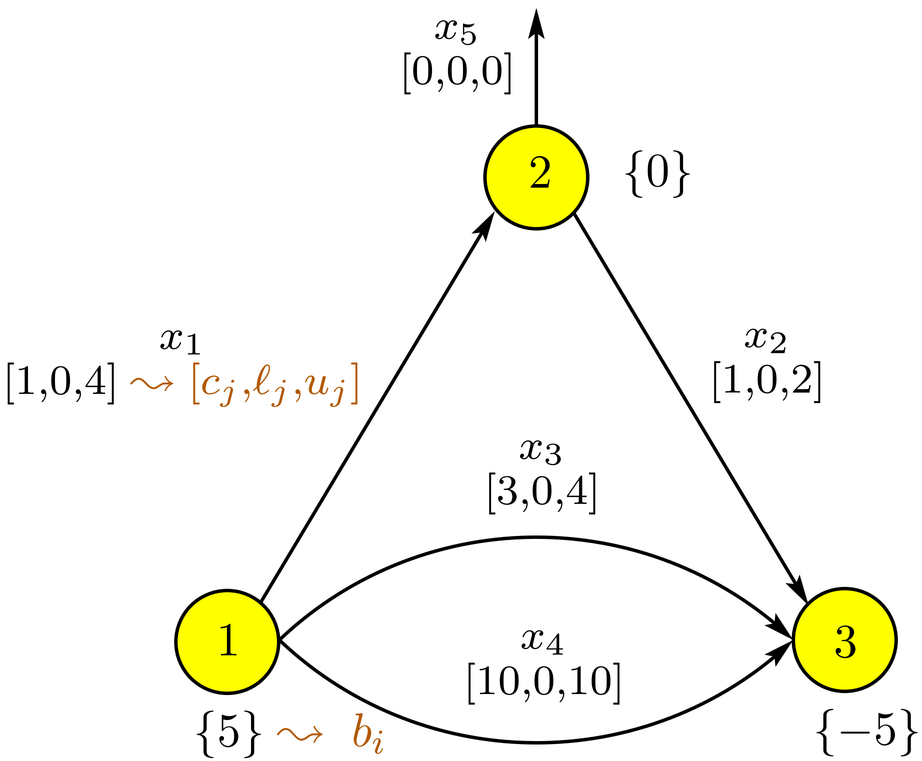
- $x_1 = -1$
- $x_4 = 1$
- $x_5 = 0$

Example

Let us apply one iteration of the simplex method to

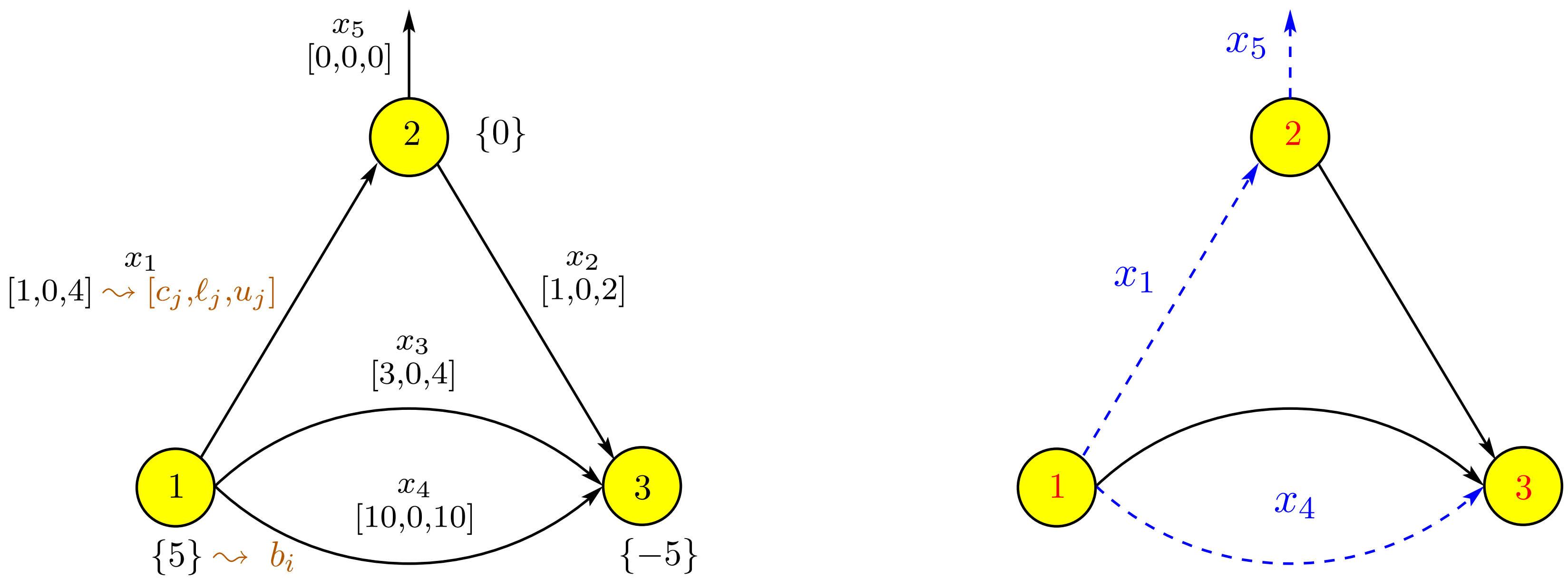
$$\min\ x_1 + x_2 + 3x_3 + 10x_4$$

$$\begin{pmatrix} 1 & 0 & 1 & 1 & 0 \\ -1 & 1 & 0 & 0 & 1 \\ 0 & -1 & -1 & -1 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} = \begin{pmatrix} 5 \\ 0 \\ -5 \end{pmatrix} \qquad \begin{array}{c} 0 \le x_1 \le 4 \\ 0 \le x_2 \le 2 \\ 0 \le x_3 \le 4 \\ 0 \le x_4 \le 10 \\ 0 \le x_5 \le 0 \end{array}$$



Example

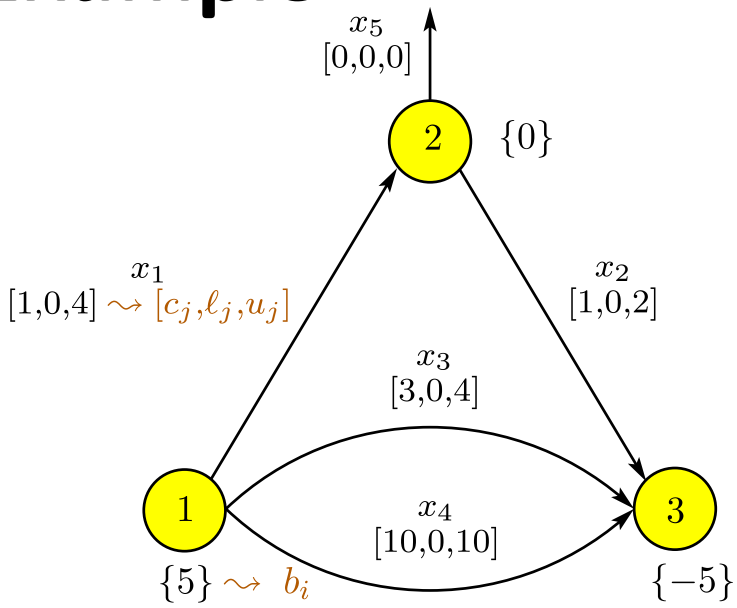
Let us consider the basis B corresponding to variables (x_1, x_4, x_5)

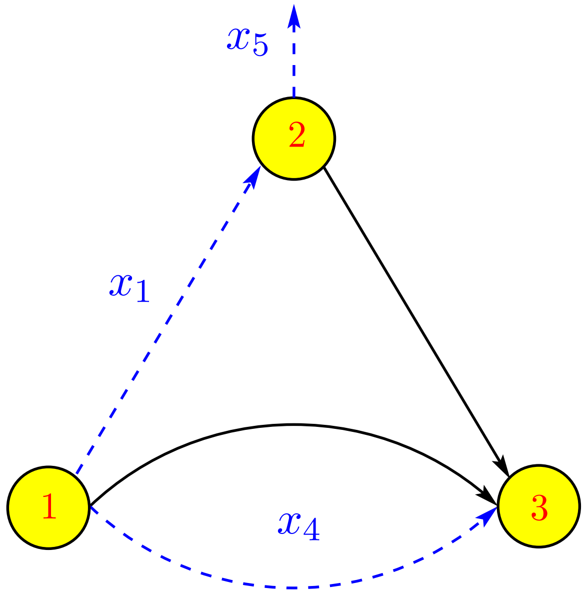


Moreover, let us assume that:

- non-basic variable x_2 is set to its lower bound 0
- non-basic variable x_3 is set to its upper bound 4

Example





- x_2: lower bound 0
- x_3: upper bound 4

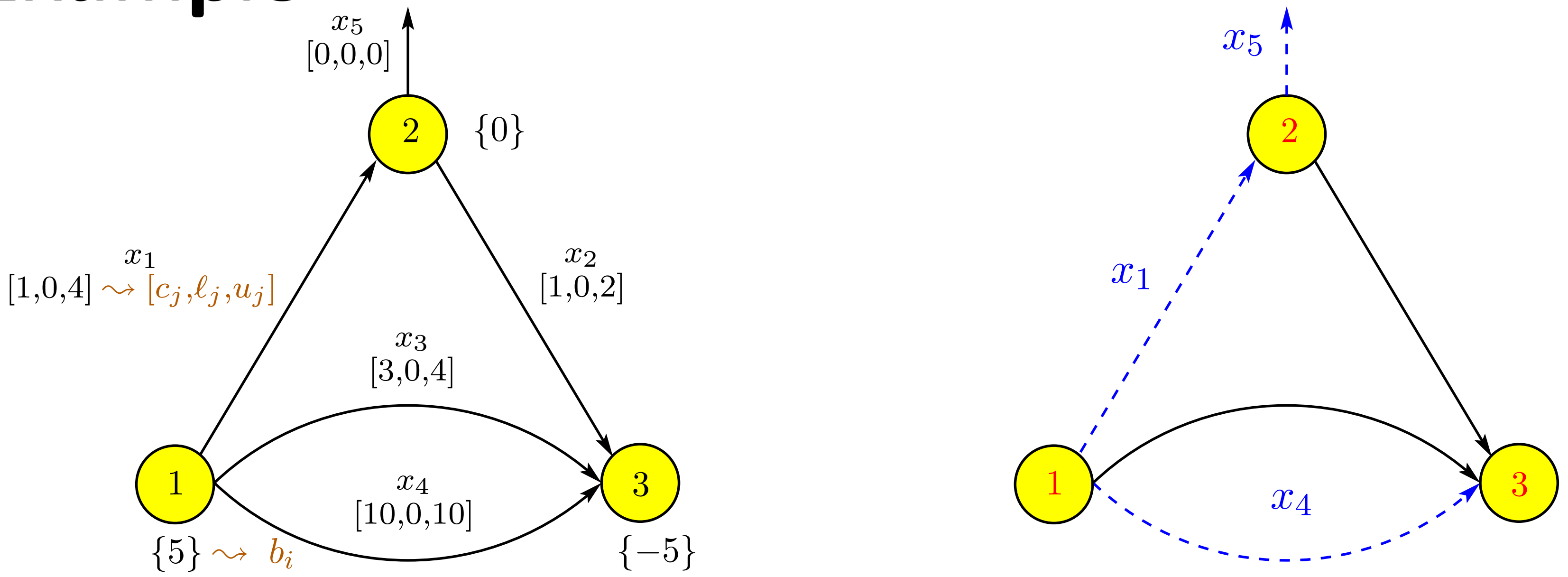
Let us compute the initial basic solution: $x_{\mathcal{B}} = B^{-1}b - B^{-1}R\,x_{\mathcal{R}}$

So $x_{\mathcal{B}} = B^{-1}(5\mathsf{e}_1 - 5\mathsf{e}_3) - B^{-1}a_2\,0 - B^{-1}a_3\,4 = 5B^{-1}(\mathsf{e}_1 - \mathsf{e}_3) - 4B^{-1}a_3$
$= B^{-1}(\mathsf{e}_1 - \mathsf{e}_3)$

The path from 1 to 3 is $P = (1, x_4, 3)$ with orientation sequence (1)
So the only non-zero value for a basic variable is for x_4, with value 1

Hence the basis is feasible and its solution is $(x_1, x_2, x_3, x_4, x_5) = (0, 0, 4, 1, 0)$

Example



- x_2: lower bound 0
- x_3: upper bound 4

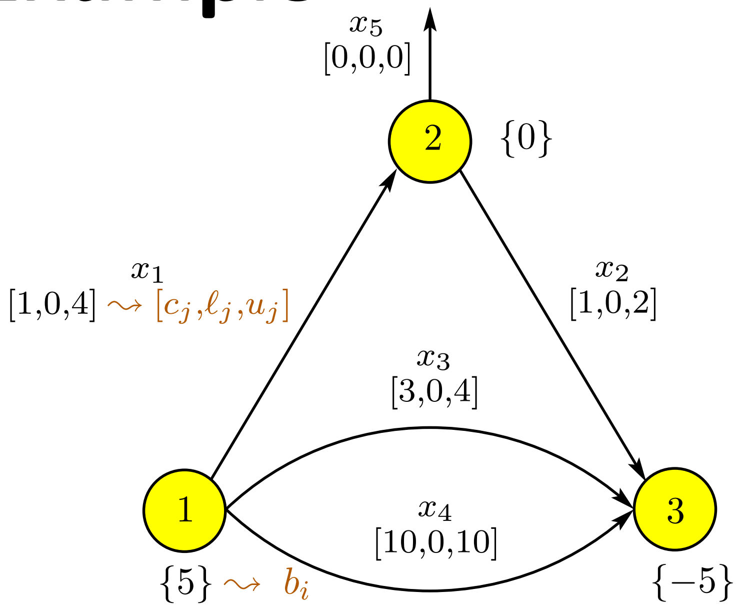
Let us do the pricing, i.e.,
compute $d_j = c_j - c_{\mathcal{B}}^T B^{-1} a_j = c_j - \pi^T a_j$ for each non-basic variable x_j

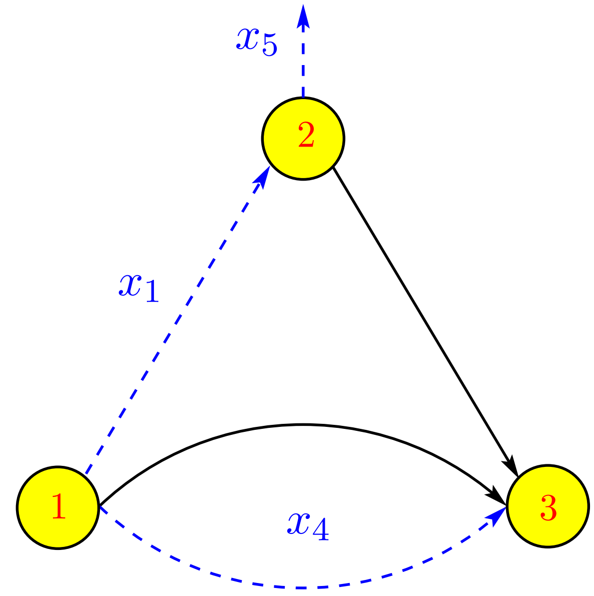
The solution to $\pi^T B = c_{\mathcal{B}}^T$ is $(\pi_1, \pi_2, \pi_3) = (1, 0, -9)$, and so:

- for x_2: $d_2 = c_2 - \pi^T(\mathsf{e}_2 - \mathsf{e}_3) = c_2 - \pi_2 + \pi_3 = -8$
- for x_3: $d_3 = c_3 - \pi^T(\mathsf{e}_1 - \mathsf{e}_3) = c_3 - \pi_1 + \pi_3 = -7$

Only variable x_2 is candidate for entering the basis

Example





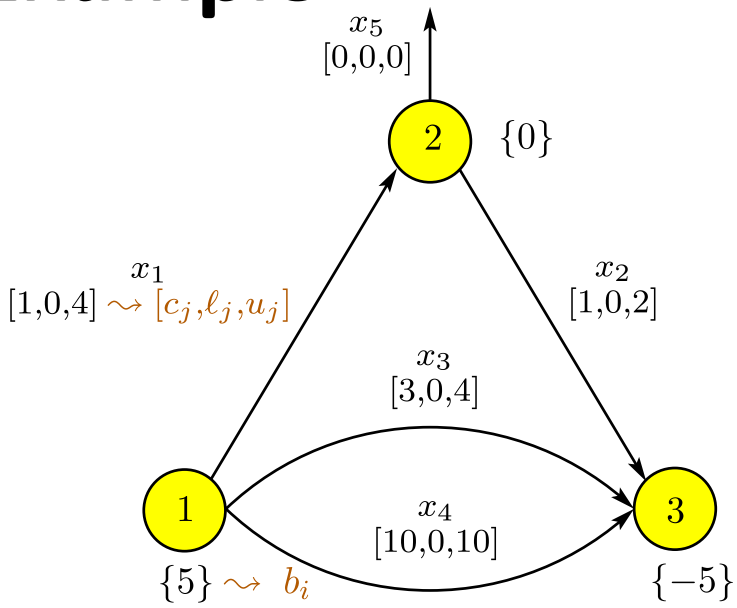
- x_2: lower bound 0
- x_3: upper bound 4
- $(x_1, x_2, x_3, x_4, x_5) = (0, 0, 4, 1, 0)$

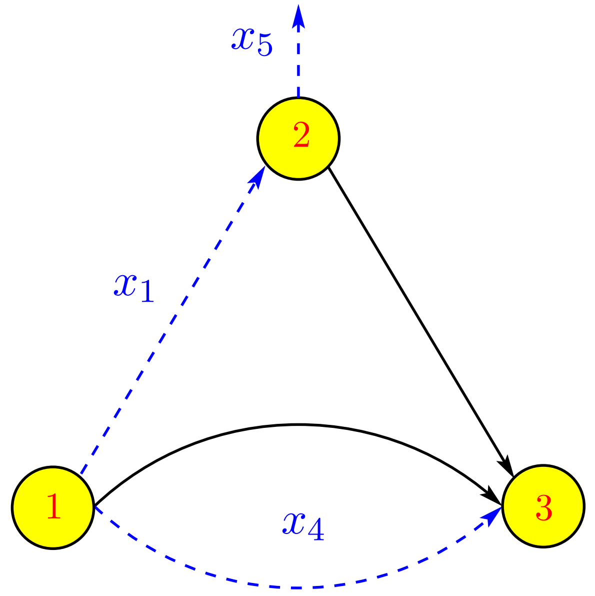
Let us do the ratio test.
We need to compute $\alpha_2 = B^{-1}a_2$, and we get $\alpha_2^T = (-1, 1, 0)$. Then

$$\theta = \min(u_q - \ell_q, \min\{\tfrac{\beta_i - \lambda_i}{\alpha_q^i} \mid \alpha_q^i > 0\}, \min\{\tfrac{\beta_i - \mu_i}{\alpha_q^i} \mid \alpha_q^i < 0\})$$
$$= \min(2, \tfrac{1-0}{1}, \tfrac{0-4}{-1}) = 1$$

The outgoing basic variable is x_4.

Example





- Non-basic variable x_2 enters the basis
- Basic variable x_4 leaves the basis with value 0
- New basis $\bar{B}$ corresponds to (x_1, x_2, x_5)
- New basic solution: $\bar{\beta}_p = x_q + \theta$, $\bar{\beta}_i = \beta_i - \theta \alpha_q^i$ if $i \neq p$
  - $\bar{x}_2 = 0 + 1 = 1$
  - $\bar{x}_1 = 0 - 1(-1) = 1$
  - $\bar{x}_5 = 0 - 1(0) = 0$
- The basic solution for the new basis is $(\bar{x}_1, \bar{x}_2, \bar{x}_3, \bar{x}_4, \bar{x}_5) = (1, 1, 4, 0, 0)$
  And the process continues...