COBRA=COmBinatorial optimisation and Related Algorithms

Rumen Andonov (Part 1) and Sophie Pinchinat (Part 2)

University of Rennes 1 and INRIA Rennes Bretagne-Atlantique

Contents of Part 1 (Rumen Andonov)

- Algorithms, S. Dasgupta, C. H. Papadimitriou, et U. V. Vazirani, McGraw-Hill 2006
- Graphes et algorithmes, Michel Gondran et Michel Minoux, Eyrolles, 1995
- Networks : An Introduction, M.E. Newman, 2010
- Aide à la décision : Une approche par les cas, Philippe Vallin, Daniel Vanderpooten, Ellipses, 2002
- Problems and exercises in Operations Research, Leo Liberti, Ecole Polytechnique, 2006
- Formulating logical implications in combinatorial optimisation, Frank Plastria, European Journal of Operational Research 140 (2002) 338–353

Modeling (formulating linear programs)

Exercise : Modeling in service of the elegance

An elegant and pretty student has to fly to the United States for her summer internship. Besides her professional possessions and a few necessary objects, she must decide upon a certain number of clothes from her wardrobe for her stay. Due to aerial rules, it turns out there are only 4 kilograms of clothes she is allowed to take on the plane. A first selection in her wardrobe leads to the student keeping, besides the dress she has decided to wear on the plane, 3 skirts, 3 (pairs of) pants, 4 tops and 3 dresses. The weight in grams of each piece of clothing is shown in Table (1). The student decides that :

1. she must wear at least one dress
2. if she takes the skirt $n^0 1$ then she will also take the top $n^0 2$ as it matches very well with that skirt
3. she won't take the top 4 if she takes the tops 1 and 2
4. if she takes no skirts, no pants, no tops, she needs to take at least one dress.
5. if she takes no skirts, she needs to take at least two pants.
6. if she doesn't take all her dresses, then she needs at least two skirts.

Modeling in service of the elegance II

Clothing	Skirt			Pants			Top				Dress		
	1	2	3	1	2	3	1	2	3	4	1	2	3
Weight (g)	500	400	700	600	500	500	400	300	300	400	600	700	800

Table 1: Weight in grams for each piece of clothing.

		top			
		1	2	3	4
skirt	1	+	+		+
	2	+			+
	3			+	
pants	1	+		+	
	2		+		
	3			+	+

Table 2: Acceptable combinations (outfits)

The goal is to maximize the number of different outfits she will be able to wear in the USA. A dress makes up an outfit. The other outfits will consist of a top paired with a skirt or pants. However, the rules of fashion allow only certain combinations, indicated by a cross in the table (2).

Exercise : Optimizing an anti-fire system

The port company COSMAR wishes to improve its anti-fire security by buying new anti-fire systems SLIC. The company identifies 7 zones of the port to be specially protected. These zones are denoted by X on the below figure.

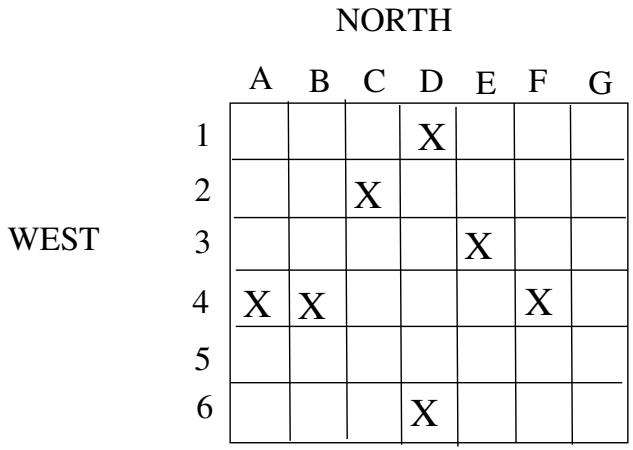


Figure 1: The 7 zones to be protected by the company COSMAR

COSMAR intends to allocate the SLIC systems on the North and West sides of the port. A SLIC system protects the zones situated on the same row or column. For example, a SLIC installed on D covers zones (1,D) et (6,D).

Optimizing an anti-fire system

Questions:

1. Formulate the corresponding linear program that allows to minimize the number of the anti-fire systems needed to cover the 7 special zones.
2. Add a constraint requiring that these anti-fire systems cannot be placed on consecutive columns, that at least one SLIC is allocated on row 1 or on column A, and that at least 60% of the SLIC are placed on the North side.
3. Add a constraint requiring that if no SLIC has been placed on a even-numbered row, then at least two SLICs need to be placed on odd-numbered rows.
4. Add a constraint requiring that if the number of SLICs placed on columns A,B,C,D is less than 2, then the number of SLICs placed on columns E, F, G must be at least 2.
5. Since the cost of one SLIC is 8070 euros, COSMAR wishes to buy at most 3 of them. For the non-protected zones COSMAR plans to take out insurance against fire. This insurance costs 2000 euros for each zone on lines 1, 2, 3 and 3000 euros for the other lines. Modify the previous linear model in order to minimize the total cost.
6. Are you aware of software packages permitting to solve such problems?

Tasks allocation

m tasks must be run on n CPUs which have the same performance. The length of each task is known (say L_i for the ith task). The goal is to allocate all tasks to the processors so that the completion time of the last task is minimized.

1. Formulate the corresponding linear program.
2. Add a constraint forcing task 2 to be executed on P_2 if task 1 is executed on P_1.
3. Add a constraint forcing tasks 3 and 4 to be both executed on P_1 if task 1 is executed on P_1.
4. Add a constraint forcing at least one of the processors to run no more than some given constant time E.
   Hint: *Denote by M an upper bound for the sum of task lengths and us it in a big-M constraint.*

Formulating logical implications as linear programs

Is modeling an "art"?

Yes, unfortunately modeling cannot be reduced to a standard set of procedure and sometimes can be quite discouraging. But some "tricks of the trade" do exist which are very helpful.

Basic Logical Implication Principle (LIP)

A *logical implication* is a property expressed as :
If A holds **then** B holds

Theorem 1 (LIP-Logical Implication Principle)

Let x_i be a 0-1 variable for all i in some finite index set I and y any variable satisfying $0 \leq y \leq 1$, then the logical implication
If $x_i = 0, \forall i \in I$ **then** $y = 0$
is exactly expressed by the inequality

$$y \leq \sum_{i \in I} x_i \tag{1}$$

Proof. Straightforward .

Question: How to adapt the theorem to the case : $0 \leq y \leq c$ where $c > 1$?

Using complementarity

Theorem 2

Let x_i be a 0-1 variable for all i in some finite index set I ; let I_0 and I_1 be two disjoint subsets of I and let y be an integer or continuous variable satisfying $0 \leq y \leq 1$. Then the logical implication
If $x_i = 0, \forall i \in I_0$ and $x_i = 1, \forall i \in I_1$ **then** $y = 0$
is exactly expressed by the inequality

$$y \leq \sum_{i \in I_0} x_i + \sum_{i \in I_1} (1 - x_i) \tag{2}$$

while the similar logical implication
If $x_i = 0, \forall i \in I_0$ and $x_i = 1, \forall i \in I_1$ **then** $y = 1$
is exactly expressed by the inequality

$$1 - y \leq \sum_{i \in I_0} x_i + \sum_{i \in I_1} (1 - x_i) \tag{3}$$

Proof. Direct application of Th. 1 to the 0-1 variables $x_i, i \in I_0$ and $(1 - x_i), i \in I_1$ and respectively y or $1 - y$.

Using complementarity II

Instead of remembering the theorem, it is easier to apply the following complementarity rule directly:

When the logical implication involves some $x_i = 1$ and/or $y = 1$ instead of 0, use (LIP) replacing each such x_i and/or y by its corresponding complementary variables $1 - x_i$ and/or $1 - y$.

Example : **If** $x_1 = 1$ and $x_2 = 0$ and $x_3 = 1$ **then** $y = 0$
is equivalent to
If $1 - x_1 = 0$ and $x_2 = 0$ and $1 - x_3 = 0$ **then** $y = 0$
which is expressed by

$$y \leq (1 - x_1) + x_2 + (1 - x_3) \tag{4}$$

or

$$x_1 - x_2 + x_3 + y \leq 2 \tag{5}$$

Linearization of quadratic 0-1 variables

In order to linearize, we often need to set $y_{ij} = x_i x_j$ where $y_{ij}, x_i, x_j \in \{0, 1\}$ which means exactly

$y_{ij} = 1$ **If and only if** $x_i = 1$ and $x_j = 1$.

In other words,
If $y_{ij} = 1$ **then** $x_i = 1$
If $y_{ij} = 1$ **then** $x_j = 1$
If $x_i = 1$ and $x_j = 1$ **then** $y_{ij} = 1$
We apply LIP to these three logical implications and obtain:

$$1 - x_i \leq 1 - y_{ij} \tag{6}$$

$$1 - x_j \leq 1 - y_{ij} \tag{7}$$

$$1 - y_{ij} \leq 1 - x_i + 1 - x_j \tag{8}$$

Other techniques for modeling

Write a model with objective function: $\min \max\{e_1, e_2, \ldots, e_n\}$ where $e_1, e_2, \ldots, e_n$ are linear functions.

- Introduce $y \in R$ s.t. $y \geq e_1, y \geq e_2, \ldots, y \geq e_n$. Then $\min y$.

Write a model with objective function: $\max \min\{e_1, e_2, \ldots, e_n\}$ where $e_1, e_2, \ldots, e_n$ are linear functions.

- Introduce $y \in R$ s.t. $y \leq e_1, y \leq e_2, \ldots, y \leq e_n$. Then $\max y$.

Write a model with objective function: $\min \mid e \mid$ where e is a linear functions

- Note that $\mid e \mid = \max\{e, -e\}$. Then use $\min - \max$ model.

Big-M constraints

Let $x \geq 0$ be such that, if $x > 0$ then $K \leq x \leq M$.

- Introduce $y \in \{0, 1\}$ and add the constraints: $x \leq My$ and $x \geq Ky$.

Let $0 \leq x \leq M$ and $0 \leq z \leq N$ represent incompatible activities. That is:
If $x > 0$ then $z = 0$ and (vice versa) if $z > 0$ then $x = 0$.

- Introduce $s, t \in \{0, 1\}$ and add the constraints: $x \leq Ms$, $z \leq Nt$ and $s + t = 1$.

Exclusive OR relation: Let the function $f(x_1, x_2, \ldots, x_n)$ be such that $\nu \leq f(x_1, x_2, \ldots, x_n) \leq M$ or (otherwise) $f(x_1, x_2, \ldots, x_n) \leq \mu$

Introduce $y \in \{0, 1\}$. Add the constraints:

$$f(x_1, x_2, \ldots, x_n) \leq \mu + My \text{ and } f(x_1, x_2, \ldots, x_n) \geq \nu - M(1 - y) \tag{9}$$

Linear Programming (LP)

Why ?

- This formalism successfully models a large number real situations.
- Efficient solvers like (IBM ILOG CPLEX, Gurobi, COIN, MINOS etc.) exist today.
- Modeling languages like "A Mathematical Programming Language (AMPL)" have been developped.
- Pyomo : Python-based, open-source optimization modeling language for formulating, solving, and analyzing optimization models.

Any linear program can be written in its **standard form** :

$$
\begin{array}{lrcl}
\max & c_1 x_1 + \cdots + c_n x_n & & \\
\text{s. c.} & a_{11} x_1 + \cdots + a_{1n} x_n & \leq & b_1 \\
 & \vdots \qquad\qquad\quad \vdots & & \vdots \\
 & a_{m1} x_1 + \cdots + a_{mn} x_n & \leq & b_m \\
 & x_1 \geq 0,\ x_2 \geq 0, \ldots,\ x_n \geq 0 & &
\end{array}
$$

A LP requires :

- an objective function (**linear**)
- $m+n$ constraints (**linear**) .
- n variables $x_i, i = 1, \ldots, n$

LP : matrix forms

$$\text{Maximize} \qquad \sum_{j=1}^{n} c_j x_j$$

$$\text{s. c.} \qquad \sum_{j=1}^{n} a_{ij} x_j \leq b_j, \qquad i = 1, \ldots, m$$

$$x_j \geq 0 \qquad j = 1, \ldots, n$$

or

$$\max\{cx \mid Ax \leq b,\ x \geq 0\}$$

where

$$A = \begin{pmatrix} a_{11} & \ldots & a_{1n} \\ \vdots & & \vdots \\ a_{m1} & \ldots & a_{mn} \end{pmatrix}, \ x = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}, \ b = \begin{pmatrix} b_1 \\ \vdots \\ b_m \end{pmatrix},$$

$$c = (c_1 \ldots c_n)$$

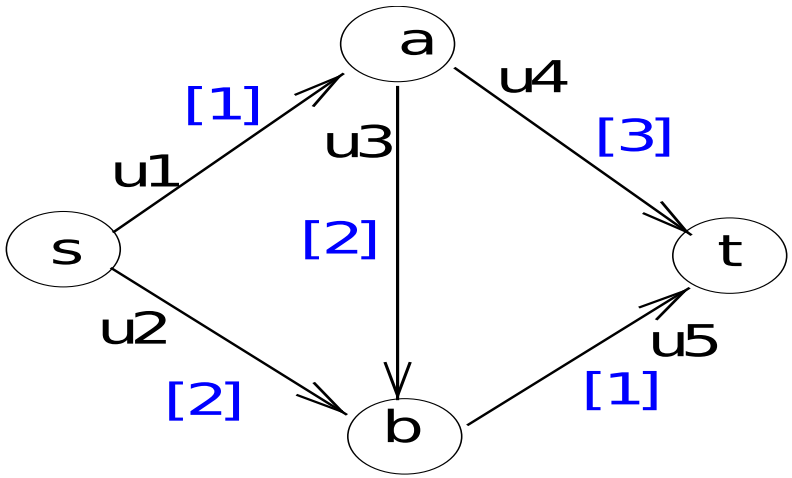
c is a given **objective**,
b is a given **right hand RHS**,
and $A^{m \times n}$ – **constraints matrix** .

LP formulation for Shortest/Longest path problem (SPP/LPP)

- Given a directed acyclic graph (DAG) $G = (V, E, s, t)$ where $m = |E|$, $n = |V|$ and weights w_{ij} associated with each arc $(i, j) \in E$. Find a path from s to t with the minimum/maximum total weight.
- Definition :
  The node-arc incidence matrix
  $A = [a_{i,j}]$ $i = 1, 2, \ldots, |V|$
  and $j = 1, 2, \ldots, |E|$ is defined by

$$a_{i,j} = \begin{cases} +1 & \text{if arc } e_j \text{ leaves node } i \\ -1 & \text{if arc } e_j \text{ enters node } i \\ 0 & \text{otherwise} \end{cases}$$

- Example



$$A = \begin{array}{c|ccccc} & u1 & u2 & u3 & u4 & u5 \\ \hline s & +1 & +1 & 0 & 0 & 0 \\ t & 0 & 0 & 0 & -1 & -1 \\ a & -1 & 0 & +1 & +1 & 0 \\ b & 0 & -1 & -1 & 0 & +1 \end{array}$$

A weighted directed graph and the corresponding node-arc incidence matrix A. The weights are in squared brackets.

- Exercise : Use the node-arc incidence matrix to describe the LP formulation of the above SPP/LPP.

LP formulation for Shortest/Longest path problem (SPP/LPP)

Any path from s to t can be represented by the vector $\mathbf{x}$ where $x_e = 1$ if the arc e belongs to the path, $x_e = 0$ otherwise and under the conditions

- flow conservation law for any intermediate vertex v :

$$\forall v \in V \sum_{(u,v)\in E} x_{uv} = \sum_{(v,u)\in E} x_{vu} \tag{10}$$

- a flow of value 1 exits the vertex s and enters the vertex t :

$$\sum_{(s,v)\in E} x_{sv} = 1 \text{ et } \sum_{(u,t)\in E} x_{ut} = 1 \tag{11}$$

- the goal is to maximize/minimize the total weight :

$$\max(\min) z = \sum_{(u,v)\in E} x_{uv} w_{uv} \tag{12}$$

Equations (10), (11) and (12) represent a linear program (LP).

LP formulation for Shortest/Longest path problem (SPP/LPP)

- Let A be the node-arc incidence matrix and denote by a_i its ith row. The flow conservation law for any intermediate vertex i is written :

$$a_i \mathbf{x} = 0 \tag{13}$$

- The linear program is :

  let $d \in R^n$ be defined as

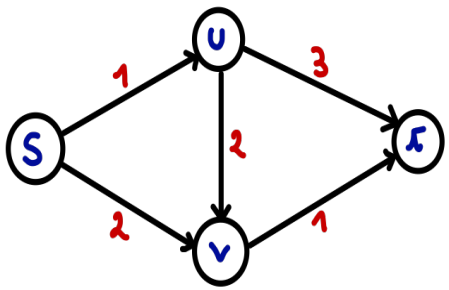
$$d_i = \begin{cases} +1 & i = s \\ -1 & i = t \\ 0 & \text{otherwise} \end{cases}$$

$$\max(\min) \quad \sum_{e \in E} x_e w_e \tag{14}$$

$$A\mathbf{x} = d \tag{15}$$

$$\forall e \in E, \quad x_e \in \{0, 1\} \tag{16}$$

Exemple:



$$A = \begin{array}{c|ccccc} & x_{su} & x_{sv} & x_{uv} & x_{ut} & x_{vt} \\ \hline s & +1 & +1 & 0 & 0 & 0 \\ t & 0 & 0 & 0 & -1 & -1 \\ a & -1 & 0 & +1 & +1 & 0 \\ b & 0 & -1 & -1 & 0 & +1 \end{array}$$

Values in red are the weights.

PL formulation versus AMPL code

- for more details see AMPL : STREAMLINED MODELING FOR REAL OPTIMIZATION (https://ampl.com)
- flow conservation law for any intermediate vertex v :

$$\forall u \in V \backslash \{s,t\} : \quad \sum_{(u,v)\in E} x_{uv} = \sum_{(v,u)\in E} x_{vu} \tag{17}$$

AMPL: conservation{v in V diff {start, target}} : sum{(u,v) in E} x[u,v] = sum{(v,u) in E} x[v,u];

- a flow of value 1 exits the vertex s and enters the vertex t :

$$\sum_{(s,v)\in E} x_{sv} = 1 \text{ et } \sum_{(u,t)\in E} x_{ut} = 1 \tag{18}$$

AMPL: source_only_one_output: sum{(start,v) in E} x[start, v] = 1;

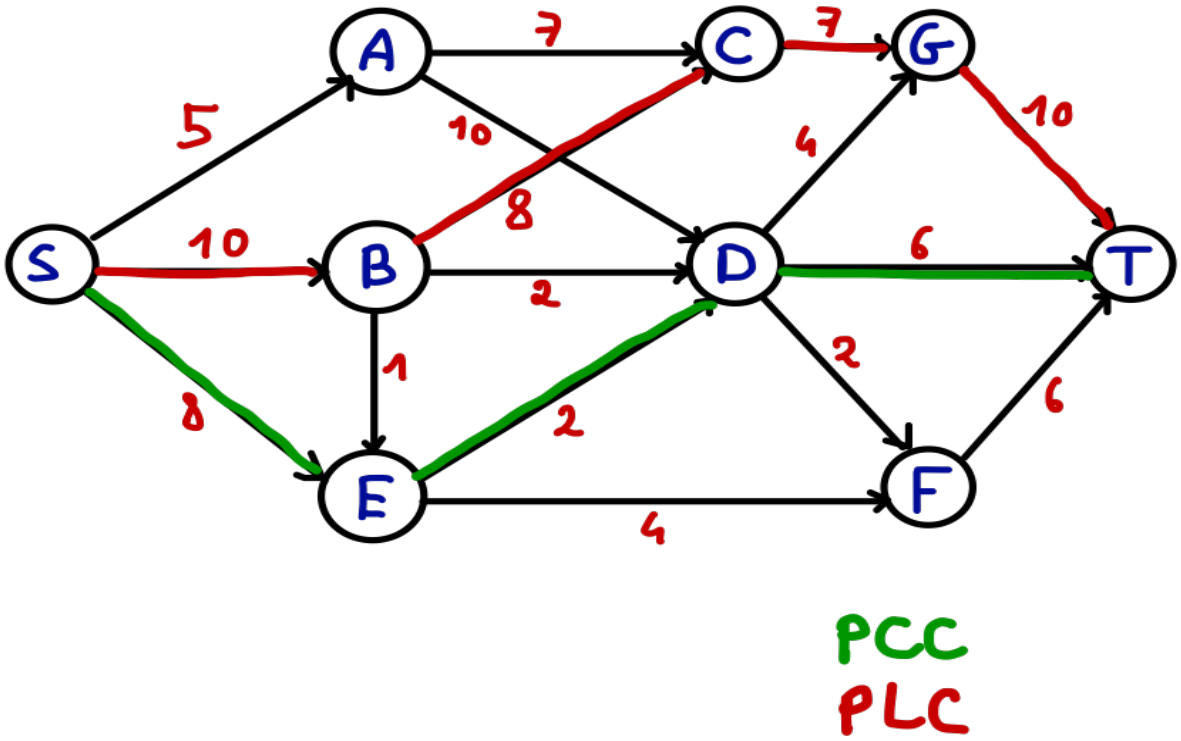
AMPL: target_only_one_input: sum{(u,target) in E} x[u, target] = 1;

- the goal is to maximize/minimize the total weight :

$$\max(\min) z = \sum_{(u,v)\in E} x_{uv} w_{uv} \tag{19}$$

AMPL: minimize pcc : sum{(u,v) in E} x[uv]*w[uv];

Exemple



Max-Flow Min-Cut (MFMC) Problem

- Let $N = (s, t, V, E, C)$ be a directed graph G with $n = |V|$ nodes, $m = |E|$ edges with edge-capacity $c(i, j) \geq 0$ and two special notes s, t which are, respectively a source and sink of G. Such graphs are called *networks* and we will be searching for feasible flows in networks.
- Denote by $\delta^+(u)$ (resp. $\delta^-(u)$) the set of output/input edges for the vertex u. A *feasible flow* is defined by a vector $\phi = [\phi_1, \phi_2, \ldots \phi_m]$, s.t. $\forall e \in E$, $0 \leq \phi_e \leq c_e$, and such that the flow conservation holds $\forall u \in V \backslash \{s, t\}$, i.e.

$$\sum_{e \in \delta^+(u)} \phi_e = \sum_{e \in \delta^-(u)} \phi_e \tag{20}$$

- By ϕ_0 we denote the size of a flow and therefore

$$\sum_{e \in \delta^+(s)} \phi_e = \sum_{e \in \delta^-(t)} \phi_e = \phi_0 \text{ (flow value)} \tag{21}$$

- The *Max-Flow Problem* consists in finding the flow that maximizes ϕ_0.

Modeling MFMC as a LP problem

The Max-Flow Problem reduces to linear programming (maximizing a linear objective function under a set of linear constraints).

- Let a_i denote the ith row of A. The flow conservation at a node i (except s, t) is expressed by :

$$a_i \phi = 0 \tag{22}$$

- The LP formulation of the MFMC is :

let $d \in R^n$ be defined by

$$d_i = \begin{cases} -1 & i = s \\ +1 & i = t \\ 0 & \text{otherwise} \end{cases}$$

$$\max \phi_0 \tag{23}$$

$$d\phi_0 + A\phi = 0 \tag{24}$$

$$\phi \leq c \tag{25}$$

$$\phi \geq 0 \tag{26}$$

$$\phi_0 \geq 0 \tag{27}$$

PL formulation versus AMPL code

- the conservation law : $\forall u \in V \backslash \{s, t\}$, i.e.

$$\sum_{(u,v)\in E} \phi_{(u,v)} = \sum_{(v,u)\in E} \phi_{(v,u)} \tag{28}$$

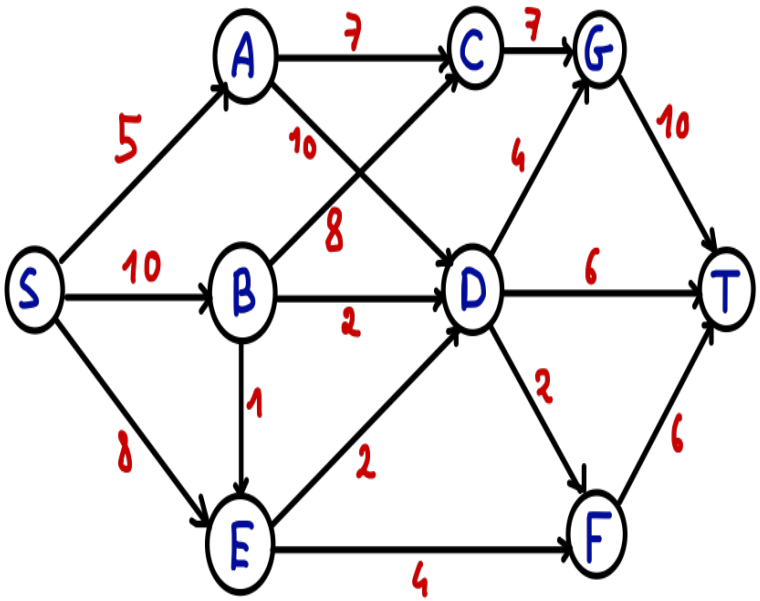
AMPL: conservation{v in V diff {start, target}} : sum{(u,v) in E} f[u,v] = sum{(v,u) in E} f[v,u];

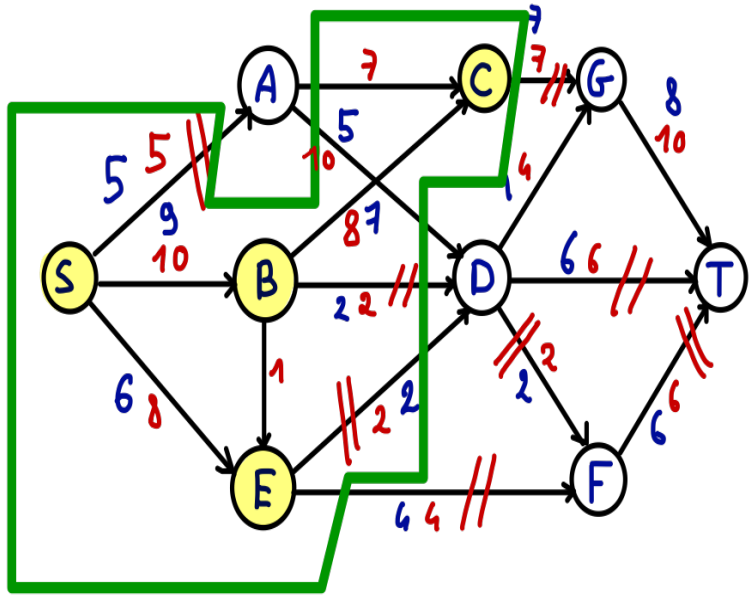
- Goal: Find a *compatible flow* $\phi' = [\phi_0, \phi_1, \phi_2, \ldots \phi_m]$ (i.e.

$$0 \leq \phi_u \leq c_u, \forall u =\in E \tag{29}$$

AMPL: limit_flow{(u,v) in E}: f[u,v] <= capacite_max[u,v];
such that ϕ_0 is *maximized* .
AMPL: maximize flow_max : sum{(u,target) in E} f[u, target];

Exemple





Exemple : manufacturing

A factory manufactures four products with profits $7, 9, 18$ et 17 € respectively. The factory uses three raw materials (ressources) A, B, C with availabilities $42, 17, 24$ units. The coefficients a_{ij} in the below table indicate the number of units of material i product j needs in ordre to be produced. The goal is to maximize the total profit.

	product 1	product 2	product 3	product 4	stock
ressource A	2	4	5	7	42
ressource B	1	1	2	2	17
ressource C	1	2	3	3	24
profit	7	9	18	17	

Exemple : manufacturing

Let x_i be the amount of product i produced.

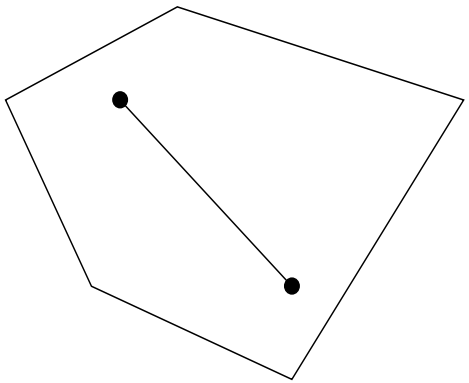
	product 1	product 2	product3	product 4	stock
ressource A	2	4	5	7	42
ressource B	1	1	2	2	17
ressource C	1	2	3	3	24
profit	7	9	18	17	

$$
\begin{array}{lrcrcrcrcr}
\text{Maximize } z = & 7x_1 & + & 9x_2 & + & 18x_3 & + & 17x_4 & & \\
\text{s. c.} & 2x_1 & + & 4x_2 & + & 5x_3 & + & 7x_4 & \leq & 42 \\
& x_1 & + & x_2 & + & 2x_3 & + & 2x_4 & \leq & 17 \\
& x_1 & + & 2x_2 & + & 3x_3 & + & 3x_4 & \leq & 24 \\
& x_1 & , & x_2 & , & x_3 & , & x_4 & \geq & 0
\end{array}
$$

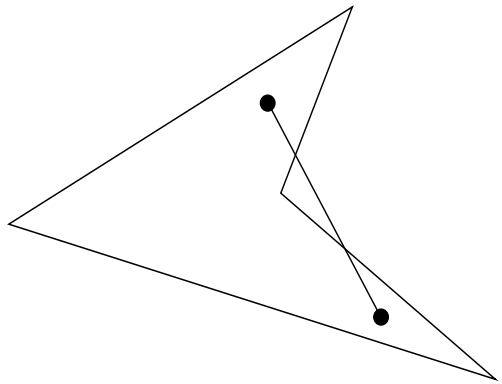
The above is a linear program in its *canonical form* .

Feasible region

The feasible region (set of *admissible/feasibles* solutions) is given by the intersection of a finite number of closed half-spaces. This region is a convex polyhedron
$P = \{x \mid Ax \leq b,\ x \geq 0\} \subseteq \mathbb{R}^n$.
P is a *convex polyhedron* dans $\mathbb{R}^n$ (if $x \in P$, $y \in P$ and $0 \leq \lambda \leq 1$ then $\lambda x + (1-\lambda)y \in P$)



convexe



non convexe

A point $x \in S$ is an *extreme point* or *vertex* of a convex set S if it *cannot* be expressed as $x = \alpha y + (1-\alpha)z$ with $y, z \in S, 0 < \alpha < 1$ and $y, z, \neq x$.

Major result:
Any linear program with finite optimal solution has an optimal extreme point (corner of the polyhedron) P.

Basic solutions

Consider a linear program in standard form, i.e.

$$z = \max\{cx \mid Ax = b,\ x \geq 0\}$$

where $A \in R^{m\times n}$, $b \in R^{m\times 1}$, $c \in R^{1\times n}$, $x \in R^{n\times 1}$, $m \leq n$ and A has full rank (rows are linearly independent).
A point x is a *basic solution* if (i) satisfies $Ax = b$ and (ii) the columns of the constraint matrix corresponding to the nonzero components of x are linearly independent. The components of x are separated into two subvectors : $n - m$ *nonbasic* variables x_N (all are zero) and m *basic* variables x_B, whose constraints coefficients correspond to an invertible $m \times m$ *basis matrix* B.
Then we have

$$Bx_B + Nx_N = b, \quad \text{and hence}$$

$$x_B + B^{-1}Nx_N = B^{-1}b = \overline{b}$$

A point x is a *basic feasible solution* if in addition $x \geq 0$.
The objective fonction can be written as

$$z = c_B x_B + c_N x_N = c_B\overline{b} + (c_N - c_B B^{-1}N)x_N$$

Extreme points of P

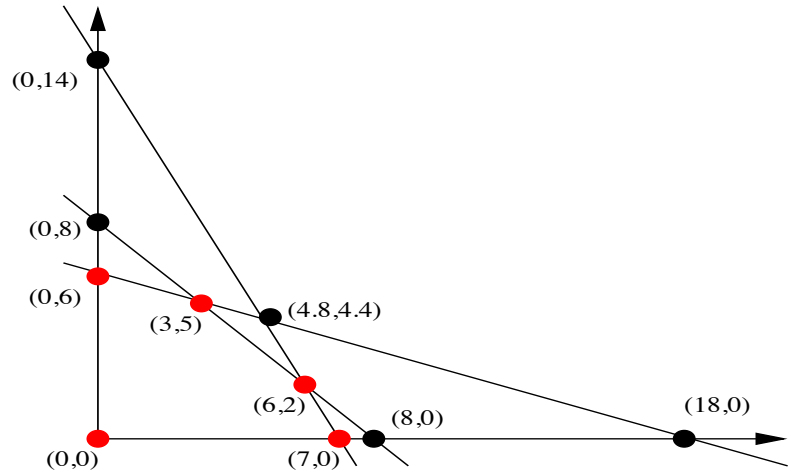
Theorem : *A point x is an extreme point of the set $\{x : Ax = b, x \geq 0\}$ iff it is a basic feasible solution.*

The number of basic feasible solutions is bounded by the number of ways that the m variables x_B can be selected among the n variables x.

$$\binom{n}{m} = \frac{n!}{m!(n-m)!}$$

The solution is amongst these extreme points of P.

$$\begin{array}{rcrcr} x_1 & + & 3x_2 & \leq & 18 \\ x_1 & + & x_2 & \leq & 8 \\ 2x_1 & + & x_2 & \leq & 14 \\ x_1 & & & \geq & 0 \\ & & x_2 & \geq & 0 \end{array}$$



The simplex method

Idea : perform an intelligent search in the set of extreme points.

Move from an extreme point to one of its adjacent if this move improves the objective.
Stop if no move improves the value of the objective.

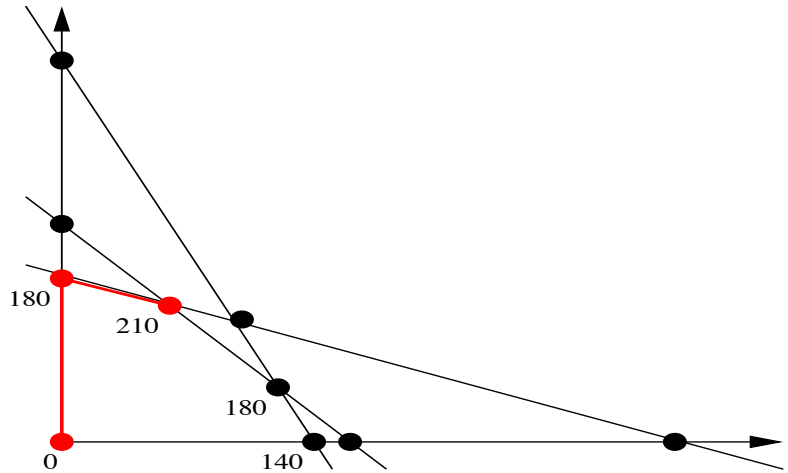


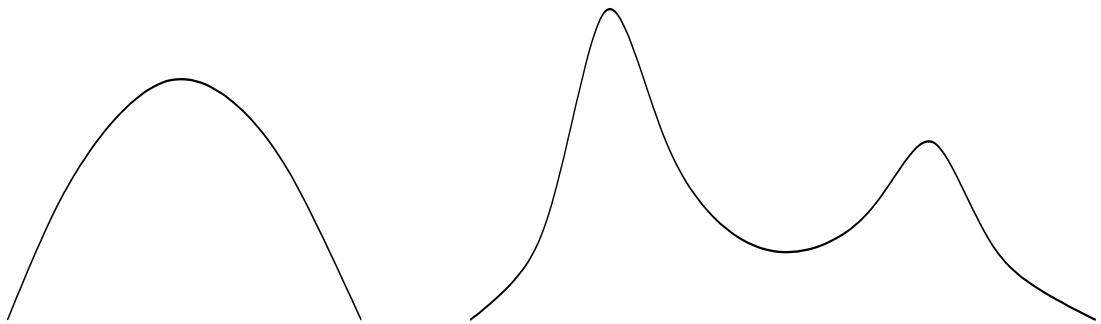
Figure 2: The value of the objective function is indicated on the vertices.

The simplex algorithm

- *Initialisation* : Chose an initial vertex $x^0 \in P$, $t = 1$.
- *Iteration t* : Let $y^1, \ldots, y^k$ be all the extreme points adjacent to x (two bases are adjacent if they have $m-1$ variables in common (connected with x^t by an edge)).
  - If $cx^t > cy^s$, $s = 1, \ldots, k$, stop. The optimal solution is x^t.
  - Otherwise, chose an adjacent vertex y^s, such that $cy^s \geq cx^t$. Set $x^{t+1} = y^s$ and move to itération $t+1$.

Simplex – why does it work ?

Since the polyhedron is convexe and the objective is linear, any local maximum is also global.



montagne convexe

montagne non–convexe

Exemple

	product1	product 2	product 3	product 4	stock
ressource A	2	4	5	7	42
ressource B	1	1	2	2	17
ressource C	1	2	3	3	24
benefice	7	9	18	17	

$$
\begin{array}{lrcrcrcrcr}
\text{Maximize } z = & 7x_1 & + & 9x_2 & + & 18x_3 & + & 17x_4 & & \\
\text{s. c.} & 2x_1 & + & 4x_2 & + & 5x_3 & + & 7x_4 & \leq & 42 \\
& x_1 & + & x_2 & + & 2x_3 & + & 2x_4 & \leq & 17 \\
& x_1 & + & 2x_2 & + & 3x_3 & + & 3x_4 & \leq & 24 \\
& x_1 & , & x_2 & , & x_3 & , & x_4 & \geq & 0
\end{array}
$$

The problem is written in its *canonical form*.

Introducing slack variables

$$
\begin{array}{rcrcrcrcrcrcrcr}
7x_1 & + & 9x_2 & + & 18x_3 & + & 17x_4 & \Longrightarrow & \text{Max} & & & & & & \\
2x_1 & + & 4x_2 & + & 5x_3 & + & 7x_4 & + & x_5 & & & & & = & 42 \\
x_1 & + & x_2 & + & 2x_3 & + & 2x_4 & & & + & x_6 & & & = & 17 \\
x_1 & + & 2x_2 & + & 3x_3 & + & 3x_4 & & & & & + & x_7 & = & 24 \\
x_1 & , & x_2 & , & x_3 & , & x_4 & , & x_5 & , & x_6 & , & x_7 & \geq & 0
\end{array}
$$

x_5	$=$	42	$-$	$2x_1$	$-$	$4x_2$	$-$	$5x_3$	$-$	$7x_4$
x_6	$=$	17	$-$	x_1	$-$	x_2	$-$	$2x_3$	$-$	$2x_4$
x_7	$=$	24	$-$	x_1	$-$	$2x_2$	$-$	$3x_3$	$-$	$3x_4$
z	$=$			$7x_1$	$+$	$9x_2$	$+$	$18x_3$	$+$	$17x_4$

Tr. 1

Basic solution and improvement

x_5, x_6, x_7 are the *basic variables*, while x_1, x_2, x_3, x_4 are the *nonbasic variables*. The *basic solution* is given by $x_1 = 0$, $x_2 = 0$, $x_3 = 0$, $x_4 = 0$ and $x_5 = 42$, $x_6 = 17$, $x_7 = 24$. The profit is $z = 0$.

Can we do better ?

Chose x_3 and increase its value by ensuring the solution stays feasible. Hence

$$\begin{array}{lcll} x_5 \geq 0 & \Rightarrow & 42 - 5x_3 \geq 0 & \Rightarrow \quad x_3 \leq 8.4 \\ x_6 \geq 0 & \Rightarrow & 17 - 2x_3 \geq 0 & \Rightarrow \quad x_3 \leq 8.5 \\ x_7 \geq 0 & \Rightarrow & 24 - 3x_3 \geq 0 & \Rightarrow \quad x_3 \leq 8 \end{array}$$

The most restrictive constraint is $x_3 \leq 8$.

Change of the basic solution

Solve in respect to x_3 and replace in the other constraints. This gives

$$
\begin{array}{rcrcrcrcrcr}
x_5 & = & 2 & - & \frac{1}{3}x_1 & - & \frac{2}{3}x_2 & + & \frac{5}{3}x_7 & - & 2x_4 \\
x_6 & = & 1 & - & \frac{1}{3}x_1 & + & \frac{1}{3}x_2 & + & \frac{2}{3}x_7 & & \\
x_3 & = & 8 & - & \frac{1}{3}x_1 & - & \frac{2}{3}x_2 & - & \frac{1}{3}x_7 & - & x_4 \\
\hline
z & = & 144 & + & x_1 & - & 3x_2 & - & 6x_7 & - & x_4
\end{array}
$$

Tr. 2

x_7 exits the basis and x_3 enters in the basis. The basic solution is $x_1 = x_2 = x_7 = x_4 = 0$, $x_5 = 2$, $x_6 = 1$, $x_3 = 8$. The profit is $z = 144$.

Can we do better ?

The only variable with non-negative coefficient is x_1. Chose it to enter the basis

$$\begin{aligned} x_5 \geq 0 &\Rightarrow x_1 \leq 6 \\ x_6 \geq 0 &\Rightarrow x_1 \leq 3 \\ x_3 \geq 0 &\Rightarrow x_1 \leq 24 \end{aligned}$$

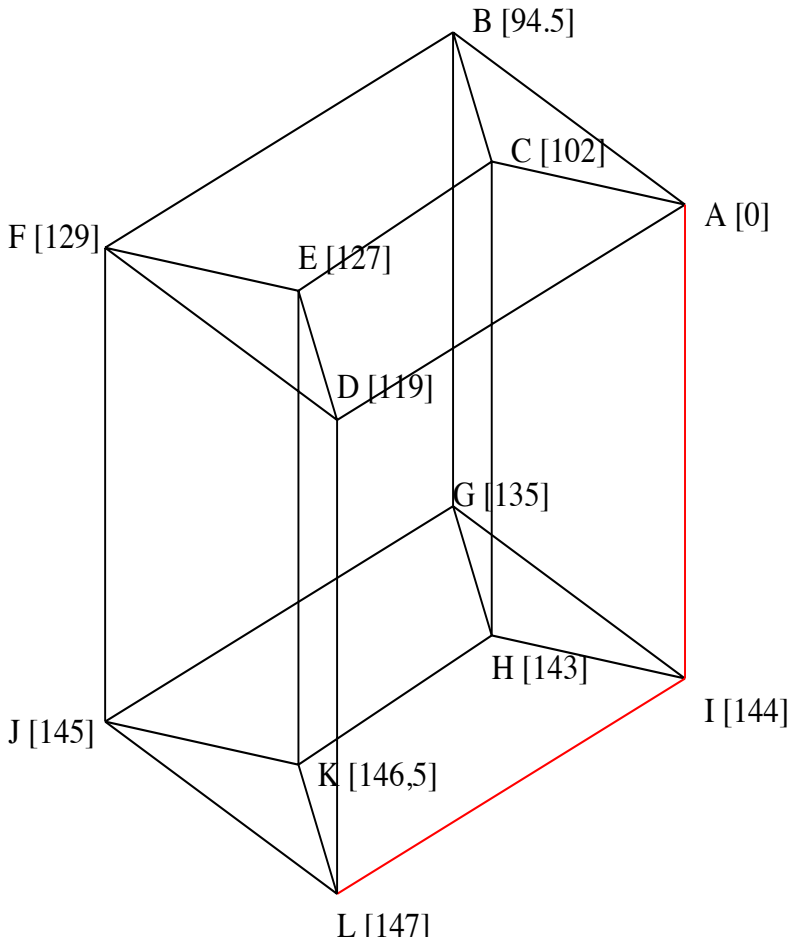
from where $x_1 \leq 3$. x_6 exits the basis and we obtain

x_5	=	1	+	x_6	−	x_2	+	x_7	−	$2x_4$
x_1	=	3	−	$3x_6$	+	x_2	+	$2x_7$		
x_3	=	7	+	x_6	−	x_2	−	x_7	−	x_4
z	=	147	−	$3x_6$	−	$2x_2$	−	$4x_7$	−	x_4

Tr. 3

The profit is $z = 147$. This is optimal since all coefficient on the last row are negative.

Geometrical interpretation



vertex	x_1	x_2	x_3	x_4	z
A	0	0	0	0	0
B	0	10.5	0	0	94.5
C	0	0	0	6	102
D	17	0	0	0	119
E	11.67	0	0	2.67	127
F	13	4	0	0	129
G	0	3	6	0	135
H	0	0	7	1	143
I	0	0	8	0	144
J	4	1	6	0	145
K	3	0	6.5	0.5	146.5
L	3	0	7	0	147

Convergence and complexity of simplex algorithm

One moves from an extreme point to another increasing the value of the objective function z. The number of vertices is bounded by $\binom{n}{m}$.

- If the objective function is bounded, the optimum will be found.
- Otherwise, the simplex detects that the objective function is unbounded (no candidat to enter in the basis).

Complexity issues:

- In 1972 Klee and Minty showed that the worst case complexity of the simplex is exponential.
- However, it is has been shown that on average the simplex behaves polynomially ($O(\min\{(m-n)^2, n^2\})$.
- In 1979 the Soviet mathematician Leonid Khachiyan described a polynomial-time algorithm for linear programming (the ellipsoid method). However, it is too slow to be of practical interest.
- In 1984 Narendra Karmarkar proposed another polynomial-time algorithm for linear programming (the interior-point method). It is considered competitive with the simplex method.

Solve graphically

$$
\begin{array}{lrcrcl}
\text{Maximize } z = & 2x_1 & + & x_2 & & \\
\text{s. c.} & 4x_1 & + & 2x_2 & \leq & 8 \\
 & & & x_2 & \leq & 2 \\
 & & & x_1, x_2 & \geq & 0
\end{array}
$$

$$
\begin{array}{lrcrcl}
\text{Maximize } z = & x_1 & + & 2x_2 & & \\
\text{s. c.} & -x_1 & + & x_2 & \leq & 2 \\
 & & & x_2 & \leq & 3 \\
 & & & x_1, x_2 & \geq & 0
\end{array}
$$

Max-Flow Min-Cut (MFMC)

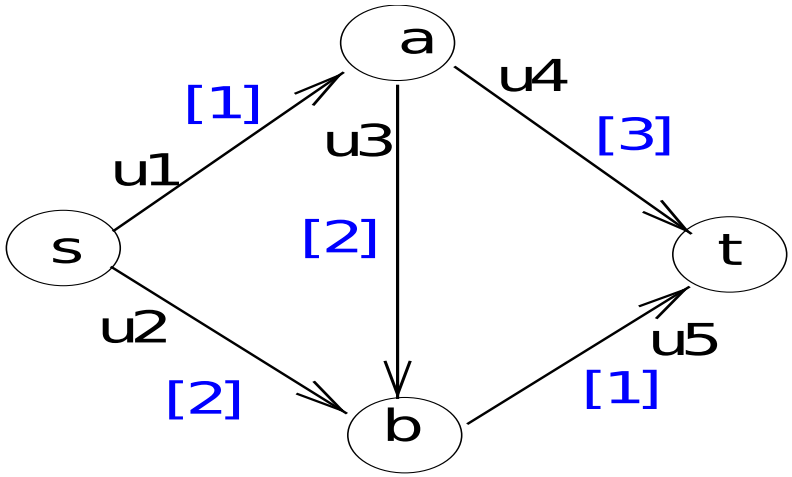
LP formulation for Shortest/Longest path problem (SPP/LPP)

- Given a directed acyclic graph (DAG) $G = (V, U)$ and weights w_j associated with each arc u_j, find a path from s to t with the minimum/maximum total weight.
- Definition :

  The node-arc incidence matrix
  $A = [a_{i,j}]$ $i = 1, 2, \ldots, |V|$
  and $j = 1, 2, \ldots, |E|$ is defined by

$$a_{i,j} = \begin{cases} +1 & \text{if arc } e_j \text{ leaves node } i \\ -1 & \text{if arc } e_j \text{ enters node } i \\ 0 & \text{otherwise} \end{cases}$$

- Example



$$A = \begin{array}{c|ccccc} & u1 & u2 & u3 & u4 & u5 \\ \hline s & +1 & +1 & 0 & 0 & 0 \\ t & 0 & 0 & 0 & -1 & -1 \\ a & -1 & 0 & +1 & +1 & 0 \\ b & 0 & -1 & -1 & 0 & +1 \end{array}$$

A weighted directed graph and the corresponding node-arc incidence matrix A. The weights are in squared brackets.

- Exercise : Use the node-arc incidence matrix to describe the LP formulation of the above SPP.

Max-Flow Min-Cut (MFMC) Problem

- Let $N = (s, t, V, E, C)$ be a directed graph G with $n = |V|$ nodes, $m = |E|$ edges with edge-capacity $c(i, j) \geq 0$ and two special notes s, t which are, respectively a source and sink of G. Such graphs are called *networks* and we will be searching for feasible flows in networks.
- Denote by $\delta^+(u)$ (resp. $\delta^-(u)$) the set of output/input edges for the vertex u. A *feasible flow* is defined by a vector $\phi = [\phi_1, \phi_2, \ldots \phi_m]$, s.t. $\forall e \in E$, $0 \leq \phi_e \leq c_e$, and such that the flow conservation holds $\forall u \in V \backslash \{s, t\}$, i.e.

$$\sum_{e \in \delta^+(u)} \phi_e = \sum_{e \in \delta^-(u)} \phi_e \tag{30}$$

- By ϕ_0 we denote the size of a flow and therefore

$$\sum_{e \in \delta^+(s)} \phi_e = \sum_{e \in \delta^-(t)} \phi_e = \phi_0 \text{ (flow value)} \tag{31}$$

- The *Max-Flow Problem* consists in finding the flow that maximizes ϕ_0.
- The Max-Flow Problem reduces to linear programming (maximizing a linear objective function under a set of linear constraints).

Modeling MFMC as a LP problem

- Let a_i denote the ith row of A. The flow conservation at a node i (except s, t) is expressed by :

$$a_i \phi = 0 \tag{32}$$

- The LP formulation of the MFMC is :

let $d \in R^n$ be defined by

$$d_i = \begin{cases} -1 & i = s \\ +1 & i = t \\ 0 & \text{otherwise} \end{cases}$$

$$\max \phi_0 \tag{33}$$

$$d\phi_0 + A\phi = 0 \tag{34}$$

$$\phi \leq c \tag{35}$$

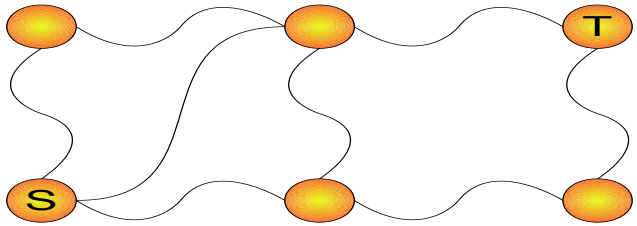
$$\phi \geq 0 \tag{36}$$

$$\phi_0 \geq 0 \tag{37}$$

The magic thing called duality [1]

[1] This section is largely influenced by the book Algorithms by S. Dasgupta, C.H. Papadimitriou, and U.V. Vazirani

Duality and Shortest paths



A *physical model* of a weighted undirected graph. Each edge is a string of length equal to the edge's weight, while each node is a knot at which the strings are tied.

How to find the shortest path from S to T?

Just *pull* S away from T until the gadget is taut.

Why does it work?

Because by pulling S away from T we solve the dual of the shortest-path problem!

Duality and Max Flow problem

Consider the below network with edge capacities

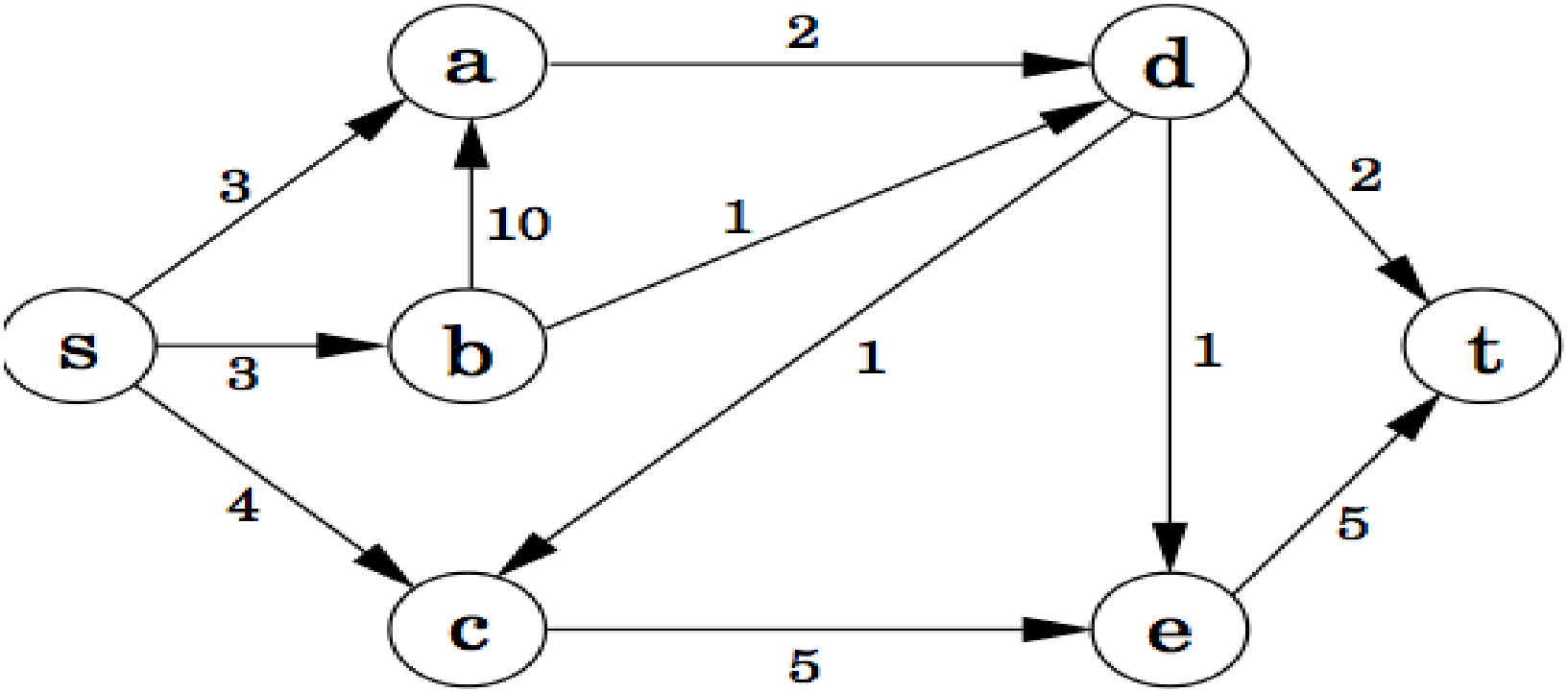


Figure 3: Find the maximum flow in the above network using the Ford-Fulkerson algorithm (1).

Ford-Fulkerson algorithm : basic idea

- Start with zero flow
- Repeat : choose an appropriate path from s to t, and increase flow along the edges of this path as much as possible.

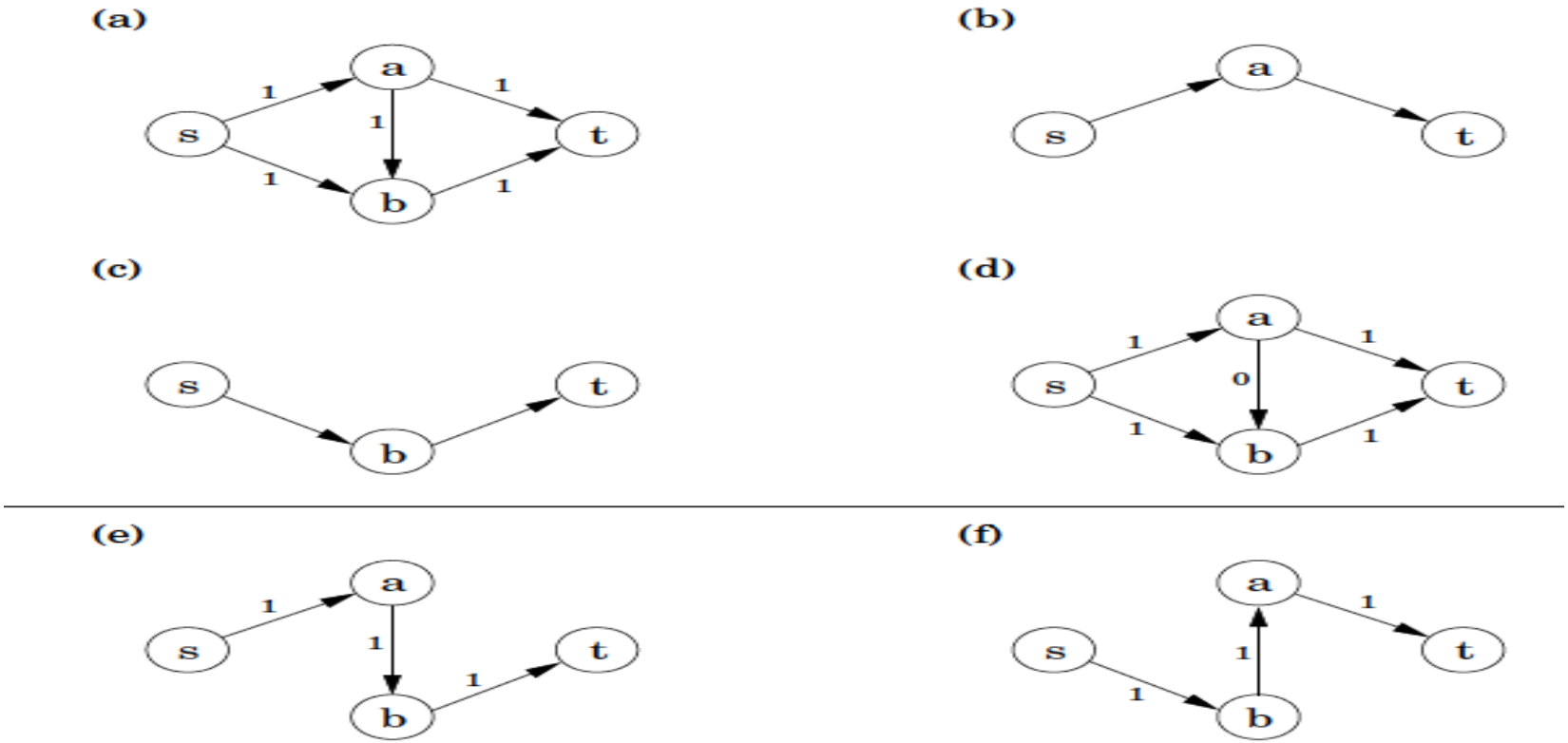


Figure 4: An illustration of the max-flow algorithm. (a) A toy network. (b) The first path chosen. (c) The second path chosen. (d) The final flow. (e) We could have chosen this path first. (f) In which case, we would have to allow this second path. Edge (b, a) of this path isn't in the original network and has the effect of canceling flow previously assigned to edge (a, b). Such a path is called *a*ugmenting path.

Ford-Fulkerson algorithm (FF): a closer look

FF algorithm searches for $s \rightsquigarrow t$ path in the *residual network* $\bar{G}(\phi) = [V, \bar{E}(\phi)]$, where for any feasible $s \rightsquigarrow t$ flow $\phi = [\phi_1, \phi_2, \ldots \phi_m]$, to any $u = (i, j) \in E$, we associate in $\bar{E}(\phi)$ two types of arcs, (u^+, u^-), with residual capacities as follows:

- *forward* edge : if $\phi_u < c_u \Rightarrow u^+ = (i, j)$ with residual capacities $c_u - \phi_u > 0$.
- *backward* edge : if $\phi_u > 0 \Rightarrow u^- = (j, i)$ with residual capacities $\phi_u > 0$.

Algorithm 1 Ford-Fulkerson algorithm

Require: A network $N = (s, t, V, E, C)$ and an initial flow ϕ
Ensure: maximal $s \rightsquigarrow t$ flow $\phi^* = [\phi_1^*, \phi_2^*, \ldots \phi_m^*]$

Initialisation : $k \leftarrow 0$, and set ϕ^k to be the current flow
while $(\exists s \rightsquigarrow t$ path in $\bar{G}(\phi^k))$ **do**
    Let π^k be the current $s \rightsquigarrow t$ path in $\bar{G}(\phi^k)$
    Let ε^k be the minimum of the residual capacities on π^k ($bottleneck(\pi^k, \phi^k)$)
    Define a new flow ϕ^{k+1} as follows
        $\phi_u^{k+1} \leftarrow \phi_u^k + \varepsilon^k$ if $u^+ \in \pi^k$
        $\phi_u^{k+1} \leftarrow \phi_u^k - \varepsilon^k$ if $u^- \in \pi^k$
        $\phi_0^{k+1} \leftarrow \phi_0^k + \varepsilon^k$
    set $k \leftarrow k + 1$
end while
$\phi^* \leftarrow \phi^k$

Min-Cut is a certificate of optimality

A truly remarkable fact: not only does Ford-Fulkerson (FF) algorithm correctly compute a maximum flow, but it also generates a short proof of the optimality of this flow!

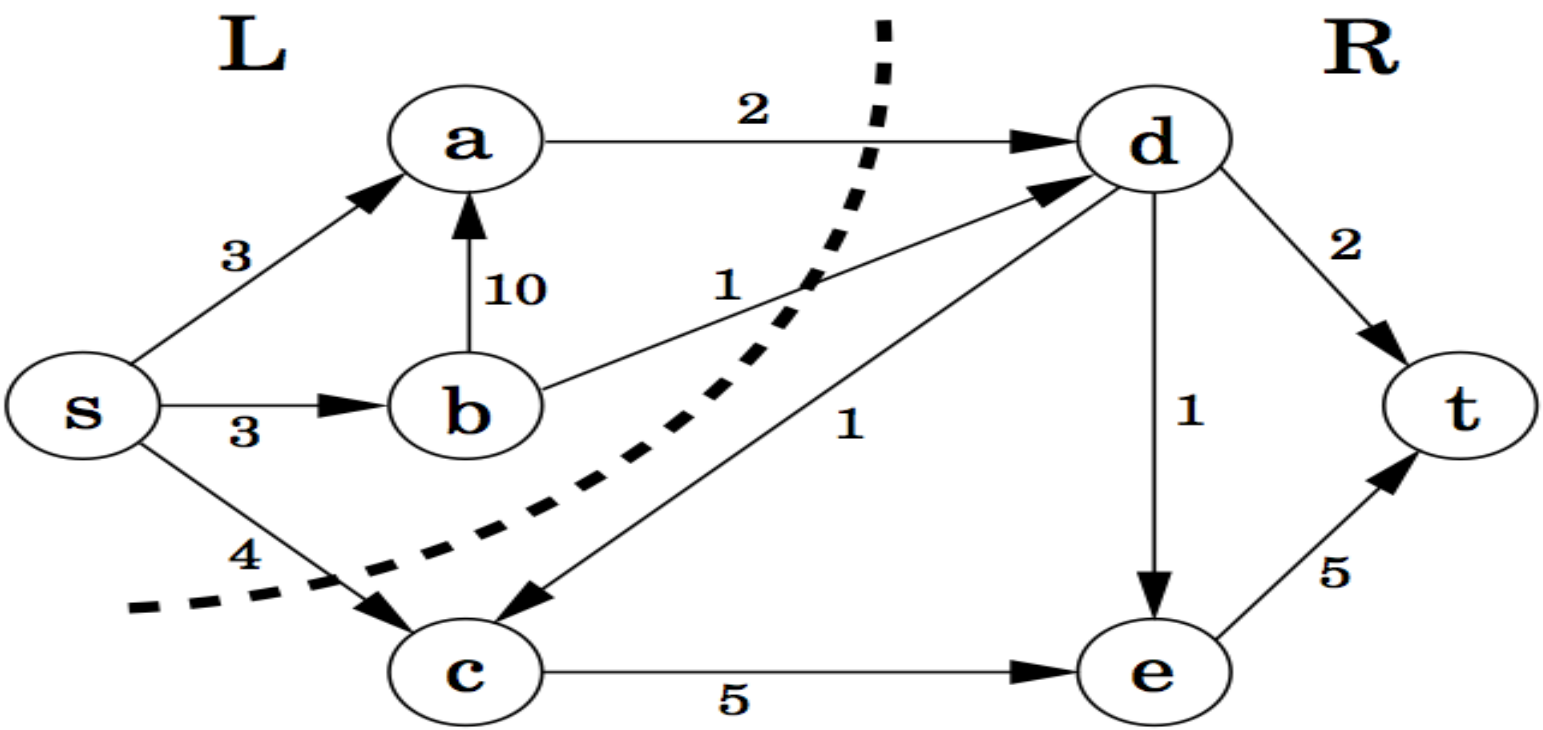


Figure 5: Partition the nodes of the above network into two groups, $L = \{s, a, b\}$ and $R = \{c, d, e, t\}$ Any $s \rightsquigarrow t$ flow must pass from L to R. Therefore, no flow can possibly exceed the total capacity of the edges from L to R, which is 7. This simply means that any $s \rightsquigarrow t$ flow of size 7, must be optimal!

MFMC : Max-Flow Min-Cut Relationships

- An $s-t$ *cut* is a partition $(A, \bar{A})$ of V, s.t. $s \in A$ and $t \in \bar{A}$.
- The *capacity* of an $s-t$ cut is $C(A, \bar{A}) = \sum_{e \in \delta^+(A)} c_e$.

Theorem

For any flow ϕ and any $s-t$ cut $(A, \bar{A})$, size(ϕ) $\leq C(A, \bar{A})$ holds.

Theorem

The size of the maximum flow in a network equals the capacity of the smallest $s-t$ cut.

Proof : Suppose ϕ is the final flow when FF algorithm terminates. We know that node t is no longer reachable from s in the residual network $\bar{G}(\phi)$. Let L be the nodes that are reachable from s in $\bar{G}(\phi)$, and let $R = V \setminus L$ be the rest of the nodes. Then (L, R) is a cut in the graph G. By the way L is defined, any edge going from L to R must be at full capacity (in the current flow ϕ), and any edge from R to L must have zero flow. Therefore the net flow across (L, R) is exactly the capacity of the cut and hence size(ϕ) = $C(L, R)$

What is duality about?

Consider the below LP

$$
\begin{array}{lrrrrrrr}
\text{Maximize} & z = & x_1 & + & 6x_2 & & \\
\text{c1} & & x_1 & & & \leq & 200 \\
\text{c2} & & & & x_2 & \leq & 300 \\
\text{c3} & & x_1 & + & x_2 & \leq & 400 \\
& & & & x_1,\ x_2 & \geq & 0
\end{array}
$$

A clever student declares the optimum solution to be $(x_1, x_2) = (100, 300)$, with objective value 1900. Can we check this answer? Suppose we take the inequality c_1 and add it to six times c_2. We get $x_1 + 6x_2 \leq 2000$. Interesting, this upper bound is not far from 1900. Can we improve it? After a little experimentation, we find that multiplying the three inequalities by 0, 5, and 1, respectively, and adding them up yields $x_1 + 6x_2 \leq 1900$. So 1900 must indeed be the best possible value! The multipliers $(0, 5, 1)$ magically constitute a *certificate of optimality*!

We'll show that such a certificate can be found systematically.

Deriving a systematic manner for computing a certificate for optimality

Let's see what we expect of these three multipliers, called y_1, y_2, y_3.

$$\begin{array}{lllll} y_1 & x_1 & & \leq & 200 \\ y_2 & & x_2 & \leq & 300 \\ y_3 & x_1 + & x_2 & \leq & 400 \end{array}$$

First, these y_i's must be nonnegative (otherwise they are unqualified to multiply inequalities). After the multiplication and addition steps, we get the bound:

$$(y_1 + y_3)x_1 + (y_2 + y_3)x_2 \leq 200y_1 + 300y_2 + 400y_3. \tag{38}$$

We want the left-hand side to look like our objective function $x_1 + 6x_2$ so that the right-hand side is an upper bound on the optimum solution. This could be obtained if $y_1 + y_3 \geq 1$ and $y_2 + y_3 \geq 6$. Thus, we get an upper bound

$$x_1 + 6x_2 \leq 200y_1 + 300y_2 + 400y_3 \qquad \text{if} \qquad \begin{array}{ll} y_1 + y_3 & \geq 1 \\ y_2 + y_3 & \geq 6 \\ y_1 \geq 0, y_2 \geq 0, y_3 \geq 0 & \end{array}$$

Since we want a bound that is as tight as possible, so we should minimize $200y_1 + 300y_2 + 400y_3$ subject to the preceding inequalities. *And this is a new linear program*!

Primal-Dual relationships and LP bounds generations

The best upper bound on the original LP is computed by solving a new LP problem

$$\begin{array}{rrrrrrrrr} \text{Minimize} & z = & 200y_1 & + & 300y_2 & + & 400y_3 & & \\ & & y_1 & + & y_2 & & & \geq & 1 \\ & & & & y_2 & + & y_3 & \geq & 6 \\ & & & & & & y_1,\ y_2,\ y_3 & \geq & 0 \end{array}$$

By design, any feasible value of this dual LP is an upper bound on the original primal LP. So if we somehow find a pair of primal and dual feasible values that are equal, then they must both be optimal. Here is just such a pair:

$$\text{Primal: } (x_1, x_2) = (100, 300); \quad \text{Dual:} (y_1, y_2, y_3) = (0, 5, 1)$$

They both have value 1900, and therefore they certify each other's optimality.

This is a main result in duality theory stating that
If a linear program has a bounded optimum, then so does its dual, and the two optimum values coincide.
We'll prove it formally.

Primal-Dual relationships in Linear Programming

Let $A \in R^{m \times n}, b \in R^{m \times 1}, c \in R^{1 \times n}$, a_i is the i-th row and A^j is the j-th col. of A.
To any primal (P) is associated its dual (D) by associating a dual variable to each primal constraint and following the rules:

Primal	Dual
max	min
variables x	constraints
constraints	variables y
objective coefficients c	constraint right hand sides c
constraint right hand sides b	objective coefficients b
$a_i x \leq b_i$	$y_i \geq 0$
$a_i x \geq b_i$	$y_i \leq 0$
$a_i x = b_i$	y_i unconstrained
$x_j \geq 0$	$yA^j \geq c_j$
$x_j \leq 0$	$yA^j \leq c_j$
x_j unconstrained	$yA^j = c_j$

Duality theory : Canonical Forms

Let be given a primal (P) in a canonical form (i.e.)

$$\text{(P)} \qquad z = \max\{cx \,|\, Ax \leq b, x \in R^n, x \geq 0\} \tag{39}$$

Its dual (D) is as follows.

$$\text{(D)} \qquad w = \min\{yb \,|\, yA \geq c, y \in R^m, y \geq 0\} \tag{40}$$

In general, the following relationships hold:

primal/dual constraint		**dual/primal constraint**	
consistent with canonical form	$\Longleftrightarrow$	variable ≥ 0	(42)
reversed from canonical form	$\Longleftrightarrow$	variable ≤ 0	(43)
eqaulity constraint	$\Longleftrightarrow$	variable unrestricted	(44)

Duality theory : Primal/Dual relationship (Illustration)

A primal problem and its dual

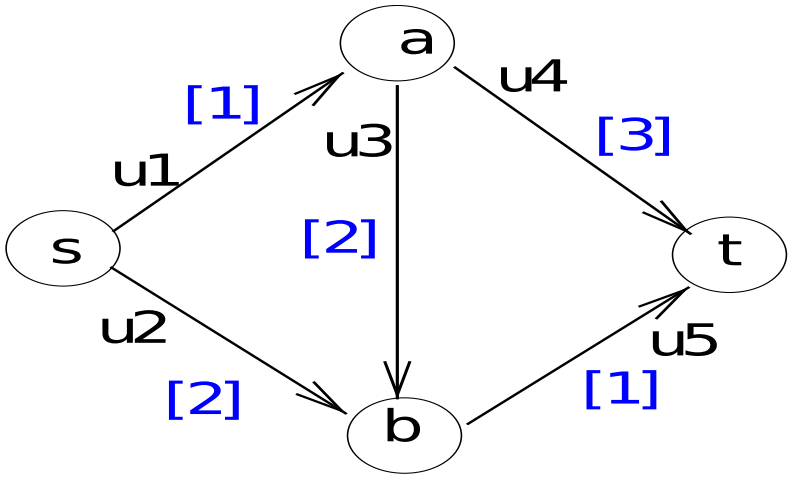
$$
\begin{array}{ll}
\max z = 6x_1 + x_2 + x_3 & \\
4x_1 + 3x_2 - 2x_3 & = 1 \\
6x_1 - 2x_2 + 9x_3 & \geq 9 \\
2x_1 + 3x_2 + 8x_3 & \leq 5 \\
x_1 \geq 0, x_2 \leq 0, x_3 \text{ unrestricted} &
\end{array}
\qquad
\begin{array}{ll}
\min w = y_1 + 9y_2 + 5y_3 & \\
4y_1 + 6y_2 + 2y_3 & \geq 6 \\
3y_1 - 2y_2 + 3y_3 & \leq 1 \\
-2y_1 + 9y_2 + 8y_3 & = 1 \\
y_1 \text{ unrestricted }, y_2 \leq 0, y_3 \geq 0 &
\end{array}
$$

Lemma :

It is easy to verify that the dual of the dual is the primal.

The shortest path problem (SPP)

Given a directed graph $G = (V, U)$ and weights $w_j \geq 0$ associated with each arc u_j, find a path from s to t with the minimum total weight.



$$A = \left|\begin{array}{c|ccccc} s & +1 & +1 & 0 & 0 & 0 \\ t & 0 & 0 & 0 & -1 & -1 \\ a & -1 & 0 & +1 & +1 & 0 \\ b & 0 & -1 & -1 & 0 & +1 \end{array}\right|$$

A weighted directed graph and the corresponding node-arc incidence matrix A. The weights are in squared brackets.

SPP : Primal-Dual LP formulation

A path from s to t can be thought of as a flow ϕ of unit 1 leaving s and entering t. The primal (P) of (SPP) and its dual (D) where a variable π_i is assigned to each node i, are given below.

(P)

$$\min \quad w\phi \tag{45}$$

$$A\phi = \left| \begin{array}{c|l} +1 & \text{row } s \\ -1 & \text{row } t \\ 0 & \text{otherwise} \end{array} \right.$$

$$\phi \geq 0 \tag{46}$$

(D)

$$\max \ \pi_s - \pi_t \tag{47}$$

$$\pi_i - \pi_j \quad \leq w_{ij} \quad \forall (ij) \in U \tag{48}$$

$$\pi_i \quad \text{unrestricted } \forall i \in V \tag{49}$$

König's Theorem

A graph $G = (U \cup V, E)$ is *bipartite* if the set of vertices $U \cup V$ is partitioned into two subsets U and V such that $U \cap V = \emptyset$ and $E(G[U]) = E(G[V]) = \emptyset$ so that all edges have one vertex in U and one in V. A *matching* of G is a set of edges meeting each vertex of G no more than once. Let $\delta(v)$ denote the set of edges having exactly one endpoint at v. If x_e is a binary variable associated with any $e \in E$, the above is equivalent to $\sum_{e \in \delta(v)} x_e \leq 1$, $\forall v \in V$ and $\forall v \in U$.
A *vertex cover* is a set of vertices $C \subseteq (U \cup V)$ such that any edge of the graph is adjacent with at least one vertex from C.

The below result is a famous characterization of maximum-cardinality matching in bipartite graphs.

Theorem

König's Theorem *: The number of edges in a maximum-cardinality matching in a bipartite graph G is equal to the minimum number of vertices needed to cover some endpoint of every edge of G.*

Proof : Write the Primal and the Dual of the above problem.

Weak Duality in Linear Programming (cont.)

Consider the following couple primal-dual (it is just a matter of convenience).

$$(P) \qquad z = \max\{cx | Ax \leq b, x \in R_+^n\} \tag{50}$$

$$(D) \qquad w = \min\{yb | yA \geq c, y \in R_+^m\} \tag{51}$$

Theorem

Weak Duality *: If x^* is feasible to P and y^* is feasible to D, then $cx^* \leq z \leq w \leq y^*b$.*

Proof : $cx^* \leq y^*Ax^* \leq y^*b$. Hence $w \geq cx^*$ for all feasible solution x^* to P , and $z \leq y^*b$ for all feasible solution y^* to D, so that $z \leq w$. Hence

1. if $cx^* = y^*b$, then x^* and y^* are optimal
2. if either is unbounded, then the other is infeasible.
   Proof : Suppose P is unbounded. By weak duality $w \geq \lambda$ for all $\lambda \in R^1$. Hence D has no feasible solution.

Weak Complementary-Slackness in Duality theory

Let us denote by a_i the i-the row of the matrix A and by A^j its j-the column.
Points $x \in R^n$ and $y \in R^m$ are *complementary*, with respect to P and D, if

$$y_i(b_i - a_i x) = 0, \text{ for } i = 1, \dots m \text{ and } x_j(c_j - yA^j) \text{ for } j = 1, \dots n \tag{52}$$

Theorem

Weak Complementary-Slackness *If feasible solutions x^* and y^* are complementary, then they are optimal solutions.*

Proof : Let $s^* = b - Ax^* \geq 0$ be the vector of slack variables of the primal, and let $t^* = y^*A - c \geq 0$ be the vector of surplus variables of the dual. Then we have

$$\begin{aligned} cx^* = & \quad (y^*A - t^*)x^* = y^*Ax^* - t^*x^* \\ = & \quad y^*(b - s^*) - t^*x^* = y^*b - y^*s^* - t^*x^*. \end{aligned} \tag{53}$$

Since x^* and y^* are complementary, we have $y^*s^* = t^*x^* = 0$. Hence $cx^* = y^*b$ and from the weak duality x^* and y^* are optimal for P and D respectively.

Duality theory in Linear Programming (cont.)

Consider the following couple primal-dual . (P) is given in the so called standard form to which any LP form can be transformed.

$$(P) \qquad z = \max\{cx | Ax = b, x \in R^n_+\} \tag{54}$$

$$(D) \qquad w = \min\{yb | yA \geq c, y \in R^m\} \tag{55}$$

We suppose $rank(A) = m \leq n$ (all redundant equations were removed). Since $rank(A) = m$, $\exists$ $m \times m$ nonsingular (invertible) submatrix B, such that $A = (B, N)$ (after permuting the columns). Then $Ax = Bx_B + Nx_N = b$ where $x = (x_B, x_N)$. Then $x_B = B^{-1}b - B^{-1}Nx_N$ and $z = c_B x_B + c_N x_N = c_B B^{-1} b + (c_N - c_B B^{-1} N)x_N$.

Definition

1. The $m \times m$ nonsingular submatrix B is called a *basis*
2. The solution $x_B = B^{-1}b, x_N = 0$ is called a *basic solution* of $Ax = b$.
3. x_b is the vector of *basic* variables and x_N is the vector of *nonbasic* variables.
4. If $B^{-1}b \geq 0$, then (x_B, x_N) is called a *basic primal feasible solution* and B is called a *primal feasible basis*.
5. Let $y = c_B B^{-1} \in R^m$. If $c_N - c_B B^{-1} N \leq 0$ then B is called a *dual feasible basis*.
6. $c_N - c_B B^{-1} N$ are called *reduced costs*.

Strong Duality in Linear Programming (cont.)

Consider the following couple primal-dual .

$$\text{(P)} \qquad z = \max\{cx | Ax = b, x \in R^n_+\} \tag{56}$$

$$\text{(D)} \qquad w = \min\{yb | yA \geq c, y \in R^m\} \tag{57}$$

Theorem

Strong Duality *: If one the above problems has an optimal solution then so does the other, and the optimal objective functions are equal. If either is infeasible, then the other is either infeasible or unbounded.*

Proof : Assume $x^* = (x_B, x_N)$ is an optimal solution for (P). Then $c_N - c_B B^{-1} N \leq 0$ (i.e. $c_N \leq c_B B^{-1} N$).
Let $y^* = c_B B^{-1}$. We have $y^* A = c_B B^{-1}(B, N) = (c_B, c_B B^{-1} N) \geq (c_B, c_N) = c$. Hence $y^* A \geq c$ (i.e. y^* is feasible for (D).
Furthermore, $z(x^*) = cx^* = c_B x_B = c_B B^{-1} b$ and $w(y^*) = y^* b = c_B B^{-1} b = z(x^*)$. Then according the Weak duality theorem the points x^* and y^* are optimal for P and D respectively.

Strong Complementary-Slackness in Duality theory

Let us denote by a_i the i-the row of the matrix A and by A^j its j-the column.
Points $x \in R^n$ and $y \in R^m$ are *complementary*, with respect to P and D, if

$$y_i(b_i - a_i x) = 0, \text{ for } i = 1, \dots m \text{ and } x_j(c_j - yA^j) \text{ for } j = 1, \dots n \tag{58}$$

Theorem

Strong Complementary-Slackness *If feasible solutions x^* and y^* are optimal solutions to P and D, respectively, then they are complementary.*

Proof : Let $s^* = b - Ax^* \geq 0$ be the vector of slack variables of the primal, and let $t^* = y^*A - c \geq 0$ be the vector of surplus variables of the dual. Then we have

$$\begin{aligned} cx^* = & \quad (y^*A - t^*)x^* = y^*Ax^* - t^*x^* \\ = & \quad y^*(b - s^*) - t^*x^* = y^*b - y^*s^* - t^*x^*. \end{aligned} \tag{59}$$

From the strong duality $cx^* = y^*b$. Hence $y^*s^* + t^*x^* = 0$ with $y^*, s^*, x^*, t^* \geq 0$ so that the result follows.

Theorem

Variant: *A pair (x^*, y^*) feasible in P and D respectively, is optimal iff.*

$$y_i^*(b_i - a_i x^*) = 0, \text{ for } i = 1, \dots m \quad x_j^*(c_j - y^*A^j) = 0, \text{ for } j = 1, \dots n \tag{60}$$

MFMC : LP formulation

Let (as usual) a_i denote the ith row of A.

- The flow conservation at a node i (except s, t) is expressed by :

$$a_i \phi = 0 \tag{61}$$

- The LP formulation of the MFMC is :

let $d \in R^n$ be defined by

$$d_i = \begin{cases} -1 & i = s \\ +1 & i = t \\ 0 & \text{otherwise} \end{cases}$$

$$\max \phi_0 \tag{62}$$

$$d\phi_0 + A\phi = 0 \tag{63}$$

$$\phi \leq c \tag{64}$$

$$\phi \geq 0 \tag{65}$$

$$\phi_0 \geq 0 \tag{66}$$

MFMC : Primal-Dual LP formulation

Assign variables $\lambda_i, i \in V$ to (68) and variables $\gamma_{i,j}, (i,j) \in E$ to (69). Then we obtain :

(P)

$$\max \phi_0 \tag{67}$$

$$d\phi_0 + A\phi = 0 \tag{68}$$

$$\phi \leq c \tag{69}$$

$$\phi \geq 0 \tag{70}$$

$$\phi_0 \geq 0 \tag{71}$$

(D)

$$\min \sum_{(i,j)\in E} \gamma_{i,j} c_{i,j} \tag{72}$$

$$\lambda_i - \lambda_j + \gamma_{i,j} \geq 0 \ \forall (i,j) \in E \tag{73}$$

$$-\lambda_s + \lambda_t \geq 1 \tag{74}$$

$$\gamma_{i,j} \geq 0 \ \forall (i,j) \in E \tag{75}$$

MFMC : Max-Flow Min-Cut Relationships

Definition

An $s-t$ *cut* is a partition $(A, \bar{A})$ of V, s.t. $s \in A$ and $t \in \bar{A}$.
The *capacity* of an $s-t$ cut is $C(A, \bar{A}) = \sum_{e \in \delta^+(A)} c_e$.

Theorem

Every $s-t$ cut determines a feasible solution with cost $C(A, \bar{A})$ to (D) as follows:

$$\gamma_{i,j} = \begin{cases} 1 & (i,j) \ \ s.t. \ \ i \in A, j \in \bar{A} \\ 0 & otherwise \end{cases} \qquad \lambda_i = \begin{cases} 0 & i \in A, \\ 1 & i \in \bar{A} \end{cases}$$

Proof: Just by checking. Note that (73) is strict inequality iff $i \in \bar{A}$ and $j \in A$. Note also that $-\lambda_s + \lambda_t = 1$.

Max-Flow Min-Cut Theorem

Theorem

The value ϕ_0 is no greater than the capacity $C(A, \bar{A})$ of any $s-t$ cut. Furthermore, the value of the maximum flow equals the capacity of the minimum cut, and a flow ϕ and cut $C(A, \bar{A})$ are jointly optimal iff

$$\phi_{i,j} = 0 \text{ for } (i,j) \in E \text{ such that } i \in \bar{A} \text{ and } j \in A \tag{76}$$

$$\phi_{i,j} = c_{i,j} \text{ for } (i,j) \in E \text{ such that } i \in A \text{ and } j \in \bar{A} \tag{77}$$

Proof. Trivial, use the complementary slackness.

Well-Solved Problems

The assignment problem (AP) : a mathematical model

- Suppose we have m individuals and m jobs. If individual i is assigned to job j, the cost will be c_{ij}.
- Find the min. cost assignment or a one-to-one *matching* of individuals to jobs.

$$\min z = \sum_{i=1}^{m}\sum_{j=1}^{m} c_{ij}x_{ij} \tag{78}$$

$$\sum_{j=1}^{m} x_{ij} = 1, i = \overline{1,m} \tag{79}$$

$$\sum_{i=1}^{m} x_{ij} = 1, j = \overline{1,m}; \tag{80}$$

$$x_{ij} \in \{0,1\} \tag{81}$$

In matrix form :

$$\min cx \quad \{Ax = b, \quad x_{ij} = 0 \text{ or } 1, \quad i,j = \overline{1,m}\} \tag{82}$$

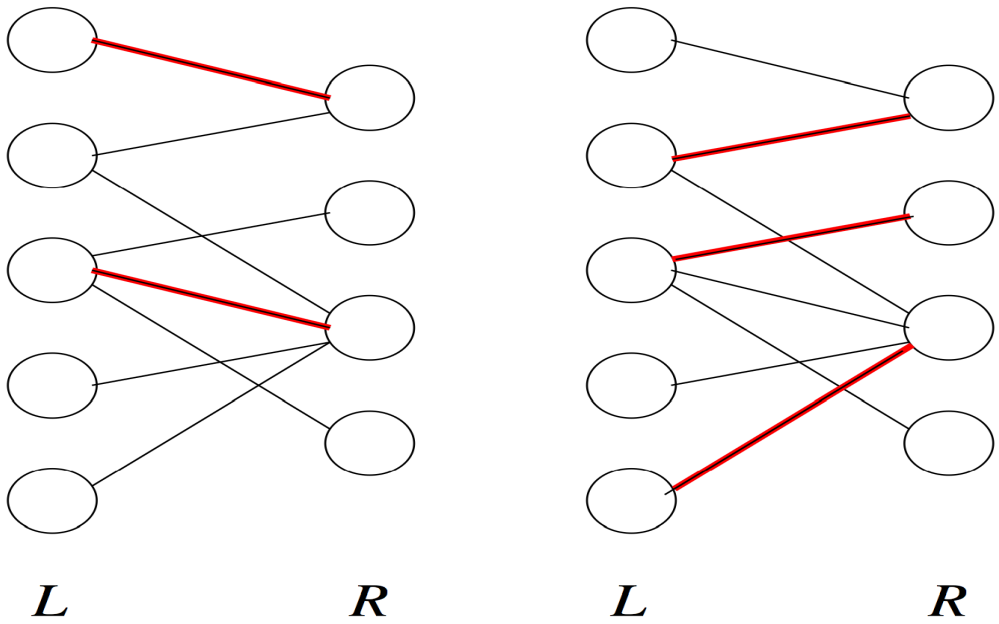
where $x = (x_{11} \dots x_{1m} \dots x_{m1} \dots x_{mm})^t$, $A^{2m \times mm}$ is the node-edge incidence matrix whose (i,j) column is $a_{ij} = e_i + e_{m+j}$, $i = \overline{1,m}, j = \overline{1,m}$; and $b = (1)^t$ (vector of $2m$ ones)

Max-Bipartite Matching

Definition

Given a graph $G = (V, E)$, a *matching* $M \subseteq E$ is a set of disjoint edges. A *a covering by nodes* is a set $R \subseteq V$ of nodes such that every edge has at least one endpoint in R.

Problem: Given bipartite graph G, find a maximum matching.



Suggestion: Use max-flow algorithm.

Question: Give the complexity of the proposed algorithm.

Matching and Covering

Definition

Given a graph $G = (V, E)$, a *matching* $M \subseteq E$ is a set of disjoint edges. A *a covering by nodes* is a set $R \subseteq V$ of nodes such that every edge has at least one endpoint in R.

Let A be the node-edge incidence matrix of a graph $G = (V, E)$ where $n = \mid V \mid$ and $m = \mid E \mid$. The maximum cardinality matching problem can be formulated as the integer program:

$$z = \max\{1x \mid Ax \leq 1, x \in Z_+^m\} \tag{83}$$

and the minimum cardinality covering problem as:

$$w = \min\{1y \mid yA \geq 1, y \in Z_+^n\} \tag{84}$$

Let z^{LP} and w^{LP} be the values of their corresponding LP relaxation. Then $z \leq z^{LP} = w^{LP} \leq w$ and hence pbs. (83) and (84) are weak-dual pair. When $z = w$ holds, the corresponding couple of problems form a strong-dual pair.

Note: Any dual feasible solution provides an upper bound on z!

However, pbs. (83) and (84) are not strong-dual pair (i.e. $z = w$ doesn't hold always) (take for example a triangle). We will see below that strong duality holds for this pair of problems when the graph G is bipartite.

Total Unimodularity

When solving an LP problem always gives integer solution?

Consider the problem (IP) $\max\{cx|Ax \leq b, x \in Z_+^n,\}$ with *integral* data (A, b).

When its linear programming relaxation (LP) $\max\{cx|Ax \leq b, x \in R_+^n,\}$ will have integral optimal solution?

Sufficient condition: if for the optimal basis B, $det(B) = \pm 1$, then the linear programming relaxation solves IP.

Proof: From Cramer's rule $B^{-1} = \frac{adj(B)}{det(B)}$ and since the adjoint matrix $adj(B)$ is an integral matrix, then B^{-1} is integral for all integral b.

Definition

An integer matrix A is called *totally unimodular* (TUM) if the determinant of each square submatrix of A equals 0,+1, or -1.

$$P(A) = \{x|Ax \leq b, x \geq 0,\} \tag{85}$$

Theorem

If A is TUM, then all the vertices of $P(A)$ are integer for any integer vector b.

Proof. First prove the case $\{x|Ax = b, x \geq 0,\}$. Any basic solution $x = B^{-1}b = \frac{adj(B)b}{det(B)}$. Then show that $(A|\mathbf{I})$ is TUM.

Total Unimodularity (cont I)

Theorem

An integer matrix A with $a_{ij} = 0, \pm 1$ is TUM if no more than two nonzero entries appear in any column, and if the rows of A can be partitioned into two sets I_1 and I_2 s.t.:

1. *If a column has two entries of the same sign, their rows are in different sets;*
2. *If a column has two entries of the different signs, their rows are in the same sets;*

Proof. By induction of the size of submatrices. Let C be a submatrix of size k. For $k = 1$ it is true. Let $k > 1$. It is obvious when C has a column of all zeros or with one nonzero. Let C has two nonzero entries in every column. Then the conditions of the theorem imply:

$$\sum_{i \in I_1} a_{ij} = \sum_{i \in I_2} a_{ij} \text{ for every column } j$$

. i.e. a linear combination of rows is zero, hence $det(C) = 0$.

Total Unimodularity (cont II)

Corollary

Any LP in standard or canonical form whose constraint matrix A is either

1. *The node-arc incidence matrix of a directed graph, or*
2. *The node-edge incidence matrix of an undirected bipartite graph,*

has only integer optimal vertices. This includes the LP formulation of shortest path, max-flow, the transport problem, the assignment problem and weighted bipartite matching.

Proof. The matrices in Case 1 satisfy the condition of the theorem with $I_2 = \emptyset$; those in Case 2 with $I_1 = V$ and $I_2 = U$, where the bipartite graph is $G = (V, U, E)$.

The assignment problem (AP)

The assignment problem (AP)

- Suppose we have m individuals and m jobs. If individual i is assigned to job j, the cost will be c_{ij}.
- Find the maximum cost assignment of individuals to jobs (the maximum weight perfect matching of a complete bipartite graph).
- We shall exploit the special structure to get a competitive *primal-dual* type of algorithm (the so called Kuhn's Algorithm (also called Hungarian Algorithm)).

The assignment problem (AP) : a mathematical model

Primal P:

$$\max \quad z = \sum_{i=1}^{m}\sum_{j=1}^{m} c_{ij}x_{ij} \tag{86}$$

$$\sum_{j=1}^{m} x_{ij} = 1, i = \overline{1,m} \tag{87}$$

$$\sum_{i=1}^{m} x_{ij} = 1, j = \overline{1,m}; \tag{88}$$

$$0 \leq x_{ij} \leq 1 \tag{89}$$

In matrix form :

$$\max \quad cx \quad \{Ax = b, \quad 0 \leq x_{ij} \leq 1, \quad i,j = \overline{1,m}\} \tag{90}$$

where $x = (x_{11} \ldots x_{1m} \ldots x_{m1} \ldots x_{mm})^t$, $A^{2m \times mm}$ is the node-edge incidence matrix whose (i,j) column is $a_{ij} = e_i + e_{m+j}$, $i = \overline{1,m}, j = \overline{1,m}$; and $b = (1)^t$ (vector of $2m$ ones)

Remark: the variables should be required to be binary, but we get that for free because the matrix A is totally unimodular and the right-hand side is integer.

The Dual Problem

u_i: dual var. related to the first m constraints (individuals).
v_j: dual var. related to the second m constraints (jobs).

Dual D:

$$\min w = \sum_{i=1}^{m} u_i + \sum_{j=1}^{m} v_j \tag{91}$$

$$u_i + v_j \geq c_{ij}, i = \overline{1,m}, j = \overline{1,m}; \tag{92}$$

$$u_i, v_j \text{ unrestricted }, i, j = \overline{1,m}, \tag{93}$$

Complementarity

- The complementarity slackness conditions are given by

$$(c_{ij} - u_i - v_j)x_{ij} = 0 \quad i, j = \overline{1, m} \tag{94}$$

- Thus if we can find a set of feasible u's, v's and x's that satisfy (94), those u's, v's and x's will be optimal.
- A feasible dual solution is given by

$$\hat{u}_i = \max_{1 \leq j \leq m} \{c_{ij}\} \quad i = \overline{1, m} \tag{95}$$

$$\hat{v}_j = \max_{1 \leq i \leq m} \{c_{ij} - \hat{u}_i\} \quad j = \overline{1, m} \tag{96}$$

(i.e. $\hat{u}_i$ is the max of c_{ij} in row i and $\hat{v}_j$ is the max of $c_{ij} - \hat{u}_i$ in column j.

Towards the solution

Consider a reduced cost coefficient matrix $\hat{c}_{ij}$ where c_{ij} is replaced by $\hat{c}_{ij} = c_{ij} - \hat{u}_i - \hat{v}_j$. This matrix will have a zero in every row and column and all its entries will be ≤ 0.
Let us associate $x_{ij} = 1$ to any zero cell (i, j). Then by complementarity condition we have the optimal solution if these x_{ij}'s are feasible for the primal. Note that in a feasible primal solution there will be exactly one $x_{ij} = 1$ in each row and exactly one $x_{ij} = 1$ in each column (i.e. there will be exactly m of these x_{ij}'s s.t. $x_{ij} = 1$, the rest being zero).

Definition

Cells(matrix coefficients) set to zero, such that no two cells occupy the same row or column are called *independent*.

Theorem

The maximum number of independent zero cells in a reduced assignment matrix is equal to the minimum number of lines to cover all zeros in the matrix.

Modifying the Reduced Matrix (RM)

Suppose that we cannot find a feasible set of positive x_{ij}'s from among the zero cells of the RM. Consider the covered matrix, the RM with zeros covered by the fewest number of lines (say k).
Let $S_r = \{i_1, i_2, \dots\}$ be the set of uncovered rows and $p = |S_r|$
Let $S_c = \{j_1, j_2, \dots\}$ be the set of uncovered columns and $q = |S_c|$.
Set $\bar{S}_r = M \setminus S_r$ and $\bar{S}_c = M \setminus S_c$, where $M = \{1, 2, \dots, m\}$.

We define

$$c_0 = \max_{i \in S_r, j \in S_c} \{\hat{c}_{ij}\} < 0 \quad (\text{ i.e. max uncovered element}) \tag{97}$$

A new dual feasible solution is given by

$$\bar{u}_i = \hat{u}_i + c_0, i \in S_r;\ \bar{u}_i = \hat{u}_i, i \in \bar{S}_r;\ \bar{v}_j = \hat{v}_j, j \in S_c;\ \bar{v}_j = \hat{v}_j - c_0, j \in \bar{S}_c$$

(In the new RM c_0 is subtracted from each uncovered element and added to each twice-covered element).

Example 1

Solve the assignment problem with the following cost matrix

	a	b	c	d
1	6	15	12	13
2	18	8	14	15
3	13	12	17	11
4	18	16	14	10

Example 2

Let A,B,C,D,E,F be young programmers we want to entrust with the development of the tasks a,b,c,d,e,f. Their skills being diverse we have marked their ability to program such or such task out of 100, and we have obtained the data below.

	a	b	c	d	e	f
A	86	94	82	84	37	85
B	59	22	56	27	30	75
C	56	19	64	20	20	22
D	54	26	95	75	17	97
E	28	68	45	49	97	19
F	31	24	88	1	17	70

1. Apply the Hungarian algorithm in order to to find the best assignment.
2. At any iteration this algorithm requires finding the maximum number of independent zero cells in a reduced assignment matrix. To prove that you have found this number, you'll apply the Ford-Fulkerson algorithm (at least in the first iteration). Indicate clearly how you use the max flow model to find the maximum number of independent zero cells and give the correspondence between the solutions of both models.

The assignment problem in terms of bipartite graphs : Max-Weight Bipartite Matching

We have a complete bipartite graph $G = (X \cup Y, E)$ with $m = \mid X \mid$ vertices X and $m = \mid Y \mid$ vertices Y and each edge has a weight $w(i, j)$. We want to find a perfect matching with maximum weight.

Some notations and definitions :

- A graph $G = (V, E)$ is bipartite if there exists partition $V = X \cup Y$ with $X \cup Y = \emptyset$ and $E \subseteq X \times Y$.
- A *matching* M in G is a set of pairwise non-adjacent edges; that is, no two edges share a common vertex.
- A *Perfect Matching* is a matching M in which every vertex is adjacent to some edge in M.
- The weight of matching M is sum of the weights of edges in M, $w(M) = \sum_{e \in M} w(e)$.

Max-Weight Bipartite Matching (cont)

Some notations and definitions (cont):

- Vertex v is *matched* if it is an endpoint of edge in M; otherwise v is *free*.
- A path is *alternating* if its edges alternate between M and $E \setminus M$.
- An alternating path is *augmenting* if both end-points are free.
- Augmenting path has one less edge in M than in $E \setminus M$; replacing the M edges by the $E \setminus M$ ones *increments size of the matching*.

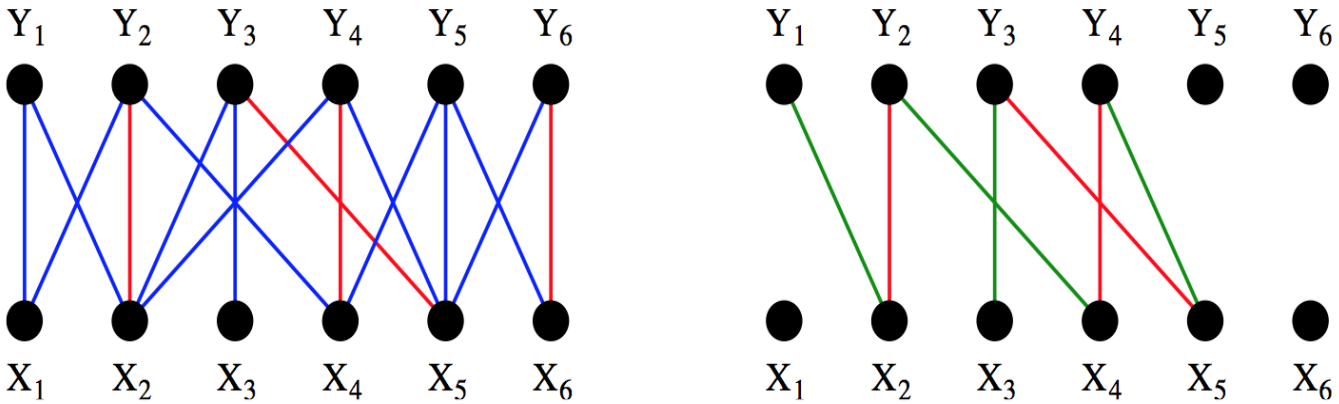


Figure 6: Left: Y2, Y3, Y4, Y6, X2, X4, X5, X6 are matched, other vertices are free. ; Right: Y1, X2, Y2, X4, Y4, X5, Y3, X3 is alternating.

Max-Weight Bipartite Matching (cont)

Some notations and definitions (cont):

- In the primal problem (P) the objective now is to maximize $w(M)$.
- A vertex *labeling* is a function $l : X \cup Y \to R$. This function is also called a *potential*. It sets the values of the dual problem variables.
- The dual (D) is a minimisation problem and the constraint (92) becomes

$$l(x) + l(y) \geq w(x, y). \tag{98}$$

- A *feasible labeling* is one such that (98) is satisfied.
- the *Equality Graph* is $G_l = (X \cup Y, E_l)$ where: $E_l = \{(x, y) : l(x) + l(y) = w(x, y)\}$.

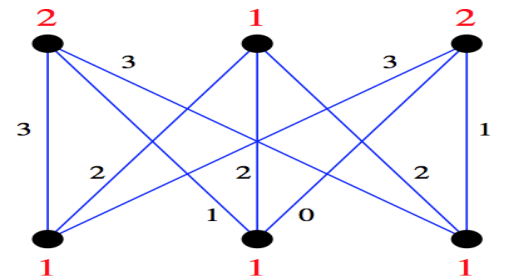
Theorem

[Kuhn-Munkres theorem] *: If the labeling l is feasible and M is a perfect matching in E_l, then M is a max-weight matching.*

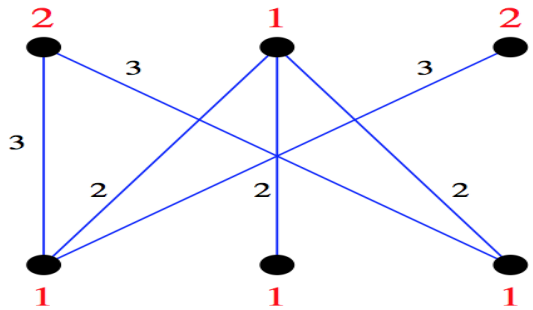
Proof: Follows from the complementary-slackness theorem.

Max-Weight Bipartite Matching

The KM theorem transforms the problem from an optimization problem of finding a max-weight matching into a combinatorial one of finding a perfect matching.



A feasible labeling ℓ



Equality Graph G_ℓ

Sketch of the Hungarian algorithm :

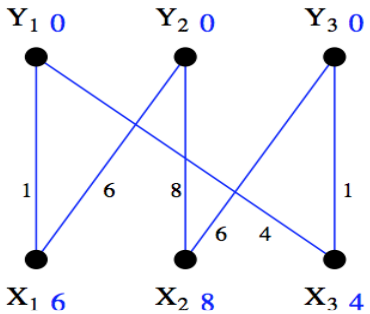
Start with any feasible labeling l and some matching M in E_l.
While M is not perfect repeat the following:

1. Find an augmenting path for M in E_l; this increases size of M.
2. If no augmenting path exists, improve l to l' such that $E_l \subset E_{l'}$. Go to 1.

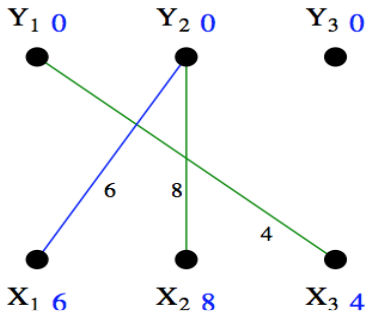
In each step of the loop we will either be increasing the size of M or E_l so this process must terminate.

Hungarian algorithm in terms of bipartite graphs

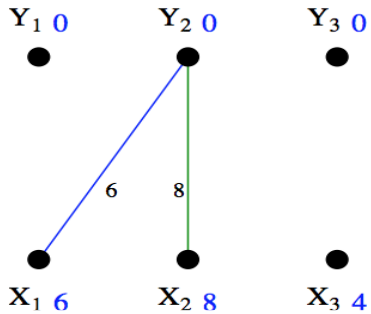
Finding an initial feasible labeling is simple. Just use: $\forall y \in Y, l(y) = 0,\ \forall x \in X, l(x) = \max_{y \in Y} w(x, y)$



Original Graph

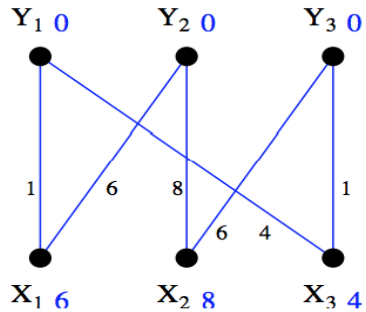


Eq Graph+Matching

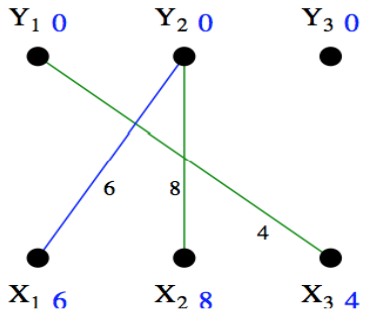


Alternating Tree

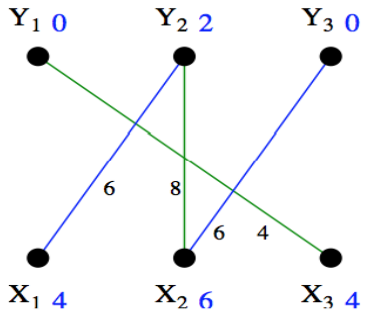
Max-Weight Bipartite Matching



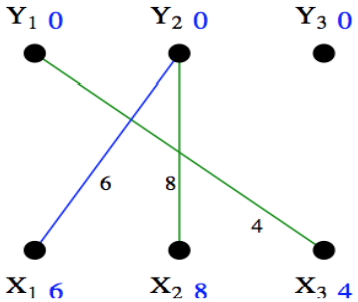
Original Graph



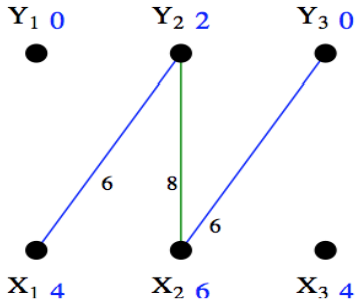
Old E_ℓ and $|M|$



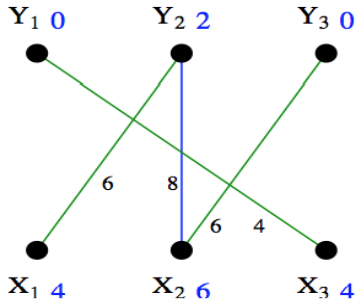
new Eq Graph



Orig E_ℓ and M



New Alternating Tree



New M