Discrete Optimization and Integer Programming.

Linear program modeling with MiniZinc

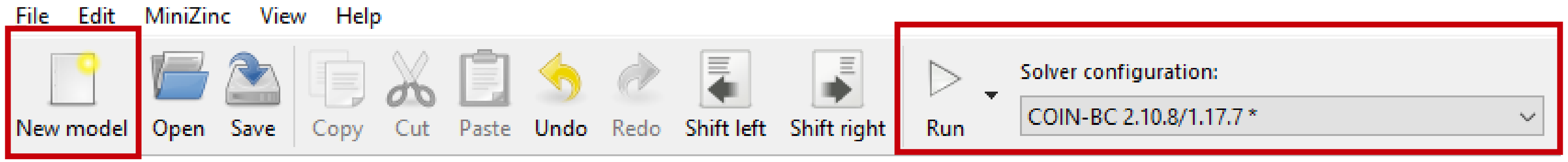
Security Level:

MiniZinc fast start

Setup

Download and set up MiniZinc compiler and IDE https://www.minizinc.org/software.html
Use MiniZinc Handbook if you need help https://www.minizinc.org/doc-2.6.4/en/index.html

First run



1. Create New model
2. Write the following test model

```
var float: x;
var float: y;
solve minimize x + 2*y;
constraint x + y >= 3;
constraint y >= 0;
```

3. Select COIN-BC solver configuration and click “Run”
4. In case of success you will get the output looks like:

```
Running setup_test.mzn
  No match for -verbose - ? for list of commands
  No match for 1 - ? for list of commands
  x = 3.0;
  y = -0.0;
  % time elapsed: 378msec
  ----------
  ==========
  Finished in 415msec.
```

LP model structure

```
% VARIABLES
var float: x1;
var float: x2;
var float: x3;

% OBJECTIVE
solve minimize x1 + 3*x2 - x3;

%CONSTRAINTS
constraint x1 + x2 + x3 >= 3;
constraint -x1 + 2*x2 >= 2;
constraint -x1 + 5*x2 + x3 <= 7;
constraint x1 >= 0;
constraint x2 >= 0;
constraint x3 >= 0;
```

Define **variables**. If you want to solve Linear programming model, then variable domains should be defined by **float numbers**. Otherwise problem will be interpreted as MILP.

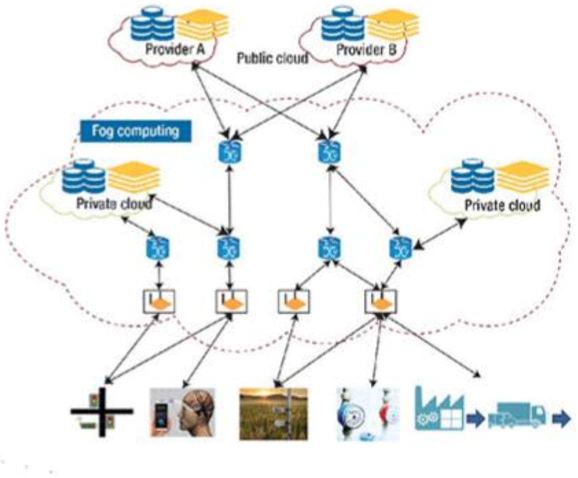
Set satisfiability solving mode or choose optimization direction (**min**/max). In case you solving optimization problem, objective function formula should be given.

Set up **constraints** to make problem statement correct.

Assignment problems

Applications

- Manufacturing systems – assign workers to tasks
- Distributed computing – assign subtasks to servers
- Sudoku – assign number to cells
- Agriculture – assign plants to fields
- ….



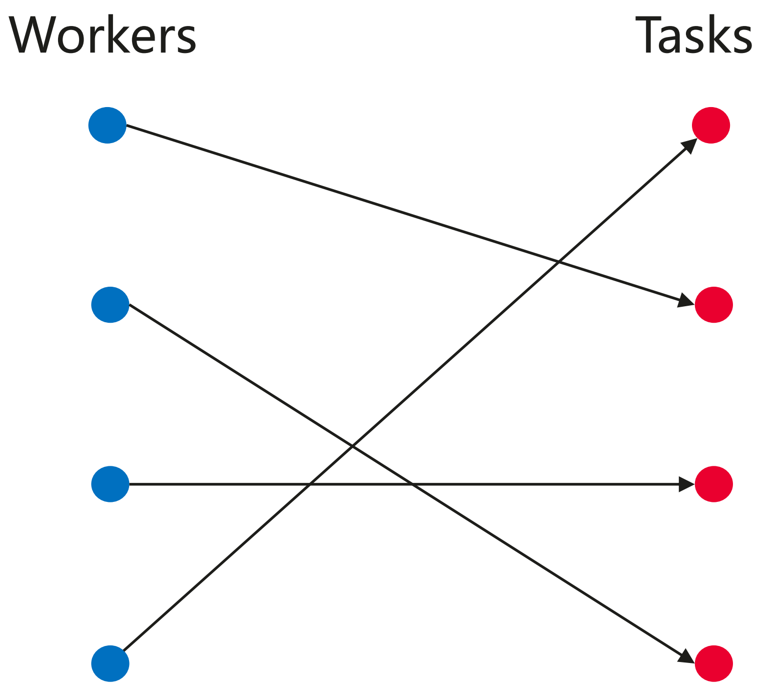
	6		1		4		5	
		8	3		5	6		
2								1
8			4		7			6
		6			3			
7			9		1			4
5								2
		7	2		6	9		
	4		5		8		7	

Assignment problem: minimal cost

Very simple formulation

Given:
- set of workers $\boldsymbol{W}$
- set of tasks $\boldsymbol{N}, |\mathbf{W}| = |\mathbf{N}|$
- cost function $c: W \times N \to \mathbb{Z}$

Goal:
Assign worker on each task with minimal cost.



Assignment problem: minimal cost

Linear programming model

<u>Variables</u>

- $\boldsymbol{x_{ij}} \in [\mathbf{0}, \mathbf{1}]$ -- equals to 1, if worker $i \in W$ assigned on task $j \in N$.

<u>Objective</u>

$$\min \sum_{i \in W} \sum_{j \in N} c_{ij} x_{ij}$$

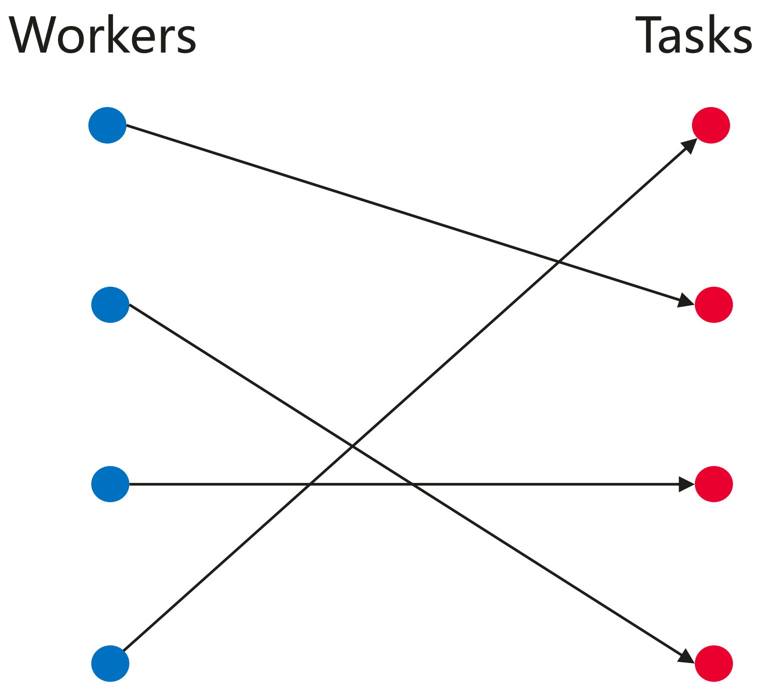
<u>Constraints</u>

- Each worker should be assigned on exactly one task

$$\forall i \in W: \sum_{j \in N} x_{ij} = 1.$$

- Each task should be assigned to only one worker

$$\forall j \in N: \sum_{i \in W} x_{ij} = 1.$$

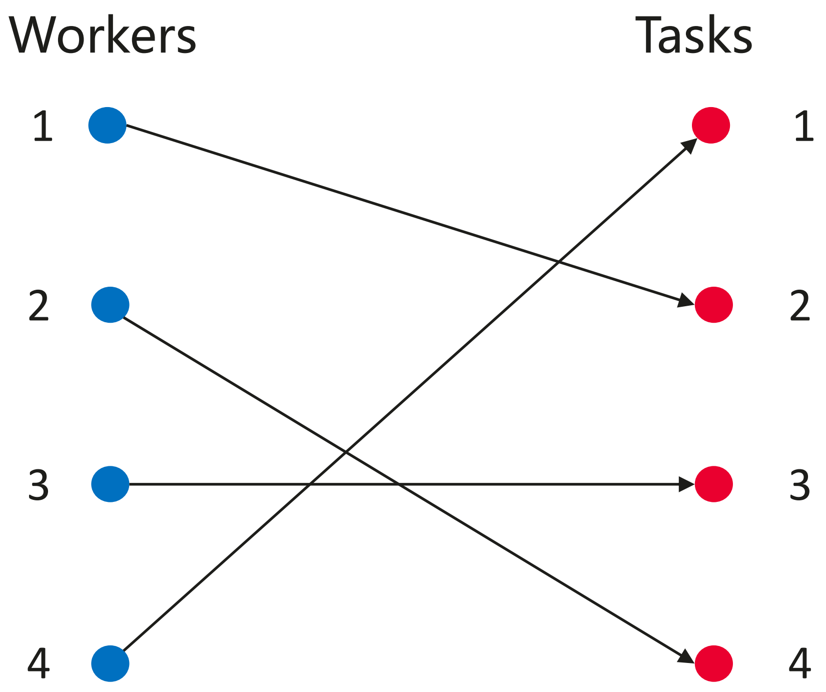


Assignment problem: minimal cost

Example

- $W = \{1, 2, 3, 4\}$
- $N = \{1, 2, 3, 4\}$,
- Cost function

	1	**2**	**3**	**4**
1	1	3	1	1
2	2	5	4	7
3	4	4	3	2
4	6	4	7	3



Solution cost: 19

Assignment problem: minimal cost

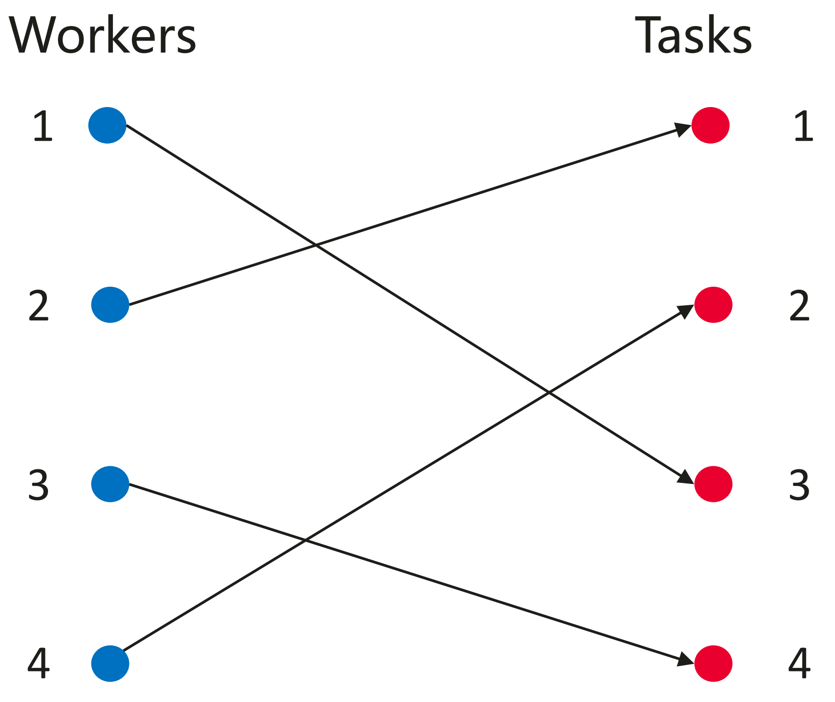
Linear program with [0,1] variables

```
% DATA
% num workers
int: w;
% num tasks (we know that n = w!)
int: n;
% cost function matrix
array[1..w,1..n] of int: c;

% VARIABLES
% if worker j is assigned on task i
array[1..w,1..n] of var 0..1: x;

% OBJECTIVE
% minimize total cost
solve minimize sum(i in 1..w, j in 1..n)(c[i,j]*x[i,j]);

%CONSTRAINTS
% all workers are assigned on exactly one task
constraint forall (i in 1..w) (
  sum(j in 1..n) (x[i,j]) = 1
);
% all tasks are assigned to exactly one worker
constraint forall (j in 1..n) (
  sum(i in 1..w) (x[i,j]) = 1
);
```



Solution cost: 9.0

LP solving using MiniZinc

Attach data

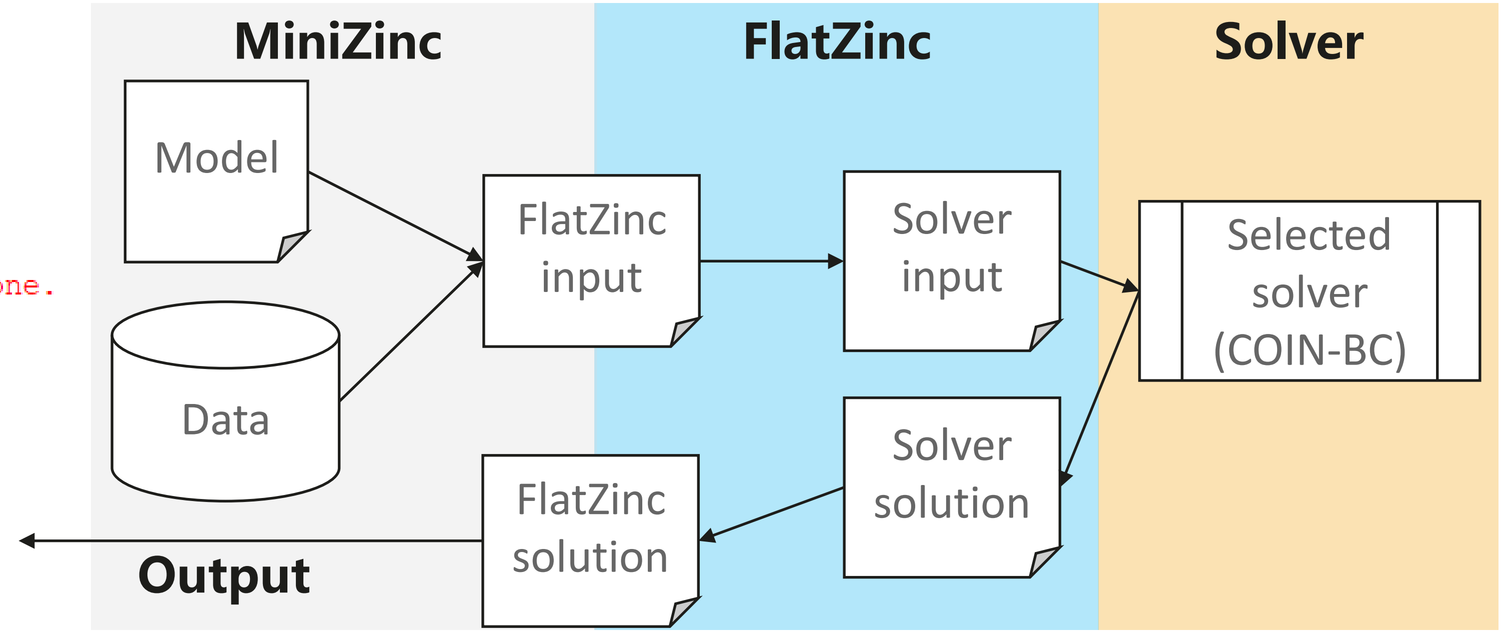
```
% DATA
% num workers
int: w;
% num tasks (we know that n = w!)
int: n;
% cost function matrix
array[1..w,1..n] of int: c;
```

If data is not defined in model .mzn, it should be given in format .dzn (or .json)

```
% num workers
w = 4;
% num tasks
n = 4;
% cost
c=[|
   1, 3, 1, 1|
   2, 5, 4, 7|
   4, 4, 3, 2|
   6, 4, 7, 3|];
```

Solve

```
%%%mzn-stat: flatFloatConstraints=9
%%%mzn-stat: flatFloatVars=17
%%%mzn-stat: flatTime=0.146882
%%%mzn-stat: method=minimize
%%%mzn-stat: paths=0
%%%mzn-stat-end
  MIPWrapper: adding the 17 Phase-1 variables... done.
  MIPosicbcWrapper: adding constraints physically... done.
 Model creation...
  Calling CbcMain with command 'cbc -solve -quit'...
%%%mzn-stat: nodes=0
%%%mzn-stat: objective=9
%%%mzn-stat: solveTime=0.003
%%%mzn-stat-end
```

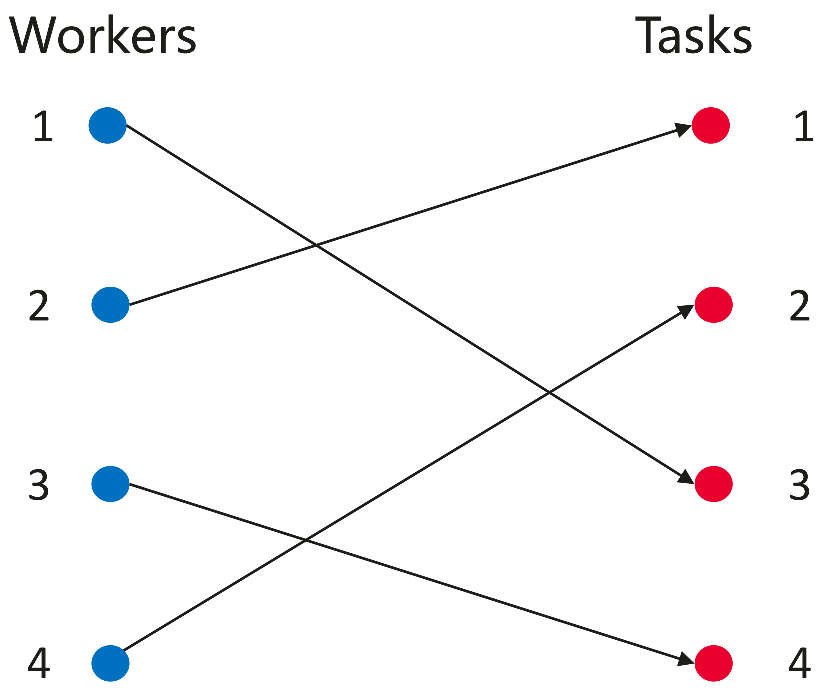


Assignment problem: minimal cost

Example

- $W = \{1, 2, 3, 4\}$
- $N = \{1, 2, 3, 4\}$,
- Cost function

	1	**2**	**3**	**4**
1	1	3	1	1
2	2	5	4	7
3	4	4	3	2
4	6	4	7	3



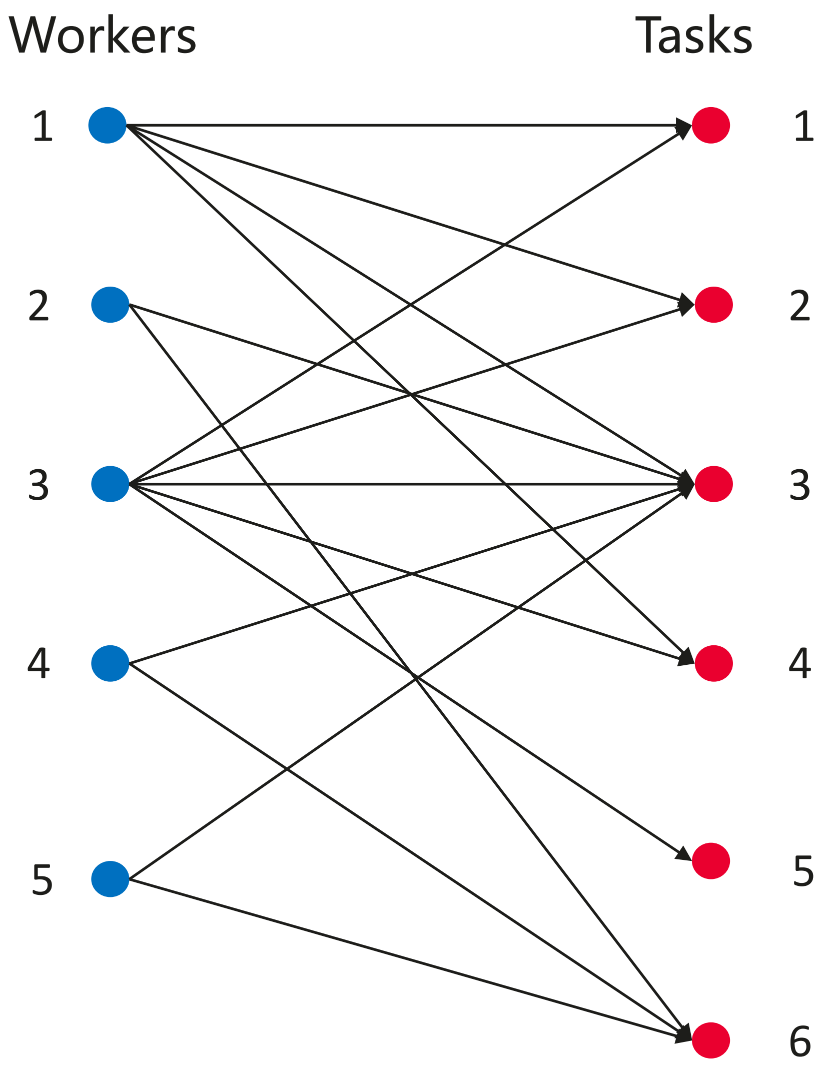
Optimal solution cost: 9.0

Assignment problem: maximum matching

Very simple formulation

Given:
- set of workers $\boldsymbol{W}$
- set of tasks $\boldsymbol{N}, |\mathbf{W}| = |\mathbf{N}|$
- Adjacency function $e: W \times N \rightarrow \{0,1\}$

Goal:
Maximize number of tasks to be processed.



Assignment problem: maximum matching

Linear programming model

<u>Variables</u>

- $\boldsymbol{x_{ij}} \in [\mathbf{0}, \mathbf{1}]$ -- equals to 1, if worker $i \in W$ assigned on task $j \in N$.

<u>Objective</u>

$$\max \sum_{i \in W} \sum_{j \in N} x_{ij}$$

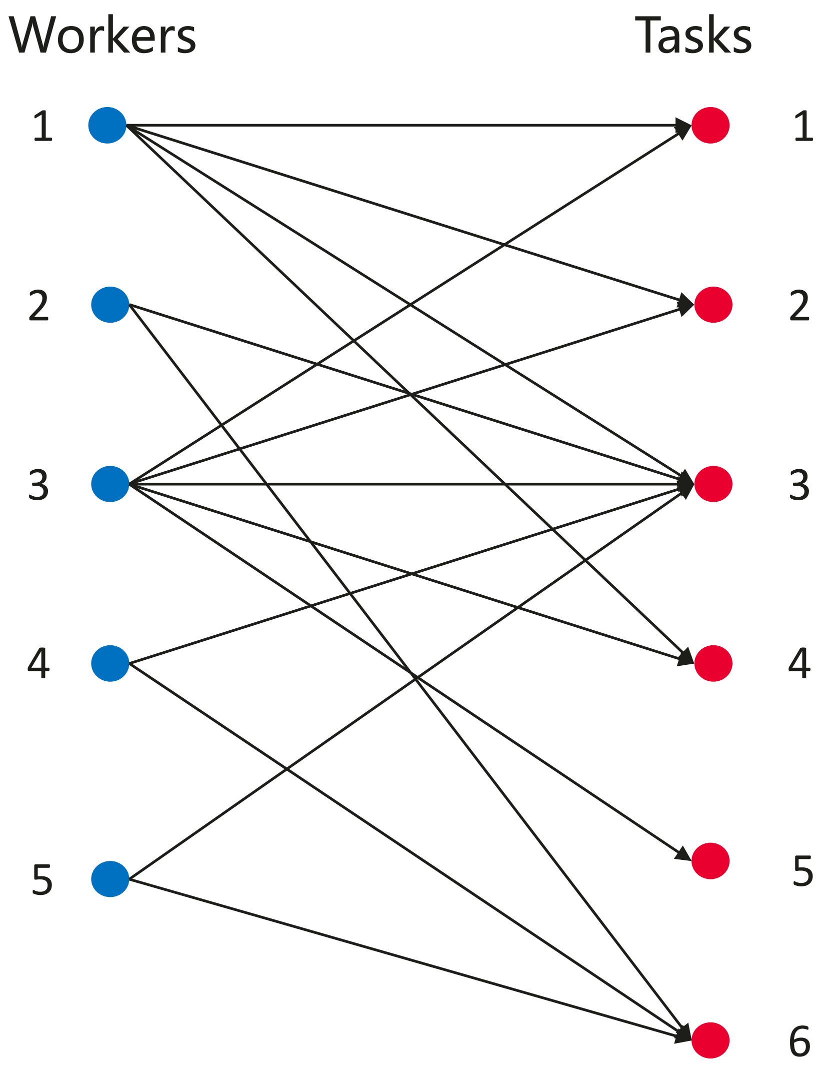
<u>Constraints</u>

- Each worker can be assigned on not more than one task available for him

$$\forall i \in W: \sum_{j \in N | e_{ij}=1} x_{ij} \leq 1.$$

- Each task should be assigned to not more than one worker which can process it

$$\forall j \in N: \sum_{i \in W | e_{ij}=1} x_{ij} \leq 1.$$



Assignment problem: maximum matching

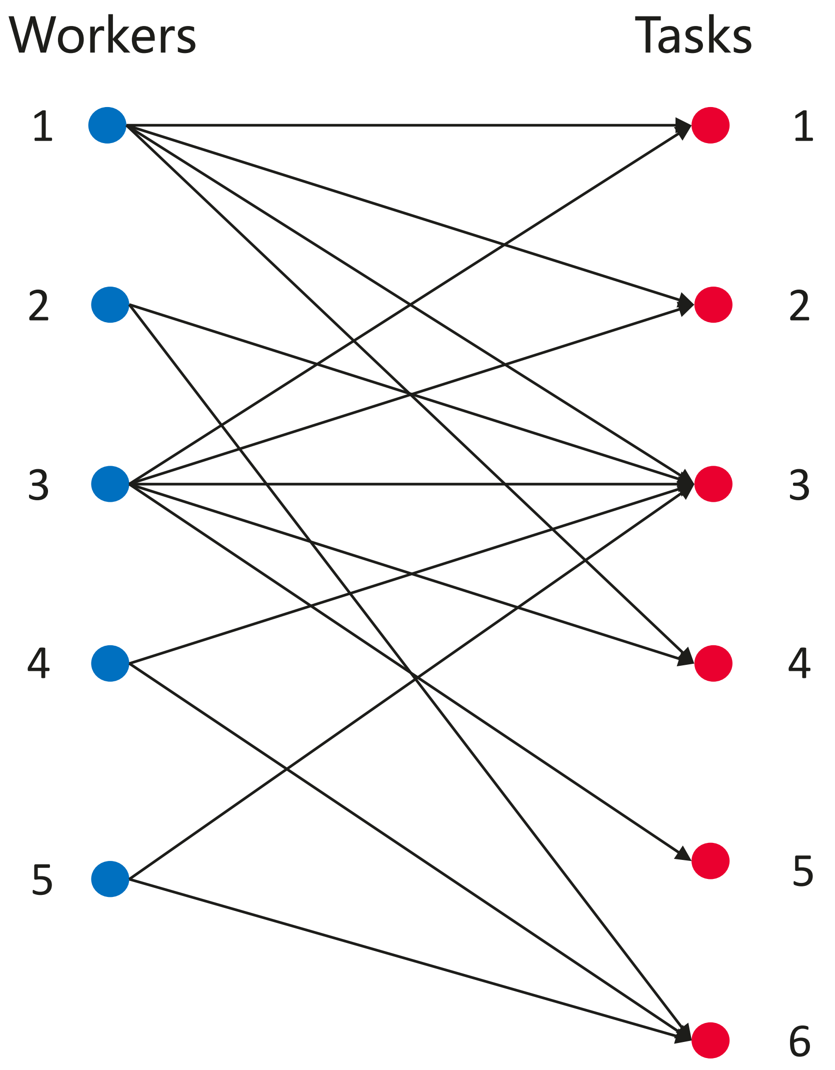
Linear program with [0,1] variables

```
% DATA
% num workers
int: w;
% num tasks
int: n;
% adjacency matrix
array[1..w,1..n] of int: e;

% VARIABLES
% if worker j can be assigned on task i
array[1..w,1..n] of var 0.0..1.0: x;

% OBJECTIVE
% maximize number of tasks to be processed
solve maximize sum(i in 1..w, j in 1..n)(x[i,j]);

%CONSTRAINTS
% all workers can be assigned on not more than one task
constraint forall (i in 1..w) (
  sum(j in 1..n) (x[i,j]) <= 1
);
% all tasks can be processed by not more than one worker
constraint forall (j in 1..n) (
  sum(i in 1..w) (x[i,j]) <= 1
);
% adjacency constraint
constraint forall (i in 1..w, j in 1..n) (
  x[i,j] <= e[i,j]
);
```



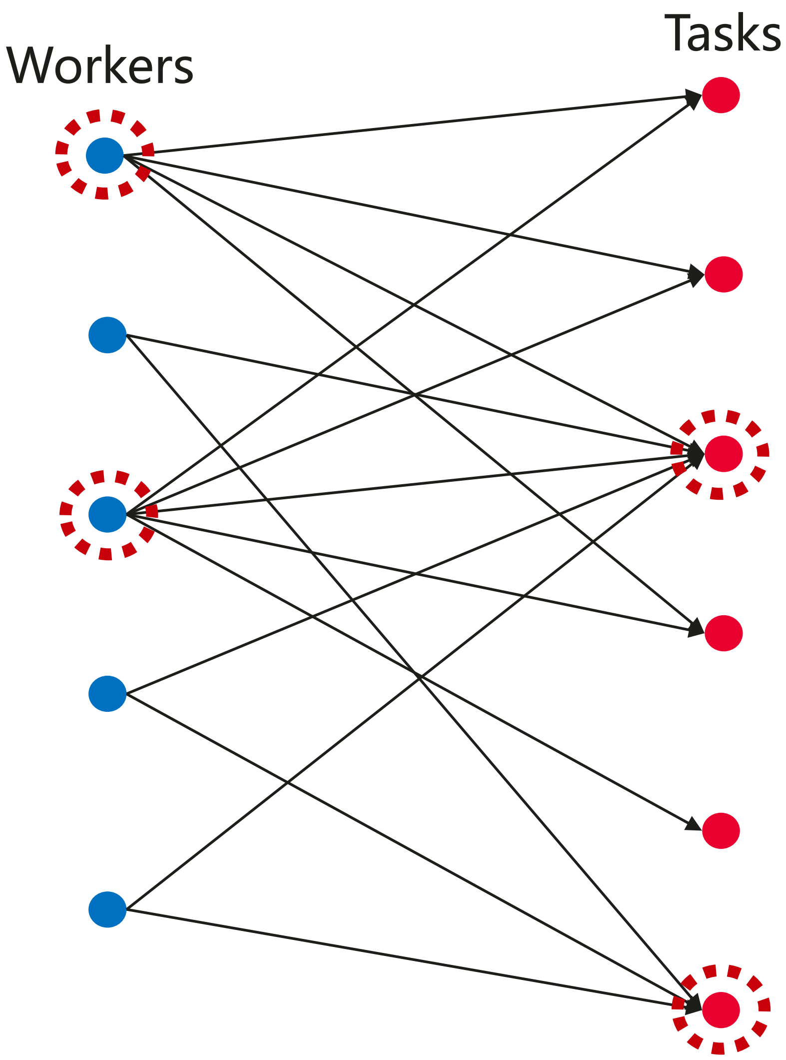
Objective: 4.0

Assignment problem: maximum matching

Graph theory

- Kőnig's theorem (1931)

In any bipartite graph, the number of edges in a maximum **matching** equals the number of vertices in a minimum **vertex cover**.



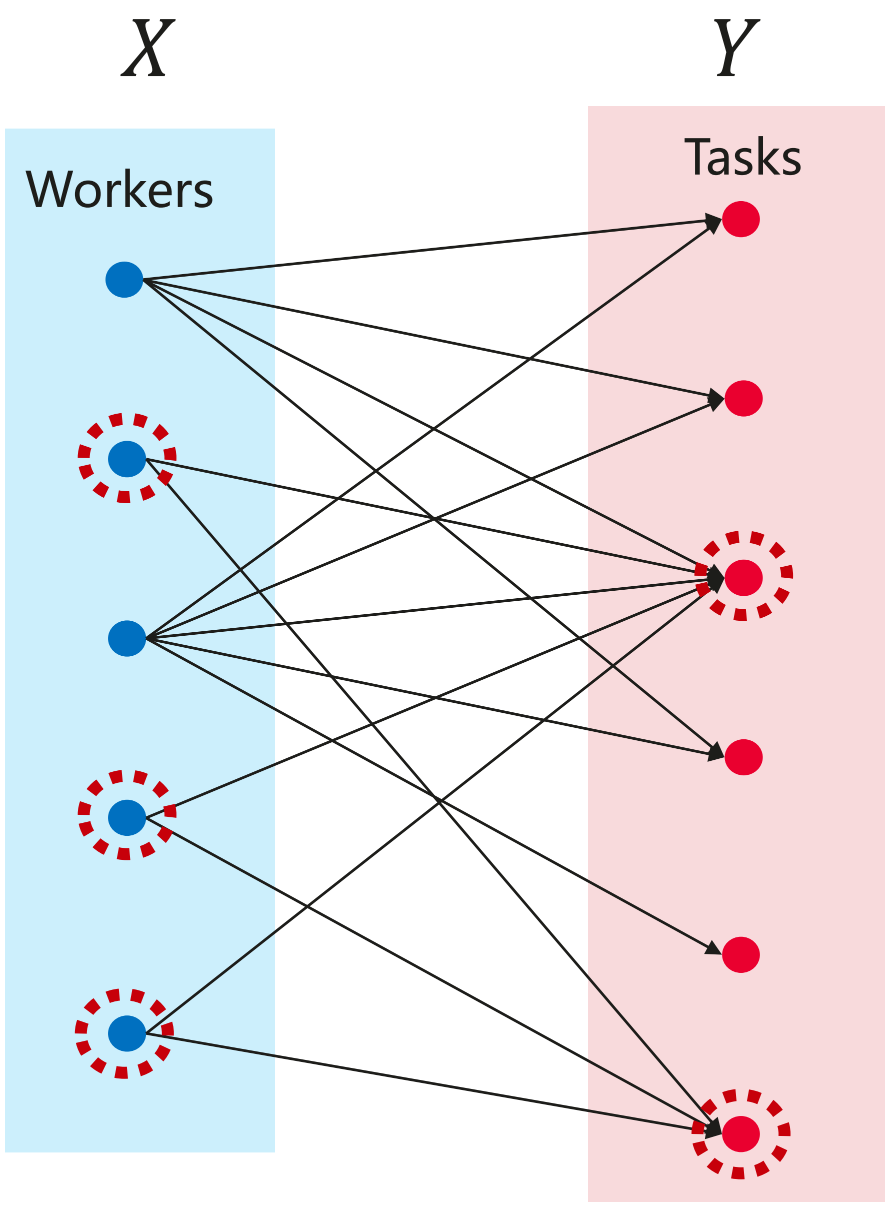
Minimum vertex cover size: 4

Assignment problem: maximum matching

Graph theory

- Kőnig's theorem (1931)
- Hall's marriage theorem (1935)

Suppose we have bipartite graph with bipartite sets X and Y. Let $N_G(A)$ - neighborhood of the subset of vertices A in the graph G.
There is an X-perfect matching if and only if for every subset A of X holds $|A| \leq |N_G(A)|$



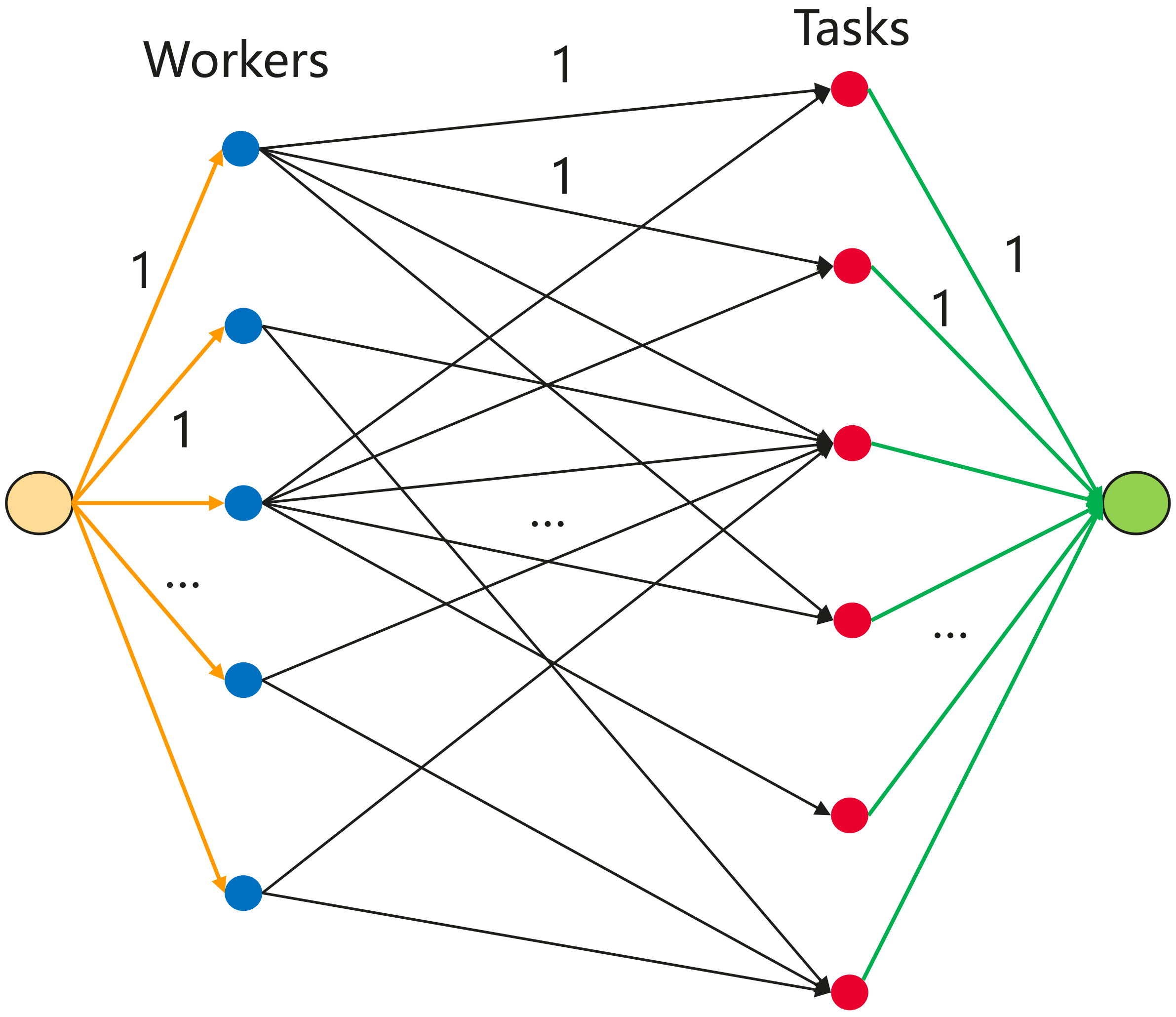
Perfect matching is impossible!

Assignment problem: maximum matching

Graph theory

- Kőnig's theorem (1931)
- Hall’ s marriage theorem (1935)
- Flow-based algorithms
  - Ford-Fulkerson (1955)
  - Dinic’ s algorithm (1970)
  - Edmonds–Karp algorithm (1972)

Finding max flow from source to sink which is related to optimal problem solution.

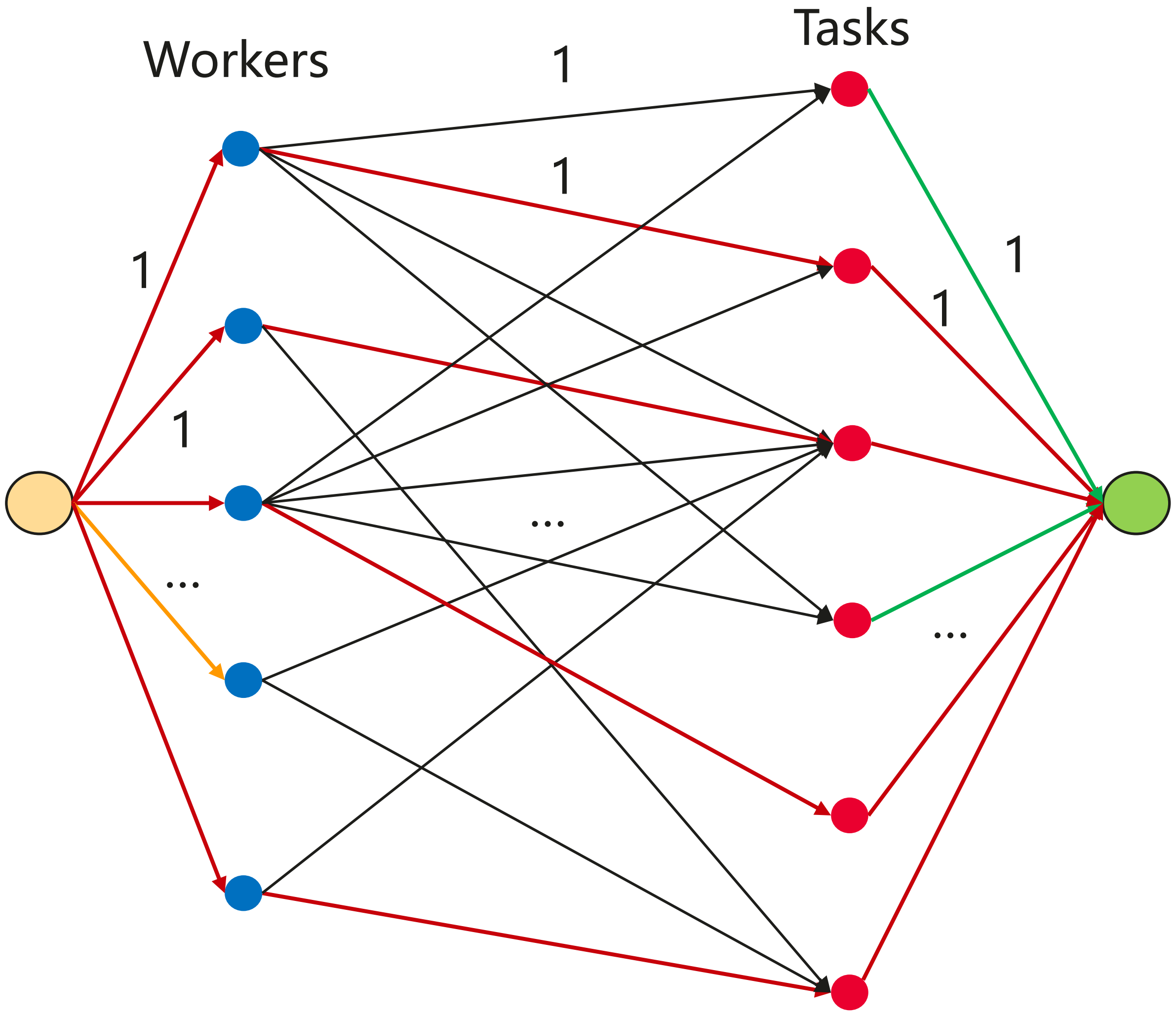


Assignment problem: maximum matching

Graph theory

- Kőnig's theorem (1931)
- Hall’ s marriage theorem (1935)
- Flow-based algorithms
  - Ford-Fulkerson (1955)
  - Dinic’ s algorithm (1970)
  - Edmonds–Karp algorithm (1972)

Finding max flow from source to sink which is related to optimal problem solution.



LP model structure

```
% DATA
% num workers
int: w;
% num tasks
int: n;
% adjacency matrix
array[1..w,1..n] of int: e;

% VARIABLES
% if worker j can be assigned on task i
array[1..w,1..n] of var 0.0..1.0: x;

% OBJECTIVE
% maximize number of tasks to be processed
solve maximize sum(i in 1..w, j in 1..n)(x[i,j]);

%CONSTRAINTS
% all workers can be assigned on not more than one task
constraint forall (i in 1..w) (
  sum(j in 1..n) (x[i,j]) <= 1
);
% all workers can be processed by not more than one worker
constraint forall (j in 1..n) (
  sum(i in 1..w) (x[i,j]) <= 1
);
% adjacency constraint
constraint forall (i in 1..w, j in 1..n) (
  x[i,j] <= e[i,j]
);
```

Define **data** which will be used in variable arrays sizes, domains, objectives and constraints

Define **variables**. If you want to solve Linear programming model, then variable domains should be defined by **float numbers**. Otherwise problem will be interpreted as MILP.

Set satisfiability solving mode or choose optimization direction (min/**max**). In case you solving optimization problem, objective function formula should be given.

Set up **constraints** to make problem statement correct.

Diet problem

Stigler diet problem

For a moderately active man weighing 154 pounds, how much of each of 77 foods should be eaten on a daily basis so that the man's intake of nine nutrients will be at least equal to the recommended dietary allowances (RDAs) suggested by the National Research Council in 1943, with the cost of the diet being minimal?

Stated by **George Stigler**, a 1982 Nobel Laureate in economics.

Nutrient	Daily consumption
Calories	3,000 Calories
Protein	70 grams
Calcium	0.8 grams
Iron	12 milligrams
Vitamin A	5,000 IU
Thiamine (Vitamin B_1)	1.8 milligrams
Riboflavin (Vitamin B_2)	2.7 milligrams
Niacin	18 milligrams
Ascorbic Acid (Vitamin C)	75 milligrams

Diet problem

Stigler diet problem

For a moderately active man weighing 154 pounds, how much of each of 77 foods should be eaten on a daily basis so that the man's intake of nine nutrients will be at least equal to the recommended dietary allowances (RDAs) suggested by the National Research Council in 1943, with the cost of the diet being minimal?

Stated by **George Stigler**, a 1982 Nobel Laureate in economics.

Solution

Food	Annual	Daily (gram)	Annual Cost (in $ 1939 ~0.1$ 2022)
Wheat Flour	370 lb.	459	$13.33
Evaporated Milk	57 cans	62	$3.84
Cabbage	111 lb.	138	$4.11
Spinach	23 lb.	29	$1.85
Dried Navy Beans	285 lb.	354	$16.80
Total Cost			$39.93

Expectations

Reality

Diet problem

Stigler diet problem
For a moderately active man weighing 154 pounds, how much of each of 77 foods should be eaten on a daily basis so that the man's intake of nine nutrients will be at least equal to the recommended dietary allowances (RDAs) suggested by the National Research Council in 1943, with the cost of the diet being minimal?

Stated by **George Stigler**, a 1982 Nobel Laureate in economics.

In 1944, Stigler calculated the best answer he could, noting with sadness:
"...there does not appear to be any direct method of finding the minimum of a linear function subject to linear conditions."

In 1947, Jack Laderman used the simplex method (then, a recent invention!) to determine the optimal solution. It took **120 man days of nine clerks** on desk calculators to arrive at the answer.

Diet problem

Do it yourself!

Stigler diet data can be found here

https://developers.google.com/optimization/lp/stigler_diet

```
%DATA
% planning horizon
int: T = 365;
% food ids: 1 to F
int: F = 77;
% nutrient ids: 1 to N
int: N = 9;
% allowed violation of nutrition target
float: delta = 0.1;
% nutrition target per day
array[1..N] of float: k;
% food nutrition per 1$ (1939)
array[1..F, 1..N] of float: s;
% VARIABLES

% OBJECTIVE - MINIMIZE TOTAL COST

% CONSTRAINTS
```

Template and data will be uploaded on Google classroom