Modeling and Optimization with OPL

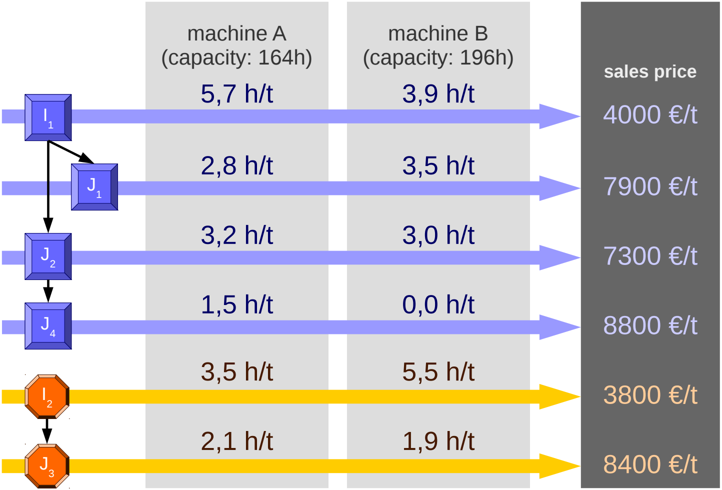
4 Optimization of Graph Problems

Andreas Popp

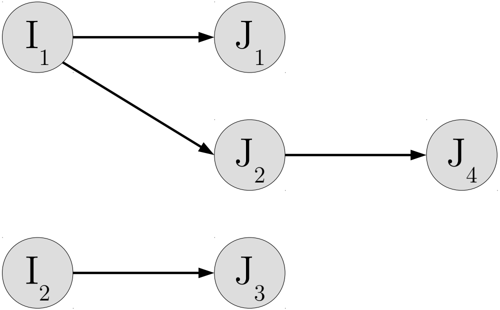
Inhalt

4.1 Short introduction into graph theory

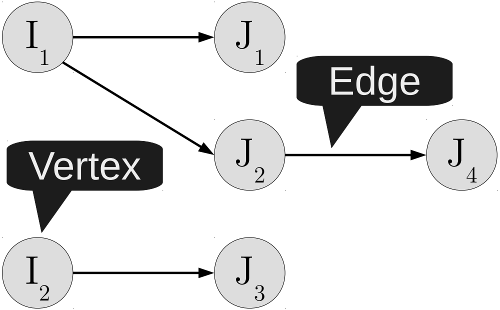
Example: Lewig Adelburg



Concept of a graph: components



Concept of a graph: components



Concept of a graph: formal description

- **Directed** Graphs are defined as a tuple $G = (V, E)$ with a set of vertices V and a set of edges $E \subseteq V \times V$.

  in example:
  $G = (\{I_1, I_2, J_1, J_2, J_3, J_4\}, \{(I_1, J_1), (I_1, J_2), (I_2, J_3), (J_2, J_4)\})$
- **Undirected** graphs are graphs whose edges do not have a specific direction.
- **Weighted** graphs are defined as a tuple $G = (V, E, g)$ with a set of vertices V, a set of edges $E \subseteq V \times V$ and a weight function $g : E \to \mathbb{R}$.

4.2 Representation of graphs in OPL

Sequence dependet production problem

Index sets:

I set of products

R set of ressources

Parameters:

p_i price of product $i \in I$

c_r capacity of ressource $r \in R$

v_{ri} capacity consumption of product $i \in I$ on ressource $r \in R$

E set of edges in the sequence graph

Decision variables:

x_i production quantity of product $i \in I$

Modellbeschreibung:

$$\begin{aligned}
\max \quad & \sum_{i \in I} p_i \cdot x_i & & \\
s.t. \quad & \sum_{i \in I} v_{ri} \cdot x_i \leq c_i & \forall r \in R \quad & \text{(I)} \\
& x_i \geq \sum_{(i,j) \in E} x_j & \forall i \in I \quad & \text{(II)} \\
& x_i \geq 0 & \forall i \in I &
\end{aligned}$$

A. Popp

Graphs in optimization problems

Application example for graphs

$$x_i \geq \sum_{(i,j) \in E} x_j \qquad \forall i \in I$$

Question: How can the graph be represented in the optimization model?

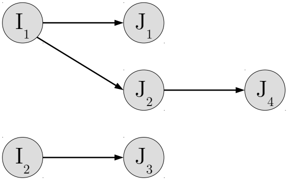
The adjacency matrix

Definition: adjacency matrix

The adjacency matrix of a grap $G = (V, E)$ with $V = \{V_1, \ldots, V_N\}$ is a quadratic $N \times N$-Matrix (a_{ij}), which holds:

$$a_{ij} = \begin{cases} 1 & \text{Edge } V_i \rightarrow V_j \text{ exists} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

Adjacency matrix in the example



↓ Translation into adjacency matrix ↓

$$
\begin{array}{c c}
 & \begin{array}{cccccc} I_1 & I_2 & J_1 & J_2 & J_3 & J_4 \end{array} \\
\begin{array}{c} I_1 \\ I_2 \\ J_1 \\ J_2 \\ J_3 \\ J_4 \end{array} &
\begin{pmatrix}
0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}
\end{array}
$$

A. Popp

Application of adjacency matrixes in optimization problems

```
{string} I = ...;
int a [I,I] = [
  [0, 0, 1, 1, 0, 0],
  [0, 0, 0, 0, 1, 0],
  [0, 0, 0, 0, 0, 0],
  [0, 0, 0, 0, 0, 1],
  [0, 0, 0, 0, 0, 0],
  [0, 0, 0, 0, 0, 0],
];
```

$$x_i \geq \sum_{(i,j)\in E} x_j \qquad \forall i \in I$$

↓ OPL ↓

```
forall(i in I)
  x[i] >= sum (j in I)(a[i,j]*x[j]);
```

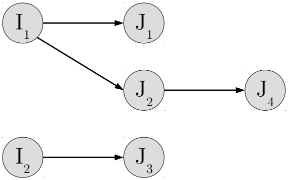

A. Popp

Adjacency lists

Definition: adjacency lists

The adjacency list of a vertex $v \in V$ of a graph $G = (V, E)$ is a set $A_v \subseteq V$, which contains all successors of v.

Adjacency lists in the example



$$
\begin{array}{ll}
A_{I_1} = \{J_1, J_2\} & A_{I_2} = \{J_3\} \\
A_{J_1} = \{\} & A_{J_2} = \{J_4\} \\
A_{J_3} = \{\} & A_{J_4} = \{\}
\end{array}
$$

Application of adjacency lists in optimization problems

```
{string} I = ...;
{string} A[I] = [
  {"J1", "J2"},
  {"J3"},
  {},
  {"J4"},
  {},
  {}
];
```

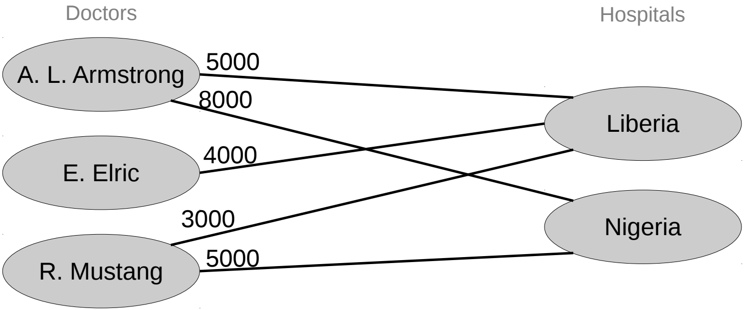
$$x_i \geq \sum_{(i,j) \in E} x_j \qquad \forall i \in I$$

↓ OPL ↓

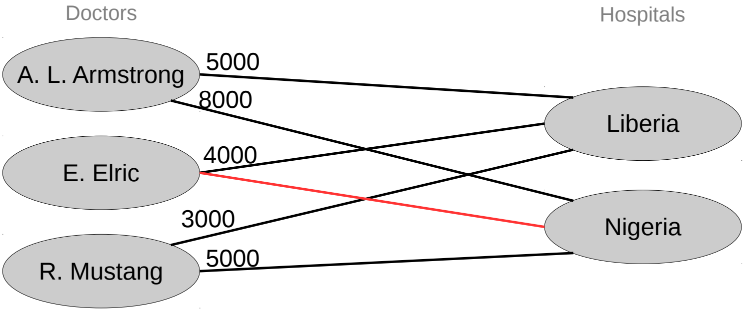
```
forall (i in I)
  x[i] >= sum(j in A[i])(x[j]);
```

4.3 OPL: custom tuples as data structure

Example: Relieve Doctors



Example: Relieve Doctors



Model: Assignment problem

Index sets:

R set of ressources

T set of tasks

Parameters:

E set of egdes in the assignment graph

c_{rt} cost of each edge $(r,t) \in E$

Decision variables:

x_{rt} binary variable representing the choice of edge $(r,t) \in E$

Model description:

$$
\begin{aligned}
\min \quad & \sum_{(r,t)\in E} c_{rt} \cdot x_{rt} & & \\
s.t. \quad & \sum_{(r,t)\in E} x_{rt} = 1 & \forall t \in T & \qquad \text{(I)} \\
& \sum_{(r,t)\in E} x_{rt} \leq 1 & \forall r \in R & \qquad \text{(II)} \\
& x_{rt} \in \{0,1\} & \forall (r,t) \in E &
\end{aligned}
$$

Representation of missing edges in OPL

- Assign prohibitively high costs to missing edges. Disadvantages:
  - unnecessary binary variables
  - susceptible to machine rounding errors
  - only applicable to weighted graphs (if at all)
- Adjacency matrix. Disadvantages:
  - unnecessary binary variables
- Adjacency lists. Disadvantages:
  - `x[r in R][t in A[r]]` → Error: „Variable indexer size not allowed for a generic array.“

Tuple data structure

Tuples are custom data structurs, consisting of elements of other data types.

Definition of a new tuple data type

```
tuple Name_of_the_tuple_data_type {
  Data_type_of_1st_Element Name_of_1st_Element;
  Data_type_of_2nd_Element Name_of_2nd_Element;
  ...
}
```

Example: Edges as tuple data type

```
{string} V = {"A", "B", "C"};
tupel edge {
  string start;
  string end;
};
```

Tuple literals and elements

In a tuple data type's literals the elements are sorted into angle brackets.

Example: Definition of an edge as literal

```
edge e = <"A", "B">;
```

Single elements of a tuple data type are adressed with a dot.

Example: getting the starting vertex of an edge

`e.start` $\rightarrow$ `"A"`

Application of tuple data type (Alternative 1)

Vertices and Edges shall be defined as above.

Application example

$$\sum_{(r,t)\in E} x_{rt} = 1 \qquad \forall t \in T$$

↓ OPL ↓

```
forall(t in T)
  sum(<r,t> in E)(x[<r,t>]) == 1;
```

4.4 OPL: conditional operators

Conditional operators

Using a colon, we can apply conditions to iteration indexes, which have to be fulfilled for an index to be incorporated by the operator:

`sum(`*iteration index* `in` *index set* `:` *condition*`)`

resp.

`forall(`*iteration index* `in` *index set* `:` *condition*`)`

Conditions are logical expressions (not boolean decision variables!)

Construtction of conditions

Literals for logical values

`true`, `false`

Comparison operators for logical values

math. notation	$=$	$\neq$	$\leq$	$<$	$\geq$	$>$
OPL Syntax	`==`	`!=`	`<=`	`<`	`>=`	`>`

Logical operators for logical values

math. notation	$\neg$	$\wedge$	$\vee$	$\veebar$
OPL syntax	`!`	`&&`	`||`	`!=`

Application of tuple data type (Alternative 2)

Vertices and Edges shall be defined as above.

Application example

$$\sum_{(r,t)\in E} x_{rt} = 1 \qquad \forall t \in T$$

↓ OPL ↓

```
forall(t in T)
  sum(e in E : e.task == t)(x[e]) == 1;
```