

Modeling and Optimization with OPL

4 Optimization of Graph Problems

Andreas Popp



These slides are licensed under a Creative Commons Attribution-ShareAlike 4.0 International License.

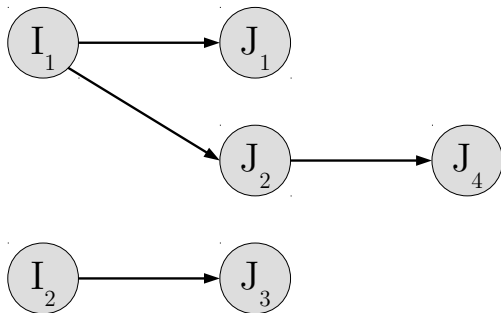
4.2 Representation of graphs in OPL

4.4 OPL: conditional operators

4.1 Short introduction into graph theory

CC-BY-SA
A. Popp

Concept of a graph: components



4.1 Short
introduction into
graph theory

4.2 Representation
of graphs in OPL

4.3 OPL: custom
tuples as data
structure

4.4 OPL:
conditional
operators

CC-BY-SA
A. Popp

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 $g : E \rightarrow \mathbb{R}$.

Sequence dependent production problem

Index sets:

I set of products

R set of resources

Parameters:

p_i price of product $i \in I$

c_r capacity of resource $r \in R$

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

E set of edges in the sequence graph

Decision variables:

x_i production quantity of product $i \in I$

Modellbeschreibung:

$$\max \sum_{i \in I} p_i \cdot x_i$$

$$\text{s.t.} \quad \sum_{i \in I} v_{ri} \cdot x_i \leq c_i \quad \forall r \in R \quad (\text{I})$$

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

$$x_i \geq 0 \quad \forall i \in I$$

4.1 Short
introduction into
graph theory

4.2 Representation
of graphs in OPL

4.3 OPL: custom
tuples as data
structure

4.4 OPL:
conditional
operators

Graphs in optimization problems

Application example for graphs

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

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

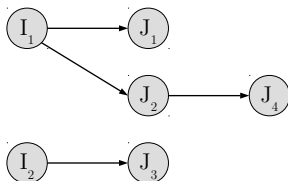
4.1 Short
introduction into
graph theory

4.2 Representation
of graphs in OPL

4.3 OPL: custom
tuples as data
structure

4.4 OPL:
conditional
operators

Adjacency matrix in the example



↓ Translation into adjacency matrix ↓

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

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],
];
```

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

↓ OPL ↓

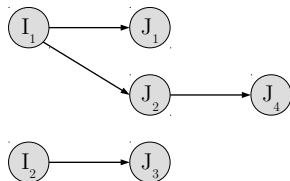
```
forall(i in I)
  x[i] >= sum (j in I)(a[i,j]*x[j]);
```

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 .

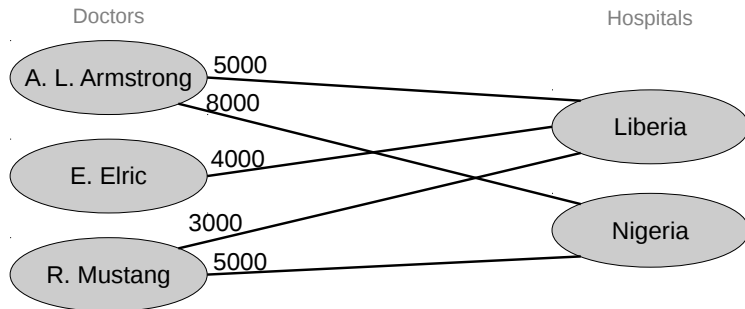
Adjacency lists in the example



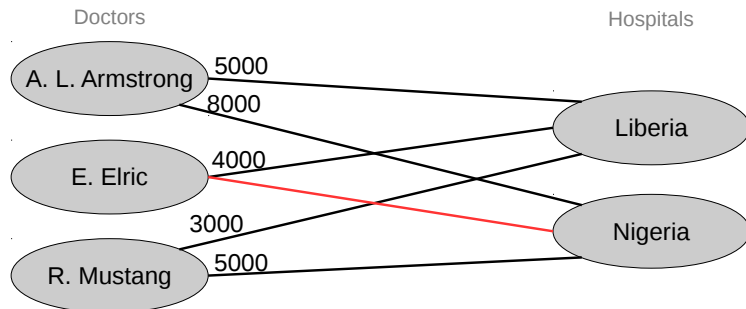
$$\begin{array}{ll} A_{I_1} = \{J_1, J_2\} & A_{I_2} = \{J_3\} \\ A_{J_1} = \{\} & A_{J_2} = \{J_4\} \\ A_{J_3} = \{\} & A_{J_4} = \{\} \end{array}$$

4.3 OPL: custom tuples as data structure

Example: Relieve Doctors



Example: Relieve Doctors



4.3 OPL: custom tuples as data structure

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

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

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

$$\sum_{(r,t) \in E} x_{rt} \leq 1 \quad \forall r \in R \quad (\text{II})$$

$$x_{rt} \in \{0, 1\} \quad \forall (r, t) \in E$$

- 18/25 ◀ ◻ ▶ ◀ ◻ ▶ ◀ ≡ ▶ ◀ ≡ ▶ ≡ ↺ 🔍 ↻

Definition of a new tuple data type

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

```
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"};
tuple edge {
    string start;
    string end;
};
```

4.3 OPL: custom tuples as data structure

Example: getting the starting vertex of an edge

20/25 ◀ ◻ ▶ ◀ ◻ ▶ ◀ ≡ ▶ ◀ ≡ ▶ ≡ ↺ 🔍 ↻

Application of tuple data type (Alternative 1)

Vertices and Edges shall be defined as above.

Application example

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

↓ OPL ↓

```
forall(t in T)
  sum(<r,tt
```

4.1 Short
introduction into
graph theory

4.2 Representation
of graphs in OPL

4.3 OPL: custom
tuples as data
structure

4.4 OPL:
conditional
operators

4.4 OPL: conditional operators

$$\text{sum}(\textit{iteration index in index set} : \textit{condition})$$

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

23/25 ◀ ◻ ▶ ◀ ◻ ▶ ◀ ≡ ▶ ◀ ≡ ▶ ≡ ↺ 🔍 ↻

Construction 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	$\underline{\vee}$
OPL syntax	!	&&		!=

4.1 Short
introduction into
graph theory

4.2 Representation
of graphs in OPL

4.3 OPL: custom
tuples as data
structure

4.4 OPL:
conditional
operators

Application of tuple data type (Alternative 2)

Vertices and Edges shall be defined as above.

Application example

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

↓ OPL ↓

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

4.1 Short
introduction into
graph theory

4.2 Representation
of graphs in OPL

4.3 OPL: custom
tuples as data
structure

4.4 OPL:
conditional
operators