Modellierung und Optimierung mit OPL

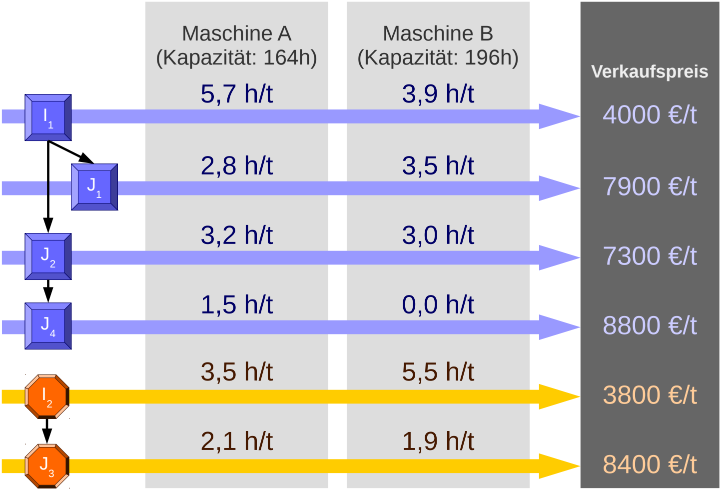
4 Optimierung von Graphenproblemen

Andreas Popp

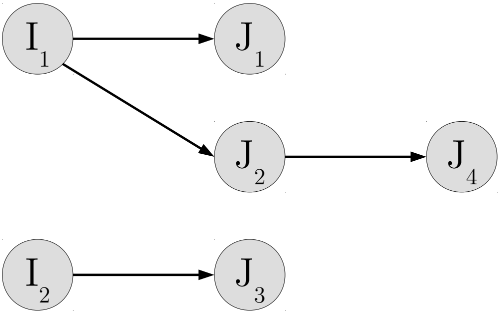
Inhalt

4.1 Kurzeinführung in die Graphentheorie

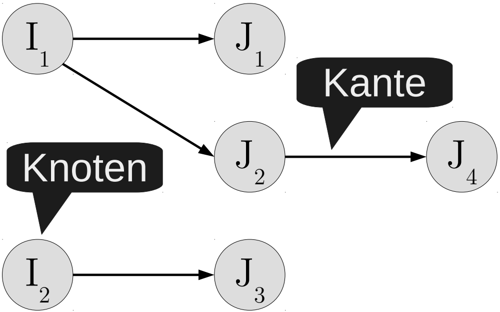
Beispiel: Lewig Adelburg



Graphenbegriff: Komponenten



Graphenbegriff: Komponenten



Graphenbegriff: Formale Beschreibung

- **Gerichtete** Graphen sind definiert als ein Tupel $G = (V, E)$ mit einer Knotenmenge V und einer Kantenmenge $E \subseteq V \times V$.

  Im Beispiel:

  $G = (\{I_1, I_2, J_1, J_2, J_3, J_4\}, \{(I_1, J_1), (I_1, J_2), (I_2, J_3), (J_2, J_4)\})$
- **Ungerichtete** Graphen sind Graphen, deren Kanten keine feste Richtung haben.
- **Gewichtete** Graphen sind definiert als ein Tupel $G = (V, E, g)$ mit einer Knotenmenge V, einer Kantenmenge $E \subseteq V \times V$ und einer Gewichtungsfunktion $g : E \to \mathbb{R}$.

4.2 Abbilden von Graphen in OPL

Reihenfolgeabhängiges Produktionsproblem

Indexmengen:

I Menge der Produkte
R Menge der Ressourcen

Parameter:

p_i Preis von Produkt $i \in I$
c_r Kapazität von Ressource $r \in R$
v_{ri} Kapazitätsverbrauch von Produkt $i \in I$ auf Ressource $r \in R$
E Menge der Kanten im Reihenfolgegraph

Entscheidungsvariablen:

x_i Produktionsmenge von Produkt $i \in I$

Modellbeschreibung:

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

Graphen in Optimierungsproblemen

Anwendungsbeispiel für Graphen

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

Frage: Wie kann der Graph in einem Optimierungsmodell abgebildet werden?

Die Adjazenzmatrix

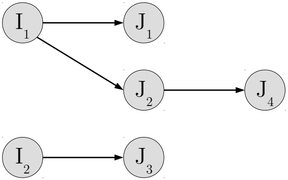
Definition: Adjazenzmatrix

Die Adjazenzmatrix eines Graphen $G = (V, E)$ mit $V = \{V_1, \ldots, V_N\}$ ist eine quadratische $N \times N$-Matrix (a_{ij}), für die gilt:

$$a_{ij} = \begin{cases} 1 & \text{Kante } V_i \rightarrow V_j \text{ existiert} \\ 0 & \text{sonst} \end{cases} \qquad (1)$$

A. Popp

Adjazenzmatrix im Beispiel



↓ Übersetzung in Adjazenzmatrix ↓

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

Anwendung von Adjazenzmatrizen in Optimierungsproblemen

```
{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 ↓

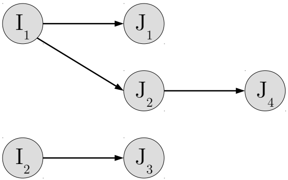
```
forall(i in I)
  x[i] >= sum (j in I)(a[i,j]*x[j]);
```

Adjazenzlisten

Definition: Adjazenzliste

Die Adjazenzliste eines Knoten $v \in V$ eines Graphen $G = (V, E)$ ist eine Menge $A_v \subseteq V$, welche alle Nachfolger von v beinhaltet.

Adjazenzlisten im Beispiel



$$A_{I_1} = \{J_1, J_2\} \qquad A_{I_2} = \{J_3\}$$
$$A_{J_1} = \{\} \qquad A_{J_2} = \{J_4\}$$
$$A_{J_3} = \{\} \qquad A_{J_4} = \{\}$$

Anwendung von Adjazenzlisten in Optimierungsproblemen

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

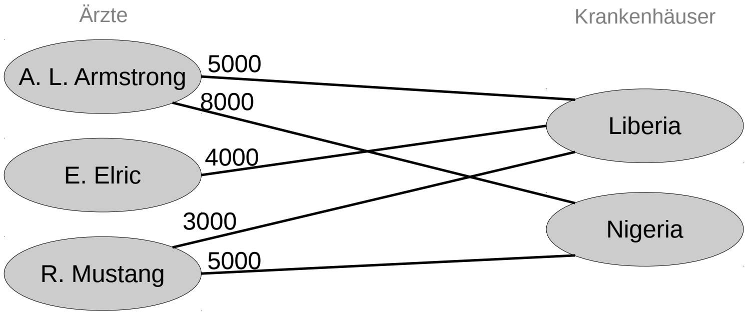
$$x_i \geq \sum_{(i,j) \in E} x_j \qquad \forall i \in I$$

↓ OPL ↓

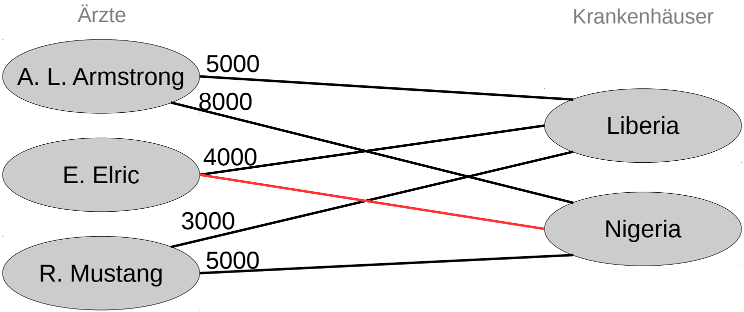
```
forall (i in I)
  x[i] >= sum(j in A[i])(x[j]);
```

4.3 OPL: Selbstdefinierte Tupel als Datenstruktur

Beispiel: Relieve Ärzte



Beispiel: Relieve Ärzte



Modell: Zuordnungsproblem

Indexmengen:

R Menge der Ressourcen

T Menge der Aufgaben

Parameter:

E Menge der Kanten im Zuordnungsgraphen

c_{rt} Kosten für die Auswahl der Kante $(r, t) \in E$

Entscheidungsvariablen:

x_{rt} Binärvariable, die angibt ob die Kante $(r, t) \in E$ ausgewählt wurde

Modellbeschreibung:

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

Abbildung der fehlenden Kanten in OPL

- Fehlende Kanten mit prohibitiv hohen Kosten versehen. Nachteile:
  - überflüssige Binärvariablen
  - anfällig für Maschinenrundungsfehler
  - wenn dann nur in gewichteten Graphen möglich
- Adjazenzmatrix. Nachteile:
  - überflüssige Binärvariablen
- Adjazenzlisten. Nachteile:
  - `x[r in R][t in A[r]]` $\rightarrow$ Fehlermeldung: „Die Größe des Variablenindexers ist für einen generischen Array nicht zulässig.“

Datenstruktur Tupel

Tupel sind selbstdefinierte Datenstrukturen, die aus Elementen anderer Datenstrukturen bestehen.

Definition eines neuen Tupel-Datentyps

```
tuple Name_der_Tupel-Datenstruktur {
  Datentyp_des_1._Elements Name_des_1._Elements;
  Datentyp_des_2._Elements Name_des_2._Elements;
  ...
}
```

Beispiel: Kanten als Tupel-Datentyp

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


A. Popp

Tupel-Literale und -Elemente

In Literalen eines Tupel-Datentyps werden die Elemente der Reihenfolge nach in spitzen Klammern zugeordnet.

Beispiel: Definition einer Kante als Literal

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

Einzelne Elemente eines Tupel-Datentyps werden mit einem Punkt angesprochen.

Beispiel: Auslesen des Startknotens einer Kante

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

Anwendung von Tupel-Datentypen (Variante 1)

Knoten und Kanten seien wie oben definiert.

Anwendungsbeispiel

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

↓ OPL ↓

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

4.4 OPL: Operatoren mit Bedingungen

Operatoren mit Bedingungen

Mithilfe eines Doppelpunkts lassen sich Bedingungen an Laufindizes stellen, die erfüllt werden müssen, damit der Index vom Operator berücksichtigt wird:

sum(*Laufindex* in *Indexmenge* : *Bedingung*)

bzw.

forall(*Laufindex* in *Indexmenge* : *Bedingung*)

Bedingungen sind logische Ausdrücke (keine Boolean-Entscheidungsvariablen!)

A. Popp

Konstruktion von Bedingungen

Literale für Wahrheitswerte

`true`, `false`

Vergleichsoperatoren für Wahrheitswerte

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

Logische Verknüpfungen für Wahrheitswerte

math. Schreibweise	$\neg$	$\wedge$	$\vee$	$\veebar$
OPL-Syntax	`!`	`&&`	`||`	`!=`

Anwendung von Tupel-Datentypen (Variante 2)

Knoten und Kanten seien wie oben definiert.

Anwendungsbeispiel

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

$\downarrow$ OPL $\downarrow$

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