Modellierung und Optimierung mit OPL

4 Optimierung von Graphenproblemen

Andreas Popp



Dieser Foliensatz ist lizenziert unter einer Creative Commons Namensnennung - Weitergabe unter gleichen Bedingungen 4.0 International Lizenz 4 Optimierung von Graphenproblemen

CC-BY-SA A. Popp

4.1 Kurzeinführung in die

4.2 Abbilden von Graphen in OPL

Selbstdefinierte
Tupel als
Datenstruktur

4.1 Kurzeinführung in die Graphentheorie

4.2 Abbilden von Graphen in OPL

4.3 OPL: Selbstdefinierte Tupel als Datenstruktur

4 Optimierung von Graphenproblemen

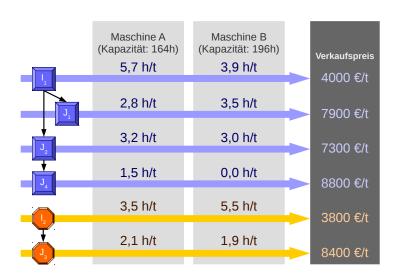
CC-BY-SA A. Popp

4.1 Kurzeinführung in die Graphentheorie

Graphen in OPL

4.3 OPL: Selbstdefinierte Tupel als Datenstruktur

Beispiel: Lewig Adelburg



4 Optimierung von Graphenproblemen

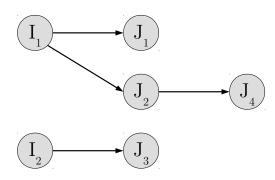
CC-BY-SA A. Popp

4.1 Kurzeinführung in die Graphentheorie

4.2 Abbilden von Graphen in OPL

I.3 OPL: Selbstdefinierte Fupel als Datenstruktur

Graphenbegriff: Komponenten



4 Optimierung von Graphenproblemen

CC-BY-SA A. Popp

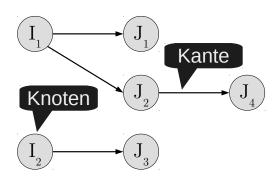
4.1 Kurzeinführung in die Graphentheorie

4.2 Abbilden von Graphen in OPL

4.3 OPL: Selbstdefinierte Tupel als Datenstruktur

4.4 OPL: Operatoren mi

Graphenbegriff: Komponenten



4 Optimierung von Graphenproblemen

CC-BY-SA A. Popp

- 4.1 Kurzeinführung in die Graphentheorie
- 4.2 Abbilden von Graphen in OPL
- 4.3 OPL: Selbstdefinierte Tupel als Datenstruktur
- 1.4 OPL: Operatoren mit Bedingungen

4.1 Kurzeinführung in die Graphentheorie

4.2 Abbilden von Graphen in OPL

4.3 OPL:
Selbstdefinierte
Tupel als
Datenstruktur

4.4 OPL: Operatoren mi Bedingungen

▶ **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 mit einer Knotenmenge V, einer Kantenmenge $E \subseteq V \times V$ und einer Gewichtungsfunktion $g : E \to \mathbb{R}$.

in die Graphentheorie

4.2 Abbilden von Graphen in OPL

4.3 OPL: Selbstdefinierte Tupel als Datenstruktur

4.4 OPL: Operatoren mit Bedingungen

4.2 Abbilden von Graphen in OPL

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:

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

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

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

$$x_i \ge 0 \qquad \forall i \in I$$

4.2 Abbilden von Graphen in OPL

4.3 OPL: Selbstdefinierte Tupel als Datenstruktur

4.4 OPL: Operatoren mit Bedingungen

Anwendungsbeispiel für Graphen

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

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

4.2 Abbilden von Graphen in OPL

4.3 OPL: Selbstdefinierte Tupel als Datenstruktur

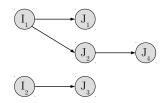
4.4 OPL:
Operatoren mit

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:

$$a_{ij} = \left\{ egin{array}{ll} 1 & {\sf Kante} \ V_i
ightarrow V_j \ {\sf existiert} \ 0 & {\sf sonst} \end{array}
ight. \eqno(1)$$

Adjazenzmatrix im Beispiel



\downarrow Übersetzung in Adjazenzmatrix \downarrow

4 Optimierung von Graphenproblemen

> CC-BY-SA A. Popp

4.1 Kurzeinführui in die Graphentheorie

4.2 Abbilden von Graphen in OPL

4.3 OPL: Selbstdefinierte Tupel als Datenstruktur

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, 0],
      [0, 0, 0, 0, 0, 0],
      [0, 0, 0, 0, 0, 0],
      [0, 0, 0, 0, 0, 0],
];
```

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

↓ OPL ↓

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

4 Optimierung von Graphenproblemen

CC-BY-SA A. Popp

in die
Graphentheorie

4.2 Abbilden von Graphen in OPL

Selbstdefinierte
Tupel als
Datenstruktur

4.2 Abbilden von Graphen in OPL

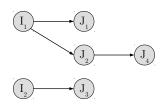
4.3 OPL: Selbstdefinierte Fupel als Datenstruktur

1.4 OPL: Operatoren mit Podingungen

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



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

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

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

 \downarrow OPL \downarrow

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

4 Optimierung von Graphenproblemen

CC-BY-SA A. Popp

in die
Graphentheorie

4.2 Abbilden von Graphen in OPL

4.3 OPL: Selbstdefinierte Tupel als Datenstruktur

peratoren mit edingungen

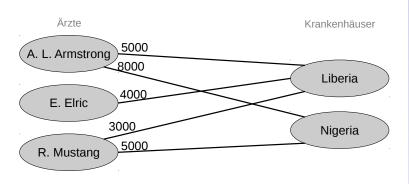
4.2 Abbilden von Graphen in OPL

4.3 OPL: Selbstdefinierte Tupel als Datenstruktur

4.4 OPL: Operatoren mit Bedingungen

4.3 OPL: Selbstdefinierte Tupel als Datenstruktur

Beispiel: Relieve Ärzte



4 Optimierung von Graphenproblemen

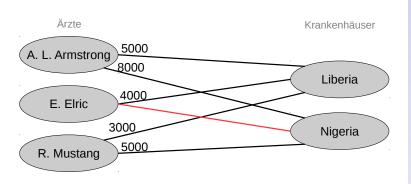
> CC-BY-SA A. Popp

4.1 Kurzeinführun in die Graphentheorie

4.2 Abbilden von Graphen in OPL

4.3 OPL: Selbstdefinierte Tupel als Datenstruktur

Beispiel: Relieve Ärzte



4 Optimierung von Graphenproblemen

> CC-BY-SA A. Popp

4.1 Kurzeinführung in die Graphentheorie

4.2 Abbilden von Graphen in OPL

4.3 OPL: Selbstdefinierte Tupel als Datenstruktur

A. Popp

4.4 OPL: Operatoren mit Bedingungen

Indexmengen:

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:

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

CC-BY-SA A. Popp

4.1 Kurzeinführung in die Granhentheorie

4.2 Abbilden von Graphen in OPL

4.3 OPL: Selbstdefinierte Tupel als Datenstruktur

- 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]] → Fehlermeldung: "Die Größe des Variablenindexers ist für einen generischen Array nicht zulässig."

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;
};
```

4 Optimierung von Graphenproblemen

CC-BY-SA A. Popp

4.1 Kurzeinführung in die

4.2 Abbilden von Graphen in OPL

4.3 OPL: Selbstdefinierte Tupel als Datenstruktur

4.3 OPL: Selbstdefinierte Tupel als Datenstruktur

4.4 OPL: Operatoren mit Bedingungen

Reihenfolge nach in spitzen Klammern zugeordnet.

Beispiel: Definition einer Kante als Literal

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

In Literalen eines Tupel-Datentyps werden die Elemente der

Beispiel: Auslesen des Startknotens einer Kante

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

4.3 OPL: Selbstdefinierte Tupel als Datenstruktur

4.4 OPL: Operatoren mit Bedingungen

Knoten und Kanten seien wie oben definiert.

Anwendungsbeispiel

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

```
forall(t in T)

sum(\langle r,t \rangle) in E)(x[\langle r,t \rangle]) == 1;
```

4.4 OPL: Operatoren mit Bedingungen

4 Optimierung von Graphenproblemen

CC-BY-SA A. Popp

in die
Graphentheorie

4.2 Abbilden von Graphen in OPL

4.3 OPL: Selbstdefinierte Tupel als Datenstruktur

4.4 OPL:

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!)

.3 OPL: elbstdefinierte upel als

4.4 OPL:

Operatoren mit Bedingungen

Literale für Wahrheitswerte

true, false

Vergleichsoperatoren für Wahrheitswerte

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

Logische Verknüpfungen für Wahrheitswerte

math. Schreibweise	_	\wedge	V	$\underline{\vee}$
OPL-Syntax	!	&&	11	!=

6.3 OPL: Selbstdefinierte Tupel als

4.4 OPL: Operatoren mit Bedingungen

Knoten und Kanten seien wie oben definiert.

Anwendungsbeispiel

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

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