Job-Shop



Bernd Härtenstein WS2010/11

Inhalt

[1 Aufgabe 1](#_Toc283167900)

[2 LP-Modell 1](#_Toc283167901)

[3 Das Programm 3](#_Toc283167902)

[4 Mögliche Erweiterungen 4](#_Toc283167903)

# Aufgabe

Die Aufgabe des Programmes besteht darin, eine vorgegebene Zahl von Produkten hinsichtlich einer optimalen, kürzesten Gesamtdurchlaufzeit auf verschiedene Maschinen einzuplanen.

Beschränkungen

Für das Programm gibt es die folgenden Beschränkungen:

* maximal 10 Produkte/Maschinen und
* maximale Bearbeitungsdauer von 99 Zeiteinheiten eines Produkts auf einer Maschine.

# LP-Modell

Um das LP-Modell das dem Job-Shop-Programm zugrunde liegt zu erklären, wird ein Beispiel mit zwei Maschinen, drei Produkten und den in der Tabelle ‑ dargestellten Bearbeitungsdauern verwendet.

2‑2: Bearbeitungsdauern

|  |  |  |
| --- | --- | --- |
|  | Maschine 1 | Maschine 2 |
| Produkt 1 | 3 | 2 |
| Produkt 2 | 1 | 2 |
| Produkt 3 | 2 | 1 |

**Anzahl Variablen**: Anz. Maschinen \* Anz. Produkte \* t *(Summe Bearbeitungsdauern)* \* 2

**→** 2 \* 3 \* 11 \*2 = 132 Variablen

**Zielfunktion**: 5 \* XM1\_P1\_t1 + 10 \* XM1\_P1\_t2 + … + 60 \* XM1\_P1\_t11 + … + 60 \* XM1\_P3\_t11 + … + 60 \* XM2\_P3\_t11 – XBelohnung1 - … - XBelohnung66 **→** MIN!

Die Restriktionen 1 – 6 stellen sicher, dass die Bearbeitungsdauern der Produkte auf den Maschinen von den Zeitpunkten t1 bis t11 denen in der Tabelle entsprechen.

**Restriktion 1**: XM1\_P1\_t1 + XM1\_P1\_t2 + … + XM1\_P1\_t11 = 3

**Restriktion 2**: XM1\_P2\_t1 + XM1\_P2\_t2 + … + XM1\_P2\_t11 = 2

**…**

**Restriktion 6**: XM2\_P3\_t1 + XM2\_P3\_t2 + … + XM2\_P3\_t11 = 1

Eine Maschine kann nur ein Produkt pro Zeitpunkt bearbeiten.

**Restriktion 7**: XM1\_P1\_t1 + XM1\_P2\_t1 + XM1\_P3\_t1 <= 1

**Restriktion 8**: XM1\_P1\_t2 + XM1\_P2\_t2 + XM1\_P3\_t2 <= 1

**…**

**Restriktion 28**: XM2\_P1\_t11 + XM2\_P2\_t11 + XM2\_P3\_t11 <= 1

Ein Produkt kann nur auf einer Maschine pro Zeitpunkt bearbeitet werden.

**Restriktion 29**: XM1\_P1\_t1 + XM2\_P1\_t1 <= 1

**Restriktion 30**: XM1\_P1\_t2 + XM2\_P1\_t2 <= 1

**…**

**Restriktion 61**: XM1\_P3\_t11 + XM2\_P3\_t11 <= 1

Da es sinnvoll wäre, dass ein Produkt auf einer Maschine möglichst am Stück bearbeitet wird (Rüstzeiten), müssen noch ein paar Restriktionen eingeführt werden die den Solver dazu bringen es so zu machen. Diese Restriktionen sind auch der Grund weshalb die Anzahl der Variablen nicht nur: Anz. Maschinen \* Anz. Produkte \* t *(Summe Bearbeitungsdauern)* ist, sondern das Selbe multipliziert mit Zwei.

**Restriktion 62**: XM1\_P1\_t1 + XM1\_P1\_t2 – 2 \* XBelohnung1 >= 0

**Restriktion 63**: XM1\_P1\_t2 + XM1\_P1\_t3 – 2 \* XBelohnung2 >= 0

**…**

**Restriktion 73**: XM1\_P1\_t11 – 2 \* XBelohnung11 >= 0

**Restriktion 74**: XM1\_P2\_t1 + XM1\_P2\_t2 – 2 \* XBelohnung12 >= 0

**…**

**Restriktion 85**: XM1\_P2\_t11 – 2 \* XBelohnung22 >= 0

**…**

**Restriktion 127**: XM2\_P3\_t11 – 2 \* XBelohnung66 >= 0

# Das Programm

Das JobShop-Javaprogramm wurde in der Entwicklungsumgebung „Netbeans IDE 6.8“ programmiert. Als Solver wurde lp\_solve 5.5 verwendet.

In Abbildung sind die verwendeten Klassen abgebildet.

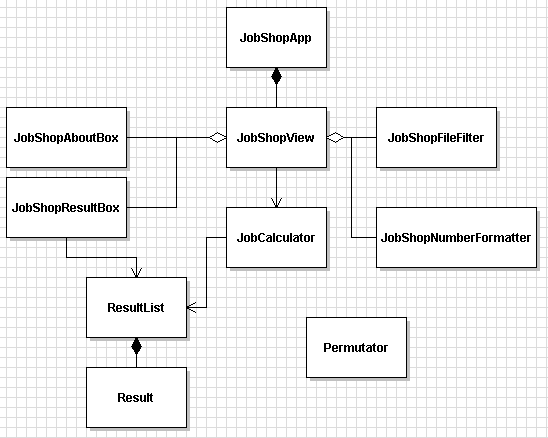


Abbildung 2: Klassendiagramm

Programmablauf

Im Hauptfenster des JobShops (JobShopView) können die Bearbeitungsdauern der Produkte für jede Maschine eingetragen werden. Dabei kontrolliert der JobShopNumberFormatter, dass es sich bei den Einträgen um Zahlen zwischen 0 und 99 handelt.

Beim Klick auf den Berechnen-Button wird der JobCalculator aufgerufen. Der JobCalculator erstellt dann das LP-Modell und berechnet ein optimales Ergebnis mit dem LP-Solve. Dieses Ergebnis speichert er dann in der ResultList ab, in welcher alle Punkte auf welcher Maschine welches Produkt an welchem Zeitpunkt gebaut wird (Result), gespeichert werden. Die ResultList wird dann an die JobShopResultbox übergeben, welche dann das Ergebnis anzeigt.

Der JobShopFileFilter kontrolliert, dass nur (Text-)Dateien gespeichert/geladen werden, deren Bezeichnung mit „.job“ endet.

Die JobShopAboutBox zeigt ein Fenster an in welchem Informationen wie Autor und Erstellungsdatum angegeben sind.

Die Permutator-Klasse ist ein erster Entwurf um das Job-Shop-Problem so zu lösen, wie in der alten Job‑Shop‑Version.

# Mögliche Erweiterungen

Es sind noch ein paar Erweiterungen für den JobShop denkbar:

* Schönere, grafische Darstellung des Ergebnisses.
* Berechnen eines Ergebnisses ohne Solver, wie in der alten JobShop-Version.
* Beheben eines Bugs. Wenn eine Datei geladen wird und ohne eine Änderung vorgenommen zu haben noch einmal abgespeichert wird, dann wird der Eintrag im ersten Feld (Maschine1/Produkt1) mit 0 gespeichert. Der Grund ist bis jetzt noch unbekannt.