

Optimierung und Entscheidungsunterstützung (OuE)

Anna Prenzel

15.03.2013

Hochschule Zittau/Görlitz

Themen für heute:

- 1 Einführung
- 2 Fallstudie
- 3 Lösung mit KI
- 4 Constraint Satisfaction Probleme

Kurze Frage:

Was macht Software alles möglich?

Arten von Software

Software ermöglicht

- Produktivität (Büroarbeit, Mediengestaltung)
- Kommunikation (Soziale Netzwerke, Chat)
- Transaktionen, Geschäftsabläufe (Banküberweisung, Personalverwaltung)
- Unterhaltung (Filme, Spiele)

... das war alles!?

SW für Planung und Organisation



Noch ein Video?



SW für Planung und Organisation

Software, die dabei hilft, wichtige *Entscheidungen* zu treffen, z.B.

- beim Bau eines Fußballstadions
- bei der Kalkulation von Ticketpreisen
- für die Produktions- oder Transportplanung
- ...

SW für Planung und Organisation

Fragen, die mit Hilfe dieser Software beantwortet werden:

- Zu welchem Zeitpunkt sollte welches Produkt zu welchem Preis angeboten werden?
- Was ist die günstigste Reihenfolge zur Bearbeitung von Produktionsaufträgen?
- Wie kann ich meine Fahrzeuge optimal beladen?

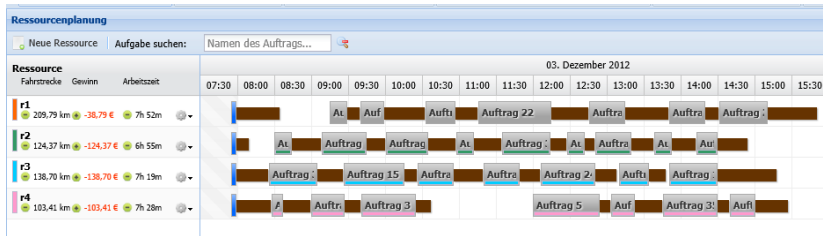
... um den Gewinn zu maximieren und den Aufwand zu senken

SW für Planung und Organisation

Bedingungen, unter denen Entscheidungen getroffen werden:

- Es gibt einen sehr großen Handlungsspielraum.
- Vor- und Nachteile einer Entscheidung müssen abgewogen werden.
- Viele Rahmenbedingungen und Vorschriften müssen beachtet werden.

Eine Flottenplanungssoftware



Ziel:

- Fahrtkosten minimieren
- Touren zusammenstellen, die Gewinn bringen
- alle Termine einhalten

Beispiel

N Aufträge i mit Einnahmen $e_i \in \mathbb{R}$, $i \in \{1, \dots, N\}$ gehören zu einer Tour. Es gilt:

$$\text{Gewinn} = \sum_{i \in \{1, \dots, N\}} e_i - \text{Fahrtkosten} \quad (1)$$

- Das Einfügen eines neuen Auftrages in die Tour kann zu einem reduzierten Gewinn führen.
- Dies wird aber u.U. durch hohe Einnahmen ausgeglichen.
- Eine nachträgliche Optimierung der Fahrzeit kann ebenfalls den Gewinn steigern.

testadmin5, j7, j5

Beispiel

Das Ganze wird noch erschwert durch Randbedingungen:

- Zeitfenster für Aufgaben
- Vorgeschriebene Fahrer für bestimmte Aufgaben (Qualifikationen)
- Arbeitszeiten der Fahrer

KI?

Wir hatten doch
KI, ich kann das
alles
programmieren!

OK, ich wollte
eigentlich nicht
Mathematik
studieren!

Gut, dann
programmieren
wir das jetzt mal!



Backtracking zur Erzeugung einer Tour

... unter Berücksichtigung von Zeitfenstern

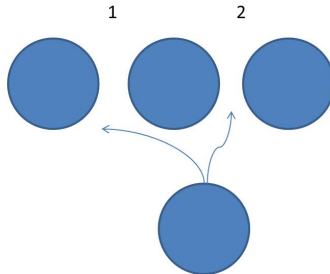
Job
-int EarliestStart
-int LatestStart
-int Location-ID

DistanceMatrix
-int[,] distances

Backtracking zur Erzeugung einer Tour

```
bool backtrack(List<Job> partial, List<Job> pending){  
  
    if(pending.size() == 0){ return true;}  
    Job insert = pending.removeAt(0);  
  
    for(int i=1; i < partial.Count; i++){  
  
        partial.insert(job,i);  
  
        if(isValid(partial)){  
  
            if(backtrack(partial, pending)){  
                return true;  
            }  
            else  
                partial.remove(job, i);  
        }  
        else  
            partial.remove(job,i);  
    }  
    pending.insert(job,0); return false;}  
}
```

Backtracking zur Erzeugung einer Tour



Backtracking zur Erzeugung einer Tour

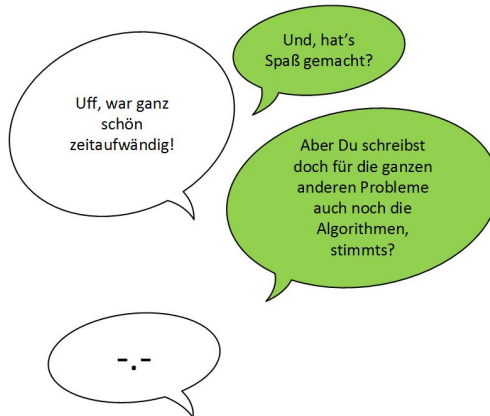
Was macht *IsValid(List partial)*?

- überprüft Zeitfensterüberlappung
- beachtet dabei auch Entfernungen in der Entfernungsmatrix

Optimierung der Startlösug?

Dafür brauchen wir eine *Bewertungsfunktion*, z.B. `double
QualityTraveltime(List solution).`
Und dann?

Backtracking zur Erzeugung einer Tour



Vision

- Wir bauen einen Universal-Solver, der viele verschiedene Entscheidungsprobleme lösen kann.
- Wir erklären dann dem Computer unser Problem, und er löst es mit Hilfe dieses Solvers.
- Dies geschieht *deklarativ*: wir beschreiben, welche Eigenschaften die Lösung mindestens erfüllen muss.

Feststellung

Wir müssen unser Problem in einer Sprache formulieren, die der Computer versteht.

Beschreibung von Entscheidungsproblemen

CSP

Ein Constraint Satisfaction Problem (V, D, C) besteht aus einer Menge an Variablen V , ihren Domänen D und einer Menge von Randbedingungen (Constraints) C über V .

CSP: Variable

- Eine Variable ist ein Platzhalter für einen Wert.
- Jede Entscheidungsvariable x besitzt eine Domäne D_x . Sie enthält alle Werte, die diese Variable annehmen kann.

CSP: Constraint

- Constraints haben eine bestimmte Reichweite (scope)
 $s_c = \langle x_1, x_2, x_n \rangle$, wobei x_i eine Variable ist.
- Ein Constraint C ist eine Relation auf den Domänen der Variablen innerhalb der Reichweite des Constraints
$$C \subseteq D_{x_1} \times D_{x_2} \times \dots \times D_{x_n}$$

CSP: Lösung

- Eine *Belegung* oder *Instanziierung* ist eine vollständige Wertebelegung der Variablen (die nicht unbedingt alle Constraints erfüllt).
- Eine *Lösung* ist eine Belegung, sodass alle Constraints erfüllt werden.
- Eine *Teillösung* ist eine Belegung einer Teilmenge von Variablen, sodass alle Constraints erfüllt werden.

CSP - CSOP

- Ziel beim CSP: eine oder alle Lösungen zu finden.

CSP + Zielfunktion = CSOP

CSOP

Ein Constraint Optimization Problem erweitert ein CSP um eine Zielfunktion. Es soll nicht nur eine Lösung, sondern die beste Lösung in Bezug auf den Zielfunktionswert gefunden werden.

Beispiel CSP

$$x_1 \in \{A, B, C, D\}$$

$$x_2 \in \{B, C\}$$

$$x_3 \in \{C, D\}$$

$$x_4 \in \{A, B, C\}$$

Die Constraints lauten:

$$x_i \neq x_j, \text{ für } i \neq j, i \in \{1, \dots, n\}, j \in \{1, \dots, n\}$$

Eine Lösung ist somit z.B. $\{A, B, D, C\}$

Fallstudie Flottenplanung

Formulierung als CSOP

... genauer gesagt als MIP, d.h.

- alle Randbedingungen als lineare Gleichungen formulieren
- ein Teil der Variablen ist ganzzahlig oder binär 0-1

Fallstudie Flottenplanung

Es gibt eine Menge von Kunden $K = \{1, \dots, N\}$, die sich an verschiedenen Standorten befinden.

Entscheidungsvariablen:

- Variable $x_{ij} \in \{0, 1\}, i \in \{1, \dots, N\}, j \in \{1, \dots, N\}$ gibt an, ob der Fahrer direkt von Kunde i zu Kunde j fährt.
- Variable $s_i \in \mathbb{R}, i \in \{1, \dots, N\}$ gibt die Ankunftszeit bei Kunde i an.

Fallstudie Flottenplanung

Eingabevariablen:

- Der frühest- und spätestmögliche Besuchstermin bei Kunde i (Zeitfenster) wird an die Variablen $e_i \in \mathbb{N}$ und $l_i \in \mathbb{N}, i \in \{1, \dots, N\}$ übergeben.
- Variable $d_i \in \mathbb{N}, i \in \{1, \dots, N\}$ enthält die Dauer des Besuchs bei Kunde i .
- Die Entfernungen zwischen den Standorten liegen als Entfernungsmatrix $C = c_{ij} \in \mathbb{R}, \forall i, j \in \{1, \dots, N\}$ vor.

Fallstudie Flottenplanung

Randbedingungen:

- Es gilt $\sum_{j \in \{1, \dots, N\}} x_{ij} = 1, i \in \{1, \dots, N\}$, da jeder Kunde genau einmal besucht wird.
- Die Besuchstermine dürfen einander nicht überlappen, sodass gilt:
$$x_{ij}(s_i + d_i + c_{ij} - s_j) \leq 0, i \in \{1, \dots, N\}, j \in \{1, \dots, N\}$$
- Die Besuchstermine müssen innerhalb der Zeitfenster liegen:
$$e_i \leq s_i \leq l_i, i \in \{1, \dots, N\}$$

Zielfunktion: Minimiere die Gesamtfahrstrecke: min

$$\sum_{i \in \{1, \dots, N\}} \sum_{j \in \{1, \dots, N\}} c_{ij} x_{ij}$$

Modellierung vs. Lösung

Mit dieser Sprache werden wir noch viele andere Probleme beschreiben!

- Ihr werdet *Solver* kennen lernen, die verschiedene Probleme lösen können, die in dieser Weise formuliert sind.
- Die *Modellierung* ist dabei von der *Lösung* vollständig getrennt.
- Auch die *Daten* sind vom Modell getrennt.

Modellierung vs. Lösung

Daten

Modell

Solver

Beispiel für einen CSP-Solver

```
back_solve(partial, variables){  
    if leer(variables) return true  
  
    entferne  $V_{\{i\}}$  aus variables  
  
    for jeden Wert in  $D_{\{i\}}$   
        erweitere partial zu partial_neu  
  
        if valid(partial_neu)  
            if back_solve(partial_neu, variables)  
                return true  
  
    fuege  $V_{\{i\}}$  in variables ein  
  
    return false  
}
```

Aber:

- Nicht jeder Solver ist gleich gut geeignet → wir werden lernen warum.
- In manchen Fällen kommt man doch besser, wenn man selbst programmiert.
- Dies muss man von Fall zu Fall abschätzen.

Aufgaben eines Entwicklers

- CSP/CSOP erkennen und formulieren
- Geeignete Solver auswählen
- Solver konfigurieren und ggf. für das spezielle Problem „tunen“
- Tests durchführen und verschiedene Solver(-konfigurationen) statistisch vergleichen

Das war's für heute :-)