

Grundlagen der Künstlichen Intelligenz

28. Constraint-Satisfaction-Probleme: Zerlegungsmethoden

Malte Helmert

Universität Basel

24. April 2015

Constraint-Satisfaction-Probleme: Überblick

Kapitelüberblick Constraint-Satisfaction-Probleme:

- 22.–23. Einführung
- 24.–26. Kernalgorithmen
- 27.–28. Problemstruktur
 - 27. Constraint-Graphen
 - 28. Zerlegungsmethoden

Zerlegungsmethoden

Kompliziertere Graphen

Was, wenn der Constraint-Graph kein Baum ist und nicht in Komponenten zerfällt?

- Idee 1: **Konditionierung**
- Idee 2: **Baumzerlegung**

Konditionierung

Konditionierung

Konditionierung

Idee: Führe Backtracking mit Forward Checking durch, bis der Constraint-Graph **eingeschränkt auf die restlichen Variablen** in Komponenten zerfällt bzw. ein Baum ist.

Restproblem $\rightsquigarrow$ Algorithmus für einfache Constraint-Graphen

Konditionierung

Konditionierung

Idee: Führe Backtracking mit Forward Checking durch, bis der Constraint-Graph **eingeschränkt auf die restlichen Variablen** in Komponenten zerfällt bzw. ein Baum ist.

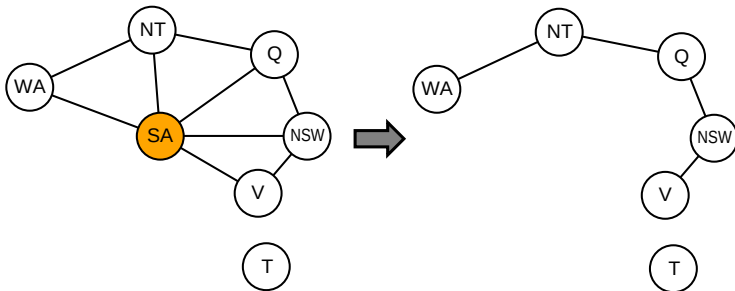
Restproblem $\rightsquigarrow$ Algorithmus für einfache Constraint-Graphen

Cutset conditioning: wähle Variablenordnung so, dass vordere Variablen möglichst kleines **Cutset** bilden (Variablenmenge, ohne die der Constraint-Graph zyklfrei ist).

Zeitaufwand: n Variablen, davon $m < n$ im Cutset, Wertebereiche der Größe k : $O(k^m \cdot (n - m)k^2)$
(Optimale Cutsets zu finden ist seinerseits NP-vollständig.)

Konditionierung: Beispiel

Australien-Beispiel: Cutset der Grösse 1 reicht aus:



Baumzerlegung

Baumzerlegung

Grundidee der **Baumzerlegung**:

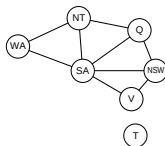
- Zerlege Constraint-Netz in kleinere **Teilprobleme** (überlappend).
- Finde Lösungen für die Teilprobleme.
- Setze Teillösungen zu Gesamtlösung zusammen.

Weitere Details:

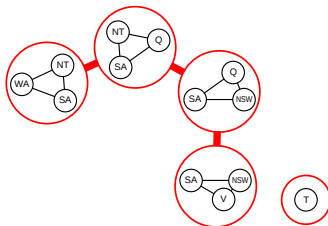
- „Zusammensetzungsproblem“ für Teillösungen ist auch ein Constraint-Netz (**Meta-Constraint-Netz**).
- Teilprobleme so wählen, dass das Meta-Constraint-Netz einen **Baum/Wald** als Constraint-Graph hat.
 ↪ Zusammensetzen mit effizientem Algorithmus für Bäume

Baumzerlegung: Beispiel

Constraint-Netz:



Baumzerlegung:



Baumzerlegung: Definition

Definition (Baumzerlegung)

Betrachte ein Constraint-Netz $\mathcal{C}$ mit Variablen V .

Eine **Baumzerlegung** von $\mathcal{C}$ ist ein Graph $\mathcal{T}$ mit folgenden Eigenschaften.

Bedingungen an die Knoten:

- Jeder **Knoten** von $\mathcal{T}$ entspricht einer Teilmenge der Variablen von $\mathcal{C}$. Wir nennen so einen Knoten und die zugehörige Variablenmenge ein **Teilproblem** von $\mathcal{C}$.
- Jede **Variable** in V tritt in **mindestens einem** Teilproblem von $\mathcal{T}$ auf.
- Für jeden **nichttrivialen Constraint** R_{uv} von $\mathcal{C}$ treten u und v gemeinsam in **mindestens einem** Teilproblem von $\mathcal{T}$ auf.

...

Baumzerlegung: Definition

Definition (Baumzerlegung)

Betrachte ein Constraint-Netz $\mathcal{C}$ mit Variablen V .

Eine **Baumzerlegung** von $\mathcal{C}$ ist ein Graph $\mathcal{T}$ mit folgenden Eigenschaften.

...

Bedingungen an die Kanten:

- Für jede Variable $v \in V$ sei $\mathcal{T}_v$ die Menge der Knoten, die Teilproblemen entsprechen, zu denen v gehört.
- Für jede Variable v muss $\mathcal{T}_v$ **zusammenhängend** sein, d.h. von jedem Knoten in $\mathcal{T}_v$ ist jeder Knoten in $\mathcal{T}_v$ erreichbar, ohne Knoten ausserhalb von $\mathcal{T}_v$ zu besuchen.
- $\mathcal{T}$ ist **zykelfrei** (ein Wald)

Gute Baumzerlegungen

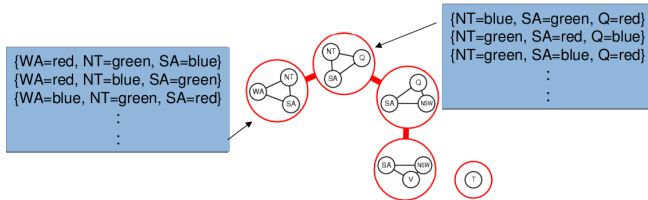
Ziel: jedes Teilproblem hat möglichst wenige Variablen

- entscheidend: Teilproblem mit den meisten Variablen
- dessen Variablenzahl minus 1 heisst **Weite** der Zerlegung
- beste Weite aller Zerlegungen: **Baumweite** des Constraint-Graphen (NP-vollständig zu berechnen)

CSP-Lösung mit Baumzerlegung: Algorithmus

Algorithmus:

- Finde **alle Lösungen** für **alle Teilprobleme** und baue daraus ein baumartiges **Meta-Constraint-Netz**.
- Constraints darin: Teillösungen müssen **kompatibel** sein, d. h. gemeinsame Variablen identisch belegen.
- Löse Meta-Netz mit Verfahren für baumartige Netze.



Zeitaufwand: $O(nk^{w+1})$ bei Weite w der Zerlegung
(Setzt spezialisierte Version von revise voraus; sonst $O(nk^{2w+2})$.)

Zusammenfassung

Zusammenfassung: Kapitel

- Führe **komplizierte** Constraint-Graphen auf **einfache** zurück.
- **Cutset-Konditionierung**:
 - Wähle **möglichst wenige** Variablen (Cutset), so dass nach Belegung dieser Variablen **Restproblem** strukturell einfach ist.
 - **Suche** über Belegung der Variablen im Cutset
- **Baumzerlegung**: bilde **baumartiges** Meta-Constraint-Netz
 - Meta-Variablen: **Gruppen** von Ursprungsvariablen, die gemeinsam alle Variablen und Constraints abdecken
 - **Werte** entsprechen konsistenten Belegungen der Gruppe
 - Constraint zwischen **überlappenden** Gruppen, um **Kompatibilität** zu sichern
 - Gesamtalgorithmus exponentiell in **Weite** der Zerlegung (Grösse der grössten Gruppe)

Zusammenfassung: CSPs

Constraint-Satisfaction-Problem (CSP)

allgemeiner Formalismus für Probleme, bei denen

- Variablen so belegt werden müssen, dass
 - bestimmte Bedingungen (Constraints) erfüllt sind.
-
- Algorithmen: **Backtracking-Suche + Inferenz**
(z. B. Forward Checking, Kantenkonsistenz, Pfadkonsistenz)
 - Variablen- und Wertreihenfolge wichtig
 - mehr Effizienz: Ausnutzen der **Struktur des Constraint-Graphen** (unabhängige Komponenten; Bäume)

Weiterführende Themen

weitere, hier nicht behandelte Aspekte:

- **Backjumping:** Backtracking über mehrere Ebenen auf einmal
- **No-Good Learning:** Inferieren zusätzlicher Constraints anhand von Information, die beim Backtracking gesammelt wird
- **lokale Suchverfahren** im Raum der totalen, aber nicht notwendigerweise konsistenten Belegungen
- **handhabbare Constraint-Klassen:** Identifikation von Constraint-Typen, die polynomielle Algorithmen erlauben
- Verallgemeinerung mit Qualitätsfunktion für Lösungen:
Constraint-Optimierungs-Probleme (COP)

⇒ mehr als genug Stoff für einsemestrige Vorlesung