Constraint Satisfaction

Jochen Renz

Literaturempfehlung:

Kondrak, van Beek: A Theoretical Evaluation of Selected Backtracking Algorithms, in:

Artificial Intelligence, 89:365-387 1997

(`http://www.lpaig.uwaterloo.ca/~vanbeek/papers.html`)

Constraint Satisfaction Probleme (CSP)

- Constraints: Einschränkungen, Bedingungen
  $\rightsquigarrow$ Constraint Satisfaction: Erfüllung von Einschränkungen

- CSPs sind eine **einheitliche Darstellungsweise** von Problemen bei denen es verschiedene Möglichkeiten zur Auswahl gibt, die von Constraints eingeschränkt werden.

$\rightsquigarrow$ Dies ermöglicht Entwicklung von **allgemeinen, intelligenten Lösungsverfahren** für CSPs

$\rightsquigarrow$ dadurch lassen sich sehr viele verschiedene Problemstellungen auf einheitliche Weise lösen.

$\rightsquigarrow$ eine der erfolgreichsten Methoden der Artificial Intelligence mit unzähligen Anwendungen in verschiedenen Gebieten

Alltägliche Beispiele für CSPs

1. **Menüauswahl in einer Pizzeria**:
   jeweils verschiedene Möglichkeiten: Antipasti, Zuppe, Insalate, Pizze, Primi, Secondi, Dolci, ...
   Constraints: insgesamt nicht mehr als 150 ATS; wenn Fisch dann Wein als Getränk; höchstens einen Fischgang; kein Rindfleisch;...

2. **Erstellung eines persönlichen Stundenplans**:
   Verschiedene Vorlesungen zur Auswahl
   Constraints: nicht montags und nicht vor zehn Uhr; unbedingt Konzepte der AI; nicht mehr als 4 Übungsstunden; mindestens eine Theorievorlesung;...

3. **Einrichtung eines WG Zimmers**:
   Plazierung verschiedener Möbelstücke: Bett, Sofa, Schreibtisch, Stereoanlage,...
   Constraints: Bett nicht neben Heizung, Schreibtisch möglichst am Fenster, Stereoanlage gegenüber vom Sofa,...

Formale Definition von CSPs

- Ein CSP Γ ist ein Tupel $\langle \mathcal{V}, (W_1, \ldots, W_m), \mathcal{C} \rangle$:
  1. endliche Menge $\mathcal{V}$ von **Variablen** $x_1, \ldots, x_m$
  2. zugehörige **Wertebereiche** $W_1, \ldots, W_m$, sowie
  3. eine endliche Menge $\mathcal{C}$ von **Constraints** $C(x_i, \ldots, x_j) \subseteq W_i \times \ldots \times W_j$.

- Es gibt Constraints verschiedener Stelligkeit: n-stellige Constraints beschränken die mögliche Belegung von n Variablen, d.h., n-stellige Constraints sind **n-stellige Relationen**.
  - Unäre Constraints $C(x_i)$ beschränken den Wertebereich W_i einer Variablen x_i.
    z.B. $C(x_i) = \{1, 3, 5, 7, 8\}$
  - Binäre Constraints $C(x_i, x_j)$ beschränken den Wertebereich $W_i \times W_j$ eines Paares x_i, x_j.
    z.B. $C(x_i, x_j) : \{(1, 2), (3, 5), (7, 3), (8, 2)\}$
  - Ternäre Constraints,...

Lösung eines CSPs

- **Lösung**: Belegung aller Variablen aus $\mathcal{V}$ mit Werten aus $\mathcal{W}$, so daß alle Constraints aus $\mathcal{C}$ erfüllt sind.

- **Partielle Lösung**: Belegung einer Teilmenge $\mathcal{V}' \subset \mathcal{V}$ von Variablen, die alle Constraints erfüllt, die sich auf Variablen aus $\mathcal{V}'$ beziehen.

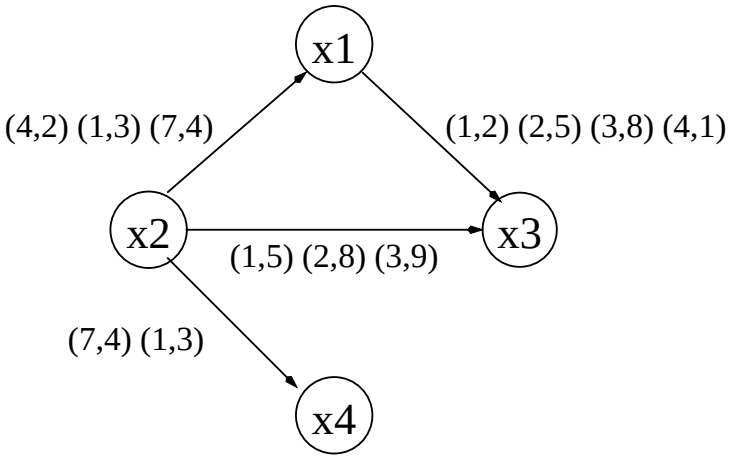
- Hat CSP eine Lösung, so ist das CSP **konsistent** (erfüllbar), andernfalls **inkonsistent** (unerfüllbar)

- Mögliche **Fragestellungen**:
  - Gibt es eine Lösung
  - Finde eine Lösung/alle Lösungen
  - Folgert ein bestimmtes Constraint aus dem CSP
  - ...

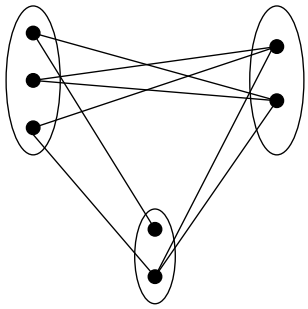
- naive Lösungsmethode (brute force): Sukzessive Aufzählung aller möglicher Belegungen und Prüfung ob alle Constraints erfüllt sind.

Graphische Darstellung von CSPs

- **Constraint Graph** (für binäre Constraints):
  - Variablen als Knoten,
  - Constraints als Kanten des Graphen



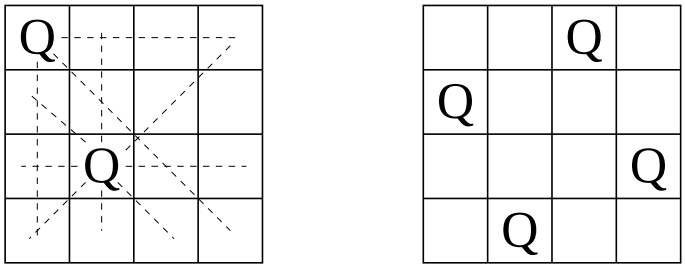
- Erweiterter Constraint Graph (für binäre Constraints): zeigt erlaubte Tupel an



- Hypergraph für nicht-binäre Constraints

Beispiel: n-Damen Problem

- Plaziere n Damen auf $n \times n$ großem Schachbrett, so daß sich die Damen nicht gegenseitig bedrohen.



- **Verschiedene Repräsentationsmöglichkeiten** des Problems, z.B.
  - Eine Variable pro Dame, ein Wert für jedes der Felder: $n^{n \times n}$ verschiedene Belegungen
  - Eine Variable pro Feld, je zwei mögliche Werte (Dame oder keine Dame): $2^{n \times n}$ verschiedene Belegungen
  - (*) Eine Variable pro Spalte, ein Wert für jede Zeile (in jeder Spalte kann nur eine Dame sein): n^n verschiedene Belegungen

- Constraints für (*):
  $C(x_i, x_j) = \{(b_i, b_j) : (b_i \neq b_j) \wedge (|i-j| \neq |b_i - b_j|)\}$

Grundlegendes Lösungsverfahren: Chronologisches Backtracking

- Idee: Systematische Erweiterung partieller Lösungen bis eine Lösung gefunden ist
- Einteilung der Variablen:
  - bereits belegte Variablen (partiellen Lösung)
  - momentan zu belegende Variable
  - noch unbelegte Variablen
- Verfahren: Chronologisches Backtracking (BT)
  Geg.: CSP Γ, partielle Belegung $B = (b_i, \ldots, b_j)$
  1. Prüfe ob B (partielle) Lösung von Γ ist. Falls nicht gebe *FALSE* zurück. (ggf. Lösung ausg.)
  2. Wähle eine noch unbelegte Variable x_k
  3. Für jede mögliche Belegung $b_k \in W_k$ von x_k, rufe BT rekursiv mit $B' = (b_i, \ldots, b_j, b_k)$ auf
  4. Falls alle Aufrufe *FALSE* liefern ist man in Sackgasse und gibt *FALSE* zurück. (Sonst: Γ kons.)

     $\rightsquigarrow$ Backtracking zur vorigen Variable (hier x_j)

Eigenschaften von Backtracking

- Es wird ein **Suchbaum** aufgespannt, der systematisch durchlaufen wird: Jede partielle Belegung ist ein Knoten
- Wird auch Baumsuche genannt
- Prüfung einer partiellen Belegung: Konsistenztest der momentanen Variable mit den bereits belegten Variablen in deren Reihenfolge
- Vorteil gegenüber brute force Methode: Falls eine partielle Belegung bereits inkonsistent ist, wird keine Erweiterung davon mehr untersucht

Intelligentere Varianten

- **Look-back** Verfahren:
  - Welche bereits belegte Variable ist für die Inkonsistenz einer partiellen Belegung verantwortlich?
  - Backtracking kann zu dieser Variable erfolgen

  $\rightsquigarrow$ Backjumping

  $\rightsquigarrow$ Conflict-Directed Backjumping

- **Look-ahead** Verfahren:
  - Welche zukünftigen Variablenbelegungen werden durch bereits belegte Variablen unmöglich?
  - Diese zukünftigen Belegungen müssen nicht mehr vorgenommen werden.

  $\rightsquigarrow$ Forward-Checking

  $\rightsquigarrow$ Lokale Konsistenz

- Hybride Verfahren: Kombination von look-back and look-ahead

Backjumping (BJ)

- Läuft die Suche bei der momentan belegten Variable x_i in eine Sackgasse, so springt BJ zu der **tiefsten Variable** x_h, die mit x_i **verglichen** wurde:
  - Wird Belegung von x_h verändert, so ist eine konsistente Belegung von x_i wieder möglich
  - Ändern der Belegung einer Variablen zwischen x_h und x_i ist garantiert erfolglos, da der Grund für die Sackgasse nicht beseitigt ist.

- 6-Damen Beispiel (siehe extra Folie):
  - Graue Felder im Schachbrett kollidieren mit bereits gesetzten Damen. Nummer im Feld gibt die erste Dame an, die mit dem Feld kollidiert.
  - Erweiterung von 25364 gibt Konflikt mit x_6
  - x_4 ist der tiefste Knoten, der mit x_6 verglichen.
  - backjump zu x_4 umgeht die schattierten Knoten.

  $\rightsquigarrow$ es wird erkannt, daß 2536 inkonsistent ist mit Variable x_6.

Conflict-Directed Backjumping (CBJ)

- Jede Variable x_i hat eigene **Konfliktmenge** $K(x_i)$, die alle bereits belegten Variablen enthält, die **inkonsistent** mit aktueller Belegung von x_i sind.

- Ist Belegung b_i von x_i inkonsistent mit Belegung b_k einer früheren Variable x_k, so wird x_k zu $K(x_i)$ hinzugenommen.

- Läuft x_i in Sackgasse, gibt es **backjump zur tiefsten Variable** x_h die in $K(x_i)$ enthalten ist. $K(x_i)$ (außer x_h) wird zu $K(x_h)$ hinzugenommen.

- 6-Damen Beispiel (siehe extra Folie):
  - Sackgasse bei Erweiterung von Knoten 25314, $K(x_6) = \{x_1, x_2, x_3, x_5\}$.
  - Backjumping zu x_5. $K(x_5) = \{x_1, x_2, x_3\}$.
  - x_5 ebenfalls Sackgasse, daher backjump zu x_3.

  $\rightsquigarrow$ es wird erkannt, daß 253 inkonsistent ist mit dem Variablenpaar x_5, x_6.

Forward Checking (FC)

- Nach Belegung der momentanen Variable werden **Wertebereiche der zukünftigen Variablen gefiltert** und alle inkonsistenten Werte entfernt.
- Falls Sackgasse auftritt, werden Effekte des forward checking der momentanen Belegung rückgängig gemacht und eine neue Belegung wird gewählt.
- Falls alle Belegungen der momentanen Variable ausprobiert wurden, wird chronologisches backtracking zur zuletzt belegten Variablen gemacht.
- 6-Damen Beispiel (siehe extra Folie):
  - Nach Belegung von x_1, x_2, x_3 werden alle grauen Felder aus den jeweiligen Wertebereichen der zukünftigen Variablen entfernt
  - Vorteil: Es werden nur partiell konsistente Knoten besucht (2531, 25314 und 2536)
  - Danach wird mit Knoten 26 weitergemacht

Lokale Konsistenz

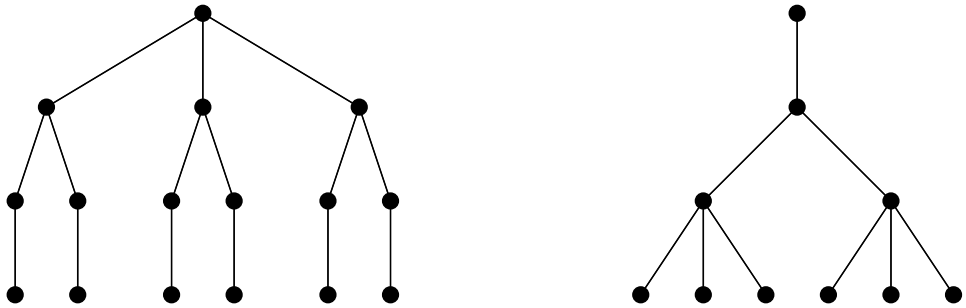
- Erweiterte Form des Forward Checking
- **2-Konsistenz** (Kantenkonsistenz): Ein Paar (x_i, x_j) ist 2-konsistent, falls es für jedes $b_i \in W_i$ ein $b_j \in W_j$ gibt, so daß $C(x_i, x_j)$ erfüllt ist.
  - (x_i, x_j) wird wie folgt 2-konsistent gemacht: $W_i := W_i \cap \{b_i : \exists b_j \in W_j \wedge (b_i, b_j) \in C(x_i, x_j)\}$
  - Ein CSP Γ ist 2-konsistent, falls alle Paare (x_i, x_j) 2-konsistent sind
- Erweiterung auf **k-Konsistenz**: Für jede partielle Lösung von $k-1$ Variablen gibt es eine konsistente Belegung der k-ten Variablen

  $\rightsquigarrow$ Ein CSP kann in $O(n^k)$ Zeit k-konsistent gemacht werden (n: Anzahl der Variablen)
- 6-Damen Beispiel (siehe extra Folie):
  - 253 inkonsistent, weil (x_5, x_6) nicht 2-konsistent

Zusätzliche Verwendung von Heuristiken

- Reihenfolge in der Variablen ausgewählt werden:
  - Zuerst die Variable mit dem kleinsten verbleibenden Wertebereich (fail first)

    $\leadsto$ Verkleinerung des Suchbaums:



  - Dynamische vs. statische Variablenordnung

- Reihenfolge in der eine Belegung gewählt wird:
  - Zuerst die Belegung die den restlichen Suchraum am wenigsten einschränkt

    $\leadsto$ erhöht die Wahrscheinlichkeit, daß eine Lösung schnell gefunden wird

Vergleich der Verfahren

- Wichtige Kenngrößen:
  - Anzahl besuchter Knoten im Suchbaum
  - Anzahl der Konsistenztests
- Üblicherweise empirische Vergleiche von Verfahren: jedoch teilweise uneinheitliche Ergebnisse, die je nach untersuchtem Problem variieren.
- Theoretische Ergebnisse (selbe Variablenordnung):
  - besuchte Knoten: BT $>$ BJ $>$ CBJ/FC
  - Konsistenztests: BT $>$ BJ $>$ CBJ (FC unvergleichbar)
- aber: Gegeben ein CSP Γ und eine Variablenordnung für CBJ. Es gibt eine Variablenordnung für BT, so daß BT nie mehr Knoten besucht als CBJ um Γ zu lösen [Chen, van Beek, 2001]
- Tip: look-ahead mit verschiedenen Heuristiken

Zusammenfassung

- CSPs ermöglichen einheitliche Lösungsverfahren für verschiedenste Problemstellungen
  - look-back Verfahren
  - look-ahead Verfahren
  - verschiedene Heuristiken
  - Kombination der Verfahren und Heuristiken
- daher sehr viele mögliche und bereits existierende Anwendungen, z.B.:
  - Vision und Bildanalyse
  - Konfiguration, Layout, Design
  - Scheduling
  - Räumliches und zeitliches Schließen
  - Sprachverarbeitung
  - ...
- sehr lebendiges Forschungsgebiet innerhalb der AI