Künstliche Intelligenz

Teil II: Schwache Problemlösemethoden[1]

Robert Jäschke
Gerhard Gossen

FG Wissensbasierte Systeme/Forschungszentrum L3S
Leibniz Universität Hannover

Sommersemester 2015

[1]Dieser Foliensatz basiert auf Material von Mirjam Minor, Humboldt-Universität Berlin, WS 2000/01

Agenda

3. Constraints
   - Constraint Satisfaction Problem (CSP)
   - Anwendungsbeispiele
   - Mögliche Fragestellungen
   - Formale Definitionen
   - Beispiele
   - Binäre Constraint-Netze als Graphen
   - CSP als Suchproblem
   - Backtracking-Suche
   - Heuristiken zur Lösung von CSP-Suchproblemen
   - Komplexität/Unentscheidbarkeit
   - Constraint-Propagierung
   - Kantenkonsistenz-Algorithmus AC-3
   - Bildverstehen als Constraint-Problem
   - Zeitliches Schließen als Constraint-Problem

Constraint Satisfaction Problem (CSP)

Problem bei Suchverfahren: Kombinatorische Explosion

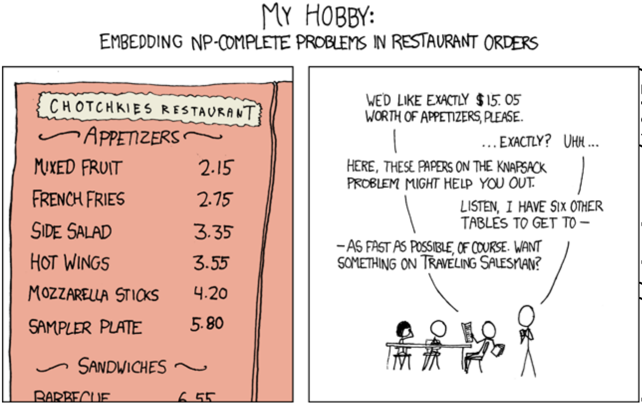
Häufig sind zusätzlich zum eigentlichen Problem noch eine Reihe einschränkender Nebenbedingungen (*Constraints*) gegeben, die ebenfalls zu erfüllen sind.

Wie können Constraints bei der Suche nach einer Lösung helfen?

- durch Testen von Zwischenergebnissen bzgl. der Constraints lassen sich Irrwege frühzeitig erkennen
- durch Verkettung mehrerer Constraints treten manchmal Lösungen hervor (*Constraint Propagation*)

Anwendungsbeispiele

- Stundenplanerstellung
- Auslosung von Gegnern bei Turnieren
- Transportprobleme, Logistik, Produktionsplanung
- Interpretation zweidimensionaler Linienzeichnungen, Bilderkennung
- Algebren zum zeitlichen Schließen



https://xkcd.com/1045/

Mögliche Fragestellungen

- Ist eine Lösung unter den Nebenbedingungen möglich?
- Wie sieht der Lösungsraum aus?
- Falls keine Lösung existiert, kann man durch minimale Abweichung von den Constraints (abschwächen bzw. weglassen) zu einer Lösung kommen?

Formale Definitionen

Gegeben: Variablenmenge $V = \{v_1, \dots v_n\}$ mit den Wertebereichen $dom(v_i)$ für alle v_i $(i = 1, \dots, n)$.
Sei $dom(V) = dom(v_1) \times \cdots \times dom(v_n)$

Definition (k-stelliges Constraint)

Ein *k-stelliges Constraint* C über $V' = \{v'_1, \dots, v'_k\} \subseteq V$ ist eine Teilmenge $C \subseteq dom(v'_1) \times \cdots \times dom(v'_k)$.

- Ein Constraint mit Stelligkeit $k = 1$ heißt *unäres Constraint*
- Ein Constraint mit Stelligkeit $k = 2$ heißt *binäres Constraint*

Formale Definitionen

Ein *Constraint-Netz* $\mathcal{C}$ über V ist eine Menge $\mathcal{C} = \{C_1, \ldots, C_m\}$, wobei jedes C_i ein Constraint über einer Menge $V_i \subseteq V$ ist. Ein *binäres Constraint-Netz* ist ein Constraint-Netz, das nur unäre und binäre Constraints enthält.

Sei $\beta : V \to \bigcup_i dom(v_i)$ eine Belegung, die jedem $v_i \in V$ ein Element aus $dom(v_i)$ zuordnet.

Die Belegung β *erfüllt* ein Constraint C über $V' = \{v'_1, \ldots, v'_k\}$, falls $(\beta(v'_1), \ldots, \beta(v'_k)) \in C$. Dann heißt $(\beta(v'_1), \ldots, \beta(v'_k)) \in C$ *lokale Lösung*.

Die Belegung β erfüllt das Constraint-Netz C und heißt *globale Lösung*, falls β alle $C \in \mathcal{C}$ erfüllt.

Beispiele: (1)

Das folgende Problem soll mit Constraints formuliert werden:
Finde zwei natürliche Zahlen x und y, so dass $x + y = 7$ unter der Voraussetzung $x > y > 2$.

- $V = \{x, y\}$, $dom(x) = dom(y) = \mathbb{N}$
- $C_1 = \{y \in \mathbb{N} \mid y > 2\} \subseteq dom(y)$ über $V_1 = \{y\}$
- $C_2 = \{(x, y) \in \mathbb{N}^2 \mid x > y\} \subseteq dom(x) \times dom(y)$ über $V_2 = \{x, y\}$
- $C_3 = \{(x, y) \in \mathbb{N}^2 \mid x + y = 7\} \subseteq dom(x) \times dom(y)$ über $V_3 = \{x, y\}$
- $\mathcal{C} = \{C_1, C_2, C_3\}$

$x = 2$ und $y = 1$ ist eine lokale Lösung bzgl. C_2
$x = 4$ und $y = 3$ ist eine globale Lösung.

Beispiele: (2)

Das Constraintnetz $\mathcal{C} = \{C_1, C_2, C_3, C_4\}$ sei wie folgt gegeben:

- $V = \{x, y, z\}$, $dom(x) = dom(y) = dom(z) = [0, 1]$
- $C_1 = \{(x, y) \mid x > y\}$ über $V_1 = \{x, y\}$
- $C_2 = \{y \mid y > 0.5\}$ über $V_2 = \{y\}$
- $C_3 = \{(x, z) \mid x + z = 1\}$ über $V_3 = \{x, z\}$
- $C_4 = \{(x, z) \mid x < z\}$ über $V_4 = \{x, z\}$

$(0.5, 0.7, 0.5)$ ist eine lokale Lösung bzgl. C_3

Eine globale Lösung existiert nicht (Aus C_1, C_2, C_4 folgt $z > x > y > 0.5$ im Widerspruch zu C_3: $x + z = 1$). Die gegebenen Voraussetzungen sind inkonsistent. Ohne C_4 wäre $(0.7, 0.6, 0.3)$ eine globale Lösung.

Binäre Constraint-Netze als Graphen

- Knoten entsprechen den Variablen
- Kanten entsprechen den binären Constraints
- unäre Constraints sind als Einschränkung des Wertebereichs der Variablen enthalten

Constraint-Netze höherer Ordung lassen sich in binäre Constraint-Netze überführen.

→ Vereinfachung/Vereinheitlichung der Probleme

→ verschiedene Verfahren (speziell für endliche Wertebereiche) zum Finden einer Lösung

Aber: Komplexität der Umwandlung hoch

CSP als Suchproblem

Ausgangszustand: Die leere Zuweisung, d.h. alle Variablen sind undefiniert

Nachfolgerfunktion: Belegung einer weiteren Variablen, so dass kein Konflikt mit vorherigen Zuweisungen entsteht.

Zieltest: Sind alle Variablen definiert?

Pfadkosten: konstant, z.B. 1 pro Schritt

CSP als Suchproblem: Lösungsansatz

Jede Lösung hat Tiefe $|V|$. Daher ist Tiefensuche geeignet.

Es kommt nicht auf den Suchpfad an. Daher sind lokale Methoden gut geeignet.

Achtung

Die Suche findet in dem auf der vorhergehenden Folie beschriebenen Suchraum statt, nicht in dem davor eingeführten Graphen.

Backtracking-Suche

Die Zustandsübergangsoperatoren sind kommutativ, d.h. es macht keinen Unterschied, ob wir zuerst $x \leftarrow v$ zuweisen und dann $y \leftarrow w$ oder umgekehrt.

Daher beschränken wir uns bei jeder Verzweigung im Suchbaum auf die verschiedenen Belegungen von nur einer Variablen.

Sind alle Belegungen dieser Variablen unerfüllbar, gehen wir eine Ebene im Suchbaum zurück → Backtracking.

Heuristiken zur Lösung von CSP-Suchproblemen

Heuristik der maximal eingeschränkten Variablen: Bevorzuge die Variable mit dem kleinsten noch möglichen Wertebereich.

Heuristik der minimalen Breitenordnung: Wähle in dem noch verbleibenden Constraint-Graphen die Ecke mit dem höchsten Eckengrad.

Heuristik des minimalen Konflikts: Wert der Zuordnung $x \leftarrow w$ ist Summe über alle noch undefinierten Variablen y der Anzahl der Werte $v \in dom(y)$, so dass die Belegung $\{x \leftarrow w, y \leftarrow v\}$ ein Constraint verletzt.

Beispiel

4×4-Dame mit $(a, 1)$ belegt. Ist $(b, 3)$ oder $(b, 4)$ besser?

- *Heuristik der minimalen Breitenordnung* ist nützlich bei unterschiedlichen Knotengeraden im Constraint-Graphen.
- *Heuristik der maximalen eingeschränkten Variablen* ist nützlich, wenn die Constraints stark einschränken, d.h. die Belegung einer Variablen die anderen Variablen stark beeinflusst.
- mögliche Kombination: erst alle max. eingeschränkten Variablen suchen, ggf. unter diesen dann die mit min. Beite wählen. Diese dann mit Heuristik des minimalen Konflikts belegen.

Komplexität/Unentscheidbarkeit

Constraint-Erfüllungsprobleme sind bei endlichen Wertebereichen im allgemeinen NP-vollständig.

In den meisten praktischen Anwendungen sind sie aber um Größenordnungen besser als allgemeine Suchverfahren.

Gleichungssysteme und speziell Diophantische Gleichungen (ganzzahlige Lösungen für $a_1 x_{1,1}^{i_{1,1}} \dots x_{n,1}^{i_{n,1}} + \dots + a_m x_{1,m}^{i_{1,m}} \dots x_{n,m}^{i_{n,m}} = c$) lassen sich als CSP beschreiben. Diese sind beweisbar unentscheidbar, d.h. es ist kein universeller Lösungsalgorithmus für CSP möglich.

Constraint-Propagierung

Idee: Constraints propagieren und so sukzessive die Wertebereiche der Variablen einschränken (und damit den Suchraum verkleinern).

Kann als vorbereitender Schritt vor der Suche eingesetzt werden oder nach jedem Suchschritt durchgeführt werden.

Constraint-Propagierung: Beispiel

An der Tafel:

Das Constraintnetz $\mathcal{C} = \{C_1, C_2, C_3\}$ sei wie folgt gegeben:

- $V = \{x, y, z, w\}$,
  $dom(x) = dom(y) = dom(z) = dom(w) = \{1, 2, 3, 4\}$
- $C_1 = \{(x, y) \mid x > y\}$ über $V_1 = \{x, y\}$
- $C_2 = \{(y, z) \mid y > z\}$ über $V_2 = \{y, z\}$
- $C_3 = \{(z, w) \mid z > w\}$ über $V_3 = \{z, w\}$

– Was passiert bei Tiefensuche?

Constraint-Propagierung: Konsistenz von Problemen

Wir betrachten nur zweistellige CSPs, da sich alle anderen darauf zurückführen lassen.

Eine Variable a heisst *konsistent*, wenn jede ihrer Belegungen alle einstelligen Constraints der Form $C(x)$ löst. Ein CSP heisst *knotenkonsistent* (oder 1-*konsistent*), wenn jede Variable konsistent ist.

Ein zweistelliges Constraint $C(x, y)$ heisst konsistent, wenn sich jede Belegung $\{x \leftarrow v\}$ zu $\{x \leftarrow v, y \leftarrow w\}$ ohne Verstoß gegen $C(x, y)$ erweitern lässt. Ein CSP heisst *kantenkonsistent* (oder *lokal konsistent*), wenn es knotenkonsistent ist und wenn jedes zweistellige Constraint konsistent ist.

Constraint-Propagierung

Ziel der Propagierung: Konstruktion eines kleineren kantenkonsistenten CSPs durch Streichen von Werten aus den Wertebereichen.

Ist ein Constraint $C(x, y)$ nicht konsistent, gibt es (nach Def.) eine Belegung $\{x \leftarrow v\}$, die sich durch kein w zu $\{x \leftarrow v, y \leftarrow w\}$ ohne Verstoß gegen $C(x, y)$ erweitern lässt. Dann kann v aus dem Wertebereich von x gestrichen werden.

(Passiert dies innerhalb eines Suchbaums, muss ggf. beim Backtracking das Streichen rückgängig gemacht werden.)

An der Tafel: Beispiel

Constraint-Propagierung: Ergebnis

Gelingt es nicht, ein kantenkonsistentes CSP zu erzeugen, ist das ursprüngliche Problem unlösbar.

Gelingt es, so folgt nicht notwendigerweise, dass das Problem lösbar ist. Aber zumindest ist der Suchraum verringert worden.

Kantenkonsistenz-Algorithmus AC-3

Erzeugt ein CSP mit möglicherweise reduzierten Wertebereichen:

Eingabe: csp, ein binäres CSP mit Variablen $\{x_1, x_2, \ldots, x_n\}$

Lokale Variablen: $queue$, eine Schlange mit Kanten; enthält anfangs alle Kanten von csp

while $queue$ nicht leer **do**
  $(x_i, x_j) \leftarrow$ remove-first($queue$)
  if remove-inconsistent-values(x_i, x_j) **then**
    for each x_k in neighbors(x_i) **do**
      füge (x_k, x_i) zur $queue$ hinzu (falls dort noch nicht vorhanden).

Kantenkonsistenz-Algorithmus AC-3

```
function remove-inconsistent-values(x_i, x_j)
/* gibt true bei Entfernung eines Wertes zurück */
    removed ← false
    for each v in dom(x_i) do
        wenn für kein w ∈ dom(x_j) das Paar (v, w) das
        Constraint zwischen x_i und x_j erfüllt
            dann lösche v aus dom(x_i); removed ← true
    return removed
```

Bildverstehen als Constraint-Problem

Aufgabe

Zweidimensionales Abbild (Linienzeichnung) eines ebenflächig begrenzten dreidimensionalen Körpers (Polyeder) interpretieren.

Voraussetzungen an Zeichnung

- keine Schatten oder Bruchlinien
- verdeckte Kanten sind nicht sichtbar
- alle Eckpunkte sind Schnittpunkte genau drei aufeinandertreffender Flächen
- „Allgemeiner Beobachtungstandpunkt“: bei geringen Ortsveränderungen darf kein Schnittpunkt seinen Typ wechseln

Bildverstehen als Constraint-Problem

Vom Objekt bis zur internen Repräsentation sind in der Praxis mehrere Schritte nötig:
Objekt → Bild → Pixel → Kontrastbereiche → Kanten → interne Repräsentation

Zweck:

- Existiert eine physikalisch mögliche Interpretation?
- zulässige Interpretation finden
- Objekte identifizieren (insbes. Aussenkanten)

Voraussetzungen sind in Praxis nicht immer erfüllt (z.B. Spitze Cheops-Pyramide, Würfel frontal gesehen)

Bildverstehen als Constraint-Problem

Mögliche Linientypen

Begrenzungslinien

Linien, bei denen eine der angrenzenden Flächen verdeckt ist – die äußeren Kanten des Objekts. Kennzeichnung mit einem Pfeil „$\rightarrow$“.
Die Pfeile der Begrenzungslinien sind so gerichtet, daß die Körperfläche immer rechts liegt.

Innenlinien

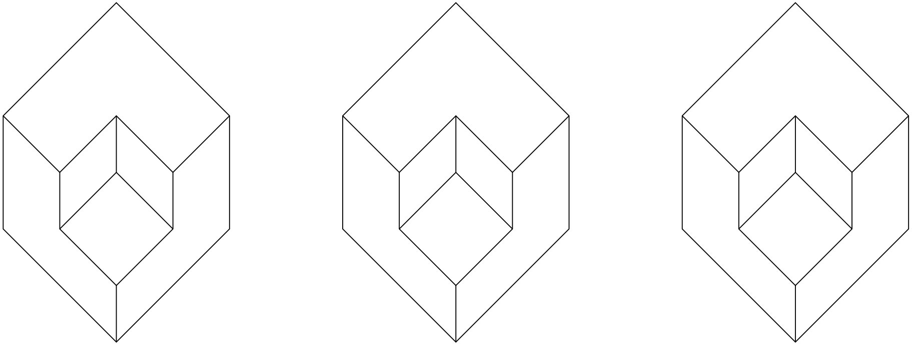
Kanten im Inneren des Objekts

Konkave Linien: Beide Begrenzungsflächen schließen den Beobachtungsstandpunkt ein. Ein Beispiel wäre der Blick in ein geöffnetes Buch. Kennzeichnung mit einem „$-$“.

Konvexe Linien: Beide Grenzflächen sind vom Beobachter abgewandt, wie bei einem Würfel. Kennzeichnung mit einem „$+$“.

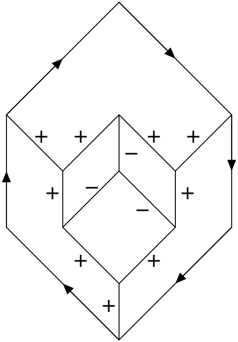
Bildverstehen als Constraint-Problem

Beispiele zur Beschriftung

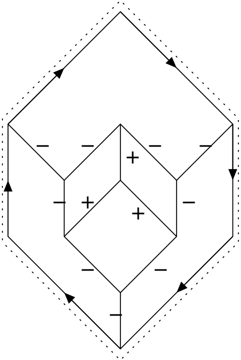


Bildverstehen als Constraint-Problem

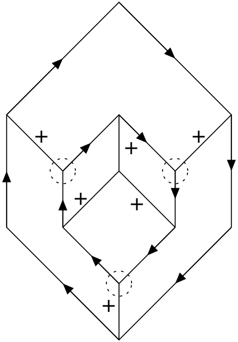
Beispiele zur Beschriftung



Großer Würfel mit herausgeschnittenem kleinen Würfel



Kleiner Würfel in einer Ecke (hier fehlt eigentlich der umgebende Körper)

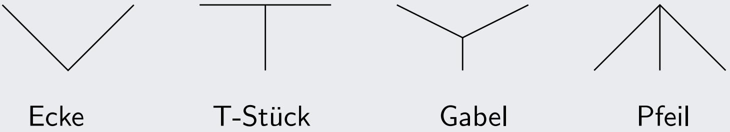


Großer Würfel mit herausstehendem kleinen Würfel (hier treffen sich in einigen Schnittpunkten mehr als drei Flächen)

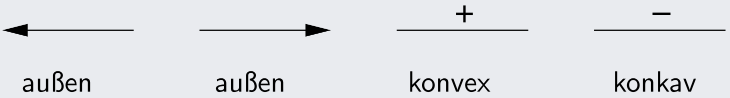
Bildverstehen als Constraint-Problem

Arten von Schnittpunkten und ihre Beschriftung

4 Typen von Schnittpunkten (aufgrund der Voraussetzungen)



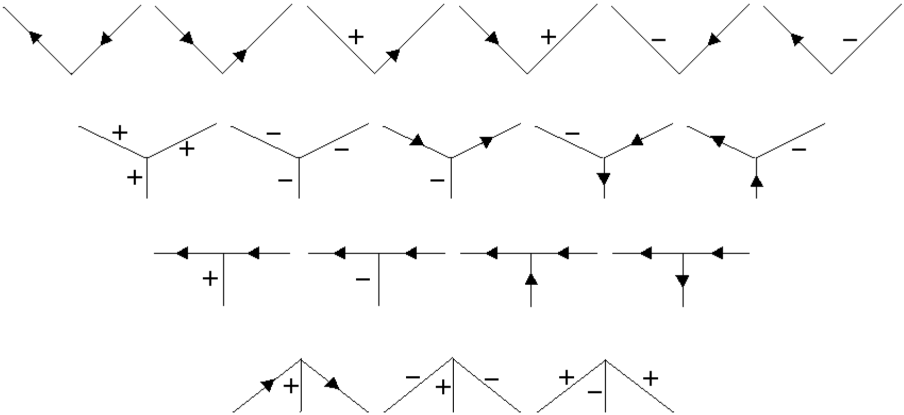
3 Typen von Linien



Es gibt für jede Kante jedes Knotens vier mögliche Beschriftungen.
Insgesamt also $4^2 + 4^3 + 4^3 + 4^3 = 208$ Möglichkeiten.

Bildverstehen als Constraint-Problem

Die 18 physikalisch möglichen Beschriftungen



Für eine gegebene 2D-Zeichnung soll eine konsistente Beschriftung der Kanten gefunden werden, d.h.

- Ecken gemäß den 18 zulässigen Typen
- längs einer Kante darf sich die Beschriftung nicht ändern

Als Constraint-Problem:

Variablen sind die *Kanten* mit Wertebereich {„←“, „→“, „−“ , „+“}

Constraints ergeben sich an den *Ecken* (zulässige Typen)

Für realistische Bilder existiert stets (mindestens) eine konsistente Beschriftung. Es gibt manchmal aber auch konsistente Beschriftungen bei unrealistischen Bildern.
In komplexeren Bildern weitere Constraints durch Licht/Schatten und Bruchlinien.

Einfaches Suchverfahren

1. Auswahl einer Ecke, diese beschriften
2. Fortlaufend Nachbarecken beschriften, solange dies möglich ist.
3. Beim Auftreten von Widersprüchen Backtracking (alternative Ecken und Beschriftungen in 2.)

Verfahren von Waltz

1. Alle Ecken mit den dort möglichen Beschriftungstypen versehen
2. Fortlaufend jeweils Paare von Nachbarecken vergleichen:
   Bei beiden Ecken die nicht miteinander vereinbaren Beschriftungstypen entfernen, bis keine Änderungen mehr möglich sind

Bildverstehen als Constraint-Problem

Anwendungen

Figure 6: Airplane tracking on apron and runway (top), person tracking on apron and taxiway (bottom). Different messages describing each situation in terms of object, area and danger level are shown. Moreover, in the first sequence a still object is not detected, while the second sequence shows the system robustness to low illumination.

Zeitliches Schließen als Constraint-Problem

Aufgabe

Für eine Menge von Aussagen über zeitliche Zusammenhänge prüfen ob sie konsistent sind bzw. weitere, nicht explizit gegebene Zusammenhänge finden.

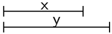
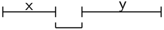
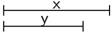
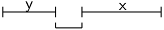
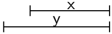
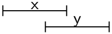
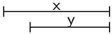
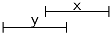
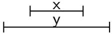
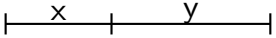
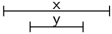
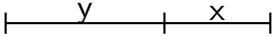
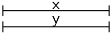
Beispiel

Gegeben seien die Zeitintervalle A, B, C, D mit folgenden Beziehungen:

- A beginnt während B
- B beginnt nicht vor C
- B liegt zeitlich völlig nach D
- C und D beginnen gleichzeitig

Was lässt sich über die Beziehung zwischen A und D sagen ?

Die folgenden Beziehungen zwischen Zeitintervallen sollen als Relationen über Intervallen definiert werden.

	$s(x, y)$	STARTS(x,y)		$b(x, y)$	BEFORE(x,y)
	$s_i(x, y)$	Inverses		$b_i(x, y)$	Inverses
	$f(x, y)$	FINISHES(x,y)		$o(x, y)$	OVERLAPS(x,y)
	$f_i(x, y)$	Inverses		$o_i(x, y)$	Inverses
	$d(x, y)$	DURING(x,y)		$m(x, y)$	MEETS(x,y)
	$d_i(x, y)$	Inverses		$m_i(x, y)$	Inverses
				$e(x, y)$	EQUALS(x,y)