Grundlagen der Künstlichen Intelligenz

9. Constraint-Satisfaction-Probleme: Einführung

Malte Helmert

Universität Basel

8. April 2013

Einordnung

Einordnung:

Constraint-Satisfaction-Probleme

Umgebung:

- statisch vs. dynamisch
- deterministisch vs. nicht-deterministisch vs. stochastisch
- vollständig vs. partiell vs. nicht beobachtbar
- diskret vs. stetig
- ein Agent vs. mehrere Agenten

Lösungsansatz:

- problemspezifisch vs. allgemein vs. lernend

Constraint-Satisfaction-Probleme: Überblick

Kapitelüberblick Constraint-Satisfaction-Probleme:

- Einführung ($\rightsquigarrow$ dieses Kapitel)
- Algorithmen ($\rightsquigarrow$ Kapitel 10)
- Problemstruktur ($\rightsquigarrow$ Kapitel 11)

Einführung und Beispiele

Constraints

Was ist ein Constraint?

constraint = Einschränkung, Nebenbedingung (math.)

Bedingung, die jede Lösung eines Problems erfüllen muss

Verwendung:

- Mathematik: Anforderung an die Lösung eines Optimierungsproblems (z. B. Gleichung, Ungleichung)
- Software-Testing: Spezifikation von Invarianten für Überprüfung von Datenkonsistenz (z. B. Assertions)
- Datenbanken: für Integritätsbedingungen (z.B. foreign key)

Constraint-Satisfaction-Probleme informell

Gegeben:

- eine Menge von Variablen mit einem bestimmten Wertebereich
- eine Menge von Constraints (Bedingungen), die diese Variablen erfüllen müssen
  - meist binär, d. h. jede Bedingung spricht über zwei Variablen

Gesucht:

- eine Belegung der Variablen, die alle Constraints erfüllt

Beispiele

Beispiele für Constraint-Satisfaction-Probleme

- lateinische Quadrate
- 8-Damen-Problem
- Sudoku
- Graphfärbung
- Erfüllbarkeit in Aussagenlogik

Komplexere Beispiele:

- Datenbankanfragen
- Gleichungs- und Ungleichungssysteme

Beispiel: lateinische Quadrate

lateinische Quadrate

Wie kann man

- eine $n \times n$-Matrix mit n Symbolen bilden,
- so dass jedes Symbol genau einmal in jeder Zeile und Spalte auftritt?

$$\begin{bmatrix} 1 \end{bmatrix} \quad \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix} \quad \begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \\ 3 & 1 & 2 \end{bmatrix} \quad \begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 1 \\ 3 & 4 & 1 & 2 \\ 4 & 1 & 2 & 3 \end{bmatrix}$$

Es gibt 12 verschiedene lateinische Quadrate der Grösse 3, 576 der Grösse 4, 161'280 der Grösse 5, ..., 5'524'751'496'156'892'842'531'225'600 der Grösse 9.

Beispiel: 8-Damen-Problem

8-Damen-Problem

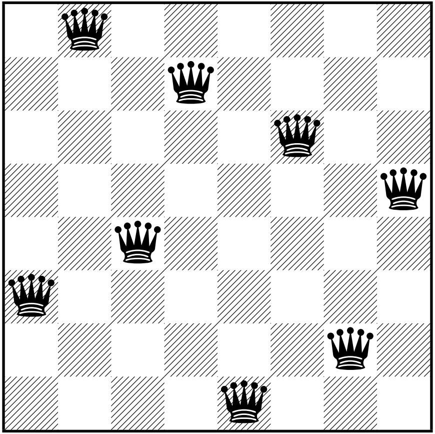
Wie kann man

- 8 Damen auf einem Schachbrett platzieren,
- so dass sich keine zwei Damen bedrohen?

- ursprünglich 1848 vorgestellt
- Varianten: Brettgrösse; andere Figuren; mehr Dimensionen

Das Problem hat 12 Lösungen, wenn man symmetrische Lösungen (Rotationen, Spiegelungen) nur einmal zählt

8-Damen-Problem: eine Lösung



eine Lösung des 8-Damen-Problems

Beispiel: Sudoku

Sudoku

Wie kann man

- eine partiell gefüllte 9×9-Matrix mit den Zahlen 1–9 füllen,
- so dass jede Zeile, jede Spalte und jeder der neun 3×3-Blöcke jede Zahl genau einmal enthält?

2	5			3		9		1
	1				4			
4		7				2		8
		5	2					
				9	8	1		
	4				3			
			3	6			7	2
	7							3
9		3				6		4

Beispiel: Sudoku

Sudoku

Wie kann man

- eine partiell gefüllte 9×9-Matrix mit den Zahlen 1–9 füllen,
- so dass jede Zeile, jede Spalte und jeder der neun 3×3-Blöcke jede Zahl genau einmal enthält?

2	5	8	7	3	6	9	4	1
6	1	9	8	2	4	3	5	7
4	3	7	9	1	5	2	6	8
3	9	5	2	7	1	4	8	6
7	6	2	4	9	8	1	3	5
8	4	1	6	5	3	7	2	9
1	8	4	3	6	9	5	7	2
5	7	6	1	4	2	8	9	3
9	2	3	5	8	7	6	1	4

Beispiel: Sudoku

Sudoku

Wie kann man

- eine partiell gefüllte 9×9-Matrix mit den Zahlen 1–9 füllen,
- so dass jede Zeile, jede Spalte und jeder der neun 3×3-Blöcke jede Zahl genau einmal enthält?

2	5	8	7	3	6	9	4	1
6	1	9	8	2	4	3	5	7
4	3	7	9	1	5	2	6	8
3	9	5	2	7	1	4	8	6
7	6	2	4	9	8	1	3	5
8	4	1	6	5	3	7	2	9
1	8	4	3	6	9	5	7	2
5	7	6	1	4	2	8	9	3
9	2	3	5	8	7	6	1	4

Zusammenhang lateinische Quadrate?

Sudoku: Trivia

- wohlgeformte Sudokus haben genau eine Lösung
- dafür müssen im Minimalfall 17 Felder vorausgefüllt werden (McGuire et al., 2012)
- 6'670'903'752'021'072'936'960 gelöste Konfigurationen
- nur 5'472'730'538 unter Ausnutzung von Symmetrie

Beispiel: Graphfärbung

Graphfärbung

Wie kann man

- die Knoten eines gegebenen Graphen mit k Farben
- so einfärben, dass zwei benachbarte Knoten nie dieselbe Farbe haben?

(Der Graph und k sind Parameter des Problems.)

Beispiel: Graphfärbung

Graphfärbung

Wie kann man

- die Knoten eines gegebenen Graphen mit k Farben
- so einfärben, dass zwei benachbarte Knoten nie dieselbe Farbe haben?

(Der Graph und k sind Parameter des Problems.)

NP-vollständiges Problem

- selbst für den Spezialfall planarer Graphen und $k = 3$
- einfach für $k = 2$ (auch für allgemeine Graphen)

Beispiel: Graphfärbung

Graphfärbung

Wie kann man

- die Knoten eines gegebenen Graphen mit k Farben
- so einfärben, dass zwei benachbarte Knoten nie dieselbe Farbe haben?

(Der Graph und k sind Parameter des Problems.)

NP-vollständiges Problem

- selbst für den Spezialfall planarer Graphen und $k = 3$
- einfach für $k = 2$ (auch für allgemeine Graphen)

Zusammenhang Sudoku?

Vier-Farben-Problem

berühmtes mathematisches Problem: Vier-Farben-Problem

- kann man planare Graphen immer mit 4 Farben einfärben?
- Vermutung aufgestellt von Francis Guthrie (1852)
- 1890 erster Beweis, dass 5 Farben ausreichen
- mehrmals falsche Beweise, die über 10 Jahre Bestand hatten

Vier-Farben-Problem

berühmtes mathematisches Problem: Vier-Farben-Problem

- kann man planare Graphen immer mit 4 Farben einfärben?
- Vermutung aufgestellt von Francis Guthrie (1852)
- 1890 erster Beweis, dass 5 Farben ausreichen
- mehrmals falsche Beweise, die über 10 Jahre Bestand hatten
- gelöst 1976 durch Appel und Haken: 4 Farben reichen
- Appel und Haken reduzierten Problem auf 1936 Fälle, die durch Computer überprüft wurden
- erstes grosses offenes Problem in der Mathematik, das per Computer gelöst wurde
  $\rightsquigarrow$ führte zu Kontroverse: Ist das ein Beweis?

Erfüllbarkeit in Aussagenlogik

Erfüllbarkeit in Aussagenlogik

Wie kann man

- eine gegebene Menge von Aussagevariablen
  so mit wahr/falsch belegen,
- dass eine gegebene Menge von Klauseln
  (Formeln der Art $X \vee \neg Y \vee Z$) erfüllt (wahr) wird?

Erfüllbarkeit in Aussagenlogik

Erfüllbarkeit in Aussagenlogik

Wie kann man

- eine gegebene Menge von Aussagevariablen so mit wahr/falsch belegen,
- dass eine gegebene Menge von Klauseln (Formeln der Art $X \vee \neg Y \vee Z$) erfüllt (wahr) wird?

Anmerkungen:

- NP-vollständig (Cook 1971; Levin 1973)
- Klauselform (statt beliebiger Logikformeln) ist keine echte Einschränkung
- Beschränkung auf Klauseln der Länge 3 wäre ebenfalls keine echte Einschränkung

Erfüllbarkeit in Aussagenlogik

Erfüllbarkeit in Aussagenlogik

Wie kann man

- eine gegebene Menge von Aussagevariablen so mit wahr/falsch belegen,
- dass eine gegebene Menge von Klauseln (Formeln der Art $X \vee \neg Y \vee Z$) erfüllt (wahr) wird?

Anmerkungen:

- NP-vollständig (Cook 1971; Levin 1973)
- Klauselform (statt beliebiger Logikformeln) ist keine echte Einschränkung
- Beschränkung auf Klauseln der Länge 3 wäre ebenfalls keine echte Einschränkung

Zusammenhang zu vorigen Problemen (z.B. Sudoku)?

Praktische Anwendungen

- Es gibt Tausende praktischer Anwendungen von Constraint-Satisfaction-Problemen.
- Dies gilt allein schon für das Erfüllbarkeitsproblem der Aussagenlogik.

Einige Beispiele:

- Verifikation digitaler Schaltkreise
- Timetabling (Erstellen von Zeit- und Belegungsplänen)
- Planung von Wasser- und Elektrizitätsleitungen, Strassen und ähnlicher Infrastruktur
- Zuweisung von Frequenzspektren (Rundfunk, Mobilfunk)
- Konfiguration von elektronischen Geräten (z. B. Drucker)

Constraint-Netze

Constraint-Netze: informell

Constraint-Netze: informelle Definition

Ein Constraint-Netz ist definiert durch

- eine endliche Menge von Variablen
- einen endlichen Wertebereich für jede Variable
- eine Menge von Constraints (hier: binäre Relationen), denen die Variablen genügen müssen

Wir suchen eine Lösung für das Netz, d. h. eine Belegung der Variablen, die allen Constraints genügt.

Oft sagt man Constraint-Satisfaction-Problem (CSP) statt Constraint-Netz.

Im strengen Sprachgebrauch ist ein „CSP“ aber das algorithmische Problem der Lösungsfindung für Constraint-Netze.

Constraint-Netze: formal

Definition (binäres Constraint-Netz)

Ein (binäres) Constraint-Netz ist ein 3-Tupel $\mathcal{C} = \langle V, \text{dom}, (R_{uv}) \rangle$ mit:

- V ist eine nicht-leere, endliche Menge von Variablen,
- dom ist eine Funktion, die jeder Variable v einen nicht-leeren endlichen Wertebereich $\text{dom}(v)$ zuordnet
- $(R_{uv})_{u,v \in V, u \neq v}$ ist eine Familie von binären Relationen (Constraints) über den Variablen, wobei für alle $u \neq v$ gilt: $R_{uv} \subseteq \text{dom}(u) \times \text{dom}(v)$

Häufige Generalisierungen:

- unendliche Wertebereiche (z.B. $\text{dom}(v) = \mathbb{Z}$)
- mehrstellige Constraints (z.B. Erfüllbarkeit in Aussagenlogik)

Binäre Constraints

Binäre Constraints:

- Wenn u, v zwei Variablen sind, drückt der Constraint R_{uv} aus, welche gemeinsamen Belegungen der Variablen in Lösungen erlaubt sind.
- Wenn $R_{uv} = \mathrm{dom}(u) \times \mathrm{dom}(v)$, ist der Constraint trivial: es besteht keine Einschränkung, und er wird bei der Beschreibung normalerweise weggelassen (ist aber formal gesehen immer vorhanden!)
- Constraints R_{uv} und R_{vu} sprechen über dieselben Variablen. Daher gibt man normalerweise nur einen der beiden an.

Unäre Constraints

Unäre Constraints:

- Oft ist es praktisch, auch zusätzliche Einschränkung an einzelne Variablen als Constraints zu notieren.
- Man spricht dann von unären Constraints.
- Ein unärer Constraint R_v für $v \in V$ entspricht einfach einer Reduktion von $\mathrm{dom}(v)$ auf die durch R_v erlaubten Werte.
- Formal sind unäre Constraints unnötig, aber ihre Verwendung erlaubt es oft, Constraint-Netze klarer zu beschreiben.

Kompakte Kodierung und allgemeine Constraint-Löser

Constraint-Netze bieten kompakte Kodierungen
einer grossen Belegungsmenge:

- Betrachte ein Problem mit n Variablen, deren Wertebereiche alle jeweils k Elemente umfassen.

$\rightsquigarrow$ k^n Belegungen

Kompakte Kodierung und allgemeine Constraint-Löser

Constraint-Netze bieten kompakte Kodierungen
einer grossen Belegungsmenge:

- Betrachte ein Problem mit n Variablen,
deren Wertebereiche alle jeweils k Elemente umfassen.

$\rightsquigarrow$ k^n Belegungen

- Für die Beschreibung als Constraint-Netz
muss man maximal $\binom{n}{2} = O(n^2)$ Constraints angeben,
von denen jeder aus maximal k^2 Paaren besteht

$\rightsquigarrow$ Kodierungsgrösse $O(n^2 k^2)$

- Die Zahl der Belegungen ist also exponentiell grösser
als die Beschreibung des Constraint-Netzes

Kompakte Kodierung und allgemeine Constraint-Löser

Constraint-Netze bieten kompakte Kodierungen
einer grossen Belegungsmenge:

- Betrachte ein Problem mit n Variablen, deren Wertebereiche alle jeweils k Elemente umfassen.

$\rightsquigarrow$ k^n Belegungen

- Für die Beschreibung als Constraint-Netz muss man maximal $\binom{n}{2} = O(n^2)$ Constraints angeben, von denen jeder aus maximal k^2 Paaren besteht

$\rightsquigarrow$ Kodierungsgrösse $O(n^2 k^2)$

- Die Zahl der Belegungen ist also exponentiell grösser als die Beschreibung des Constraint-Netzes
- Damit eignen sich solche Beschreibungen von Constraint-Netzen als Input für allgemeine Constraint-Löser.

Beispiel: 4-Damen-Problem

4-Damen-Problem als Constraint-Netz

- Variablen: $V = \{v_1, v_2, v_3, v_4\}$
  v_i kodiert Zeile der Dame in der i-ten Spalte
- Wertebereiche:
  $\mathrm{dom}(v_1) = \mathrm{dom}(v_2) = \mathrm{dom}(v_3) = \mathrm{dom}(v_4) = \{1, 2, 3, 4\}$
- Constraints: für alle $1 \leq i < j \leq 4$ setzen wir: $R_{v_i,v_j} = \{\langle k, l\rangle \in \{1, 2, 3, 4\} \times \{1, 2, 3, 4\} \mid k \neq l \wedge |k - l| \neq |i - j|\}$
  z. B. $R_{v_1,v_3} = \{\langle 1, 2\rangle, \langle 1, 4\rangle, \langle 2, 1\rangle, \langle 2, 3\rangle, \langle 3, 2\rangle, \langle 3, 4\rangle, \langle 4, 1\rangle, \langle 4, 3\rangle\}$

	v_1	v_2	v_3	v_4
1				
2				
3				
4				

Beispiel: Sudoku

Sudoku als Constraint-Netz

- Variablen: $V = \{v_{ij} \mid 1 \leq i, j \leq 9\}$; v_{ij}: Inhalt Zeile i, Spalte j
- Wertebereiche: $\mathrm{dom}(v) = \{1, \dots, 9\}$ für alle $v \in V$
- unäre Constraints: $R_{v_{ij}} = \{k\}$,
  wenn $\langle i, j \rangle$ vorgegebenes Feld, in dem k steht
- binäre Constraints: für alle $v_{ij}, v_{i'j'} \in V$ setzen wir
  $R_{v_{ij} v_{i'j'}} = \{\langle a, b \rangle \in \{1, \dots, 9\} \times \{1, \dots, 9\} \mid a \neq b\}$,
  falls $i = i'$ (dieselbe Zeile), $j = j'$ (dieselbe Spalte)
  oder $\langle \lceil \frac{i}{3} \rceil, \lceil \frac{j}{3} \rceil \rangle = \langle \lceil \frac{i'}{3} \rceil, \lceil \frac{j'}{3} \rceil \rangle$ (derselbe Block)

2	5			3		9		1
	1				4			
4		7				2		8
		5	2					
				9	8	1		
	4				3			
			3	6			7	2
	7							3
9		3				6		4

Belegungen und Konsistenz

Belegungen

Definition (Belegung, partielle Belegung)

Sei $\mathcal{C} = \langle V, \mathrm{dom}, (R_{uv}) \rangle$ ein Constraint-Netz.
Eine partielle Belegung von $\mathcal{C}$ (oder von V) ist eine Funktion

$$\alpha : V' \to \bigcup_{v \in V} \mathrm{dom}(v)$$

mit $V' \subseteq V$ und $\alpha(v) \in \mathrm{dom}(v)$ für alle $v \in V'$,

Wenn hierbei $V' = V$ gilt, heisst α auch totale Belegung oder einfach Belegung.

- $\rightsquigarrow$ partielle Belegung weist einigen oder allen Variablen Werte aus deren Wertebereich zu
- $\rightsquigarrow$ (totale) Belegung muss auf allen Variablen definiert sein

Konsistenz

Definition (inkonsistent, konsistent, verletzter Constraint)

Eine partielle Belegung α für ein Constraint-Netz $\mathcal{C}$ heisst inkonsistent, wenn es zwei Variablen u, v gibt, für die α definiert ist und für die gilt: $\langle \alpha(u), \alpha(v) \rangle \notin R_{uv}$.

Man sagt dann: α verletzt den Constraint R_{uv}.

Eine partielle Belegung heisst konsistent, wenn sie nicht inkonsistent ist.

triviales Beispiel: leere Belegung ist immer konsistent

Lösung

Definition (Lösung, lösbar)

Sei $\mathcal{C}$ ein Constraint-Netz.

Eine konsistente totale Belegung für $\mathcal{C}$ heisst Lösung von $\mathcal{C}$.

Wenn eine Lösung von $\mathcal{C}$ existiert, heisst $\mathcal{C}$ lösbar.

Wenn keine Lösung existiert, heisst $\mathcal{C}$ inkonsistent.

Konsistenz vs. Lösungseigenschaft

Achtung: dass eine partielle Belegung α konsistent ist, heisst nicht, dass α zu einer Lösung ergänzt werden kann.

Es heisst nur, das bisher (auf den Variablen, für die α definiert ist) kein Constraint verletzt ist.

Beispiel (4-Damen-Problem): $\alpha = \{v_1 \mapsto 1, v_2 \mapsto 4, v_3 \mapsto 2\}$

	v_1	v_2	v_3	v_4
1	q			
2			q	
3				
4		q		

Komplexität von Constraint-Satisfaction-Problemen

Satz (CSPs sind NP-vollständig)

Zu entscheiden, ob ein gegebenes Constraint-Netz lösbar ist, ist ein NP-vollständiges Problem.

Beweis.

Mitgliedschaft in NP:

Eine geratene Lösung kann in polynomieller Zeit in der Eingabegrösse überprüft werden.

NP-Härte:

Der Spezialfall der Graphfärbbarkeit ist bereits als NP-vollständig bekannt. □

Schärfe von Constraint-Netzen

Definition (schärfer, echt schärfer)

Seien $\mathcal{C} = \langle V, \mathrm{dom}, R_{uv} \rangle$ und $\mathcal{C}' = \langle V, \mathrm{dom}', R'_{uv} \rangle$ Constraint-Netze mit denselben Variablen.

$\mathcal{C}$ heisst schärfer als $\mathcal{C}'$, in Symbolen $\mathcal{C} \sqsubseteq \mathcal{C}'$, wenn gilt:

- $\mathrm{dom}(v) \subseteq \mathrm{dom}'(v)$ für alle $v \in V$
- $R_{uv} \subseteq R'_{uv}$ für alle $u, v \in V$ (einschliesslich trivialer Constraints)

Ist mindest eine dieser Teilmengenbeziehungen echt, dann heisst $\mathcal{C}$ echt schärfer als $\mathcal{C}'$, im Symbolen $\mathcal{C} \sqsubset \mathcal{C}'$.

englisch: tighter, strictly tighter

Äquivalenz von Constraint-Netzen

Definition (äquivalent)

Seien $\mathcal{C}$ und $\mathcal{C}'$ Constraint-Netze mit denselben Variablen.
$\mathcal{C}$ und $\mathcal{C}'$ heissen äquivalent, geschrieben $\mathcal{C} \equiv \mathcal{C}'$,
wenn sie dieselben Lösungsmengen besitzen.

Zusammenhang zwischen Schärfe und Äquivalenz?

Ausblick

CSP-Algorithmen

Im Folgekapitel betrachten wir Lösungsalgorithmen für Constraint-Netze.

Grundkonzepte:

- Suche: systematisches Ausprobieren von partiellen Belegungen
- Backtracking: Verwerfen inkonsistenter partieller Belegungen
- Inferenz: Herleiten schärferer äquivalenter Constraints, um Suchraum zu verkleinern (Backtracking früher möglich)