Einführung in die Methoden der Künstlichen Intelligenz

--- Vorlesung vom 3.5.2007 ---
Sommersemester 2007

Prof. Dr. Ingo J. Timm, Andreas D. Lattner
Professur für Wirtschaftsinformatik und Simulation (IS)

4. Constraint Satisfaction Problems

JOHANN WOLFGANG GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Inhalt der Vorlesung

1. Constraint Satisfaction Problems (CSP)
2. Backtracking-Suche für CSPs
3. Lokale Suche für CSPs
4. Die Struktur von Problemen
5. Zusammenfassung

JOHANN WOLFGANG GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Constraint-Probleme

- Herkömmliche Suchprobleme
  - Struktur der Zustände des Problemraums wird vom Suchalgorithmus nicht weiter genutzt
  - Die Zielbedingung ist entweder erfüllt oder nicht erfüllt
- Constraint Satisfaction Problem
  - Die Zielbedingung besteht aus einer Menge von Teilbedingungen die der Zustand alle erfüllen muss
  - Unterscheidung: erfüllt / teilweise erfüllt / nicht erfüllt
  - Constraint-Löser nutzen diese Unterscheidung aus

CSPs

JOHANN WOLFGANG GOETHE UNIVERSITÄT FRANKFURT AM MAIN

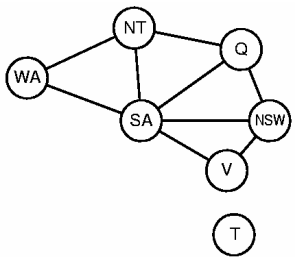
Constraint Satisfaction Problem

- Definiert durch
  - eine Menge Constraint-Variablen $X_1,X_2,\ldots,X_n$
  - eine Menge von Constraints $C_1,C_2,\ldots,C_m$
- Jede Variable X_i hat einen nicht-leeren *Wertebereich* D_i von möglichen *Werten*
- Jedes Constraint C_i besteht aus einer Teilmenge der Variablen und spezifiziert die erlaubten Kombinationen von Werten für diese Teilmenge
- Ein *Zustand* ist durch eine *Zuweisung* von Werten zu den Variablen definiert $\{X_i=v_i, X_j=v_j,\ldots\}$
- Eine Zuweisung, die die Constraints nicht verletzt, heißt *konsistent*
- Eine *Lösung* ist eine vollständige Zuweisung, die die Bedingungen nicht verletzt

CSPs

JOHANN WOLFGANG GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Beispiel: Kartenfärbeproblem



- Einfärben der Bundesstaaten kann als CSP definiert werden
- Ziel: Staaten so einfärben, daß die Nachbarstaaten unterschiedliche Farben haben

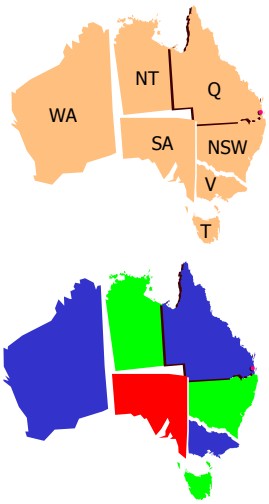
- Constraint-Graph (CG)
  - Variablen als Knoten
  - Constraints als Kanten

CSPs

JOHANN WOLFGANG GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Beispiel: Kartenfärbeproblem

- Variablen sind hier Regionen
  - WA, NT, Q, NSW, V, SA, T
- Wertebereich
  - {rot, grün, blau}
- Constraints
  - z.B. für WA und NT
    {(rot,grün),(rot,blau),(grün,rot),(grün,blau) , (blau,rot),{blau,grün)}
- Viele Lösungen, z.B.
  - *{WA=blau, NT=grün, Q=blau, NSW=grün, SA=rot, T=grün, V=blau}*



CSPs

JOHANN WOLFGANG GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Eigenschaften

- Problem als CSP behandeln hat Vorteile
- Repräsentation der Zustände als Standardmuster (Menge von Variablen mit Werten)
  - Deswegen kann Nachfolgerfunktion generisch geschrieben werden
  - Effektive Heuristiken können entwickelt werden, weil kein Extra-Domänenwissen benötigt wird.
  - Struktur des Graphen kann zur Vereinfachung des Lösungsprozesses verwendet werden, so daß die Zeitkomplexität meistens exponentiell wird
- CSP kann als *inkrementelle* Formulierung eines Standard-Suchproblems definiert werden
  - Initialzustand, die leere Zuweisungsmenge {}
  - Nachfolgerfunktion, ein Wert kann einer Variablen zugewiesen werden (ohne Konflikt)
  - Zieltest: aktuelle Zuweisung ist vollständig
  - Pfadkosten, konstante Kosten, z.B. 1 für jeden Schritt
- Lösung hat Tiefe *n,* wenn *n* Variablen vorhanden
- Deswegen wird häufig Tiefensuche verwendet

CSPs

JOHANN WOLFGANG GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Eigenschaften (2)

- Pfad ist irrelevant → *lokale Suchverfahren* können auch eingesetzt werden
- Einfachstes CSP: *diskrete* Variablen und endlichen Wertebereich *(finite domain)*
- Bsp.: Kartenfärbeproblem, 8-Damenproblem
- Wenn max(Wertebereich) = *d,* dann ist die Anzahl der möglichen vollständigen Zuweisungen $O(d^n)$, also exponentiell in der Anzahl der Variablen
- Finite domains beinhalten *boolsche CSPs*
- Boolsche CSPs können NP-vollständig sein (Bsp.: 3-SAT Problem)
- Deswegen Lösungen nicht besser als exponentiell

CSPs

JOHANN WOLFGANG GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Eigenschaften (3)

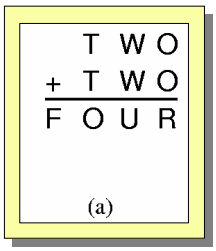
- Diskrete Variablen können auch nicht-endliche Wertebereiche haben (infinite domains), z.B. $\mathbb{N}$
- Bsp.: Jobanweisungen
  - Nicht sinnvoll, alle Kombinationen aufzulisten → constraint language
  - Job_1 braucht 5 Tage und muss vor Job_3 liegen, dann wäre eine algebraische Beschreibung möglich: *StartJob1 + 5 < StartJob3*
- Spezielle Lösungen für *lineare* constraints vorhanden, keine für *kontinuierliche* constraints
- CSPs mit kontinuierlichen Wertebereichen sind sehr verbreitet und erforscht im OR Bereich
- Bsp.: Timing der Experimente für das Hubble Teleskop

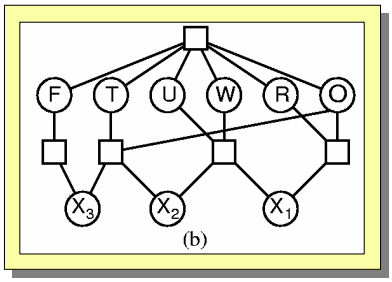
CSPs

UNIVERSITÄT FRANKFURT AM MAIN

Typen von Constraints

- Unär, bindet einen Wert an eine Variable (z.B. *SA = rot*)
- Binär, (z.B. *SA≠NSW*), kann als CG repräsentiert werden
- High-order constraints haben drei oder mehr Variablen (z.B. Kryptoarithmetik-Problem)
  - jeder Buchstaben eine Zahl
  - *alldiff(F,T,U,W,R,O)*
  - Constraints:
    $O + O = R + 10 * X_1$
    $X_1 + W + W = U + 10 * X_2$
    $X_2 + T + T = O + 10 * X_3$
    $X_3 = F$
    mit
    X_1, X_2, X_3 als Hilfsvariablen {0,1}
  - Constraint Hypergraph
- Jedes High-order constraint kann in mehrere binäre constraints überführt werden
- Prioritäts-Constraints: Präferenzen z.B. über Kosten



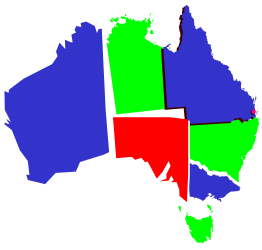


CSPs

UNIVERSITÄT FRANKFURT AM MAIN

Kommutativität

- Jeder Suchalgorithmus aus VL 2+3 kann verwendet werden
- Annahme: Breitensuche, *b* auf Level 0 ist *nd*, weil jede der *d* Werte an jede der *n* Variablen zugewiesen werden kann
- Nächstes Level: *(n-1)d*, d.h. es werden $n!d^n$ Blätter erzeugt, obwohl nur d^n mögliche Zuweisungen existieren
- → Wichtige Eigenschaft: Kommutativität
- Problem ist kommutativ, wenn die Reihenfolge einer Menge von Aktionen keine Auswirkungen auf Ergebnis hat
- → Nachfolger: alle möglichen Zuweisungen für nur *eine* Variable pro Knoten
- Bsp.: *SA=rot, SA=blau,* aber nicht *SA=rot, NSW=grün*
- Mit dieser Restriktion d^n



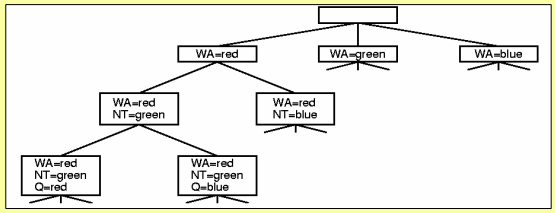
Backtracking-Suche

JOHANN WOLFGANG GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Backtracking-Suche für CSPs

- Begriff "Backtracking" (BT) bei Tiefensuche: Suche nach Wert für eine Variable
- Backtracking, wenn Zuweisung nicht möglich
- Unterschiede:
  - Erweitern der Zuweisung, kein Ersetzen
  - Kein(e) Initialzustand, Nachfolgerfunktion, Zieltest
- Teil eines Suchbaumes

```
function BACKTRACKING-SEARCH(csp) returns a solution, or failure
  return RECURSIVE-BACKTRACKING({ }, csp)

function RECURSIVE-BACKTRACKING(assignment, csp) returns a solution, or failure
  if assignment is complete then return assignment
  var ← SELECT-UNASSIGNED-VARIABLE(VARIABLES[csp], assignment, csp)
  for each value in ORDER-DOMAIN-VALUES(var, assignment, csp) do
    if value is consistent with assignment according to CONSTRAINTS[csp] then
      add {var = value} to assignment
      result ← RECURSIVE-BACKTRACKING(assignment, csp)
      if result ≠ failure then return result
      remove {var = value} from assignment
  end
  return failure
```



Backtracking-Suche

JOHANN WOLFGANG GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Reihenfolge von Variablen und Wert

- Pures Backtracking gehört zur uninformierten Suche →ineffizient für große Probleme
- In VL 3: Wissen über Problem kann helfen, um Suche effizienter zu machen
- Hier: effizient ohne domain-spezifisches Wissen
- Stattdessen Methoden, die die folgenden Fragen beantworten:

1. Welche Variable soll als nächstes betrachtet werden und in welcher Reihenfolge sollen deren Werte ausprobiert werden?
2. Welche Implikationen haben die aktuellen Zuweisungen auf die freien Variablen?
3. Wenn Pfad scheitert (d.h. Zustand, in dem eine Variable keine legalen Werte mehr hat), kann die Suche eben dieses für nachfolgende Pfade vermeiden?

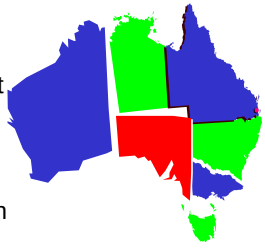
Backtracking-Suche

JOHANN WOLFGANG GOETHE UNIVERSITÄT FRANKFURT AM MAIN

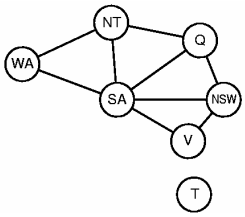
Reihenfolge von Variablen und Wert

$var \leftarrow$ SELECT-UNASSIGNED-VARIABLE(VARIABLES[csp], $assignment$, csp)

- **Minimum Remaining Values (MRV)**
  - Zuerst die Variable mit den wenigsten legalen Werten
  - Bsp.: *WA=blau, NT=grün* → nur eine Möglichkeit für *SA (rot)*
  - Es macht Sinn, *SA* anstatt von*Q* zu wählen
  - Faktor 3-3000 besser als BT
  - auch: Heuristik der am meisten einschränkenden Variable



- **MRV hilft nicht am Anfang →Degree Heuristik**
  - Reduktion des *b* durch Auswahl der Variablen, die in den meisten constraints zu anderen freien Variablen involviert ist
  - Bsp.: *SA=5, T=0, NSW=3*

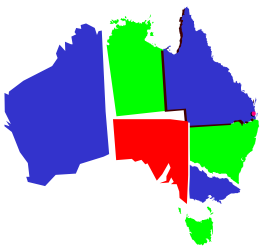


Backtracking-Suche

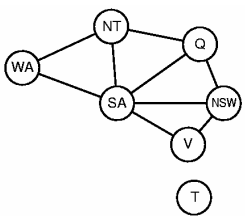
JOHANN WOLFGANG GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Reihenfolge von Variablen und Wert

- **Least Constraining Value (LCV)**
  - Variable ist ausgewählt
  - Werte müssen ausgewählt werden
  - Präferenz: Wert, der die wenigsten Wahlmöglichkeiten für die Nachbar-Variablen ausschliesst
  - Bsp.: Annahme: *WA=blau, NT=grün*, nächste Wahl wäre *Q*: *Rot* wäre keine gute Wahl, weil das die letzte legale Möglichkeit für *SA* ist, deswegen *blau*



- **Generell: LCV Heuristik versucht, maximale Flexibilität für folgende Varaiblenzuweisun-gen zu lassen**



Backtracking-Suche

JOHANN WOLFGANG GOETHE UNIVERSITÄT FRANKFURT AM MAIN

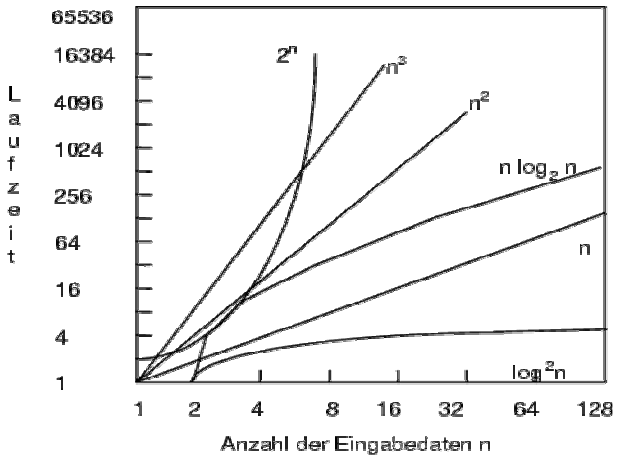
Komplexität

- Komplexität von Algorithmen
  - Worst-case behavior, z.B. Sortieren in $O(n^2)$
  - Liefert obere Schranke für besten Algorithmus
  - Zeigt nicht, dass kein besserer Algorithmus existiert
    z.B. Sortieren in $O(n \log n)$
- Komplexität von Problemen
  - Worst-case behavior
  - Grobe Klassifikation
  - Effizent lösbar: tractable
  - Nicht effizient lösbar: intractable
  - Liefert untere Schranke für den besten Algorithmus

Backtracking-Suche

JOHANN WOLFGANG GOETHE UNIVERSITÄT FRANKFURT AM MAIN

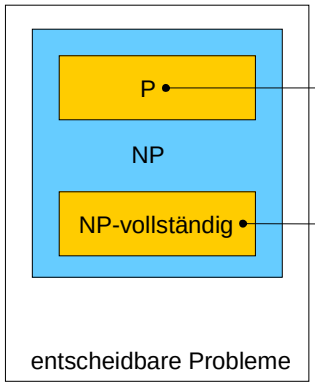
Komplexitätsklassen



[Strube, 2003]

NP-Vollständigkeit

- Grundgedanke
  - NP-vollständige Probleme sind die schwierigsten Probleme in NP
  - Wenn für ein einziges NP-vollständiges Problem ein effizienter Algorithmus gefunden würde, dann wäre *P = NP.*

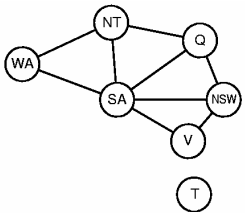


NP-vollständige Probleme angehen durch polynomiale Vorverarbeitung

Backtracking-Suche

Forward Checking

- Bisher: Select-Unassigned-Variable
- Reduzierung des Suchraumes aber auch anders möglich → *Forward checking*
- Forward checking (FC)
  - Wenn der Variablen *X* ein Wert zugewiesen wird, schaut FC auf jede freie Variable, die mit *X* verbunden ist



	WA	NT	Q	NSW	V	SA	T
1. Beginn	RGB	RGB	RGB	RGB	RGB	RGB	RGB
2. WA=rot	R	GB	RGB	RGB	RGB	GB	RGB
3. Q=grün	R	B	G	RB	RGB	B	RGB
4. V=blau	R	B	G	R	B		RGB

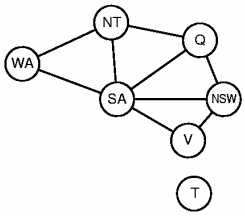
- 3. *NT* und *SA* nur einzelner Wert: Eliminierung von Verzweigungen durch Informations-Propagierung, MRV wählt automatisch *NT* und *SA* aus
- 4. *SA* ist leer → Inkonsistenz erkannt

Backtracking-Suche

JOHANN WOLFGANG GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Constraint Propagation

- Forward checking findet nicht alle Inkonsistenzen, z.B. 3. wenn *WA=rot & Q=grün*, ist für *NT* und *SA* nur *blau* übrig → Nachbarn!
- Forward checking schaut nicht weit genug voraus
- *Constraint propagation* ist ein Ausdruck für Implikationen eines constraints von einer auf die anderen Variablen
- Bsp.: zusätzlich *NT*- und *SA*-Bedingungen prüfen



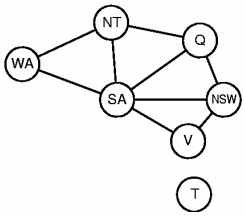
	WA	NT	Q	NSW	V	SA	T
1. Beginn	RGB	RGB	RGB	RGB	RGB	RGB	RGB
2. WA=rot	R	GB	RGB	RGB	RGB	GB	RGB
3. Q=grün	R	B	G	RB	RGB	B	RGB
4. V=blau	R	B	G	R	B		RGB

Backtracking-Suche

JOHANN WOLFGANG GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Kanten-Konsistenz

- Schnelle Methode für constraint propagation
- Kante ist gerichtet, z.B. *SA – NSW*
- Die Kante ist *konsistent*, wenn es für *jeden* Wert *x* von *SA* mindestens *einen* Wert *y* von *NSW* gibt, der mit *x* konsisten ist (gegeben aktuelle Wertebereiche von *SA* und *NSW)*
- 3. *SA=blau, NSW={rot,blau}.* Für *SA=blau* gibt es eine konsistente Zuweisung *NSW=rot*, also ist die Kante von *SA* nach *NSW* konsistent
- Gegenteil: inkonsistent: *NSW=blau*, kein Wert für *SA*



	WA	NT	Q	NSW	V	SA	T
1. Beginn	RGB	RGB	RGB	RGB	RGB	RGB	RGB
2. WA=rot	R	GB	RGB	RGB	RGB	GB	RGB
3. Q=grün	R	B	G	RB	RGB	B	RGB
4. V=blau	R	B	G	R	B		RGB

Backtracking-Suche

Kanten-Konsistenz

- Kanten-Konsistenz kann
  - als Vorbereitung zur Suche und
  - als Propagierungsschritt (wie FC) eingesetzt werden
- Wiederhole solange, bis Inkonsistenzen verschwinden
- AC-3 als Algorithmus
- Zeitkomplexität $O(n^2 d^3)$
- Aber: auch diese Funktion findet nicht alle möglichen Inkonsistenzen

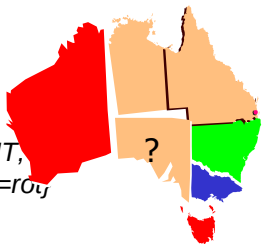
```
function AC3(csp) returns the CSP, possibly with reduced domains
  inputs: csp, a binary CSP with variables {X1, X2, ..., Xn}
  local variables: queue, a queue of arcs, initially all the arcs in csp

  while queue is not empty do
    (Xi, Xj) ← REMOVE-FRONT(queue)
    if REMOVE-INCONSISTENT-VALUES(Xi, Xj) then
      for each Xk in NEIGHBORS[Xi] do
        add (Xk, Xi) to queue

function REMOVE-INCONSISTENT-VALUES(Xi, Xj) returns true iff we remove a value
  removed ← false
  for each x in DOMAIN[Xi] do
    if no value y in DOMAIN[Xj] allows (x,y) to satisfy the constraint between Xi and Xj
      then delete x from DOMAIN[Xi]; removed ← true
  end
  return removed
```


Spezielle Constraints

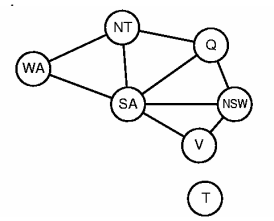
- Ressourcen constraint (auch *atmost* constraint)
- Bsp.: $PA_1,\ldots,PA_4$ ist Anzahl von Personen, die an jeder der vier Aufgaben arbeiten
- Constraint, dass nicht mehr als 10 Personen insgesamt verplant sind ist *atmost(10,PA_1,PA_2,PA_3,PA_4)*
- Inkonsistenz kann durch Prüfen der Summe der Minimalwerte der Wertebereiche gefunden werdenz, z.B.: wenn jede Variable den Wertebereich *{3,4,5,6}* hat, kann *atmost* nicht erfüllt werden
- Löschen von Werten, z.B. {2,3,4,5,6} →5,6 können gelöscht werden, um constraint konsistent zu halten
- Bei großen Ressourcenproblemen (z.B. logistische Probleme) kaum möglich, Wertebereich von Variablen als Menge von Integern zu definieren
- Obere und untere Grenzen
- Bounds propagation als Lösung
- Bsp.: Zwei Flüge 271 und 272 mit Kapazitäten von 165 und 385 Passagieren, Wertebereiche am Anfang:
  - *Flight271* $\in$ *[0,165]*
  - *Flight272* $\in$ *[0,385]*
- Constraint: beide Flüge müssen 420 Passagiere befördern
  - *Flight271 + Flight272 [420,420]*
- Durch bounds propagation bekommt man
  - *Flight271* $\in$ *[35,165]*
  - *Flight272* $\in$ *[255,385]*
- Hohe Praxisrelevanz!

Intelligentes Backtracking

- BT heisst auch chronologisches BT
- Nimmt letzte gemerkte Variable
- Besser wäre intelligente Auswahl



- Bsp.: Fixe Reihenfolge *Q, NSW, V, T, SA, WA, NT*, partielle Zuweisung *{Q=rot, NSW=grün,V=blau,T=rot}*
- Nächste Variable *SA*: Problem!
- Backtracking zu *T* und neue Zuweisung → keine gu Wahl
- Besser wäre hier: gehe zurück zu der Menge der Variablen, die das Problem verursacht haben →*Konfliktmenge*
- Im Bsp.: Konfliktmenge für *SA* ist *{Q,NSW,V}*



- *Backjumping* statt Backtracking

Backtracking-Suche

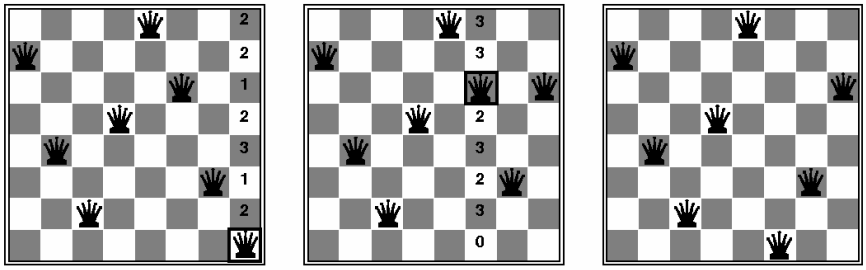
Lokale Suche: min-conflict Heuristik

- Lokale Suche effektiv bei CSPs
- Heuristik: Wert auswählen, der die Anzahl der Konflikte mit anderen Variablen minimiert
- Sehr effektiv, *n*-Damenproblem in weniger als 50 Schritten, auch bei 1.000.000 Damen
- Praxisrelevant

```
function MIN-CONFLICTS(csp, max-steps) returns a solution or failure
  inputs: csp, a constraint satisfaction problem
          max-steps, the number of steps allowed before giving up
  local variables: current, a complete assignment
                   var, a variable
                   value, a value for a variable

  current ← an initial complete assignment for csp
  for i = 1 to max-steps do
      if current is a solution for csp then return current
      var ← a randomly chosen, conflicted variable from VARIABLES[csp]
      value ← the value v for var that minimizes CONFLICTS(var, v, current, csp)
      set var = value in current
  return failure
```

Lokale Suche für CSP

Bsp.: Min-conflict Heuristik



- Lösung für 8-Damenproblem in zwei Schritten
- Jedesmal wird eine Damen für neue Zuweisung ausgewählt
- Die Anzahl der Konflikte ist angezeigt
- Algorithmus folgt den minimalen Konflikten
- Bei mehreren Minima: zufällige Auswahl

Lokale Suche für CSP

Vergleich von CSP-Algorithmen

Problem	BT	BT+MRV	FC	FC+MRV	Min-Conflicts
USA	(> 1,000K)	(> 1,000K)	2K	60	64
n-Queens	(> 40,000K)	13,500K	(> 40,000K)	817K	4K
Zebra	3,859K	1K	35K	0.5K	2K
Random 1	415K	3K	26K	2K	
Random 2	942K	27K	77K	15K	

- **Algorithmen**
  - Einfaches Backtracking
  - Backtracking mit MRV Heuristik
  - Forward checking
  - Forward checking mit MRV Heuristik
  - Minimum conflicts lokale Suche
- **Probleme**
  - Kartenfärbeproblem mit vier Farben für die USA
  - Anzahl der checks, um *n*-Damenproblem für (*n=2-50)* zu lösen
  - Zebra-Puzzle
  - Zwei künstliche Probleme
- **Median über Konsistenzchecks (5 Durchläufe) zur Lösung des Problems**
- **() = keine Lösung gefunden innerhalb der Anzahl Checks**
- **Ergebnis:**
  - FC mit MRV ist besser als BT Algorithmen
  - Gilt nicht immer für Min-Conflicts

Lokale Suche für CSP

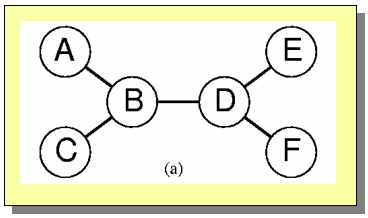
Struktur von Problemen

- Unabhängige Teilprobleme (z.B. Tasmanien und Festland)
- Kann erreicht werden, in dem man Komponenten sucht, die Verbindungen im Graph enthalten
- Komponente = Teilproblem
- Wenn Zuweisung S_i Lösung eines Teilproblems ist, ist $\cup_i S_i$ eine Lösung von $\cup_i CSP_i$
- Warum ist das wichtig?
- Annahme: CSP_i hat c Variablen aus n, c ist eine Konstante, es gibt dann n/c Teilprobleme mit minimal d^c Zeit für die Lösung
- Insgesamt $O(d^c n/c)$, d.h. linear in n
- Ohne Zerlegung $O(d^n)$, d.h. exponentiell
- Konkret: eine Zerlegung eines boolschen CSP mit $n=80$ in vier Teilprobleme mit $c=20$ reduziert den schlechtesten Fall von nahezu unendlich auf weniger als eine Sekunde!

Struktur von Problemen

Struktur von Problemen

- Meistens sind Teilprobleme in CSP nicht unabhängig, sondern haben Verbindung
- Einfachster Fall: Wenn CG einen Baum formt
- *Zeitaufwand für baumstrukturierte CSPs ist linear in n*



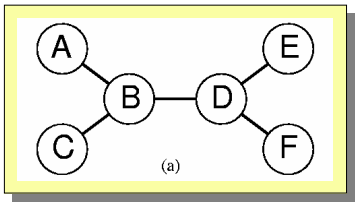
Struktur von Problemen

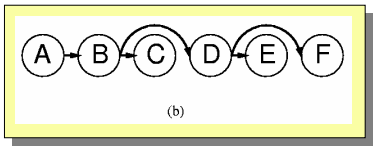
Struktur von Problemen

JOHANN WOLFGANG GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Algorithmus: $O(nd^2)$

1. Wähle Wurzel beliebig und ordne alle Variablen, so daß die Eltern eines Knotens jeweils vor dem Kind stehen (b)
2. For j from n down to 2
   Kanten-Konsistenz (X_i, X_j), mit X_i = Vaterknoten von X_j
3. For j from 1 to n
   Wertzuweisung aller Werte an X_j, die konsistent mit dem Wert von X_i sind, mit X_i = Vaterknoten von X_j



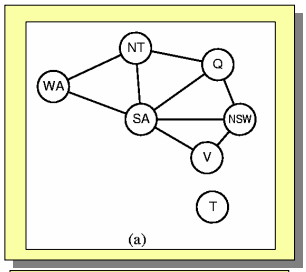


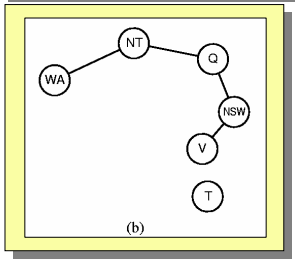
- Mit diesem Wissen kann man versuchen, generellere Probleme auf Bäume zu *reduzieren.* Zwei Ansätze dazu:
  - *Cutset conditioning*
  - *Baumzerlegung*

Cutset conditioning

JOHANN WOLFGANG GOETHE UNIVERSITÄT FRANKFURT AM MAIN

- Bsp.: Eliminieren von *SA,* fester Wert, Wert wird für alle anderen Variablen gelöscht
- Wir bekommen dann einen Baum
- "Cutset", weil der Algorithmus eine Teilmenge *S* wählt, so daß der CG ein Baum wird, *S* wird auch "cycle cutset" genannt



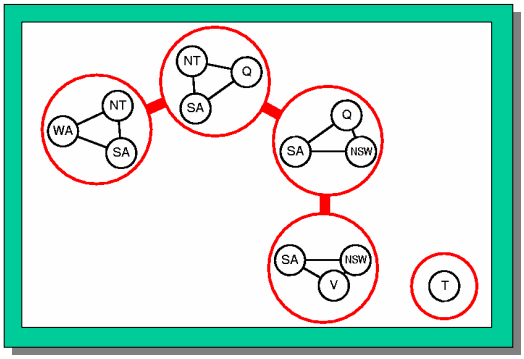


Struktur von Problemen

JOHANN WOLFGANG GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Baumzerlegung

- Idee: Zerlegung des CG in Teilmengen, die verbunden sind
- Jedes Teilproblem wird separat gelöst und am Ende wird kombiniert
- Funktioniert gut, solange die Teilprobleme nicht zu groß werden (wie divide-and-conquer)
- Hier: fünf Teilprobleme



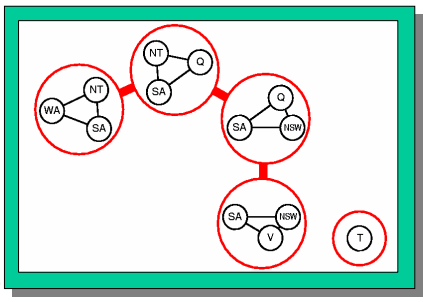
- **Bedingungen:**
  - Jede Variable des Originalproblems muß mindestens in einem Teilproblem zu finden sein
  - Wenn zwei Variablen im Originalproblem verbunden sind, müssen sie zusammen in mindestens einem Teilproblem zu finden sein (mit constraint)
  - Wenn eine Variable in zwei Teilproblemen in dem Baum erscheint, muß sie in jedem Teilproblem erscheinen, wenn diese über einen Pfad verbunden sind (z.B. *SA*)

Struktur von Problemen

JOHANN WOLFGANG GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Baumzerlegung

- *Breite* eines Baumes ist definiert als die Größe des größten Teilproblemes – 1
- Breite des CGs ist definiert durch den minimalen Baum aller Baumzerlegungen
- Wenn ein Graph die Breite *w* hat und wir die korrespondierenden Baumzerlegungen haben, kann das Problem in $O(nd^{w+1})$ gelöst werden
- → CSPs mit Constraint Graphen von begrenzter Baumbreite sind in polynomialer Zeit lösbar!



Struktur von Problemen

JOHANN WOLFGANG GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Zusammenfassung

- Constraint Satisfaction Problem (CSP) besteht aus Variablen mit Bedingungen (constraints). Viele reale Probleme werden damit gelöst. Die Struktur eines CSP kann mit Hilfe eines Constraint Graphen repräsentiert werden.
- Backtracking-Suche wird häufig für die Lösung verwendet
- Minimal Remaining Values (MRV) und Degree Heuristiken sind domänenunabhängige Methoden um die nächste Variable beim Bachtracking auszuwählen. Die Least-Constraining-Value (LCV) Heuristik hilft bei der Reihenfolge der Variablenwerte.
- Forward checking und Kanten-Konsistenz sind Methoden zur Reduzierung des Verzweigungsfaktors des Problems.

JOHANN WOLFGANG GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Zusammenfassung

- Conflict Backjumping merkt sich die Variablen, die zum Scheitern beigetragen haben und springt dahin zurück.
- Lokale Suchalgorithmen können mit großem Erfolg auf CSPs angewendet werden, wenn sie min-conflict Heuristiken verwenden.
- Die Komplexität zur Lösung eines CSP ist stark abhängig von der Struktur des Constraint Graphen. Baum-strukturierte Probleme können in linearer Zeit gelöst werden. Cutset Conditioning kann ein generelles CSP in ein CSP mit Baumstruktur reduzieren. Baumzerlegungstechniken zerlegen ein CSP in einen Baum von Teilproblemen und sind effizient, wenn die Baumbreite des Constraint Graphen schmal ist.