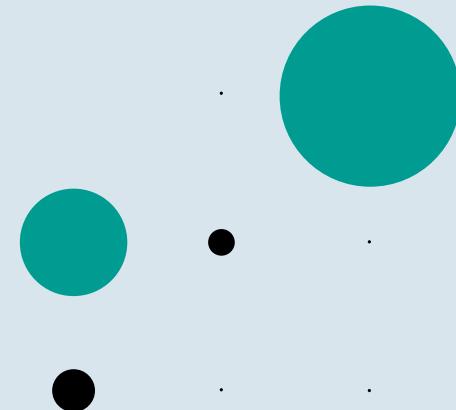




Constraint Satisfaction Problems (CSP)

Probleme unter Randbedingungen -



Inhalt

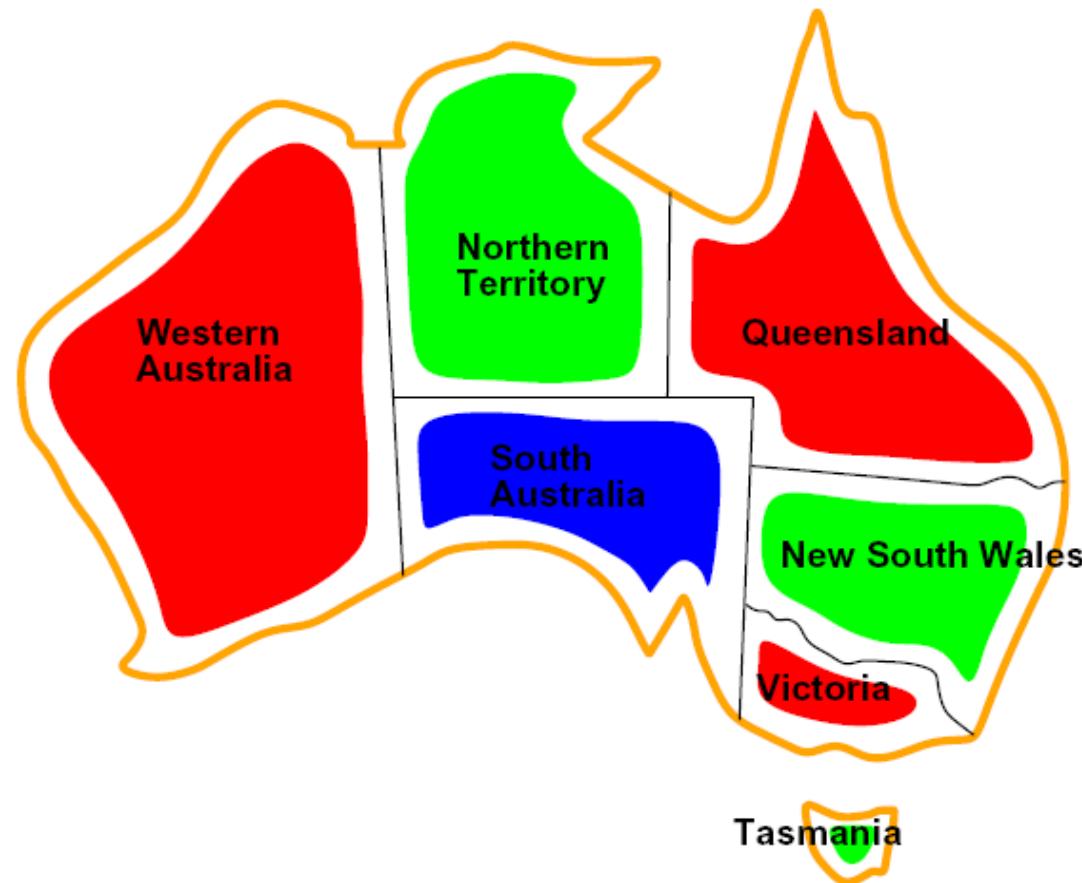
- Constraint Satisfaction Problems
 - Problemstellung
 - Beispiele
- Backtracking
 - Algorithmus
 - Heuristik für Variablen- und Werte-Reihenfolge
 - Forward Checking
- Constraint Propagation (Beschränkungsweitergabe)
 - Kantenkonsistenz
 - AC-3-Algorithmus
- Problemstrukturen
 - Zusammenhangskomponenten
 - CSP in Baumstruktur
 - Schnittmengenkonditionierung
- Lokale Suche

Inhalt

- Constraint Satisfaction Problems
 - Problemstellung
 - Beispiele
- Backtracking
 - Algorithmus
 - Heuristik für Variablen- und Werte-Reihenfolge
 - Forward Checking
- Constraint Propagation (Beschränkungsweitergabe)
 - Kantenkonsistenz
 - AC-3-Algorithmus
- Problemstrukturen
 - Zusammenhangskomponenten
 - CSP in Baumstruktur
 - Schnittmengenkonditionierung
- Lokale Suche

Typisches CSP-Beispiel: Färben von Landkarten

- Färbe Länder
(Variablen)
- mit Farben R, G, B
(Werte)
- so dass,
benachbarte Länder
unterschiedliche
Farben haben
(Randbedingungen,
Constraints)



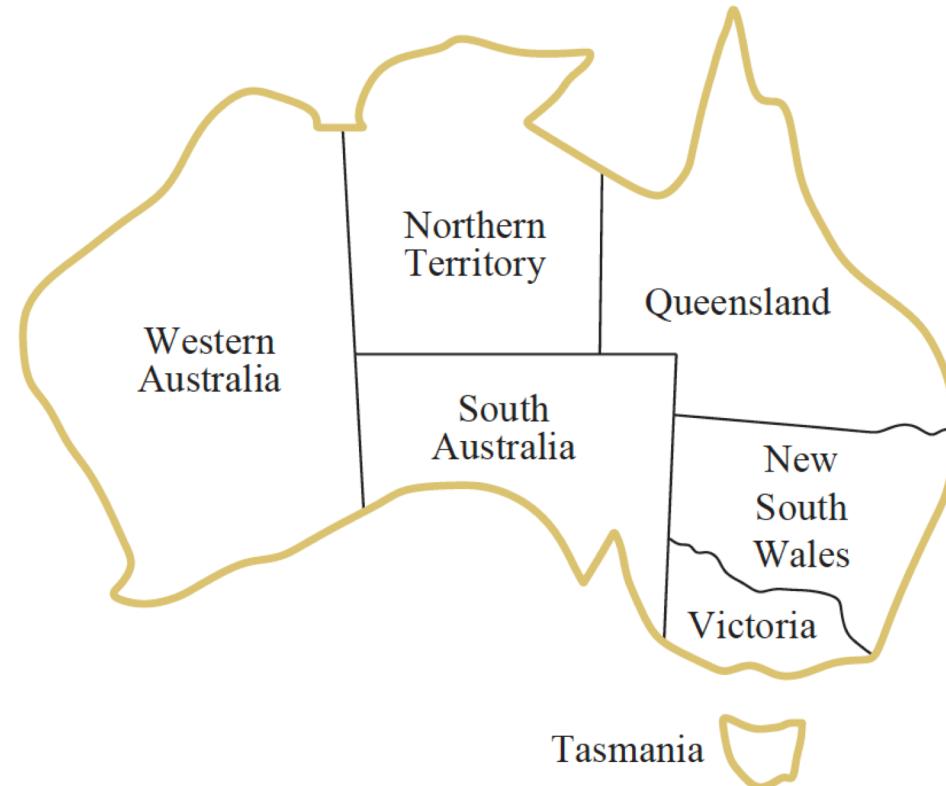
Constraint Satisfaction Problem (CSP)

- Ein Constraint Satisfaction Problem (CSP) ist festgelegt durch:
 - eine Menge von Variablen: X_1, X_2, \dots, X_n
 - Wertebereiche (value domains) für die Variablen: D_1, D_2, \dots, D_n
 - eine Menge von Beschränkungen (constraints) für die Variablen
- Ein CSP ist gelöst, wenn jeder Variable ein Wert zugewiesen ist, so dass sämtliche Beschränkungen erfüllt sind.

- Wertebereiche können endlich oder unendlich sein.
- Eine Beschränkung heißt **k-stellig**, falls k Variablen beteiligt sind.
2-stellige Beschränkungen heißen auch **binär**.
- CSP's lassen sich prinzipiell durch allgemeine Suche in einem Zustandsraum lösen.
Jedoch: Spezielle CSP-Lösungsverfahren sind effizienter.
Wesentliche Ansätze: **constraint propagation** und **Reihenfolgeheuristiken**.

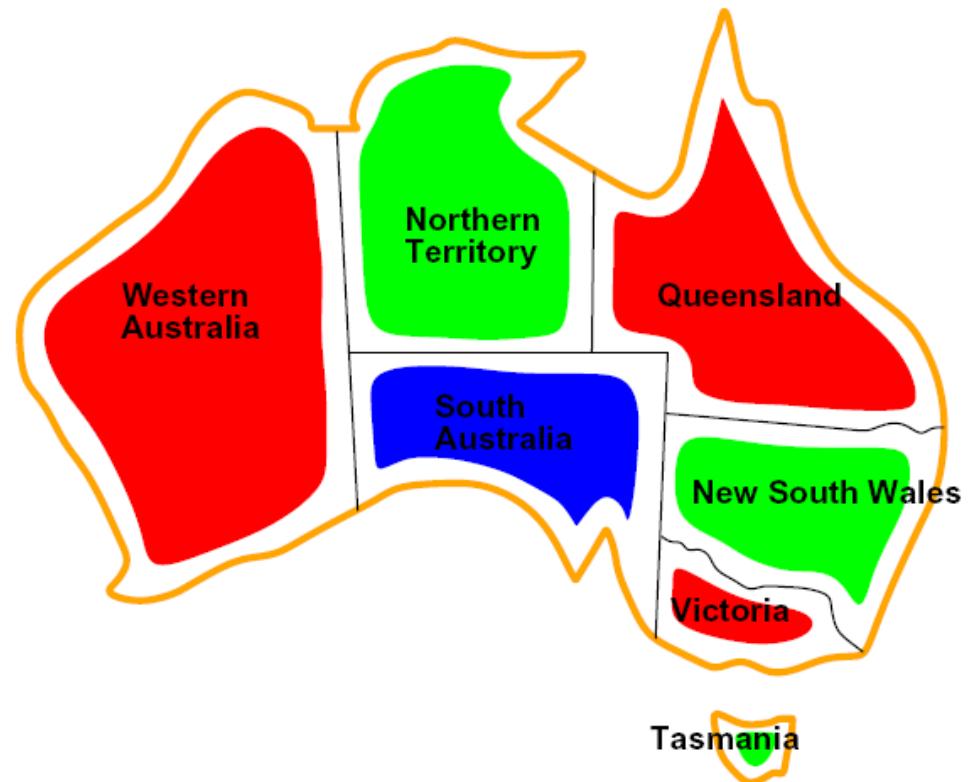
Beispiel: Färben von Landkarten

- **Variablen:**
WA, NT, SA, Q, NSW, V, T
- **Wertebereich:**
 $D = \{R, G, B\}$
für alle Variablen gleich
- **Beschränkung:**
alle benachbarte Regionen
haben unterschiedliche
Farben:
 - WA \neq NT
 - WA \neq SA
 - ...

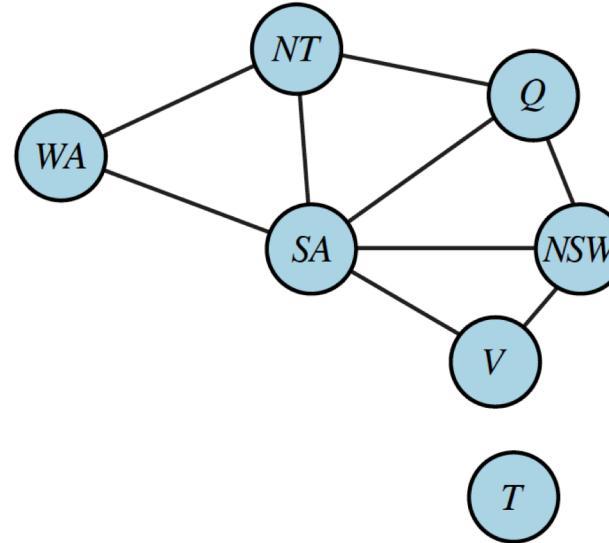
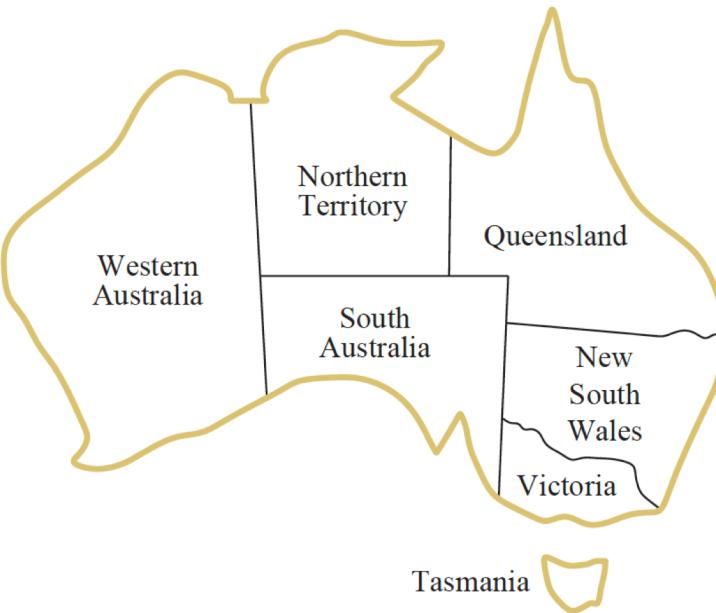


Lösung

- Mögliche Lösung:
 - WA = R
 - NT = G
 - SA = B
 - Q = R
 - NSW = G
 - V = R
 - T = G
- Lösung muss nicht eindeutig sein.



Constraint Graph



- Falls Beschränkungen binär sind, dann lassen sich die Beschränkungen als Graph darstellen
→ **Constraint Graph**.
- Knoten = Variablen
- Kanten = Beschränkungen
- **Hypergraph** bei k-stelligen Beschränkungen ($k > 2$).

Beispiel: Damenproblem

- **Variablen:**

x_{ij} für $1 \leq i \leq 8$ (Zeilen)
und ' a ' $\leq j \leq 'h'$ (Spalten)

- **Wertebereich:**

{0, 1} (nicht belegt, belegt)

- **Beschränkungen:**

Damen dürfen sich nicht gegenseitig
schlagen können

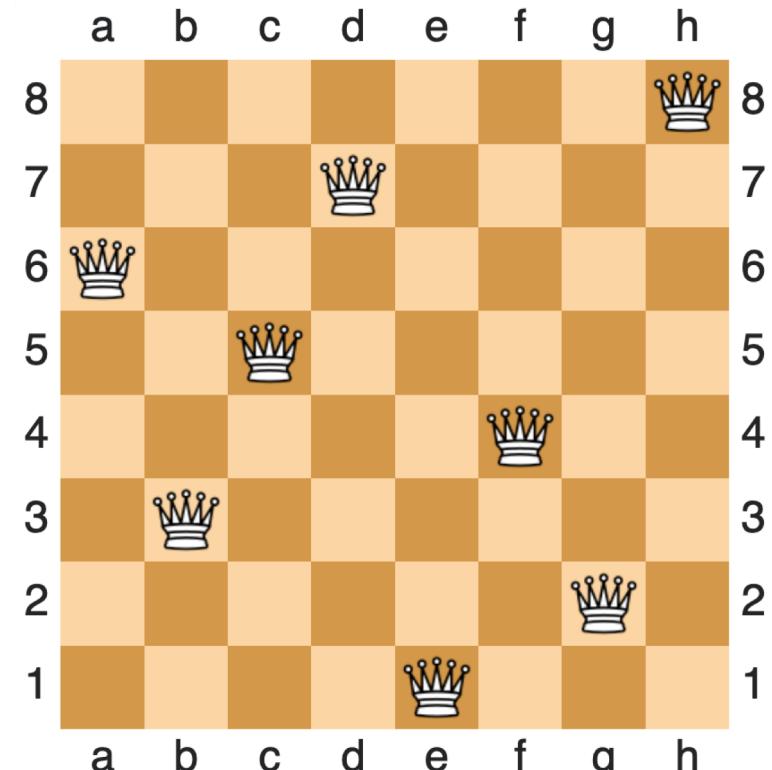
$$\forall i, j, k \ (x_{ij}, x_{ik}) \in \{(0,0), (0,1), (1,0)\}$$

$$\forall i, j, k \ (x_{ij}, x_{kj}) \in \{(0,0), (0,1), (1,0)\}$$

$$\forall i, j, k \ (x_{ij}, x_{i+k, j+k}) \in \{(0,0), (0,1), (1,0)\}$$

$$\forall i, j, k \ (x_{ij}, x_{i+k, j-k}) \in \{(0,0), (0,1), (1,0)\}$$

$$\sum_{i,j} x_{ij} = 8 \text{ (8-Damenproblem)}$$



<https://de.wikipedia.org/wiki/Damenproblem>

Beispiel: Sudoku

- **Variablen:**
 A_1, A_2, \dots, A_9
...
 I_1, I_2, \dots, I_9
- **Wertebereich:**
 $\{1, 2, \dots, 9\}$
- **Beschränkungen:**
 - $\text{Alldiff}(A_1, A_2, \dots, A_9)$
 - ...
 - $\text{Alldiff}(A_1, B_1, \dots, I_1)$
 - ...
 - $\text{Alldiff}(A_1, A_2, A_3, B_1, B_2, B_3, C_1, C_2, C_3)$
 - ...

	1	2	3	4	5	6	7	8	9
A			3		2		6		
B	9			3		5			1
C			1	8		6	4		
D			8	1		2	9		
E	7								8
F			6	7		8	2		
G			2	6		9	5		
H	8			2		3			9
I			5		1		3		

Kryptoanalytisches Beispiel

- **Variablen:**

F, T, O, W, U, R

X_1, X_2, X_3 (Hilfsvariablen für den Übertrag)

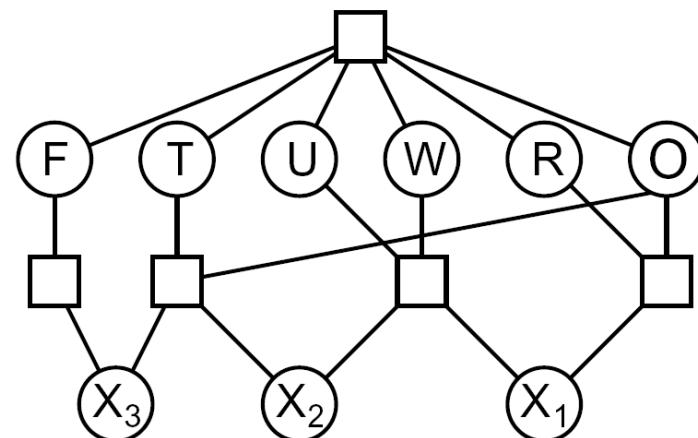
- **Wertebereich:**

{0, 1, 2, ..., 9}

- **Beschränkungen:**

- Alldiff(F, T, O, W, U, R)
- $O + O = R + 10 \cdot X_1$
- ...

$$\begin{array}{r} \text{T} \ \text{W} \ \text{O} \\ + \ \text{T} \ \text{W} \ \text{O} \\ \hline \text{F} \ \text{O} \ \text{U} \ \text{R} \end{array}$$



Hypergraph mit k-stelligen
Beschränkungen ($k \geq 2$).

Timetabling-Beispiel aus [Ertel, Grundkurs Künstliche Intelligenz]

Für die Raumaufteilung am Albert-Einstein-Gymnasium bei den mündlichen Abiturprüfungen soll der Hausmeister einen Plan erstellen. Er hat folgende Informationen: Die vier Lehrer Maier, Huber, Müller und Schmid prüfen die Fächer Deutsch, Englisch, Mathe und Physik in den aufsteigend nummerierten Räumen 1, 2, 3 und 4. Jeder Lehrer prüft genau ein Fach in genau einem Raum. Außerdem weiß er folgendes über die Lehrer und ihre Fächer:

1. Herr Maier prüft nie in Raum 4.
2. Herr Müller prüft immer Deutsch.
3. Herr Schmid und Herr Müller prüfen nicht in benachbarten Räumen.
4. Frau Huber prüft Mathematik.
5. Physik wird immer in Raum 4 geprüft.
6. Deutsch und Englisch werden nicht in Raum 1 geprüft.

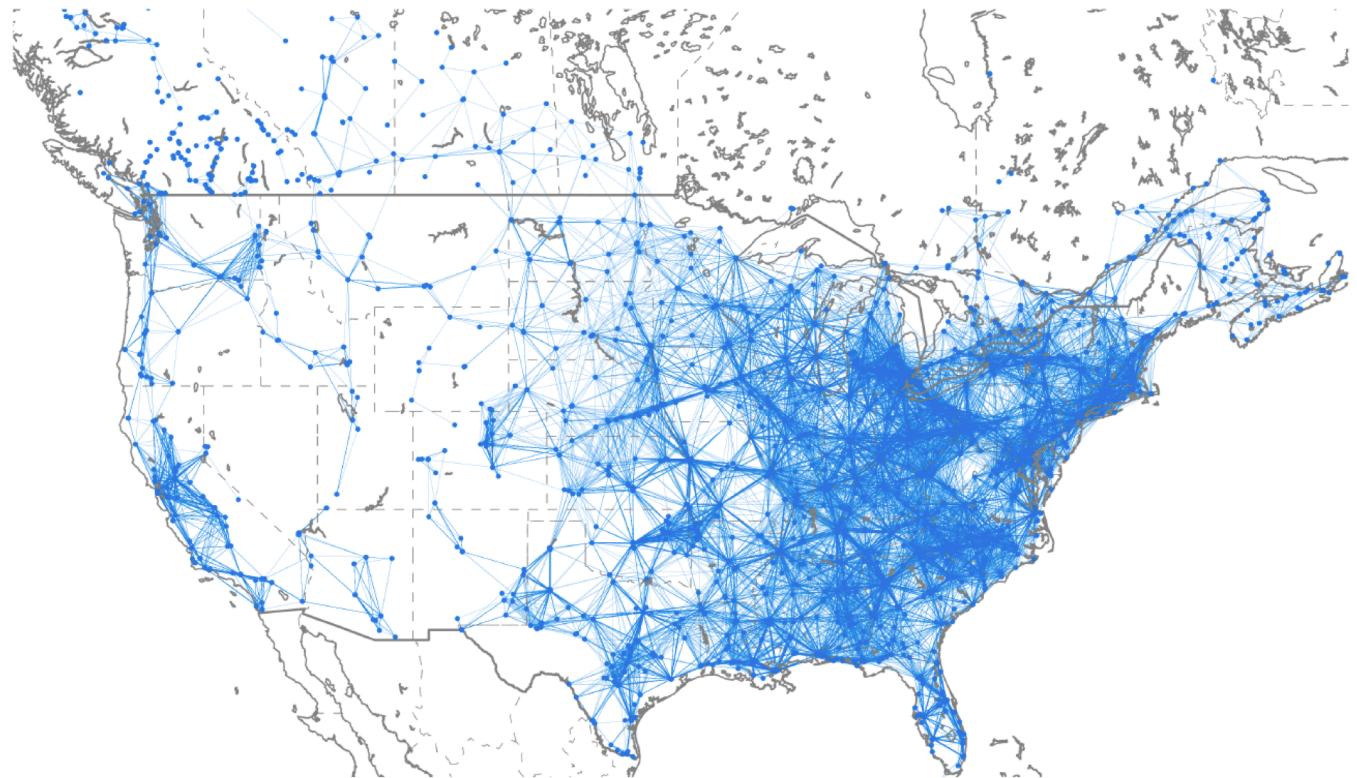
Wer prüft was in welchem Raum?

Anwendungen

- Timetabling (Veranstaltungen, Räume, Dozenten, Semester)
- Programmplanungen (Sport, Konferenzen, ...)
- Logistikprobleme
- Fertigungsplanung
- Konfigurationen (Hardware, Auto, ...)
- ...

Typisches größeres CSP-Problem

- Frequenzauktion der US Federal Communications Commission (FCC)
- Entscheidend im Auktionsdesign war die Konstruktion eines CSP-Solvers, um festzustellen, ob eine Frequenzzuweisung durchführbar ist.
- Dazu wurde ein Constraint-Graph definiert mit
 - 2990 Stationen und
 - 2 575 466 Beschränkungen.



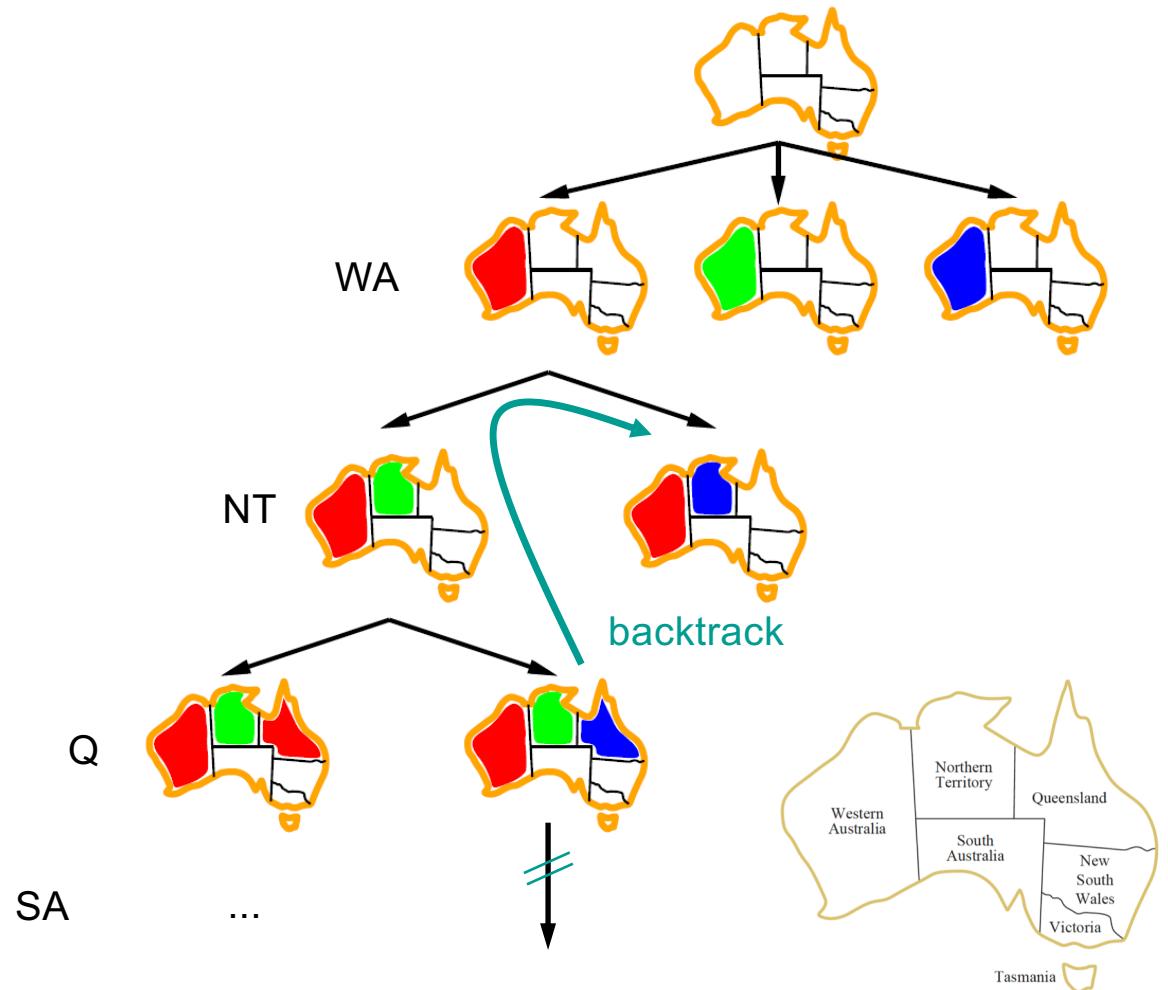
[Newman u.a., *Solving the Station Repacking Problem*, AAAI Conference on Artificial Intelligence 2016]

Inhalt

- Constraint Satisfaction Problems's
 - Problemstellung
 - Beispiele
- Backtracking
 - Algorithmus
 - Heuristik für Variablen- und Werte-Reihenfolge
 - Forward Checking
- Constraint Propagation (Beschränkungsweitergabe)
 - Kantenkonsistenz
 - AC-3-Algorithmus
- Problemstrukturen
 - Zusammenhangskomponenten
 - CSP in Baumstruktur
 - Schnittmengenkonditionierung
- Lokale Suche

Backtracking-Suche für CSP's

- Entspricht einer **Tiefensuche**,
- wobei in jeder Ebene (im Tiefensuchbaum) für genau eine Variable verschiedene Werte durchprobiert werden.
- Einsatz von **Heuristiken** für Reihenfolge von Variablen und Werte.
- Die Tiefensuche geht zurück, wenn einer Variablen keine weiteren Werte mehr zugewiesen werden können.
- Dazu werden die letzten Variablenzuweisungen zurückgenommen
→ **backtrack**



Backtracking-Algorithmus

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

```
function BACKTRACK(csp, assignment) returns a solution or failure
    if assignment is complete then return assignment
    var  $\leftarrow$  SELECT-UNASSIGNED-VARIABLE(csp, assignment)
    for each value in ORDER-DOMAIN-VALUES(csp, var, assignment) do
        if value is consistent with assignment then
            add {var = value} to assignment
            inferences  $\leftarrow$  INFERENCE(csp, var, assignment)
            if inferences  $\neq$  failure then
                add inferences to csp
                result  $\leftarrow$  BACKTRACK(csp, assignment)
                if result  $\neq$  failure then return result
                remove inferences from csp
            remove {var = value} from assignment
    return failure
```

Reihenfolge der Variablen

Reihenfolge der Werte

1. Weglassen,
2. Forward-Checking oder
3. AC-3-Algorithmus

Backtracking-Schritte

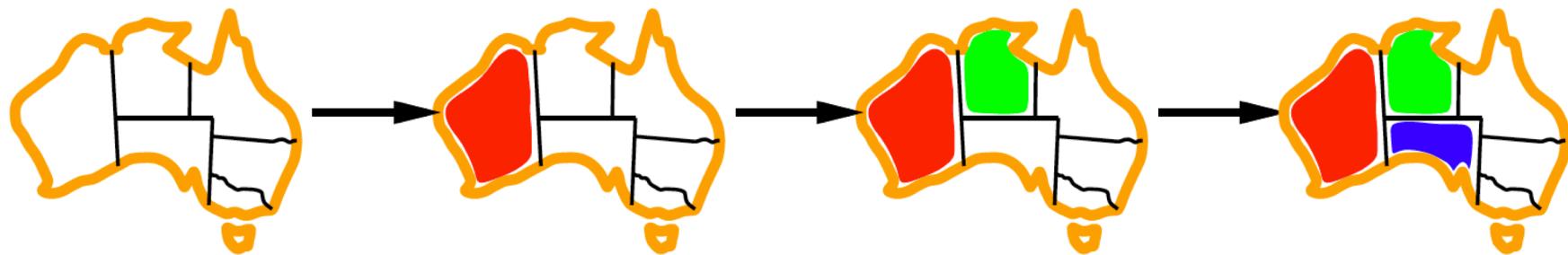
Russel u. Norvig. Künstliche Intelligenz – Ein moderner Ansatz.

Verbesserung der Effizienz

- **Reihenfolge der Variablen:**
Welche Variable wird zuerst betrachtet?
- **Reihenfolge der Werte:**
Welche Werte werden zuerst betrachtet?
- **Constraint propagation (Weitergabe der Beschränkungen)** um Wahlmöglichkeiten einzuschränken.
- Versuche Problemstruktur auszunutzen.
→ Beschneidung des Suchraums.

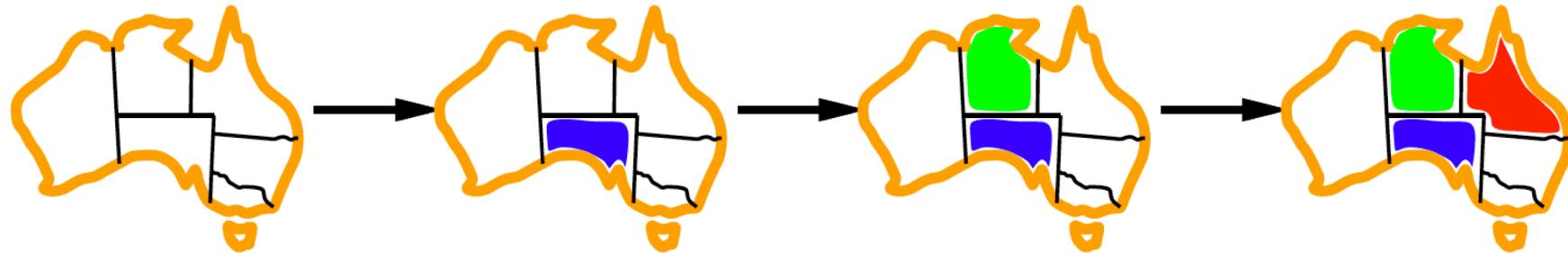
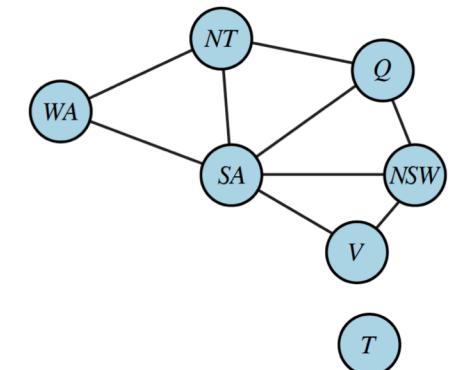
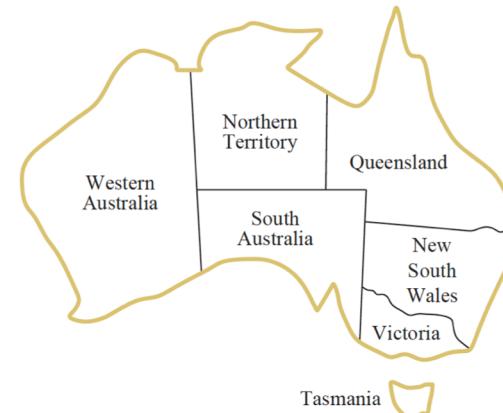
Reihenfolge der Variablen: Most constrained variable first

- Wähle die Variable mit den wenigsten erlaubten Werte
- → Reduziert den Verzweigungsfaktor.



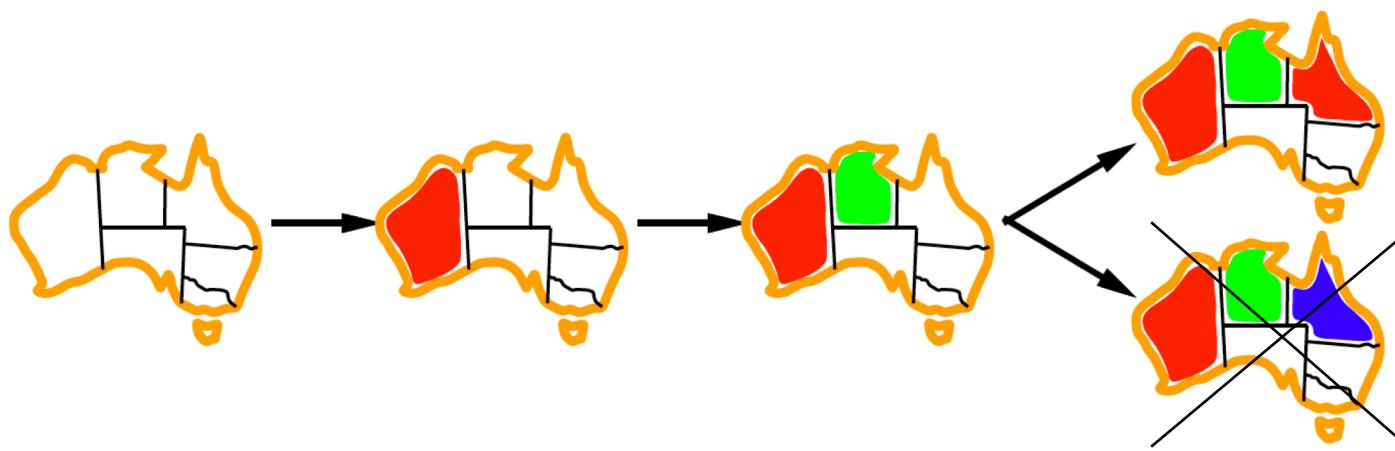
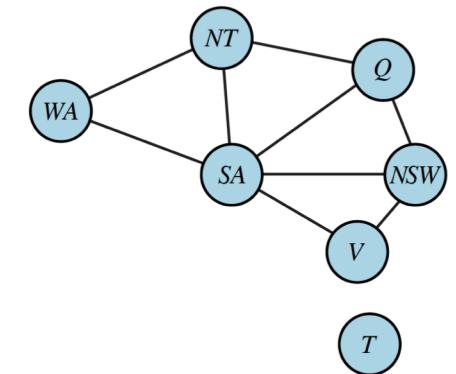
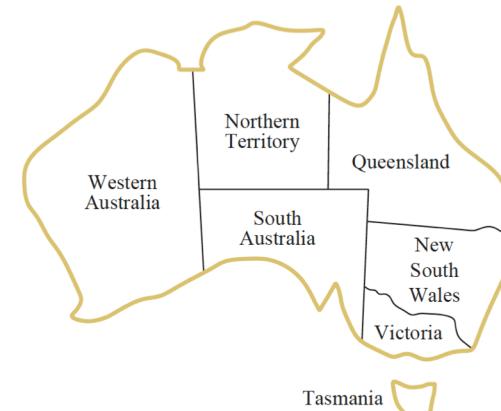
Reihenfolge der Variablen: Most constraining variable first

- Wähle die Variable mit den meisten Beschränkungen
- Das ist die Variable im Constraint-Graph mit dem **höchsten Knotengrad**.
- → Reduziert den Verzweigungsfaktor in den darauf folgenden Schritten.



Reihenfolge der Werte: Least constraining value first

- Bei einer gegebenen Variable wird derjenige Wert zuerst ausgewählt, der die wenigsten Werte der benachbarten Variablen im Constraint-Graphen ausschließt.

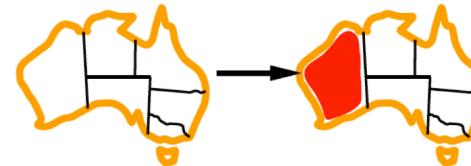
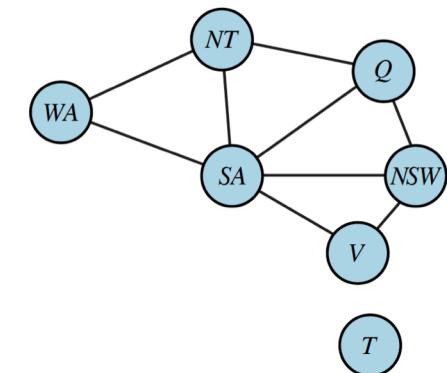


1 Wert für SA möglich

Kein Wert für SA möglich

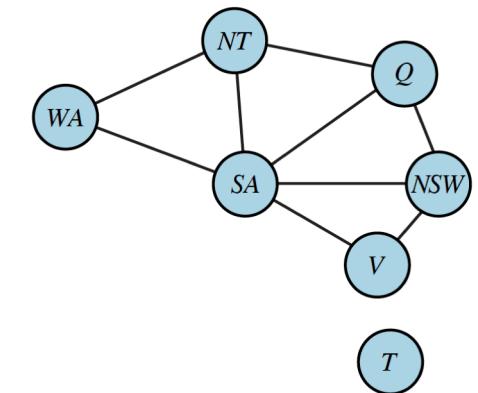
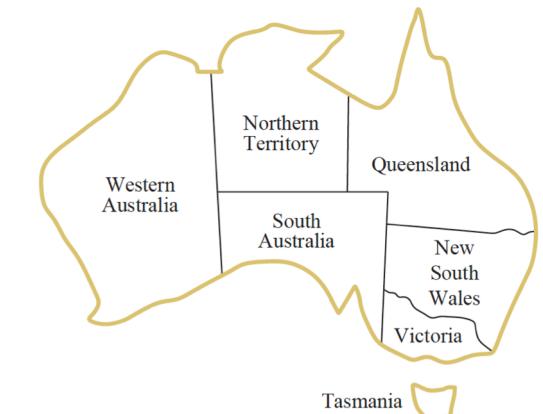
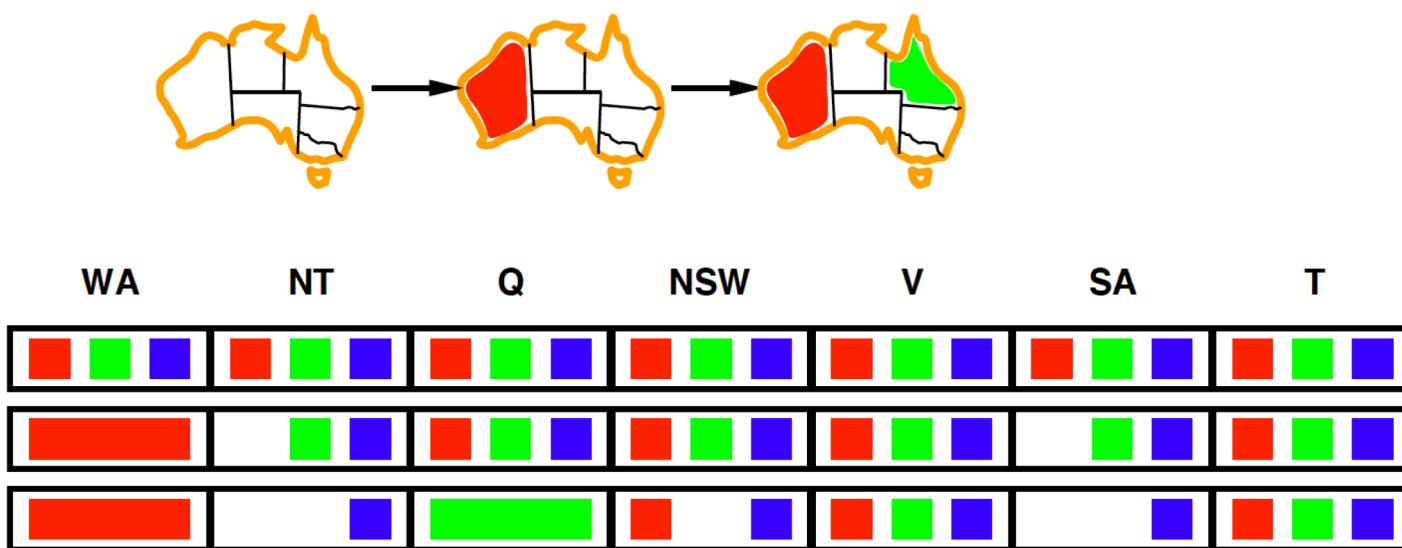
Forward Checking

- Immer wenn einer Variablen ein Wert zugewiesen wird, werden bei allen benachbarten Variablen im Constraint-Graphen alle illegalen Werte gelöscht.
- Wertebereich wird also reduziert!
- Falls alle Werte für eine Variable gelöscht sind, dann failure-Zustand und veranlasse backtracking.

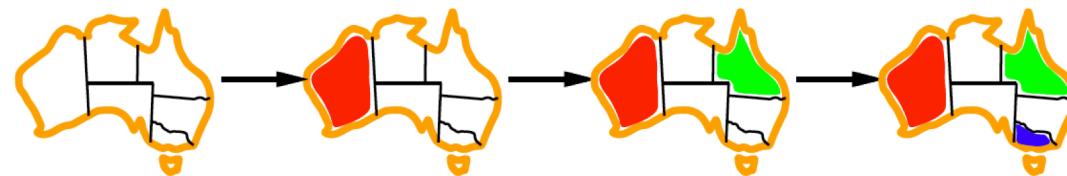


WA	NT	Q	NSW	V	SA	T
■ Red	■ Green ■ Blue	■ Red ■ Green ■ Blue				
■ Red		■ Red ■ Green ■ Blue	■ Red ■ Green ■ Blue	■ Red ■ Green ■ Blue		■ Red ■ Green ■ Blue

Beispiel für Forward Checking fortgesetzt

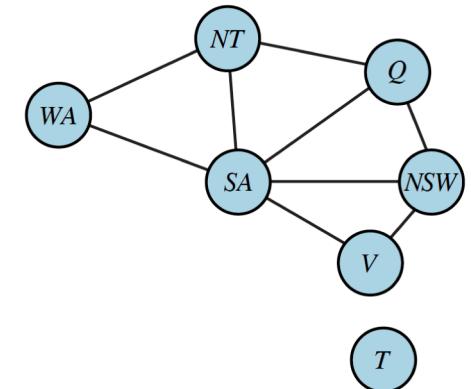


Beispiel für Forward Checking fortgesetzt



WA	NT	Q	NSW	V	SA	T
■ ■ ■ ■ ■	■ ■ ■ ■ ■	■ ■ ■ ■ ■	■ ■ ■ ■ ■	■ ■ ■ ■ ■	■ ■ ■ ■ ■	■ ■ ■ ■ ■
■ ■ ■ ■ ■		■ ■ ■ ■ ■	■ ■ ■ ■ ■	■ ■ ■ ■ ■	■ ■ ■ ■ ■	■ ■ ■ ■ ■
■ ■ ■ ■ ■			■ ■ ■ ■ ■	■ ■ ■ ■ ■	■ ■ ■ ■ ■	■ ■ ■ ■ ■
■ ■ ■ ■ ■				■ ■ ■ ■ ■		■ ■ ■ ■ ■

- SA hat keine erlaubten Werte mehr.
Daher failure und backtracking.

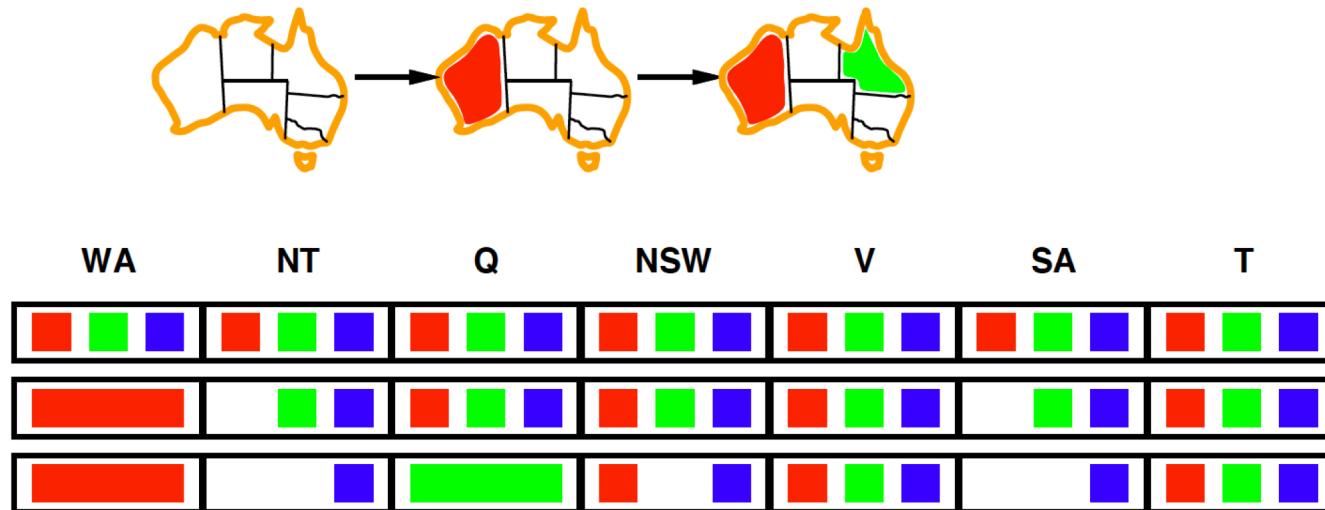


Inhalt

- Constraint Satisfaction Problems's
 - Problemstellung
 - Beispiele
- Backtracking
 - Algorithmus
 - Heuristik für Variablen- und Werte-Reihenfolge
 - Forward Checking
- Constraint Propagation (Beschränkungsweitergabe)
 - Kantenkonsistenz
 - AC-3-Algorithmus
- Problemstrukturen
 - Zusammenhangskomponenten
 - CSP in Baumstruktur
 - Schnittmengenkonditionierung
- Lokale Suche

Idee: Forward Checking mehrstufig durchführen

- Nachdem WA = R und Q = G zugewiesen wurde, sind für NT und SA nur noch B als einziger Wert möglich.
- Würde auf NT bzw. SA ein weiterer Forward-Checking-Schritt durchgeführt werden, dann ergäbe sich ein failure-Zustand.



Kantenkonsistenz

- Variable X mit Wertebereich D_X
- Variable Y mit Wertebereich D_Y
- X heißt **kantenkonsistent** zu Y, falls es zu jedem Wert in $x \in D_X$ ein Wert $y \in D_Y$, so dass (x,y) den binären Constraint zwischen X und Y erfüllt.



B ist kantenkonsistent zu A und umgekehrt



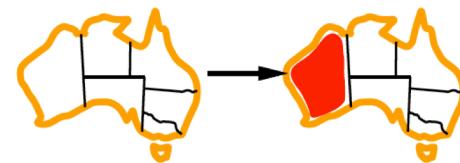
B ist kantenkonsistent zu A und umgekehrt



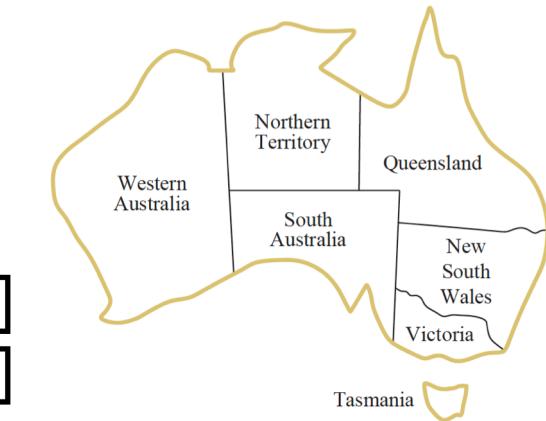
A ist kantenkonsistent zu B
aber nicht umgekehrt

Kantenkonsistenz eines CSPs

- Ein **CSP** heißt **kantenkonsistent**, falls für jede Kante (X, Y) im Constraint-Graphen X kantenkonsistent zu Y ist und umgekehrt.



WA	NT	Q	NSW	V	SA	T
[Red, Green, Blue]						
[Red]	[Green, Blue]	[Red, Green, Blue]	[Red, Green, Blue]	[Red, Green, Blue]	[Green, Blue]	[Red, Green, Blue]



Beide CSP's sind kantenkonsistent.

AC-3-Algorithmus zur Erzwingung der Konsistenz in einem CSP

function AC-3(*csp*) **returns** false if an inconsistency is found and true otherwise

queue \leftarrow a queue of arcs, initially all the arcs in *csp*

while *queue* is not empty **do**

$(X_i, X_j) \leftarrow \text{POP}(\text{queue})$

if REVISE(*csp*, X_i , X_j) **then**

if size of $D_i = 0$ **then return** false

for each X_k **in** $X_i.\text{NEIGHBORS} - \{X_j\}$ **do**

 add (X_k, X_i) to *queue*

return true

- Hier Preprocessing-Variante:
starte mit allen Kanten im CSP.
- AC-3 kann auch in der Backtracking-Funktion für INFERENCE eingesetzt werden.
Dann wird *queue* nur mit den Kanten von den Nachbarknoten der aktuellen Variable befüllt.

function REVISE(*csp*, X_i , X_j) **returns** true iff we revise the domain of X_i

revised \leftarrow false

for each x **in** D_i **do**

if no value y in D_j allows (x,y) to satisfy the constraint between X_i and X_j **then**

 delete x from D_i

revised \leftarrow true

return *revised*

Russel u. Norvig. Künstliche Intelligenz – Ein moderner Ansatz.

Eigenschaft von AC-3

- AC-3 benötigt $O(n^2d^3)$, wobei n die Anzahl der Variablen und d die maximale Anzahl von Werten aller Variablen ist.
 - Jede Kante kann in die queue maximal d-mal eingefügt werden
 - $O(n^2)$ viele Kante
 - REVISE benötigt $O(d^2)$
- AC-3 kann nicht alle Inkonsistenzen erkennen.
Mögliche aber teure Abhilfe: k-Konsistenz, wobei k Variablen in die Konsistenz-Prüfung einbezogen werden.

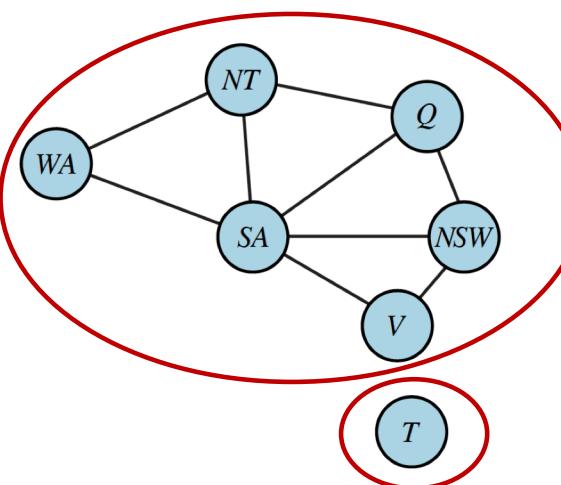
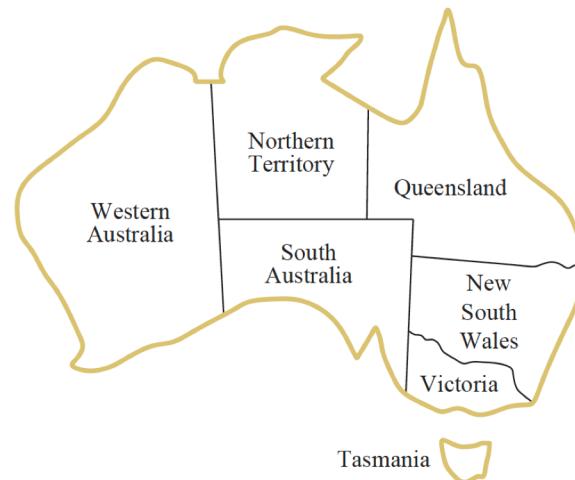


Inhalt

- Constraint Satisfaction Problems's
 - Problemstellung
 - Beispiele
- Backtracking
 - Algorithmus
 - Heuristik für Variablen- und Werte-Reihenfolge
 - Forward Checking
- Constraint Propagation (Beschränkungsweitergabe)
 - Kantenkonsistenz
 - AC-3-Algorithmus
- Problemstrukturen
 - Zusammenhangskomponenten
 - CSP in Baumstruktur
 - Schnittmengenkonditionierung
- Lokale Suche

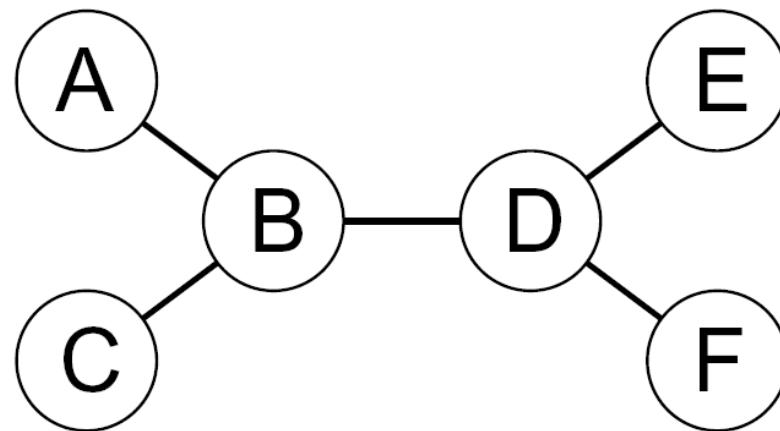
Zusammenhangskomponenten

- Hat der Constraint-Graphen mehrere Zusammenhangskomponenten, dann lässt sich jede Komponente einzeln betrachten.
- Drastische Reduktion des Suchraums:
 - Suchraumgröße im Ausgangsproblem: $O(d^n)$, wobei n die Anzahl der Variablen und d die maximale Anzahl von Werten aller Variablen ist.
 - Bei Zerlegung in Komponenten, wobei jede Komponente c Variablen hat, ergibt sich eine Suchraumgröße von:
 $O((n/c)^*d^c) = O(n)$. (c ist eine Konstante)



Beispiel lässt sich in
2 Zusammenhangs-
komponenten zerlegen.

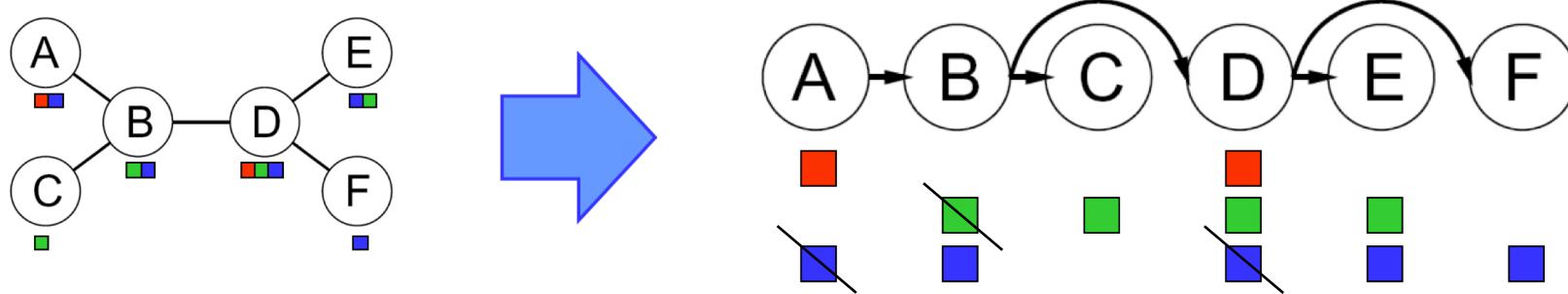
CSP mit Baumstruktur



- Ist der Constraint-Graph ein Baum, dann kann das CSP in $O(nd^2)$ gelöst werden.
- Baum = zusammenhängend + zyklenfrei

Algorithmus für CSP mit Baumstruktur

1. Wähle irgendein Knoten als Wurzel → **Wurzelbaum** (Baum mit gerichteten Kanten)
2. Ordne Variablen so, dass Elternknoten vor Kindknoten stehen: X_1, X_2, \dots, X_n .
→ **Topologische Sortierung des Wurzelbaums.**



3. Entferne Werte rückwärts:

```
for i = n to 2 do
    mache Elternknoten von  $X_i$  kantenkonsistent zu  $X_i$ ;
```

4. Weise Werte vorwärts zu:

```
for i = 1 to n do
    weise  $X_i$  einen Wert zu, der konsistent zu Elternknoten von  $X_i$  ist;
```

Schnittmengenkonditionierung

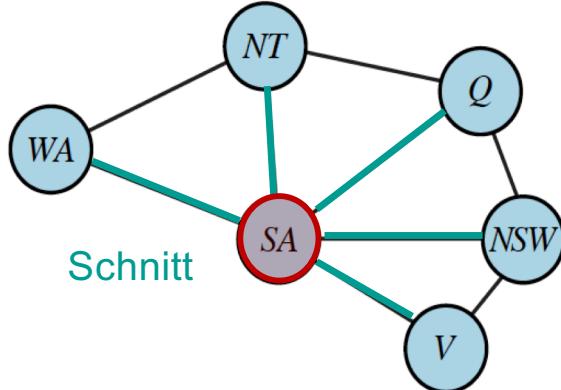
- **Konditionierung:**

Weise einer Variablen X eines CSPs einen Wert zu und mache Nachbarn kantenkonsistent.
Variable X wird so ausgewählt, dass $CSP \setminus X$ (Constraint-Graph ohne X) ein Baum ist!

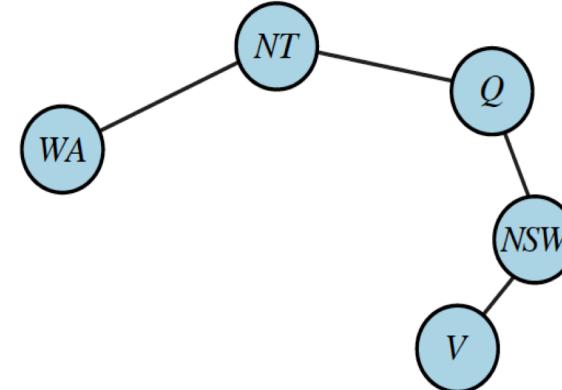
- **Schneide X aus CSP heraus und löse $CSP \setminus X$**

mit Algorithmus für CSP mit Baumstruktur.

- Die Kanten, die X mit Knoten aus $CSP \setminus X$ verbinden, wird in der Graphentheorie auch **Schnitt(menge)** (**cutset**) genannt.



Variable SA wird gewählt und die
Nachbarn kantenkonsistent gemacht



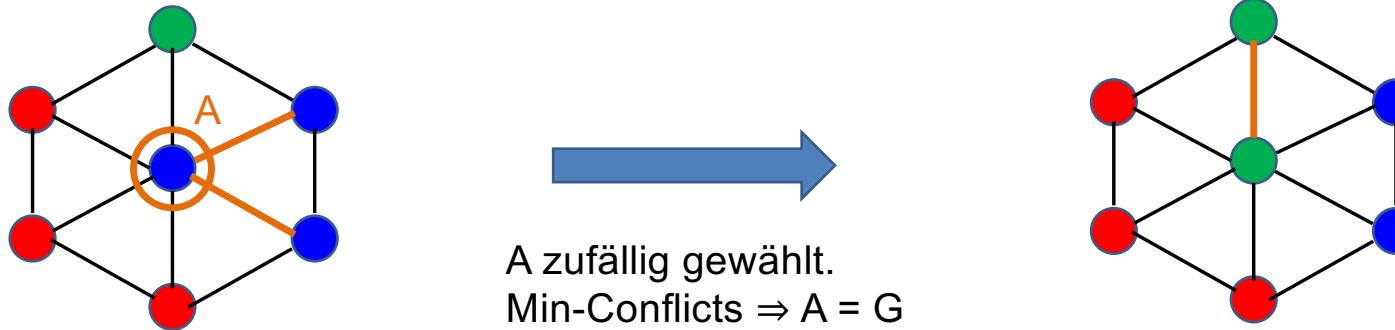
$CSP \setminus SA$ ist ein Baum

Inhalt

- Constraint Satisfaction Problems's
 - Problemstellung
 - Beispiele
- Backtracking
 - Algorithmus
 - Heuristik für Variablen- und Werte-Reihenfolge
 - Forward Checking
- Constraint Propagation (Beschränkungsweitergabe)
 - Kantenkonsistenz
 - AC-3-Algorithmus
- Problemstrukturen
 - Zusammenhangskomponenten
 - CSP in Baumstruktur
 - Schnittmengenkonditionierung
- Lokale Suche

Lokale Suche mit der Min-Conflicts-Heuristik

- Ausgangspunkt ist eine vollständige Zuweisung an die Variablen eines CSP's. Zuweisung muss nicht konsistent sein.
- Wähle eine **konfliktverursachende Variable zufällig** aus und wähle einen neuen Wert, so dass eine constraint-Verletzung beseitigt wird.
- **Min-Conflicts-Heuristik:** wähle den Wert so aus, dass die wenigsten Konflikte mit anderen Variablen verursacht werden.
- Wiederhole solange, bis Lösung gefunden oder Iterationsgrenzwert erreicht.



Beispiel: N-Damen-Problem

- Min-Conflicts ist für viele CSP's überraschend effektiv.
- N-Damen-Problem funktioniert besonders gut.
- Für $N = 10^6$ sind nach einer anfänglich zufälligen Zuweisung nur etwa 50 Schritte notwendig.
- Grund: Lösungen sind im Zustandsraum dicht verteilt.

