Mesterséges intelligencia előadássorozat

Az előadás diái az AIMA könyvre épülve (http://aima.cs.berkeley.edu) készültek a University of California, Berkeley mesterséges intelligencia kurzusának anyagainak felhasználásával (http://ai.berkeley.edu).

These slides are based on the AIMA book (http://aima.cs.berkeley.edu) and were adapted from the AI course material of University of California, Berkeley (http://ai.berkeley.edu).

Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Mesterséges Intelligencia és Rendszertervezés Tanszék

Mesterséges intelligencia

Kényszerkielégítési problémák I.

Előadó: Dr. Hullám Gábor
Előadás anyaga: Dr. Gézsi András, Dr. Hullám Gábor

Mire való a keresés?

- Feltételezéseink a világról: egyetlen ágens, determinisztikus cselekvések, teljesen megfigyelhető állapotok, diszkrét állapottér
- Tervezés: cselekvések egy sorozata
  - Mit akarunk elérni: a célhoz vezető *útvonal*at
  - Az útvonalaknak eltérő költségeik, mélységeik lehetnek
  - A heurisztikák probléma-specifikus segítséget adhatnak
- Felfedező módszerek: változókhoz érték hozzárendelése
  - Az elérendő cél a fontos, nem az ahhoz vezető út
  - Minden útvonal azonos hosszúságú (bizonyos megfogalmazásokban)
- Kényszerkielégítési problémák: speciális felfedező módszerek

Kényszerkielégítési (korlátkielégítési) problémák Constraint Satisfaction Problems (CSP):

Állapot: Az állapot a leíró változók és a hozzájuk rendelt értékek által definiált.

Az x_k változók egy-egy D_k értéktartományból (halmazból) veszik fel értékeiket.

Célállapotteszt:

1. Az összes állapotot leíró változóhoz rendeltünk megengedett értéket
2. Az összes kényszert kielégítettük

Kényszerkielégítési (korlátkielégítési) problémák
Constraint Satisfaction Problems (CSP):

Kényszerkielégítési probléma definíciója = $\{\mathcal{X},\mathcal{D},\mathcal{C}\}$

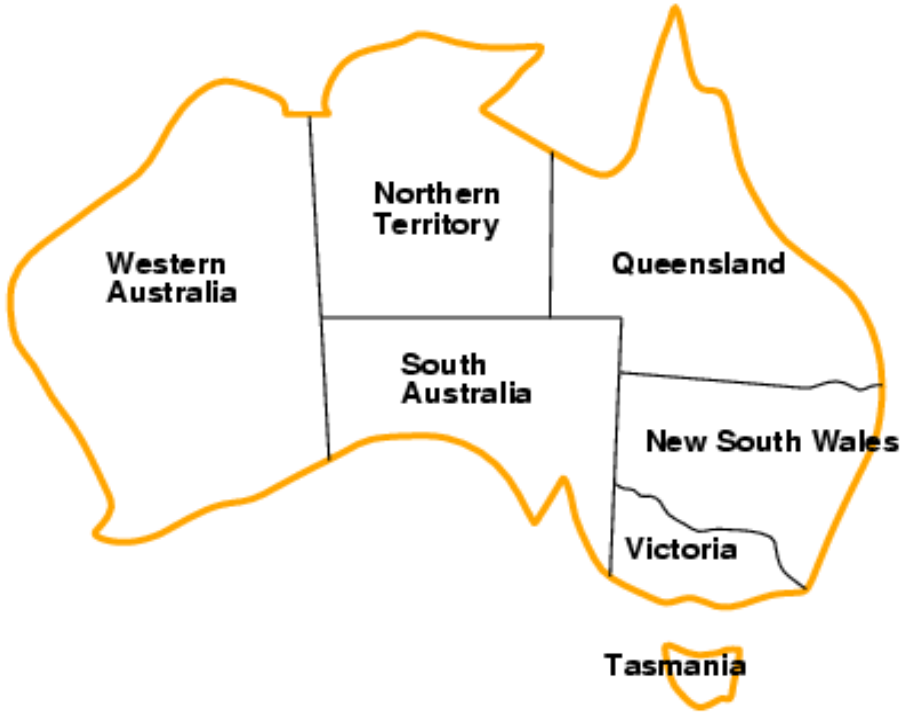
- $\mathcal{X}$ változók egy halmaza: $\{X_1,X_2,\ldots,X_n\}$
- $\mathcal{D}$ tartományok egy halmaza: $\{D_1,D_2,\ldots,D_n\}$
  $D_i = \{v_1,v_2,\ldots,v_k\}$ az X_i változóhoz.
- $\mathcal{C}$ kényszerek egy halmaza:

  A változók által felvehető értékek megengedett kombinációi.
  C_j = ‹hatáskör, reláció›, ahol

  hatáskör = változók rendezett halmaza
  (amelyekre a kényszer vonatkozik)

  reláció = az értékek, amelyeket a változók felvehetnek
  (explicit felsorolás v. implicit függvény)

1. példa: Térképszínezés



Változók: $\mathcal{X} = \{WA, NT, Q, NSW, V, SA, T\}$

Értéktartományok: $D_i = \{$vörös, zöld, kék$\}$

Kényszerek: szomszédos területek színe legyen eltérő
$C = \{SA \neq WA, SA \neq NT, SA \neq Q, \ldots, NSW \neq V\}$

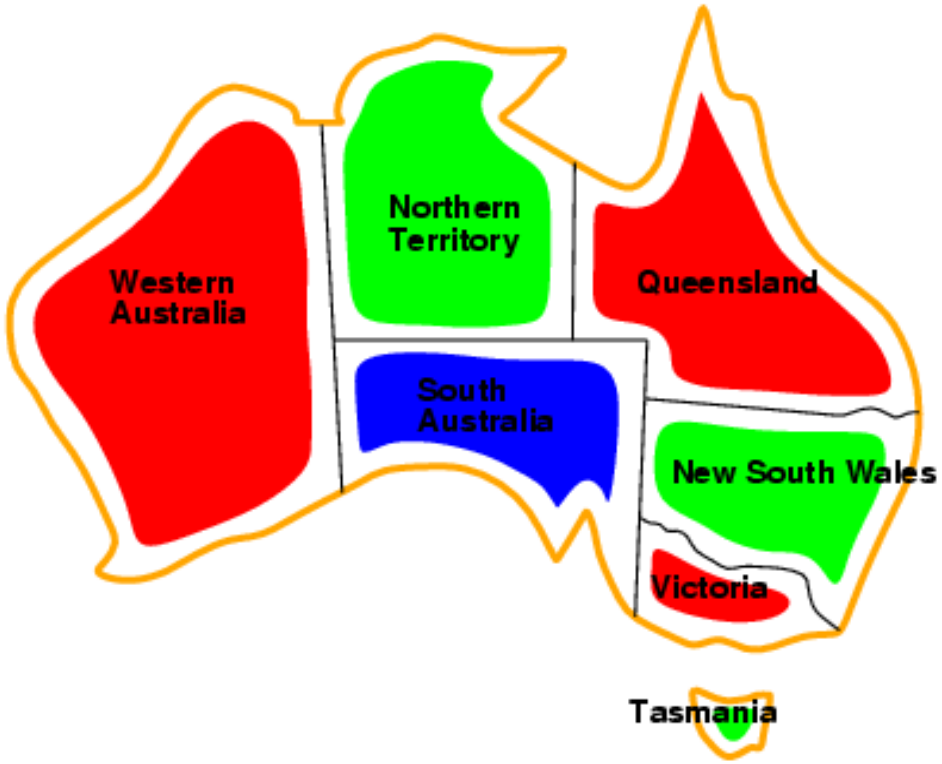
pl. SA ≠ WA jelentése: (SA, WA) értékét csakis a

{(vörös, zöld), (vörös, kék), (zöld, vörös), (zöld, kék), (kék, vörös), (kék, zöld)}

halmazból vehet fel.

Lehetséges-e?

1. példa: Térképszínezés



- **Megoldás**: **teljes** és **konzisztens** változó-érték **hozzárendelés**
- pl. WA = vörös, NT = zöld, Q = vörös, NSW = zöld, V = vörös, SA = kék, T = zöld

  (T lehet akármi, mert nem határos senkivel

  - de több megoldás is van)

2. példa: N-királynő probléma

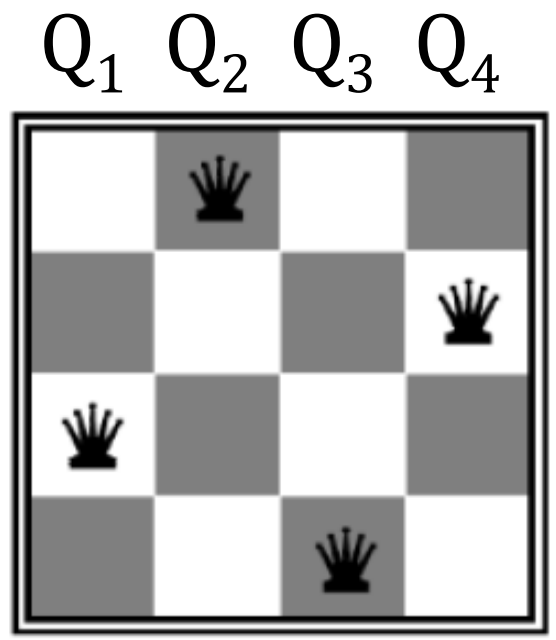
Változók: X = $\{Q_1,Q_2,\ldots,Q_N\}$

Értéktartományok: D_j = {1,2,…,N}
(sor száma az *i*. királynő esetén)

Kényszerek:

Implicit: ∀ i,j (Q_i,Q_j) nem „támadják egymást” –
nincsenek egy sorban, illetve átlóban egymással

Explicit:
(Q_1,Q_2) ∈ {(1,3), (1,4), …}
(Q_1,Q_3) ∈ {(1,2), (1,4), …}
(Q_1,Q_4) ∈ {(1,2), (1,3), …}
…



3. példa: Betűrejtvény

Változók: X = {*F, T, U, W, R, O,* X_1, X_2, X_3}

Értéktartományok:

D_i = {0,1,2,3,4,5,6,7,8,9}
{*F, T, U, W, R, O*} esetén

D_i = {0,1}
{X_1, X_2, X_3} esetén

Kényszerek:

MindKülönböző(*F, T, U, W, R, O*)

$O + O = R + 10 \cdot X_1$

$W + W + X_1 = U + 10 \cdot X_2$

…

4. példa: Sudoku

Változók: $\mathcal{X} = \{A1, A2, \ldots, A9, \ldots, I1, I2, \ldots, I9\}$

Értéktartományok:

$D_i = \{1,2,3,4,5,6,7,8,9\}$

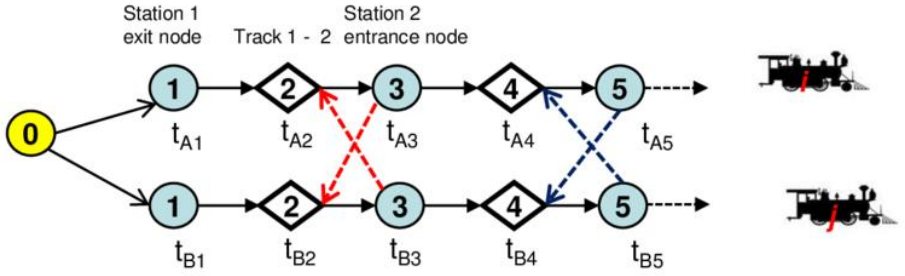
Kényszerek:

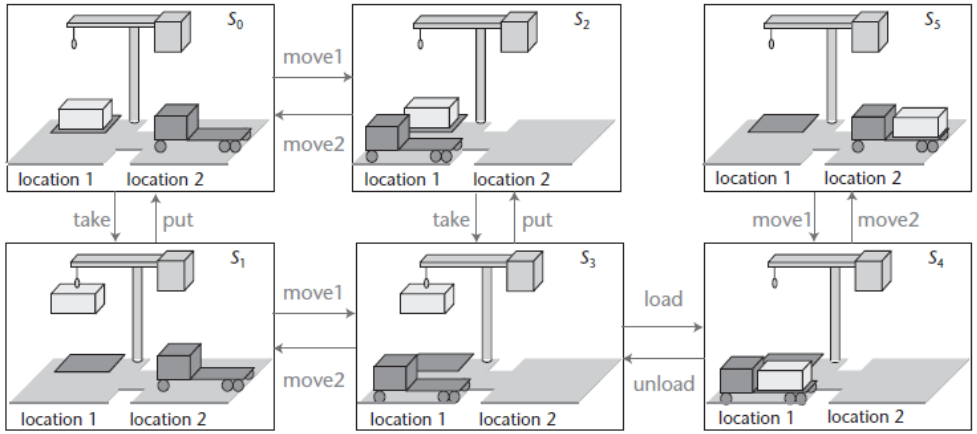
MindKül(A1,A2,A3,A4,A5,A6,A7,A8,A9)
MindKül(B1,B2,B3,B4,B5,B6,B7,B8,B9)
…
MindKül(A1,B1,C1,D1,E1,F1,G1,H1,I1)
MindKül(A2,B2,C2,D2,E2,F2,G2,H2,I2)
…
MindKül(A1,A2,A3,B1,B2,B3,C1,C2,C3)
MindKül(A4,A5,A6,B4,B5,B6,C4,C5,C6)
…

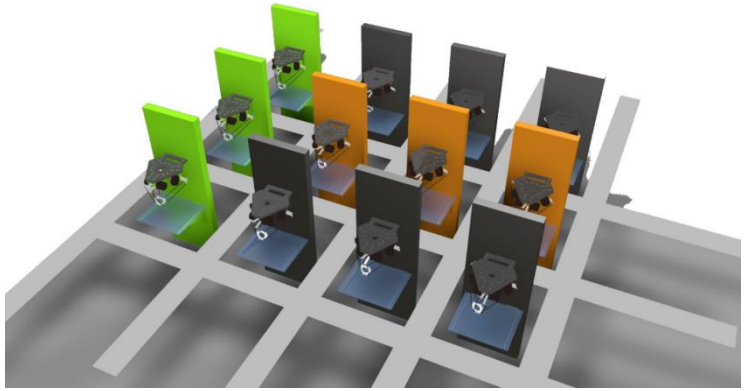
	1	2	3	4	5	6	7	8	9
A			3		2		6		
B	9			3		5			1
C			1	8		6	4		
D			8	1		2	9		
E	7								8
F			6	7		8	2		
G			2	6		9	5		
H	8			2		3			9
I			5		1		3		

Gyakorlati kényszerkielégítési problémák

- **Hozzárendelési problémák**, pl. ki, mit, hol tanítson
- **Menetrendi problémák**, pl. vasút
- **Szállításütemezés**
- **Gyári ütemezés**
- **Áramkörtervezés**







CSP problémák típusai

Változók alapján:

- **Diszkrét** változók
  - **véges** értéktartományok:
    - pl. logikai típusú CSP, Boole-féle kielégítési vizsgálatok (NP-teljes)
  - **végtelen** értéktartományok:
    - egész számok, karaktersorozatok, stb.
    - pl. ütemezési problémák, változók a munkaszakaszok kezdete/vége
- **Folytonos** változók
  - fizikai állapotváltozók,
  - pl. Hubble-űrteleszkóp megfigyeléseinek kezdő/vég-időpontja

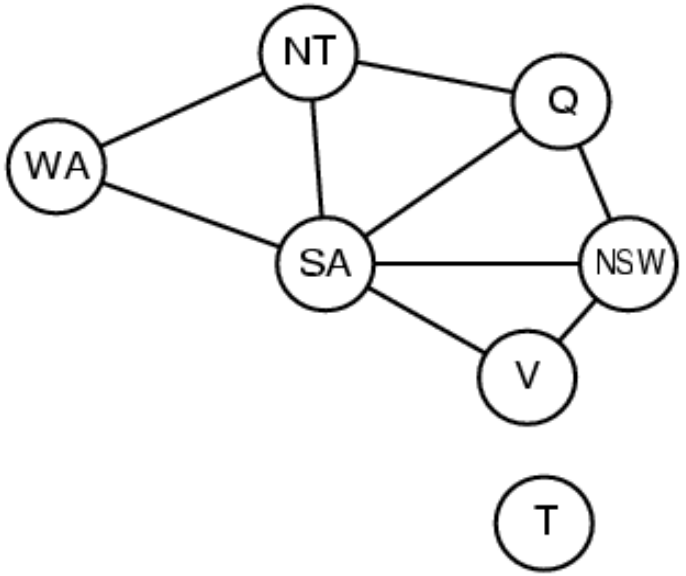
CSP problémák típusai

Kényszerek alapján:

- **Unáris** kényszer: egyetlen egy változóra vonatkozik, pl. SA ≠ zöld
- **Bináris** kényszer: két változó viszonyára vonatkozik, pl. SA ≠ WA
- **Magasabb-rendű** kényszer: 3 vagy több változó viszonyára vonatkozik,
  (pl. oszloponkénti változó korlátok betűrejtvényekben,
  Sudoku kényszerei)

- **Preferecia**-kényszer:
  Egy érték jobb(an preferált), mint egy másik, pl. a vörös jobb, mint a zöld
  Gyakran a változó-hozzárendelés költségeként reprezentálják
  => korlátozott optimalizációs problémák

Kényszergráfok

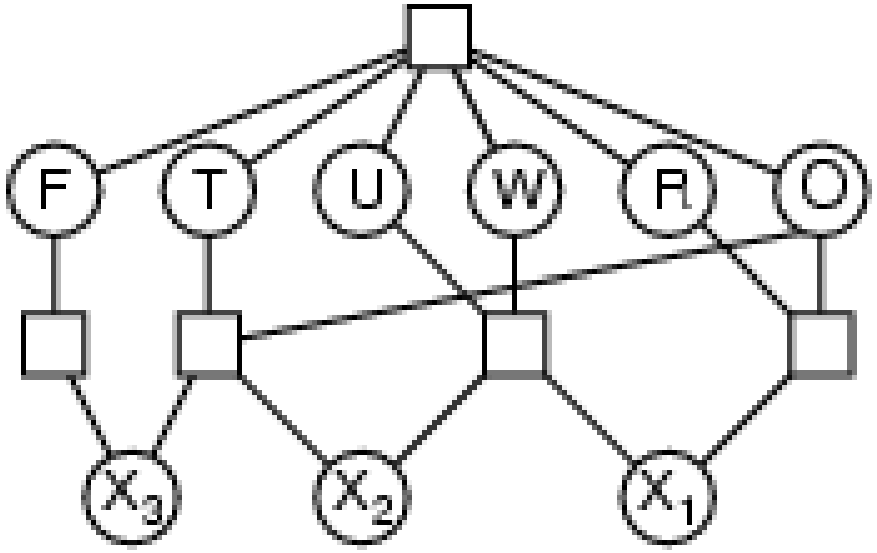


- **kényszergráf**: csomópontjai a változók
  és élei a bináris kényszerek

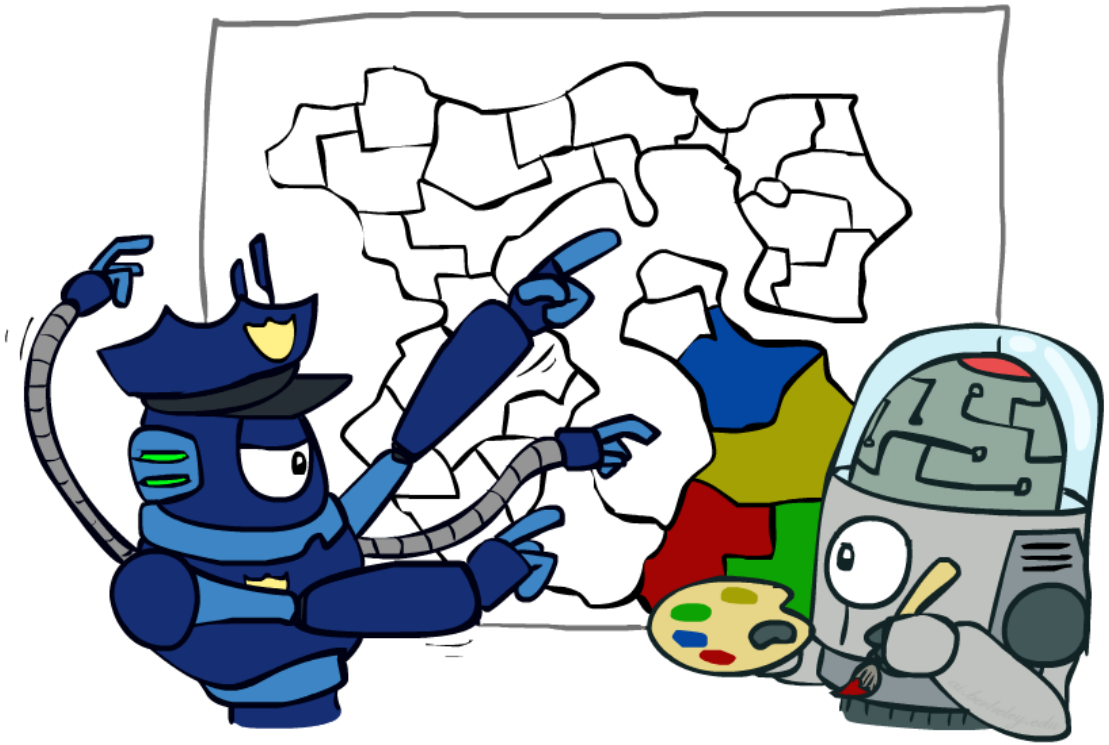
- Itt csak **bináris CSP**-k:
  egy-egy kényszer (legfeljebb) 2 változót köt össze

Másik példa: betűrejtvény
Kényszer-hipergráf

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



Kényszerkielégítési problémák megoldása



A probléma megfogalmazása

Kezdeti állapot: összes változó-hozzárendelés üres { }

Operátor: értéket hozzárendelni egy még nem lekötött változóhoz úgy, hogy az eddigi hozzárendelésekkel ne ütközzön

→ kudarc, ha nincs megengedett hozzárendelés

Célállapotteszt: ha az aktuális hozzárendelés teljes és mindegyik kényszer teljesül

A probléma megfogalmazása

Keresési fa

n db változó esetén minden megoldás *n* mélységben fekszik

→ mi történik, ha a **mélységi keresést** használjuk?

Elágazások száma *L* mélységben (*L=0,1,2,…*), ha minden változó *d* számú értéket vehet fel (ez a változó tartományának mérete)

b = (n – *L*) *d*, (mert *L* változó már értéket kapott)

azaz $n! \cdot d^n$ levélcsomópont

(miközben d^n lehetséges hozzárendelés van)

(pl. 8 betűs betűrejtvény, *n*=8, *d*=10, levelek száma $4 \cdot 10^{12}$)

Keresés

Változó-hozzárendelés **kommutatív**, azaz például

(WA = vörös) majd (NT = zöld)

ugyanaz, mint (NT = zöld) majd (WA = vörös)

Egy-egy csomópontban csakis egyetlen egy változó hozzárendelése történhet meg, így:

→ $b = d$ és a fának d^n levele van

Keresés

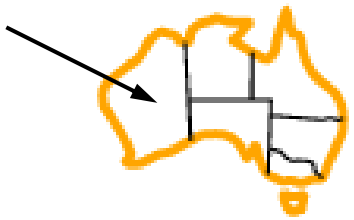
- Változó-hozzárendelés **kommutatív**, azaz például
- Egy-egy csomópontban csakis egyetlen egy változó hozzárendelése történhet meg, így:

  → $b = d$ és a fának d^n levele van

- Alapvető nem informált algoritmus (keresés) CSP problémák megoldására:
  visszalépéses keresés (backtracking search), azaz

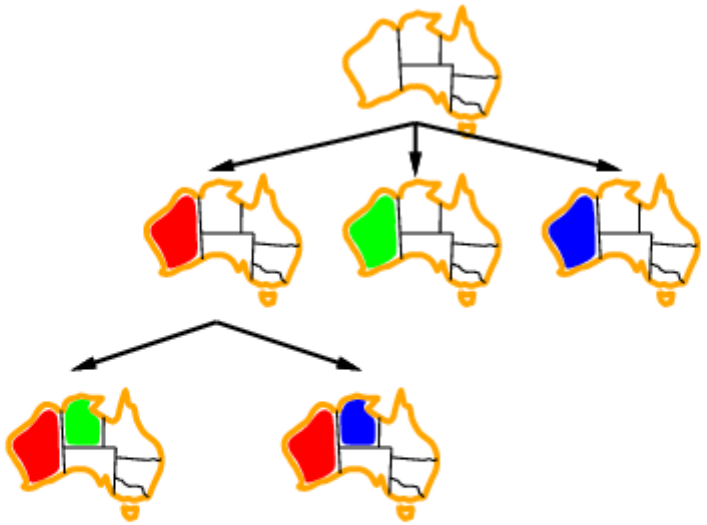
  mélységi keresés
  - minden szinten egyetlen egy változó-hozzárendeléssel
  - ha sérül valamelyik kényszer, visszalép
    (egyszer sérült kényszer mélyebben nem jöhet helyre)

Visszalépéses keresés

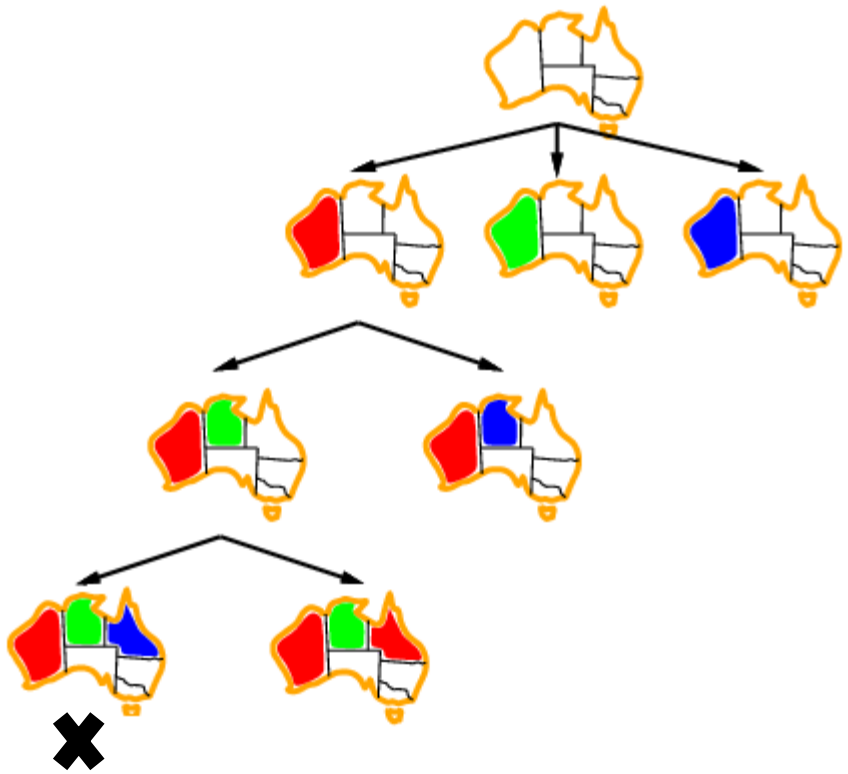


Visszalépéses keresés

Visszalépéses keresés



Visszalépéses keresés



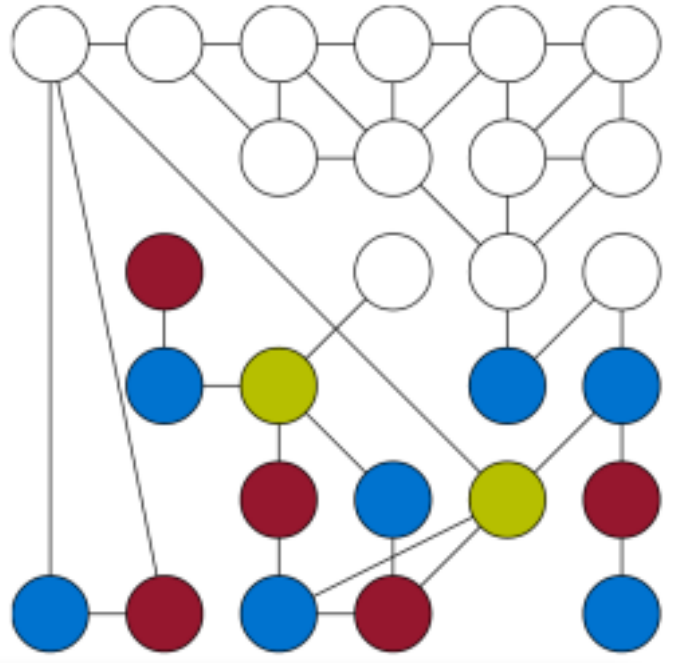
Visszalépéses keresés

```
function VISSZALÉPÉSÉS-KERESÉS(csp) returns egy megoldást, vagy meghiúsul
    return REKURZÍV-VISSZALÉPÉSES({},csp)

function REKURZÍV-VISSZALÉPÉSES(hozzárendelések, csp) returns egy megoldást, vagy meghiúsul
    if hozzárendelések teljes then return hozzárendelések
    var ← HOZZÁRENDELETLEN-VÁLTOZÓ-KIVÁLASZTÁSA(VÁLTOZÓK[csp],hozzárendelések,csp)
    for each érték in TARTOMÁNY-ÉRTÉKEK-SORRENDEZÉSE(var, hozzárendelések, csp) do
        if érték konzisztens a hozzárendelések-kel KÉNYSZEROK(csp) szerint then
            hozzáad {var = érték } hozzárendelések-hez
            eredmény ← REKURZÍV-VISSZALÉPÉSES(hozzárendelések, csp)
            if eredmény ≠ meghiúsulás then return eredmény
```

Néhány gyengeség

- Tekintsük az itt látható részleges hozzárendelést
- A csomópontokat lentről felfelé, balról jobbra járjuk be
- Vajon a jelenlegi hozzárendelés kudarcra van ítélve?
- A naiv visszalépéses keresés túl későn veszi észre a problémát



A visszalépéses keresés hatékonyságának növelése

- Általános, tárgyterület-független heurisztikákkal növelhetjük a hatékonyságot
- Szűrés:
  - Ki tudjuk korán szűrni a kudarcra ítélt megoldásokat?
- Sorrendezés:
  - Változók sorrendezése: Melyik változóhoz rendeljünk értéket a következő lépésben?
  - Értékek sorrendezése: Melyik értéket rendeljük hozzá először a változóhoz?
- Struktúra:
  - Ki tudjuk használni a probléma struktúráját?