Alkalmazott mesterséges intelligencia (AMI)

http://www.mit.bme.hu/oktatas/targyak/vimibb01

11. ea. (2023. ősz)

Kényszerkielégítési problémák
(Constraint Satisfaction Problem, CSP)

http://http://mialmanach.mit.bme.hu/aima/ch05 5. fejezet

Előadó: Pataki Béla

a fóliák
Dobrowiecki Tadeusz és
Hullám Gábor anyagainak
felhasználásával készültek

https://www.esrcheck.com/2023/06/05/artificial-intelligence-ai-experts-sign-statement-on-ai-risk/

BME I.E. 414, 463-26-79
pataki@mit.bme.hu,
http://www.mit.bme.hu/general/staff/pataki

(A keresés témaköréhez tartozik)

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

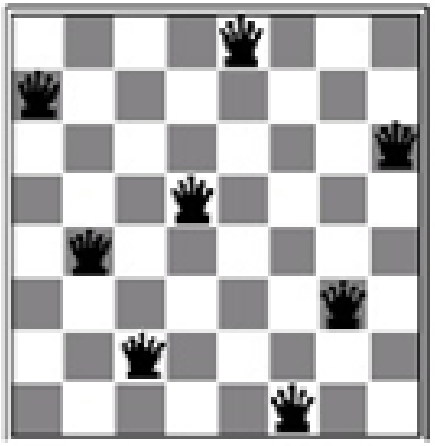
Állapot:

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 *számára megengedett* értéket
2. Az adott korlátok teljes halmazát kielégítettük

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



Térképszínezés


```
  T W O
+ T W O
-------
F O U R
```

Órarend készítés:

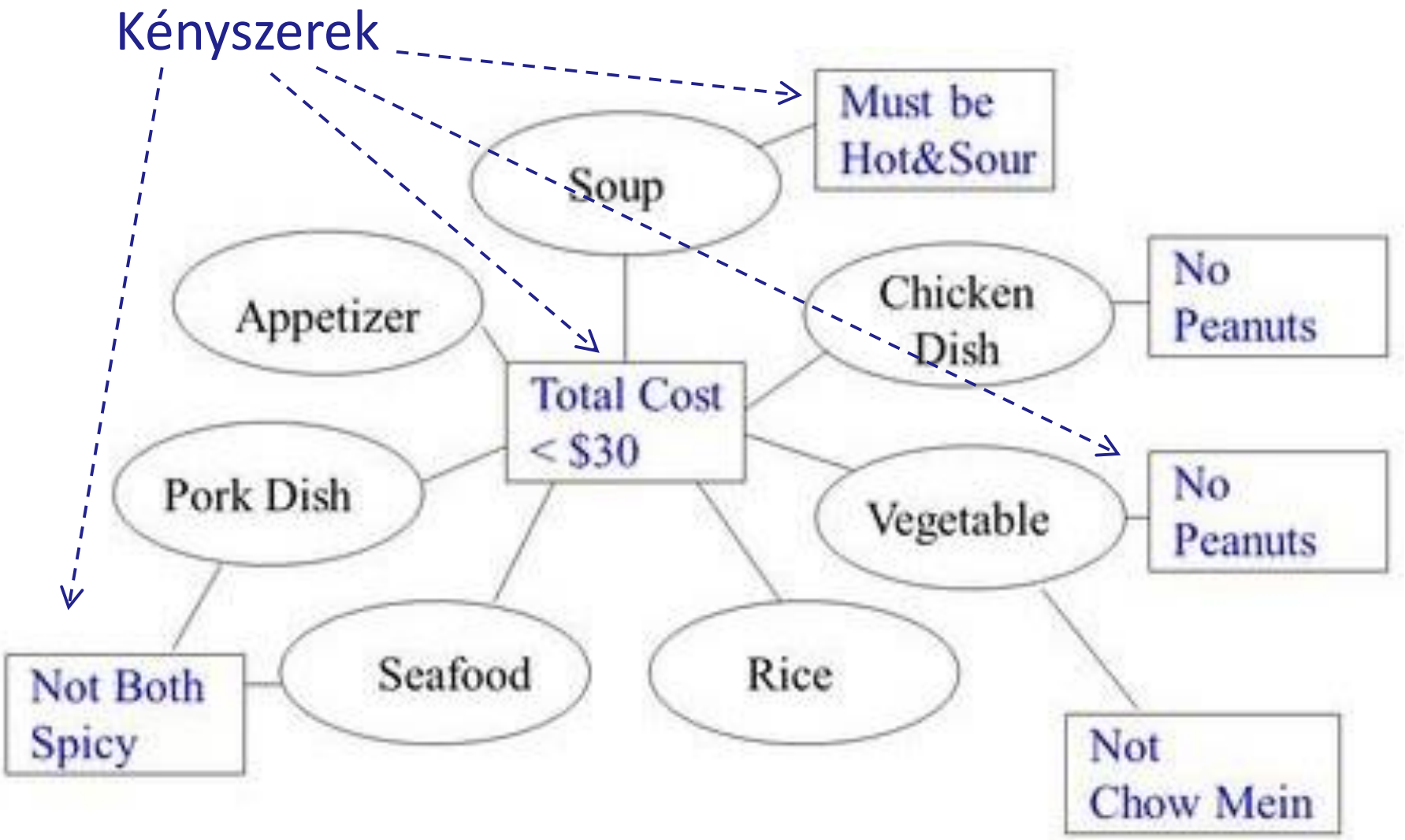
Hétfő 10-12, Pataki Béla, Alk. MI-Bprof3szem, IE220

…

1. *egyszerre egy teremben csak egy évfolyam*
2. *egyszerre egy teremben csak egy oktató,*
3. *senki se lehet egyszerre két helyen*
4. *egy évfolyam egyszerre csak egy helyen lehet (?)*

- stb.

Menü-összeállítás az étlap alapján



- **Hozzárendelési problémák**, pl. melyik árut, melyik raktárba, milyen mennyiségben
- **Menetrendi problémák**, pl. az egyetemi órarend/ teremfoglalás
- **Szállításütemezés**
- **Gyári ütemezés**

Például **Raktár-tervezési probléma**

Egy szupermarket-lánc raktárakat szeretne telepíteni a bolti hálózat kiszolgálására. Mit tudunk?

- $L_1, \ldots, L_n$ lokáció, ahol raktár megépíthető.
- Csak K db. raktárra van szükség ($K < n$).
- Minden lokációra jellemző CP_i kapacitás: hány boltot képes kiszolgálni
- Minden bolthoz rendelni kell egy raktárt.
- S_j bolt ellátása az L_i lokációból $P_{i,j}$ pénzbe kerül.
- Az összköltséget TC konstans alatt kell tartani.

CSP problémák típusai

Változók alapján:

Diszkrét változók

véges értéktartományok:

pl. Boole-típusú (IGEN/NEM) vagy többértékű (Tavasz/Nyár/…)

végtelen értéktartományok:

egész számok halmaza, füzérek, stb.

pl. job scheduling, változók a munkaszakaszok kezdete/vége

Folytonos változók

megfigyeléseket határoló időpontok,

fizikai állapotváltozók

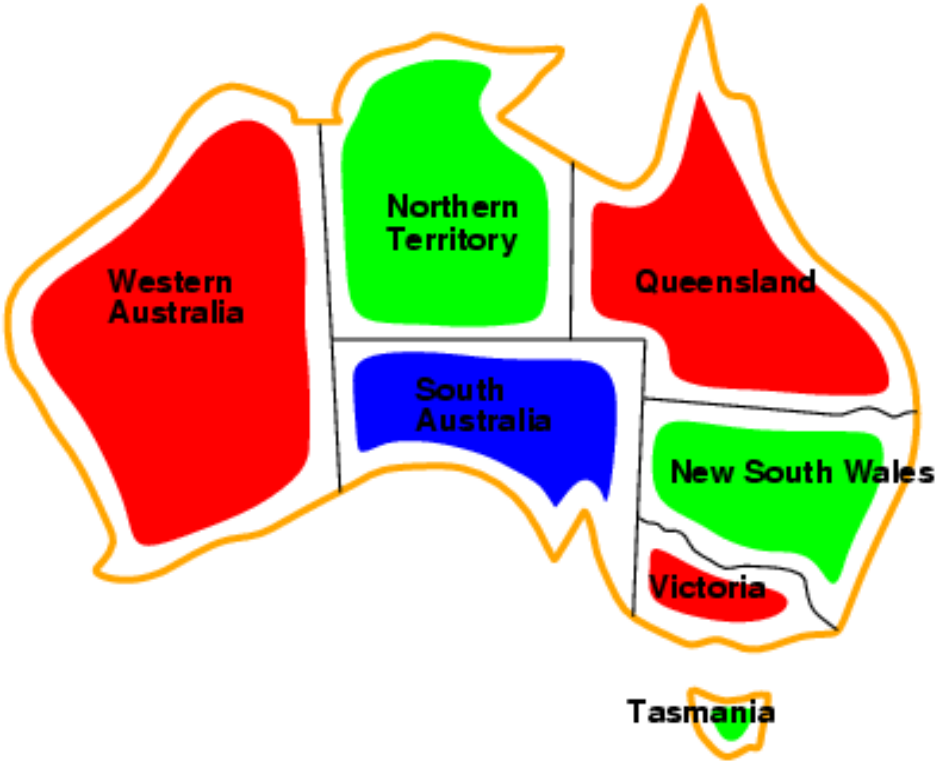
***Kényszerek, korlátok alapján*:**

Unáris korlát: egyetlen egy változóra vonatkozik, pl. SA ≠ green, x<5

Bináris korlát: két változó viszonyára vonatkozik, pl. SA ≠ WA, x<y

Magasabb-rendű korlát: 3 vagy több változó viszonyára vonatkozik, (pl. oszloponkénti változó korlátok kriptoaritmetikai feladványokban)

pl. Térképszínezés



Színezzük ki Ausztrália térképén az egyes államokat 3 színnel (piros, kék, zöld) úgy, hogy semelyik két szomszédos államnak ne legyen azonos a színe!

Demonstratív példa: Térképszínezés

Változók: *WA, NT, Q, NSW, V, SA, T*

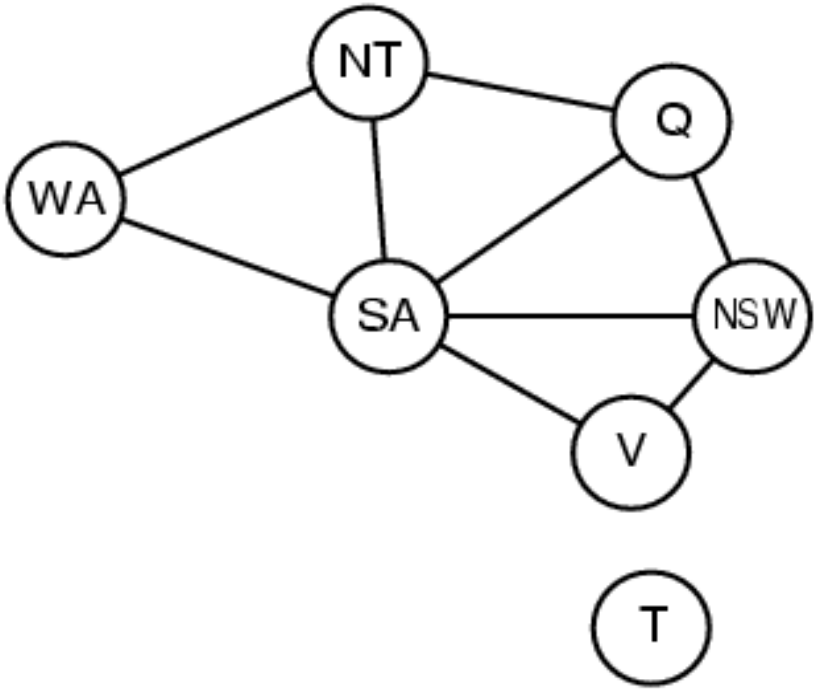
Értéktartományok D_i = {**red**, **green**, **blue**}

Korlátok: szomszédos területek színe legyen eltérő

pl. WA ≠ NT, ill. más megfogalmazásban: (WA,NT) értékét csakis a {(red, green), (red, blue), (green, red), (green, blue), (blue, red), (blue, green)} halmazból vehet fel.

Lehetséges-e?

Korlátok gráfja



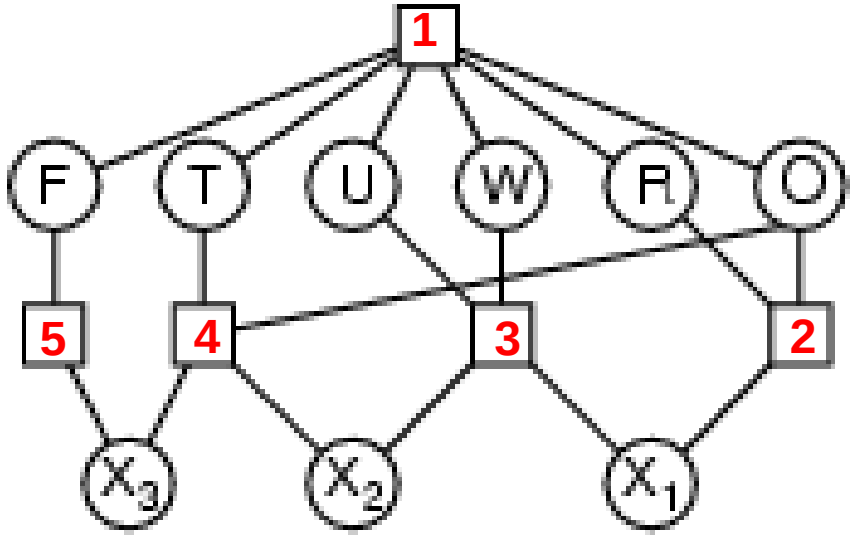
korlátgráf: csomópontjai a változók és élei a korlátok

Itt csak **bináris CSP**-k: egy-egy korlát 2 változót köt össze

...

Másik példa: kriptoaritmetika

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



Változók: *F T U W R O* X_1 X_2 X_3

Értéktartományok: {*0,1,2,3,4,5,6,7,8,9*}, {*0,1*}

Korlátok:

1. *Mind-eltérő(F, T, U, W, R, O)*
2. $O + O = R + 10 \cdot X_1$
3. $X_1 + W + W = U + 10 \cdot X_2$
4. $X_2 + T + T = O + 10 \cdot X_3$
5. $X_3 = F, T \neq 0, F \neq 0$

A probléma megfogalmazása

Kezdeti állapot: összes változó-hozzárendelés üres { }

Operátor: értéket hozzárendelni egy még nem lekötött változóhoz úgy, hogy az eddigi hozzárendelésekkel ne ütközzön (egyik kényszer se sérüljön)

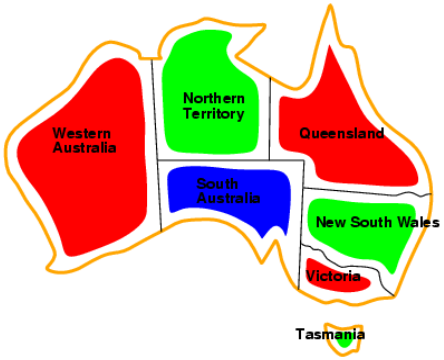
$\rightarrow$ kudarc, ha nincs megengedett hozzárendelés

Célállapotteszt: ha az aktuális hozzárendelés teljes, és mindegyik kényszer teljesül (egyik se sérül)

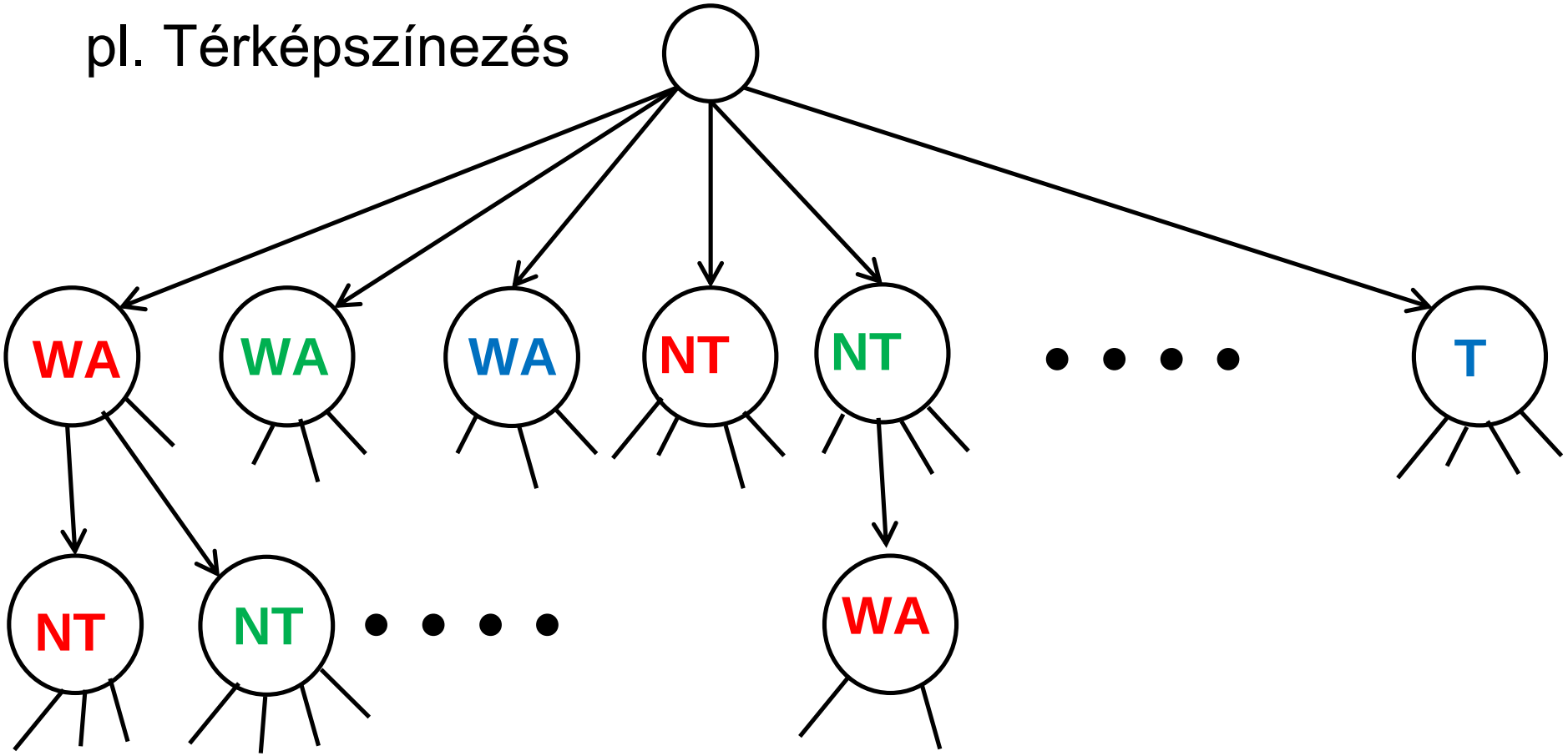
Keresési fa

n db változó esetén <u>minden megoldás *n* mélységben</u> fekszik

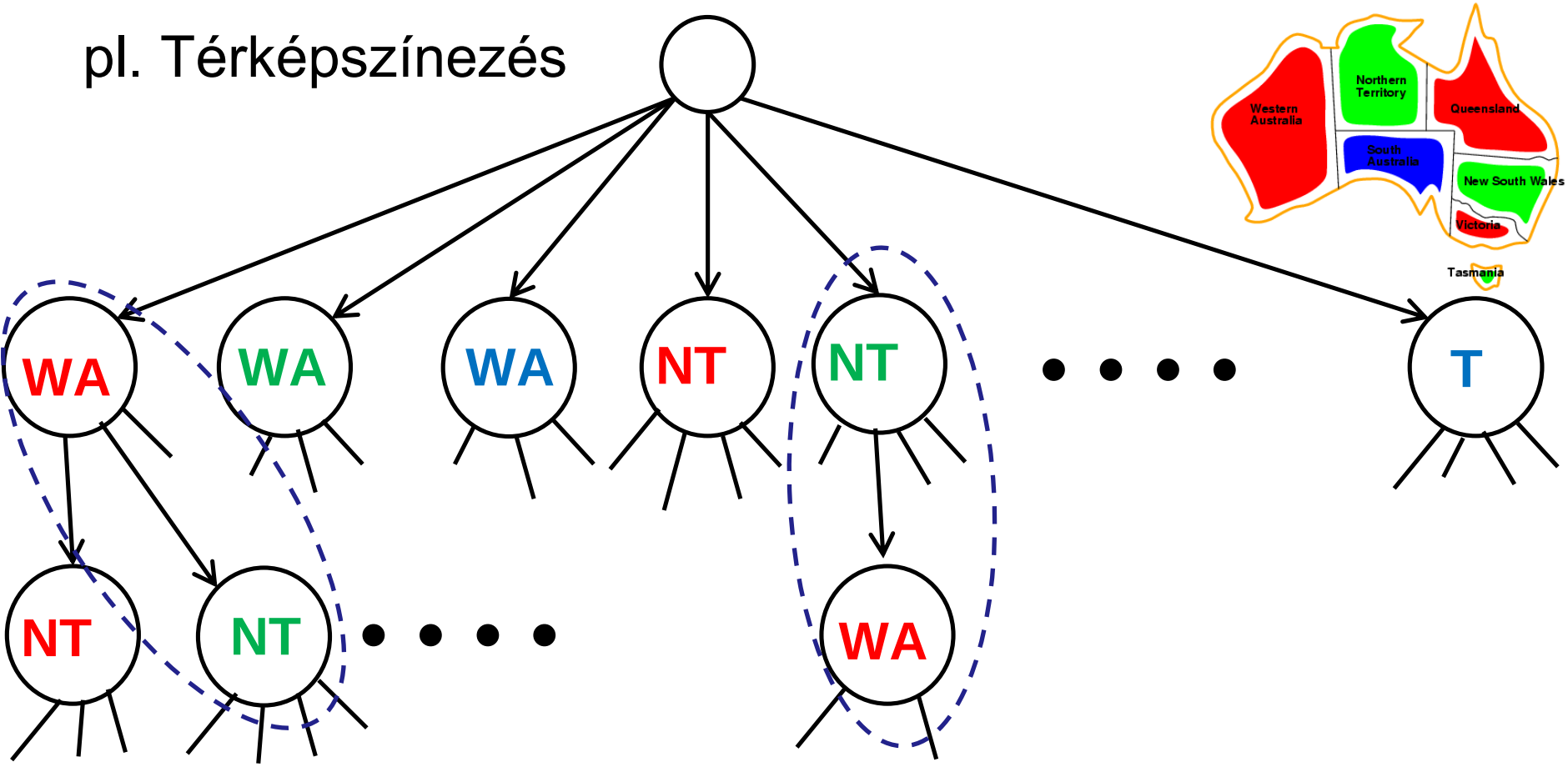
→ mi van, ha a szélességi kereséshez hasonlóan építjük fel a keresési fát használjuk (után, mechanikusan)?



pl. Térképszínezés



pl. Térképszínezés



Az első szinten: 7 terület és 3 szín ⇒ **21 csomópont**

A második szint: mindegyik ágán!: 6 maradék terület és 3 szín ⇒ **18 csp.**

⇒ összesen (7·3) · (6·3) ⇒ **21·18 = 378 csp.**

És így tovább…. Végül (7.6.5.4.3.2.1)*3^7= 11 022 480 levél csomópont!

Viszont pl. a bejelölt két értékadáspár (és még egy csomó) tökéletesen azonos eredményre vezet FELESLEGESEN BONYOLULT, REDUNDÁS KERESÉSI FA!

Keresési fa

n db változó esetén <u>minden megoldás *n* mélységben</u> fekszik

→ mi van, ha a szélességi jellegű keresést használjuk (bután, mechanikusan)?

Elágazások száma *L* mélységben (*L=0,1,2,…*), ha minden változó *d* számú értéket vehet fel (ez a változó értékkészlete v. más néven doménje)

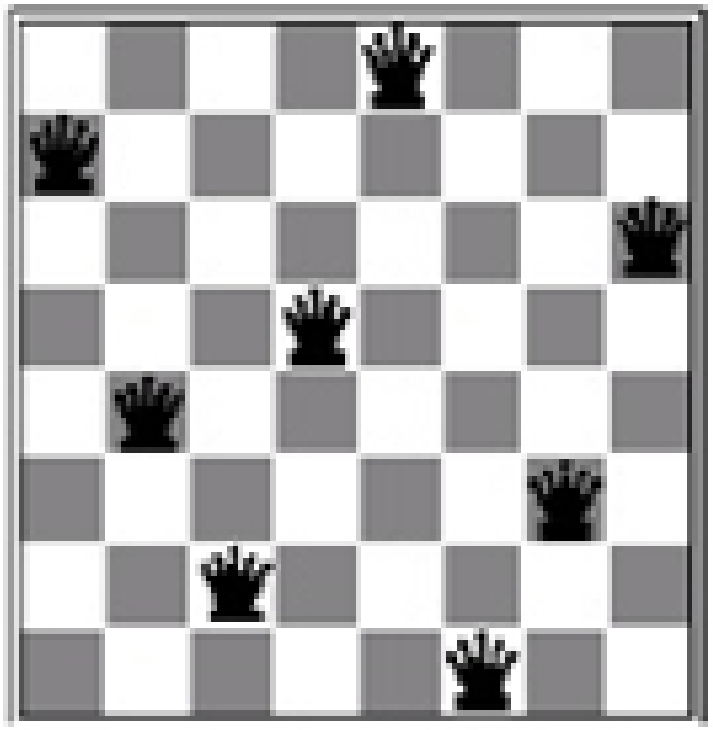
$$b = (n - L)\cdot d, \quad \text{(mert } L \text{ változó már értéket kapott)}$$

azaz $\boldsymbol{n!\cdot d^{n}}$ levélcsomópont (miközben $\boldsymbol{d^{n}}$ lehetséges hozzárendelés van)

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

Megfogalmazás (modell) hatása a problémamegoldásra

Az *n*-királynő probléma: egy *n*x*n*-es sakktáblán helyezzünk el *n* királynőt (vezért) úgy, hogy egyik se üsse a másikat!



Hasonlítsunk össze három modellt

1. modell, a változók: x_{ij} (*a sakktáblamezők pozíciója: $m=n^2$*)

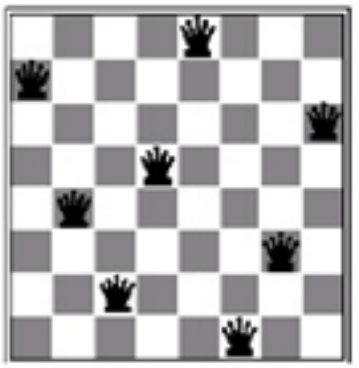
értékkészlet: {0, 1} (*van rajta királynő vagy nincs*)

kényszerek (sorok, oszlopok, átlók)

n x *n*-es sakktáblán az *n*-királynő probléma néhány *n*-re

modell	változó	értékkészl.(*d*)	levelek sz.(d^n)	*n* = 4	*n* = 8	*n* = 20 .
1.	n^2	2	$(2)^{(n\hat{}2)}$	$6{,}6\cdot10^4$	$1{,}8\cdot10^{19}$	$2{,}6\cdot10^{120}$

Megfogalmazás (modell) hatása a problémamegoldásra



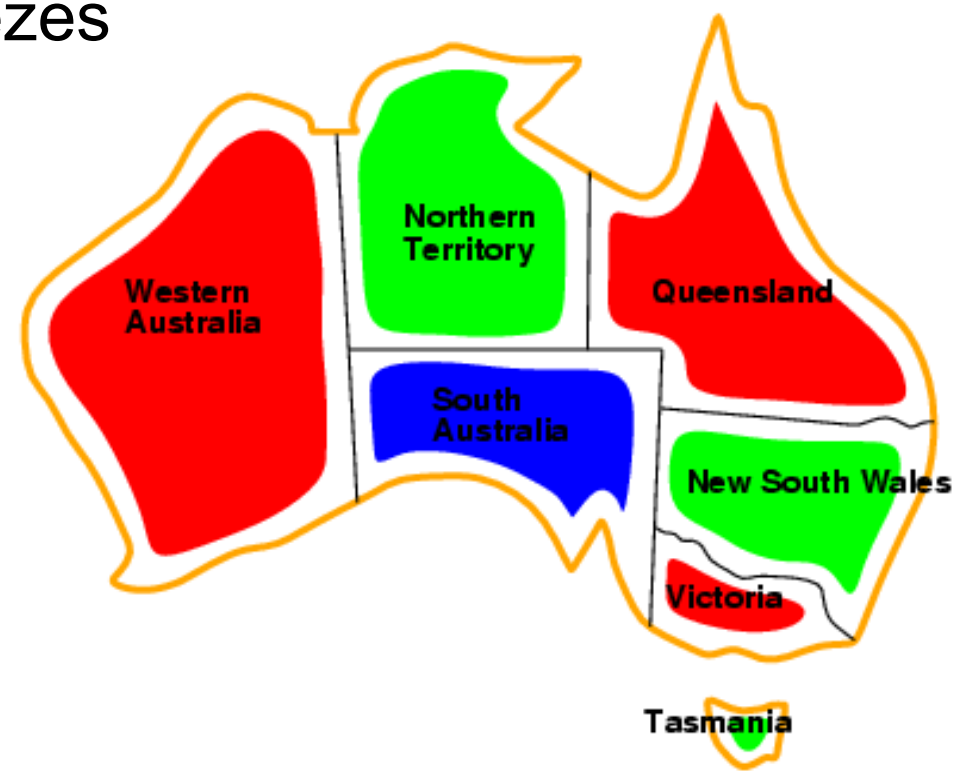
2. modell, a változók: $x_1, \ldots, x_n$ (*egy-egy királynő által elfoglalt mező pozíciója*)
értékkészlet: $\{0, 1, 2, \ldots, n^2-1\}$ (*mezőindex*)
kényszerek (sorok, oszlopok, átlók)

3. modell, a változók: $x_1, \ldots, x_n$ (az 1., 2. stb. sorban álló *királynő oszlopindexe*)
értékkészlet: $\{1, 2, \ldots, n\}$ (*oszlop-index*)
kényszerek (sorok, oszlopok, átlók)

n x *n*-es sakktáblán az *n*-királynő probléma néhány *n*-re

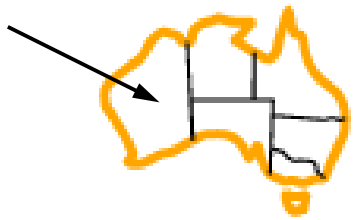
modell	változó	értékkészl.(*d*)	levelek sz.(d^n)	$n = 4$	$n = 8$	$n = 20$.
1.	n^2	2	$(2)^{(n\hat{}2)}$	$6{,}6 \cdot 10^4$	$1{,}8 \cdot 10^{19}$	$2{,}6 \cdot 10^{120}$
2.	n	n^2	$(n^2)^n$	$6{,}6 \cdot 10^4$	$2{,}8 \cdot 10^{14}$	$1{,}1 \cdot 10^{52}$
3.	n	n	n^n	256	$1{,}6 \cdot 10^7$	$1{,}0 \cdot 10^{26}$

Másik példa: Térképszínezés



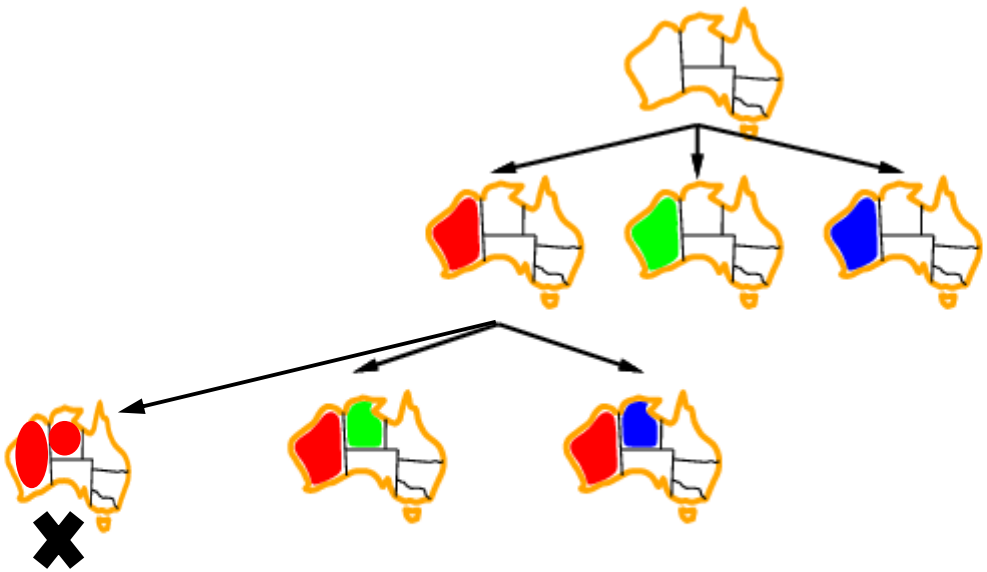
Színezzük ki Ausztrália térképén az egyes államokat 3 színnel (piros, kék, zöld) úgy, hogy semelyik két szomszédos államnak ne legyen azonos a színe!

Visszalépéses keresés

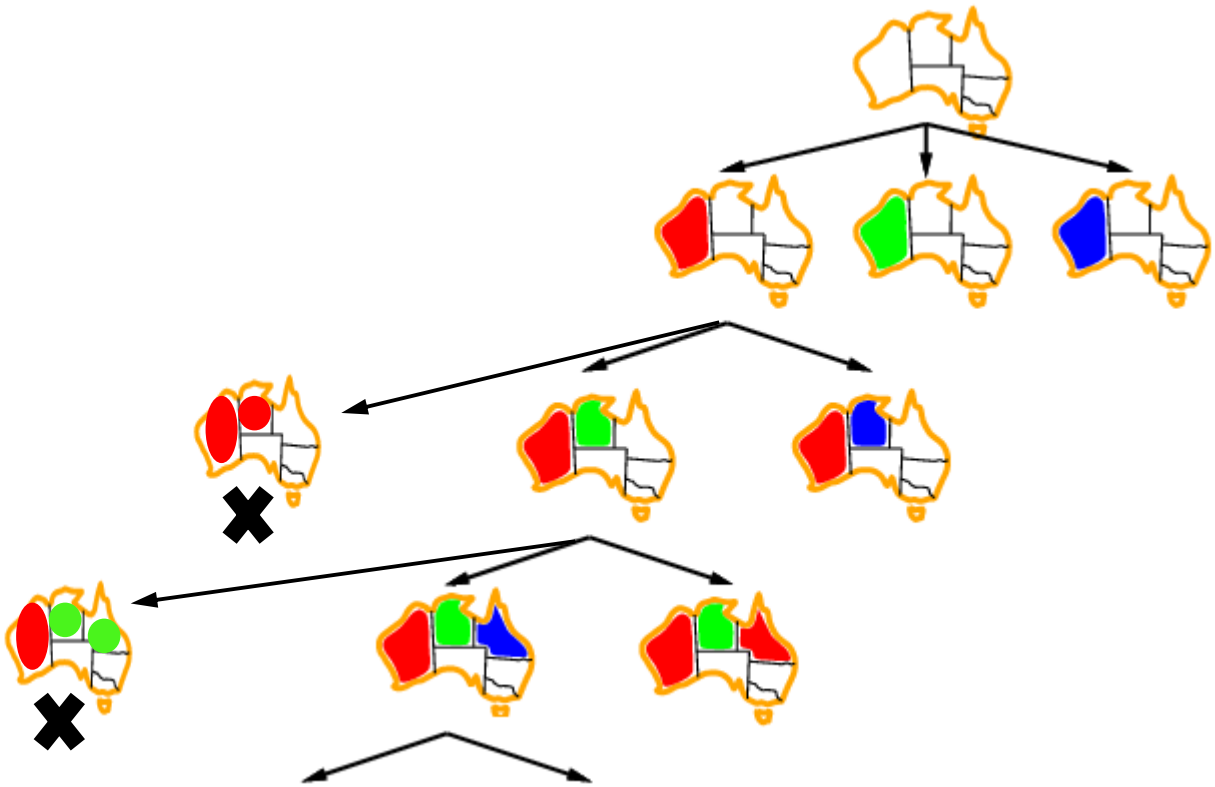


Visszalépéses keresés

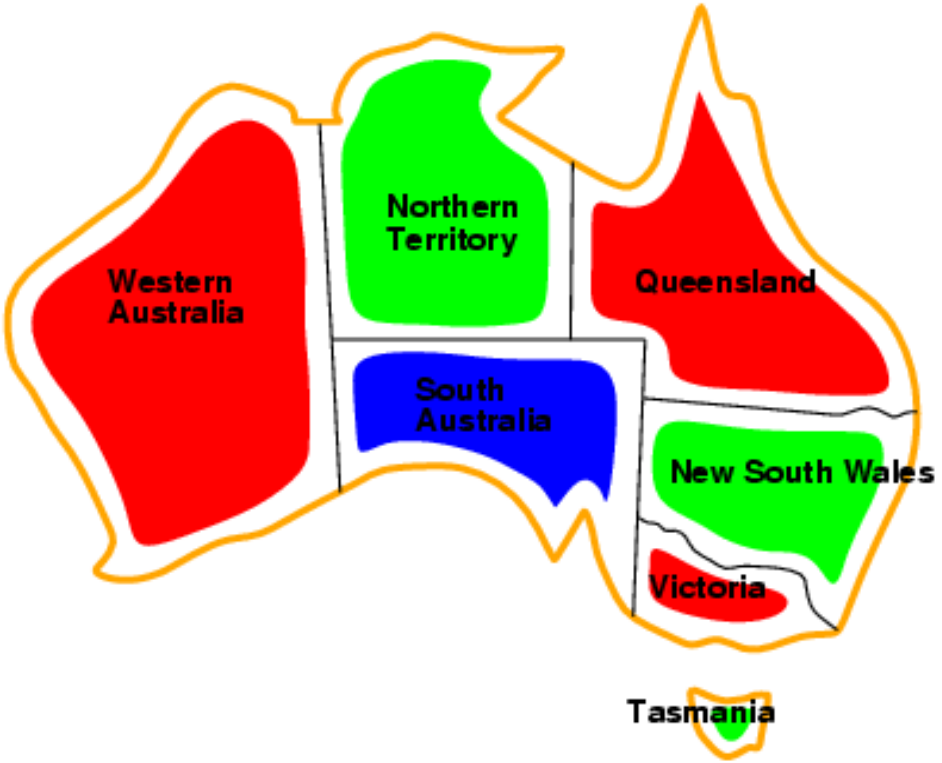
Visszalépéses keresés



Visszalépéses keresés



pl. Térképszínezés



Megoldás: ha előállítottunk egy **teljes** és **konzisztens** változó-érték hozzárendelést

pl. WA = red, NT = green, Q = red, NSW = green, V = red, SA = blue, T = green

(T lehet akármi, mert nem határos senkivel

- ettől függetlenül is több megoldás van)

Korlátozás kielégítés (CSP) - Keresés

Változó-hozzárendelés **kommutatív**, azaz például

(WA = red) majd (NT = green)

ugyanaz, mint (NT = green) majd (WA = red)

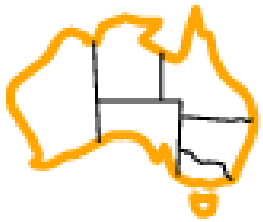
Egy-egy csomópontban csakis egyetlen egy változó hozzárendelése történhet meg, így: → a fának n változó, d lehetséges érték ($b = d$) esetén maximum d^n levele van

****

Alapvetően nem informált algoritmus (keresés) CSP problémák megoldására: **mélységi keresés, visszalépéses keresés**.
Minden szinten egyetlen egy változó-hozzárendeléssel - ha sérül valamelyik kényszer, visszalép (egyszer sérült kényszer mélyebben nem jöhet helyre!).

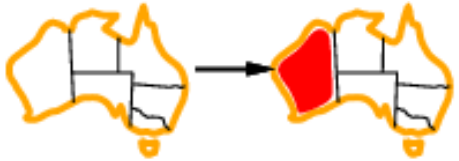
Előretekintő ellenőrzés/1 (keresés)

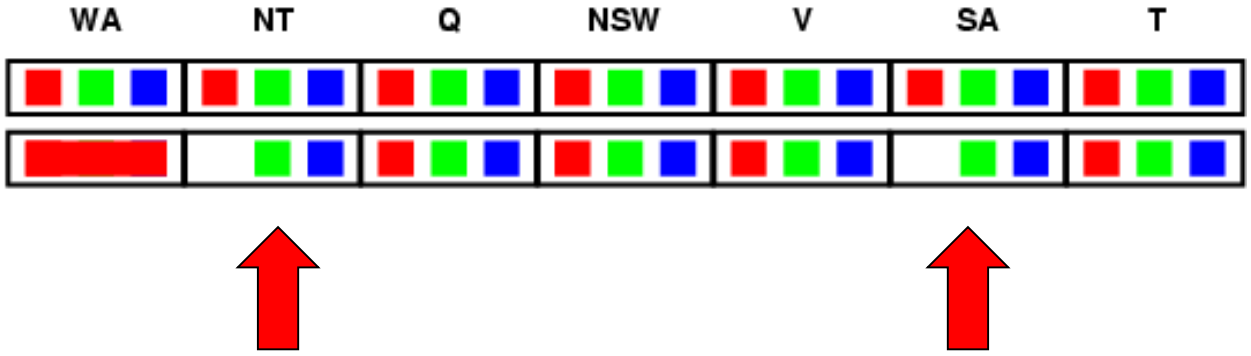
Az előretekintő ellenőrzés minden egyes alkalommal, amikor egy X változó értéket kap, minden, *az X-hez kényszerrel kötött*, lekötetlen Y-t megvizsgál, és Y tartományából törli az X számára választott értékkel inkonzisztens értékeket.

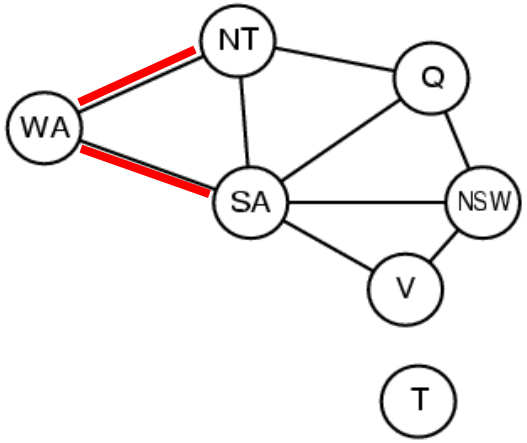


Előretekintő ellenőrzés/2 (keresés)

Az előretekintő keresés minden egyes alkalommal, amikor egy X változó értéket kap, minden, az X-hez kényszerrel kötött, lekötetlen Y-t megvizsgál, és Y tartományából törli az X számára választott értékkel inkonzisztens értékeket.

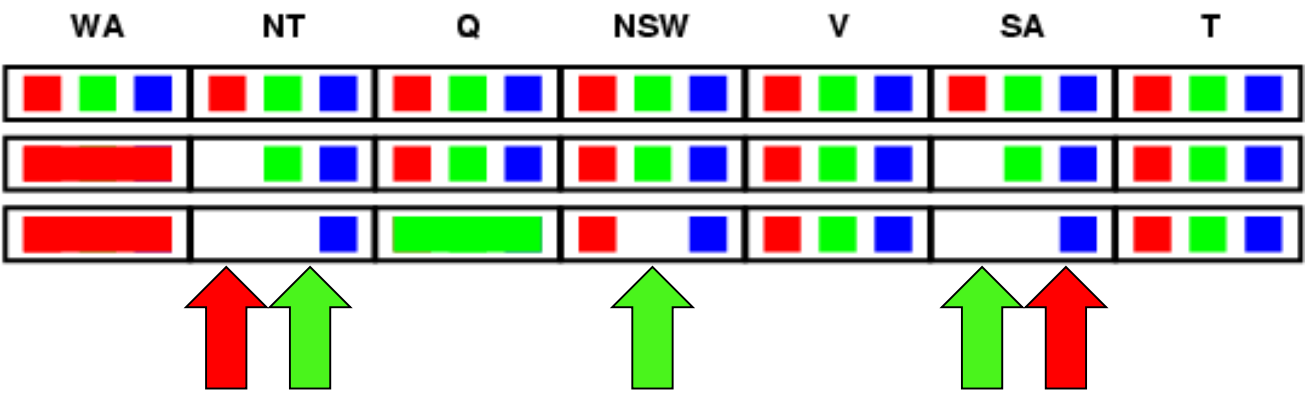


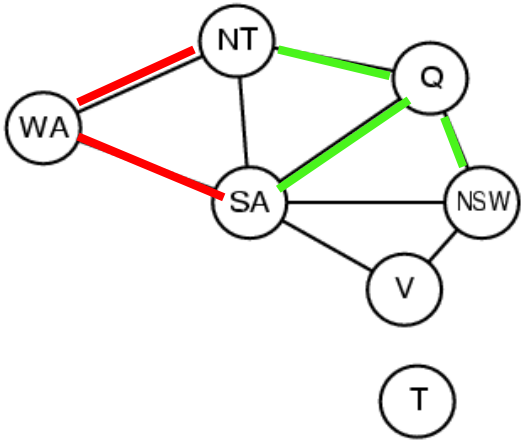




Előretekintő ellenőrzés/3 (keresés)

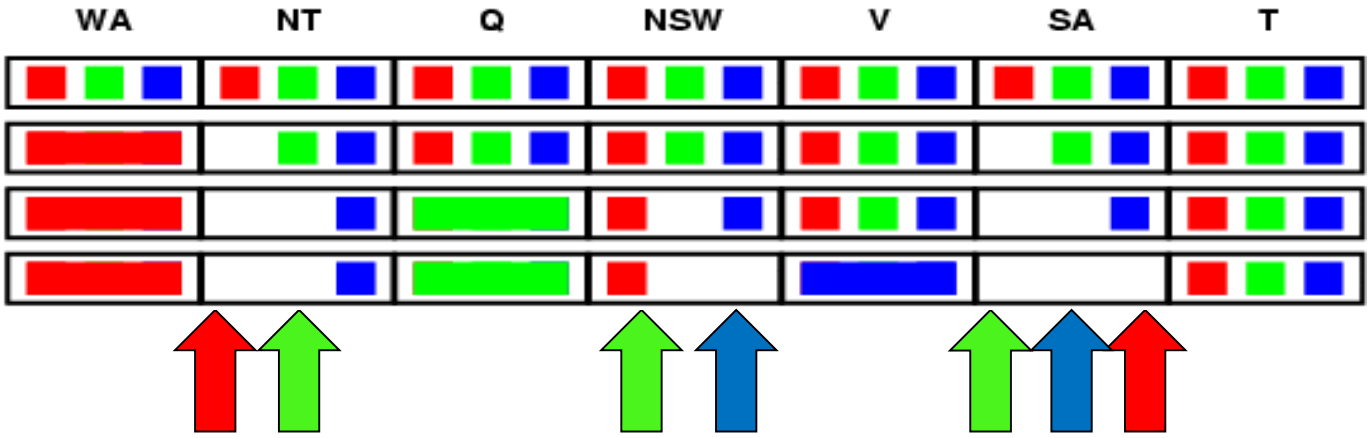
Az előrenéző ellenőrzés minden egyes alkalommal, amikor egy *X* változó értéket kap, minden, az *X*-hez kényszerrel kötött, még nem hozzárendelt *Y*-t megvizsgál, és *Y* lehetséges értékei közül törli az X számára választott értékkel inkonzisztens értékeket.

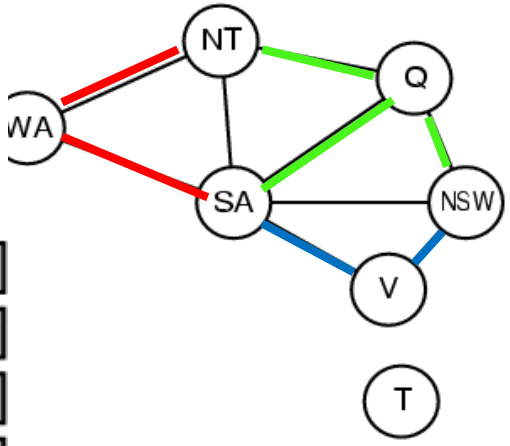




Előretekintő ellenőrzés/3 (keresés)

Az előrenéző ellenőrzés minden egyes alkalommal, amikor egy *X* változó értéket kap, minden, az *X*-hez kényszerrel kötött, még nem hozzárendelt *Y*-t megvizsgál, és *Y* lehetséges értékei közül törli az X számára választott értékkel inkonzisztens értékeket.

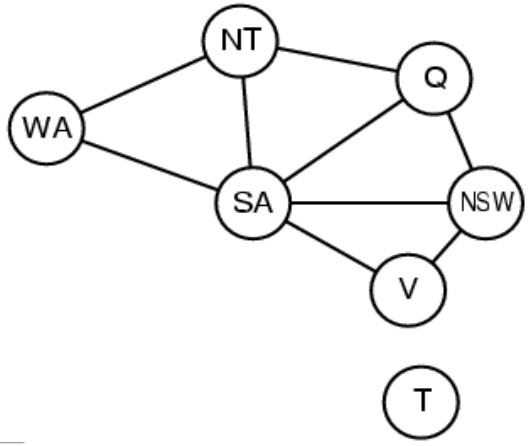




Korlátozás előreterjesztése

Az ***előretekintő ellenőrzés*** ugyan sok inkonzisztenciát észrevesz, de nem mindent.

Ráadásul nem látja jól előre a kudarcokat.



NT és SA egyszerre nem lehet kék. Ez csak később derül majd ki, most még nem sérül kényszer, mert egyik se kapott még értéket, csak kiürült az értékkészletük. Bonyolultabb vizsgálattal (a kényszereket megvizsgálva minden lépés után az értékkészletekre) ezt előre kideríthetjük.

Visszalépéses keresés hatékonyságának növelése (általános heurisztikák CSP-khez)

Általános módszerekkel is komoly gyorsítást el lehet érni:

- Melyik változóval foglalkozzunk a legközelebb?
- Milyen sorrendben vizsgáljuk az értékeit?
- Érzékelhetjük-e jó előre a kudarcokat? (korai nyesés)

ezek az ún. tárgyterület-független heurisztikák

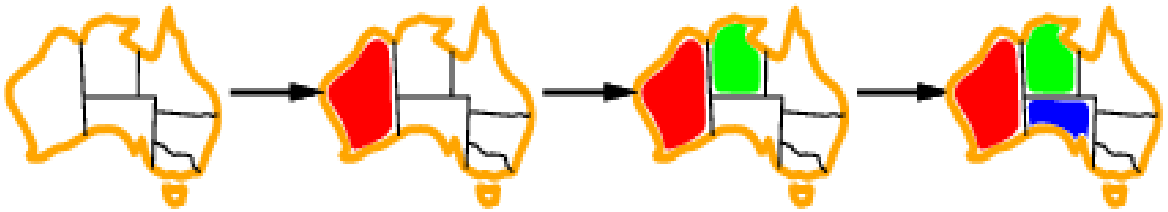
1. *A legkevesebb fennmaradó érték* ötlete (melyik változót válasszuk?)

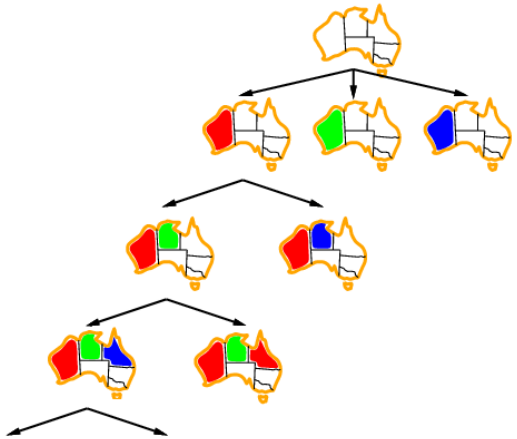
A leginkább korlátozott változó:

a legkisebb számú megengedett értékkel rendelkező változóval kezdjünk, ill. folytassunk (***lokálisan kicsi az elágazási tényező!***)

⇒ **legkevesebb fennmaradó érték** heurisztika

(*minimum remaining variables*, **MRV**)





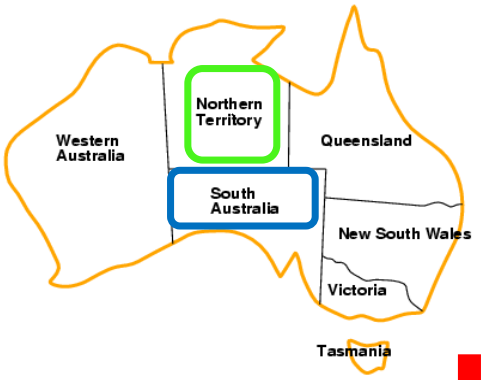
NT ill. SA csak 2 megengedett érték (piros már nem lehet), minden más 3. Nem érdemes Ausztrália „túlsó” (keleti) oldalán folytatnunk a színezést)

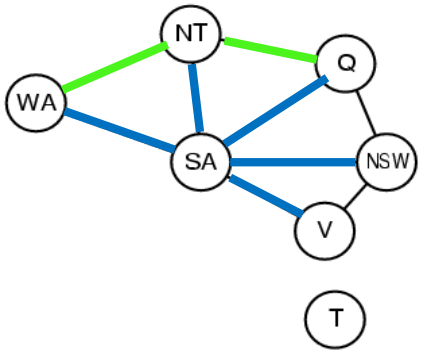
2. *Fokszám heurisztika* ötlete

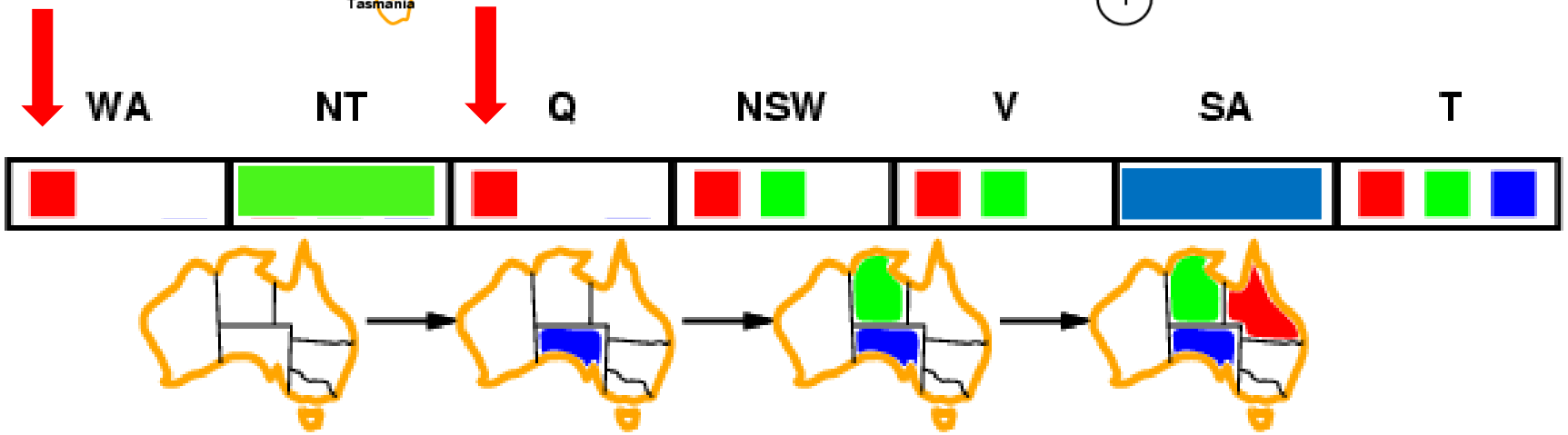
(melyik változót válasszuk?)

Az MRV-heurisztika semmit sem segít abban, hogy melyik régiót válasszuk ki elsőként Ausztrália kiszínezésekor, mert a kiinduláskor mindegyik régiónak három megengedett színe van.

A ***későbbi választások elágazási tényezőjét csökkentheti***, ha azt a változót választjuk ki, amely a legtöbbször szerepel a még hozzárendeletlen változókra vonatkozó kényszerekben.

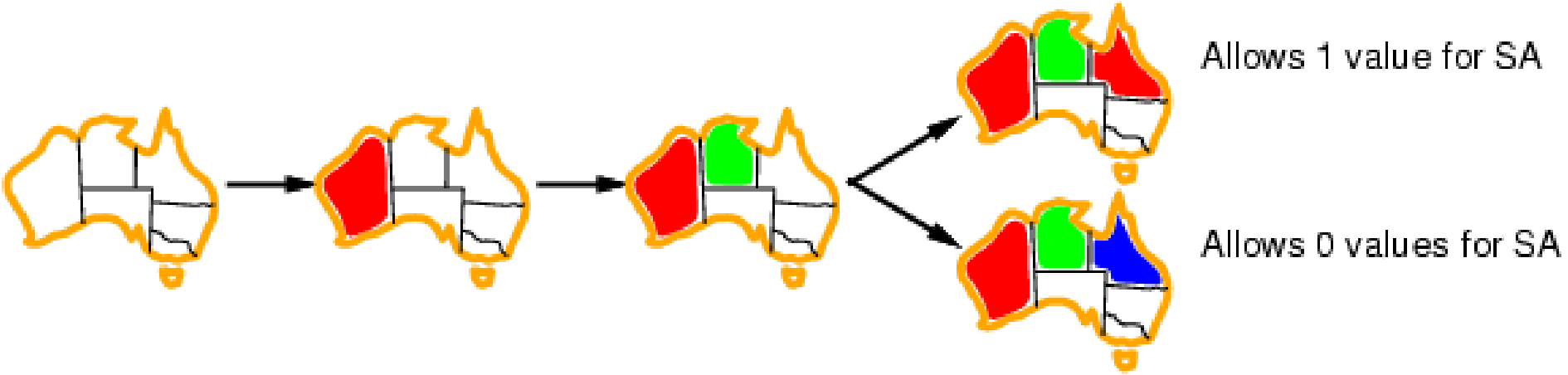






3. *A legkevésbé korlátozó érték* ötlete

Előnyben részesítjük azt az értéket, amely a legkevesebb választást zárja ki a kényszergráfban a szomszédos változóknál.



Korlátozáskielégítés (CSP) lokális kereséssel

Kiindulás: teljes állapotleírás = minden változónak van értéke (esetleg rossz, nem teljesült kényszerekkel)

Operátorok: megváltoztatják a változók hozzárendelését, hogy csökkenjen a sérült kényszerek száma

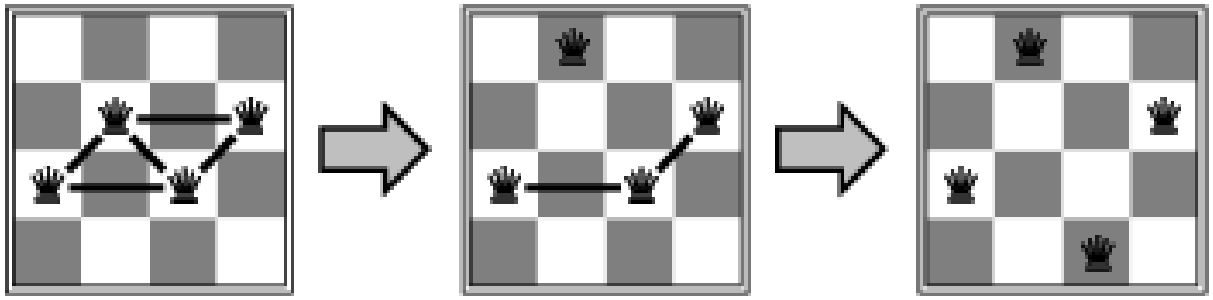
Változó szelekció: véletlen módon, bármely konfliktusban lévő (valamelyik kényszer sérül) változót választhatjuk

****

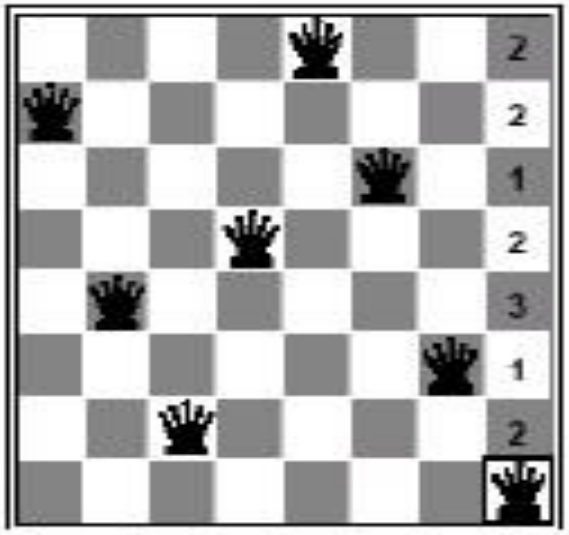
Min. konfliktus heurisztika:

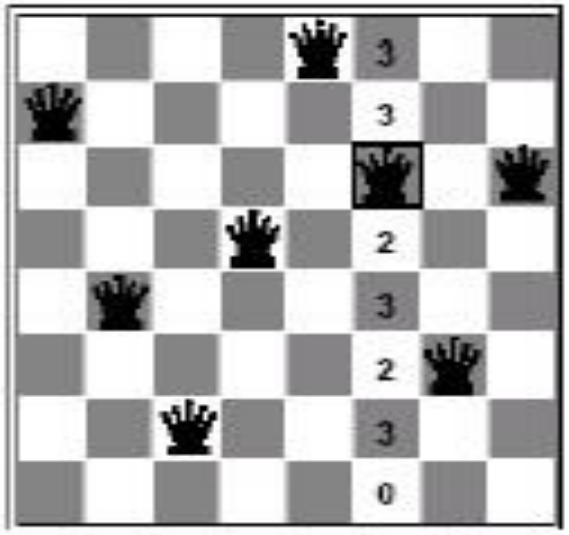
azt az értéket állítjuk be, amely a legkevesebb számú korlátot sérti, pl. hegymászó: *h(n)* = sérült korlátok száma, *h*(*n*) csökkentése a cél (itt most lefele mászunk a völgybe)

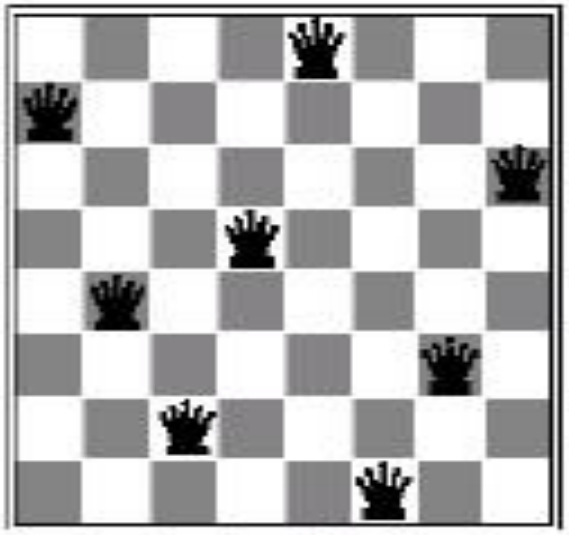
h(n) = a támadások száma



CSP lokális kereséssel / Min. konfliktus heurisztika







Min. konfliktus heurisztika nagyon hatékony, nagy valószínűséggel gyorsan old meg nagyon nagy problémaeseteket (10 millió királynő!).