Mesterséges Intelligencia MI

Kényszerkielégítési problémák (Constraint Satisfaction Problem, CSP)

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

Pataki Béla

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

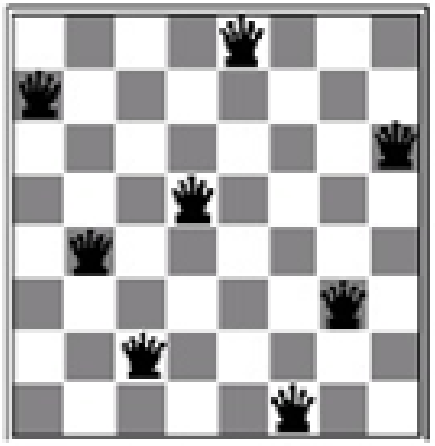
Kényszerkielégítési (korlátkielégítési) problémák
Constraint Satisfaction Problems (CSP):

állapot: Az állapot a leíró változók és a hozzájuk rendelt értékek által definiált. Az x_k változók egy-egy D_k érték-tartományból (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:

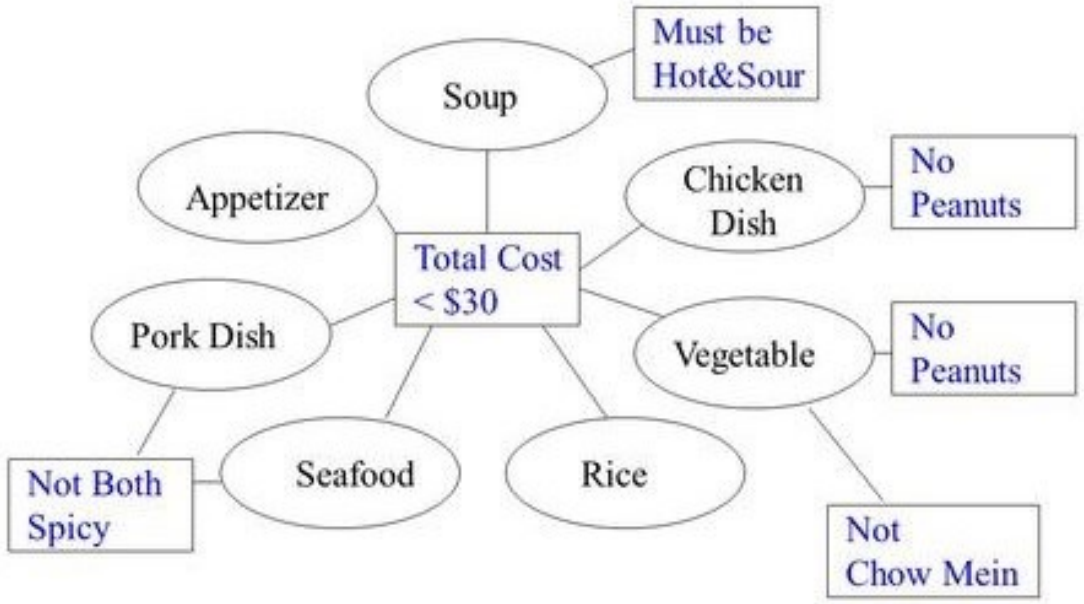
Szerda 14-16, Pataki Béla,MI-BSc5szem, Q-I

...

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.

Menüösszeállítás az étlap alapján



- **Hozzárendelési problémák**, pl. ki, mit, hol tanítson
- **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, ..., 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 L_i lokációból $P_{i,j}$-be kerül.
- Az összköltséget TC konstans alatt kell tartani.

CSP problémák típusai

Változók alapján:

Diszkrét változók

véges értéktartományok:

pl. Boole-típ. CSPs, Boole-féle kielégítési vizsgálatok (NP-t)

végtelen értéktartományok:

egész számok, füzérek, stb.

pl. job scheduling, változók a munkaszakaszok kezdete/vége

Folytonos változók

megfigyeléseket határoló időpontok,

fizikai állapotváltozók

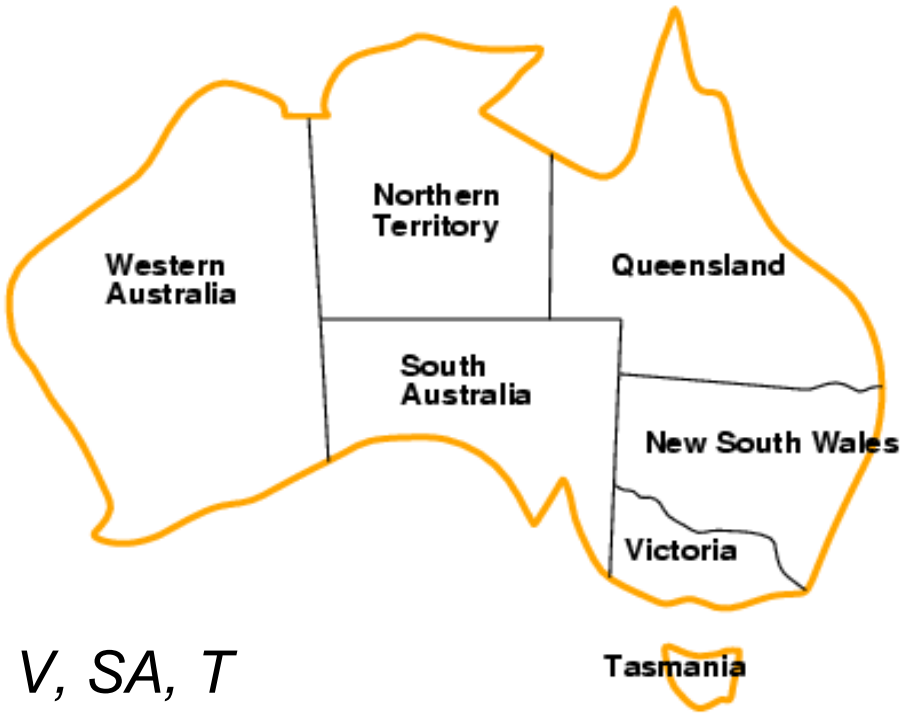
***Kényszerek, korlátok alapján*:**

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

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

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

Példa: Térképszínezés



Változók: *WA, NT, Q, NSW, V, SA, T*

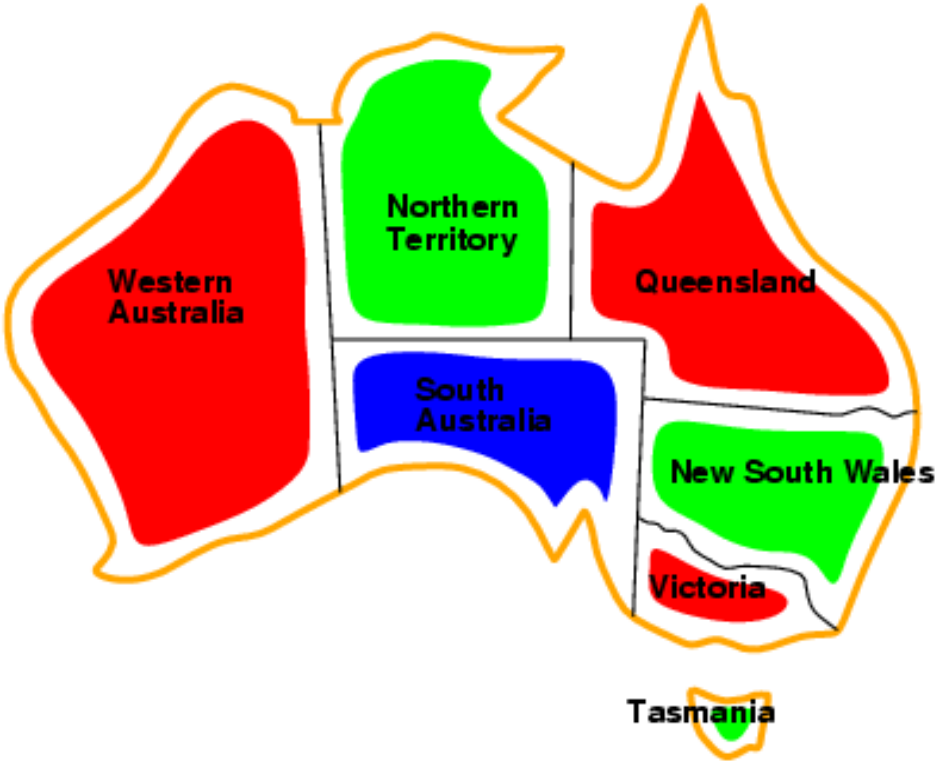
Értéktartományok D_i = {red, green, blue}

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?

pl. Térképszínezés

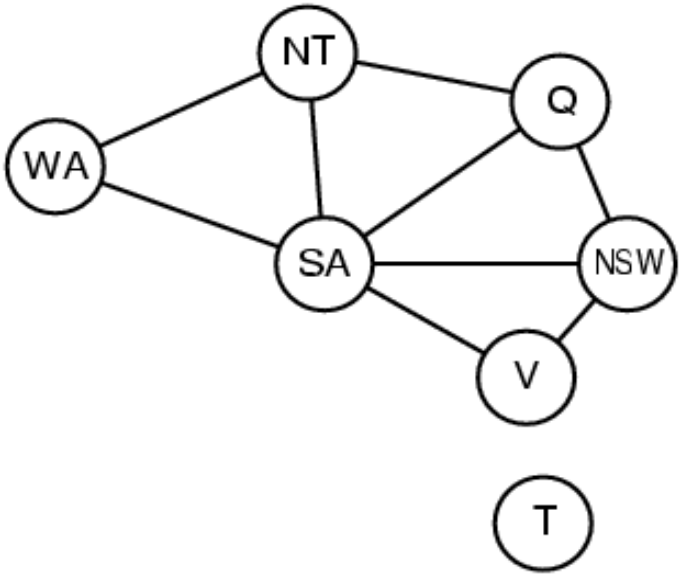


Megoldás: **teljes** és **konzisztens** változó-érték hozzárendelés

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

(T lehet akármi, mert nem határos senkivel
- különben is több megoldás van)

Korlátok gráfja



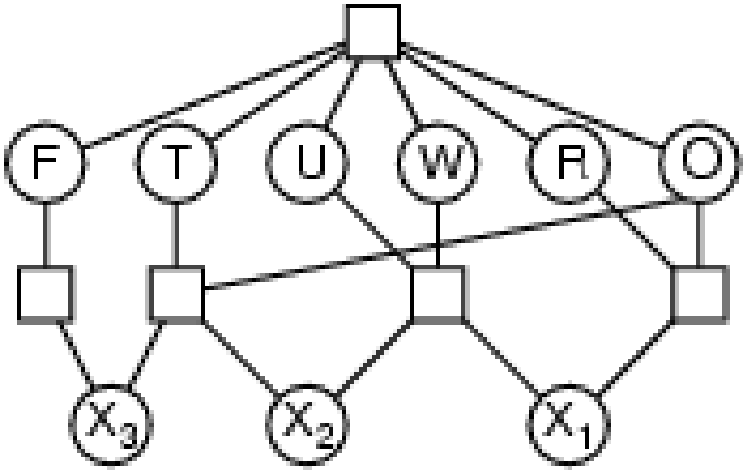
korlátgráf: csomópontjai a változók és élei a korlátok

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

...

Másik példa: kriptoaritmetika

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



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)

→ kudarc, ha nincs megengedett hozzárendelés

Célállapotteszt: ha az aktuális hozzárendelés teljes és mindegyik kényszer teljesül

****

Keresési fa

n db változó esetén minden megoldás *n* mélységben fekszik

→ mi van, ha a **szélességi keresést** használjuk?

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

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

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

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

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:

→ $b = d$ és a fának d^n levele van

****

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

Megfogalmazás (modell) hatása a problémamegoldásra

Az n-királynő probléma tanulsága: hasonlítsunk össze három modellt

1. modell, a változók: x_{ij} (*a sakktáblamezők pozíciója*)
értékkészlet: {0, 1} (*van rajta királynő vagy nincs*)
kényszerek (sorok, oszlopok, átlók)

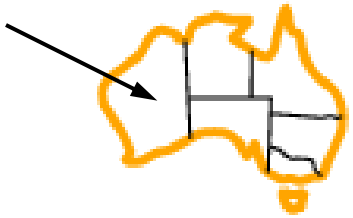
2. modell, a változók: $x_1, \ldots, x_n$ (*egy királynő által elfoglalt mező pozíciója*)
értékkészlet: {0, 1, 2, …, n^2-1} (*mezőindex*)
kényszerek (sorok, oszlopok, átlók)

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

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

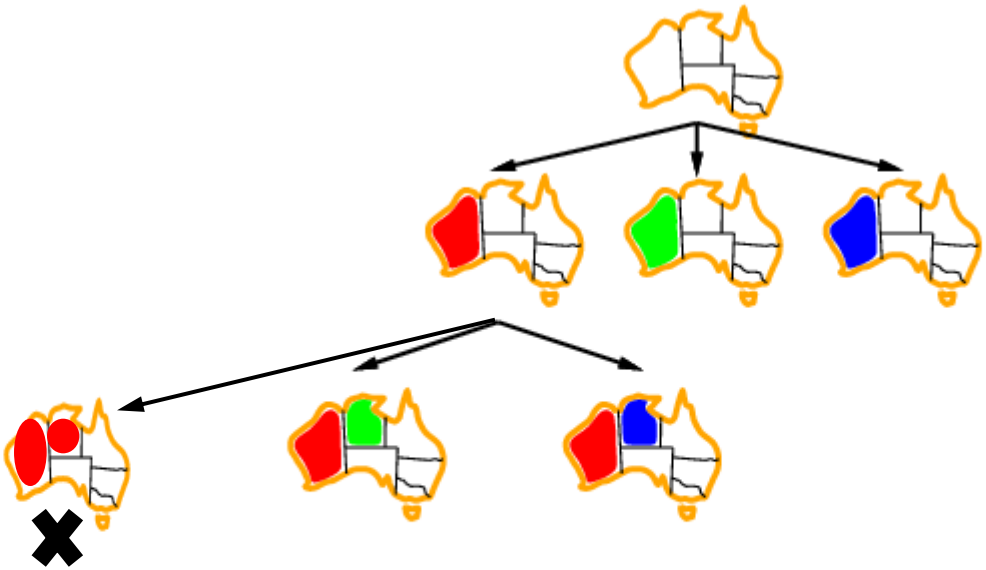
modell	változó	értékkészl.(*d*)	levelek sz.(d^n)	n = 4	n = 8	n = 20
1.	n^2	2	$(2)^{(n\hat{}2)}$	$6.6 \cdot 10^4$	$1.8 \cdot 10^{19}$	$2.6 \cdot 10^{120}$
2.	*n*	n^2	$(n^2)^n$	$6.6 \cdot 10^4$	$2.8 \cdot 10^{14}$	$1.1 \cdot 10^{52}$
3.	*n*	n	n^n	256	$1.6 \cdot 10^7$	$1.0 \cdot 10^{26}$

Visszalépéses keresés

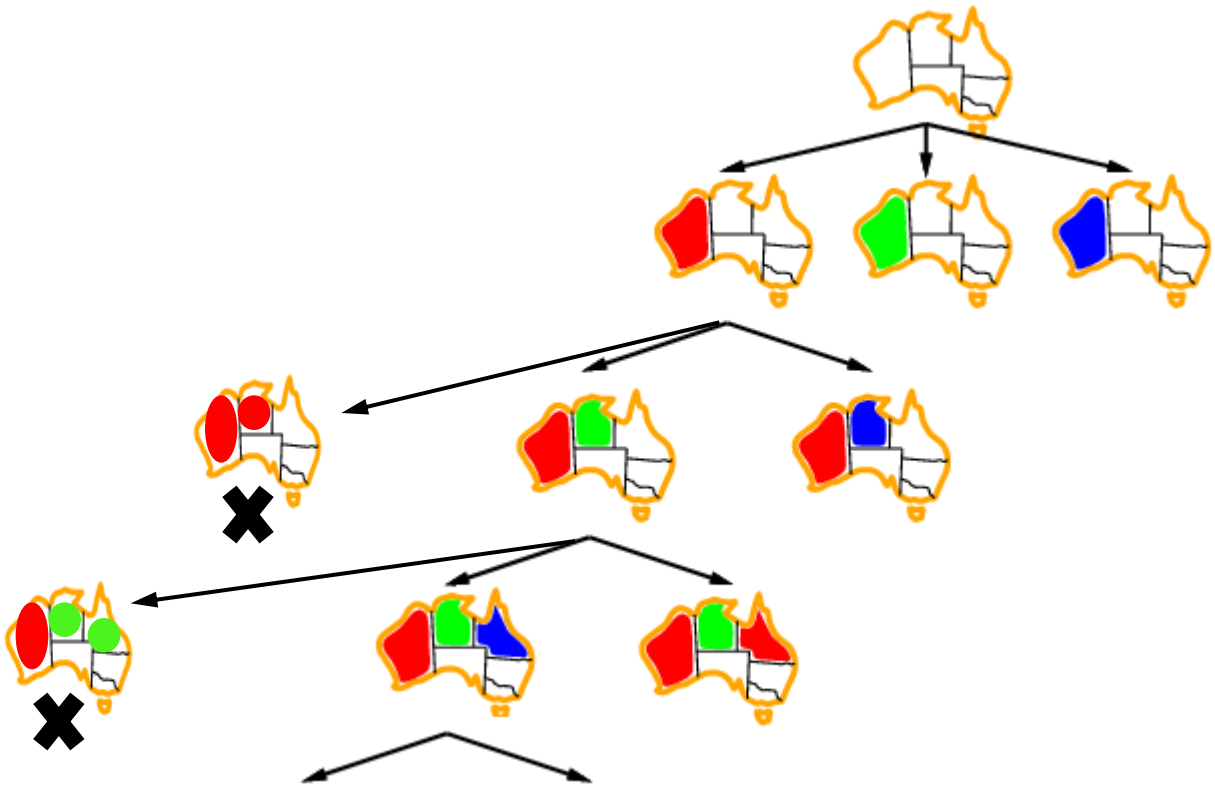


Visszalépéses keresés

Visszalépéses keresés

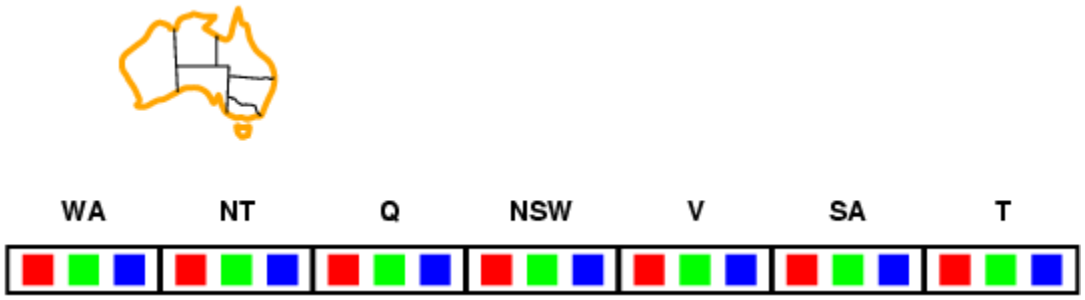


Visszalépéses keresés



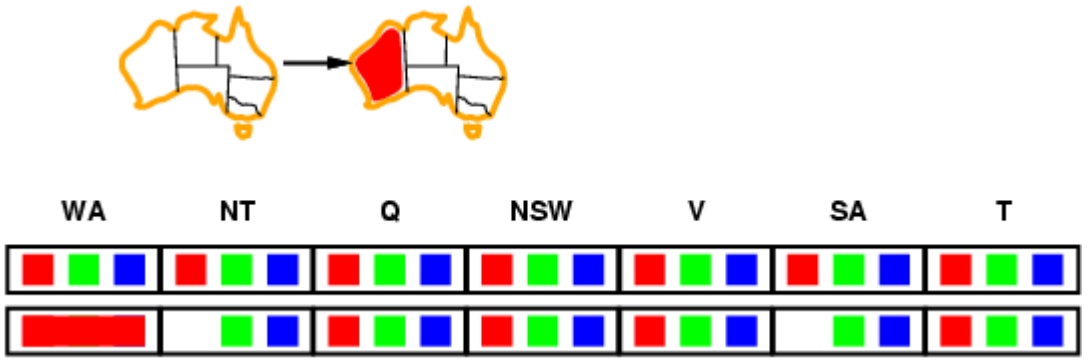
Előretekintő ellenőrzés (keresés)

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.



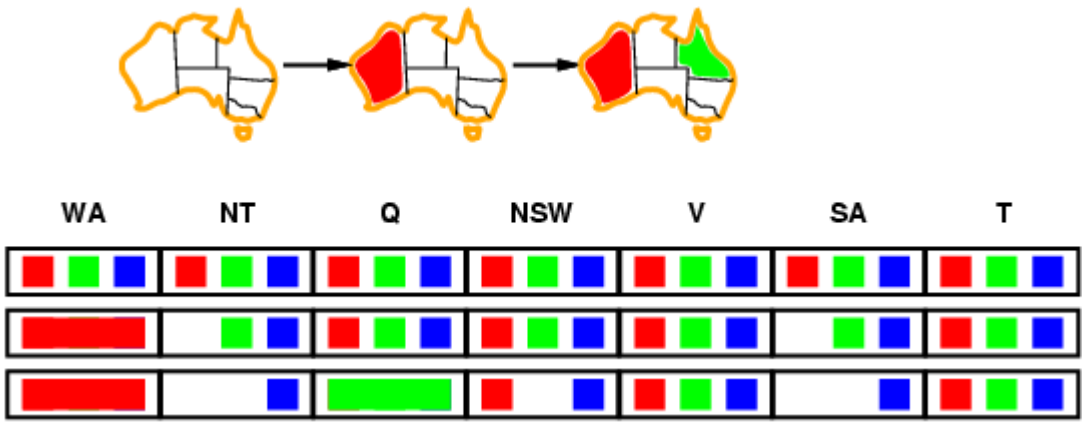
Előretekintő ellenőrzés (keresés)

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.



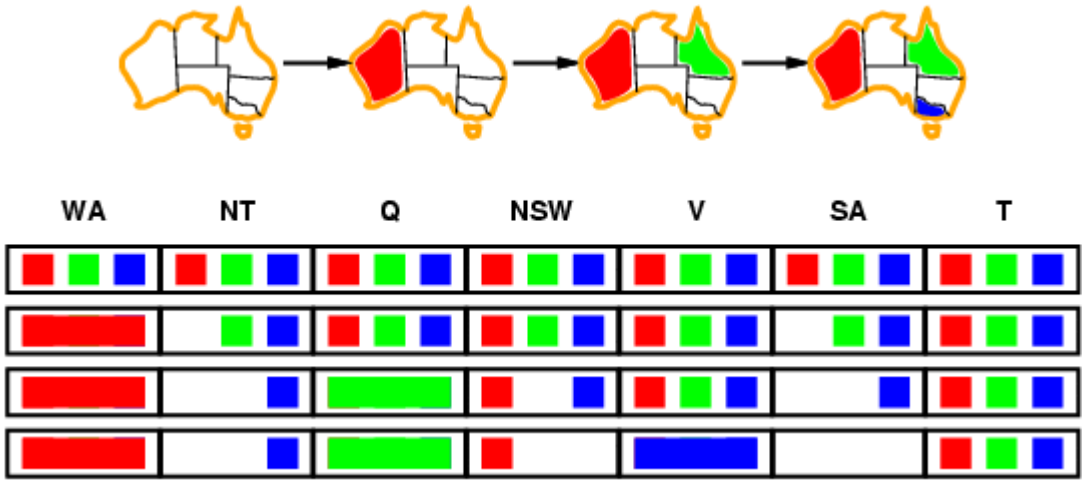
Előretekintő ellenőrzés (keresés)

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



Előretekintő ellenőrzés (keresés)

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

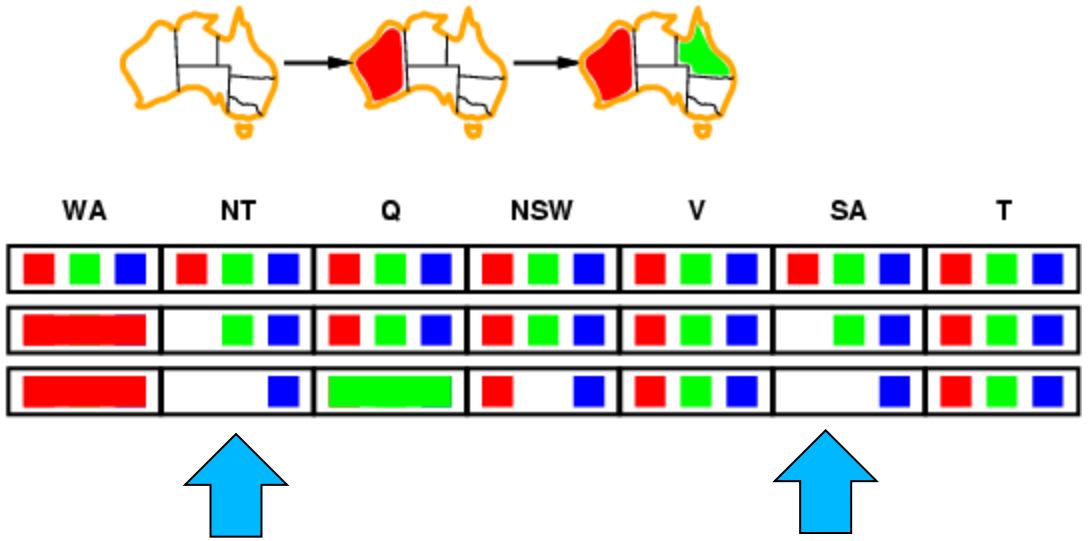


Korlátozás előreterjesztése

Kvíz következik!

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

Ráadásul nem látja jól előre a kudarcokat.



NT és SA egyszerre nem lehet kék.

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

Kvíz: Melyik változóval foglalkozzunk először, és milyen értéket rendeljünk hozzá?

Kényszerek:

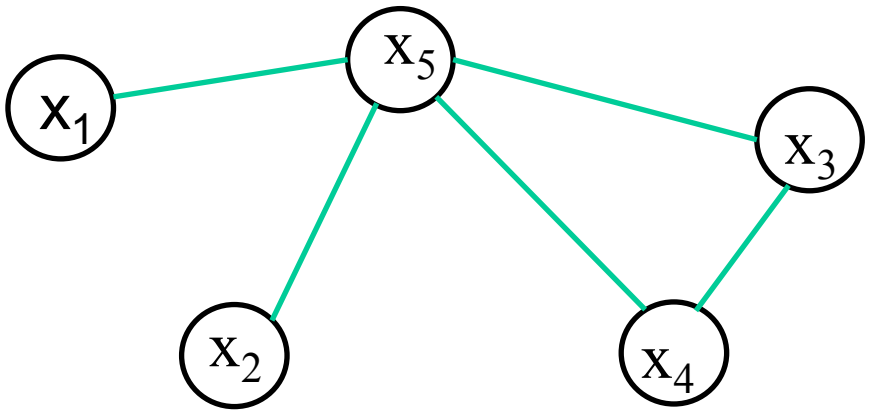
$x_1 > x_5$

$x_2 > 1 + x_5$

$x_3 > 2 \cdot x_5$

$x_4 > x_5 - 1$

$x_3 = 2 + x_4$



Mindegyik változó értékkészlete az egyjegyű nem negatív egészek halmaza: {0,1,2,…,9}

A. x1 ⇐ 5

B. x2 ⇐ 1

C. x3 ⇐ 2

D. x5 ⇐ 0

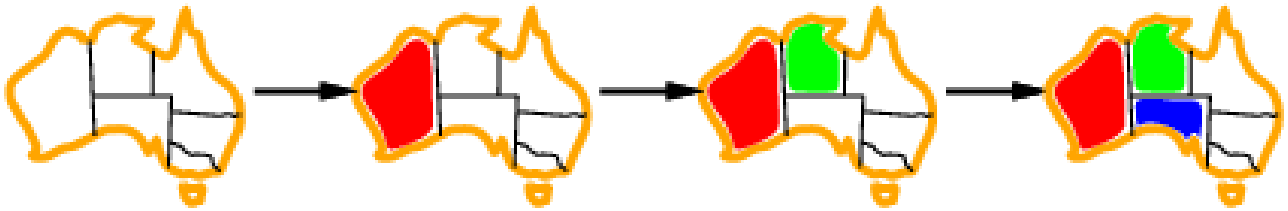
1. *A legkevesebb fennmaradó érték* ötlete (melyik változót válasszuk?)

A leginkább korlátozott változó:

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

= **legkevesebb fennmaradó érték** heurisztika

(min remaining variables, MRV)

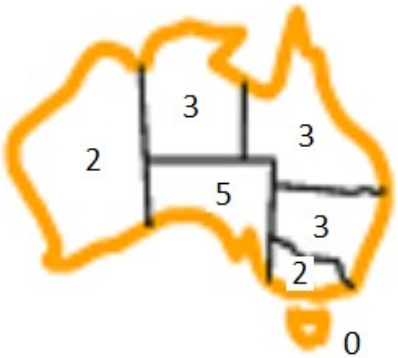


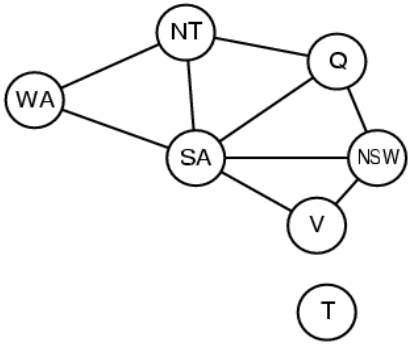
(NT ill. SA csak 2 megengedett érték (piros már nem lehet), minden más 3)

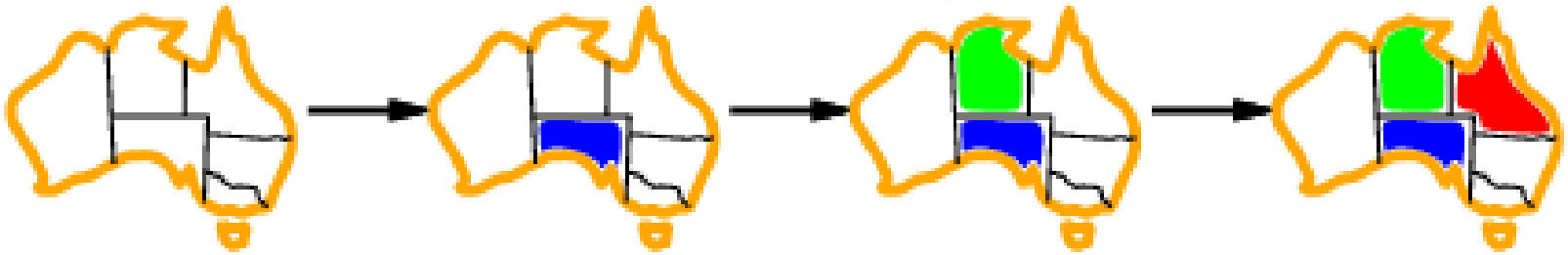
2. ***Fokszám heurisztika*** ötlete (melyik <u>változót</u> válasszuk?)

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.

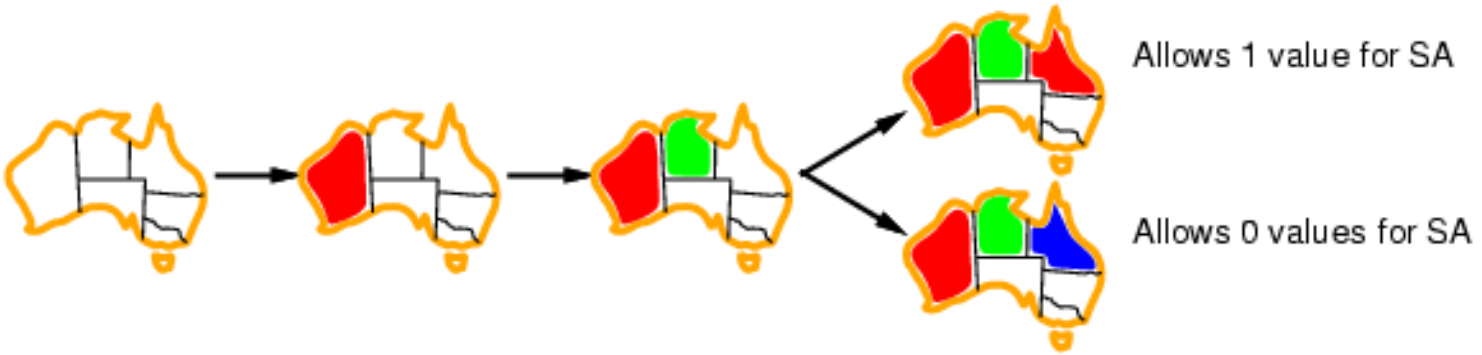






3. *A legkevésbé korlátozó érték* ötlete

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

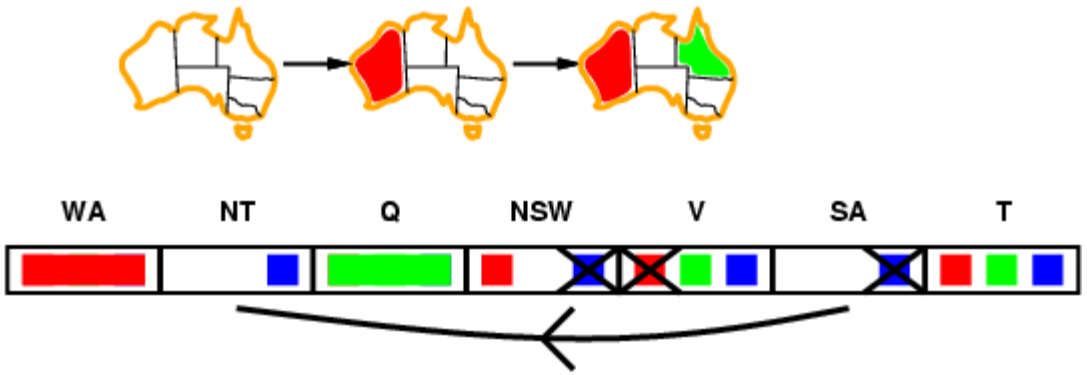


Élkonzisztencia

X → *Y* él konzisztens akkor és csak akkor, ha

X minden x_j értékére létezik Y-nak valamilyen megengedett y_k értéke

Ha X értéket veszít, a szomszédjait (akikkel valamilyen kényszerkapcsolatban van) újra kell ellenőrizni



Az élkonzisztencia-ellenőrzés alkalmazható előfeldolgozó lépésként a keresés megkezdése előtt, vagy a keresési folyamat minden egyes hozzárendelését követő terjesztési lépésként

CSP lokális kereséssel

(Hegymászó, szimulált lehűtés, …, lásd később)

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

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

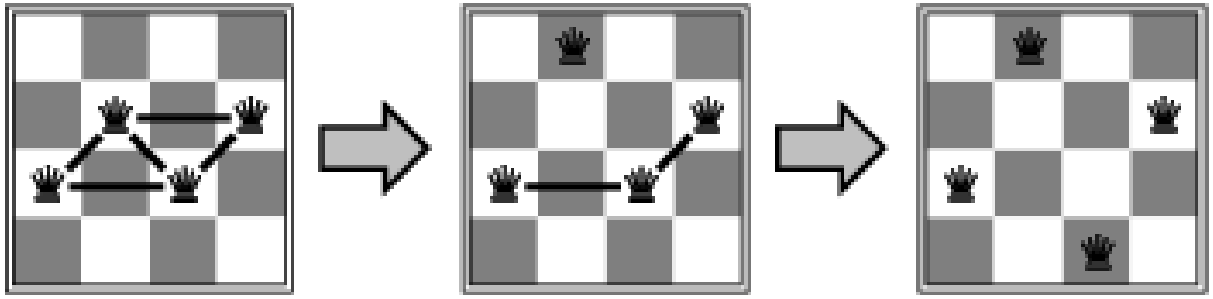
Változó szelekció: véletlen módon, bármely konfliktusban lévő (valamelyik kényszer sérül) változót választhatjuk

****

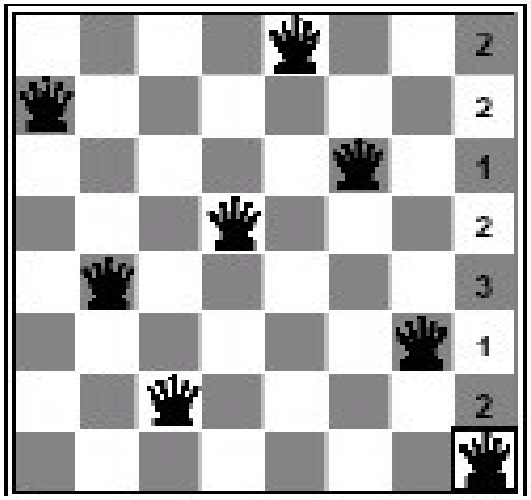
Min. konfliktus heurisztika:

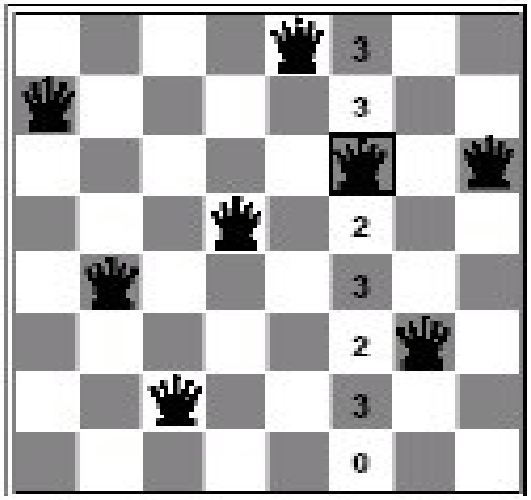
azt az értéket állítjuk be, amely a legkevesebb számú korlátot sérti, pl. hegymászó: *h(n)* = sérült korlátok száma, *h*(*n*) csökkentése a cél (itt most lefele mászunk a völgybe)

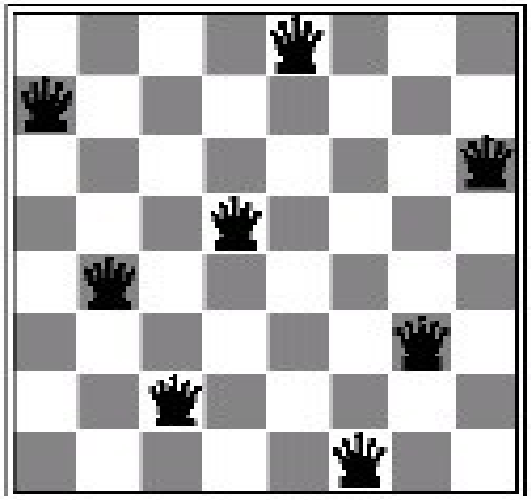
h(n) = a támadások száma



CSP lokális kereséssel / Min. konfliktus heurisztika







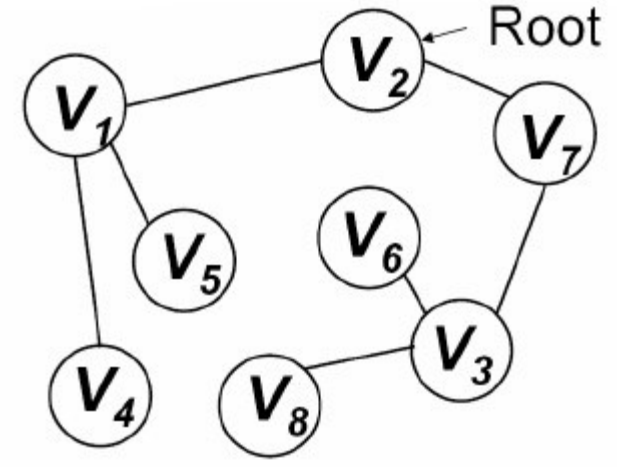
Min. konfliktus heurisztika nagyon hatékony, nagy valószínűséggel gyorsan old meg nagyon nagy problémaeseteket (10 millió királynő!).

CSP struktúrája – miben segíthet?

CSP gráfja: független komponensek … (komplexitás-számítás)

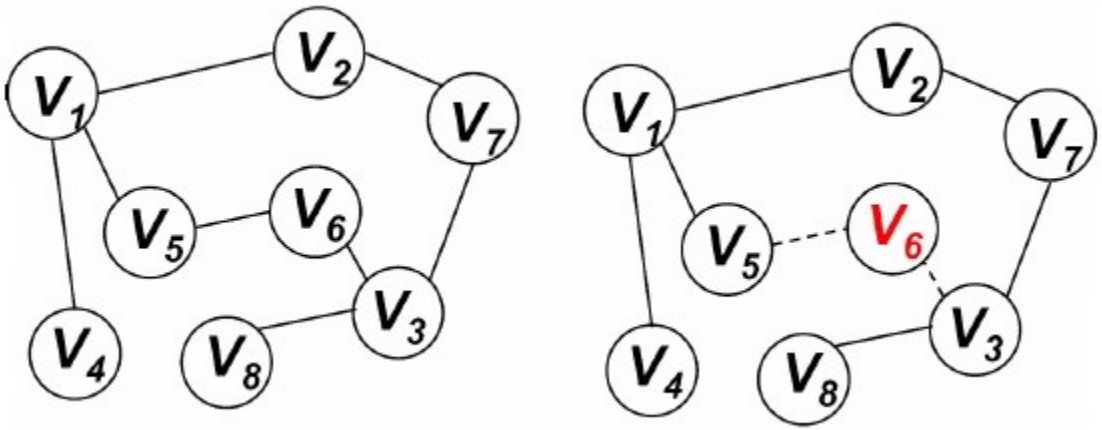
CSP fa gráf: megoldás könnyű, változószámban lineáris

válasszunk egy levelet gyökérnek (lokális keresés kiinduló áll.)
élkonzisztencia-nyesés gyerekektől a szülőik felé



CSP struktúrája – miben segíthet?

CSP hurkos gráf: megoldás általánosságban NP nehéz



Gráf CSP konvertálása fába:
vágóhalmaz
fa-**dekompozíció**

Megoldáskeresés
V_6 minden értékére