2. visszalépés

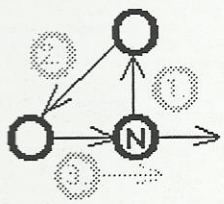
1. redukciós operátorok: csak F címkéjû levelekre alkalmazható!

elõállítjuk a részproblémákat, s felfûzzük az akt. hiperútra az adott csúcshoz, mint szülõhöz (szülõnél egy mutató fog az 1. gyermekre mutatni).
Szülõnél operátor törlése (mert azt már alkalmaztuk).

Gyermekek: megjelenik az adott probléma
szülõre visszamutató mutató
mindegyiknél köv. mutató a testvérre
felcímkézzük
F címke esetén: összes alk. red. op. felvétele

2. visszalépés: csak N címkéjû levelekre alkalmazható!

Ha a gyermekek közt megjelent egy N címkéjû csúcs, akkor visszalépés: valamilyen más redukcióval kell a hiperutat megoldani.
Redukciós operátor hatásának visszavonása:



Kitöröljük a szülõ gyermekeit: az N címkéjû csúcsban van egy visszamutató a szülõre (1.), ahonnan elérhetõ az 1. gyermek (2.). Innen van egy mutató a következõ testvérre, ezáltal a szülõ gyermekeit be tudjuk járni, s ki tudjuk õket törölni.

A visszalépés mûveletét kell még akkor is alkalmaznunk, amikor az adott csúcsban elfogytak az alkalmazható operátorok. F címkéjû a csúcs, õt kellene redukálni, de már az összes operátor elfogyott: ekkor is visszalépés.

A vezérlõ:

1. inicializálás:

S maga a probléma
null (gyermek)
null (testvér)
null (szülő)
összes alk. red. op.
F

ha a címka F, akkor keresésrõl van szó

2. keresés (pszeudó-kód):

```
1        van = választFlevél(n); //(*)
2        while (van)
3        {
4           op = választRedOp(n);//(*)
5           if (op != 0)
6           {
7              GY = alk(n, op);
8              hiperútbõvítés(GY, n, op);
9              if (Ncímkéjû(n, m)) visszalép(m);
10          }
11          else visszalép(n);
12
13          van = választFlevél(n);
14       }
```

magyarázat:

1 egy F címkéjû levéllel tér vissza, ha van ilyen

2 addig dolgozunk, míg van F címkéjû csúcs

3

4 a kiválasztott csúcs esetében választunk egy operátort

5 ha van alkalmazható operátor, akkor (7-8-9), különben meg visszalépés

6

7 elõállítjuk a csúcs gyermekeit (GY)

8 a gyermekeket felfûzzük: maga a probléma, szülõ mutat az 1. gyermekre,
gyermekek köv. mutatója a testvérre mutat,
gyermekek visszamutatnak a szülõre,
felcímkézzük a gyermekeket

9 a szülõtõl bejárjuk a gyermekeket, s N címkéjût keresünk
ha találunk (m-ben adja vissza), akkor visszalépés

10

11 ha az 5-ös pontban nem találtunk alkalmazható operátort, akkor visszalépés

12

13 keressük a következõ F címkéjû csúcsot.

14

Ha a `start` csúcsból lépünk vissza: sikertelen a megoldáskeresés.

Beszélhetünk szisztematikus ill. heurisztikus keresésről. Ezen megoldáskeresõk a (*)-gal jelölt két választásban térhetnek el lényeges módon (`valasztFlevel`, `valasztRedOp`).
Heurisztikus esetben hogyan válasszunk? HA van becslés a megoldás költségére, akkor válasszuk a LEGNEHEZEBBET! A `valasztFlevel()` válassza a legnehezebb problémát!

2.4.2. Keresõgráffal megoldást keresõk

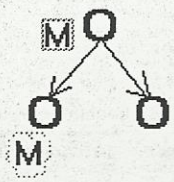
Állapottér-reprezentációs esetben akkor volt meg a megoldás, amikor a tesztelésre kiválasztott csúcs eleget tett a célfeltételnek.

Problémaredukciós esetben a terminálás figyelésére bevezetjük a címkemódszert. Háromféle címkét fogunk alkalmazni:

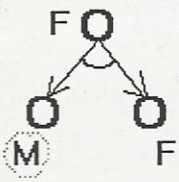
1. M (megoldható *egyszerû* probléma)
2. N (nem megoldható *egyszerû* probléma)
3. F (a többi problémának ez a címkéje, azt jelzi, hogy még folyik a munka)

Két eljárás: - címkézõ (az elõbb a visszalépéses módszernél tulajdonképpen egy ilyet használtunk. Amikor az adatbázisba bekerül egy új csúcs, akkor azt címkézzük).

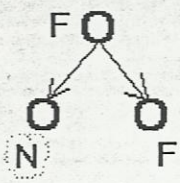
- címkemódosító (a címkék módosíthatóak. Ha a címkézõ egy F-tõl különbözõ címkét ad egy csúcsnak, akkor ennek a szülõre nézve következményei lesznek).



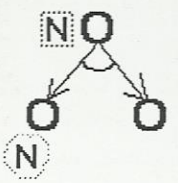
ha õ VAGY-gyermek, akkor a szülõ is M lesz



ekkor még nem tudunk mit mondani!



még nem tudunk mit mondani!



ha õ ÉS-gyermek és N címkéjû, akkor a szülõ is N lesz

Ha a csúcsnál egy F-tõl különbözõ címke jelenik meg, akkor indul a címkemódosító eljárás a csúcs szülõjére nézve.

Ha a `start` csúcsban M címke jelenik meg: van megoldás, elõállt a megoldás

Ha a `start` csúcsban N címke jelenik meg: nem oldható meg

Egyedül a `start` csúcs címkéjét kell tehát figyelni! Ha F: folytatjuk a keresést. Ha M, N: sikeres, sikertelen terminálás.

Az adatbázis ebben az esetben egy keresõgráf. Hiperutak, melyek a `start` csúcsból indulnak. Ezek valamelyikéből remélünk megoldást. ÉS-VAGY fagráfok.
A hiperutak közül választunk egyet, másrészt ha kiválasztottuk: ezen a hiperúton a még meg nem oldott problémák közül IS választani kell.

Itt is beszélhetünk szisztematikus ill. heurisztikus keresésrõl. Heurisztikus esetben: melyik hiperúton reméljük a megoldást, ill. azon belül a legnehezebb megoldást választjuk ki.

Ha kiválasztottuk: kiterjesztjük. Az *összes* redukciós operátort alkalmazzuk rá. A leveleket címkézzük, s ha kell módosítjuk a szülõt is. Megoldás: a `start` csúcs címkéje M-re vált.

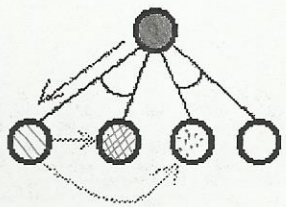
Az N címkéjû csúcsokból induló hiperútrészleteket töröljük.

Az A-algoritmus valamilyen módosítását fogjuk alkalmazni. Ez lesz az:

2.4.2.1 AO-algoritmus

Adatbázis: *s*-bõl induló hiperutak (keresés során már felderített hiperutak)

csúcsinformáció:
1. maga a probléma
2. 1. gyermekre mutató mutató
3. következõ ÉS testvér **(új)**
4. következõ VAGY testvér **(új)**
5. szülõre mutató mutató
6. kiértékelõ-függvény **(új)**
7. címke
8. információ a redukciós operátorokról



Mûvelet: kiterjesztés mûvelete (pszeudó-kóddal):

```
while (vanMégRedOp(n, op))
{
   GY = alk(n, op);
   hiperútbõvítés(GY, n, op);
}
címkemódosító();
kiértékelõfüggvényMódosító();
NcímkéjûHiperutakEltávolítása();
```

Vezérlõ: (pszeudó-kóddal):

1. inicializálás:

S
maga a probléma
null (1. gyermek)
null (köv. ÉS testvér)
null (köv. VAGY testvér)
null (szülő)
kiértékelő-függvény
F (címke)
összes alk. red. op.

2.
```
van = választHiperút(h);
while (van && (címke(h)!=M))
{
   választFlevél(h, n);
   kiterjeszt(h, n);
   van = választHiperút(h);
}
```

Kiértékelõfüggvény:

$$f(n) = \begin{cases} 0, \text{ ha M címkéjû} \\ \square, \text{ ha N címkéjû} \\ h(n), \text{ ha F címkéjû és } n \text{ levél} \\ \min_r \left\{ r(n, \{n_1, \ldots, n_k\}) + \sum_{i=1}^{k} f(n_i) \right\}, \text{ ha F címkéjû és } n \text{ nem levél} \end{cases}$$

Vissza a lap tetejére

Mesterséges intelligencia 1

Elõadó: Dr. Várterész Magdolna

3. Kétszemélyes, teljes információjú játékok

3.1. A játékok osztályozása

A játékokat két nagy csoportba tudjuk sorolni:

1. szerencsejátékok
   (itt minden a véletlen mûve, pl.: rulett)
2. stratégiai játékok
   (a játékos befolyásolhatja a játék kimenetét, pl.: sakk, malom, amõba, üzleti "játékok", harci "játékok")

Neumann János maga is foglalkozott ezzel. Harsányi János 1994-ben közgazdasági Nobel-díjat kapott; a játékelmélet az õ nevéhez fûzõdik.

Stratégiai játékok osztályozása:

résztvevõk száma:	2,3,...,n személyes játékok (néha a véletlent is személynek veszik)
véges játékok:	olyan stratégiai játékok, amelyben minden állásban véges sok szabályos lépés közül lehet választani, s véges sok lépés után véget ér
zérusösszegû:	ha nyeremények + veszteségek = 0 (beszélhetünk nem zérusösszegû stratégiai játékokról is)
véletlen szerepe:	1. sztochasztikus (pl. bridge) 2. determinisztikus (pl. sakk)
teljes információjú:	tudom, hogy a másik mit lépett. A játékosok a játékkal kapcsolatos összes információt ismerik, s felváltva lépnek.

Leírás: játékszabályok megadása
Milyen játékállások fordulhatnak elõ, az egyes állásokban milyen szabályos lépések vannak, mik ezeknek az elõfeltétele.
Célfeltétel: mikor kell befejezni, milyen állásban, s ekkor kit kell gyõztesnek kikiáltani.

A kétszemélyes játékot is le kell írni. Ehhez a következõ leírást alkalmazzuk:

3.2. A játékok reprezentációja és a reprezentáció szemléltetése

H legyen a játék állásainak a halmaza, {A, B} a két játékos.

All = { (h, x) | h ∈ H, x ∈ {A, B} } - ez az állapottér, ahol *x* azt jelzi, hogy ki következik

Kezdõállapot:

k = (i, I) ,ahol i ∈ H kezdõállás, I ∈ {A, B} kezdõ játékos

Végállapot:

V = { (h, x) | h ∈ H végállások, x nyertes v. vesztes } megállapodás kérdése, hogy *x* nyertes vagy vesztes

Lépés:

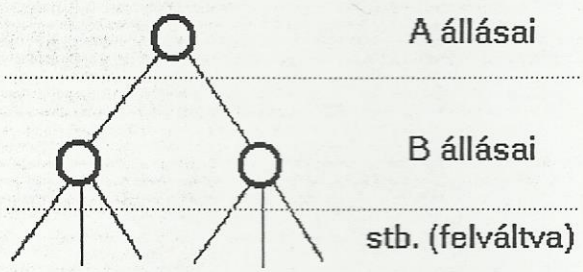
l: H → H

elõfeltétel$_l$(h) az egyes lépésekhez elõfeltételek is tartoznak

Operátorok:

O = { o: (h, x) → (h', x') | l: h → h', elõfeltétel$_l$(h) teljesül, h,h' ∈ H és x' = x ellenfele }

A játék állásait egy reprezentációs gráffal tudjuk szemléltetni. A reprezentációs gráf egy játékgráf:



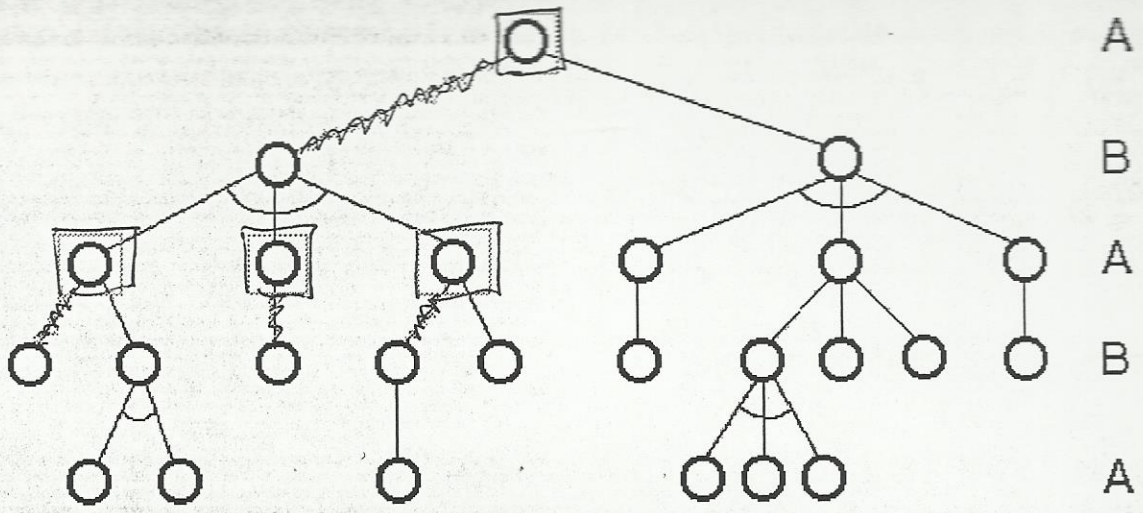
Véges játék esetén véges a gráf is. Minden út benne véges.
Körök: a játékszabályok ezt általában nem engedik, vagy adnak rá valami korlátozást (a köröket fává egyenesítéssel is fel lehet oldani).

A `start` csúcsból egy levélbe vezetõ út: egy lehetséges játszma.

3.3. A stratégia

Egy játékos számára hogyan lehetne egy teret adni, ami befolyásolná a játék kimenetét? Stratégia. Ez mindig egy (1) játékoshoz tartozik. Minden, a játék során elõforduló állapotban amikor õ következik a stratégia megmondja, hogy mit lépjen. A stratégia tehát egy *döntés*.
Jó a stratégia, ha minden állapotra van terv, *bármit* is lép az ellenfél.
Lehetséges hiperutak: a játékos különbözõ stratégiái. Ahány különbözõ hiperút, annyi különbözõ lehetséges stratégia a játékos számára.

Gyûjtsük össze a játék során elõfordulható összes állapotot:



Tegyük fel, hogy például *A* szeretne nyerni. Ekkor a `start` állapotnak szerepelnie kell a stratégiában. Menjünk tovább a zölddel jelölt részen. *B* viszont (az ellenfél) bármit léphet, ezért az ÖSSZES olyan állapotnak is szerepelnie kell a stratégiában amelyet *B* léphet, hiszen az ellenfél minden válaszlépésére reagálni kell tudni!

Ahol *A* következik lépni: választunk a szabályos lépések közül. VAGY lépések vannak, lehet választani. *B* lépései ÉS élek. Ha *B* lép, akkor a stratégiába fel kell venni mindazon állapotokat, ahova *B* léphet.

! Át kell alakítani a játékfát az ADOTT játékos alapján ÉS-VAGY gráffá. Ebben kell hiperutakat keresni! Az adott játékosnál VAGY élek lesznek, az ellenfélnél pedig ÉS élek.

Hogyan választunk: úgy, hogy a játék kimenetele lehetõleg nyeréssel végzõdjön, ne veszítsünk.

Nyerõ stratégia: a stratégiával játszva a másik játékos lépéseitõl függetlenül garantáltan nyerünk! Ha a levélelemek mind nyerõállások, akkor a stratégia nyerõ.
Feladat: a stratégiák közül válasszunk nyerõ stratégiát.

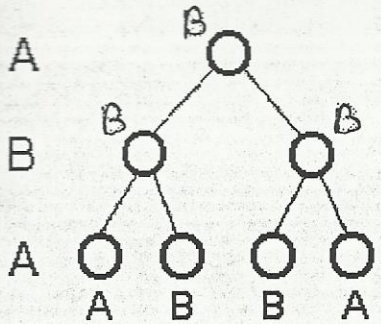
3.4. Adott állásban a következõ lépés kiválasztásának módszerei

Minden kétszemélyes, teljes információjú játék esetén az egyik (és csakis az egyik) játékos számára garantáltan van nyerõ stratégia. Ha van, akkor több is lehet.
Kérdés: melyik játékos számára lesz?

Az állítás csupán azt garantálja, hogy az egyiknek van nyerõ stratégiája.

Bizonyítása konstruktív módon történik, s ehhez az egész játékfát elõ kell állítani:

A leveleket megcímkézzük azalapján, hogy ott ki nyert. Ezután visszafelé haladunk (itt középsõ szint), s megnézzük, hogy ki lép (itt B). Mivel B tud úgy lépni innen, hogy õ nyerjen, ezért B-t írunk be (szürkével jelölve). Mint látható, ez egy B nyerõ játék. A `start` csúcs címkéje határozza meg, hogy kinek van nyerõ stratégiája. Ez megkonstruálja a nyerõ játékost, s a



nyerõ stratégiát is. Címkemódszer. A címke jelöli, hogy az adott állásból kinek van nyerõ stratégiája.

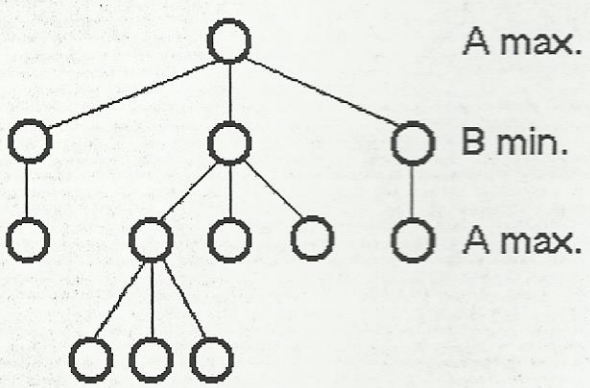
Az elõbbi címkemódszerrel akkor van gond, ha túl nagy a játékfa (nem tudjuk elõállítani), s így explicit módon nem tudjuk meghatározni hogy ki nyer. Ekkor használjuk a következõ módszerek egyikét:

3.4.1. Mini-max módszer

Nem építjük fel az adott állásban a teljes játékfát (pl. nem tudjuk, mert olyan nagy lenne).

Amit tehetünk: néhány lépésre elõre kombinálunk. Az adott állásból kiinduló játékfának csak egy részét állítjuk elõ. A kialakuló állások (levélelemek) nem végállások, de mégis valahogy minõsíteni kellene õket, hogy tudjunk választani.

Valamilyen heurisztikára lesz szükség, amely megmondja, hogy az adott játékos számára az adott állás mennyire jó, mennyire ígéretes. Minél jobb az állás, annál nagyobb értéket rendelünk hozzá.
Döntetlen: 0 körüli érték.
Ha az ellenfél számára jobb: negatív érték.



Az azonos szinten lévõ levélelemeket kiértékeljük (heurisztika). Az õsöknél ezt vesszük figyelembe, ill. azt, hogy ki következik lépni.

Ha *A* következik:	a legnagyobb értékû gyermeket fogjuk választani (nekünk az a legjobb)
Ha *B* következik:	a legkisebb értékût választjuk (ez *B*-nek a legjobb, ezért úgy számítunk, hogy õ azt fogja lépni, ami neki a legjobb)

Ez a módszer az alapja minden kétszemélyes játékkal kapcsolatos tanácsadásnak!

3.4.2. Nega-max módszer

Egy állás kiértékelõfüggvénye az egyik játékos szempontjából értékeli ki az állásokat. Amennyire jó nekünk egy állás, annyira rossz az ellenfélnek (és fordítva).
B számára egy állás heurisztikája pontosan az ellentetje az *A* heurisztikájának.

$$-h_A(n) = h_B(n)$$

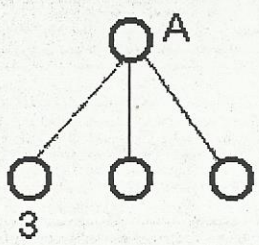
Nem kell szintrõl-szintre megvizsgálni, hogy ki következik! Levélszinten az ellenfél szerinti kiértékelõfüggvényeket adjuk meg. Ez a játékos pontosan a szülõ szint ellenfele. A szülõ szinten a lenti értékek az ellentettjére változnak, s ezek közül a maximumot vesszük.

A módszer elõnye: nem kell egyik szinten minimumot, másikon maximumot venni. Eredménye amúgy azonos az elõzõ módszerével. Mindig a maximumot kell venni.

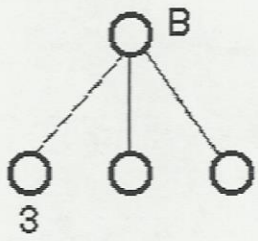
3.4.3. Alfa-béta vágás

Generálás közben fog kiderülni, hogy mekkora részlet kell.
Amint van a szülõnek gyermeke, a szülõ pontos értékének alsó vagy felsõ becslését meg tudjuk adni.

Max. szülõ ☐	alsó becslését tudjuk megadni a végleges értéknek
Miṇ. szülõ ☐	felsõ becslését tudjuk megadni a végleges értéknek

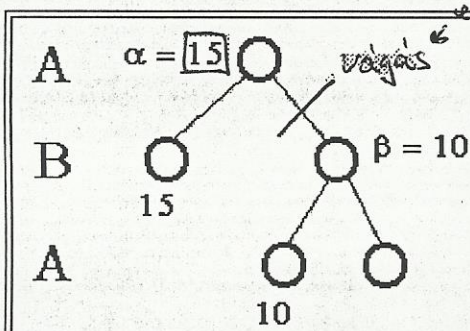
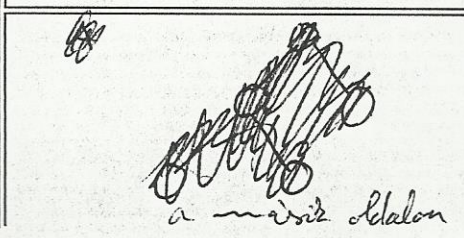


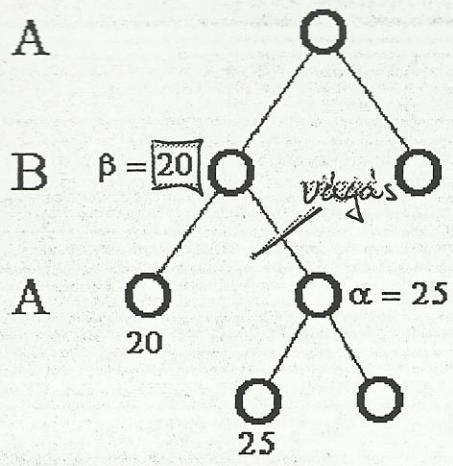
a csúcs értéke legalább 3 lesz



a csúcs értéke legfeljebb 3 lesz

Lehet, hogy még nem állítottuk elõ az összes gyermeket, de már tudunk valamit mondani a szülõrõl.

	A zölddel jelölt résznél vágunk, mivel azt a részt már nem kell elõállítani, hiszen már biztos, hogy nem fog "beleszólni" a felette lévõ részbe. ☐-vágás, mert az ☐ alatt vágtuk el
	A zölddel jelölt résznél vágunk, mivel az az alatti rész



végleges, pontos értékének a kiszámítása már biztos hogy nem fogja módosítani az eredményt.

β-vágás, mert a β alatt vágtuk el

Ez a módszer is ugyanazt a javaslatot eredményezi, mint a mini-max módszer.
Elõnye: csökken a játékfa mérete, mert bizonyos részeket nem kell elõállítani.

$\alpha \geq \beta$ - ha ez teljesül, akkor lehet vágni

toc

Vissza a lap tetejére