Vizsgakérdések az MI előadás anyagából. 2011.

1. A Russel féle négy cél MI rendszer
2. Megoldás keresés az állapottérben: hegymászó keresés, Hanoi tornyai példával bemutatva.
3. Dekompozíciós módszer, Hanoi tornyai példával bemutatva
4. Közönséges irányított gráfok tulajdonságai
5. ÉS/VAGY gráfok tulajdonságai, hiperutak
6. ÉS/VAGY gráfok átalakítása irányított gráffá
7. A visszalépéses keresés két változata (algoritmusok és a működés ismertetése)
8. A gráfkeresés alapalgoritmusa és módosításai
9. Az általános gráfkereső algoritmus (algoritmus és a működés, jellemzők ismertetése)
10. Neminformált gráfkeresések
11. Az A, A*, Ac algoritmusok, tulajdonságaik
12. A* hatékonysága: memória és futásidő.
13. Állapottér reprezentáció: fogalmak, alkalmazott keresőrendszerek
14. Probléma redukció, boroskancsók példával illusztrálva.
15. Nim játék leírás, nyerő stratégia
16. Nyerő stratégia teljes információjú játékoknál (irányított gráffal, ÉS/VAGY gráffal)
17. Minimax, negamax algoritmusok játékfákon
18. Alfabéta algoritmus (az algoritmus és a működés ismertetése)
19. Ítéletkalkulus alapfogalmai
20. Tételbizonyítás az ítéletkalkulusban, konjunktív normálforma, rezolúció.
21. Predikátumkalkulus alapfogalmai
22. Klózformára alakítás a predikátumkalkulusban
23. Egyesítési algoritmus, rezolúció.
24. Szakértői rendszerek gyakori tudásábrázolási technikái.
25. Fuzzy halmazelmélet fogalmai, alapvető műveletei.
26. Fuzzy logika fogalmai, alapvető műveletei.
27. Neurális hálók általános jellemezői, működésük.
28. Evolúciós algoritmus alapfogalmak, műveletek, általános algoritmus és működése.

1. A Russel féle négy cél MI rendszer

Russel 1996: négyféle MI rendszer létrehozás a cél
(Bertrand Russell angol matematikus, logikus, filozófus és szociológus, Kingston III. earlje, Nobel-díjas közéleti személyiség.)

1. Az emberhez hasonlóan gondolkozó rendszerek
Szükséges: Emberi gondolkodás megértése
kognitív pszichológia: folyamat megértés, leírás
MI: működő modell készítés
kognitív tudomány: emberi megismerés,
gondolkodás modellezés (MI+ kogn. pszich.)

2. Az emberhez hasonlóan cselekvő rendszerek
intelligens program készítés
int. ellenőrzés: Turing teszt (kérdező – ember/gép?)
szükséges képességek:
(következtetés, észlelés, mozgatás)
természetes nyelvmegértés,
tudásreprezentáció, aut. következtetés,
gépi tanulás, szgépes látás, robotika

3. Racionálisan gondolkodó rendszerek
A helyes gondolkodás törvényeit alkalmazni
Szillogizmusok, következtetési szabályok (modus ponens)
formális logika
Logicista irányzat: formális logika felhasználása
problémák: reprezentáció (informális, bizonytalan)
futási idő

4. Racionálisan cselekvő rendszerek
Racionális cselekvés egy célért
Ágens fogalom: érzékelő és cselekvő program
MI: racionális ágensek tanulmányozása, készítése
szükséges: racionális gondolkodás
ésszerű cselekvések
ágens - robot fejlesztés

2. Megoldás keresés az állapottérben: hegymászó keresés, Hanoi tornyai példával bemutatva.

Ahhoz, hogy a számítógép értelmezni tudja, hogy melyik csomópontban áll és hogy merre léphet tovább, valamilyen módon azonosítani kell számára a csomópontot. Erre a legcélszerűbbnek az látszik, hogy a vektor elemeit össze adjuk (összeg függvény), és azt a következő lépést választjuk, amelyik a szabályok értelmében a pillanatnyi állapotból következik és legkisebb az összegfüggvény értéke. Azért célszerű ezt választani, mert amint megjelenik az (1,1,1) állapot az algoritmus végpontot ér el. Ennél kisebb összérték nem lehetséges.

Az algoritmus:

1. Pill_állapot = (3,3,3) „kezdő állapot”
2. While (Pill_állapot != (1,1,1){
3. Szabályok alapján következő lehetséges csomópontok meghatározása
4. Pill_Állapot = min_sum(lehetséges állapotok)
}

END

Látható, hogy a szabályokat addig alkalmazzuk a pillanatnyi állapotra, amíg minden elérhető csomópontot meg nem határoztunk, majd kiválasszuk közülük a legkisebb összegfüggvény értékűt, és a pillanatnyi állapotot felülírjuk ezzel az új csomóponttal. Addig ismételgetjük a műveletsort, amíg a célállapotot el nem érjük.

Ez a módszer igen egyszerű, de sajnos nem mindig ér el végpontot. Vannak olyan esetek, amelyekben nem találja meg a megoldást. Heurisztika hiányában például egy negyedik korong rendszerbe helyezésekor körbenjár, azaz három csomópont által meghatározott körből sosem talál ki. Fontos hibája, hogy nem az optimális utat találja meg (**optimális út**: az a megoldás, amelyik a legrövidebb (legkisebb költségű - ez később fontos lesz) útvonal a start és a cél között) Ha valaki az algoritmus alapján végigköveti a bejárt utat, maga is könnyedén meggyőződhet arról, hogy egy ponton a rendszer eltéved és felesleges csomópontokat érint. A szabályok között szerepel, hogy egy csomópontból nem léphetünk vissza a szülő objektumba, azaz a megelőző csomópontba. Így kizárjuk a két pont közötti ingázást.

3. Dekompozíciós módszer, Hanoi tornyai példával bemutatva

A dekompozíció általánosítása a redukciónak: egy feladatot több részfeladatra bontunk, majd azokat tovább részletezzük, amíg nyilvánvalóan megoldható feladatokat nem kapunk.

Wikipédia: A **dekompozíció** azt jelenti, hogy nem két, hanem több részre bontjuk a problémát, azaz a problémát egy esetleg összetettebb, de strukturáltabb problémával reprezentáljuk.

Valamilyen M problémarészt általában a legtöbb probléma esetén találhatunk, sőt a Pólya-heurisztika szerint általában ez az első feladatunk. A gondot általában az okozza, hogy az N problémarészt általában még mindig túl nehéz megoldani, és olyan dekompozíciót több okból is nehéz találni, amely megfelelően osztaná fel a problémát. Ha könnyű a dekompozíció, az valószínűleg azt jelenti, hogy M kis súllyal vesz részt az eredeti P probléma felépülésében, és a probléma nehézségének nagy részét még mindig az N ismeretlen megoldású problémarész teszi ki. Azaz a dekompozíció maga is egy nehéz probléma, amire nem lehet általános receptet adni. Ha azonban véletlenül észrevesszük a dekompozíció lehetőségét, az nagyon gyümölcsöző eredménnyel szokott járni.

Részfeladatokra bontjuk a problémát:
3-as rúdról A, B a 2-es rúdra
A C-t az 1-es rúdra, majd
A,B-t az 1-es rúdra átvisszük.
Dekompozíciós reprezentáció:
<n,i,j,k> - n korongot mozgatunk, az i-dikről a j-dikre a k-dik rúd segítségével.
A dekompozíció operátor: <n-1,i,k,j>,<1,i,j,k>,<n-1,k,j,i>
Többszöri alkalmazás – egyszerű problémákra bontás

4. Közönséges irányított gráfok tulajdonságai

ÉS/VAGY GRÁF (HIPERGRÁF)

Egy csúcspontnak lehetnek ÉS utódai és VAGY utódai..

Egy ÉS kapcsolat akkor valósulhat meg, ha a kapcsolat
összes tagja megvalósul, a VAGY kapcsolat igazzá
válásához elég egy tagjának igaz volta.

Ábrázolás:
körívvel kötjük össze az ÉS utódokhoz vezetı éleket
és nem kötjük össze a VAGY utódhoz vezetıket.

Ha egy gráfban minden csúcspont utóda VAGY utód, akkor az adott gráf egy közönséges irányított gráf.

Minden olyan területen alkalmazható, ahol a problémák az
ÉS és VAGY művelet segítségével fogalmazhatóak meg.

Irányított gráf: csúcsok és irányított élek együttese.
N csúcsok halmaza
A=NxN élek halmaza (rendezett csúcspárok), →,←,↔
(n,m) n-ből m-be irányított él n: szülő, m: utód
Egy csúcsból induló élek száma
$|\{(n,m)\in A\}|\leq\sigma$ minden n ∈N csúcsra (σ tulajdonság)
Élek költsége (műveleteknél) c:A →R
$c(n,m)\geq\delta>0$ minden (n,m) ∈A élre (δ tulajdonság)
Az R=(N,A,c) gráfok a σ és δ tulajdonsággal: δ-gráfok

út hossza: élek száma
Erős komponensek: csúcshalmaz, minden csúcspár közt
oda-vissza út vezet
Irányított kör (n =m)

↓

5. ÉS/VAGY gráfok tulajdonságai, hiperutak

ÉS/VAGY GRÁF (HIPERGRÁF)

Az ÉS utódokhoz vezető élköteg a hiperél, vagy másképpen
k-adrendű él.

Egy k-adrendű él egy csúcsból egy k elemű csúcshalmazba
vezet:
(n,M) az a hiperél, amelyik az
n csúcsból az M = {m1,...,mk}
csúcshalmazba mutat.
Az n csúcs a szülő, M az utódhalmaz, elemei az utódok.
nevezzük.

Az olyan gráfot, amelyik hiperéleket tartalmaz, hipergráfnak,
vagy másképpen ÉS/VAGY gráfnak nevezzük

Hiperút:
egy olyan részgráf, amelynek minden csúcsából egyetlen hiperél indul ki.

Ha a hiperút kezdőcsúcsa n, végpontjainak halmaza K, akkor n→K jelöli az n-ből K-ba vezető hiperutat.

Megoldásgráf:
Egy olyan út, amely a startcsúcsból indul ki és a célcsúcsok halmazába vezet.
(Egy ilyen út az ÉS/VAGY gráfban egy részgráfot határoz meg.)

KERESÉS ÉS/VAGY GRÁFOKON

A közönséges gráfoknál alkalmazott stratégiák nem alkalmazhatóak változtatás nélkül, hiszen ÉS/ VAGY gráfok esetén nem csupán egy, a startcsúcsból egy terminális csúcsba vezető utat keresünk, hanem egy részgráfot, amely a probléma megoldásgráfja.

6. ÉS/VAGY gráfok átalakítása irányított gráffá

ÉS/VAGY gráfok kezelése nehézkes.

Átalakíthatóak irányított gráfokká.

Új csúcsok bevezetése: utódcsúcs = átalakítandó hiperél utódainak halmaza.

E műveletet kiterjesztjük a kezdőcsúcstól a célig.

7. A visszalépéses keresés két változata (algoritmusok és a működés ismertetése)

Visszafelé haladó keresés
Ha a problématér a cél felől nézve egyszerűbb (kevesebb alternatívát mutat), mint a start felől nézve, akkor érdemes visszafelé haladó keresést alkalmazni.
A talált megoldási utat azonban a starttól a cél felé haladva kell értelmezni. (Van-e az útnak inverze?)

Visszafelé haladó keresés feltételei
A műveletek invertálhatóak legyenek (legalábbis a visszafelé haladó keresés által alkalmazottak).
Konkrét célállapotot kell választani. (Ettől a talált megoldás költsége is függ.)
Mit tegyünk, ha a fenti két feltétel nem áll fenn, de visszafelé haladó keresést akarunk megvalósítani? → Probléma redukció

Lényeg: az alkalmazott produkciós szabály hatása vissza vonható. A reprezentációs gráfban vissza lehet lépni.
Globális adatbázis tartalma:

a repr. gráf egyetlen útja. Ehhez nyilván kell tartani
feladattól függően:
utolsó csúcsot/ minden csúcsot és/vagy élt.
„ minden csúcsnál a ki nem próbált éleket.

8. A gráfkeresés alapalgoritmusa és módosításai

Lényeg: egyszerre több utat tart nyilván és mindig a
legígéretesebbet folytatja.
„ Globális adatbázis: a reprezentációs gráf egy
részhalmaza
Keresőgráf (a részhalmaz)
Nyílt csúcsok, az utak vége, amelynek lehetnek utódai (NYÍLT)
Zárt csúcsok: ismerjük már az utódait
Művelet: Γ operátor
egy csúcs összes utódát előállítja és kiterjeszthetjük velük a
keresőgráfot.
A gráfkeresés alapalgoritmusa
Algoritmus
Procedure gráfkeresés
1. G ←{s}; NYÍLT ←{s}
2. While not üres(NYÍLT) loop
3. n ←elem(NYÍLT)
4. If cél(n) then exit endif
5. G ←G∪Γ(n)
6. NYÍLT ←NYÍLT-{n}, NYÍLT←NYÍLT∪Γ(n)
7. Endloop
end

↓

9. Az általános gráfkereső algoritmus (algoritmus és a működés, jellemzők ismertetése)

A gráfkeresés algoritmusa
1. A keresőgráf egy fa gráf (G). Azok a levélcsúcsok, amelyeknek még nem vizsgáltuk az utódait, alkotják a nyílt halmazt (N). A keresőgráf többi csúcsa zárt. A csúcsokhoz nyilvántartjuk az ősét és az oda vezető út költségét (g(n)) is.
2. A keresőgráf és a nyílt halmaz kezdetben csak a startcsúcsból áll.
3. Ha a nyílt halmaz üres, nincs megoldás, vége.
4. Kiválasztunk egy n csúcsot a nyílt halmazból, úgy hogy f(n) értéke minimális legyen. (n eleme N-nek)
5. n-et kivesszük a nyílt halmazból.
6. Ha ez megoldás, akkor vége.
7. Kiterjesztjük az n csúcsot: utódai alkotják az M halmazt.
8. Minden M-beli elemre (m elme M-nek) megvizsgáljuk, hogy m szerepel-e már a keresőgráfban (m eleme-e G-nek)
9. Ha nem, vagy ha g(m) > g(n)+c(n,m) vagyis olcsóbb utat találtunk, akkor az új utat jegyezzük meg a régi helyett, és g a nyíltba is belekerül.
10. Folytatjuk a 3. pontnál.

Módosítások:
Költségfüggvény egy feszítőfabeli útra g: G →R.
g(n) megadja az s-ből n-be vezető út költségét
Az optimális g* a teljes reprz. gráfon értelmezett, így egy n∈G csúcsra g(n)≥g*(n).
Konzisztencia a feszítőfa és g közt:
g a feszítőfabeli utak költségét mutatja

10. Neminformált gráfkeresések

Neminformált gráfkeresések
Az ált. gráfkereső algoritmus változatai
Nincs plusz információ a kiértékelő fgv-ben (vak keresés)
Keresés algoritmusok
Mélységi
Szélességi
egyenletes

Mélységi keresés
Ha minden c(n,m) költség egységnyi, és a kiértékelő fgv:f(n)=- g(n) minden n∈NYÍLT csúcsra.
(g(n) úthossz=költség)
Mélységi korlát: MK ≥ g(m). Nem talál mindig megoldást.
Nemdeterminisztikus: azonos mélységű csúcsok választása
Hasonló a visszalépéses kereséshez (azonos ha a repr. gráf fa)

Egyenletes keresés (uniform-cost)
Ha a kiértékelő fgv:
f(n)= g(n) minden n∈NYÍLT csúcsra.
Mindig talál megoldást, ha létezik.
Legoptimálisabb út.
Nemdeterminisztikus: azonos mélységű csúcsok választása
Egy csúcsot csak egyszer terjeszt ki
Ha c(n,m)=1, azonos a szélességi kereséssel.

11. Az A, A*, A^c algoritmusok, tulajdonságaik

Lemma:
Ha az A algoritmus nem terminál, akkor minden NYÍLT halmazba került csúcs véges sok lépés után kiterjesztésre kerül.
TÉTEL:
Az A algoritmus mindig talál megoldást feltéve, hogy létezik megoldás.
Példa: 8-as kirakó játék, f(n)=g(n)+h(n) és h(n) = W(n), a jelenlegi rossz helyek száma.

A* algoritmus
Def: A* algoritmusnak nevezzük az A algoritmust, ha

a h fgv alulról becsli az opt út költségét; $h(n) \leq h^*(n)$
minden $n \in N$ csúcsra.
Ez a heurisztika megengedhetőségi tul.
Egy gráfkereső alg. megengedhetőségi tul. heurisztikával
megengedhetőnek nevezünk
Lemma
Az A* alg. által kiterjesztésre kiválasztott bármely n
csúcsra az $f(n) \leq f^*(s)$ egyenlőtlenség teljesül.

TÉTEL
Az A* algoritmus mindig optimális megoldás
megtalálásával terminál, feltéve, hogy létezik
megoldás.
Példa: 8-as kirakó játék, $f(n)=g(n)+h(n)$ és $h(n)=P(n)$,
a távolságheurisztika.
Egy hely célbeli helyétől mért Hamilton távolság.(jobbra,
balra, fel, le összesen) – ezek összege P(n).
$h(n) \leq h^*(n)$ teljesül.
(6 lépés kell az opt. megoldáshoz)

Ac algoritmus
Def: A h fgv kielégíti a monoton megszorítás feltételét ha
értéke bármely él mentén legfeljebb az él költségével csökken:
$h(n)-h(m) \leq c(n,m)$ minden $(n,m) \in A$ élre
Lemma: Ha a h heurisztikára teljesül a monoton
megszorítás, akkor egy tetszőleges n csúcsba vezető
opt. út mentén a g+h fgv. összeg értéke mon.
növekvő.
Def: Ac (következetes) algoritmusnak nevezzük az A algoritmust,
amelynek h fgv-e mon. megszorításos és minden $t \in T$ csúcsban
$h(t)=0$.

TÉTEL
Amikor az Ac alg. egy n nyílt csúcsot kiterjesztésre
kiválaszt, akkor n-be már opt. utat talált, azaz
$g(n)=g^*(n)$.
Köv.:
egy csúcs legfeljebb egyszer kerül kiterjesztésre
(zárt csúcs nem kerül vissza a NYÍLT-ba)
Opt. uton halad.
TÉTEL
Az Ac alg. opt. megoldás megtalálásával terminál,
feltéve, hogy létezik megoldás

12. A* hatékonysága: memória és futásidő.

. Az A* algoritmus hatékonysága
Memóriaigény
Memóriaigényen a terminálásig kiterjesztett, zárt
csúcsok számát értjük
Ez jó becslés a kereső fa (glob. memória) méretre
A reprezentációs gráf mérete ált. ismeretlen (lehet ∞)
Összehasonlíthatók az A* heurisztikái egy feladatnál
Egy alg. jobb a másiknál, ha az utóbbi minden olyan csúcsot
kiterjeszt, amelyet az előbbi is.

Pl. 15-ös kirakó esetén a nulla, a W és P heurisztikánál
$0(n) \leq W(n) \leq P(n)$ minden $n \in N$ csúcsra, és
P a legjobb, legkiesebb memóriaigényű

Memóriaigény
A_2 jobban informált, mint az A_1, ha a célcsúcsok kivételével minden $n \in N$ csúcsra teljesül, hogy $h_1(n) < h_2(n)$, ahol $f_1=g+h_1$ (A_1), $f_2=g+h_2$ (A_2) és h_1 h_2 megengedhető heurisztikák.
A* -nál a jobban informált változatok adnak jobb eredményt (a pontosabban becslő heurisztikák), kevesebb memória szükséges

Futási idő
Az idő függ: a kiterjesztések számától + szabály kiválasztás idejétől. (Legyen állandó a szabály kiv.)
A kiterjesztések számát a kiterjesztett csúcsok függv.ben vizsgáljuk.
Hány kiterjesztést végez az algoritmus?
Legyen megoldás és legyen k db kiterjesztett csúcs
Alsó korlát: mon. megszorításos heurisztika esetén k (minden csúcsot egyszer terjeszt ki)
Felső korlát: a legbonyolultabb problémánál minden $n_1, n_2, .., n_{k-1}$ csúcshoz $2_0, 2_1, \ldots, 2_{k-2}$ különböző út vezet s-ből, →utanként egy kiterjesztés: össz 2_{k-1}

13. Állapottér reprezentáció: fogalmak, alkalmazott keresőrendszerek

állapottér: Ha egy problémát ábrázolni akarunk, akkor arra a legjobb módszer, hogy egy gráfban felvesszük az adott probléma során létrejövő összes állapotot. Majd ebben a gráfban keressük meg azt az útvonalat, amely a start csúcsból a cél csúcsba megy. Tehát az állapottér nem más mint a probléma kapcsán felmerülő összes lehetséges állapot együttese. Természetesen komolyabb feladatok esetén ez az állapottér nem reprezentálható. Gráfban biztosan nem. Ha csak a sakkra gondolunk, akkor egy játszma lehetséges állásai megfelelnek a gráf egyes csomópontjainak. Az egyik csomópontból a másikba vezető út az adott lépés, a start csúcs ismert, hiszen az a játék kiinduló állapota, de a célcsúcsok egy halmazt alkotnak, hiszen a játék számtalan módon befejeződhet. Tehát ha elképzeljük a sakk állapotterét, láthatjuk, hogy az egy végtelen tér, amiben egy végtelen gráfot tudunk létrehozni.

Az **állapottér reprezentáció** az a mód, ahogyan az állapottér elemeit, az állapotokat bemutatjuk, felvesszük.

Állapottér reprezentáció fogalma
Kezdőállapotok: része az állapottérnek
Összes lehetséges/ egy kitüntetett
Célfeltétel. kezdőállapothoz célállapot halmaz
Explicit megadás/ feltételek
Megoldás: kezdőállapotból célállapotba vezető művelet sorozat
Opt. megoldás: min költségű
Állapottér szemléltetés: reprezentációs gráffal (állapotgráffal)

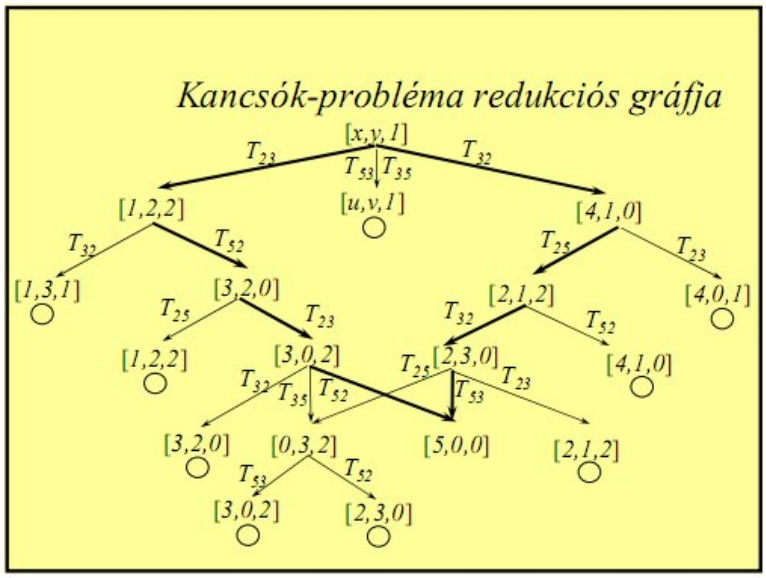
Állapottéren működő kereső rendszerek
Előre haladó ker. rendszerek: kezdőállapot – cél á.
Visszafelé haladó ker. rendszer: célállapot – kezdő á.
Sokszor egyszerűbb. A műveletsort az inverzére cserélve előrehaladó keresést kapunk.
Kétirányú kereső rendszerek.
Két keresés egy időben, cél, ill. kezdőállapotból indítás (pl. előre és visszafelé haladó ker.)
Terminálás ha találkoznak

14. Probléma redukció, boroskancsók példával illusztrálva.

A három kancsó probléma: Írj olyan programot, amely egy vízzel teli nyolc literes kancsóból egy három és egy öt literes üres kancsó segítségével kimér pontosan négy liter vizet a nyolc literes kancsóba!
Miért nem oldja meg a kancsó-problémát egy visszafelé haladó keresés?
Kiindulási célállapotot kiválasztása nem egyszerű: Nem elérhető célállapot például a (2,2,1).
A visszafelé haladó kereséssel talált (4,0,1)→(5,0,0) út nem értelmezhető a feladat megoldásaként.
Probléma redukció
Hogyan határozható meg egy csak részben ismert állapothoz az azt megelőző állapot?
Két kérdésre keressük a választ:
– Van-e olyan művelet, amellyel elérhető az éppen vizsgált állapot?
– Melyik az a megelőző állapot, amelyikből egy kiválasztott művelet az éppen vizsgált állapotba vezet?
Probléma redukció: Valamilyen módon szűkíteni kell a lehetséges megoldások halmazát.

Problémadekompozíciós reprezentációhoz kell:
A feladat részproblémáinak ált. leírása
Eredeti probléma
Az egyszerűen megoldható részproblémák
Dekompozíciós oprátorok: a problémához megoldandó részproblémákat rendelnek.

Szemléltetés gráffal: a-ból b-be akkor vezet él, ha a b állapotra egy alkalmas műveletet végrehajtva visszakapjuk a-t.
Műveletek: az M művelet redukciós operátora a-hoz azt a b-t rendeli, amelyre M-t alkalmazva a-t visszakapjuk. (állapot helyett állapotleírás lehet)
Problémaredukciós reprezentáció:
Állapottér-reprezentáció
Minden művelethez redukciós operátor
Eltérés az állapotgráftól:egy csúcs lehet állapotcsoport, él irány ellentétes a művelettel, célcsúcs a kezdőállapot



15. Nim játék leírás, nyerő stratégia

1.2 A Nim játék.
Felváltva lépnek
Egy sorból vehető el
max minden gyufaszál.
Győztes: aki utoljára húzott
Nyerő állások: a játékos léphet úgy, hogy győzzön
Vesztő állások: ha nincs ilyen lehetőség a lépésre

Heurisztika nyerő stratégiára:
Soronként a db-ok kettes számr.-ben
XOR művelet a számokra
Eredmény nulla: vesztő állások
Eredmény>0 : nyerő állások
(páros számú 1 legyen minden oszlopban)

0	0	1	1	=	3
0	1	0	1	=	5
1	0	0	0	=	8
1	1	1	0		(XOR)

Nim játék tulajdonságai

Állítás: azon állások vesztők, melyekre az *XOR* csupa nullát eredményez.

1. Lemma

Ha egy állásban az *XOR* nem csupa nullát eredményez, akkor van lépés, mely az *XOR* szerint nullát eredményez.

2. Lemma

Ha egy állapotban az *XOR* csupa nulla, akkor nincs lépés, mely eredményeként az *XOR* nulla lesz.

Nyerő stratégia

Ha *XOR* nem nulla, akkor le tudjuk nullázni és az ellenfél nem tud olyat lépni, hogy számunkra megint nulla legyen.

$\Longrightarrow$ nyerő stratégia.

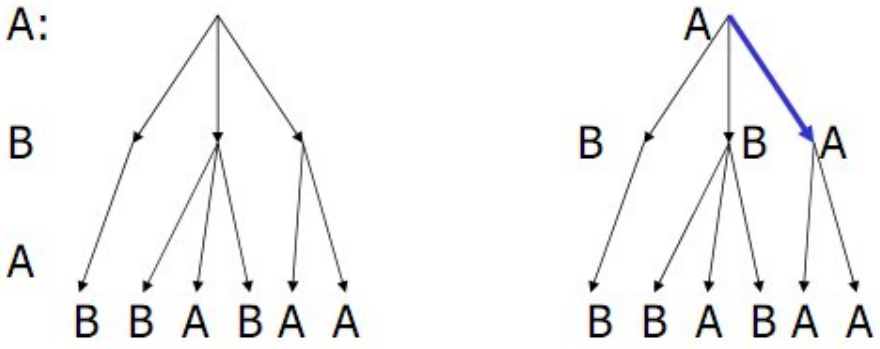
16. Nyerő stratégia teljes információjú játékoknál (irányított gráffal, ÉS/VAGY gráffal)

1.4 Nyerő stratégia létezése
Nyerő stratégia egy játékos számára: minden szituációban van olyan lépése, hogy győzni tud.
TÉTEL:
Egy teljes információjú játék esetén mindig létezik az egyik játékos számára nyerő stratégia , ha a játék nem végződhet döntetlennel.
Biz.: egy konstruktív biz. a teljes játékfával
levelek címkézése a győztes betűjével (A, B)
Közbülső csúcsok címkézése visszafelé haladva:
Ha B lép: ha az utódok közt van B címke, akkor B, különben A
Ha A lép: fordítva

Biz.:

- A hozzárendelés egyértelmű (1 szint esetén, n-szintnél teljes ind)
- Leolvasható a nyerő stratégia

Példa:



1.5 Nyerő stratégia meghat. ÉS/VAGY fával

A kezdő játékos szempontjából írjuk le:

- 0,2,4,... szinteken a csúcsok VAGY kapcsolatban vannak
- 1,3,... szinteken minden lehetséges csúcs választható: ÉS kapcs.

(Nyerő stratégia meghat. ÉS/VAGY fával)

Nyerő/ vesztő címkézés:
Nyerő ha:
A számára nyerő levél /
van nyerő címkéjű VAGY utódja /
összes ÉS utódja nyerő címkéjű
Vesztő ha:
B számára nyerő levél /
összes VAGY utódja vesztő címkéjű /
van vesztő címkéjű ÉS utódja
Nem szükséges minden csúcsot címkézni!

17. Minimax, negamax algoritmusok játékfákon

Minimax algoritmus:
bonyolultabb játékok esetén használják;
nem építheto meg a teljes játékfa;
nem talál biztosan nyero stratégiát;
„eros" vagy „elég jó" lépés;
Neumann János MINIMAX tétele - nulla-összegu játékokra: Minden nulla-összegu játék esetén létezik egy olyan keveréke a stratégiáknak, mely minimax tulajdonsággal rendelkezik.

A játékos strat.	B játékos strat.			
	-1	3	2	9
	5	4	4	6
	8	-2	6	-2

A nem tudja mit fog **B** lépni

A biztosan nyer 4-et a 2. strat.-val,

B legkevesebbet veszít a 2. strat.-val

A azon stratégiát választja, amely a legkisebb nyereségek maximumát adja.

Minimax játékos:

$$\max_i \left(\min_j M_{ij} \right) \leq \min_j \left(\max_i M_{ij} \right)$$

(igaz minden $n \times m$ mátrixra)

Nyeregpont: ahol egyenlő a két oldal.

NEGAMAX algoritmus:
A minimax algoritmusnál a különbözo játékosok lépéseinél minimumot vagy maximumot kerestünk;
A negamax algoritmus egyesíti a kétfajta optimális lépést:
minden lépésben maximumot számol,
ellenben az elozo szint negált értékei szerint.
a javasolt lépés a csúcs negált értéku utódjába történo lépés;

Minimax/negamax algoritmus költséges mert nagyon sok csúcsot kell generálni.

18. Alfabéta algoritmus (az algoritmus és a működés ismertetése)

- **Minimax/negamax algoritmus költséges:**
  - nagyon sok csúcs ot kell generálni
- **Tudjuk**, hogy az értékelés a minimax szabály szerint történik;
- Módszer, mely figyelembe veszi a már kiszámított csúcsok értékét
- Olyan csúcsokat **nem** értékel ki, ahova racionális játék során nem jutunk el.

Kiértékelés során bevezetett változók:

α – *MAX* szint utódjainak maximuma; csak növekedhet.

β – *MIN* szint utódjainak minimuma; csak csökkenhet.

Vágás: muvelet, melynek eredményeként nem értékeljük ki egy csúcshoz tartozó többi utódcsúcsot.
Vágási kritériumok:
MIN csúcs alatt vágunk, ha az egyik őséhez rendelt α érték nagyobb, mint a csúcs β értéke. (alfa vágás)
MAX csúcs alatt vágunk, ha az egyik őséhez rendelt β érték kisebb, mint a csúcs α értéke. (béta vágás)
Egy kiértékelést abba lehet hagyni, ha: $\alpha \geq \beta$

Felépíti a bejárt játékfát
Terminál, ha minden célcsúcs rákövetkezőt feldolg.
Nyilvántartja a bejárt fa alfa, béta értékeit
Ellenőriz-nél, ha vágás lehet: „igaz”
(minimax algoritmus alfa, béta vágásokkal)

19. Ítéletkalkulus alapfogalma

Ítéletkalkulus (nulladrendű predikátumkalkulus)
Kijelentő mondatok reprezentálása, kötőszavakkal
egyszerű részmondatok – igaz/ hamis érték.
Pl. A1: Ha süt a nap, akkor Péter strandra megy.
A2: HA Péter strandra megy, akkor úszik.
A3: Péternek nincs lehetősége otthon úszni.
Köv.: B. Ha süt a nap, akkor Péter nem marad
otthon.
Cél: formális bizonyítása a következtetésnek

Szintaxis
Jelkészlet: elválasztó jelek
logikai műveletek
ítéletváltozók / logikai változók
Formulák: atomi formulák (atom): log. Változó
jólformált formula
atomi formula
ha A, B formulák, akkor a műveletekkel
képzett kifejezések
Pl. ((¬p∧ (q ∨ r)) → s)

Szemantika
elvonatkoztatott igazságértéket ad két lépésben
Interpretáció: igaz/hamis érték minden változóhoz
Kiértékelés: műveletek igazságtáblái alapján
Kielégíthetőségi tulajdonság
Kielégíthető egy formula, ha van olyan interpretációja, amely
igaz. Az interpretáció modellje a formulának.
Érvényes egy formula: ha minden interpretációja igaz
(tautológia)
Kielégíthetetlen egy formula, ha minden interp. hamis

Formulák ekvivalenciája
Két formula ekvivalens, ha minden interp. logikai értéke
megegyezik A ≡B
Logikai törvények
Nevezetesebb A ≡B összefüggések. Pl.
A↔B ≡(A→B) ∧(B →A)
(A→B) ≡¬A∨B,
¬¬A≡A
¬(A ∧B) ≡¬A∨¬B,
¬(A ∨ B) ≡¬A ∧ ¬B
A ∧ (A ∨ B) ≡ A
A ∧ A ≡ A, …

Logikai következmény
A B formula az A1, A2, …,An formulák log.
Következménye, ha minden olyan interp., amelyben
A1, A2, …,An igaz, igaz a B is.
Visszavezethető:
az A1
∧ A2
∧ … ∧ An →B formula érvényességére,
az A1
∧ A2

∧ … ∧ An ∧¬B formula kielégíthetetlenségére.

20. Tételbizonyítás az ítéletkalkulusban, konjunktív normálforma, rezolúció.

Log. következmény jelölés: A1, A2, …,An ⇒B
Ai: feltétel / premissza/ axióma
B: következmény/ konklúzió/ tétel
Pl. az előző példa (A1, A2, A3, B)
p → q, q → r, ¬ (s∧r) ⇒ p→¬ s
vagy visszavezetve:
(p → q) ∧(q → r) ∧ ¬ (s∧r) →(p→¬ s)
(p → q) ∧(q → r) ∧ ¬ (s∧r) ∧ ¬(p→¬ s)
Bizonyítás a tételbizonyítás módszereivel

Igazságtábla módszere
minden lehetséges interp. + kiértékelés
Szemantikus alapú módszer
az interp. csoportosítása jelentés alapján + kiért.
Szintaktikus alapú eljárás:
axiómák és levezetések halmaza
axióma: érvényes formula sémák
Levezetési szabály: érv. formulákból érv. formulát hoz létre

21. Predikátumkalkulus alapfogalmai

3.1 Esőrendű predikátumkalkulus
Igaz/ hamis állítások egy alaphalmaz felett
kvantorok (létezik, minden)
Pl. lássuk be, hogy A1, A2 ⇒B, ahol
A1: Van olyan páciens, aki minden doktorban megbízik
A2: A kuruzslókban egyetlen páciens sem bízik meg.
B: Egyetlen doktor sem kuruzsló.

Szintaxis
Elválasztójelek, log. műveleti jelek, kvantorok
Elemváltozók (x,y,z…), elemkonstansok (a,b,…)
Függvényszimb.(f,g,h,…), ítéletváltozó (p,q,…)
Predikátumszimb. (log. függv.: P, Q R,…)
Kifejezések
Term: minden elemkonstans, elemváltozó
n-argumentumú függv. szimbólum (t1,…,tn term)
f(t1,…,tn)

Szintaxis
Kifejezések
Atomi formula: minden ítéletváltozó
n-argumentumú predikátumszimbólum
t1,…,tn term: P(t1,…,tn)
Jólformált formula: minden atomi formula
ha A, B formulák, a műveletekkel képzett form.
Ha A formula és x változó: ∀xA és ∃xA

Preferencia sorrend: ∀∃¬∧∨→↔
Kvantor hatáskör:szabad,kötött változók
∀x(P(x,y) ∧ ∃yQ(x,y))

Kiértékelés
A, B igazságértéke ismert, -- műveletek igazságtáblái
∀xA és ∃xA igaz, ha minden/legalább egy x∈U esetén A igaz

Kielégíthetőségi tulajdonság
kielégíthető,
érvényes,
kielégíthetetlen
Eltérés az ítéletkalkulustól: nem tudjuk az összes interp. megadni, kiértékelni.

22. Klózformára alakítás a predikátumkalkulusban

Logikai következmény
Hasonló az ítéletkalkulushoz
Tételbizonyításból csak a rezolúció vehető át (klózforma kibővítés)
Klózformára hozás lépései
↔, majd → kiküszöbölése
Negáció hatáskör predikátumra csökkentés
Változók átnevezése (kötött változók különbözzenek)
Összes kvantor a formula bal oldalára
prenex normálforma: prefixum és mátrix

Klózformára hozás lépései
∃ kiküszöbölése
∃xP(x) formulák: kielégíthető ha P(a) is
P(a) helyettesítés (Skolem-konstans)
∀x ∃y P(x,y) formulák: legyen g:x →y (Skolem-függv.)
∀xP(x,g(x)) helyettesítés
∀x1 … ∀xk ∃yQ1z1 ...Qnzn A(x1,…,xk , y, z1,…zr) esetén
elhagyjuk ∃y-t, g(x1,…,xk) y helyébe
Elhagyjuk a prefixumot
Mátrixot konjunktív normálformára hozzuk, klózhalmaz
Változók átnevezése: klózonként eltérő legyen

23. Egyesítési algoritmus, rezolúció.

Pl. modus ponens: A, A →B ⇒B. „A” két példányát egyezővé kell tenni
- egy predikátum két példányát egyezővé kell tenni
Ált. eljárás az egyezővé tételre
Helyettesítés: az α={v1/t1,…, vn/tn} halmazt

helyettesítésnek nev., ha v1, …,vn különböző
változók, t1, …,tn termek és t i
<>vi
Legyen A egy formula. Az Aα az A egy példánya, és
úgy képezzük, hogy minden vi–t t i
-re cserélünk egyidejűleg.

Egyesítő: Az A1,…,An formulákat egyesíthetőnek nev. ha
∃α, amelyre A1α=…= Anα. α a formulák egyesítője
Legáltalánosabb egyesítő egy δ, ha bármely α hoz van
olyan α', hogy α= δα'.
él: meghat. a δ helyettesítést. Ehhez kell:
Különbségi halmaz. Az A1,…,An formulák különbségi
halmazát úgy kapjuk, hogy kiválasztjuk balról-jobbra az
első nem azonos pozíciót minden formulában, majd
vesszük az innen induló részformulákat.

Egyesítő algoritmus (δ meghatározása)
Legyen δ üres halmaz
Amíg minden formula azonos nem lesz, vagy HIBA:
Képezzük a formulák D különbségét
Ha van D-ben egy v változó és t term, hogy v∉t
- alkalmazzuk a {v/t} helyettesítést
- alkalmazzuk δ-ra {v/t} –t.
különben HIBA.

24. Szakértői rendszerek gyakori tudásábrázolási technikái.

SZR:
olyan program, amely az ember probléma megoldó
képességét modellezi
(szűk szakterületen)
Működés:
tudásbázis, következtető mechanizmus,
konzultáció

Felépítés:
felhasználói interfész, ismeretszerző modul, köv.
mechanizmus, tudásbázis, munkamemória,
magyarázó képesség
Technikák:
tudásábrázolás (szabályok, logikai kif., frame,
eljárás/függvény),
következtetés (adat, célvezérelt),
ismeretszerzés (interjú, esettanulmány,...)

Tudás : egy adott szakterület ismereteit jelenti
felszíni, mélyrétegű
szakterületenként más-más forma
Tudástípusok:
procedurális
deklaratív
strukturált

Tudásábrázolási technikák:

deklaratív: logika, O- T -É hármasobjektumok
strukturált: szemantikus háló, framescript, szabálycsoport
procedurális: szabály, eljárás, függvényAgenda, stratégia

SZR fejlesztés
Egy fejlesztési modell:
1. Előtanulmány (újradefiniálás)
2. Tudásbeszerzés (újabb ismeretek)
3. implementálás (finomítás)
4. Tesztelés (visszacsatolások!)
5. Dokumentálás
6. Rendszerkövetés

Gyakori alkalmazási területek:
ipar, üzleti élet, egészségügy, szállítás, mezőgazdaság, katonaság, űrkutatás,...
Főbb probléma típusok
ellenőrzés 6% konfigurálás 10%
diagnosztika 30% oktatás 5%
értelmezés 18% tervezés 9%
hiba elhárítás 17%

25. Fuzzy halmazelmélet fogalmai, alapvető műveletei.

A különböző tudományterületeken gyakran van szükség matematikai modellek alkalmazására olyan esetekben, amikor a kiinduló adatok nem precízek, hiányosak, nem egyértelműen értelmezhetőek, illetve életlen relációkat alkalmaznak. Ezekre a valószínűség számítás mellett a bizonytalan adatok fuzzy-halmazokkal leírása, és az azokkal végzett fuzzy-halmazelmélet és a többértékű logikán alapuló fuzzy-logika használható fel.

A fő különbséget a neuronhálókkal az jelenti, hogy bár mindkettő képes leképezni az összefüggések matematikai ismerete nélkül becsülve, közelítve a leképzés szabályait, de míg a neuronhálók numerikus pontok közt végzik a leképzést, a fuzzy-rendszerek halmazok közti asszociációt valósítanak meg részhalmazok közti leképzések megadásával.

A fuzzy-rendszereknél szükséges az ismeretek jól strukturálhatósága; ugyan megadhatók szimbolikus formában is ismeretek, e módszerek ezeket is numerikusan kezelik.

Alkalmazási területe elsősorban a szabályozástechnikában terjedt el, de újabban mintafelismerésre, képanalízisre használják. Gyakori problématípusok ezenkívül: közelítő következtetések, adatanalízis, optimalizálás, döntéshozatal és információ-visszakeresés.

Fuzzy rendszerekről általában
Fuzzy információ
nem precíz, pontatlan kifejezések (idős ember, magas nyereség)
bizonytalan információkon alapuló kif. (hitelképes vállalat)

életlen relációk (körülbelül egyenlő)

Előfordulás

mat. modellek, döntések, adatok analízise

A bizonytalanság jellege:

- determinisztikus bizonytalanság (többértékű)

Fuzzy technika (Zadeh 65)

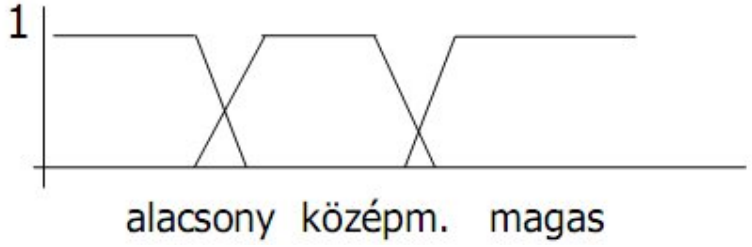
- „hozzátartozás foka”: elem halmazhoz tartozása
  - ~Igazságfok (predikátummal)
- A technika alapja:
  - fuzzy halmazelmélet, fuzzy logika
- fuzzy rendszer: halmaz- halmaz leképezés
  $A_i \rightarrow B_i$ formájú leképezéseket aggregál

Fuzzy halmazelmélet

- klasszikus halmazelmélet A= {x | x> 120}
  - Tagsági/ karakterisztikus függvénye

$$\mu(x) = \begin{cases} 1 & ha \ x > 120 \\ 0 & különben \end{cases}$$

- életlen, fuzzy halmazok: hozzátartozás foka



- Tagsági / tartalmazási függvény:
  μ :x ---> [0,1]
- Megadási formák:

$$A = \{ x, \mu_A(x) | \ x \in X \}$$

$$A = \mu_A(x_1)/x_1 + \mu_A(x_2)/x_2 + \ldots + \mu_A(x_n)/x_n = \Sigma \mu_A(x_i)/x_i$$

$$A = \int_x \mu_A(x)/x \ .$$

26. Fuzzy logika fogalmai, alapvető műveletei.

Műveletek

Legyen A, B fuzzy halmaz, $\mu_A(x)$, $\mu_B(x)$ tart. függv.

- Aritmetikai műveletek:
  C=A°B (° lehet + - / *)

$$\mu_C(z) = \mu_{A \circ B}(z) = \sup_{z = x \circ y} \min\left((\mu_A(x), \mu_B(y)\right).$$

- Halmazműveletek:
  - metszet (minimum operátor)
    $\mu_{A \wedge B}(x) = \min((\mu_A(x), \mu_B(x))$

Műveletek

Legyen A, B fuzzy halmaz, $\mu_A(x)$, $\mu_B(x)$ tart. függv.

- Halmazmüv.:
  - egyesítés (maximum operátor)
    $\mu_{A \vee B}(x) = \max((\mu_A(x), \mu_B(x))$
  - szorzat
    $\mu_{AB}(x) = \mu_A(x) * \mu_B(x)$
  - Komlemens
    $\mu_{\neg A}(x) = 1 - \mu_A(x)$
- T-norma S-norma művelettel definiálás
  Metszet: $\mu_{A \wedge B}(x) = T(\mu_A(x), \mu_B(x))$

T-norma S-norma műveletek

Egy $T: [0,1]^2 \to [0,1]$ függvényt *T-normának* nevezünk, ha teljesül:

1. $T(a, 1) = a$ (1 az egységelem)
2. $a \le b$ és $c \le d \Rightarrow T(a,c) \le T(b,c)$ (monoton növő)
3. $T(a,b) = T(b,a)$ (kommutatív)
4. $T(a, T(b,c)) = T(T(a,b),c)$ (asszociatív)

Egy $S: [0,1]^2 \to [0,1]$ függvényt *T-konormának*, vagy *S-normának* nevezünk, ha

1. $S(a, 0) = a$ (0 az egységelem)
2. $a \le b$ és $c \le d \Rightarrow S(a,c) \le S(b,c)$ (monoton növő)
3. $S(a,b) = S(b,a)$ (kommutatív)
4. $S(a, S(b,c)) = S(S(a,b),c)$ (asszociatív).

Reláció műveletek
projekció,
cilindrikus kiterjesztés
max-min kompozíció
R1 és R2 max-min kompozíciója X,Y felett egy R fuzzy reláció:
$R = R1 \circ R2 = \{((x,z), \sup_y \min(\mu R1(x,y), \mu R2(y,z)) \mid x \in X, y \in Y, z \in Z\}$.

27. Neurális hálók általános jellemezői, működésük.

definíció: információ feldolgozó rendszer, amely sok egyszerű feldolgozó elemből áll, melyek egymással irányított, súlyozott kapcsolatokkal vannak összekötve, és az információt a kapcsolatok révén továbbítják.

A mesterséges neurális hálózat, egy biológiai indíttatású gép/program, ami a biológiai neurális hálózat néhány tulajdonságát modellezi. Az alkalmazások többsége technikai jellegű, és nem kognitív modell. A mesterséges neurális hálók nem csak a biológia, hanem más tudományterületek (matematika, fizika, pszichológia) eredményeit is felhasználják.

A neurális hálózatok előnyei:
· fejlesztésük, vagy szimulációjuk bizonyos mérvű egyszerűséggel végezhető,
· a hálózat a feladathoz alkalmazkodik, "tanul",
· a memória szétosztott, a párhuzamosság elvileg nagyobb sebességet tesz lehetővé,
· nincs szükség a hagyományos értelemben vett programozásra,
· komplex alkalmazásokhoz a Neumann-elvű gépek szimulátorok, vagy gyorsítókártyák segítségével használhatók.

A hátrányai:
· mivel ma még nem létezik a Neumann-elvnél jobb számítástechnikai megoldás, az ilyen gépeken megvalósított nagyméretű neurális hálózatok számításai gyakorlati alkalmazásokra még sokszor elfogadhatatlan végrehajtási időt igényelnek,
· a teljesítmény nagyban függ az adatok előfeldolgozásától, a neurális hálózat számára történő prezentációtól,
· a kapott eredmények gyakran nehezen értelmezhetők - a hálózat olyan fekete doboz, melybe nincs betekintésünk, tehát a működés eredményességét statisztikai módszerekkel kell mérni. Ez sok potenciális felhasználóban bizalmatlanságot kelt.

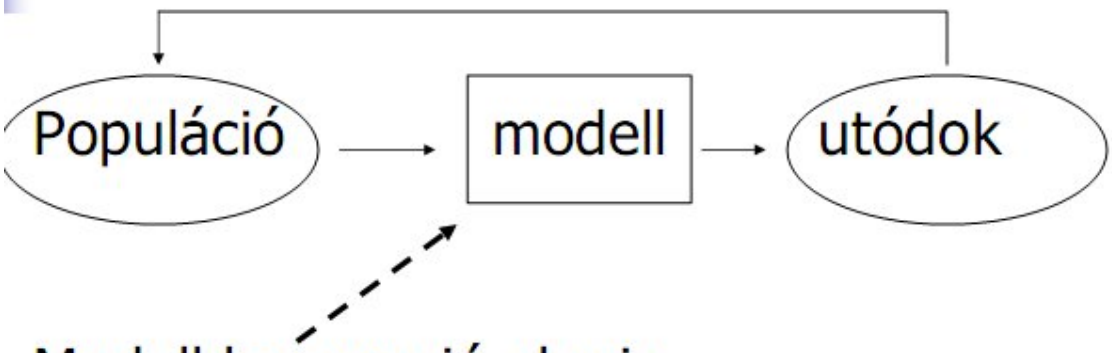
NH feldolgozóeleme
cella, processzor elem PEi (i=1,2, ...,n)
állapotok halmaza Ai (t)
input/output adatok: Xj -> PEi -> Yi
inputfüggvény $NETi(t) = \sum(Xj(t) * Wij(t))$
új állapot: Fi állapotfüggvénnyel
Ai(t+1) = Fi(Ai(t), inputok) szigmoid, lin.
output: Oi kimeneti függvény
Yi(t)= Oi(Ai(t))

28. Evolúciós algoritmus alapfogalmak, műveletek, általános algoritmus és működése.

Evolúciós algoritmusok

A neurális hálózatok tanításánál felvetődött problémák megoldására születtek az evolúciós algoritmusok. A neurális hálózatok tanítása három alapvetően eltérő módszer szerint történhet a hálózatot tanító be- és kimeneti értékek függvényében: *ellenőrzött tanulásról* beszélünk, ha adott mindkét adat, *nem ellenőrzött tanulásról* beszélünk, ha csak a bemeneti adatok állnak rendelkezésünkre (ekkor a hálózatnak magának kell az adatok közti összefüggéseket megtalálni), és *analitikus tanulásról* beszélhetünk, ha nem lépésenként, hanem elméleti úton határozzuk meg a hálózat súlyszámait. Az ellenőrzött tanulásnál a súlytényezők helyes megválasztása a tanítás célja, melyet *kritériumfüggvény* bevezetésével valósítanak meg, amely függvénye a hálózat tényleges és optimális súlyszámaiknak, és meghatározza a hálózat jóságát. A tényleges feladat a kritérium függvény minimumának megkeresése. Erre a feldolgozott adatok és a hálózat struktúrája szerint több féle szélsőérték-keresési eljárást dolgoztak ki: a feladat többnyire megoldható gradiens módszerekkel. Az olyan esetekben, amikor a kritériumfüggvény felületének lokális minimumai vannak, a gradiens módszerek nem alkalmazhatóak, ilyenkor a minimumkeresésnél lehetővé teszik zaj bevezetésével a felületen való felfelé mozgást, ami lehetővé teszi a lokális minimumokból való kiszabadulást. Ezt nevezzük *sztochasztikus szélsőérték-keresésnek*, e módszerek egyik speciális esetei az *evolúciós algoritmusok*, melyek biológiai analógiákra épülnek: az élőlények *genotípusának* és *fenotípusának* változását veszik a fejlődés alapjául. A fenotípus jelenti a külső megjelenést, a genotípus a génstruktúrát. Egy génstruktúrához a körülményektől függően több fenotípus is tartozhat; a környezethez való alkalmazkodás elérhető mind a genotípus, mind a fenotípus változtatásával. Az evolúciós algoritmusok négy fő csoportja: a *genetikus algoritmusok*, a *genetikus programozás*, az *evolúciós stratégiák* és az *evolúciós programozás*.

Evolúciós algoritmus koncepció



Modell koncepció alapja

- Biológiai evolúció,
- Biológiai rendszer, faj információ csere
- Matematikai modell

Modell koncepció alapja
Biológiai evolúció,
Öröklés (szülő – utód), vírusok, baktériumok
Egy biológiai rendszer
Immun rendszer
faj információ csere pl. méhek, hangyák
Matematikai modell
Lineáris kombináció alapú
Valószínűségi modell a populáció alapján
Valószínűségi modell a korábbi populációk alapján

EA alapfogalmak
individum/kromoszóma (egy megoldás)
Populáció (megoldás halmaz)
szülő, utód (régi- új megoldás)
kereső operátorok (műveletek)
Rekombináció/crossover, mutáció, szelekció
Fitnesz (megoldás értékelésre)
generáció

EA általános ciklus
stratégiai par. választása
populáció inicializálás
individumok értékelése
generációs ciklus:
szülők választása, autódok generálása
utódok értékelése, uj populáció előállítás
megállási feltétel
eredmény