Állapottér-reprezentáció állapot-gráfja

Állapottér modell	Állapot-gráf
állapot	csúcs
művelet hatása egy állapotra	irányított él
művelet költsége	él költsége
kezdő állapot	startcsúcs
célállapot	célcsúcs

Gráf-reprezentáció: állapot-gráf, startcsúcs, célcsúcsok

egy műveletsorozat hatása	irányított út
megoldás	irányított út a startcsúcsból egy célcsúcsba

1, VL1 algoritmus

(Nincs kör és mélységi korlát figyelés) véges körmentes irányított gráfon terminál, és ha van megoldás akkor talál egyet.

```
Recursive procedure VL1(akt : N) return (A* ; hiba)
    if cél(akt) then return(nil) endif
    for ∀ új ∈ Γ(akt) loop
        megoldás := VL1(új)
        if megoldás ≠ hiba then
            return(fűz((akt,új), megoldás) endif
    endloop
    return(hiba)
end
```

volt - 2, Vezérlési stratégiák

- Elsődleges(általános): független a feladattól és annak modelljétől: nem merít sem a feladat ismereteiből sem a modell sajátosságaiból.
  - 1. nem módosítható: - lokális keresések - evolúciós algoritmus
  - 2. módosítható: - visszalépéses keresések - gráfkeresések
- Másodlagos(modell függő): nem függ a feladat ismereteitől , de épít a feladatat modelljének általános elemére. pl. Lineáris input stratégia (rezolúciónál)
- Heurisztikus: a feladattól származó, annak modelljében nem rögzített a megoldást segítő speciális ismeret. pl. Manhattan-heurisztika

3, Szimulált hűtés algoritmusa

akt := start ; k := 1 ; i := 1

while not(akt ∈ T f(akt) régóta nem változik) loop

  if i > L_k then k := k+l; i := l

  új := select(Γ(akt)-π(akt))

  if f(új) ≤ f(akt) or

  f(új) >f(akt) and $e^{\frac{f(akt)-f(új)}{T_k}}$ > rand[0,1]

then akt := új

i := i+l

endloop

return akt

4, Mélységi / Szélességi /Egyenletes gráfkeresés (Nem informált)

Elnevezés	**Definíció**	**Eredmények**
Mélységi	f = -g | c(n,m) = 1	Végtelen gráfokban mélységi korláttal megoldást garantál.
Szélességi	f = g | c(n,m) = 1	Végtelen gráfokban a legrövidebb megoldást adja, egy csúcsot csak egyszer terjeszt ki.
Egyenletes	f = g	Végtelen gráfokban a legolcsóbb megoldást adja, egy csúcsot legfeljebb egyszer terjeszt ki.

5, Keresőrendszer részei

	Lokális keresés	Visszalépéses keresés	Gráfkeresés
Globális mintaterület	Egy csúcs és annak a szűk környezete.	Egy út a startcsúcsból az aktuális csúcsba és az arról leágazó még ki nem próbált élek.	A startcsúcsból induló, már feltárt részgráf.
Keresési szabály	Az aktuális csúcsot minden lépésben egy az annak környezetében levő "jobb" csúccsal cseréljük le.	A nyilvántartott út végéhez egy új, még ki nem próbált él hozzáfűzése, vagy a legutolsó él törlése (visszalépés)	Az egyik útvégi csúcs kiterjesztése.
Vezérlési stratégia	A "jobbság" eldöntésére kiértékelő függvényt használ, ami remélhetőleg annál jobb értéket ad egy csúcsra, minél közelebb van a célhoz.	A visszalépés szabályát csak a legvégső esetben alkalmazza. Feltételei: zsákutca, zsákutca torkolat, kör, mélységi korlát.	Mindig a lekedvezőbb csúcs kiterjesztésére törekszik

6, B algoritmus

A B algoritmust az A algoritmusból kapjuk úgy, hogy bevezetjük az F aktuális küszöbértéket, majd az a -1. lépést kiegészítjük az F:= f(s) értékadással, a -4. lépést pedig helyettesítjük az if $\min_f$(NYILT) < f
then n:= $\min_g$ (m $\in$ NYILT | f(m) < F)
else n:= $\min_f$ (NYILT); F:= f(n)
endif elágazással.

7, Neuronhálók

Bemenő értékek (számok) együtteséből kimenő értéket (számokat) előállító rendszer, amely egymáshoz kapcsolódó, tanítható számoló egységekből áll. Részei: - mesterséges neuron, - hálózati topológia, - tanulási szabály
Legegyszerűbb mesterséges neuronháló: perceptron modell. (Rosenblott perceptronok)

8, Mit tesz az általános gráfkereső algoritmus akkor, amikor már egy korábban felfedezett csúcshoz talál minden addiginál olcsóbb utat? Mi legyen az olcsóbb úton újra megtalált n csúcs leszármazottaival?

1. Járjuk be és javítsuk ki a pointereket és élköltségeket!
2. Kerüljük el egy jó kiértékelő függvénnyel, hogy ilyen történjen!
3. Semmi mást ne tegyünk, csak legyen az m csúcs újra nyílt!

volt - 9, Győztes stratégia

A győztes stratégia egy olyan elv, amelynet betartva egy játékos az ellenfél minden lépésére tud olyan válasz adni, hogy megnyerje a játékot.

10, Nyerő stratégia

A nyerő stratégia NEM egyetlen győztes játszma, hanem olyan győztes játszmák összessége, amelyek közöl az egyiket biztos végig tudja játszani az a játékos aki rendelkezik a nyerő stratégiával.

11, Felügyelt tanulás 3 fajtája

Válasszunk egy f: P x X -> Y paraméteres leképezést a vizsgált problémát modellező φ: X -> Y leképezés közelítéséhez, majd azon $\Theta \in P$ paramétert keressük (paraméteres tanulás), amelyre az (x_n,y_n) (n=1...N) tanító minták mellet (ahol $y_n = \varphi(x_n)$) az alábbi kifejezés értéke elég kicsi.

$\frac{1}{N}\sum_{n=1}^{N} l(f(\Theta, x_n), y_n)$ (ahol -> l a hiba függvény, y_n az elvárt kimenet, $f(\Theta, x_n)$ = t_n a számított kimenet)

volt - 12, Dekomozíciós reprezentáció

- a feladata részproblémáinak általános leírását
- a kiinduló problémát
- az egyszerű problémákat
- a dekomponáló műveleteket: D: probléma$^+$ és D(p) = $<p_1,...,p_n>$

13, k means (Hard clustering)

Az algoritmus célja, az adatpontok megadott számú klaszterbe sorolása. A pontok az eljárás minden iterációjában a hozzájuk legközelebbi klaszterbe kerülnek, amelyeket a középpontjukkal azonosítunk.

14, Általános gráfkereső algoritmus

```
G:= ({start}, ∅); NYILT := {start};  g(start) := 0; π(start) := nil
loop
    if empty(NYILT) then return nincs megoldás
    n := min_f(NYILT)
    if cél(n) then return megoldás
    NYILT := NYILT - {n}
    for ∀m ∈ Γ(n) - π(n) loop
        if m ∉ G or g(n) + c(n,m) < g(m) then
        π(m):= n; g(m):= g(n)+c(n,m); NYILT:=NYILT U {m}
    endloop
    G:= G U {(n,m) ∈ A | m ∈ Γ(n) - π(n) }
endloop
```

Jelölések:
− keresőgráf (G) : a reprezentációs gráf eddig felfedezett és egyben el is tárolt része
− nyílt csúcsok halmaza (OPEN) : kiterjesztésre várakozó csúcsok, amelyeknek gyerekeit még nem vagy nem eléggé jól ismerjük
− kiértékelő függvény (f: OPEN → R) : kiválasztja a megfelelő nyílt csúcsot kiterjesztésre

15, VL2 algoritmus

Recursive procedure VL2(út : N*) return (A*; hiba)

```
akt := utolsó_csúcs(út)
if cél(akt) then return(nil) endif
if hossza(út) >= korlát then return(hiba) endif
if akt ∈ maradék(út) then return(hiba) endif
for ∀ új ∈ Γ(akt) − π(akt) loop
    megoldás := VL2(fűz(út, új))
    if megoldás ≠ hiba then
        return(fűz((akt,új), megoldás)) endif
endloop
return(hiba)
end
```

volt - 16, Mi a heurisztika, miért és mikor használjuk?

- Heurisztikus függvénynek nevezzük azt a h: N -> R fv-t, amelyik egy csúcsnál megbecsüli a csúcsból a célba vezető optimális út költségét.
- Feladattól származó, annak modelljében nem rögzített, a megoldást segítő speciális ismeret.
- Közvetlenül építjük be az algoritmusba, hogy annak a hatékonysága és eredményessége javuljon, habár erre semmiféle garanciát nem nyújt. A hatékonyság növelése alatt a memóriaigény és a futási idő csökkentését értjük.

17, k means (Hard clustering)

Az algoritmus célja, az adatpontok megadott számú klaszterbe sorolása. A pontok az eljárás minden iterációjában a hozzájuk legközelebbi klaszterbe kerülnek, amelyeket a középpontjukkal azonosítunk.

18, Evolúciós algoritmus

Elemei: - kódolás
- rátermettségi függvény
- evolúciós operátorok
- kezdő populáció, megállási feltétel
- stratégiai paraméterek

```
Procedure EA
populáció := kezdeti populáció
while terminálási feltétel nem igaz
loop
    szülők := szelekció(populáció)
    utódok := rekombináció(szülők)
    utódok := mutáció(utódok)
    populáció := visszehelyezés(populáció utódok)
endloop
```

volt - 19, Gépi tanulás

Tanulási modellek:
-Induktív : felügyelt tanulás, felügyelet nélküli tanulás, megerősítéses tanulás
f leképezést (annak paramétereit) $x_n \in X$ (n=1..N) bemenetek (minták) alapján tanuljuk.

-Adaptív (inkrementális tanulás):

Egy már megtanult f leképezést egy új minta anélkül módosíthat, hogy a korábbi mintákat újra megvizsgálnánk

volt - 20, Visszalépéses keresés vs általános gráfkeresés

(munkaterület, keresési szabály, vezérlés, milyen feltételek mellett ad meo-t, tárigény, futási idő)

Visszalépéses

- Munkaterulet: egy ut a startcsucsbol az aktualis csucsba (az utrol leagazo meg ki nem probalt elekkel egyutt)
  - kezdetben a startcsucsot tartalmazo nulla hosszusagu ut
  - terminalas celcsucs eleresekor vagy a startcsucsbol valo visszalepeskor
- kereses szabalyai: a nyilvantartott ut vegehez egy uj el hozzafuzese, vagy a legutolso el torlese
- vezerles strategiaja: a visszalepes szabalyat csak a legvegso esetben alkalmazza
- Veges kormentes iranyitott grafokon a VL1 mindig terminal es ha letezik megoldas akkor talal egyet

Egyszerű gráf keresés

- Munkaterulet: startcsúcsból kiinduló már feltárt útjai a reprezentációs gráfnak (keresőgráf), valamint a feltárt utak végei (nyílt csúcsok)
  - kiindulo erteke: startcsucs
  - terminalasi feltetel: vagy celcsucsot terjeszt ki vagy nincs nyilt csucs
- Keresesi szabaly: egy nyilt csucs kiterjesztese
- Vezerlesi startegiaja: a legkedvezőbb csúcs kiterjesztésére törekszik, és ehhez egy kiértékelő függvényt használ.

A GK véges d-gráfban mindig terminál.

Ha egy véges �-gráfban létezik megoldás, akkor a GK megoldás megtalálásával terminál.

volt - 21, Heurisztika:

Feladattól származó, annak modelljében nem rögzített, a megoldást segítő speciális ismeret. Közvetlenül építjük be az algoritmusba, hogy annak hatékonysága és eredményessége javuljon, habár erre semmiféle garanciát nem nyújt. (Kombinatorikus robbanás elkerülése) A hatékonyság növelése alatt a memóriaigény és a futásidő csökkentését értjük.

volt - 22, Mi a heurisztika es hol vezetjuk be a KR-ben?

Heurisztikus függvénynek nevezzük azt a h:N→R függvényt, amelyik egy csúcsnál megbecsüli a csúcsból a célba vezető („hátralévő") optimális út költségét. h(n) ~ h*(n)

23, Útkeresési problémára optimális megoldási utat adó algoritmusok felsorolása

Szélessegi, egyenletes, A*, A_c, B

24, Mit reprezentál a játékfa és az és vagy gráf a 2 személyes játékoknál

A játékfa a lehetséges lépéseket es azt reprezentálja, hogy ki mikor nyerhet.

Az ES/VAGY fa a nyerő stratégia megkereséséhez használható

25, A*, A_c, B algoritmusok futási idejének jellemzése

A*algoritmusnál a futási idő legrosszabb esetben exponenciálisan függ a kiterjesztett csúcsok számától, de ha olyan heurisztikát választunk, amelyre már A_C algoritmust kapunk, akkor a futási idő lineáris lesz. Persze ezzel a másik heurisztikával változik a kiterjesztett csúcsok száma is, így nem biztos, hogy egy A_C algoritmus ugyanazon a gráfon összességében kevesebb kiterjesztést végez, mint egy csúcsot többször is kiterjesztő A*algoritmus. A B algoritmus futási ideje négyzetes, és ha olyan heurisztikus függvényt használ, mint az A*algoritmus (azaz megengedhetőt), akkor ugyanúgy optimális megoldást talál (ha van megoldás) és a kiterjesztett csúcsok száma (mellesleg a halmaza is) megegyezik az A*algoritmus által kiterjesztett csúcsokéval.

26, Gráfkeresésnél mi történik ha olcsóbb utat talál?

Ha m (eleme) G és g(m) < g(n) + c(n, m), akkor: π(m) = n és g(m) = g(n) + c(n, m), és az m csúcsot nyílttá tesszük. Ha a kiértékelő függvény csökkenő, a korrektség megától helyreáll.

27, Mi a szelekció, milyen a jó szelekció, típusai

Célja: a rátermett egyedek kiválasztása úgy, hogy a rosszabbak kiválasztása is kapjon esélyt.

- Rátermettség arányos (rulett kerék algoritmus): minél jobb a rátermettségi függvényértéke egy elemnek, annál nagyobb valószínűséggel választja ki
- Rangsorolásos: rátermettség alapján sorba rendezett egyedek közül a kisebb sorszámúakat nagyobb valószínűséggel választja ki.
- Versengő: véletlenül kiválasztott egyedcsoportok (pl. párok) legjobb egyedét választja ki.
- Csonkolásos v. selejtezős: a rátermettség szerint legjobb (adott küszöbérték feletti) valahány egyedből véletlenszerűen választ néhányat.

28, Minimax módosításaiként bevezetett algoritmusok felsorolása, melyik milyen tulajdonságát javítja a minimaxnak?

- Átlagoló kiértékelés: Célja a kiértékelő függvény esetleges tévedéseinek simítása.
- Váltakozó mélységű kiértékelés: Célja, hogy a kiértékelő függvény minden ágon reális értéket mutasson.
- Szelektív kiértékelés: Célja a memória-igény csökkentése
- Negamax algoritmus: Negamax eljárást könnyebb implementálni.