1 AZ MI FOGALMA

1.1 Miről ismerhető fel az MI?

Megoldandó feladat: nehéz
- A feladat problématere hatalmas
-Szisztematikus keresés helyett intuícióra, kreativitásra (azaz heurisztikára) van szükségünk ahhoz, hogy elkerljük a kombinatorikus robbanást.
Szoftver viselkedése: intelligens (tárol ismeretet, automatikusan következtet, tanul, gépi látás, gépi cselekvés)
-Turing teszt vs. kínai szoba elmélet
-"mesterjelölt szintű" mesterséges intelligencia
Felhasznált eszközök: sajátosak
-Átgondolt reprezentáció a feladat modellezéséhez
-Heurisztikával megerősített hatékony algoritmusok
-Gépi tanulás módszerei

2 MODELLEZÉS ÉS KERESÉS

feladat → útkeresési probléma → megoldás
Feladat és útkeresési probléma között helyezkedik el a modellezés. Itt található az állapottér-reprezentáció, a probléma dekompozíció, a korlátprogramozási modell, és a logikai reprezentáció.
Az útkeresési problémát egy gráffal reprezentálhatjuk. Az útkeresési probléma és a megoldás között található a keresés. A keresésbe tartoznak a lokális keresések, a visszalépéses keresések, a gráfkeresések, az evolúciós algoritmus, a rezolúció és a szabályalapú következtetés.

2.1 Mire kell a modellezésnek fókuszálni

Problématér elemei: probléma lehetséges válaszai.
Cél: egy helyes válasz (megoldás) megtalálása
Keresést segítő ötletek (heurisztikák):
-Problématér hasznos elemeinek elválasztása a haszontalanoktól.
-Az elemek szomszédsági kapcsolatainak kijelölése, hogy a probléma tér elemeinek szisztematikus bejárását segítsük.
-Adott pillanatban elérhető elemek rangsorolása.
-Kiinduló elem kijelölése.

2.2 Útkeresési probléma

Egy útkeresési problémában a problématér elemeit egy olyan élsúlyozott irányított gráf csúcsai vagy útjai szimbolizálják, amelyik gráf nem feltétlenül véges, de a

csúcsainak kifoka véges, és van egy közös pozitív alsó korlátja az élek súlyának (költségének) (δ-gráf).
A megoldást ennek megfelelően vagy egy célcsúcs, vagy egy startcsúcsból célcsúcsba vezető út (esetleg a legolcsóbb ilyen) megtalálása szolgáltatja.
Számos olyan modellező módszert ismerünk, amely a kitűzött feladatot útkeresési problémává fogalmazza át.

2.3 Gráf fogalmak

-csúcsok, irányított élek:$\rightarrow$ N, A $\subseteq$ NxN
-él n-ből m-be $\rightarrow$ (n,m) $\in$ A (n,m $\in$ N)
-n utódai $\rightarrow$ γ (n) = { m $\in$ N | (n,m) $\in$ A }
-n szülei $\rightarrow$ π(n) $\in$ Π(n) = { m $\in$ N | (m,n) $\in$ A }
-irányított gráf $\rightarrow$ R=(N,A)
-véges sok kivezető él $\rightarrow$ | γ (n) | $< \infty$ ($\forall$ n $\in$ N)
-élköltség $\rightarrow$ c:A $\rightarrow$ $\mathbb{R}$
-δ tulajdonság ($\delta \in \mathbb{R}^+$) $\rightarrow$ c(n,m) $\geq \delta > 0$ ($\forall$(n,m)$\in$A)
-δ-gráf $\rightarrow$ δ-tulajdonságú, véges sok kivezető élű, élsúlyozott irányított gráf
-irányított út $\rightarrow$ $\alpha = (n,n_1),(n_1,n_2),...,(n_{k-1},m) = <n,n_1,n_2,...,n_{k-1},m>$ $n \rightarrow^{\alpha} m$, $n \rightarrow m$, $n \rightarrow M$ ($M \subseteq N$) $n \rightarrow m$, $n \rightarrow M$
-út hossza $\rightarrow$ az út éleinek száma: $|\alpha|$
-út költsége $\rightarrow$ $c(\alpha)=c^{\alpha}(n,m) := \sigma_{j=1..k} c(n_{j-1},n_j)$ ha $\alpha = <n=n_0,n_1,n_2,...,n_{k-1},m=n_k>$
-opt. költség $\rightarrow$ $c^*(n,m):=\min_{\alpha \in \{ n \rightarrow m \}} c^{\alpha}(n,m)$ {δ gráfokban ez végtelen sok út esetén is értelmes. Értéke ∞, ha nincs egy út se.}
-opt. költség2 $\rightarrow$ $c^*(n,M):=\min_{\alpha \in \{ n \rightarrow m \}} c^{\alpha}(n,M)$
-opt. költségű út $n \rightarrow^* m := min_c\{\alpha |\ \alpha \in \{n \rightarrow m\}\}$
-opt. költségű út $n \rightarrow^* M := min_c\{\alpha |\ \alpha \in \{n \rightarrow M\}\}$

2.4 Gráfreprezentáció fogalma

Minden útkeresési probléma rendelkezik egy (a probléma modellezéséből származó) gráfreprezentációval, ami egy (R,s,T) hármas, amelyben:
-R=(N,A,c) δ-gráf az ún. reprezentációs gráf,
-az s$\in$N startcsúcs,
-a T$\subseteq$N halmazbeli célcsúcsok.
És a probléma megoldása:
-egy t$\in$T cél megtalálása, vagy
-egy s$\rightarrow$T, esetleg s$\rightarrow^*$T optimális út megtalálása
(s-ből Tegyik csúcsába vezető irányított út, vagy s-ből T egyik csúcsába vezető legolcsóbb irányított út)
Az útkeresési problémák megoldásához azok a reprezentációs gráfjainak nagy mérete miatt speciális (nem determinisztikus, heurisztikus) útkereső algoritmusokra van szükség, amelyek:
-a startcsúcsból indulnak, amely az első aktuális csúcs;
-minden lépésben nem-determinisztikus módon új aktuális csúcso(ka)t választanak

a korábbi aktuális csúcs(ok) alapján (gyakran azok gyerekei közül);
-tárolják a már feltárt reprezentációs gráf egy részét;
-megállnak, ha célcsúcsot találnak vagy nyilvánvalóvá válik, hogy erre semmi esélyük.

2.5 Kereső rendszer (KR)

Procedure KR
1. ADAT:= kezdeti érték
2. while !terminálási feltétel(ADAT) loop
3. SELECT SZ FROM alkalmazható szabályok
4. ADAT := SZ(ADAT)
5. endloop
end
AHOL:
-ADAT: globális munkaterület, tárolja a keresés során megszerzett és megőrzött ismeretet
-alkalmazható szabályok: keresési szabályok, megváltoztatják a globális munkaterület tartalmát
-SELECT: vezérlési stratégia, alkalmazható szabályok közül kiválaszt egy "megfelelőt"

2.6 Kereső rendszerek vizsgálata

-helyes-e (azaz korrekt választ ad-e)
-teljes-e (minden esetben választ ad-e)
-optimális-e(optimális megoldást ad-e)
-idő bonyolultság
-tár bonyolultság

3 GÉPI TANULÁS

-Egy programozási feladat megoldásához meg kell adnunk a feladat modelljét és készítenünk kell egy ehhez illeszkedő algoritmust, amely a feladat megoldását előállítja.
-Gépi tanulással a modell (reprezentáció és/vagy heurisztika), illetve a megoldó algoritmus (többnyire annak bizonyos paraméterei) állhatnak elő automatikusan.
-A tanuláshoz a megoldandó probléma néhány konkrét esetére, a tanító példákra van szükség.
-A gépi tanulási módszereket három csoportba szokás sorolni: felügyelt-, nemfelügyelt, és megerősítéses tanulásra attól függően, hogy a tanító példák input-output párok, csak inputok, vagy input-hasznosság párok.

4 Állapottér-reprezentáció

Állapottér: a probléma leírásához szükséges adatok által felvett érték-együttesek (azaz állapotok) halmaza
-az állapot többnyire egy összetett szerkezetű érték
-gyakran egy bővebb alaphalmazzal és egy azon értelmezett invariáns állítással definiáljuk
Múveletek: állapotból állapotba vezetnek.
-megadásukhoz: előfeltétel és hatás leírása
-invariáns tulajdonságot tartó leképezés
Kezdőállapot(ok) vagy azokat leíró kezdeti feltétel
Végállapot(ok) vagy célfeltétel

4.1 Hanoi tornyai probléma

Állapottér: AT=$\{1,2,3\}^n$
megjegyzés: a tömb i-dik eleme mutatja az i-dik korong rúdjának számát, a korongok a rudakon méretük szerint fentről lefelé növekvő sorban vannak.
Művelet:Rak(honnan,hova): AT→AT honnan, hova ∈ {1,2,3}
HA a honnan és hova létezik és nem azonos, és van korong a honnan rúdon, és a hova rúd üres vagy a mozgatandó korong (honnan rúd felső korongja) kisebb, mint a hova rúd felső korongja, AKKOR this[honnan legfelső korongja] := hova

4.2 Állapottér-reprezentáció gráf-reprezentációja

δ-gráf állapot-gráf
-csúcs: állapot
-irányított él: művelet hatása
-élköltség: művelet költsége
startcsúcs kezdőállapot
célcsúcsok végállapotok
irányított út egy műveletsorozat hatása

4.3 Állapottér vs. problématér

-Az állapottér-reprezentáció és a problématér között szoros kapcsolat áll fenn, de az állapottér többnyire nem azonos a problématérrel.
-A problématér elemeit többnyire nem az állapotok, hanem a startcsúcsból induló különböző hosszúságú irányított utak.
-A hanoi tornyai problémánál például egy megoldást egy irányított út szimbolizál, amelyik a startcsúcsból a célcsúcsba vezet.
-Van amikor a megoldás egyetlen álapot (azaz csúcs), de ebben az esetben is kell találni egy odavezető operátor-sorozatot (azaz irányított utat).

4.4 Állapot-gráf bonyolultsága

Állapot-gráf bonyolultsága → Problématér mérete → Keresés számításigénye

A bonyolultság elsősorban a start csúcsból kivezető utak száma az oda-vissza lépések nélkül, amely nyilván függvénye a
-csúcsok és élek számának
-csúcsok ki-fokának
-körök gyakoriságának, és hosszuk sokféleségének

Ugyanannak a feladatnak több modellje lehet: érdemes olyat keresni, amely kisebb problémateret jelöl ki.
-Az előző reprezentációnál a problématér mérete, azaz a lehetséges utak száma, óriási. Készítsünk jobb modellt!
-Bővítsük az állapotteret, és használjunk új műveletet!
-Műveletek előfeltételének szigorításával csökken az állapot-gráf átlagos ki-foka.

4.5 Művelet végrehajtásának hatékonysága

A művelet kiszámítási bonyolultsága csökkenthető, ha az állapotokat extra információval egészítjük ki, vagy az invariáns szigorításával szűkítjük az állapotteret.

4.6 Hogyan "látja" egy keresés a reprezentációs gráfot?

Egy keresés fokozatosan fedezi fel a reprezentációs gráfot: bizonyos részeihez soha nem jut el, de a felfedezett részt sem feltétlenül tárolja el teljesen, sőt, sokszor torzultan "látja" azt: ha például egy csúcshoz érve nem vizsgálja meg, hogy ezt korábban már felfedezte-e, hanem új csúcsként regisztrálja, akkor az eredeti gráf helyett egy fát fog tárolni.

4.7 Reprezentációs gráf "fává egyenesítése"

Ha a keresés nem vizsgálja meg egy csúcsról, hogy korábban már felfedezte-e, akkor az eredeti reprezentációs gráf helyett annak fává kiegyenesített változatában keres.
Előny: eltűnnek a körök, de a megoldási utak megmaradnak.
Hátrány: duplikátumok jelennek meg, sőt a körök kiegyenesítése végtelen hosszú utakat eredményez.
A kétirányú (oda-vissza) élek szörnyen megnövelik a kiegyenesítéssel kapott fa méretét. De bármelyik keresésnél eltárolhatjuk egy csúcsnak azt a szülőcsúcsát, amelyik felől a csúcsot elértük. Így egy csúcsból a szülőjébe visszavezető él könnyen felismerhető és figyelmen kívül hagyható.

5 Probléma dekompozíció

Egy probléma dekomponálása során a problémát részproblémákra bontjuk, majd azokat tovább részletezzük, amíg nyilvánvalóan megoldható problémákat nem kapunk.
Sokszor egy probléma megoldását akár többféleképpen is fel lehet bontani részproblémák megoldásaira.

5.1 Dekompozíciós reprezentáció fogalma

A reprezentációhoz meg kell adnunk:
-a feladat részproblémáinak általános leírását
-a kiinduló problémát
-az egyszerű problémákat, amelyekről könnyen eldönthető, hogy megoldhatók-e vagy sem
-a dekomponáló műveleteket:
D: probléma $\rightarrow$ probléma$^+$
D(p)= $<p_1,...,p_n>$

5.2 A dekompozíció modellezése ÉS/VAGY gráffal

Egy dekompozíciót egy ún. ÉS/VAGY gráffal szemléltetjük: -egy csúcs egy részproblémát jelöl, a startcsúcs a kiinduló problémát, a célcsúcsok a megoldható egyszerű problémákat. -egy élköteg egy dekomponáló művelet hatását írja le, és a dekomponált probléma csúcsából a dekomponálással előállított részproblémák csúcsaiba vezet.

- egy élköteg élei mutatják meg, hogy a dekomponált probléma megoldásához mely részproblémákat kell megoldani. Az élköteg élei között ezért ún. "ÉS" kapcsolat van: hiszen minden részproblémát meg kell oldani.
- egy csúcsból több élköteg is indulhat, ha egy probléma többféleképpen dekomponálható. Ezen élkötegek élei között ún. "VAGY" kapcsolat áll fenn: hiszen választhatunk, hogy melyik élköteg mentén oldjunk meg egy problémát.

5.3 ÉS/VAGY gráfok

1. AZ R=(N,A) élsúlyozott irányított hiper-gráf, ahol az
   -N a csúcsok halmaza
   -A $\subseteq$ { (n,M)$\in$ NxN$^+$ | $0 \neq |M| < \infty$ } a hiper-élek halmaza, |M| a hiper-él rendje
   -(c(n,M) az (n,M) költsége)
2. Egy csúcsból véges sok hiper-él indulhat
3. $(0<\delta \leq c(n,M))$

5.4 Megoldás-gráf

Az eredeti problémát egyszerű problémákra visszavezető dekomponálási folyamatot az ÉS/VAGY gráf speciális részgráfja, az ún. megoldás-gráf jeleníti meg, amelyben:

-szerepel a startcsúcs

-a startcsúcsból minden más csúcsba vezet út, és minden csúcsból vezet út egy megoldás-gráfbeli célcsúcsba

-egy éllel együtt az összes azzal "ÉS" kapcsolatban álló él is (azaz a teljes élköteg) része a megoldás-gráfnak

-nem tartalmaz "VAGY" kapcsolatban álló él párokat

A megoldás a megoldás-gráfból olvasható ki.

5.5 Az n csúcsból az M csúcs-sorozatba vezető irányított hiper-út fogalma

Az $n^{\alpha}\rightarrow M$ hiper-út ($n\in N, M\in N^{+}$) egy olyan véges részgráf, amelyben:

-M csúcsaiból nem indul hiper-él

-M-en kívüli csúcsokból csak egy hiper-él indul

-minden csúcs elérhető az n csúcsból egy közönséges irányított úton.

A megoldás-gráf egy olyan hiper-út, amely a startcsúcsból csupa célcsúcsba vezet.

5.6 Hiper-út bejárása

Az $n\rightarrow M$ hiper-út egy bejárásán a hiper-út csúcsaiból képzett sorozatoknak a felsorolását értjük:

-első sorozat: <n>

-a C sorozatot a $C^{k\leftarrow K}$ sorozat követi (ahol a $k\in C$, és k minden C-beli előfordulásának helyén a K sorozat szerepel) feltéve, hogy a hiper-útnak van olyan (k,K) hiper-éle, ahol $k\notin M$.

5.7 Útkeresés ÉS/VAGY gráfban

Amikor a startcsúcsból induló hiper-utakat (ezek között vannak a megoldás-gráfok is, ha egyáltalán vannak ilyenek) a bejárásukkal írjuk le, akkor ezek a bejárások olyan közönséges irányított utak, amelyek csúcsai az eredeti ÉS/VAGY gráf csúcsainak sorozatai. Ezen utakból egy olyan közönséges irányított gráfot készíthetünk, amelyben a startcsúcs az ÉS/VAGY gráf startcsúcsából álló egy elemű sorozat, a célcsúcsait leíró sorozatok pedig kizárólag az ÉS/VAGY gráf célcsúcsainak egy részét tartalmazzák.

Ha ebben a közönséges gráfban megoldási utat találunk, akkor az egyben az eredeti ÉS/VAGY gráf megoldás-gráfja is lesz.

6 Keresések

6.1 KR vezérlési szintjei

Három féle vezérlési stratégiát különböztetünk meg:
-általános (független a feladattól, és annak modelljétől: nem merít sem a feladat ismereteiből, sem a modell sajátosságaiból.)
-modellfüggő (nem függ a feladat ismereteitől, de épít a feladat modelljének általános elemeire.)
-heurisztikus (a feladattól származó, annak modelljében nem rögzített, a megoldást segítő speciális ismeret)

Másik megközelítés alapján, kétféle általános stratégiát különböztetünk meg:
-nemmódosítható (lokális keresések, evolúciós algoritmus, rezolúció)
-módosítható (visszalépéses keresések, gráfkeresések)

6.2 Lokális keresések

A lokális keresés olyan KR, amely a probléma reprezentációs gráfjának egyetlen csúcsát (aktuális csúcs) és annak szűk környezetét tárolja (a globális munkaterületén). Kezdetben az aktuális csúcs a startcsúcs, és a keresés akkor áll le, ha az aktuális csúcs a célcsúcs lesz.
Az aktuális csúcsot minden lépésben annak környezetéből vett "jobb" csúccsal cseréli le (keresési szabály).
A "jobbság" eldöntéséhez (vezérlési stratégia) egy kiértékelő (cél-, rátermettségi-, heurisztikus) függvényt használ, amely reményeink szerint annál jobb értéket ad egy csúcsra, minél közelebb esik az a célhoz.

6.3 Hegymászó algoritmus

Mindig az aktuális (akt) csúcs legjobb gyermekére lép, amelyik lehetőleg nem a szülője.
(Megjegyzés: Az eredeti hegymászó algoritmus nem zárja ki a szülőre való lépést, viszont nem engedi meg, hogy az aktuális csúcsot egy rosszabb értékű csúcsra cseréljük, ilyenkor inkább leáll.)
Hátrányok:
Csak erős heurisztika esetén lesz sikeres: különben "eltéved" (nem talál megoldást), sőt zsákutcába jutva "beragad".
Segíthet, ha:
-véletlenül választott startcsúcsból újra- és újra elindítjuk (random restart local search)
-k darab aktuális csúcs legjobb k darab gyerekére lépünk (local beam search)
-gyengítjük a mohó stratégiáját (simulated annealing)
Lokális optimum hely körül vagy ekvidisztans felületen (azonos értékű szomszédos csúcsok között) található körön, végtelen működésbe eshet.
Segíthet ha:
-növeljük a memóriát (tabu search)

6.4 Tabu keresés

A globális munkaterületén az aktuális csúcson (akt) kívül nyilvántartja még:
-az utolsó néhány érintett csúcsot: Tabu halmaz
-az eddigi legjobb csúcsot: optimális csúcs (opt)

Egy keresési szabály minden lépésben
-az aktuális csúcsnak a legjobb, de nem a Tabu halmazban lévő gyerekére lép
-ha akt jobb, mint az opt, akkor opt az akt lesz
-frissíti akt-tal a sorszerkezetű Tabu halmazt

Terminálási feltételek:
-ha az opt a célcsúcs
-ha az opt sokáig nem változik

Előnyök:
Tabu méreténél rövidebb köröket észleli, és ez segíthet a lokális optimum hely illetve az ekvidisztans felület körüli körök leküzdésében.

Hátrányok: A tabu halmaz méretét kísérletezéssel kell belőni.
Zsákutcába futva nem-módosítható stratégia miatt beragad.

6.5 Szimulált hűtés

A keresési szabály a következő csúcsot véletlenszerűen választja ki az aktuális (akt) csúcs gyermekei közül.
Ha az így kiválasztott új csúcs kiértékelő függvény-értéke nem rosszabb, mint az akt csúcsé (itt f(új) $\leq$ f(akt)), akkor elfogadjuk aktuális csúcsnak.
Ha az új csúcs függvényértéke rosszabb (itt f(új) $>$ f(akt)), akkor egy olyan véletlenített módszert alkalmazunk, ahol az új csúcs elfogadásának valószínűsége fordítottan arányos az | f(akt) - f(új) | különbséggel:
$e^{\frac{f(akt)-f(uj)}{T}} >$ random[0,1]

6.6 Hűtési ütemterv

Egy csúcs elfogadásának valószínűségét az elfogadási képlet kitevőjének T együtthatójával szabályozhatjuk. Ezt egy (T_k,L_k) k=1,2,... ütemterv vezérli, amely L_1, majd L_2 lépésen keresztül T_2, stb. lesz.
$e^{\frac{f(current)-f(new)}{T_k}} >$ rand[0,1]
Ha T_1, T_2, ... szigorúan monoton csökken, akkor egy ugyanannyival rosszabb függvényértékű új csúcsot kezdetben nagyobb valószínűséggel fogad el a keresés, mint később.

6.7 Lokális kereséssel megoldható feladatok

Erős heurisztika nélkül nincs sok esély a cél megtalálására.

-Jó heurisztikára épített kiértékelő függvénnyel elkerülhetőek a zsákutcák, a körök.

A sikerhez az kell, hogy egy lokálisan hozott rossz döntés ne zárja ki a cél megtalálását!

-Ez például egy erősen összefüggő reprezentációs-gráfban automatikusan teljesül, de kifejezetten előnytelen, ha a reprezentációs-gráf egy irányított fa.

(Például az n-királynő problémákat csak tökéletes kiértékelő függvény esetén lehetne lokális kereséssel megoldani.)

6.8 A heurisztika hatása a KR működésére

A heurisztika olyan, a feladathoz kapcsolódó ötlet, amelyet közvetlenül építünk be egy algoritmusba azért, hogy annak eredményessége és hatékonysága javuljon (egyszerre képes javítani a futási időt és a memóriaigényt), habár erre általában semmiféle garanciát nem ad.