ÖSSZEFOGLALÁS

a Bsc záróvizsga mesterséges intelligenciáról szóló témaköréhez

Az MI az informatikának az a területe, amelyik az intelligens gondolkodás számítógépes reprodukálása szempontjából hasznos elveket, módszereket, technikákat kutatja, fejleszti, rendszerezi.

Miről ismerhető fel az MI?

- Megoldandó feladatai: nehezek. A feladat problématere hatalmas. A megoldás megtalálása intuíciót igényel, hogy annak keresése során kombinatorikus robbanás ne következzen be.
- A megoldást biztosító szoftverek intelligensen működnek. Úgy, mintha a háttérben egy ember tevékenykedne (Turing teszt), továbbá gyakran tanulnak működésük közben, és ennek következtében javulhat az eredményességük és hatékonyságuk.
- Megoldási módszereire az átgondolt reprezentáció és heurisztikával megerősített algoritmusok használata jellemző.

Sok feladat fogalmazható át útkeresési problémává.

- Adott egy *élsúlyozott irányított δ-gráf*, ahol a kivezető élek számára felső (σ), az élek költségére pedig pozitív alsó korlát (δ) van. Kijelölünk ebben egy startcsúcsot és egy vagy több célcsúcsot. Ez a *reprezentációs gráf*. Ebben keressünk egy startcsúcsból kiinduló célcsúcsba futó utat, esetleg keressük a legolcsóbb ilyet.
- Az úgynevezett *problématér* a startcsúcsból kiinduló utak halmaza, amelyben két út akkor szomszédos, ha az egyik egy éllel hosszabb a másiknál. Szűkíteni a problémateret bizonyos utak levágásával lehet. (pl. Hanoi tornyai probléma). Speciális esetben a problématér lehet a csúcsok halmaza, és a közöttük levő élek jelölik ki a szomszédsági kapcsolatokat. (pl. n-királynő probléma)

Állapottér-reprezentáció – egy probléma leíró (reprezentáló) modell

- Négy eleme van:
    - Állapottér, amely a probléma homlokterében álló adat (objektum) lehetséges értékeinek (állapotainak) halmaza, amelyet egy alkalmas invariáns leszűkíthet.
    - Műveletek (előfeltétel+hatás), amelyek állapotból állapotba vezetnek.
    - Kezdőállapot(ok) vagy azokat leíró kezdőfeltétel
    - Célállapot(ok) vagy célfeltétel.
- Az *állapot-gráf* az állapotokat, mint csúcsokat, a műveletek hatásait, mint éleket tartalmazó gráf, amelyben útkeresési feladatként lehet a problémát kitűzni.

Keresések

Egy általános *kereső rendszer* részei: a *globális munkaterület* (a keresés memóriája), a *keresési szabályok* (a memória tartalmát változtatják meg), és a *vezérlési stratégia* (alkalmas szabályt választ). A vezérlési stratégiának van egy általános, elsődleges eleme, lehet egy másodlagos (az alkalmazott reprezentációs modell sajátosságait kihasználó) eleme és a konkrét feladatra építő eleme. Ez utóbbi a *heurisztika*, a konkrét feladatból származó extra ismeret, amelyet közvetlenül a vezérlési stratégiába építünk be az eredményesség és a hatékonyság javítása céljából.

Lokális keresések

- A keresés egyetlen csúcsot (és annak környezetét) látja (tárolja a globális munkaterületen), amelyet minden lépésben a szomszédjai közül vett lehetőleg „jobb" gyerekcsúccsal cserél le (ez a csere a keresési szabály). A vezérlési stratégia a „jobbság" eldöntéséhez *célfüggvényt* (rátermettségi függvényt) használ, amely olyan heurisztikára épül, amely várhatóan annál jobb értéket ad egy csúcsra, minél közelebb esik az a célhoz. Mivel a keresés „elfelejti", hogy honnan jött, a döntések nem vonhatók vissza, ez egy *nem-módosítható vezérlési stratégia*.
- Lokális kereséssel megoldható feladatok azok, ahol egy lokálisan hozott rossz döntés nem zárja ki a cél megtalálását. Ehhez vagy egy erősen összefüggő reprezentációs-gráf, vagy jó heurisztikára épített célfüggvény kell. Jellemző alkalmazás: adott tulajdonságú elem keresése, függvény optimumának keresése.
- Nevezetes algoritmusok:
    - *Hegymászó algoritmus*: Minden lépésben az aktuális csúcs legjobb gyermekére lép, de kizárja a szülőre való visszalépést. Körök, ekvidisztans felület esetén eltévedhet.
    - *Tabu keresés*: Az aktuális csúcson (n) kívül nyilvántartja még az eddig legjobbnak bizonyult csúcsot (n^*) és az utolsó néhány érintett csúcsot; ez a (sor tulajdonságú) *tabu halmaz*. Minden lépésben az aktuális csúcs szomszédjai közül, kivéve a tabu halmazban levőket, a legjobba lép, frissíti a tabu halmazt, és ha n jobb, mint az n^*, akkor n^*-ot lecseréli. A tabu halmaz méretét nehéz belőni.
    - *Szimulált hűtés algoritmusa*: A következő csúcs választása véletlenszerű. Ha a kiválasztott csúcs (r) célfüggvény-értéke rosszabb, mint az aktuális csúcsé (n), akkor az új csúcs elfogadásának valószínűsége fordítottan arányos $f(n) - f(r)$ különbséggel.

Visszalépéses keresések

- A startcsúcsból az aktuális csúcsba vezető utat (és az arról leágazó még ki nem próbált éleket) tartja nyilván (globális munkaterületen), a nyilvántartott út végéhez egy új (ki nem próbált) élt fűzhet vagy a legutolsó élt törölheti (visszalépés szabálya), a visszalépést a legvégső esetben alkalmazza. A visszalépés teszi lehetővé azt, hogy a továbblépésről hozott korábbi döntést megváltozhasson. Ez tehát egy *módosítható vezérlési stratégia*. A keresésbe sorrendi és vágó heurisztika építhető. Mindkettő az aktuális csúcsból kivezető, még ki nem próbált élekre vonatkozik.
- Visszalépés feltételei: zsákutca, zsákutca torkolat, kör, mélységi korlát.
    - VL1 (nincs kör- és mélységi korlát figyelés) véges körmentes irányított gráfokon terminál, és ha van megoldás, akkor talál egyet.
    - VL2 (általános) véges δ-gráfokon terminál, és ha van megoldás a mélységi korlát alatt, akkor talál egyet.
- Könnyen implementálható, kicsi memória igényű, mindig terminál, és ha van, akkor megoldást talál. De nem garantál optimális megoldást, egy kezdetben hozott rossz döntést csak nagyon sok lépés után képes korrigálni és egy zsákutca-szakaszt többször is bejárhat, ha abba többféle úton is el lehet jutni.

Gráfkeresések

- A globális munkaterületén a startcsúcsból kiinduló már feltárt utak találhatók (ez az ún. *kereső gráf*), külön megjelölve az utak azon csúcsait, amelyeknek még nem (vagy nem eléggé jól) ismerjük a rákövetkezőit. Ezek a *nyílt csúcsok*. A keresés szabályai egy-egy nyílt csúcsot terjesztenek ki, azaz előállítják (vagy újra előállítják) az összes rákövetkezőjét. A vezérlési stratégia a legkedvezőbb nyílt csúcs kiválasztására törekszik, ehhez egy *kiértékelő függvényt* (*f*) használ. Mivel egy nyílt csúcs, amely egy adott pillanatban nem kerül kiválasztásra, később még kiválasztódhat, ezért itt egy módosítható vezérlési stratégia valósul meg.
- A keresés minden csúcshoz nyilvántart egy odavezető utat (π visszamutató pointerekkel), valamint az út költségét (*g*). Ezeket az értékeket működés közben alakítja ki, amikor a csúcsot először felfedezi vagy később egy olcsóbb utat talál hozzá. Mindkét esetben (amikor módosultak a csúcs értékei) a csúcs nyílttá válik. Amikor egy már korábban kiterjesztett csúcs újra nyílt lesz, akkor az esetleges korábban felfedezett leszármazottainál a visszafelé mutató pointerekkel kijelölt út költsége nem feltétlenül egyezik majd meg a nyilvántartott *g* értékkel, és az sem biztos, hogy ezek az értékek az eddig talált legolcsóbb útra vonatkoznak. Csökkenő kiértékelő függvényt (egy csúcs *f* értéke nem nő az algoritmus működése során, sőt, amikor egy csúcs visszakerül a nyílt csúcsok közé, akkor annak új *f* értéke kisebb annál, mint amivel onnan korábban kikerült) használva viszont a küszöbcsúcsok (minden korábbi kiterjesztésnél mért értéknél nagyobb vagy egyenlő *f* értékű csúcs) kiterjesztésének pillanatában ilyen inkonzisztencia garantáltan nem áll fenn.
- Nevezetes gráfkeresések:
    - Nem-informált gráfkeresések: *mélységi gráfkeresés* ($f = -g$, minden (n,m) élre $c(n,m)=1$), *szélességi gráfkeresés* ($f = g$, $c(n,m)=1$), *egyenletes gráfkeresés* ($f = g$)
    - Heurisztikus gráfkeresések (*h* a heurisztikus függvény, amely minden csúcsban a h^* hátralevő optimális költséget becsli): *előre tekintő gráfkeresés* ($f = h$), *A algoritmus* ($f = g+h$, $h \geq 0$), *A* algoritmus* ($f = g+h$, $h^* \geq h \geq 0$ – *h megengedhető*), A^C algoritmus ($f = g+h$, $h^* \geq h \geq 0$, minden (n,m) élre $h(n)-h(m) \leq c(n,m)$), *B algoritmus* (ahol $f= g+h$, $h \geq 0$ mellett a *g*-t használjuk két szomszédos küszöb csúcsok kiterjesztése közötti kiterjesztéseknél).
- Véges δ-gráfokon minden gráfkeresés terminál, és ha van megoldás, talál egyet. A nevezetes gráfkeresések végtelen nagy gráfokon is találnak megoldást (az előre-tekintő kivételével, és a mélységi gráfkeresés csak megfelelő mélységi korlát mellett), ha van megoldás.
- Az *A**, A^C algoritmusok optimális megoldást találnak, ha van megoldás. Az A^C algoritmus egy csúcsot legfeljebb egyszer terjeszt csak ki.
- Egy gráfkeresés futási idejét a kiterjesztések számával mérjük a kiterjesztett csúcsok számának függvényében. (Egy csúcs általában többször is kiterjesztődhet, de δ-gráfokban csak véges sokszor.) *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.

Kétszemélyes (teljes információjú, zéró összegű, véges***) játékok***

- a játékokat állapottér-reprezentációval szokás leírni, és az állapot-gráfot faként ábrázoljuk.
- A *győztes (vagy nem-vesztes) stratégia* egy olyan elv, amelyet betartva egy játékos az ellenfél minden lépésére tud olyan választ adni, hogy megnyerje (ne veszítse el) a játékot. Valamelyik játékosnak biztosan van győztes (nem-vesztes) stratégiája.
- Győztes (nem-vesztes) stratégia keresése a *játékfában* kombinatorikus robbanást okozhat, ezért részfa kiértékelést szoktak alkalmazni a soron következő jó lépés meghatározásához.
- A *minimax* algoritmus az aktuális állásból felépíti a játékfa egy részét, kiértékeli annak leveleit, majd szintenként váltakozva (ellenfél szintjein) minimum, illetve (saját szintjein) maximum értékeket futtat fel. A gyökérhez került érték jelzi a soron következő lépést.
- A *minimax* legismertebb módosítása az *alfa-béta* algoritmus, amely egyfelől kisebb memória igényű (egyszerre csak egy ágat tárol), másfelől egy sajátos vágási stratégia miatt jóval kevesebb csúcsot vizsgál meg.

Mesterséges neuronhálók

- Bemenő értékek (számok) együtteséből kimenő értéket (számokat) előállító rendszer, amely egymáshoz kapcsolódó, tanítható számoló egységekből áll.
- Mesterséges neuronháló részei: *mesterséges neuron* (bemenő értékekből egy kimenő értéket számoló egység), *hálózati topológia* (több hasonló mesterséges neuron egymáshoz kapcsolásának módja: irányított gráf), *tanulási szabály* (egy neuron számítási képletét módosító eljárás).
- Legegyszerűbb mesterséges neuronháló: perceptron modell. Lépcsős *aktivizációs függvényű* ún. Rosenblatt perceptronok (a bemenő értékeket egy-egy súllyal szorozza, a szorzatokat összegzi, és abból a lépcsős függvény segítségével kimenetet számol) egy rétegű előrecsatolt hálója, felügyelt tanulással (delta-szabály). A tanulás során egy perceptron *i.* bemenetéhez (x_i) tartozó súly (w_i) a várt (t) és a számított (o) kimenet különbégének függvényében változik: $w_i = w_i + \eta * x_i * (t - o)$. Ez a modell csak lineárisan szeparálható problémák megoldására alkalmas.
- Számos egyéb modell is létezik (Backpropagation, Hopfield, stb.)

Evolúciós algoritmus

- Egy adott pillanatban a problématérnek nem csak egy elemét (*egyedét*) tárolja, hanem azoknak egy részhalmazát (*populációját*), és arra törekszik, hogy a populáció egyedei minél jobbak legyenek (a helyes válaszhoz közelítsenek). Ennek megítéléséhez egy *rátermettségi függvényt* használ. Az egyedeket úgy kódolja, hogy azok tulajdonságai darabolhatóak legyenek: egy kódszakasz megváltoztatásával más tulajdonságú egyedet kapjunk.
- A populációt lépésről lépésre változtatja úgy, hogy egyedeinek egy részét azokra hasonló, de lehetőleg jobb egyedekre cseréli. (A populáció megváltozása visszavonhatatlan. Ez tehát egy nem-módosítható stratégiájú keresés.) Ehhez négy műveletet hajt végre:
    - *Szelekció (kiválasztás)*: Kijelölünk rátermett egyedeket (kevésbé rátermettek is kaphatnak esélyt)
    - *Rekombináció (keresztezés)*: A kiválasztott egyedek párjaiból utódokat állítunk elő.
    - *Mutáció*: Az utódokat kismértékben módosítjuk.
    - *Visszahelyezés*: Új populáció kialakítása az utódokból és a régi populációból.

Automatikus logikai következtetés

- A_1 , … , $A_n \Rightarrow C$ bizonyítására készült algoritmusok, amelyek válaszadásra is alkalmasak. Ez utóbbi esetben a „ki, mi, mit, kivel, mikor” stb. típusú kérdéseket a „van-e olyan aki, ami, amit, akivel, amikor” alakú mondatként formalizáljuk, és a választ a bizonyítás során az így bevezetett egzisztenciálisan kvantált változó helyébe került értékből nyerjük.
- <u>*Rezolúció*</u>.
    - Az A_1 , … , A_n , $\neg C$ kielégíthetetlenségét belátó módszer.
    - A formulákat KNF alakra (klózok konjunkciójára) kell hozni, amelyekből kiindulva újabb klózokat hozunk létre (rezolúciós lépés: pl. $r \vee s$, $\neg r \vee u \Rightarrow s \vee u$) egészen addig, amíg az ellentmondást jelző „üres klóz” meg nem jelenik.
    - Elsőrendű formulák esetében a KNF-re hozás speciális lépést, a Skolemizálást igényli, a rezolúciós lépéshez pedig változó-helyettesítést alkalmazunk.
    - Az elsőrendű rezolúció helyes, teljes, parciálisan eldönthető módszer.
    - Nem-determinisztikus algoritmus, amely egy nem-módosítható stratégiájú keresés. Gyorsításához számos másodlagos (sorrendi illetve szűkítő) stratégiát ismerünk.
- <u>*Szabályalapú következtetés*</u>.
    - Az axiómákat a bennük szereplő implikációk alapján tényekre és szabályokra osztjuk, és a következtetéshez az A, $A \rightarrow B \Rightarrow B$ sémát (modus ponens) használjuk.
    - A bizonyítás (következtetési lánc) olyan állítások sorozata, amelyben egy állítás vagy egy tény, vagy a sorozat megelőző állításaiból és egy szabályból levezethető állítás. A sorozat utolsó állítása a célállítás.
    - Kritikus kérdés annak eldöntése, hogy A', $A \rightarrow B$ formulák esetén alkalmazható-e a modus ponens (azaz illeszkedik-e A és A'), és mi ekkor a következmény. Ennek hatékony vizsgálatához a formulák alakjára korlátozásokat szoktak bevezetni, de ettől sérülhet a teljesség.
    - A következtetési láncot lehet előre- vagy visszafelé haladó módszerrel is keresni, de mindkét esetben egy visszalépéses kereséssel érdemes ezt végezni.
    - A formulák alakjának speciális megválasztása esetén a bizonyítás egy ÉS/VAGY gráfbeli hiperút keresésével állítható elő.
- Az emberi logika és a klasszikus logika számos ponton különbözik. Az emberi gondolkodás hiányos és ellentmondásos axiómarendszerrel is tud dolgozni; állításai sokszor nem egyértelműen igazak vagy hamisak, bizonytalanok; a következtetések sokszor procedurális (végrehajtható) jellegűek; az állítások a predikátum központú megfogalmazás helyett többnyire strukturált objektum elvű formában állnak rendelkezésre. Ezért van jelentősége a különféle alternatív következtetési módszereknek (zárt világ feltételezés, default következtetés, Bayes hálók, fuzzy következtetés, szemantikus hálók, leíró logikák, stb.).