Kísérletek minimax és alfa-béta nyesés algoritmusokkal kétszemélyes játékokban

Mesterséges intelligencia házi feladat

2014./2015. ősz

Tallér Bátor (UGA4OQ)

Török Attila (FULGDF)

Konzulens:  
 Balassa Gábor

Tartalomjegyzék

[Feladatválasztás 3](#__RefHeading__147_1848191074)

[A megvalósítás 4](#__RefHeading__151_1848191074)

[Az ágens implementációk 4](#__RefHeading__149_1848191074)

[Tapasztalatok 5](#__RefHeading__153_1848191074)

1. Feladatválasztás

Házi feladatunkként a minimax algoritmus kétszemélyes játékokban való alkalmazását, majd alfa-béta nyesés alkalmazásával való meggyorsítását, hatékonyabbá tételét választottuk. A konkrét játék, amelyet felhasználunk, a feladat szövegében is példaként említett nim, annak is az a változata, ahol nem elkerülendő, hanem a cél az utolsó kavics elvétele.

A szoftver, amelyet megvalósítunk, egyrészt egy általános nim-keretrendszer lesz, amely tetszőleges algoritmussal dolgozó ágenseket tud egymás ellen játszatni. Mérni fogja az időt, amely az egyes döntéseikhez szükséges volt, követni a játék állását (az egyes kupacokban pillanatnyilag lévő kavicsok számát), és kinyilvánítani a nyertest. Képes lesz továbbá mindezeket az adatokat naplózni is, későbbi elemzés céljából. A skálázhatóságot az adja, hogy a játékok elején beállítható lesz, hogy hány kupac legyen, és mennyi kavics legyen bennük.

Másrészt pedig nyilván tartalmazni fog néhány ágens-megvalósítást is, amelyekkel a méréseket elvégezzük. Az egyik természetesen a minimax stratégiát fogja alkalmazni, opcionálisan beállítható felső korláttal a keresőfájának mélységére. Egy másik hasonlóan fog működni, de az alfa-béta nyesést is alkalmazza, gyorsabb döntések reményében. Ezeken felül még több primitív ágenst is tervezünk megvalósítani, (például véletlenszerű kupacból vesz el véletlenszerű számú kavicsot, vagy mindig a legelső, nem üres kupacból vesz el egyet, és hasonlók), amelyek főként ellenségként fognak szolgálni a bonyolultabb algoritmusúak ellen, hogy a játékok automatikusan lefuttathatóak és tesztelhetőek legyenek.

A minimax stratégiát alkalmazó ágenseknél az egyszerűség kedvéért azt feltételeztetjük az ágenssel, hogy az ellenfele ugyanazokat a döntéseket hozza az egyes helyzetekben, mint ő maga tenné. Emellett az az érvünk, hogy a minimax elv feltételezi, hogy mindkét játékos optimálisan játszik. Amennyiben tehát az ágens a saját stratégiáját optimálisnak hiszi, alkalmazhatja azt az ellenség szemszögéből is. Ha pedig mégis tudna jobbat, amit az ellenség alkalmazhatna szerinte, akkor ő maga is aszerint játszana, tehát mégis egyformán játszanának.

Ezek mellett, amennyiben időnk engedi, beépítenénk egy „emberi játékos” ágenst is, amely cselekvéseit a felhasználó határozza meg, így ezen keresztül tulajdonképpen bármilyen algoritmus kipróbálható lesz a szoftverben, annak módosítása, vagy további bővítése nélkül is.

Megvalósításhoz a C# nyelvet választottuk, a grafikus felülethez a jól megszokott Windows Forms technológiát fogjuk használni.

1. A megvalósítás
   1. Az ágens implementációk

**RandomHeapAgent:** Minden körben egy véletlenszerűen kiválasztott, még nem üres kupacból vesz el pontosan egy követ.

**RandomAgent:** Minden körben egy véletlenszerűen kiválasztott, még nem üres kupacból vesz el véletlenszerű számú, de legalább egy követ.

**MinimaxAgent:** A pillanatnyi állapotból kiindulva bejárja a teljes állapotfát, és a minimax algoritmus szerint értékeli az egyes lépések eredményességét. Amennyiben egy játékterv végén az ágens nyerne, 1-es pontszámot rendel hozzá, amennyiben vesztene, -1-est, ha pedig a játszma véget érte nélkül elérte a bejárási mélység-korlátot, jobb ismeret híján 0-val fog számolni.

**PruningMinimaxAgent:** Ugyanúgy működik, mint a sima MinimaxAgent, de már alfa-béta nyesést is alkalmaz, így jelentősen lecsökken a fa bejárandó részeinek száma, ezáltal a döntési sebesség jelentősen nő, anélkül, hogy ez bármilyen hátránnyal járna a döntések minőségét illetően.

**SummingMinimaxAgent:** Egy alternatív pontszámítást használó, minimaxhoz hasonló elven működő ágens, amely nem egyszerűen minimumot és maximumot keres az egyes csomópontokban, hanem összegzi az adott részfából kiindulva elérhető végállapotok pontszámát, így egy finomabb felbontású értéket kap egy-egy lépés jóságáról, ami nagy mértékben javítja a meghozott döntések eredményességét. Ennek viszont az a hátulütője, hogy nem lehet nyesni a fát, hiszen az összegzés miatt az egyes csomópontok értékének meghatározásához mindenképp teljesen be kell járni a hozzájuk tartozó részfát. Ezen felül a bejárási mélység korlátozása is jelentősebb romlást okoz az ágens döntéseinek jóságában.

1. Tapasztalatok