Debreceni Egyetem
Informatikai Kar

Kétszemélyes logikai játékok

Témavezető:
Mecsei Zoltán
Egyetemi tanársegéd

Készítette:
Szabó Dániel András
Programtervező Informatikus

Debrecen
2010

Ezúton szeretnék köszönetet mondani:

Témavezetőmnek, Mecsei Zoltánnak,
valamint
Kovács Tímeának
Kovács Tibornak
Tarcsa Bálint Dávidnak

Tartalomjegyzék

„Minden úgy működik, ahogy elvárjuk. Feltéve, hogy elvárásaink helyesek.”
(Hyman Rosen)

Bevezetés

Az informatikán belül jó pár évtizedre tekint vissza a mesterséges intelligencia. Ehhez képest még ma is sokféle megközelítés létezik, hogy mit is takar ez a fogalom tulajdonképpen. Rendkívül szerteágazó ez a tudományterület, és viszonylag kevéssé ismert, pedig a hétköznapi életben is sokszor találkozunk vele.

A kutatók kezdetben úgy gondolták, hogy amiként a numerikus számítási feladatokban a számítógép hamar túlszárnyalta az ember képességeit, úgy tulajdonképpen akármilyen intelligenciát követelő feladatban belátható időn belül helyettesíteni fogja tudni az embert. Ezek a remények nem váltak be, viszont egyes szűk területeken nagy eredmények születtek. Bizonyos dolgokban, amire azt mondjuk, hogy intelligenciát követelnek, valóban az embert akár helyettesíteni, de mindenképpen segíteni tudja a számítógép. Egyik tipikus terület a kétszemélyes logikai játékok. Itt napjainkban vagyunk azon a határon, hogy a gép már esélyes ellenfele lehet egy logikai játékban még a profi játékosoknak is. Ennek a dolgozatnak a célja, hogy vázolja a mesterséges intelligencia legfontosabb területeit, és részletesen bemutassa, hogy a kétszemélyes logikai játékok hogyan „gondolkodnak”, hogyan valósul meg bennük a mesterséges intelligencia.

1. Mesterséges intelligencia

A mesterséges intelligenciával a 20. század közepétől kezdett foglalkozni a tudomány. Maga a kifejezés John McCarthytól származik, aki 1956-ban használta egy a témának szentelt konferencián.

1.1 Mi a mesterséges intelligencia?

Ennek a kérdésnek a megválaszolásához tudnunk kéne, hogy mi is az intelligencia maga. Emberekről nagyfokú egyetértéssel állapítjuk meg, hogy intelligens, sőt össze is tudjuk hasonlítani másokkal, miközben nem sok fogalmunk van róla mi is valójában, hogyan definiálható az intelligencia. Legfeljebb néhány összetevőjét tudjuk felsorolni, olyan tulajdonságokat, amelyekről feltételezzük: szükségesek ahhoz, hogy az emberi magatartást, viselkedést intelligensnek mondhassuk. (Az intelligencia bizonyos mennyiségű tudást már eleve feltételez. Ilyenek: a problémamegoldó képesség, az absztrahálás és általánosítás képessége, az analógiák felismerése különböző szituációkban, az alakfelismerés, a nyelv megértése és helyes használata, az alkalmazkodás váratlan új helyzetekhez, a tanulás képessége, saját hibáink felismerése és kijavítása. Ennél fogva tehát a mesterséges intelligencia meghatározása is bizonytalan, a szakirodalom is különböző megközelítéseket alkalmaz, általában attól függően, hogy mely szakterületen dolgozó tudós fogalmazta meg. Nézzünk néhány definíciót:

Bellman, 1978: *„Az MI az emberi gondolkodáshoz asszociált tevékenységek, mint a döntéshozatal, problémamegoldás, tanulás automatizálása vizsgálata."*

Haugenland, 1985: *„Az MI egy izgalmas erőfeszítés a számítógépek gondolkodóvá tételére, értelemmel bíró gépek létrehozására a szó szoros értelmében."*

Charniak, 1985: *„Az MI a mentális képességek tanulmányozása számítógépes modellek segítségével.”*

Kurzweil, 1990: *„Az MI olyan funkciók megvalósítására alkalmas gépek tudománya, mely funkciókhoz intelligenciára van szükség, amennyiben azokat emberek valósítják meg.”*

Schalkoff, 1990: *„Az MI olyan tanulmányterület, amely számítási eljárásokkal próbálja magyarázni és utánozni az intelligens viselkedést.”*

Rich, 1991: *„Az MI annak tanulmányozása, hogyan lehet számítógéppel olyan dolgokat tenni, melyeket jelenleg az emberek jobban tudnak.”*

Winston, 1992: *„Az MI az érzékelést, gondolkodást és cselekvést lehetővé tevő számítások (computation) tanulmányozása.”*
Luger, 1993: *„Az MI a számítástudomány azon ága, mely az intelligens viselkedés automatizálásával foglalkozik.”*

Egyébként is, ahogy Rafael Capurro hangsúlyozza, az intelligens rendszerek kidolgozásánál nem arra kell törekedni, hogy az emberhez hasonlítsanak, mivel az "emberi elme" fogalma sokkal gazdagabb és szélesebb, és nem redukálható az intelligencia fogalmára.

Don Thomasson véleménye szerint az intelligencia egy fontos, alapvető sajátossága, hogy váratlan helyzetekre megfelelő, hasznos válaszokat ad. Ilyen értelemben a szakértői rendszereknek nincs intelligenciájuk. Nem valószínű, hogy sikerül eredeti gondolkozású MI-t létrehozni. A számítógépnek nincs véleménye, nincsenek érzelmei. Lehet arra programozni, hogy tanuljon, de az ember akar tanulni.

Searle pedig megállapította, hogy csak az "Én" képes gondolkozni, a számítógépnek azonban nincs "Én"-je, ezért csak szimulálhatja a gondolkodást.

Vagy ahogy Mérő László írja: *"Ha a mesterséges intelligencia 30 éves történetének tanulságait akarjuk összefoglalni, azt mondhatjuk, hogy az értelem az eddig kidolgozott rendszerek egyetlen szintjén, egyetlen komponensében sem érhető tetten ... Minden esetben a felhasználó az, aki az értelmet belevetíti a mesterséges intelligencia termékeibe".*

P. Arnhem véleményét se hagyjuk figyelmen kívül: a biológiai intelligencia a fejlődés terméke szorosan kapcsolódik a szervezethez, amelyet szolgál. Az élet, úgy tűnik, előfeltétele az intelligens elménék.

Számos olyan feladat létezik, melyek megoldásához véleményünk szerint intelligenciára lenne szükség (például aritmetikai feladatok), a gép azonban könnyedén meg tudja oldani. Viszont sok olyan probléma is van (például arcfelismerés), melyet az ember automatikusan old meg, anélkül, hogy elgondolkodna rajta, automatizálása mégis rendkívül bonyolult.
Ha figyelembe vesszük azt, hogy az algoritmizálható szellemi tevékenységek a számítógép rutinfeladatai, akkor a mesterséges intelligenciára a következő meghatározást adhatjuk: olyan fejlett programrendszer, amely nemdeterminisztikus problémák osztályára alkalmazható. Azok a problémák tartoznak ide, amelyekre nem dolgozható ki algoritmus, vagy ha igen, a lépések száma túllépi a kombinatorikai robbanás határát, továbbá azok, amelyek megoldásához hiányos, esetleg pontatlan információkkal rendelkezünk.

Ami pedig a természetes és mesterséges intelligencia közötti kapcsolatot illeti, a szakemberek, pszichológusok, fiziológusok, informatikusok még nem tudták eldönteni a kérdést, hogy van-e valamilyen egyezés a számítógépben lezajló folyamatok és az ember kognitív folyamatai között. Az a tény, hogy bizonyos mentális funkciókat géppel tudunk elvégeztetni, azt jelzi, hogy az emberi agy és az MI néhány funkcionális paramétere között alapvető hasonlóság van, de nem bizonyíték a természetes és mesterséges intelligencia azonosságára.

Egyáltalán, mikor mondhatjuk azt, hogy egy gép emberhez hasonlóan intelligensen cselekszik? Ennek a kérdésnek az eldöntésére Allan Turing angol matematikus dolgozott ki egy tesztet. A teszt lényegileg abból áll, hogy az ember pusztán billentyűzet és monitor közvetítésével kérdéseket tesz fel a két tesztalanynak, akiket így se nem láthat, se nem hallhat. A két alany egyike valóban ember, míg a másik egy gép – és mindketten megpróbálják

meggyőzni a kérdezőt arról, hogy ők gondolkodó emberek. Ha a kérdező hosszadalmas faggatás után sem tudja teljesen egyértelműen megállapítani, hogy a két alany közül melyik az ember, akkor a gép sikerrel teljesítette a tesztet. Ilyen számítógépet a mai napig nem sikerült építeni.
Az eddigiek alapján azt mondhatjuk, hogy a MI-rendszerek, nem általánosan intelligensek, hanem bizonyos jól specializált feladatokra nézve.

Az intelligens rendszerek jellemző vonásai:
- Elsősorban és főleg nem numerikus szimbólumokkal operálnak. A feladatokat nem pontos algoritmus alapján oldják meg, hanem heurisztikusan. A heurisztikáról ezt írja Pásztor Zoltán és Sántáné: "A heurisztika kifejezés hosszú idő óta kulcsszónak számít a mesterséges intelligencia területén. Jelentése azonban a különböző szerzőknél és különböző időpontokban más és más." S a következő definíciót adják: *"Tapasztalaton alapuló módszer, pl. egyszerűsítő feltevés vagy más hasonló eszköz, amely korlátozza vagy egyszerűsíti a megoldás keresését bonyolult, nagyméretű, illetve kevésbé megértett problémák feladatterében. Az algoritmusoktól eltérően a heurisztikák nem biztosítják az általuk szolgáltatott megoldás hibátlanságát"*
- Olyan tudáskészlettel rendelkeznek - a valóság adott szeletének modelljével -, amely az ember számára érthető, a tudásalap világosan el van választva a tudást felhasználó mechanizmustól. A rendszer képes megoldást találni olyan esetekben is, amikor nem áll rendelkezésre a feladat megoldásához szükséges összes adat (ez a feladat természetéből is következhet). Ilyenkor a megoldás, a következtetés nem lesz teljesen biztos, sőt téves is lehet.
- A rendszer képes megoldani olyan feladatokat is, amelyekben az adatok egymásnak ellentmondók; ilyenkor az emberhez hasonlóan, azt a megoldást választja, amely leginkább összhangban van az ismereteivel.
- A rendszernek, ha igényt tart az intelligens jelzőre - rendelkeznie kell a tanulás képességével is.

A MI-rendszerekben tehát a tudás- vagy ismeretbázis játssza a főszerepet. Abban különbözik a számítástechnika más területein alkalmazott konvencionális adatbázisoktól, hogy különböző típusú ismeretekből tevődik össze: a tárgyakra, az eljárásokra, a folyamatokra vonatkozó ismeretekből, az úgynevezett mindennapi tudásanyagból, a célokra, szándékokra,

motivációkra, okozati összefüggésekre vonatkozó információkból. Az ismeretek ilyen széles skálájának megragadása, ábrázolása, strukturálása úgy, hogy minél pontosabb megfelelés legyen a való világ és az ismeretbázis között, és alkalmas legyen a szükséges műveletek elvégzésére, bonyolult feladat. (Hogy a megfelelő forma milyen fontos, azt a tudománytörténet bizonyítja. A rómaiak számírásuk miatt nem váltak jó matematikussá.) A bázishoz hozzá kell rendelni egy olyan szabályrendszert, amelynek segítségével az explicit tudásanyagból következtetések útján ki lehet bontani a benne rejlő implicit tudást.

A MI-kutatás a számítógép-tudomány önálló területe, de számos tudományterülettel rokon. Ilyen területek a pszichológia, filozófia, nyelvtudomány, számítástudomány, elektronika.

Az elméleti mesterséges intelligencia kutatása során két nézet bontakozott ki. Ez egyiket erős, a másikat gyenge mesterséges intelligenciának nevezték el. A fogalmat 1980-ban John R. Searle filozófus vezette be.

A gyenge mesterséges intelligencia

Gyenge mesterséges intelligenciának nevezik azt az álláspontot, amely szerint ki lehet alakítani olyan rendszereket, amelyek úgy cselekszenek, mintha intelligensek lennének, de a gyenge MI semmit nem mond arról, hogy egy ilyen gép valóban rendelkezik-e elmével vagy sem.

Az erős mesterséges intelligencia

Az erős mesterséges intelligenciának nevezett álláspont szerint olyan rendszerek is kialakíthatóak, melyek valóban gondolkodnak, tehát elmének tekinthetőek. Ez alapján az erős MI fő kérdése, hogy: egy megfelelően programozott számítógép tekinthető-e elmének, abban az értelemben, hogy egy ilyen számítógép valóban megért dolgokat?

1.2 A kezdetek

A mesterséges intelligencia kezdetei Neumann János magyar származású matematikus - a „számítógép atyja” - és Alan Turing brit matematikus nevéhez köthetők. A számítógépet megalkotói eleve olyan feladatok elvégzésére szánták, melyeket az ember az értelmével végez, ám életének első évtizedeiben csak nagyon egyszerű, könnyen algoritmizálható feladatok elvégzésére volt képes.

A szakemberek már kezdettől fogva arra törekedtek, hogy komolyabb munkára fogják a "buta óriást". Warren Weaver 1946-ban felvetette a lehetőségét annak, hogy a számítógépet fordításra használják. Ő és Donald Booth olyan módszerre gondolt, mint amilyennel a titkosírásokat fejtik meg. A betűk és szavak gyakorisága alapján, tehát pusztán formális alapon akarták megoldani a feladatot. Claud Shannon 1950-ben megjelent sakkprogramozással foglalkozó tanulmánya a sakkprogramok hosszú sorát indította el. 1958-ban a Nobel díjas Herbert Simon azt jósolta, hogy hat éven belül számítógép lesz a sakkvilágbajnok, és tíz éven belül megjelennek a kiváló fordítóprogramok. Egyik célt sem sikerült elérni. A hatvanas évekre megcsappant az érdeklődés e problémák iránt, mert leküzdhetetlen nehézségekbe ütköztek.

Csak mintegy tíz év múlva indultak meg újra, a határtudományokban - lélektanban, nyelvészetben, neurobiológiában - elért eredményekre támaszkodva, most már egyre nagyobb intenzitással a kutatások azokon a területeken, amelyeket a mesterséges intelligencia fogalomkörébe sorolunk. Különösen a kognitív pszichológiában elért eredmények vitték előre a mesterséges intelligenciakutatást, ahogy Darab Tamás írja: *"a mesterséges intelligencia kutatása és a kognitív pszichológia szemmel láthatóan egymással kölcsönhatásban fejlődnek. Míg az utóbbi tudomány művelői a számítógép működéséről szerzett újabb ismeretek figyelembevételével dolgozzák ki elméleteiket, addig a mesterséges intelligencia kutatói pontosan ezen elméletekből kiindulva minél inkább intelligens módon működő számítógépeket próbálnak létrehozni"*.

Peter Jackson a mesterséges intelligencia fejlődési szakaszát három periódusra osztotta:

A klasszikus periódus (1955-1965)

Az első fázisban olyan elvek után folyt a kutatás, melyek általában, azaz bármely probléma felmerülése kapcsán alkalmazhatók lettek volna. Az 1957-ből származó GPS (General Problem Solver) program fejlesztői (Newell, Simon, Shaw) elérték ugyan, hogy az eljárás bizonyos játékokat jól tudott játszani, de általános problémamegoldó stratégiának nyoma sem volt a rendszerben. Az eljárás készítőinek véleménye szerint nem is tanácsos GPS-t továbbfejlesztése érdekében fáradozni, ennek megvalósítása ugyanis lehetetlennek látszik.

A romantikus periódus (1965-1975)

Az újabb kutatások során már konkrét problémákat akartak megoldani, de még mindig általános jellegű megoldási elvek után kutatva. Az eredmények kezdetlegesek voltak, s a gyakorlati felhasználásuk csak álom volt.

A modern periódus (1975-)

A modern periódus kezdetét ahhoz a kijelentéséhez köthetjük, mely szerint nem az általános elvek, hanem a gyakorlatorientált, problémaspecifikus tudás az, amely a megoldásokat nagyban előremozdítja. Így alakultak ki a szakértői rendszerek is, melyek az emberi szakértők gépi szimulálásának tekinthetők.

1.3 A mesterséges intelligencia alkalmazásai

Az emberi agyat lényegében egy bunyolult gépnek tekintették csupán, mely így egy kellően bonyolult számítógéppel modellezhető. Így a kezdeti törekvések arra irányultak, hogy egy általános gépi intelligenciát hozzanak létre, mely általánosságban képes helyettesíteni az embert.
A mesterséges intelligencia kutatások azonban szűkebb területeken nagy hasznot halytottak és viszonylag jól használható alkalmazások születtek. Néhány ilyen tudományterület:

A természetes nyelv értelmezése

Ami a fordítást illeti, már 1946-ban felmerült az ötlet, hogy a számítógépet használják fel erre a célra. Mintegy másfél évtizeden keresztül folytak a kísérletek, részeredmények is születtek (az első fordítást egy IBM 701-es számítógépen készítették 1954. január 7-én, oroszból angolra), de bizonyos szintet nem sikerült meghaladni. A fordítások alig érték el az érthetőség határát, és sokszor vicclapokba illő "ferdítések" kerültek ki a számítógépből. A hatvanas években emiatt a gépi fordítás iránti érdeklődés megcsappant. Ugyanígy a kutatásokra fordított pénz is. Mi volt ennek az oka?

Az informatikusok, s a kérdéssel foglalkozó nyelvészek azt hitték, hogy a nyelvet le lehet redukálni szókészletre és szintaxisra, és a számítógépnek elég betáplálni a két nyelv szótárát, morfológiai és szintaktikai szabályait.

Hogy miért olyan nehéz megtanítani a gépet az emberi nyelvre, annak oka elsősorban a sokértelműség, amely még az egyszerű mondatot is jellemzi. Az angol nyelvű szakirodalomban előszeretettel idézik a következő példamondatot: Time flies like an arrow. Minden normális ember csak egyféleképpen értelmezi ezt: az idő úgy repül, mint egy nyílvessző. A számítógépes elemzés azonban ezen kívül még háromféle értelmet "magyaráz bele":
- az időlegyek (lásd gyümölcslegyek) szeretik a nyilat;

- mérd az idejét (=time) azoknak a legyeknek, amelyek olyanok, mint egy nyíl;
- úgy mérd az idejét a legyeknek, mint egy nyíl.

A kudarcból a szakemberek rájöttek arra, hogy a számítógép csak akkor tud jó fordítást készíteni, ha "érti" is a nyelvet, azaz a fordításhoz szemantikai, sőt pragmatikai ismeretekre is szükség van. A következő évek kutatásaiban a nyelvészek vették át a vezető szerepet, s a kutatás súlypontja az elméleti kérdésekre helyeződött át. Megszületett egy új tudományos diszciplína is, a számítógépes nyelvészet. Ez nem a számítógép-tudomány és a nyelvészet kereszteződése, hanem alkalmazott nyelvtudomány. A nyelv mélyebb, alaposabb megismerése, a számítástechnika rohamos fejlődése, s újabban a mesterséges intelligencia kutatásában elért eredmények jelentős haladást eredményeztek. Olyan fordítórendszereket sikerült létrehozni, amelyek most már szemantikai és szövegkörnyezeti tényezőket, körülményeket, oksági összefüggéseket is figyelembe tudnak venni. Egyelőre ezek a magas szintű fordítórendszerek csak a kutatóintézetekben léteznek, de már a kereskedelmi forgalomba került rendszerek is sokat tudnak. Példaként említhetjük a LOGOS rendszert.

Mesterséges látás, képfelismerés

A kép- és alakzat-felismerés egyike azoknak a területeknek, amelyeken az ember sokkal tökéletesebb, mint a gép. Nekünk egy szempillantás és néha nagyon kevés információ elég ahhoz, hogy felismerjük embertársunkat, egy tájat, utcarészletet. Még nem is látjuk tisztán sem az arcát, sem az alakját, csupán a körvonalait, a járását, egy jellegzetes mozdulatát, s máris felismerjük barátunkat, kollégánkat vagy akárkit, akit néhányszor láttunk, még akkor is, ha bizonyos jellegzetességeik megváltoztak (szemüveges lett, megfestette a haját, stb.). A gépnek ehhez rendkívül sok információra van szüksége: az alak pontjainak egymáshoz viszonyított helyzetére, viszonyára a környezetéhez, térbeli elhelyezkedésére. Mindezek leírásához rengeteg pont koordinátáit kell megadni, s a színek, árnyalatok, a megvilágítási értékek jellemzésére az adatok további tömege szükséges.
A kérdésnek három területen van nagy fontossága: a robotikában, az írás felismerésében és az űrkutatásban.
Az elmúlt években az alakzat-felismerés területén is látványos volt a fejlődés.

Ennek ellenére a számítógép "látása" még messze elmarad az emberétől. A kutatásban két tendencia érvényesül, amelyek némileg ellentmondanak egymásnak: egy-egy konkrét feladat minél gyorsabb megoldása az egyik oldalon, az emberi látás tanulmányozása alapján egy általános metodológia kidolgozása a másikon. A baj ott van, hogy sok gyakorlati megoldás nem általánosítható, s az elméleti tanulmányok nagy részét nem alkalmazzák a gyakorlatban.

Tudásalapú vagy szakértői rendszerek

Olyan rendszerek, amelyek szakértői tanácsokat, diagnózisokat, ajánlásokat tudnak nyújtani a valós világ problémáira.
A szakértői rendszerek olyan valós problémákat oldanak meg, amelyekhez normál esetben szakértőre lenne szükség (például szakorvos). Előfordulhat, hogy szakértőből hiány van, vagy csak nehéz rá szert tenni sürgősen. Ezzel szemben a szakértői rendszer könnyen elérhető. Például előfordulhat, hogy szeretnénk megtudni egy szakértői rendszer véleményét ritka betegségek diagnózisa ügyében. Nem biztos, hogy egy általános gyakorló orvosnak megvan hozzá a megfelelő szaktudása, szakértőt nehéz keríteni, a gyors döntés azonban rendkívül fontos.
A szakértő rendszerek párbeszédes üzemmódban működnek: kérdéseket tesznek fel, amelyekre a felhasználó a szükséges információkkal válaszol. A párbeszéd mindaddig tart, amíg a rendszer elegendő információ birtokába jut, "levonja" a következtetéseket, s közli véleményét.

Például egy hibadetektáló szakértő rendszer tudásbázisában a következő szabály található:
HA AZ AUTÓ MOTORJA NEM INDUL BE
és
A DUDA NEM SZÓL
és
A LÁMPÁK HALVÁNYAN ÉGNEK
akkor
AZ AKKUMULÁTOR KI VAN MERÜLVE 90% VALÓSZÍNŰSÉGGEL.

Ha a felhasználó közli a rendszerrel, hogy a motor nem indul be, a rendszer megkérdi: "A DUDA SZÓL?" Nemleges válaszra jön az újabb kérdés: "A LÁMPÁK HALVÁNYAN ÉGNEK?" Ha erre igenlő választ kap, közli a diagnózist.
A szakértő rendszerek döntéseiket is, a feltett kérdéseket is meg tudják indokolni. Ezzel megkönnyítik saját hibáik felderítését, s növelik egyúttal a felhasználók bizalmát döntéseik iránt.

Robotika

A robottechnika és az MI viszonyáról Michael Brady ezt írja: "A robottechnika kihívást jelent az MI számára azáltal, hogy arra készteti, a reális világ reális tárgyaival foglalkozzék" (Brady et al., 1984). Ez a kihívás akkor jelentkezett, amikor a robotok odáig fejlődtek, hogy "észre" volt szükségük.
A robot elnevezés sok mindenre használható, a viszonylag egyszerű programozható kezelőkészülékektől kezdve, amelyeket autók összeszerelésére használnak, egészen a sci-fi irodalom intelligens humanoid robotjaiig. Az okosabb robotokat tekinthetjük intelligens ágenseknek, amelyek a fizikai világban működnek. A robot emberi feladatokat hajt végre, de megvannak a maga céljai, és képes arra, hogy viselkedését az új információk alapján módosítsa. Az intelligencia első feltétele, hogy a robot érzékelje a környezetét, tájékozódni tudjon benne. Ehhez "érzékszervekre" van szüksége. Mind ez idáig a látás, tapintás és újabban a hallás képességével ruházták fel a robotokat. És itt kapcsolódik a robotika az MI-kutatáshoz.

1.4 Játékok és a mesterséges intelligencia

A számítógépes játékok esetén a tudományos megközelítéstől eltérően értelmezik az MI-t. Itt talán Kurzweil definíciója áll a legközelebb a valósághoz: *„Az MI olyan funkciók megvalósítására alkalmas gépek tudománya, mely funkciókhoz intelligenciára van szükség, amennyiben azokat emberek valósítják meg.”*

Legjobban akkor felel meg a egy program a átékos elvárásainak, ha nem buta, ugyanakkor nem is legyőzhetetlen. A gépi ellenfelek valószerűen viselkednek, úgy, ahogyan azt egy élő ember is tenné, és megfelelő reakciókat adnak a játékos cselekedeteire. A játékok nem a hagyományos értelemben vett MI-t valósítják meg, hanem különféle, technikák, trükkök segítségével az intelligens viselkedés látszatát igyekszenek kelteni. Természetesen tudományos megközelítések és módszerek is szóba jöhetnek.

A játékok történetének hajnalán a számítógépek teljesítménye annyira kicsi volt, hogy játékfejlesztők többsége még csak nem is gondolt arra, hogy a program valamennyire is intelligensen viselkedjen. *„…az olyan játékokban, amikor a játékos a géppel, mint ellenféllel kerül szembe, ismét két végletes eset különböztethető meg. Az egyik - egyszerű - típusnál a számítógép véletlen döntéseket hoz, össze-vissza szaladgál, míg a másiknál alaposan átgondolt stratégiát követ, amely mindig a pillanatnyi helyzet elemzésével indul, és határozott célok elérésére törekszik. Mondani sem kell, hogy az utóbbival való játék általában élvezetesebb, és a játékos nem, vagy nem olyan hamar unja meg. Az utóbbi tipikus példái a sakkprogramok, az előbbi sajnos a játékok 80%-a."*

Ettől függetlenül az akkoriban használt számítógépeken (ez többnyire Commodore és Sinclair gépeket jelentett) is voltak olyan játékprogramok, amelyek bizonyos szinten használtak MI-t. Például Pacman, Archon nevű játékok valamint sakkprogramok. A Commodore VC20-on is létezett olyan sakkprogram (Sargon Schack cartridge), amely egészen jól játszott. Korai stratégiai játékra jó példa a Defender of the crown, melyet 1986-ban adtak ki Commodore Amigára, malyd később más géptípusokra is elkészült.

A hardverek fejlődésével aztán új utak nyíltak meg a fejlesztők előtt, hogy a játékok viselkedését viszonylag hihetővé tegyék. A korai programokban gyakran előfordultak hibák, főként az útkereső algoritmusok voltak gyengék, melynek következtében a messzebre kiküldött egységek sűrűn elakadtak. Manapság a fejlesztők már komolyan veszik az MI kérdését olyannyira, hogy ma már az egyik legkomolyabb szempont a játékosok és a fejlesztők számára is, a játék mellett pedig az egyik legkomolyabb fegyvertény.

1. A Defender of the crown stratégiai játék.

A számítógépes játékok osztályozhatók az MI alapján:

1. A játékban nincs MI. Itt nincs szükség arra, hogy a játék intelligensen viselkedjen. Ilyen például az Aknakereső, de ide sorolhatjuk az ügyességi és logikai játékok jelentős részét is. Ezekben a játékokban a program csak egy kezdő állapotot állít elő, ellenőrzi, hogy a játékos a szabályoknak megfelelően lépett-e, adminisztrálja a játék menetét, esetleg statisztikákat készít stb.

2. Kétszemélyes, teljes információjú játékok. Ezzel később részletesen is foglalkozunk. Erre a játéktípusra a tudományos MI is figyelmet fordít. Ebbe a csoportba főként táblás játékok tartoznak. Az egyes játékok MI-jének megvalósításának nehézsége jelentősen különbözik. Viszonylag egyszerű például a Tic-Tac-Toe, közepes szinten képvisel az Amőba. A sakkot a legnehezebbek között említhetjük.

3. Egyéb játékok. Ez a kategória elég tág, és különböző szempontok alapján további részletezése lehetséges. Másként kell viselkednie például egy FPS-nek, RPG-nek vagy RTS-nek.

Ha egy játékban működik MI, akkor annak általában két formája lehet:

1. Virtuális ellenfél. Itt a játékos nem látja az ellenfelét de érzékeli annak jelenlétét és hatását a játékmenetre. Példként említhetjük a sakkprogramokat, vagy RTS-eket.

2. Egységek Intelligenciája. Ebben az esetben az egységek is rendelkeznek valamiféle intelligenciával. Önállóan reagálnak például a játékos lépéseire, cselekedeteire. Ilyenek lehetnek az FPS-ek ellenséges egységei, vagy a már korábban említett útkereső algoritmusok, amikor egy RTS-ben a kiküldött egységek önállóan odatalálnak a célponthoz.

Egy programon belül a két MI típus vegyesen is megjelenhet. Egy RTS-ben például találhatunk egy fő ágat ami az általános stratégiát vezérli, de az egyes egységek (tankok, stb.) is rendelkeznek önálló intelligenciával és reagálnak a bekövetkező eseményekre.

2. Kétszemélyes, teljes információjú játékok

Ilyen játékkal szinte indenki találkozott már. Elég csak a már fentebb is említett sakkra gondolni. Nézzük most ezt a típust részletesebben.
Ha valaki egy ilyen játék alkotására vállalkozna, a következő paramétereket kell megadnia:

- a lehetséges játékállásokat
- a játékosok számát
- az egyes játékosok hogyan következnek lépni
- egy adott állásban milyen lehetséges lépései vannak a játékosoknak
- a játék folyamán milyen információval rendelkeznek a játékosok a játékmenetre vonatkozóan
- van-e a véletlennek szerepe a játékban és hol
- milyen állásban kezdődik, és mikor ér véget a játék
- az egyes játékosok mikor mennyit nyernek, vagy veszítenek

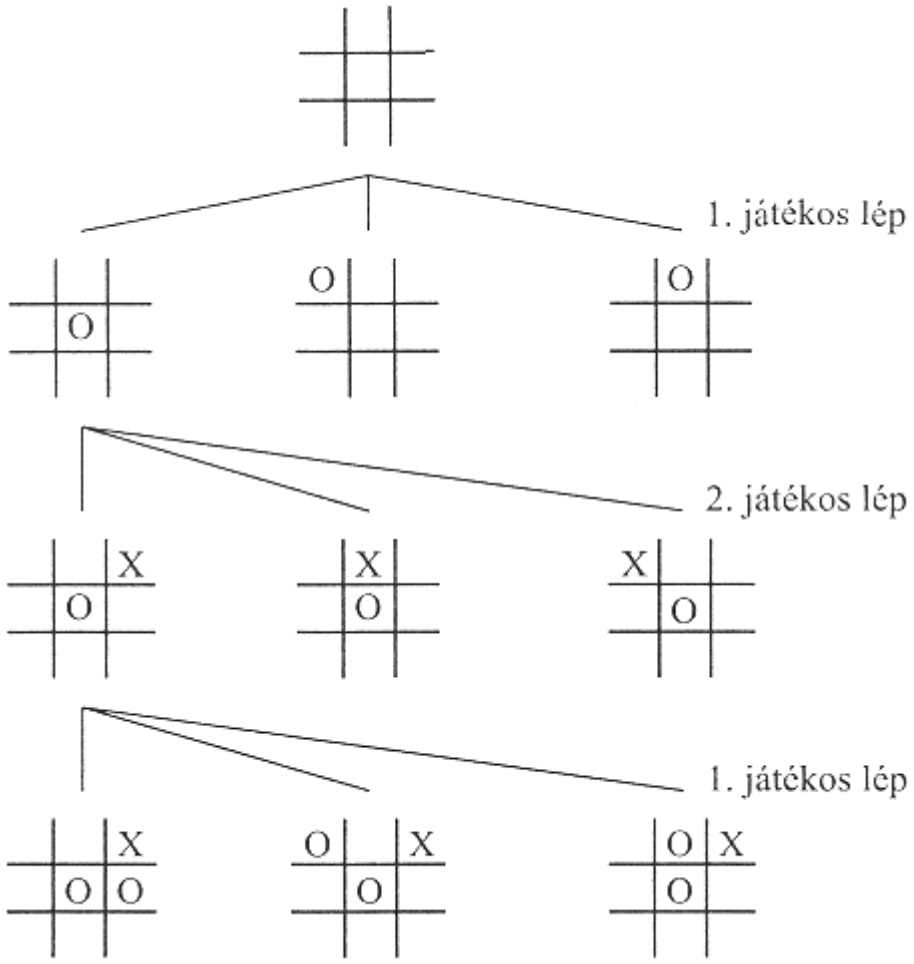
Az így megalkotott játékokat a következők szerint osztályozhatjuk:

- A játékosok száma szerint
- A játszma állásból állásba vivő lépések sorozata (diszkrét játék)
- A játékosoknak véges sok lehetséges lépése van az egyes játékállásokban, és a játszma véges sok lépés után befejeződik (véges játék)
- A játékosok a játék folyamán az összes információval rendelkeznek a játékra vonatkozóan (teljes információjú játék)
- A véletlennek nincs szerepe ajátékban (determinisztikus játék)
- A játékosok nyereségeinek és veszteségeinek összege 0 (zérusösszegű játék)

A továbbiakban játékon kétszemélyes, diszkrét, véges, teljes információjú, determinisztikus, zérus összegű játékot fogunk érteni.

2.1 A játékfa

Egy játék fája a játék összes lehetséges játszámját reprezentálja. A játék folyamán mindkét játékos tudja, hogy mik lehetséges lépések és, hogy az adott lépésnek mik a hatásai. A játék fájának van egy gyökéreleme, mely a játék kezdő állásának felel meg. A gyökérelem utáni csúcsok azokat az állásokat jelentik, amelyek a kezdő állásból egy lépésben elérhetők. A fa végén lévő csúcsok a terminális csúcsok. Ezek a játék végállapotát, a játszma végét jelentik, ahol az egyik játékos nyert, vagy döntetlen alakult ki. A páros szinteken lévő játékállásokban a kezdő játékos léphet, a páratlan szinteken lévőkben pedig az ellenfele. Egy állás annyi különböző csúcsban szerepel, ahány különböző módon eljuthatunk hozzá a játék során. Az utak, és a fa is véges hosszúságúak, hisz véges játékokról van szó. Ha a játékosok a kezdőállapotból valamelyik végállapotba érnek, akkor lejátszottak egy játszmát. A játszmákat a fában a gyökértől a terminális csúcsokba vezető utak szemléltetik.



2. A Tic-Tac-Toe fájának egy részlete.

2.2 Az állapottér-reprezentáció

Az állapottér-reprezentáció segítségével formálisan leírhatunk egy játékot.

Legyen a két játékos A és B, a játékállások halmaza H. A játék kezdőállása $a_0 \in$ H, a kezdő játékos $J_0 \in \{A, B\}$. Az egyes állapotokban megtehető lépések: $\{l \mid l: H \rightarrow H\}$. Az l lépés az a állapotban akkor tehető meg, ha teljesül l meglépésének előfeltétele az a állásra vonatkozóan. A játék az a állásban véget ér ha a végállás: végállás(a). A szabályok leírják,

hogy itt ki a nyerő játékos: nyer: {a | végállás(a)} $\rightarrow$ {jön$_a$, volt$_a$}, ahol jön$_a$ az a állásban soron következő játékos. E játék állapottér-reprezentációja az ($\mathcal{A}$, kezdő, $\mathcal{V}$, O) négyes, ahol:

- $\mathcal{A}$ = {(A, J) | a $\in$ H, J $\in$ {A, B}, J következik lépni}
- Kezdő = (A_0, J_0)
- $\mathcal{V}$ = {(a, J) | végállás(a), J nyer, ha nyer(a) = jön$_a$}
- O = {o_l | o_l(a, J) = (l(a), I), I, J $\in$ {A,B} , I != J}

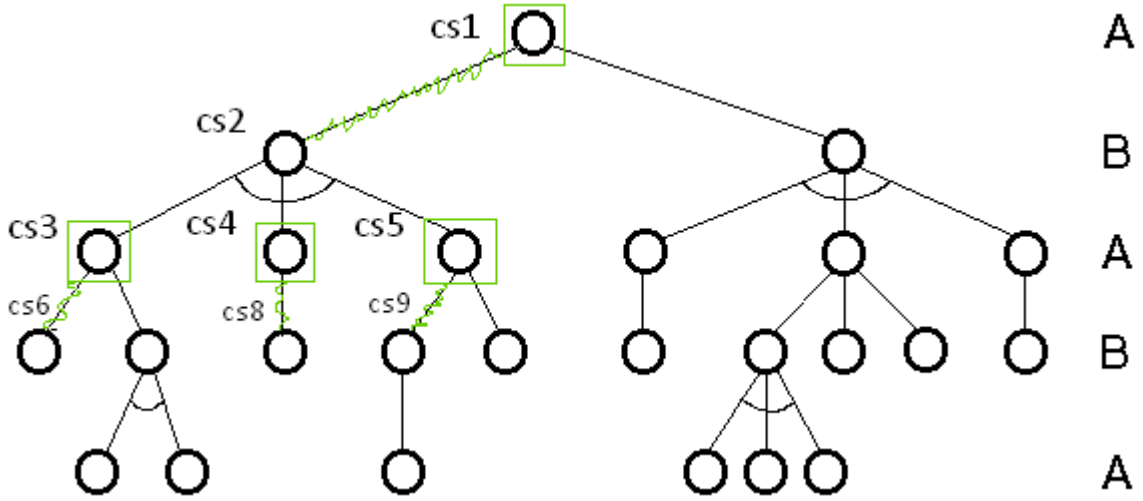
2.3 A stratégia

A J Játékos stratégiája egy olyan:

S_J : {(a, J) | (a, J) $\in$ $\mathcal{A}$} $\rightarrow$ O

döntési terv, amely J számára előírja, hogy azon játékállásokban, melyekben J következik lépni, a lehetséges lépései közül melyiket lépje meg.

A stratégia ábrázolása ÉS-VAGY fával történhet. J játékos szempontjából a J lépéseit szemléltető élek mindegyike egy élből álló hiperél lesz, mert adott állásban ahol J következik lépni, J itt eldöntheti, hogy egyik VAGY másik utat választja. Az ilyen állásokat szemléltető csúcsok a VAGY-csúcsok. J ellenfelének egy-egy állasban megtehető lépéseit szemléltető élek egy-egy élköteget alkotnak, ezek lesznek a hiperélek (ÉS-élek). Ezt is könnyű belátni, hiszen J nem tudhatja, hogy ellenfele az adott állásban mit fog lépni, így itt az összes lehetséges utat figyelembe kell vennie a stratégiája kialakításánál.



3. ábra. A kép az A játékos egy stratégiáját szemlélteti. Az A kezdi a játékot. Cs1 csomópontból cs2-be lép. A stratégiának az ellenfél összes lehetséges lépésére fel kell készülnie. B játékos cs2-ből akármelyik él mentén léphet, ezek az ÉS élek. A stratégia alapján az A játékos cs3-ból cs6-ba lép, cs4-ből cs8-ba, cs5-ből cs9-be. Nyerő stratégia esetén az összes végállapot ahová eljuthat a játszma, az A számára nyerő állás lenne.

Tegyük fel, hogy a J játékos az S_J stratégiájával játszik. Ekkor csak az S_J-t szemléltető hiperutat alkotó közönséges utak által szemléltetett játszmák játszhatók le.

Tegyük fel, hogy az A játékos az S_A, a B játékos pedig az S_B stratégiájával játszik. A két stratégia egyértelműen meghatározza a lejátszható játszmát.

A J játékos stratégiáját J nyerő stratégiájának nevezzük, ha (az ellenfelének stratégia-választásától függetlenül) minden a stratégia alkalmazása mellett lejátszható játszmában J nyer. Minden játék esetén az egyik játékos számára van nyerő stratégia.

3. Mesterséges intelligencia a kétszemélyes játékokban

1997. május 11-e kultúrtörténeti dátumnak is felfogható. A sakkvilágbajnok Garri Kaszparov leírhatatlan arckifejezéssel áll fel az asztaltól. Véget ért az IBM sakkszámítógépével folytatott párbaja. A gép 3,5-2,5 arányban legyőzte a regnáló világbajnokot.

4. ábra. Videofelvétel Garri Kaszparov orosz sakkvilágbajnokról, amint a 19. lépésben feladja a Deep Blue számítógép elleni hatmenetes párviadal döntő mérkőzését.

3.1 Deep Blue

Felmerül a kérdés, hogyan volt képes a számítógép ezt a bravúrt végrehajtani? A gépet egy Feng-hsiung Hsu nevű matematikus kezdte fejleszteni 1988-ban a Carnegie Mellon

egyetemen. Már 1989-ben megvívott Kaszparovval, aki könnyedén legyőzte. Egy évvel később Feng az IBM-hez igazolt, ahol Deep Blue néven folytatta a munkát.

A Deep Blue alap algoritmusa a minimax algoritmus, mely lényegében semmit sem változott az elmúlt évtizedekben. A következő részben a részletes működését is áttekintjük. Ami változott: a végrehajtás sebessége. A Deep Blue tulajdonképpen nem túl elegáns, a nyers erőre épít. Másodpercenként akár 50 milliárd állást tud kielemezni. Összehasonlításképpen: egy „mezei" Pentium 200-as processzoros PC ennek a tízezred részét sem. Persze az igazsághoz hozzátartozik, hogy a Deep Blue egy célszámítógép, ahol a hardvert is direkt sakkjátékhoz építették fel. Egy sakknagymester pedig, a pszichológia tesztek tanulsága szerint „alig" 500-1000 állást tud átlátni 3 perc alatt. Ezen adatok fényében az a kérdés adódik, hogy ekkora hátrányban hogyan van egyáltalán esélye az embernek, hogy-hogy nem verte agyon a gép Kaszparovot?

Ezek szerint a gép valahol nagy hátrányban kell legyen, ami lényegében ellensúlyozza hatalmas teljesítménybeli fölényét. Ezt a kognitív pszichológusok az úgynevezett kognitív sémák mennyiségében találták meg. A kognitív sémák gondolkodásunk alapvető egységei. Nem azonosak a memóriánkban tárolt adatokkal, hanem inkább azok szerveződései határozzák meg őket. Kognitív séma lehet például egy sakkozó agyában az, hogy a futó 3 egységet ér, a bástya pedig 5-öt, de lehet az is, hogy a szicíliai védelemben világos általában a királyszárnyon, sötét pedig a vezérszárnyon támad, vagy valami olyasmi is, hogy szárnytámadás előtt célszerű lezárni a centrumot. Az újabb és újabb ismeretek, tények a meglevő kognitív sémák mentén szerveződnek agyunkban úgy, hogy közben némileg meg is változtatják azokat. Kognitív sémáink határozzák meg azt, hogy milyen ismereteket vagyunk képesek egy adott pillanatban egyáltalán elsajátítani és alkalmazni. A kognitív sémák alapozzák meg az emberi intuíció működési mechanizmusát.

Pszichológiai tesztek kimutatták, hogy egy sakknagymester agyában körülbelül 100 ezer séma van. Nem teljesen helytálló, ha azt mondjuk, hogy ennyit ismer, mert ezek a sémák igen egyediek, és folyamatosan változnak is. Kérdés, hogy hány kognitív sémát ismer a Deep Blue? Mennyi intuíciója van?

Erre a kérdésre egyrészt azt válaszolhatjuk, hogy egyetlenegyet. A minimax algoritmus ugyanis arra épül, hogy adva van egy kiértékelő függvény, mely minden egyes állásra megmondja, hogy az mennyire előnyös a játékos szempontjából. Ezt az előnyösséget maximalizálja az algoritmus úgy, hogy az a fa általa áttekintett részén (például tíz lépéspárra

előre az összes lehetséges variációt figyelembe véve) az ellenfél lehető legjobb játéka esetén is a számára legjobb végkifejletre vezessen. Mivel a minimax algoritmus egy kiértékelő függvénnyel dolgozik (a Deep Blue is), azt mondhatjuk, hogy a gép egyetlen kognitív sémával rendelkezik csupán.

Ezzel szemben felmerülhet az a kifogás, hogy mi van ha a kiértékelő függvény olyan összetett, hogy Kaszparovéhoz hasonló mennyiségű sémát is magában foglal? Nézzük hát meg, hogyan működik a sakkprogramok kiértékelő függvénye! Azt látjuk, hogy meglepően kevés dolgot vesznek figyelembe. Nagyjából tucatnyi egyszerű, jól ismert jellegzetességet figyelnek (például a kettős gyalogot hátrányosnak tekintik, a nyílt vonalakat előnyösnek stb.). Ha ezeket a szempontokat mind külön sémának tekintjük, akkor sem tud a gép többet magáról a sakkról, mint ezt a néhány kognitív sémányit, szemben Kaszparov százezernyijével.

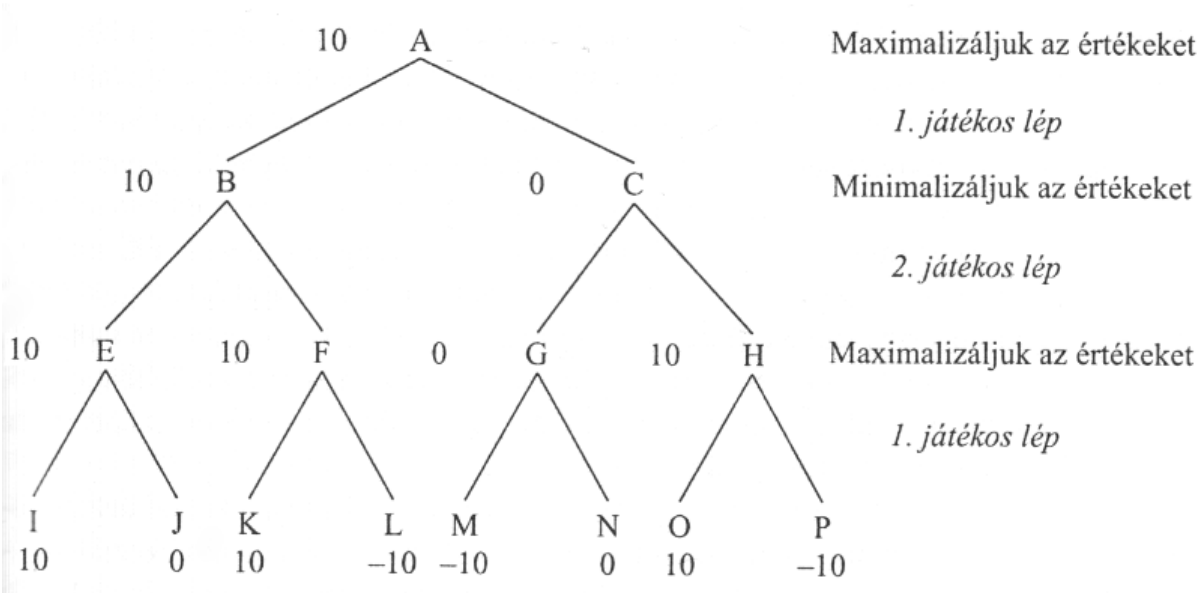
Mégis, mikor akár egy nagymester, akár egy amatőr sakkozó a géppel játszik, egyfolytában terveket, ötleteket, elképzeléseket észlel a gép részéről, holott a minimax algoritmus ismeretében elmondhatjuk, hogy ilyeneknek a programban nyoma sincs. Eszerint ezeket csak az ember vetíti bele a gép játékába. Mivel az ember a kognitív sémák segítségével észleli a világot, nehéz elképzelni, hogy ellenfelünk a gép semmi ilyesmit nem használ. Amennyire pontosan tudhatjuk, hogyan működnek a sakkprogramok kiértékelő függvényei, annyira keveset tudunk arról, hogyan működnek az ember kognitív sémái. Ahhoz mindenképpen nagyon keveset, hogy egyáltalán megpróbálkozhassunk azzal, hogy ilyesfajta gondolkodási egységeket programozzunk a számítógépekbe. Kaszparov és a Deep Blue párbajában lényegében két különböző minőség csapott össze. A nyers erő, vagyis a nagy mennyiségű állás értékelése, minimális kognitív séma segítségével, és a kevés állás áttekintése sok séma segítségével. A nagy mennyiségű állás konkrét mennyisége mostanra érte el azt a szintet, ahol a százezernyi kognitív séma fölé tud kerekedni. Kérdés, hogy növelheti-e valaki a kognitív sémáinak vagy az általa áttekinteni tudott állásoknak a mennyiségét. Sok pszichológiai vizsgálat támasztja alá azt, hogy ez nem lehetséges: nagyjából ennyi az emberi teljesítőképesség felső határa. A számítógép sebessége viszont elvileg tovább is növelhető, így hosszú távon az embernek egyre kevesebb esélye lesz a gépek ellen a kétszemélyes játékokban.

5. ábra. A Deep Blue szuperszámítógép, mely legyőzte Garri Kaszparov sakk-világbajnokot.

3.2 A minimax algoritmus

Nézzük tehát, hogyan működik a kétszemélyes játékokban általánosan használt minimax algoritmus.

Az alábbi teljes játékfában a következő módon szemléltethetjük az elvet. Legyen a minimax által támogatott játékos az 1. játékos. Ennek győzelmét jelöljük 10-zel, vereségét -10-zel, a döntetlent 0-val. A levélelemekhez hozzárendeljük ezeket az értékeket. Ezek után felfelé haladva meghatározzuk a fa feljebb lévő csomópontjaihoz tartozó értéket. Az E, F, G és H csomópontokban az első játékos következik. Ő maximalizálni próbálja a pontszámát. Az E, F és H állásból győztes állásba kerülhet, így ezek a csomópontok is 10-es értéket kapnak. A G állásból legjobb esetben döntetlent érhet el, így a G-hez tartozó maximum érték 0. Most nézzük a B és C csomópontot, ahol a 2. játékos következik. Ő az ellenfele ellen dolgozik, tehát minimalizálni igyekszik az 1. játékos pontszámát. A B csomópontban a 2. játékos akármit lép is, az 1. játékos nyer. Az érték itt az E és F csomópontokhoz tartozó értékek minimuma, ami 10. A C csomópont értéke pedig a G és H minimuma, vagyis 0 lesz, mert a 2. játékos G állásba is léphet, így az 1. játékost olyan helyzetbe kényszeríti, amelyből az már nem képes nyerni.



6. ábra. Az A, E, F, G és H csomópontokat maximalizáló, a B és C csomópontokat pedig minimalizáló csomópontoknak nevezzük.

Az előbb egyszerű dolgunk volt, mert teljes játékfával dolgoztunk. Azonban sok játék esetében (mint a sakk is), a teljes játékfát lehetetlen felépíteni a lehetséges állások hatalmas száma miatt. A Deep Blue például maximum 12 lépésre előre építette fel a fát. Ilyenkor nő meg a jelentősége a kiértékelő függvénynek, mivel a részfában a levélelemek már nem végállások lesznek. Ezeket az állásokat csak becsülni lehet, hogy milyen jó az adott játékos számára. Szükség van tehát a levélelemekhez más tetszőleges egész értékeket is hozzárendelni -10 és 10 között, mely értékek kifejezik az adott állás jóságát a támogatott játékos szempontjából. A kiértékelő függvény nélkül a minimax algoritmusnak sincs értelme. Hiányában a gép nem „gondolkozna”, csupán véletlenszerűen lépne. A játék nehézsége tulajdonképpen a részfa nagyságától, és attól függ, hogy a kiértékelő függvény mennyire jól tudja megbecsülni az állás jóságát. Például a játék nehézségi szintjének beállításakor, sokszor azt adjuk meg, hogy milyen mélységben építse fel a program a részfát.

Az algoritmus fő lépései:

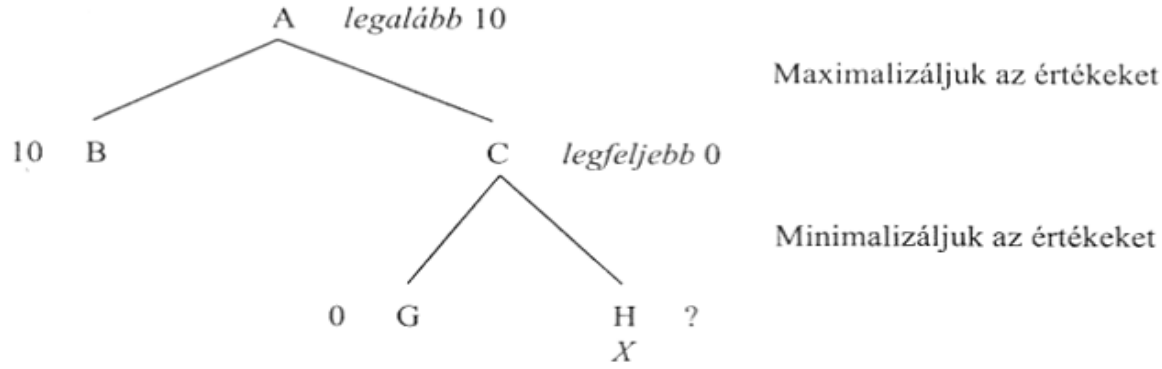
1. A játékfa állapotot szemléltető csúcsából kiinduló részének előállítása adott mélységig.
2. A részfa leveleiben található állások jóságainak becslése heurisztika segítségével.
3. Szintenként csökkenő sorrendben a részfa nem levél csúcsi jóságainak kiszámítása.

A minimax algoritmus pszeudó kódja:

```
function minimax(csúcs, mélység)
     if a csúcs levél or mélység = 0 then
          return a csúcs heurisztikus értéke
     else if játékos = A then
          max := -∞
          for az összes lehetséges lépésre
               új_állapot := előállítja az új állapotot
               v := minimax(új_állapot, mélység-1)
               if v >max then
                    max := v
               end if
          end for
          return max
     else
          min := +∞
          for az összes lehetséges lépésre
               új_állapot := előállítja az új álapotot
               v := minimax(új_állapot, mélység-1)
               if v < min then
                    min := v
               end if
          end for
          return min
     end if
end function
```

3.3 Az alfa-béta vágás

Az alfa-béta vágás lényegében a minimax algoritmus egy optimalizálása. Egy ügyes trükk, melynek segítségével elérhetjük, hogy ne kelljen végigvizsgálni a felépített fa összes csomópontját, így megnövelve a minimax algoritmus hatékonyságát.



7. ábra. A C csomópont β értéke G alapján nulla, és minimalizáló csomópont lévén ennél nagyobb már nem is lehet. Az A α értéke B alapján már 10, és mivel A maximalizáló csomópont, így C irányába biztosan nem történik lépés. Emiatt a H csomópont, és a H alatti rész vizsgálata elhagyható.

Vizsgáljuk a 7. ábrából kiindulva ezt a technikát. Tegyük fel, hogy B csomópont értékét már ismerjük. Mivel az 1.(támogatott) játékos maximalizálni igyekszik pontszámát, így már C vizsgálata nélkül is tudjuk, hogy legalább 10 pontja biztosan lesz. Tehát A értéke minimum 10. Tegyük fel, hogy G értéke szintén megvan. A 2. játékos minimalizálni akarja az 1. játékos pontjait, a C értéke legfeljebb 0 lehet. Ezek alapján láthatjuk, hogy semmi értelme meghatározni a H csomópont értékét, mivel semmi esély sincs arra, hogy befolyásolja az A értékét. Tegyük fel, hogy H értéke kevesebb lenne, mint 0. Ekkor a C nem változik, így nyilván A sem. Tehát mivel H értéke semmilyen változást nem okoz, ezért a fa H csomópontja és a H alatti része teljesen elhagyható.

Tudjuk, hogy egy maximalizáló csomópont értéke legalább az eddig megvizsgált leszármazott csomópontok értékei közül a legnagyobb lesz (ezt az értéket jelöljük α-val). A minimalizáló csomópont legfeljebb csak az eddig megvizsgált leszármazottak értékei közül a legkisebb lehet (ezt β-val jelöljük). Ha β értéke egy minimalizáló csomópontban kisebb, mint az α értéke a szülő csomópontban, akkor az adott csomópontra minden további számítás elhagyható. A fa további része levágható. Ha pedig egy maximalizáló csomópont α értéke

nagyobb, mint a szülő β értéke, akkor az előbbi esethez hasonlóan, a csomópontot felesleges tovább vizsgálni, tehát a fa további része itt is levágható.

A gyakorlatban általában adott idő alatt kétszer olyan mély fát tudunk ezzel a módszerrel megvizsgálni, mint a mezei minimax algoritmussal.

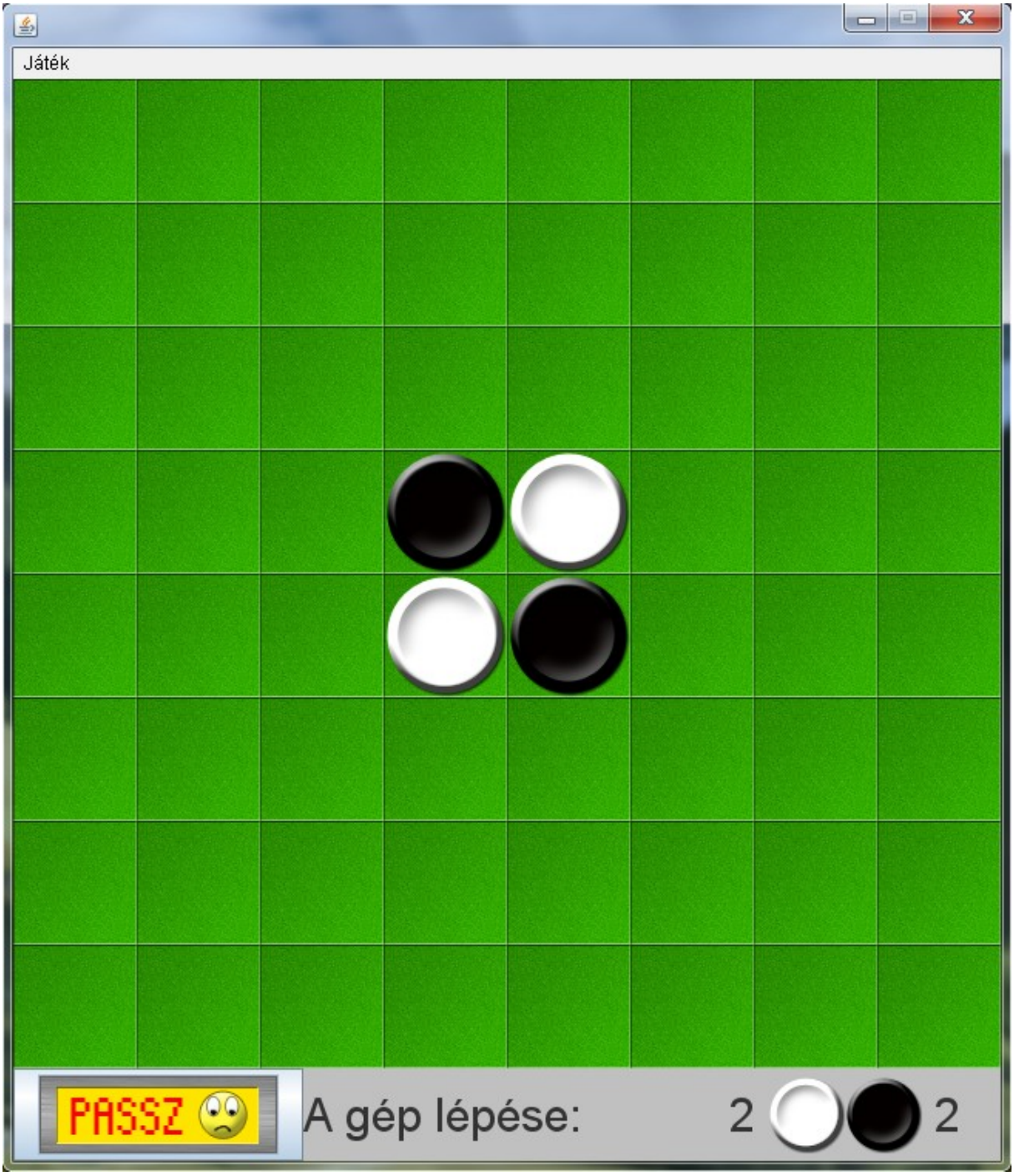
Az alfa-béta vágás pszeudó kódja:

```
function alfa-béta-minimax(csúcs, mélység, alfa, béta)
     if a csúcs levél or mélység = 0 then
          return a csúcs heurisztikus értéke
     else if játékos = A then
          for az összes lehetséges lépésre
               if alfa >= béta then break
               új_állapot := előállítja az új állapotot
               v := alfa-béta-minimax (új_állapot, mélység-1,
alfa, béta)
               if v > alfa then
                    alfa := v
               end if
          end for
          return alfa > beta ? alfa : beta
     else
          for az összes lehetséges lépésre
               if alfa >= béta then break
               új_állapot := előállítja az új álapotot
               v := alfa-béta-minimax (új_állapot, mélység-1,
alfa, béta)
               if v < béta then
                    béta := v
               end if
          end for
          return alfa < beta ? alfa : beta
     end if
end function
```

4. A Reversi megvalósítása

Most egy konkrét játék, a Reversi megvalósítását fogjuk megnézni. A játékot két játékos játssza 8x8-as négyzetrácsos táblán. A táblán fekete illetve fehér korongokkal folyik a küzdelem. Kezdéskor a tábla közepén X alakban két-két korong van elhelyezve mindkét színből. A játékosok felváltva tesznek le újabb korongokat. A játékosok célja, hogy a játék végére minél több saját színű korongjuk legyen a táblán.
A játék lényege, hogy a lépés befejezéseként az ellenfél ollóba fogott, azaz két oldalról (vízszintesen, függőlegesen vagy átlósan) közrezárt bábuit (egy lépésben akár több irányban is) a saját színünkre cseréljük. Mindkét játékosnak, minden lépésben ütnie kell. Ha egy állásban nincs olyan lépés, amivel a játékos ollóba tudna fogni legalább egy ellenséges korongot, passzolnia kell és újra ellenfele lép.
A játék akkor ér véget, ha a tábla megtelik, vagy ha mindkét játékos passzol. A játék győztese az a játékos, akinek a játék végén több korongja van a táblán. A játék döntetlen, ha mindkét játékosnak ugyanannyi korongja van a játék végén.

A játékot eredetileg úgy játszották, hogy ha egy új korong táblára kerülésekor egyszerre több irányban is beékelődnek ellenséges korongok, akkor azt a közbezárt részt kell megfordítani, amelyben több korong rabolható el, valamint nem volt kötelező minden lépésben ütni.
Így, már ritkán játsszák, többnyire a fent leírt szabályokat alkalmazzák és erre a fajtára terjed el a játék japán "újrafelfedezésétől"- az "OTHELLO" elnevezés.
Ma már, szinte csak az "OTHELLO-REVERSI"-t játssza a világ és mi is ezt vizsgáljuk most meg.



8. ábra. A program főablaka a kezdőállással.

4.1 A kiértékelő függvény

A programban egy egyszerű kiértékelő függvény dolgozik. A fehér és fekete korongok arányából határozza meg az állás jóságát az A játékos szempontjából. A programban az A-hoz van rendelve a fekete, a B-hez a fehér korong:

```
public int minimaxHasznossag(){
      if(nyertB()) return -100;
      if(nyertA()) return 100;
      return getSotetNum() - getVilagosNum();
}
```

Először is vizsgálni kell, hogy végállapotról van-e szó, hiszen ez az A szempontjából végletesen jó, vagy rossz állást jelent. Ha B nyer, a visszaadható legkisebb értéket adja itt vissza a függvény. Mivel 8x8-as tábláról van szó, a korongok különbsége -64 és 64 között lehet. A -100 tehát megfelelően kicsi érték ahhoz, hogy A vereségét jelezze. Ha A nyer az adott állásban az a legjobb neki, ilyenkor 100-zal tér vissza a függvény. Ha pedig nem végállapotról van szó, akkor a táblán lévő korongok különbségével térünk vissza.

4.2 Állapottér reprezentáció

A harmadik fejezetben láthattuk mi az állapottér reprezentáció. Most megnézzük, hogy a Reversi esetében hogyan néz ki egy lehetséges megvalósítása.

A játék világának állapotait elem 65-ösök írják le:

$H_{1,1} = H_{1,2} = \ldots = H_{8,8} = \{0,1,2\}$

Ahol:

$h_{i,j} = 0$ ha a pozíció üres

$h_{i,j} = 1$ ha a pozíción fehér korong van

$h_{i,j} = 2$ ha a pozíción fekete korong van

$h_{i,j} \in H$

valamint: J={A, B}

Ahol:

j = A ha az A játékos következik lépni

j = B ha a B játékos következik lépni

Az állapotok halmaza: $H_{1,1} \times H_{1,2} \times \ldots \times H_{8,8} \times J$

Látható, hogy a $H_{1,1} \times H_{1,2} \times \ldots \times H_{8,8} \times J$ Descartes szorzat által meghatározott halmazban olyan állapotok is előfordulnak, amelyek a valóságban nem állhatnak elő. Például, ha $h_{1,1} = 1$ és $h_{1,2}$, … ,$h_{8,8}$ mind 0, ez azt jelenti, hogy a táblán mindössze egy fehér korong van, az összes többi mező pedig üres. A játék szabályaiból viszont látszik, hogy játék közben ilyen állás nem tud előállni. Az állapotok halmazára tehát feltételeket kell megfogalmaznunk, hogy kizárjuk a nem valós állapotokat.

A $h = (h_{1,1}, h_{1,2}, \ldots, h_{8,8}, j) \in H_{1,1} \times H_{1,2} \times \ldots \times H_{8,8} \times J$ elemhatvanötös a probléma valódi állapota, ha teljesíti a következő kényszerfeltételek diszjunkcióját:

- A táblán legalább két fehér korong van:

$$F_1 \equiv \sum_{x=1}^{8} \sum_{y=1}^{8} \text{isKorong}(x, y, 1) >= 2$$

isKorong(x, y, 1) = 1 ha $h_{x,y} = 1$

isKorong(x, y, 1) = 0 ha $h_{x,y} \neq 1$

- A táblán legalább két fekete korong van:

$F_2 \equiv \sum_{x=1}^{8} \sum_{y=1}^{8}$ isKorong (x, y, 2) >= 2

isKorong (x, y, 2) = 1 ha $h_{x,y} = 2$

isKorong (x, y, 2) = 0 ha $h_{x,y} \neq 2$

Állapottér:

Az állapottér a probléma valódi állapotainak halmaza:

$\mathcal{A} = \{h \mid h \in H_{1,1} \times H_{1,2} \times \ldots \times H_{8,8} \times J \wedge \text{kényszerfeltétel}(h)\}$

kényszerfeltétel(h) $\equiv F_1 \vee F_2$

Ezekkel a feltételekkel szűkítve az állapotok halmazát, az állapottérben már csak a valóságban is előállítható játékállások lesznek.

Kezdőállapot:

$$k = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \; k \in \mathcal{A}$$

A kényszerfeltételek alapján látható, hogy a kezdőállapot is valódi állapot, bár a kezdés után többé már nem állhat elő ilyen helyzet, hogy a mindkét korongból csak kettő van a táblán.

Célállapotok:

A célállapotok az eddigiek alapján a teljes játékfa levélelemei lesznek, ahol vagy A vagy B játékos nyer, vagy döntetlen alakul ki.

A $h = (h_{1,1}, h_{1,2}, \ldots, h_{8,8}, j) \in \mathcal{A}$ valódi állapot a probléma valódi végállapota, ha teljesíti a következő célfeltételek diszjunkcióját:

- A táblán csak fehér korong van:

$$F_1 \equiv \sum_{x=1}^{8} \sum_{y=1}^{8} \text{isKorong}(x, y, 2) = 0$$

- A táblán csak fekete korong van:

$$F_2 \equiv \sum_{x=1}^{8} \sum_{y=1}^{8} \text{isKorong}(x, y, 1) = 0$$

- A táblán nincs üres mező:

$$F_3 \equiv \sum_{x=1}^{8} \sum_{y=1}^{8} \text{isKorong}(x, y, 0) = 0$$

- Nem lehet korongot lerakni a táblára:

$$F_4 \equiv \sum_{x=1}^{8} \sum_{y=1}^{8} (\text{lerakFehéret}(x, y) + \text{lerakFeketét}(x, y)) = 0$$

lerakhatFehéret(x, y) = 1, ha az x,y pozícióba le lehet rakni fehéret

lerakhatFehéret(x, y) = 0, ha az x,y pozícióba nem lehet fehéret lerakni

lerakhatFeketét (x, y) = 1, ha az x,y pozícióba sikeres le lehet rakni feketét

lerakhatFeketét (x, y) = 0, ha az x,y pozícióba nem lehet feketét lerakni

Célállapotok halmaza:

$$\mathcal{C} = \{h \mid h \in H_{1,1} \times H_{1,2} \times \ldots \times H_{8,8} \times J \wedge \text{célfeltétel}(h)\}$$

$$\text{célfeltétel}(h) \equiv F_1 \vee F_2 \vee F_3 \vee F_4$$

Az „A” játékos nyer ha a feketével van, és F_2 célfeltétel teljesül, vagy F_3 teljesül úgy, hogy fekete korongból van több a táblán, vagy F_4 teljesül úgy, hogy fekete korongból van több a táblán:

$$\text{nyert(A)} = F_2 \vee (F_3 \wedge (\sum_{x=1}^{8}\sum_{y=1}^{8} \text{isKorong}(x, y, 2) > \sum_{x=1}^{8}\sum_{y=1}^{8} \text{isKorong}(x, y, 1))) \vee (F_4 \wedge (\sum_{x=1}^{8}\sum_{y=1}^{8} \text{isKorong}(x, y, 2) > \sum_{x=1}^{8}\sum_{y=1}^{8} \text{isKorong}(x, y, 1)))$$

„B” játékos nyer, ha a fehérrel van, és F_1 célfeltétel teljesül, vagy F_3 teljesül úgy, hogy fehér korongból van több a táblán, vagy F_4 teljesül úgy, hogy fehér korongból van több a táblán:

$$\text{nyert(B)} = F_1 \vee (F_3 \wedge (\sum_{x=1}^{8}\sum_{y=1}^{8} \text{isKorong}(x, y, 1) > \sum_{x=1}^{8}\sum_{y=1}^{8} \text{isKorong}(x, y, 2))) \vee (F_4 \wedge (\sum_{x=1}^{8}\sum_{y=1}^{8} \text{isKorong}(x, y, 1) > \sum_{x=1}^{8}\sum_{y=1}^{8} \text{isKorong}(x, y, 2)))$$

Döntetlen, ha F_3 vagy F_4 teljesül úgy, hogy egyforma számú fehér és fekete korong van a táblán:

$$(F_3 \wedge (\sum_{x=1}^{8}\sum_{y=1}^{8} \text{isKorong}(x, y, 2) = \sum_{x=1}^{8}\sum_{y=1}^{8} \text{isKorong}(x, y, 1))) \vee (F_4 \wedge (\sum_{x=1}^{8}\sum_{y=1}^{8} \text{isKorong}(x, y, 2) > \sum_{x=1}^{8}\sum_{y=1}^{8} \text{isKorong}(x, y, 1)))$$

Operátorok:

Az operátorok halmaza tartalmazza a játék során megtehető összes lehetséges lépést.

O = {lerakFeketét(x, y), lerakFehéret(x, y)}, x = 1, … , 8 y = 1, … 8

Kétféle lépés lehetséges. Vagy fekete vagy fehér korongot rakhatunk le a táblára az x, y koordinátájú mezőre. Nyilvánvaló azonban, hogy nem akármelyik mezőre rakhatunk

korongot, tehát az operátorok halmazát is kényszerfeltételekkel kell szűkítenünk. Ezek a játék szabályainak felelnek meg. Hiányában a program játék közben szabálytalan lépéseket is meglépne.

A **lerakFeketét(x, y)** operátor alkalmazható egy $h = (h_{1,1}, h_{1,2}, \ldots, h_{8,8}, j) \in \mathcal{A}$ állapotra, ha teljesül a következő alkalmazási előfeltételek konjunkciója:

- az A játékos következik lépni:

$F_1 \equiv j = A$

- az x, y koordinátájú pozíció üres:

$F_2 \equiv h_{x,y} = 0$

- Ha x, y pozícióra rak korongot, akkor egy már a táblán lévő fekete koronggal közrezár legalább egy fehér korongot vízszintesen, függőlegesen vagy átlósan:

$Z_1 \equiv (h_{x,y-i} = 2) \wedge (\sum_{z=1}^{i-1}$ isKorong (x, y-z, 1) = i-1) (közrezár függőlegesen felfelé)

$Z_2 \equiv (h_{x,y+i} = 2) \wedge (\sum_{z=1}^{i-1}$ isKorong (x, y+z, 1) = i-1) (közrezár függőlegesen lefelé)

$Z_3 \equiv (h_{x+i,y} = 2) \wedge (\sum_{z=1}^{i-1}$ isKorong (x+z y, 1) = i-1) (közrezár vízszintesen jobbra)

$Z_4 \equiv (h_{x-i,y} = 2) \wedge (\sum_{z=1}^{i-1}$ isKorong (x-z, y, 1) = i-1) (közrezár vízszintesen balra)

$Z_5 \equiv (h_{x+i,y-i} = 2) \wedge (\sum_{z=1}^{i-1}$ isKorong (x+z, y-z, 1) = i-1) (közrezár átlósan jobbra-fel)

$Z_6 \equiv (h_{x+i,y+i} = 2) \wedge (\sum_{z=1}^{i-1}$ isKorong (x+z, y+z, 1) = i-1) (közrezár átlósan jobbra-le)

$Z_7 \equiv (h_{x-i,y+i} = 2) \wedge (\sum_{z=1}^{i-1}$ isKorong (x-z, y+z, 1) = i-1) (közrezár átlósan balra-le)

$Z_8 \equiv (h_{x-i,y-i} = 2) \wedge (\sum_{z=1}^{i-1}$ isKorong (x-z, y-z, 1) = i-1) (közrezár átlósan balra-fel)

i = 2, … ,7

$F_3 \equiv Z_1 \vee Z_2 \vee Z_3 \vee Z_4 \vee Z_5 \vee Z_6 \vee Z_7 \vee Z_8$

A **lerakFehéret(x, y)** operátor alkalmazható egy $h = (h_{1,1}, h_{1,2}, \ldots, h_{8,8}, j) \in \mathcal{A}$ állapotra, ha teljesül a következő alkalmazási előfeltételek konjunkciója:

- az B játékos következik lépni:

$F_1 \equiv j = B$

- az x, y koordinátájú pozíció üres:

$F_2 \equiv h_{x,y} = 0$

- Ha x, y pozícióra rak korongot, akkor egy már a táblán lévő fehér koronggal közrezár legalább egy fekete korongot vízszintesen, függőlegesen vagy átlósan:

$Z_1 \equiv (h_{x,y-i} = 1) \wedge (\sum_{z=1}^{i-1}$ isKorong (x, y-z, 2) = i-1) (közrezár függőlegesen felfelé)

$Z_2 \equiv (h_{x,y+i} = 1) \wedge (\sum_{z=1}^{i-1}$ isKorong (x, y+z, 2) = i-1) (közrezár függőlegesen lefelé)

$Z_3 \equiv (h_{x+i,y} = 1) \wedge (\sum_{z=1}^{i-1}$ isKorong (x+z, y, 2) = i-1) (közrezár vízszintesen jobbra)

$Z_4 \equiv (h_{x-i,y} = 1) \wedge (\sum_{z=1}^{i-1}$ isKorong (x-z, y, 2) = i-1) (közrezár vízszintesen balra)

$Z_5 \equiv (h_{x+i,y-i} = 1) \wedge (\sum_{z=1}^{i-1}$ isKorong (x+z, y-z, 2) = i-1) (közrezár átlósan jobbra-fel)

$Z_6 \equiv (h_{x+i,y+i} = 1) \wedge (\sum_{z=1}^{i-1}$ isKorong (x+z, y+z, 2) = i-1) (közrezár átlósan jobbra-le)

$Z_7 \equiv (h_{x-i,y+i} = 1) \wedge (\sum_{z=1}^{i-1}$ isKorong (x-z, y+z, 2) = i-1) (közrezár átlósan balra-le)

$Z_8 \equiv (h_{x-i,y-i} = 1) \wedge (\sum_{z=1}^{i-1}$ isKorong (x-z, y-z, 2) = i-1) (közrezár átlósan balra-fel)

$i = 2, \ldots, 7$

$F_3 \equiv Z_1 \vee Z_2 \vee Z_3 \vee Z_4 \vee Z_5 \vee Z_6 \vee Z_7 \vee Z_8$

Ezekkel a feltételekkel megszorítva az operátorokat most már a program csak szabályos lépéseket tud lépni a játék folyamán, és biztosítva van, hogy az operátor hatásaként csakis valódi állapot jön létre.

Ezzel megadtuk a ($\mathcal{A}$, kezdő, $\mathcal{V}$, O) négyest, a Reversi állapottér reprezentációját.

Összefoglalás

A mesterséges intelligenciát évtizedek óta kutatják. A kezdeti törekvések általános célú intelligens gépek megvalósítására irányultak. Lényegében emberszerű gép kifejlesztése volt a cél, mely bármiről beszélgetni képes az emberrel, és tulajdonképpen helyettesítheti az embert sok olyan területen melyhez intelligencia, értelmes viselkedés szükséges. Hamar felmerült azonban a kombinatorikus robbanás problémája, mely lehetetlenné teszi az ilyen célok megvalósítását a jelenlegi hardvereszközök kapacitása mellett. Az évtizedek folyamán fokozatosan távolodott ezektől a törekvésektől a tudomány, a minél szűkebb részterületeken való egyre hatékonyabb alkalmazások felé. Olyan kutatási területek alakultak így ki, mint a természetes nyelv feldolgozás, a kép, hang, beszédfelismerés, tudásalapú rendszerek, robotika, és a játékok. Ebben a dolgozatban a játékokkal, azon belül is a kétszemélyes, teljes információjú játékokkal, azok megvalósításával foglalkoztunk részletesebben. Láthattuk, hogy ezen a területen, mint a mesterséges intelligencia egy részterületén komoly fejlesztések folytak mind a hardver, mind a szoftver esetében.

A kétszemélyes játékok legfontosabb tulajdonságai, hogy végesek, determinisztikusak, diszkrétek és zérus összegűek. Minden kétszemélyes játék esetében az egyik játékos számára létezik nyerő stratégia, tehát ha minden állásban ahol az adott játékos következik lépni, e stratégia szerint lép, akkor biztos, hogy nyer, az ellenfele bármilyen lépései mellett.

Egy játékban a játszmákat a játékfával lehet reprezentálni. A játékfában a csomópontok a lehetséges játékállásokat jelölik, az élek pedig a lépéseket. Egy csomópontból annyi él indul ki, ahány szabályos lépés végrehajtható az adott állásban. A gyökérelem a kezdőállás, ahol a kezdő játékos meglépi a játék első lépését. A páratlan szinteken lévő állásokban tehát a kezdő játékos következik lépni, a páros szinteken pedig az ellenfele. A játékfa az összes lehetséges állást és az összes szabályos lépést tartalmazza, így az összes lejátszható játszmát is magában foglalja a gyökérelemből a levelekbe vezető utak által. Ebből következően lényeges, hogy a gyakorlati megvalósításnál a számítógép nem tudja a teljes játékfát felépíteni, mert egy átlagos játék esetében a variációk hatalmas száma meghaladja a feldolgozhatóság határát a jelenlegi számítási kapacitás mellett. Ilyenkor általában csak egy adott mélységig épül fel a fa és kerülnek benne kiértékelésre az állások.

A játékfa segítségével a programnak valahogyan le kell vezényelnie a játszmát. Ezt leggyakrabban a minimax nevű algoritmus segítségével teszi. Ennek lényege, hogy a levélelemeket egy egész számmal megjelöli, hogy azok számára nyerő vagy vesztes, esetleg döntetlen állásokat tartalmaznak-e. Jellemzően a nyerő állásokhoz rendeli a nagyobb értéket. A program a játékfát mindig felépíti az adott állásból, mikor ő következik lépni. Majd a fában visszafelé haladva, minden csomóponthoz hozzárendeli valamelyik értéket az előző csomópontok alapján. A páratlan szinteken lévő csomópontokhoz a gyermekek értékei közül a nagyobbat, a páros szinteken levőkhöz, pedig a kisebbet adja hozzá. Miután végzett a fával, a kiinduló csomópontból már csak lépnie kell a nagyobb értékű gyerek felé, és ez számára egy lépés lesz az egyik győztes végállás felé.

Viszont, ahogy fentebb említettük, a program általában csak adott mélységű részfát képes felépíteni, annak nagysága miatt. Ilyenkor válik fontossá a kiértékelő függvény, melyet a minimax algoritmus használ. Mivel ilyenkor már a levélelemek nem végállások lesznek, meg kell becsülnie a programnak a jóságukat a saját szempontjából. Vagyis, hogy mennyire előnyös számára az adott állás. Ezt a becslést végzi el ez a függvény. A legnagyobb, illetve a legkisebb érték továbbra is a győztes illetve a vesztes állást jelzi, viszont a nem végállásokat tartalmazó levélelemekhez, további egész számokat rendel a becslés alapján.

A kétszemélyes játékok alapvetően tehát ezt az algoritmust használják, különböző optimalizálásokkal kiegészítve. Mi az alfa-béta vágásnak nevezett optimalizációt néztük meg részletesebben is. Ennek lényege, hogy a csomópontokhoz rendelt értékeket adminisztrálja, és ezek alapján kizár olyan állásokat a fából, amelyre már biztos, hogy nem juthat el a játszma. Ezeket a farészleteket már nem kell felépítenie a programnak, így jelentősen gyorsítható a feldolgozás, illetve a felszabaduló erőforrást, esetleg mélyebb részfa felépítésére fordíthatja a program. Fontos következtetés, hogy a gépi játékos erőssége főként a kiértékelő függvény minőségétől, illetve a felépített részfa mélységétől függ.

Irodalomjegyzék

Alison Cawsey: Mesterséges intelligencia, Hungarian edition Panem Könyvkiadó Kft., 2002

Jenny Raggett, William Bains: Mesterséges intelligencia (A-Z), Akadémia kiadó, 1992

http://tablajatekos.hu/uj2001/fonak1.html

http://krono.inaplo.hu/index.php/inter/inter-halozati-jelensegek/525-a-mesterseges-intelligencia

http://www.mek.iif.hu/porta/szint/tarsad/konyvtar/informat/azinform/html/tartalom.html

http://www.sulinet.hu/eletestudomany/archiv/1997/9722/versenysakk/versenysakk.html

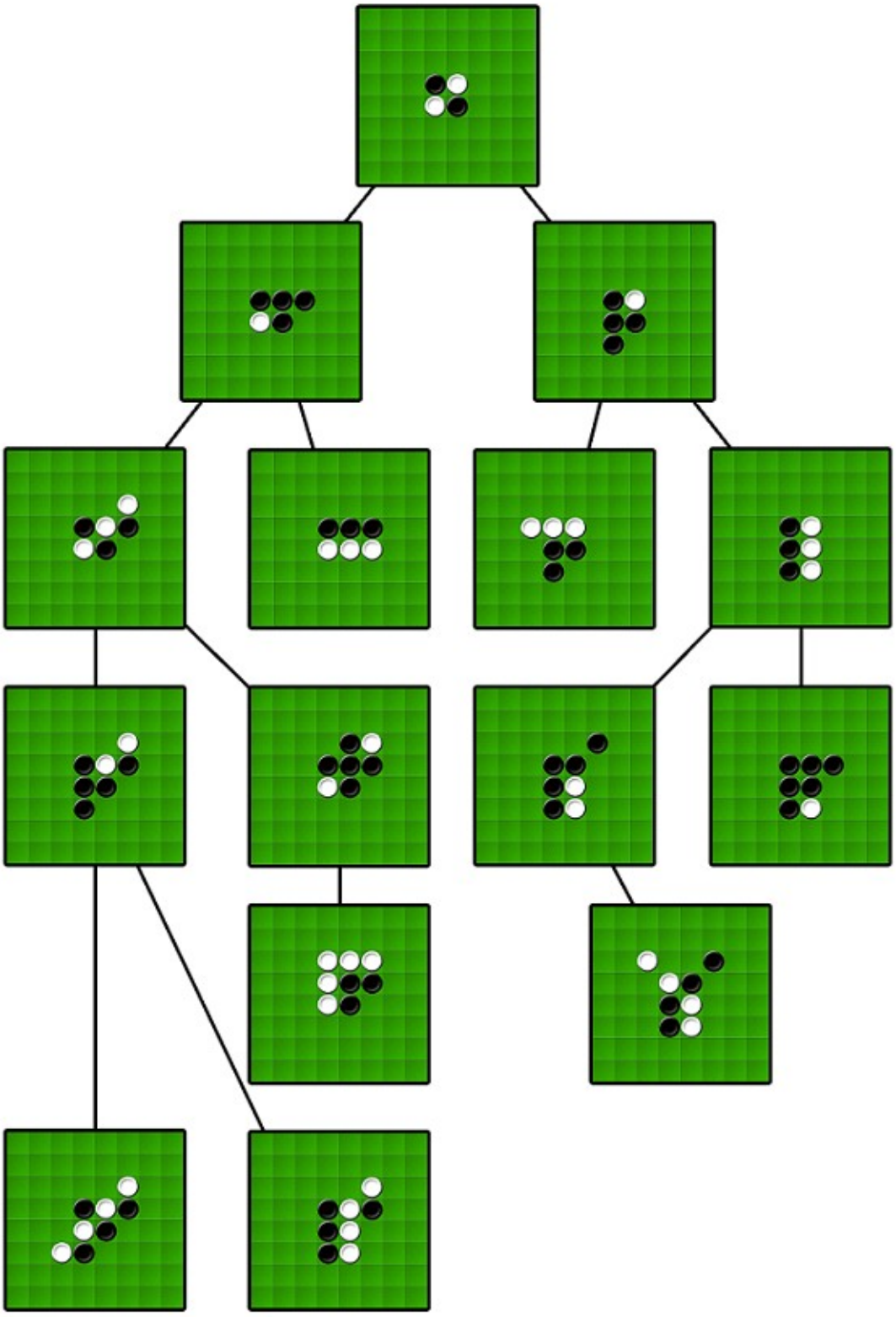
http://hu.wikipedia.org/wiki/IBM_Deep_Blue

http://index.hu/belfold/tegnapiujsag/2008/05/10/1997_a_deep_blue_megveri_kaszparovot/

http://www.inf.unideb.hu/~varteres/mi1folia/foliafo.pdf

http://yscik.com/jf/page.php?&id=291

Függelék



A Reversi játékfájának egy részlete.