Multidiszciplináris tudományok, 12. kötet, 2. sz. (2022), pp. 71–79 **https://doi.org/10.35925/j.multi.2022.2.6**

AKNAKERESŐ JÁTÉK MEGOLDÓ ALGORITMUS TOVÁBBFEJLESZTÉSE

Husóczki Dániel
hallgató, Miskolci Egyetem, Alkalmazott Matematikai Intézeti Tanszék
3515 Miskolc-Egyetemváros, e-mail: husoczki.daniel@student.uni-miskolc.hu

Baksáné Varga Erika
egyetemi docens, Miskolci Egyetem, Általános Informatikai Intézeti Tanszék
3515 Miskolc-Egyetemváros, e-mail: vargaerika@iit.uni-miskolc.hu

Absztrakt

A kutatás célja az aknakereső játék automatikus megoldó algoritmusok áttekintése és az Egypontos stratégia továbbfejlesztése. Ez az eljárás a legegyszerűbb és leggyorsabb megoldó algoritmus, mert egyetlen mező vizsgálata alapján hoz döntést a következő felfordítandó mezővel kapcsolatban. A vizsgálat során az Egypontos stratégia 4 változatát implementáltuk. Az alap algoritmus megáll, ha nem tud 100%-os biztossággal továbblépni. A további három algoritmus eltérő tippelési stratégiát alkalmaz, amikor nem tud biztonságosan dönteni. A futási eredmények igazolják, hogy a legjobb kidolgozott módszer hatékonysága kezdő és haladó szinten megközelíti a benchmark CSPS visszalépéses algoritmus eredményességét.

Kulcsszavak: *aknakereső játék, stratégiafejlesztés, játékszimuláció*

Abstract

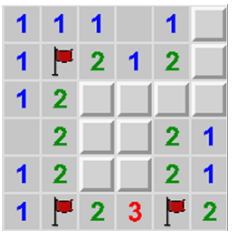
The aim of the research is to study the automatic solving algorithms of Minesweeper and to further develop the Single Point Strategy. This procedure is the simplest and fastest solving algorithm, since it makes a decision about the next tile to be uncovered based on the examination of a single tile. During the investigation, we implemented 4 versions of the Single Point Strategy. The basic algorithm stops if it cannot proceed with 100% certainty. The other three algorithms use different guessing strategies when they cannot make a safe decision. The presented results prove that the effectiveness of our best method approaches that of the benchmark CSPS backtracking algorithm at the beginner and intermediate levels.

Keywords: *Minesweeper, strategy development, game simulation*

1. Bevezetés

Az aknakereső egy egyszemélyes számítógépes logikai játék, amelyhez egy *n x m* méretű téglalap alakú pályát kell generálni, amelynek *k* darab mezőjébe aknák kerülnek, míg a többi mező üresen marad. A játék kezdetén a mezők tartalma el van rejtve. A játékos minden lépésben felfed egy mezőt. Ha aknát talál, a játékot elvesztette. Ha viszont üres mezőt, akkor a mezőben megjelenik egy szám, ami a mezővel érintkező aknamezők darabszáma. Egy mezőnek 8 szomszédja van, tehát maximum 8 akna lehet mellette. A játékos akkor nyer, ha felfedi az összes üres mezőt.

Az aknakereső játék automatikus megoldása bizonyítottan NP-teljes probléma (Kaye, 2000). Ez egyrészt azt jelenti, hogy gyorsan ellenőrizhető a megoldás helyessége, de nem biztos, hogy polinomiális időben rátalálunk a megoldásra. Ennek ideje ugyanis a tábla méretétől, az aknák számától és elhelyezkedésétől függ. Másrészt nem mindig egyértelmű egy játéktábla megoldása, mert előfordul olyan eset, amikor nincs olyan lépés, ami biztosan üres mezőt fed fel, vagyis a rejtett mezők közül véletlenszerűen, illetve egy valószínűségi változó értéke alapján kell választani *(1. ábra)*.



1. ábra. *Nem egyértelmű döntési helyzet* (minesweepergame.com)

Az aknakereső játékban három nehézségi szintet különböztetünk meg: kezdő, haladó és nehéz. Kezdő szinten a tábla mérete 8 × 8, 9 × 9 vagy 10 × 10 és tíz akna van elrejtve. A haladó szintű játékban 40 akna van és a tábla mérete 13 × 15 és 16 × 16 között változik. A legnehezebb szinten a játéktábla 16 × 30 vagy 30 × 16 méretű és 99 akna van rajta. Kezdő szinten rendszerint nincs szükség tippelésre és a négyzeteken lévő szám legtöbbször 1 vagy 2, ritkán 3. A nehézségi szint növekedésével egyre gyakoribb, hogy találgatni kell. A legmagasabb szinten a mezők számozása is magasabb, bár a 7 és a 8 ritkán fordul elő (Becerra, 2015).

A cikk célja összehasonlítani több aknakereső megoldó algoritmus hatékonyságát, azaz a megoldási idejét és a megoldás helyességét.

2. Ismert megoldó algoritmusok

2.1. Egypontos stratégia

Az Egypontos stratégia (Single Point Strategy, SPS) a legegyszerűbb és leggyorsabb, de ugyanakkor a legkevésbé hatékony ismert megoldó algoritmus. A stratégia a játéktábla egyetlen pontján rendelkezésre álló információ alapján hoz döntést, vagyis kiválaszt egy számozott mezőt és annak szomszédos mezőit az alábbi algoritmus szerint vizsgálja.

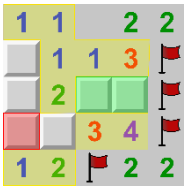
- Ha a mezőn lévő szám (szomszédos aknák száma) megegyezik a mező szomszédságában lévő megjelölt (zászlózott) mezők számával, akkor a többi fel nem fedett mezőt biztonságosan felfordíthatóként jelöli meg.
- Ha a mezőn lévő szám (szomszédos aknák száma) egyenlő a mező szomszédságában lévő megjelölt (zászlózott) mezők plusz a fel nem fedett szomszédok számával, akkor a fel nem fedett mezők mind aknát rejtenek.

Az algoritmus akkor ér véget, ha a játéktér egy iteráció után már nem változik, ami azt jelenti, hogy nincs biztonságosan felfedhető mező. Ekkor egy komplexebb megoldó algoritmusra kell átváltani, vagy tippelni kell. Ha azonban a következő iteráció után az algoritmus talál biztonságos mezőt, akkor visszatérhetünk az egypontos stratégiához. A SPS-nek több változata is létezik, melyek közül Becerra szakdolgozatában (Becerra, 2015) két algoritmust ismertet, a Naiv és a Double Set SPS eljárást.

Az alap SPS algoritmus komplexitása *n* mezőt tartalmazó játéktábla esetén $O(n^2)$. Kezdő szinten 77,4%, haladó szinten 29,1%, a legnehezebb szinten pedig mindössze 0,5% eséllyel oldja meg a feladványt (Pedersen, 2004).

2.2. Korlátozott keresési stratégia

A Korlátozott keresési stratégia (Limited Search Strategy, LSS) (Pedersen, 2004) egy akna helyének a megállapításához nem egyetlen mezőt, nem is a teljes játéktáblát, hanem a játéktáblának egy bizonyos részét, az úgynevezett "releváns zónát" vizsgálja. Visszalépéses (backtracking) algoritmust használ, ami azt határozza meg, hogy egy aknának a megjelölése egy bizonyos mezőn ellentmondáshoz vezet-e.



2. ábra. *A piros mező releváns zónája* (minesweepergame.com)

A releváns zóna azon számozott mezők halmaza, amelyek egy kiválasztott ismeretlen mező közelében vannak, és a leghasznosabbak annak meghatározásához, hogy a kiválasztott mezőn van-e akna vagy sem. A *2. ábrá*n láthatjuk, hogy a sárgával jelölt mezők tartoznak a pirossal jelölt mező releváns zónájába. A zölddel jelölt két mező közül az egyik mezőnek aknának kell lennie a körülöttük lévő számozott és zászlózott mezők miatt, ennélfogva a piros mezőtől rögtön jobbra lévő mezőnek is aknának kell lennie, így megállapíthatjuk, hogy a pirossal jelölt mezőn nincs akna. Azonban vannak olyan esetek is, amelyek nem ennyire egyértelműek, és nem állapítható meg biztonságosan az, hogy a piros mező akna-e vagy sem. Ilyenkor az algoritmus továbbhalad, és ha a későbbiekben ellentmondásba kerül, akkor használja a visszalépést. Azaz visszatér az első olyan döntésre, amely nem volt teljesen egyértelmű, és egy másik mező felfordításával újrakezdi a megoldást.

Ez az algoritmus a kezdő szintű játékot 91,7%-os, a haladó szintűt 64,3%-os, a legnehezebb szintűt pedig 17%-os arányban megnyeri. Az eljárás komplexitása megegyezik az SPS algoritmuséval, de a nehezebb szinteken, amikor többször kell tippelni, a gyakori visszalépések miatt a futási ideje hosszabb (Pedersen, 2004). Pedersen az algoritmust kiegészítette azzal, hogy tippelés előtt kiszámítja a releváns zóna mezőire annak a valószínűségét, hogy ott akna van, és ezt felhasználva a legkisebb valószínűséggel rendelkezőt választja. Ez a kiegészítés javítja az eljárás hatékonyságát (Limited Search with Probability Estimates Strategy, LSPES). Kezdő szinten 92,5%, haladó szinten 67,7% és mester szinten 25% a nyerés esélye (Pedersen, 2004).

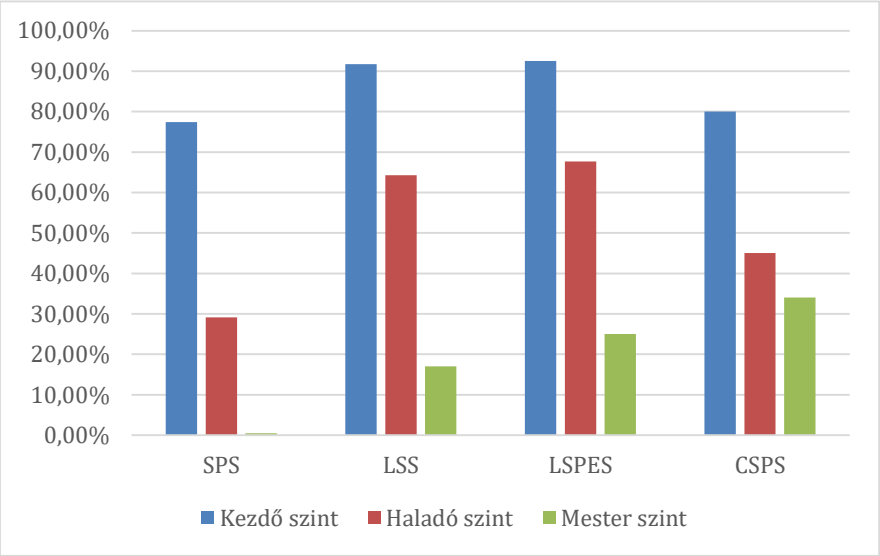
2.3. Kényszerkielégítési probléma stratégia

Az LSS algoritmus továbbfejlesztése, amikor kényszerkielégítési problémaként tekintünk az aknakereső játék megoldására (Constraint Satisfaction Problem Strategy, CSPS) (Studholme, 2001). Ez az eljárás valószínűségi értéket köt minden vizsgált mezőhöz, azaz kiszámítja az esélyét annak, hogy a mező alatt akna van. A valószínűség számítása úgy működik, hogy a kiválasztott mező esetén felteszi, hogy akna van alatta, majd a releváns zóna és a visszalépéses algoritmus alkalmazásával megszámolja, hogy mennyi olyan eset lesz, amikor az algoritmus a további lépéseiben ellentmondásra jut, és mennyi olyan

eset amikor nem. Ezeket az adatokat figyelembe véve kiszámítja, hogy mennyi az esélye az akna jelenlétének a mezőn, majd a legkisebb valószínűségű mezőt fordítja fel amikor tippelni kell.

Studholme eljárása azon alapul, hogy a játéktérben minden állapotot kényszerkielégítési problémaként kezel és a kielégítendő kényszereket egyenletekkel írja le. Az egyenleteket egy visszalépéses algoritmus oldja meg, ami minden ismeretlenhez hozzárendel egy értéket, majd pedig ellenőrzi, hogy így kielégítette-e a kényszereket. Ha nem és az egyenletrendszert még meg lehet oldani, akkor egy újabb hozzárendelésre kerül sor. Ha viszont az egyenletrendszert már nem lehet megoldani, akkor visszalép az előző konfigurációra.

Ez az algoritmus a kezdő szintű játéktáblát 80%-os, a haladó szintűt 45%-os, a legnehezebb szintűt pedig 34%-os biztossággal oldja meg (Studholme, 2001). Így ez a jelenleg ismert leghatékonyabb módszer mester szinten, amikor sokszor előfordul, hogy nem egyértelmű a következő lépés.



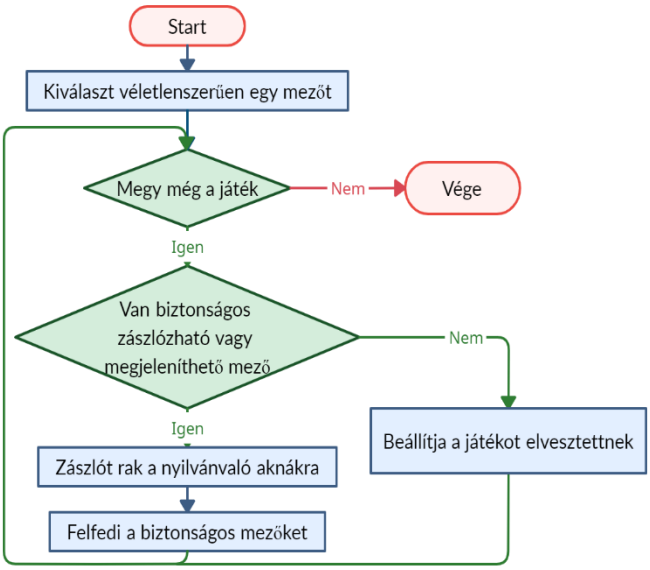
3. ábra. *Ismert megoldó algoritmusok hatékonyságának összehasonlítása*

3. Az Egypontos stratégia továbbfejlesztése

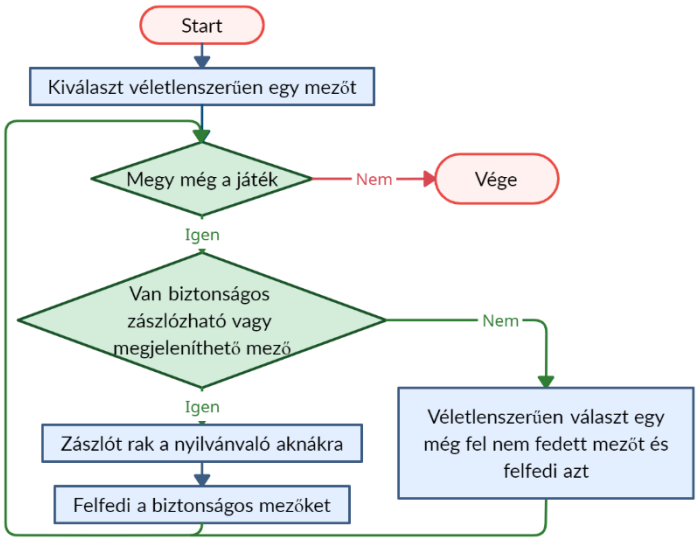
Az Egypontos stratégia előnye a gyors végrehajtási idő. Ezért ezt az algoritmust választottuk tovább-fejlesztésre azzal a céllal, hogy a játéktábla-megoldások arányát növeljük. Három algoritmust készítettünk, amelyeknek az első két lépése ugyanaz, mint az SPS eljárásban, azaz mindegyik először megkeresi a nyilvánvaló bombákat és a számozott mezőket. Az eltérés akkor jelentkezik, amikor az algoritmus elakad, vagyis tippelni kell, mert itt különböző stratégiákat implementáltunk (Husóczki, 2022).

Az alap SPS eljárás két lépésből áll *(4. ábra)*. Először is az algoritmus elkezdi megkeresni az aknákat és zászlókat rak azokra, ezután felfedi az összes olyan mezőt, amelyről azt állapítja meg, hogy biztonságos. Ha az SPS algoritmus elakad a programban, akkor nem folytatja tovább a játékot, nem próbál meg tippelni vagy átváltani másik algoritmusra, hanem elveszítettnek jelzi a jelenlegi pályát.

A véletlen választással kiegészített SPS eljárás során (Single Point Strategy with Random Tile, SPSRT) az algoritmus nem áll meg, ha kifogyott a biztonságos lépésekből, hanem véletlenszerűen választ egyet a még lefedett mezők közül *(5. ábra)*.

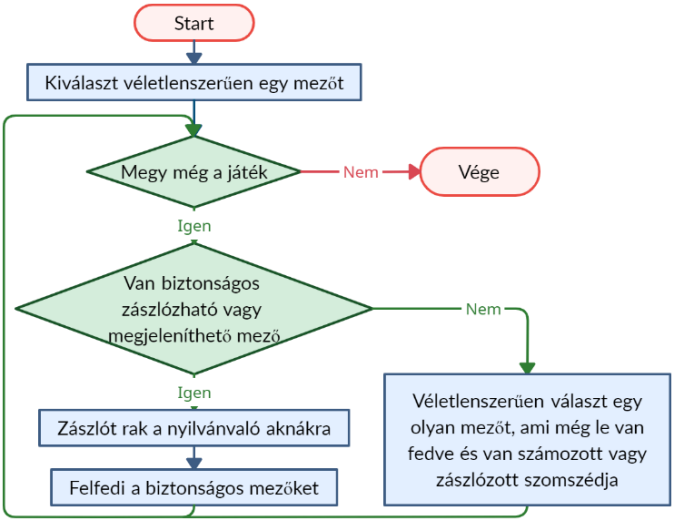


4. ábra. *Az SPS algoritmus folyamatábrája*



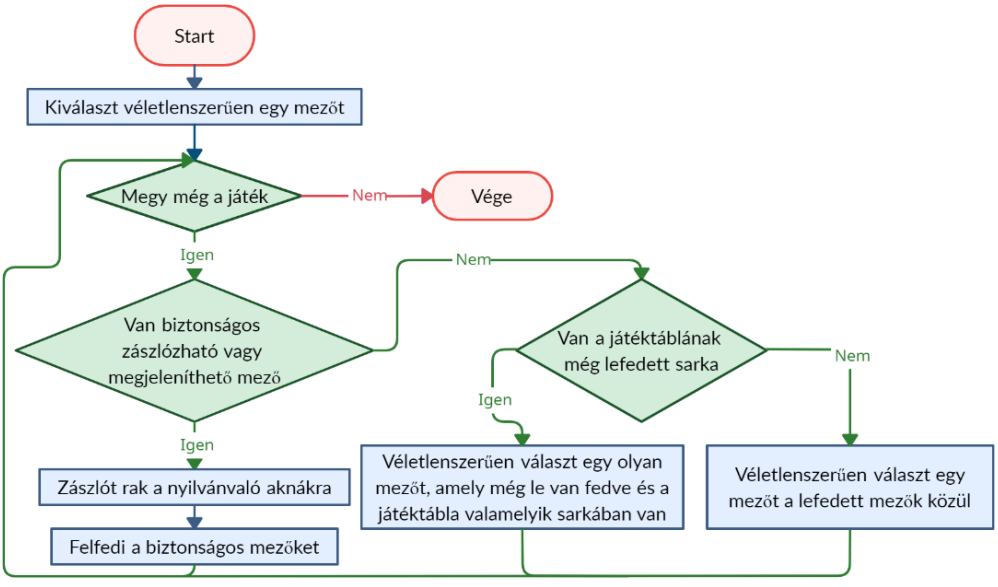
5. ábra. *Az SPSRT eljárás folyamatábrája*

Az előző módszer további finomítása a véletlen szomszéd kiválasztása (Single Point Strategy with Random Neighbouring Tile, SPSRNT). A tippeléssel felfordított mezőt úgy határozza meg, hogy kiválasztja a még lefedett mezők közül azokat, amelyeknek vannak számozott vagy zászlózott szomszédjai, és ezekből véletlenszerűen választ egyet *(6. ábra)*.



6. ábra. *Az SPSRNT algoritmus folyamatábrája*

Az utolsó módszer azt a gyakorlati megfontolást implementálja, hogy tippeléskor érdemes először a sarokpontokat felfordítani (Single Point Strategy with Random Corner Tile, SPSRCT). Azaz az algoritmus az első két lépés elvégzése után nem áll meg, hanem véletlenszerűen kiválaszt egyet a még lefedett sarokmezők közül. Miután azokat felfedte, az SPSRT eljáráshoz hasonlóan összegyűjti azokat a mezőket, amelyek még le vannak fedve, és ezekből választ egyet véletlenszerűen *(7. ábra)*.

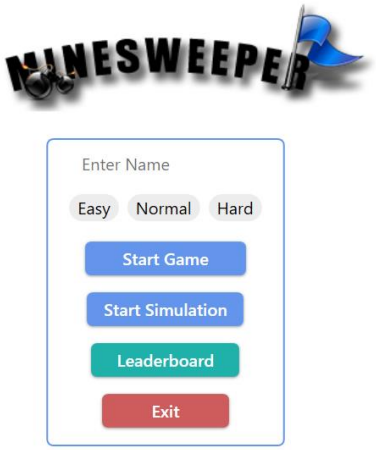


7. ábra. *Az SPSRCT eljárás folyamatábrája*

4. Futási eredmények

Az algoritmusok teszteléséhez készítettünk egy Windowsos asztali alkalmazást C# programozási nyelven (letölthető innen: https://github.com/husoczkidani/HLO5ZK_Szakdolgozat). A játéktábla generálásakor 3 nehézségi szint közül lehet választani *(8. ábra)*:

- Kezdő szint (Easy): 9 × 9 mező, 10 akna
- Haladó szint (Normal): 16 × 16 mező, 40 akna
- Mester szint (Hard): 30 × 16 mező, 99 akna



8. ábra. *Az aknakereső játék induló képernyője*

Minden megoldó algoritmust lefuttattunk 10 000 játékra minden nehézségi szinten. A kapott eredményeket a következő táblázatok szemléltetik. Az eljárások hatékonyságát a *megnyert játékok száma / összes játék* képlettel számoltuk és százalékban adjuk meg. Az utolsó oszlopban a teljes szimuláció futási ideje szerepel *perc:másodperc:milliszekundum* formátumban. A szimulációkat AMD Ryzen 3 2200G processzorral és HyperX Fury 4 × 4 GB 2666MHz memóriával rendelkező számítógépen futtattuk.

1. táblázat. *Futási eredmények kezdő szinten*

Algoritmus	**Hatékonyság**	**Zászlózott**	**Felfedett**	**Futási idő**
SPS	46,06%	59,3%	59,6%	00:21:30
SPSRT	76,84%	82,5%	78,5%	00:22:91
SPSRNT	68,41%	74,6%	71,0%	00:21:30
SPSRCT	78,64%	83,9%	80,3%	00:23:74

Az SPS algoritmus hatékonysága azt mutatja, hogy az esetek több mint felében tippelnie kellett volna az eljárásnak. A futási időket nézve látható, hogy a tippelési stratégiák bevezetésével kezdő szinten nem nőtt meg számottevően a végrehajtási idő. Viszont a hatékonyság jelentősen javult. A legjobb eljárás 32,58%-kal több játékot nyert, mint az alapalgoritmus.

2. táblázat. *Futási eredmények haladó szinten*

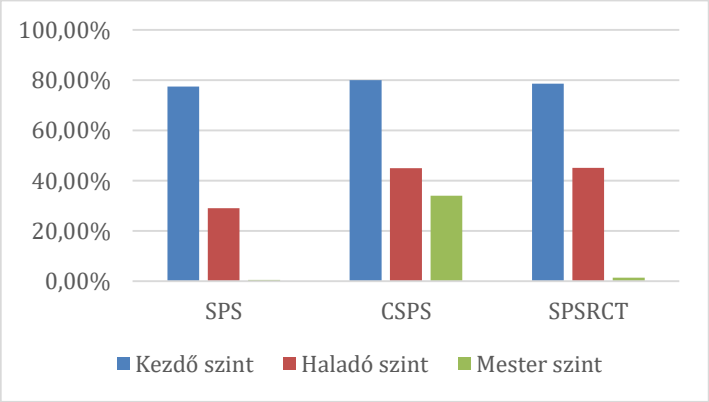
Algoritmus	**Hatékonyság**	**Zászlózott**	**Felfedett**	**Futási idő**
SPS	14,37%	39,3%	36,9%	02:03:47
SPSRT	43,16%	67,9%	61,2%	04:27:32
SPSRNT	35,79%	58,0%	51,9%	03:26:94
SPSRCT	45,14%	71,2%	64,2%	04:10:77

Haladó szinten a bizonytalan lépések száma jóval több, mint kezdő szinten. Ezt mutatja, hogy az SPS eljárás a játékoknak csak 14,37%-át tudta megoldani biztonságos lépésekkel. Ezen a szinten a legjobb eljárásunk több mint 40%-kal hatékonyabb az alapalgoritmusnál, cserébe viszont a végrehajtási idő megduplázódott.

3. táblázat. *Futási eredmények mester szinten*

Algoritmus	**Hatékonyság**	**Zászlózott**	**Felfedett**	**Futási idő**
SPS	0,07%	19,9%	21,2%	05:33:26
SPSRT	1,30%	30,2%	30,7%	09:43:02
SPSRNT	1,24%	29.3%	28.6%	11:02:78
SPSRCT	1,34%	31.2%	32.1%	11:13:79

A szimuláció lefuttatása a legnehezebb pályára jelentősen igénybe vette a számítógép erőforrásait. Azonban a hatékonyság növekedés, amit a tippelési stratégiák bevezetésével sikerült elérni, figyelemre méltó. A legjobb eljárás tizenkilencszer több esetben oldotta meg a játékot, mint az alapalgoritmus.



9. ábra. *A továbbfejlesztett algoritmus összehasonlítása az ismert eljárásokkal*

A futási eredményekből látszik, hogy az utolsó, a sarokpontok felfedésével indító eljárás a leghatékonyabb módszer mindhárom szinten. Kezdő szintű játék esetén 78,64%, haladó szinten 45,14%, míg mester szinten 1,34% eredményességet produkált. Ezek az értékek mindhárom szinten jobbak, mint az irodalomban publikált Egypontos stratégia (SPS) hatékonysági mutatók. Ezenfelül az SPSRCT eljárásunk kezdő szinten megközelíti a Studholme által felállított benchmark 80%-os eredményességi küszöböt, haladó szinten pedig meghaladja az elvárt 45%-os küszöbértéket (CSPS). Ezt szemlélteti a *9. ábra*. Figyelembe

véve, hogy az Egypontos stratégiának a legegyszerűbb a megvalósítása, a legkisebb a memóriafoglalása és a leggyorsabb a futása, az elért eredmények felülmúlják a várakozásunkat és azt igazolják, hogy az SPSRCT eljárás a gyakorlatban hatékonyan alkalmazható kezdő és haladó szintű játéktábla megoldására.

5. Összegzés

A kutatás célja az aknakereső játék megoldó algoritmusok vizsgálata után az Egypontos stratégia továbbfejlesztése volt. Ez az eljárás a legegyszerűbb és leggyorsabb megoldó algoritmus, mert egyetlen mező vizsgálata alapján hoz döntést a következő felfordítandó mezővel kapcsolatban. Ha rossz döntést hoz, a játékot elvesztettük; vagyis nem alkalmaz visszalépést, amikor patthelyzet alakul ki. Ennek az eljárásnak számos változata ismert, és a legjobb esetben kezdő szinten 77,4%, haladó szinten 29,1%, illetve mester szinten 0,5%-os eredményességi rátáról számol be az irodalom.

A vizsgálat során az Egypontos stratégia 4 változatát implementáltuk. Az alapalgoritmus megáll, ha nem tud 100%-os biztossággal továbblépni. Ezt csak referenciaként valósítottuk meg, hogy méréskor lássuk az egyes nehézségi szinteken a bizonytalan döntések arányát. A további három algoritmus eltérő tippelési stratégiát alkalmaz, amikor nem tud biztonságosan dönteni. A futási eredmények azt igazolják, hogy a sarokpontok felfedésével kezdő eljárás a leghatékonyabb módszer mindhárom szinten. A feladványoknak kezdő szinten 78,64%-át, haladó szinten 45,14%-át, míg mester szinten 1,34%-át oldotta meg sikerrel. Ezek az értékek mindhárom szinten jobbak, mint az irodalomban publikált Egypontos stratégia eredményességi mutatók. Ezenfelül az SPSRCT eljárásunk hatékonysága kezdő szinten megközelíti a benchmark 80%-os eredményességi küszöböt, haladó szinten pedig meghaladja az elvárt 45%-os küszöbértéket. Mivel a mester szintű feladványoknál sok a bizonytalan lépés, a sarokpontokkal indítás mellett szükség van az algoritmus további kiegészítésére, ami a kutatás következő fázisa lesz.

Irodalom

[1] Kaye, R. (2000). Minesweeper is NP-complete. *The Mathematical Intelligencer,* 22, pp. 9–15. **https://doi.org/10.1007/BF03025367**

[2] *The authoritative minesweeper - Strategy & Cheats*. https://minesweepergame.com/ (Letöltve: 2022. 7. 29.).

[3] Becerra, D. J. (2015). *Algorithmic Approaches to Playing Minesweeper*. Bachelor's thesis, Harvard College.

[4] Pedersen, K. (2004). *The complexity of minesweeper and strategies for game playing*. Department of Computer Science, University of Warwick, pp. 30–62.

[5] Studholme, C. (2001). *Minesweeper as a constraint satisfaction problem*. http://www.cs.toronto.edu/~cvs/minesweeper/minesweeper.pdf (Letöltve: 2022. 7. 27.).

[6] Husóczki D. (2022). *Aknakereső játék algoritmusok vizsgálata*. Szakdolgozat, Miskolci Egyetem, GÉIK, Alkalmazott Matematikai Intézeti Tanszék.