SameGame NP teljes puzzle megoldásának megközelítése különböző módszerekkel

Kárpáti Márk András

2022 Tavaszi félév

1 A SameGame játék

SameGame egy puzzle játék, amit Kuniaki Moribe talált ki chain shot! néven 1985-ben. Fujitsu FM-8/7 sorozatra volt kiadva, és egy havi lapban jelent meg egy számítógépes napi lapban. [7]. Később Eiji Fukumata a SameGame névvel újra kreaálta 1992-ben.

1.1 Szabályok

- pálya mérete 15x15
- 5 különböző fajta mező szín van.
- Kezdetben a pálya összes mezején van valamilyen szín
- Egy lépés egy csoport eltávolításából áll.
- Egy csoport legalább 2 egyszínű mező, ami érintkezik egymással, szomszédosak.
- Egy lépés $(n-1)^2$ pontot ér, ahol n az egy lépésben eltüntet mezők száma
- Nincs idő korlát.
- Nincs lépés szám korlát.
- Minden lépés után a játék pálya "összeomlik" ha van üres hely az elemek lefelé mozdulnak, ha van egy egész üres oszlop, akkor balra felé tolódnak az oszlopok.
- A játék véget ér, ha elfogytak a mezők vagy nem marad lehetséges lépés

1.2 SamGame komplexitása

A komplexitása, azt mutatja, hogy mennyire nehéz megoldani a játékot. Két fontos mértéke a játék-fa komplexitás és az állapottér komplexitás.[1] A játék-fa kompleszítás a levélcsomópontok száma a megoldási fában az induló pozícióból. Az állapottér kompleszítás az elérhető játék állások a kezdő pozicionál.

A SameGambe-ben a játék-fa komplexitás nagyjából b^d, ahol b az átlagos elágazási faktor és a d az átlagos játék hossza.

2 Kereső algoritmusok puzzle-okhoz

Klasszikus megközelítése a puzzle-k megoldásának **A*** [4] és az **IDA*** [6]. Az A* egy Legjobbat először keresés, ahol minden csúcs el van tárolva egy listában. Ez a lista rendezve van egy elfogadható kiértékelő függvénnyel. Minden iterációban az első elemet eltávolítjuk, és a gyereke elemeit hozzá adjuk a rendezett listához. Ezt addig ismételjük, amíg a célállapot nem kerül a lista legelejére. IDA* az egy iteratívan mélyülő változata az A* keresésnek. Úgy használ mélységi bejárást, hogy nem kell eltárolni az egész fát a memóriában. Addig keres mélységi eljárással, amíg nem ér levébe és a kiértékelő függvény értéke nem ér el egy küszöböt. Ha a keresés eredmény nélkül tér vissza, akkor a küszöböt növeljük.

Mind két módszer nagyon erősen függ a kiértékelő függvény minőségétől. Még ha a függvény egy elfogadható alsó becslés, akkor is elég pontosnak kell lennie. Klasszikus puzzle játékoknál mint a Nyolcas játék, annak nagyobb változatai [6] és a Sokoban [5] ez a megközelítés nagyon jól működik. Hiszen ott egy jó alul-becslő függvény a Manhattan távolság. A fő feladata ennek területnek a kiértékelő függvény javítása pl mintaadatbázisokkal. [3] [2]

Ezek a megközelítési módok elbuknak a SameGame esetén, mert nem annyira egyszerű egy elfogadható függvényt (nem becsli túl az árat) készíteni, ami még mindig elég pontos. Erre az egyik próbálkozás, hogy pontokat asszociálunk a táblán lévő csoportokhoz. Ez vissza is adja azt a pontszámot, amit nagyjából kapnánk, ha föntről lefelé haladva eltávolítanánk ezeket a csoportokat. De ha egy optimális pontszámot szeretnénk elérni a SameGame-ben, akkor a jelenlegi helyzethez egy "túl becslő" lenne szükségünk, hiszen nem a legrövidebb utat keressük, hanem a legtöbb pontot szeretnénk elérni. Egy elfogadható becslő függvényt kapunk, ha úgy tekintjük, hogy az összes ugyan olyan színű mezőt egyszerre el tudjuk távolítani a pályáról. Ezt lehet javítani azzal, hogy mínusz pontot ad, ha ezzel a táblát nem tudjuk teljesen kiüríteni. Azonban ez messze van az igazi ponttól, amit egy helyzetben lehet szerezni, és nem működik jól az A* és IDA*-val sem. A tesztek azt mutatják, hogy ezzel a becslővel ellátott A*, IDA* Szélességi kereséshez hasonlóan viselkednek.[8] A probléma az, hogy a gyerek heurisztikus értéke sokkal kisebb lesz mint a szülője, hacsak nem egy nagy csoportot távolított el. Sejtések szerint ugyan ettől szenvednek a mélységi

keresés, az Elágazás és korlátozás algoritmus is. [9] Ezekből kifolyólag jelent kihívást a SameGame a puzzle-k kutatásában.

3 Megközelítés

A cél az, hogy minél jobb algoritmust találjak, a játék megoldására. Ehhez először könnyítettem a játék szabályokon. Az így készült algoritmussal és az eredeti szabályokkal legenerált játék pályákról nagy valószínűséggel megmondható lesz, hogy meglehet-e oldani őket.

Használt Szabályok:

Főbb funkciói a programnak:

- Teljesen véletlenszerű pálya generálás, ahol a pálya mérete és a mezők típusának számossága megadható
- Pálya beolvasása megadott .csv formátumban.
- Pálya kiírása a beolvasáshoz megfelelő .csv formátumban.

A pálya .csv formátumú fájlokban van tárolva, aminek első sora oszlop és sorok száma. Ezt követi a pálya az összes adat ';' karakterrel van elválasztva.

Elmélet:

References

[1] Louis Victor Allis et al. *Searching for solutions in games and artificial intelligence*. Ponsen & Looijen Wageningen, 1994.

[2] Joseph C Culberson and Jonathan Schaeffer. Pattern databases. *Computational Intelligence*, 14(3):318–334, 1998.

[3] Ariel Felner, Uzi Zahavi, Jonathan Schaeffer, and Robert C Holte. Dual lookups in pattern databases. In *IJCAI*, pages 103–108, 2005.

[4] Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[5] Andreas Junghanns. *Pushing the limits: new developments in single-agent search*. University of Alberta, 2000.

[6] Richard E Korf. Depth-first iterative-deepening: An optimal admissible tree search. *Artificial intelligence*, 27(1):97–109, 1985.

[7] K. Moribe. Chain shot! gekkan ascii.

[8] Maarten P.D. Schadd, Mark H.M. Winands, Mandy J.W. Tak, and Jos W.H.M. Uiterwijk. Single-player monte-carlo tree search for samegame. *Knowledge-Based Systems*, 34:3–11, 2012. A Special Issue on Artificial Intelligence in Computer Games: AICG.

[9] Nageshwara Rao Vempaty, Vipin Kumar, and Richard E Korf. Depth-first versus best-first search. In *AAAI*, pages 434–440, 1991.