Гомоку: Вовед и Документација за AI Решение

Виктор Најдовски
Проф. д-р Соња Гиевска

Оливер Бутески
Проф. д-р Кире Триводалиев

Абстракт

Оваа документација се фокусира на презентирање AI решение за стратешката игра Гомоку. Покрај постигнувањето точност во играта, огромниот простор на состојби во комбинација со брзата експанзија на дрвото на одлуки ја трансформира задачата и на проблем на оптимизација. Истражуваме веќе воведена евристика и ја споредуваме нејзината изведба со други алгоритми, при што откриваме дека се истакнува во повеќето метрики.

I Вовед

Стратешките игри, како што се Шах и Го, долго време се користени како тест случаи за AI алгоритми бидејќи бараат длабоко стратешко планирање и способност за предвидување повеќе чекори нанапред. Во оваа документација се фокусираме на играта Гомоку, игра со едноставни правила, но комплексен простор на состојби. Ова ја прави идеална игра за истражување на балансот помеѓу пресметковна ефикасност и тактичка длабочина.

II Вовед во играта Гомоку

Гомоку е стратешка игра со која двајца играчи наизменично поставуваат камчиња на 15×15 табла. Целта е да се формира непрекината линија од пет камчиња во било кој правец - хоризонтално, вертикално или дијагонално. Црниот играч игра прв, што му дава предност ако не се применат правила за балансирање, како што е направено на натпревари. Главните предизвици при изградба на AI за Гомоку се огромниот простор на состојби, брз раст на дрвото на можности и долготрајната стратегија.

Огромен простор на состојби

Бројот на можни конфигурации на таблата расте експоненцијално со нејзината големина. За игра со големина $n \times n$, теоретската горна граница на можни состојби е 3^{n^2}. За табла од 15×15, што е стандард за играта Гомоку, тоа резултира во:

$$3^{n^2} = 3^{15^2} = 3^{225} \approx 10^{107} \text{ состојби},$$

што е далеку повеќе отколку што може да се обработи со методи на груба сила.

Брз раст на дрвото на можности

На почетокот на играта, првиот играч има 225 можни потези, потоа вториот 224, првиот 223 итн. Како што играта напредува, бројот на можни потези се намалува, но дрвото на можности сепак се шири брзо. Ако претпоставиме дека просечниот фактор на разгранување е од 30 до 50 (бидејќи потезите најчесто се прават во близина на веќе постоечки кластери на каменчиња), бројот на јазли на длабочина d е $O(b^d)$, каде што b е факторот на разгранување. Ако го фиксираме b да биде 30, добиваме:

$$\text{Длабочина 1: } 30^1 = 30$$
$$\text{Длабочина 2: } 30^2 = 900$$
$$\text{Длабочина 3: } 30^3 = 27000$$
$$\text{Длабочина 4: } 30^4 = 810000$$
$$\text{Длабочина 5: } 30^5 = 24300000$$

Над длабочина 6 или 7, пребарувањето станува непрактично, што наложува употреба на евристики, алфа-бета поткаструване и хеширање за оптимизација.

Долготрајна стратегија и проценка на закани

Ефективното играње бара предвидување на идните потези, идентификување на закани и препознавање на можни победнички секвенци. Нашата евристичка функција мора ефикасно да ги проценува состојбите на таблата, давајќи приоритет на потези што создаваат или блокираат критични полиња.

Со оглед на овие предизвици, нашето AI решение користи МинМакс алгоритам, подобрен со алфа-бета поткаструване и хеширање, за ефикасна навигација низ дрвото на можности.

III Предложено МинМакс Решение

МинМакс алгоритам

МинМакс е рекурзивен алгоритам за носење на одлуки кај проблеми каде двајца играчи се натпреваруваат. Алгоритамот го истражува дрвото на можни потези, претпоставувајќи дека и двајцата играчи играат оптимално. На секое ниво, алгоритамот го максимизира својот резултат, додека го минимизира најдобриот можен потег на противникот. Во нашето решение користиме ограничување на длабочина што значителни ја подобрува временската сложеност. Ограничувајќи ја длабочината на пребарување, временската сложеност се намалува на $O(b^{d_{max}}) = O(225^3)$ во нашиот случај.

Алфа Бета Поткаструване

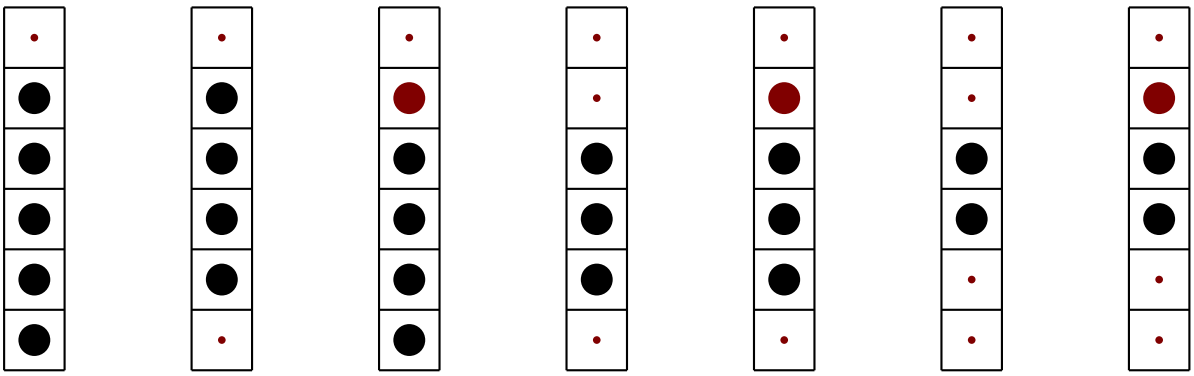
МинМакс алгоритамот ги истражува сите можни потези во дрвото на можности (до одредена длабочина). Ова ни гарантира дека ќе истражи и потези што се објективно лоши, ако тие постојат. Овие 'лоши' потези не се потребни при наоѓање на оптималната стратегија, но сепак се евалуираат. Како што длабочината на пребарување расте ова станува многу неефикасно и временски напорно. Ако наидеме на 'лоша' гранка, користејќи алфа-бета поткаструване, можеме да престанеме да ја пребаруваме, но само ако веќе имаме најдено подобра гранка. Ова значително ја подобрува перформансата на алгоритмот.

Хеширање

Методот на хеширање што ние го имплементиравме е Зобрист хеширање. Во овој пристап, секоја состојба се доделува уникатна 64-битна низа. 64-битната низа се генерира преку доделување на случајни 64-битни вредности за секој можен елемент од состојбата на играта. Овие случајни вредности се комбинираат со помош на XOR операцијата за да се создаде хеш за целата состојба. Првите два бита се користат за претставување на длабочината на дрвото на состојби, додека преостанатите бита ги претставуваат другите информации поврзани со состојбата. Кога ќе се сретне нова состојба, нејзината соодветна хеш вредност се зачувува во хеш табела. Кога истата состојба ќе се сретне повторно, наместо да се пресметува пак, ја влечеме од хеш табелата.

Евристика

Евристиката што ние ќе ја користиме е предложена во истражувањето на Han Liao[1]. Таа се фокусира на доделување на вредности на позициите базирани на тоа колку се блиску до формирање на линија од пет последователни камчиња. Дава повисоки вредности за 'живи' линии, т.е. линии со отворени краеви, и ги разгледува секвенците од две или три камчиња како потенцијални закани или можности. Дополнително, специфично доделените вредности овозможуваат евристиката да определи кога треба да даде приоритет на одбранбените потези пред офанзивните. Според евристикате имаме 7 главни секвенци на камчиња кои можат да се видат подолу.



пет по ред | жива четворка | мртва четворка | жива тројка | мртва тројка | жива двојка | мртва двојка

Постојат многу варијации на главните 7 секвенци. Секој потег се бодува врз основа на секвенцата што ја продуцира, со повисок резултат даден на подобрите секвенци. Потегот, исто така, може да произведи повеќе секвенци, придонесувајќи на неговиот целокупен резултат и правејќи го подобар при донесувањето одлуки.

Кај нас е имплементирано така што најдобрата секвенца, пет по ред, се бодува со 100.000, а секоја следна со 10.000 помалку.

Евристиката искажана преку псевдо-код

```
Let S ← 0                                                    // Move value
Let D ← {(1,0),(0,1),(1,1),(1,−1)}                           // Directions
foreach d ∈ D do
    player_streak ← 0
    opponent_streak ← 0
    open_ends ← 0
    foreach step ∈ [−4,4] do
        cell ← (x + step · d_x, y + step · d_y)
        if cell contains P then
            player_streak ← player_streak + 1
            opponent_streak ← 0
        end
        else if cell contains O then
            opponent_streak ← opponent_streak + 1
            player_streak ← 0
        end
        else
            open_ends ← open_ends + 1
        end
    end
    Update S based on player_streak, opponent_streak, open_ends
end
return S
```

IV Споредба со Random и Uniform Cost Search

Random Стратегија

Во Гомоку, стратегијата на случаен избор подразбира правење потези без какво било стратешко планирање или евристичка евалуација. Секој потег е избран рамномерно на случаен начин од легалните позиции на таблата. Оваа стратегија обично се покажува слабо во споредба со структурираните алгоритми за одлучување како што се МинМакс или UCS. Поради недостатокот на претскажување, таа е крајно ранлива на стратешки замки и закани што ги поставуваат посилните противници.

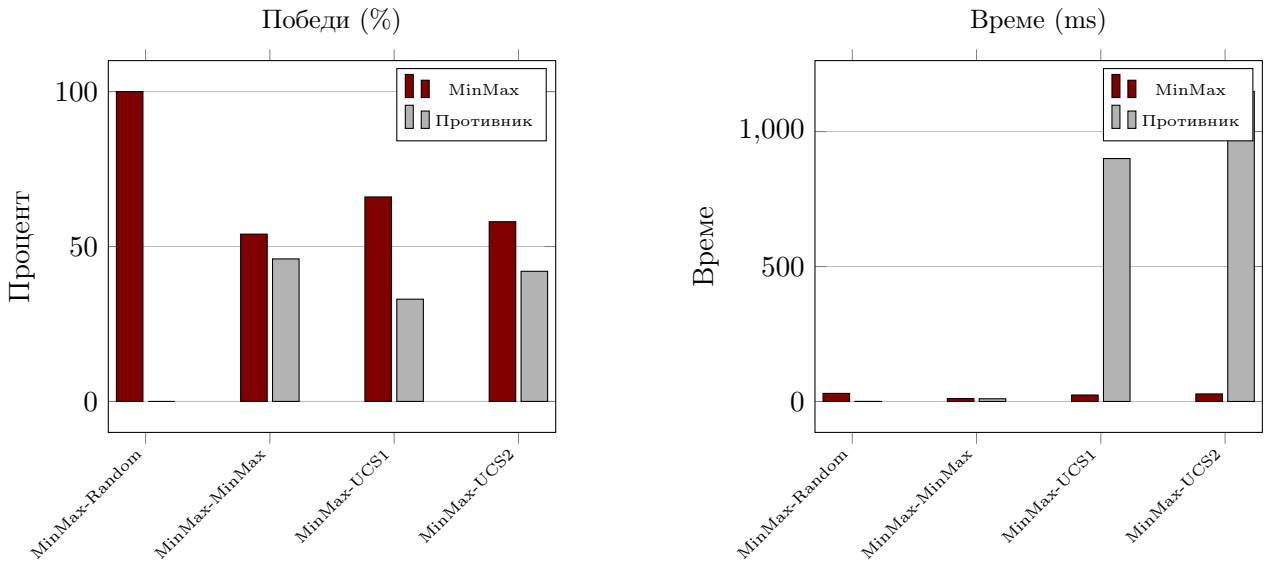
Uniform Cost Search (UCS)

Алгоритмот за пребарување на патеки Uniform Cost Search (UCS) прво го проширува јазолот со најмала цена. UCS може да се модифицира за да ги истражува состојбите во играта Гомоку со користење на функција на цена наместо длабочина. За разлика од Minimax, кој ги оценува позициите врз основа на оптимизирање на предноста, UCS се обидува да ја одреди најевтината серија потези за да се постигне победничка состојба. Поради тоа што не го намалува директно предноста на противникот, UCS може да биде помалку ефикасен во конфронтациски ситуации. Недостатокот на дефанзивна игра и стратешка длабочина често го прави послаб против МинМакс.

Визуелизација на резултати

Следните визуализации ја прикажуваат перформансата на овие AI стратегии во однос на стапките на победа и времето на пресметка.

Како што е прикажано на сликите подолу, алгоритмот MinMax значително ја надминува основната случајна (Random) стратегија и постигнува малку повисока стапка на победа кога започнува прв против самиот себе. Против UCS, MinMax одржува стапка на победа над 55% и во двета сценарија – кога започнува прв и кога започнува втор, иако неговата стапка на победа малку опаѓа кога започнува втор. Анализата на времето на пресметка покажува дека UCS бара значително повеќе време по потег во споредба со MinMax, додека случајната стратегија (Random) бара најмалку време.



Слика 1: Споредба на алгоритми

Алгоритам	**Победи (%)**	**Време (ms)**	**Време (ms) Противник**
MinMax vs Random	100	30	0.2
MinMax vs MinMax	54	11	10
MinMax vs UCS (MinMax старт)	66	24	900
MinMax vs UCS (UCS старт)	58	28	1483

Табела 1: Детални резултати

V Заклучок

Нашата цел беше да развиеме AI решение за играта Гомоку кое успешно ги надминува предизвиците на големиот простор на состојби и брзиот раст на дрвото на одлуки. Имплементираниот алгоритам, базиран на МинМакс со алфа-бета поткаструване и специјално дизајнирана евристика, постигна висока точност при значително намалена пресметковна сложеност. Како

што се очекуваше, нашите резултати се истакнаа како подобри во споредба со споредените алгоритми (UCS и случаен избор).

Литература

[1] Han Liao. "New Heuristic Algorithm to Improve the Minimax for Gomoku Artificial Intelligence."