Bot dla gry Toguz Korgool

Paweł Grzegorzewski

Politechnika Wrocławska
Informatyka algorytmiczna

17 Stycznia 2025

Promotor: Dr. Maciej Gębala, prof. uczelni

Cel i zakres pracy

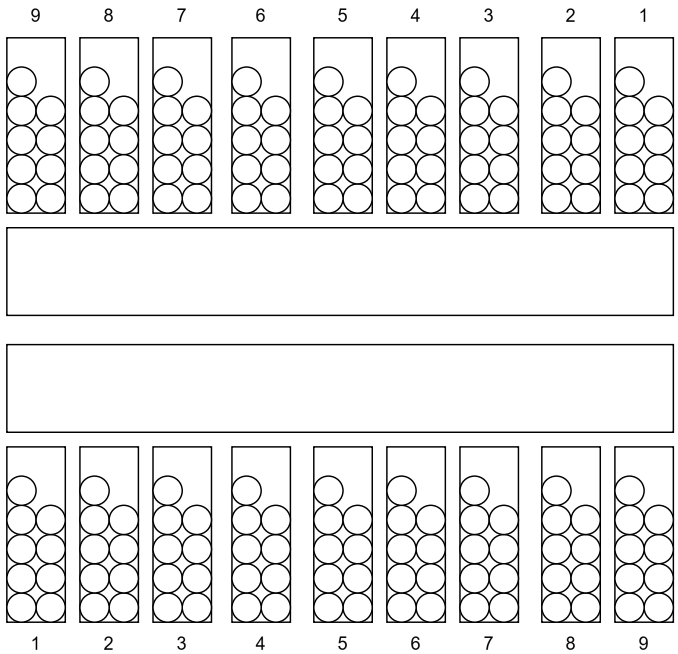
Celem pracy było stworzenie silnika grającego dla gry Toguz Korgool.

Zakres pracy obejmował implementacje dwóch różnych algorytmów oraz porównanie ich. Kluczową kwestią było również opracowanie funkcji oceny stanu planszy.

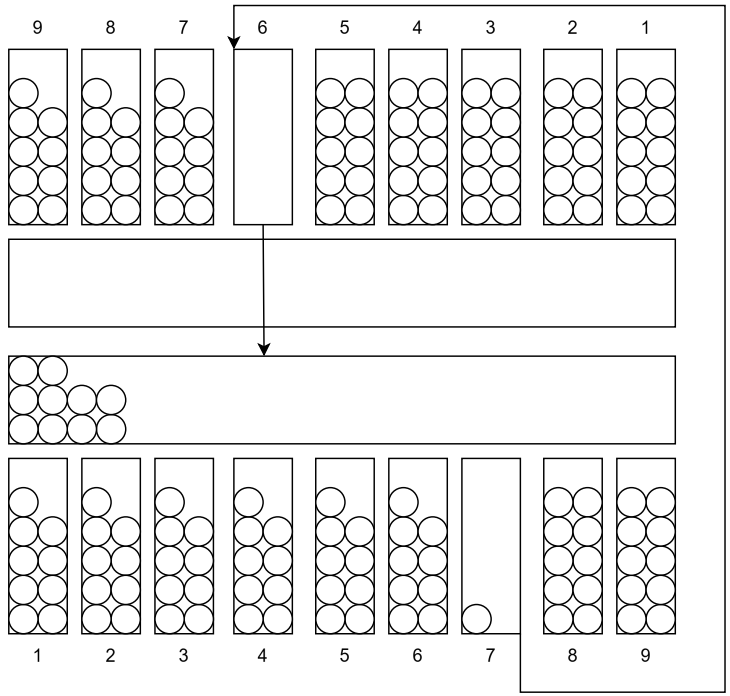
Wykorzystane narzędzia

Wszystkie funkcjonalności zostały zaprogramowane w języku C++. Zdecydowano się na wybór tego języka ze względu na jego szybkość oraz bogatą bibliotekę standardową. Wykorzystane jej elementy to m.in.: vector, array, memory, thread, future.

Zasady gry

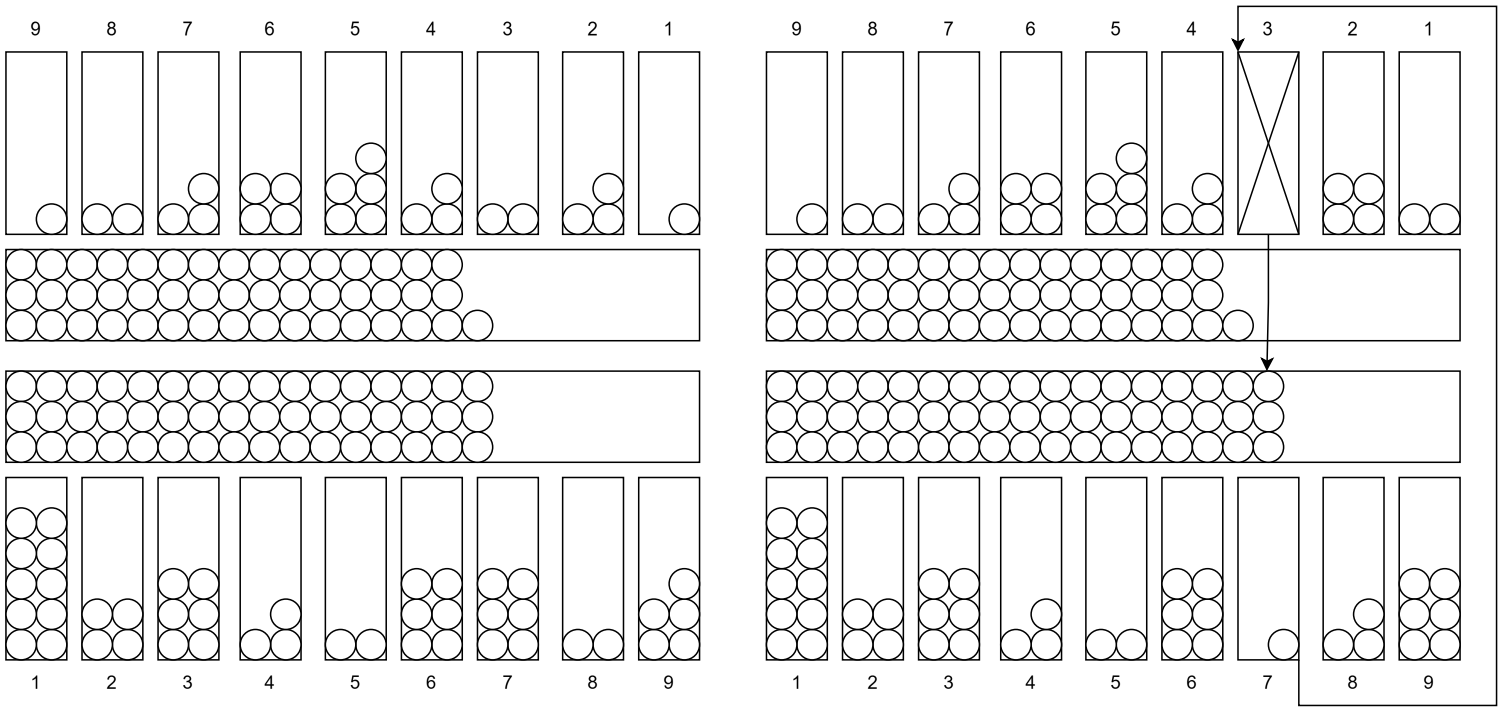


Pozycja początkowa planszy.



Przykładowy ruch zakończony zdobyciem punktów.

Tuz



Przykładowy ruch zakończony utworzeniem tuza.

Algorytm min-max

Zaimplementowano algorytm z rozwinięciem $\alpha - \beta$ cięcie. Zdecydowano się również na konstrukcję autorskiej funkcji oceny stanu oraz optymalizację jej przy pomocy algorytmu genetycznego.

Funkcja oceny stanu planszy

$$\texttt{evaluate(state)} = \sum_{i=0}^{n} w_i C_i,$$

gdzie

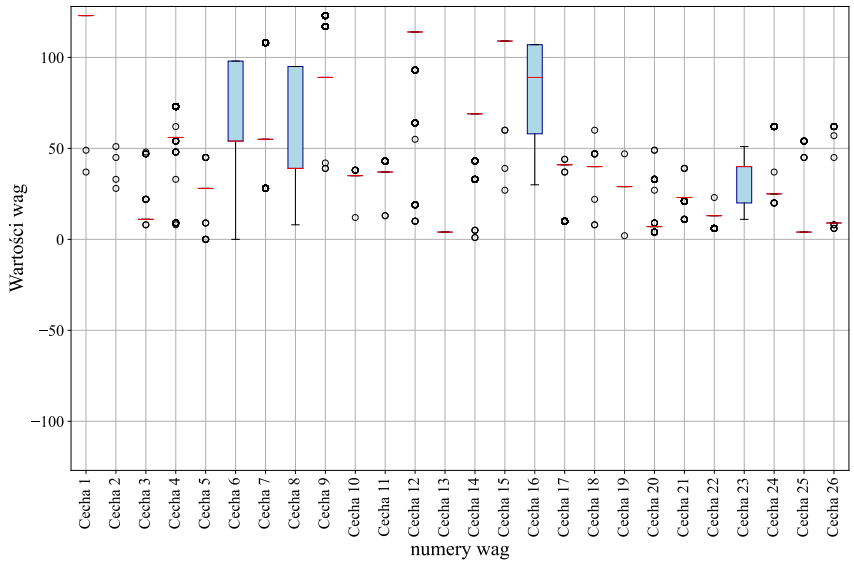
- w_i - waga cechy i,
- C_i - cecha i.

Optymalizacja funkcji oceny stanu

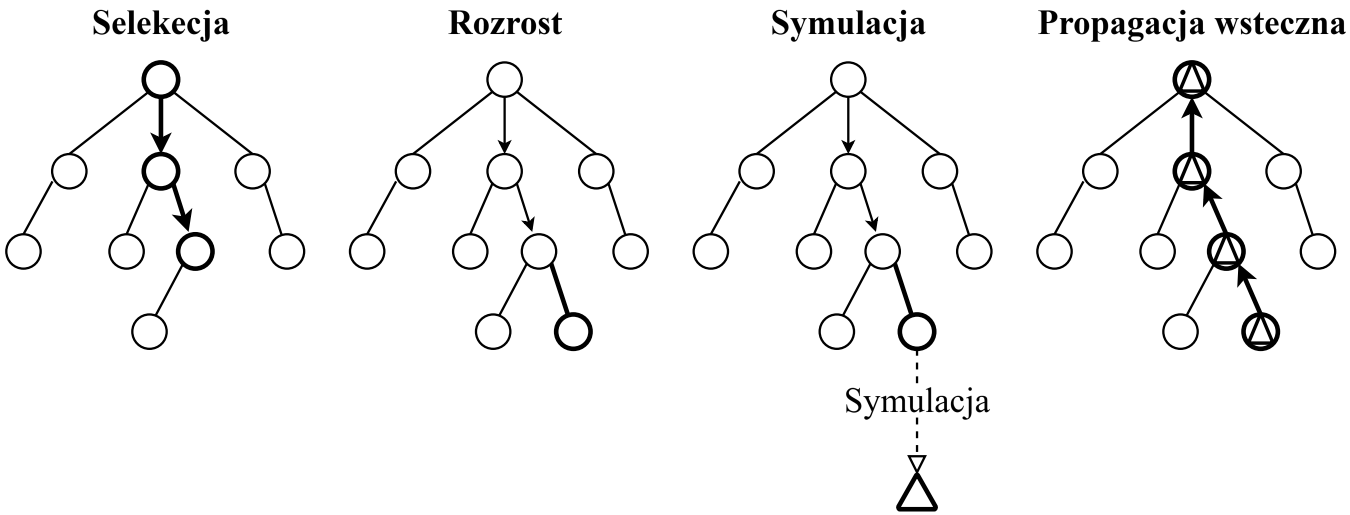
Numer cechy	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
Eksperyment 1	117	79	11	77	22	100	73	79	48	-51	-3	121	30	102	88	127	-114	22	60	44	53	56	60	-43	53	-8
Eksperyment 2	34	107	-2	47	92	64	26	125	26	63	112	48	66	42	63	119	47	11	63	6	80	80	6	12	22	79
Eksperyment 3	89	126	73	10	86	102	30	111	59	50	113	104	1	74	127	55	-2	21	12	37	27	15	17	49	14	16
Eksperyment 4	123	127	11	56	28	98	55	39	89	35	37	114	4	69	109	89	41	40	29	4	23	13	18	62	4	62

Zostało zdefiniowane 26 cech, które można podzielić na 6 grup:

- punkty zdobyte przez graczy,
- tuzy,
- potencjał punktowy tuzów,
- liczba możliwych ruchów,
- liczba korgooli w dołkach gracza,
- liczba „nieparzystych dołków”.



Monte Carlo Tree Search

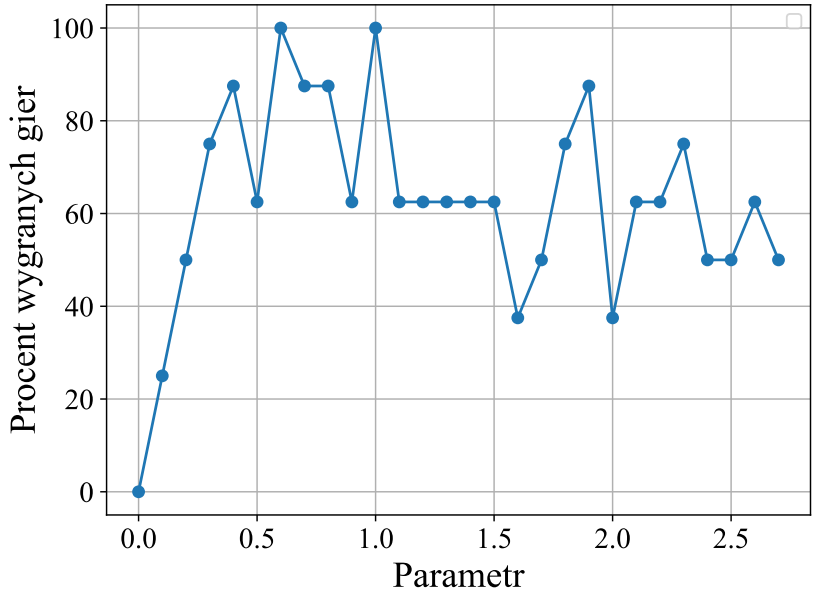


Selekcja

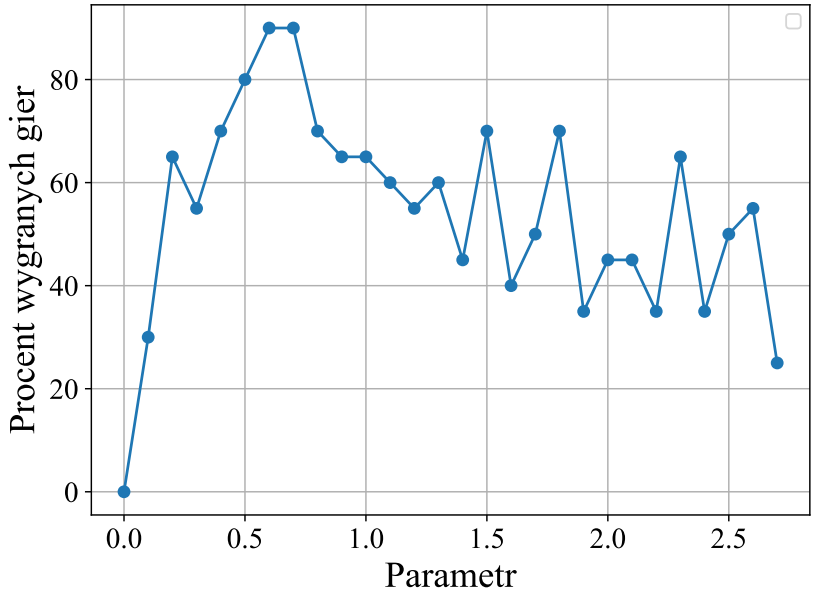
Wykorzystano do selekcji wzór UCT (z ang. Upper Confidence Bounds for Trees).

$$UCT = \frac{r_i}{n_i} + c\sqrt{\frac{\ln N}{n_i}}$$

Parametryzacja MCTS



Zależność procentu wygranych gier MCTS z min-max od parametru c. Głębokości min-max z zakresu [1,8], natomiast MCTS otrzymał 0.2 sekundy na ruch.



Zależność procentu wygranych gier MCTS z MCTS z parametrem $c = \sqrt{2}$ od parametru c. Każdy bot miał 0.2 sekundy na ruch.

Wyniki

Procent wygranych gier MCTS z 0.2 sekundy na ruch przeciwko min-max z funkcją oceny stanu bazującą tylko na punktach graczy.

Głębokość	1	2	3	4	5	6	7	8	9
MCTS jako gracz nr. 1	97%	82%	82%	74%	85%	70%	85%	81%	69%
MCTS jako gracz nr. 2	100%	60%	36%	38%	30%	29%	25%	26%	21%

Procent wygranych gier MCTS z 0.2 sekundy na ruch przeciwko min-max z autorską funkcją oceny.

Głębokość	1	2	3	4	5	6	7	8	9
MCTS jako gracz nr. 1	100%	80%	62%	28%	23%	9%	13%	22%	21%
MCTS jako gracz nr. 2	98%	48%	66%	10%	33%	7%	20%	8%	3%

Na podstawie wyników można stwierdzić, że algorytm MCTS jest lepszym wyborem w wypadku braku dobrej funkcji oceny stanu planszy.

Plany na rozwój

min-max

- implementacja ograniczeń czasowych (iterative search),
- tablica transpozycji,
- wielowątkowość,
- ulepszenie funkcji oceny stanu,
- stworzenie osobnej funkcji oceny dla gracza nr. 2.

MCTS

- wzbogacenie algorytmu o wykorzystanie funkcji oceny stanu,
- rozwiązanie problemów związanych z zaimplementowanym algorytmem wielowątkowym,
- implementacja innego rozwiązania wielowątkowego.

Dziękuję za uwagę

Paweł Grzegorzewski