Sprawozdanie

Sztuczna inteligencja i inżynieria wiedzy

Ćwiczenie 3: Algorytmy rozwiązywania gier

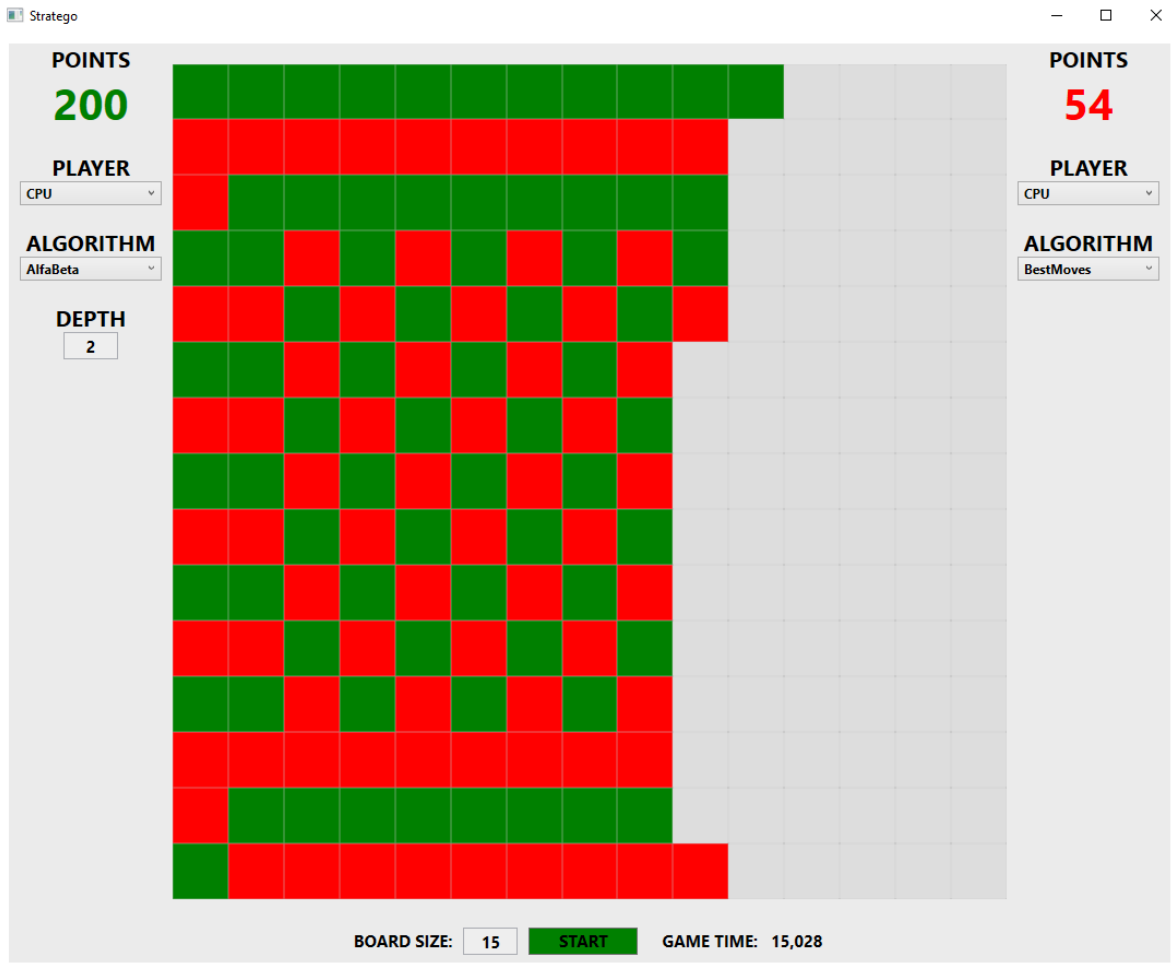
Prowadzący: Dr inż. Michał Przewoźniczek

Wykonał: Aleksander Górka

1. Wstęp

W ramach ćwiczenia została zaimplementowana gra „Stratego” wraz z algorytmami rozwiązania gier takimi jak Min-Max czy Alfa-Beta. GUI do gry zostało stworzone na silniku graficznym WPF i asynchronicznie pokazuje stan bieżącej rozgrywki między dwoma przeciwnikami. Gracz w GUI może w prosty sposób zmienić wielkość pola gry, wybierać typy poszczególnych przeciwników (człowiek lub komputer) oraz algorytmy odpowiedzialne za logikę przeciwników.

Dodatkowo w celu uproszczenia przeprowadzenia analizy została zaimplementowana aplikacja konsolowa generująca wyniki gier poszczególnych algorytmów w postaci plików o rozszerzeniu „CSV”.



Rys 1. GUI gry Stratego.

Zasady gry „Stratego”:

Polem gry jest kwadrat o dowolnej liczbie kratek. Każdy z graczy w jednej rundzie zaznacza jedną z kratek. Gracz zdobywa punkty za każdą z zamkniętych przez niego linii pionowych, poziomych lub ukośnych (kolor pól zamkniętej linii nie ma znaczenia liczy się jedynie fakt zamknięcia przez danego gracza linii). Gracz dostaje tyle punktów ile pól jest w danej zamkniętej linii lub w danych zamkniętych liniach, jeżeli jednym ruchem zamyka on więcej niż jedną linię. Minimalna długość linii za którą gracz zdobywa punkty musi być większa od 1, zatem minimalne pole gry to 2x2.

Przez zamknięcie linii jest rozumiane wypełnienie przez danego gracza ostatniego możliwego do ruchu pola w tejże linii!

Słownik:

Drzewo gry – drzewo którego węzły reprezentują kolejne stany gry.

Garbage collector – metoda automatycznego zarządzania dynamicznie przydzieloną pamięcią.

Algorytm Min-Max – metoda minimalizowania maksymalnych możliwych strat. Tworzy ona drzewo gry, w którego liściach są ocenione stany gry. Na tej podstawie wybiera najoptymalniejszy ruch w danym momencie.

Algorytm Alfa-Beta – algorytm redukujący liczbę węzłów, które muszą być rozwiązane w drzewie stworzonym przez algorytm Min-Max.

2. Zaimplementowane algorytmy i heurystyki

Algorytmy:

- **Pierwsza wolna pozycja –** komputer wykonuje ruch na pierwszą wolną pozycję. Komputer przeszukuje miejsca od góry do dołu i od lewej do prawej strony.
- **RandomSearch** (wyszukiwanie losowe) **–** komputer wykonuje ruch na losową wolną pozycję.
- **Greedy** (algorytm zachłanny) **–** komputer wykonuje pierwszy ruch który daje mu punkty. Jeżeli takowego nie ma wykonywany jest algorytm pierwsza wolna pozycja.
- **BestMoves –** komputer wykonuje ruch, który da mu najwięcej punktów w danym momencie. Jeżeli takowego nie ma wykonywany jest algorytm pierwsza wolna pozycja.
- **Min-Max –** komputer oblicza drzewo gry o podanej głębokości oraz wybiera najoptymalniejszy ruch (który zostaje oceniony przez wybraną heurystykę).
- **Alfa-Beta –** działa jak Min-Max, lecz przy okazji tworzenia drzewa redukuje on liczbę węzłów.

Ocenianie podstawowe polega na odjęciu liczby punktów zdobytych przez gracza, dla którego obliczane jest drzewo gry od liczby punktów zdobytych przez przeciwnika. Jeżeli dane pozycje ocenione zostały na tę samą liczbę punktów pierwsze są brane są od górnej lewej strony.

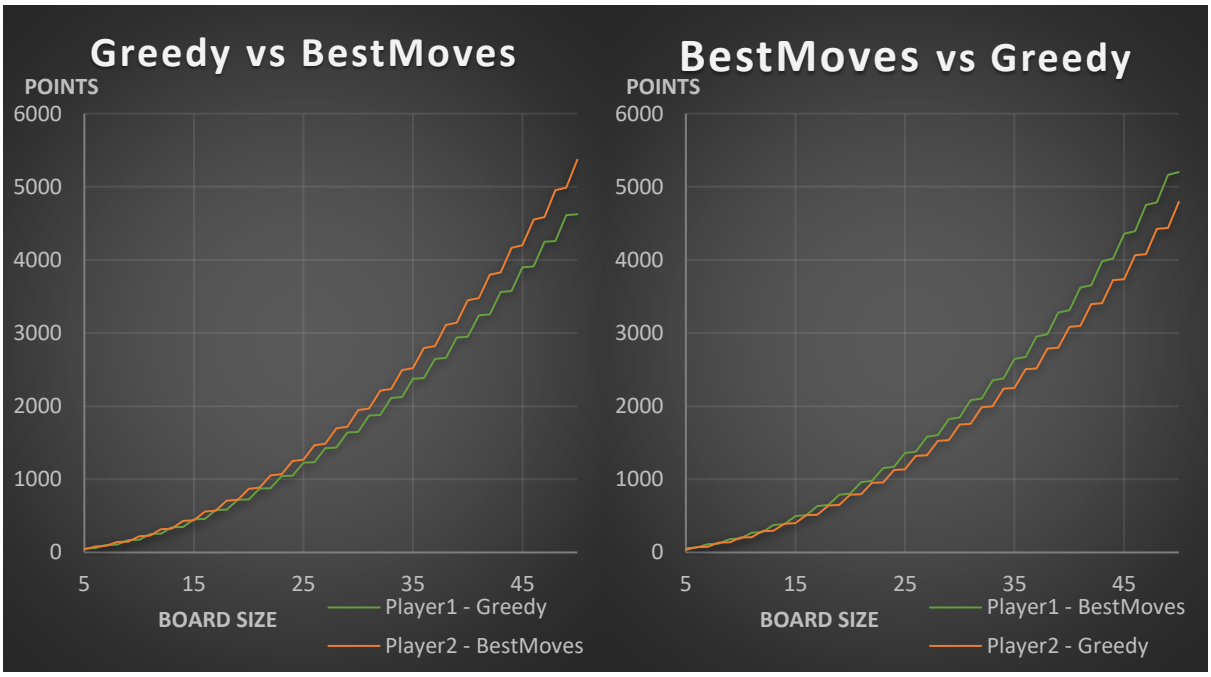
Heurystyki (dla drzewa gry) oceniające pozycję:

- **Heurystyka podstawowa:** Ocenianie podstawowe
- **Heurystyka dodatkowa:** Ocenianie podstawowe + zwiększenie liczby punktów przyznawanych za ruchy w środku planszy

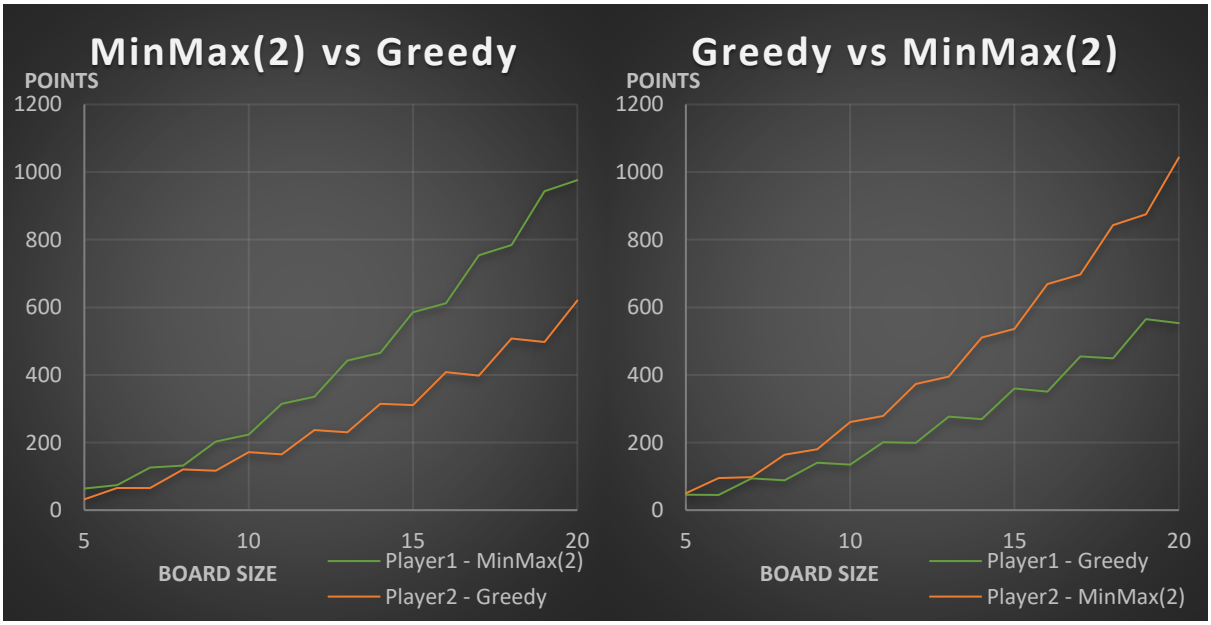
3. Testy i wyniki dla poszczególnych algorytmów

Uwagi:

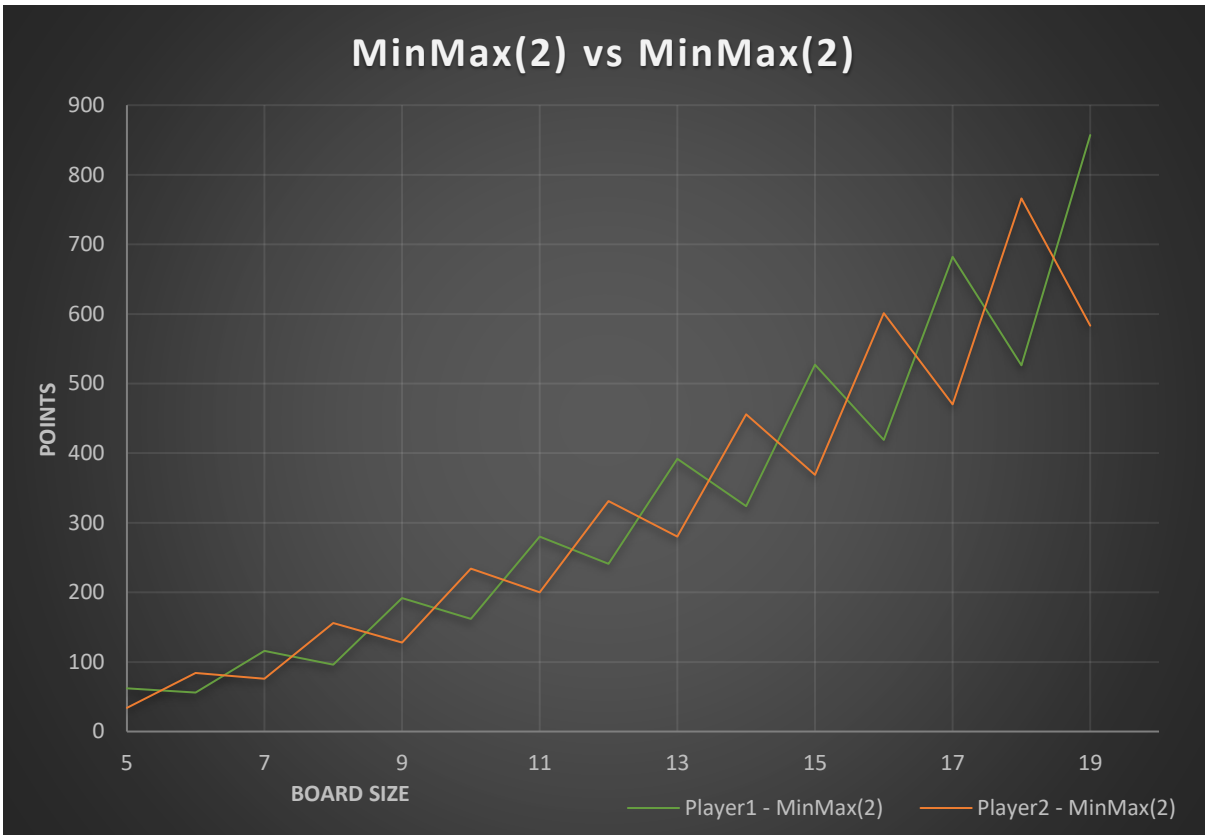
- Wynik algorytmów Min-Max i Alfa-Beta są takie same dlatego wyniki testów zostały przedstawione dla jednego z nich. Ponadto w testach nie bierze udziału algorytm pierwszej wolnej pozycji oraz algorytm wyszukiwania losowego.
- Gracz 1 zawsze rozpoczyna rozgrywkę.
- Aby sprawdzić zależność gracza rozpoczynającego od gracza wygrywającego dla poszczególnych wielkości plansz, niektóre algorytmy rozegrały ze sobą po dwie partie, aby każdy z nich raz rozpoczynał grę.



Wykres 1 i 2. Wyniki gier dla poszczególnych wielkości planszy dla algorytmów BestMoves i Greedy.



Wykres 3 i 4. Wyniki gier dla poszczególnych wielkości planszy dla algorytmów Min-Max o głębokości przeszukiwań równej 2 i Greedy.



Wykres 5. Wyniki gier dla poszczególnych wielkości planszy dla algorytmu Min-Max o głębokości przeszukiwań równej 2.

Algorytmy	Min-Max vs Min-Max		Min-Max vs Min-Max	
Głębokość przeszukiwania	**2**	**3**	**2**	**2**
Wielkość planszy	**Player1**	**Player2**	**Player1**	**Player2**
5	51	45	62	34
6	59	81	56	84
7	109	83	116	76
8	116	136	96	156
9	182	138	192	128
10	159	237	162	234
11	286	194	280	200
12	221	351	241	331
13	409	263	392	280

Tabela 1. Wyniki dla gier rozegranych z użyciem algorytmu Min-Max dla różnych wielkości planszy i głębokości przeszukań.

Wnioski:

Dla bardzo małych plansz (długość boku planszy < 5) kolejność ruchu ma ogromny wpływ na wynik, praktycznie zawsze wygrywa gracz który ma ostatni ruch. Dlatego w badaniach te wielkości plansz nie zostały wzięte pod uwagę.

Jeżeli algorytm jest znacznie lepszy od innego to wielkość planszy oraz kolejność ruchu ma znikomy wpływ na wynik, a im większa plansza, tym różnica pomiędzy poziomami gry algorytmów jest bardziej widoczna.

Zwiększenie głębokości przeszukań znacznie podnosi ilość mocy obliczeniowej potrzebnej do obliczenia drzewa gry. Z tabeli 1 wynika, że zwiększona głębokość dla heurystyki podstawowej ma znikomy wpływ na wynik, kompletnie niewspółmierny do potrzebnego nakładu mocy obliczeniowej.

Gdy algorytmy mają bardzo podobny/takie sam poziom zaawansowania gry i wielkość planszy jest odpowiednio duża to kluczowym czynnikiem jest to, który z gracz posiada ostatni ruch, ponieważ zawsze zamyka on 4 linie i jest najlepiej punktowany.

Wielkość planszy	Player1 (Min-Max 2)	Player2 (Min-Max 2)	Ostatni ruch	Wygrany
5	62	34	Player1	Player1
6	56	84	Player2	Player2
7	116	76	Player1	Player1
8	96	156	Player2	Player2
9	192	128	Player1	Player1
10	162	234	Player2	Player2
11	280	200	Player1	Player1
12	241	331	Player2	Player2
13	392	280	Player1	Player1
14	324	456	Player2	Player2
15	527	369	Player1	Player1

Tabela 2. Przedstawia wyniki gier dla poszczególnych wielkości planszy dla algorytmu Min-Max o głębokości przeszukiwań równej 2.

4. Porównanie wydajności Alfa-Beta i Min-Max

Algorytmy	Alfa-Beta	Min-Max	Stosunek wydajności do Alfa-Beta do Min-Maxa [%]
Głębokość	**Player1 Czas [ms]**	**Player2 Czas [ms]**	
2	15	18	120,00
3	348	546	156,86
4	5975	13585	227,36
5	129903	352567	271,40

Tabela 3. Porównanie wydajności Alfa-Beta i Min-Max dla planszy o wielkości 6.

Wnioski:

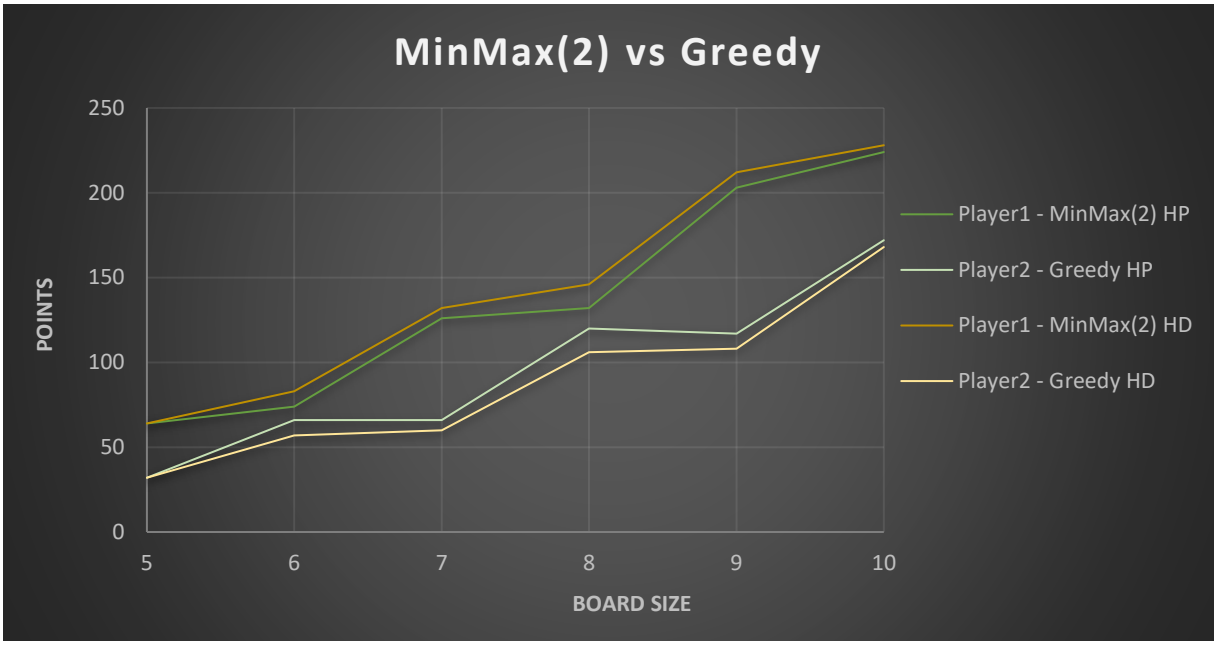
Wydajność algorytmy Alfa-Beta skaluje się wraz ze wzrostem głębokości przeszukań. Z każdym poziomem głębokości różnica w wydajności zwiększa się o około 50%.

5. Testy i wyniki dla heurystyk oceniających pozycję

Testy zostały przeprowadzone z udziałem algorytmu Alfa-Beta o głębokości przeszukiwań 2 z wykorzystaniem obu heurystyki (podstawowa kontra dodatkowa). Jednak w tym wypadku

wyniki były dokładnie takie same jak w przypadku użycia tylko jednej heurystyki (podstawowa kontra podstawowa).

Różnica pomiędzy heurystykami można zaobserwować w starciu z algorytmem Greedy. Korzystniej w tym starciu wypada heurystyka dodatkowa.



Wykres 6. Porównanie heurystyk (HP – Heurystyka podstawowa, HD – Heurystyka dodatkowa).

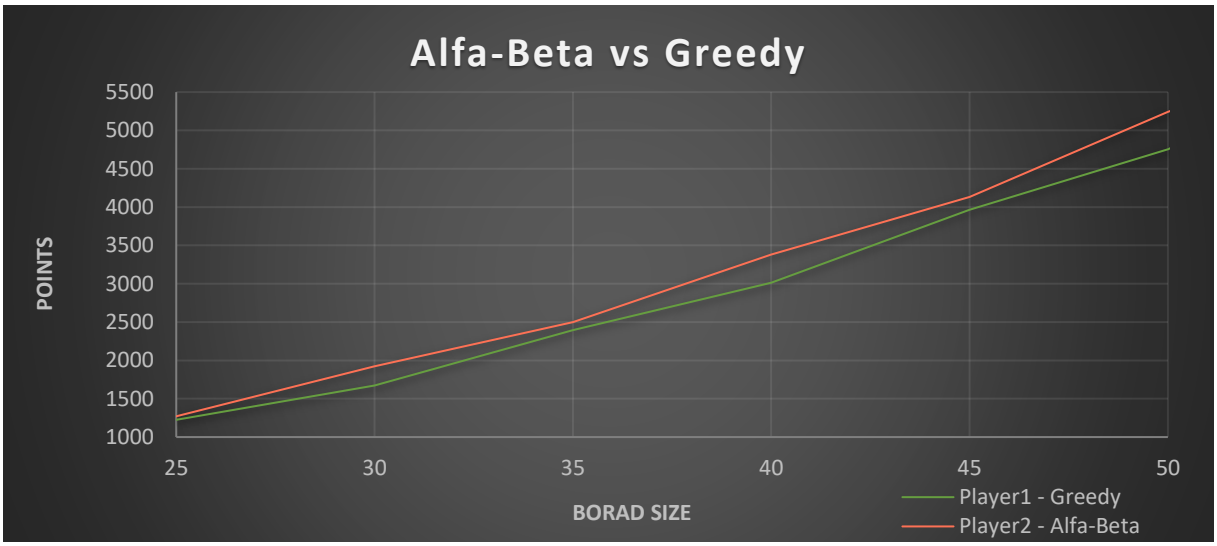
6. Optymalizacja

I. Optymalizacja danych w węzłach drzewa gry

Węzłami drzewa tworzonego w momencie obliczania najoptymalniejszego ruchu dla algorytmów Min-Max i Alfa-Beta były kolejne nowo uruchomione gry. Wiązało się to z alokowaniem dużej ilości pamięci oraz częstym użyciem garbage collectora. Pierwszym sposobem na zwiększenie wydajności było zastąpienie gier w węzłach drzewa kolejnymi ruchami graczy. Zatem obecnie obliczenia drzewa gry są wykonywane na jednej planszy, a nie na wielu, co pozwoliło na zwiększenie szybkości działania dla dużych głębokości przeszukań o ponad 30%.

II. Dynamiczna regulacja głębokości przeszukań

W początkowych fazach gry gracz ma bardzo wiele możliwości ruchu, zatem do stworzenia drzewa gry o dużej głębokości potrzebna jest znaczna ilość mocy obliczeniowej. W grze „Stratego” początkowe ruch nie mają jednak tak dużego znaczenia jak końcowe. Można zatem pozwolić sobie na zmniejszenie głębokości przeszukań w początkowych fazach gry i stopniowym jej zwiększaniu w późniejszych fazach (np. gdy liczba dostępnych pół ruchu jest odpowiednio mała). Nie powinno mieć to znacznego wpływu na wynik rozgrywki, a zabieg ten umożliwi działanie algorytmu na planszach o wielkości powyżej 20.



Wykres 7. Porównanie algorytmu Greedy z algorytmem Alfa-Beta z zaimplementowaną dynamiczną regulacją głębokości.

III. Wprowadzenie heurystyki „sztucznie" zwiększającej głębokość przeszukiwań drzewa gry

Polega na pobraniu liczby punktów jaką może zdobyć przeciwnik za najlepiej punktowany ruch i odjęciu tej liczby od wyniku obliczonego dla danego ruchu. Dzięki takiemu zabiegowi głębokość drzewa gry jest zwiększana o jeden, lecz moc obliczeniowa potrzebna do tego zabiegu jest mniejsza.

Po przeprowadzonych testach można stwierdzić, że wyniki dla algorytmu Alfa-Beta o głębokości 2 oraz 3 były dokładnie takie same jak dla głębokości 1 oraz 2 wykorzystujących heurystykę sztucznie zwiększającą głębokość. Wydajność jednak zwiększyła się o około 20% co prezentuje poniższy wykres.

Wykres 8. Porównanie zwiększenia wydajności przy użyciu HSZG (HSZG – heurystyka sztucznie zwiększająca głębokość).