Zaawansowana sztuczna inteligencja do Scrabble. Podstawy teoretyczne i opis implementacji algorytmu.

Jakub Turek
Wydział Elektroniki i Technik Informacyjnych
J.Turek@stud.elka.pw.edu.pl

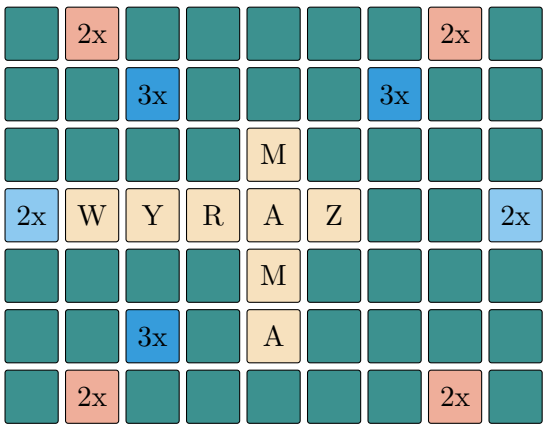
Streszczenie *Celem artykułu jest opisanie zbioru koncepcji, które posłużą do implementacji algorytmu sztucznej inteligencji grającego w grę Scrabble w języku polskim. Artykuł prezentuje zbiór teoretycznych informacji o optymalnej strategii gry w Scrabble. Ponadto zostały w nim przedstawione elementy algorytmów zastosowanych w aplikacjach Maven oraz Quackle. Autor wskazuje ich potencjalne słabe punkty i sposoby usprawnienia, a także prezentuje wyniki implementacji proponowanych poprawek.*

1. Wstęp Scrabble to „gra słowna polegająca na układaniu na określonej planszy wyrazów z losowanych liter". [1] Jest to bardzo ogólna definicja, którą należy uściślić. Scrabble jest grą przeznaczoną dla 2-4 osób. Akcesoriami do gry są: kwadratowa plansza o stałym rozmiarze 15×15, torebka wypełniona płytkami, na których nadrukowane są litery oraz ich wartości punktowe, a także stojaki, na których gracze umieszczają płytki, którymi w danej chwili dysponują.

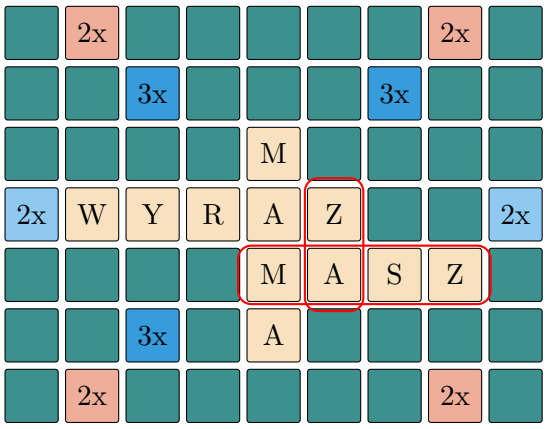
Gra rozgrywana jest w turach. Zadaniem graczy jest układanie wyrazów na planszy w taki sposób, aby tworzyły one poprawne słowa w języku, w którym prowadzona jest rozgrywka, w układzie krzyżówkowym. Układ krzyżówkowy został przedstawiony na rysunkach 1 oraz 2:

Rysunek 1 Pokazuje sytuację początkową obrazującą pewien moment rozgrywki.

Rysunek 2 Pokazuje poprawny ruch zawodnika, który powoduje powstanie więcej niż jednego słowa. Wszystkie wyrazy utworzone przez jeden ruch muszą być poprawne. W podanym przykładzie słowa „za" i „masz" są poprawne.



Rysunek 1: Fragment planszy. Gracze ułożyli kolejno słowa: „wyraz" oraz „mama".



Rysunek 2: Fragment planszy. Ruch przez dołożenie liter A, S, Z tworzy dwa wyrazy w układzie krzyżówkowym.

W trakcie jednego ruchu zawodnik może

układać płytki tylko w jednym kierunku (pionowo lub poziomo). Utworzony przez zawodnika wyraz musi być spójny, to znaczy, że wszystkie płytki muszą przylegać do siebie bezpośrednio lub poprzez płytki już istniejące na planszy. Wymagane jest, aby tworzone słowo przylegało do przynajmniej jednej płytki, która jest już umieszczona na planszy (nie dotyczy to pierwszego ruchu).

Punktacja za dane zagranie jest obliczana jako suma punktów za wszystkie płytki, które wchodzą w skład utworzonych wyrazów (a więc również tych, które przed zagraniem znajdowały się na planszy), z uwzględnieniem niewykorzystanych premii wynikających z pozycji płytki na planszy:

Premia literowa podwaja lub potraja wartość danej płytki.

Premia słowna podwaja lub potraja wartość całego wyrazu.

2. Strategia optymalna Kluczowym zagadnieniem podczas implementacji algorytmu sztucznej inteligencji do gier planszowych (w tym Scrabble) jest zbadanie istnienia strategii optymalnej.

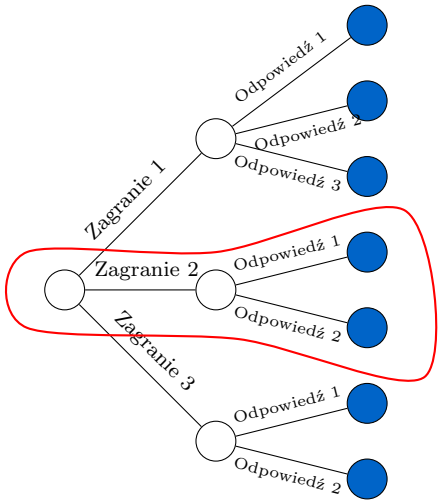
Twierdzenie 1. Istnieje optymalna strategia gry w Scrabble.

Aby dowieść twierdzenie 1 należy uprzednio zdefiniować przestrzeń stanów w Scrabble.

Definicja 1. Stan rozgrywki po danej turze opisują jednoznacznie rozmieszczenie klocków na planszy P oraz zagranie Z. Przejście między stanami determinuje zmiana $(\Delta P, \Delta Z)$.

Dysponując definicją 1 można dowieść twierdzenie 1. Ilustrację dla dowodu 1 przedstawia rysunek 3.

Dowód 1. Dla ułatwienia analiza rozgrywki zostanie przeprowadzona wychodząc od stanu końcowego. Nie wpływa to na ogólność dowodu. Dla dowolnej rozgrywki prawdziwe jest twierdzenie, że do stanu końcowego można wejść tylko na skończoną liczbę sposobów, które reprezentują legalne zagrania. Przeprowadzając rozumowanie iteracyjne możliwe jest przejście do stanu początkowego. Należy zauważyć, że w każdej iteracji analizowana liczba wejść do stanu jest zawsze skończona, ponieważ skończona jest liczba kombinacji, na które można wylosować litery oraz liczba zagrań legalnych dla danej kombinacji liter. Podobne rozumowanie można przeprowadzić dla każdego stanu końcowego budując skończone drzewo przestrzeni stanów. Skoro drzewo przestrzeni stanów jest skończone, można na podstawie jego znajomości w każdym zagraniu wyznaczyć ruch, który maksymalizuje prawdopodobieństwo wygranej.



Rysunek 3: Ilustracja dowodu istnienia strategii optymalnej. Stany końcowe oznaczono kolorem niebieskim.

Zostało udowodnione, że deterministyczna wersja Scrabble dla dwóch graczy należy do problemów klasy *PSPACE-Complete*. [3] Klasa *PSPACE-Complete* jest najbardziej obszerną klasą problemów wielomianowych. Każdy problem należący do przestrzeni problemów wielomianowych można przekształcić, w wielomianowym czasie, do problemu klasy *PSPACE-Complete*.

3. Strategia zależna od fazy gry Niestety, w praktyce wykorzystywanie strategii optymalnej nie jest możliwe. Dla rozgrywki końcowej 7 klocków przeciwko 7 klockom średnia ilość gałęzi w drzewie przestrzeni stanów wynosi 200. Kolejnym problemem jest głębokość drzewa, która może sięgać czternastu po-

ziomów (gdy gracze wykładają klocki pojedynczo) lub więcej (dochodzi możliwość pasowania). Wartości te są zbyt duże aby stosować pełny algorytm typu $\alpha - \beta$, a odnoszą się wyłącznie do sytuacji, w której posiadamy pełną informację o rozgrywce (to znaczy znane są klocki przeciwnika). Na pozostałych etapach rozgrywki analiza przestrzeni stanów nie jest opcją. Wynika z tego, że algorytm sztucznej inteligencji do Scrabble musi zmieniać strategię rozgrywki w zależności od etapu gry.

Wyróżnia się cztery zasadnicze fazy rozgrywki: [4]

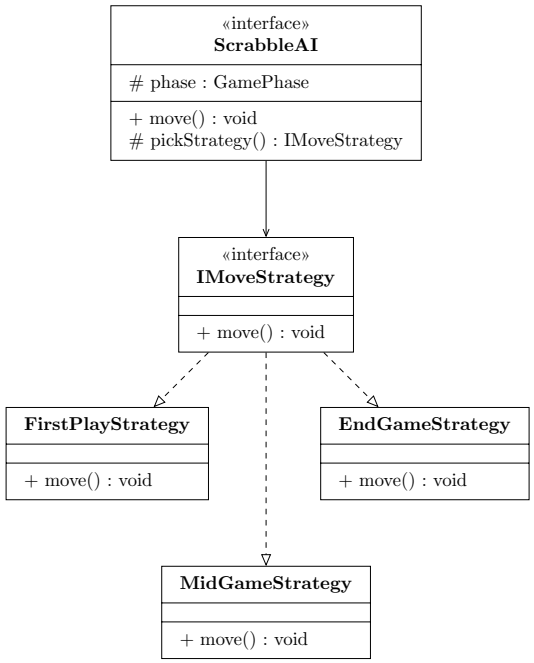
Opening Play[1] faza otwarcia. Dotyczy wyłącznie pierwszego zagrania, a więc kiedy na planszy nie znajduje się jeszcze żaden klocek.

Mid Game faza śródgry. Rozpoczyna się po otwarciu, a kończy wraz z rozpoczęciem fazy PEG.

Pre-End Game faza przedkońcowa rozgrywki. Rozpoczyna się w momencie, w którym rozgrywka może zakończyć się w dwóch turach, a więc kiedy w worku pozostaje siedem lub mniej klocków. Kończy się wraz z rozpoczęciem fazy EG.

End Game faza zakończenia. Rozpoczyna się, gdy w worku nie ma już klocków i trwa aż do końca gry. Charakteryzuje się tym, że w trakcie jej trwania zawodnicy dysponują pełną informacją o grze, a więc również o literach posiadanych przez przeciwnika.

Wykonywany algorytm będzie różny w zależności od fazy rozgrywki. Podczas implementacji takiego rozwiązania warto wykorzystać wzorzec projektowy Strategia. Strategia to wzorzec, który „definiuje rodzinę algorytmów, pakuje je jako osobne klasy i powoduje, że są one w pełni wymienne. Zastosowanie strategii pozwala na to, aby zmiany w implementacji przetwarzania były całkowicie niezależne od strony klienta, który z nich korzysta". [5] Wzorzec strategii dostosowany do opisywanej dziedziny problemu został schematycznie przedstawiony na rysunku 4.



Rysunek 4: Przykład użycia wzorca projektowego Strategia w kontekście implementacji algorytmu.

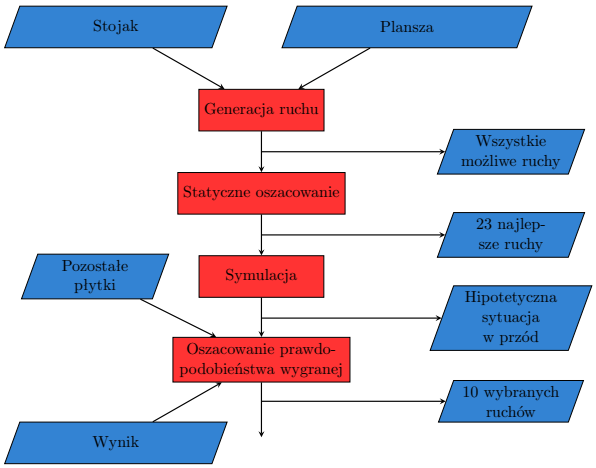
4. Opening Play Celem fazy OP jest wybranie najwyżej punktowanego zagrania. W algorytmie Quackle wybór ten jest dokonywany poprzez przeszukiwanie słownika pod kątem posiadanych liter. Wynik punktowy uzyskany za zagranie jest obliczany dynamicznie.

Wyszukiwanie najlepszego otwarcia może być zoptymalizowane czasowo. Wykorzystując znajomość słownika można przeprowadzić analizę najlepszych otwarć dla wszystkich możliwych kombinacji klocków. Każdej kombinacji liter można przypisać prostą funkcję skrótu, która sortuje te litery alfabetycznie. Po wstępnej analizie słownika można przygotować mapę, która każdej funkcji skrótu przyporządkuje najlepsze zagranie oraz jego wartość punktową. Dzięki temu możliwe jest wykonanie pierwszego zagrania w czasie jednost-

[1] Opening Play, w skrócie OP. W dalszej części artykułu dla określenia faz rozgrywki stosowane będą wyłącznie nazwy skrótowe, a więc OP, MG, PEG oraz EG.

kowym[2].

5. Mid Game W fazie MG w aplikacji Quackle wykorzystywany jest algorytm z heurystyczną funkcją oszacowania oraz symulacją dwóch zagrań „w przód". Schemat tego algorytmu został przedstawiony na rysunku 5. [6]



Rysunek 5: Schemat algorytmu dla fazy MG.

Algorytm składa się z czterech kroków:

1. Na podstawie stanu planszy i dostępnych klocków generowane są wszystkie możliwości ruchu.
2. Dla każdej możliwości ruchu obliczana jest funkcja oszacowania. Ruchy są sortowane względem malejącej wartości funkcji oszacowania. Lista ruchów jest skracana do najlepszych 23 pozycji.
3. Dla każdego z pozostałych ruchów wykonywana jest symulacja stanu rozgrywki na dwa kroki w przód.
4. Dla każdego rezultatu oszacowania, biorąc pod uwagę liczbę płytek pozostałych do końca rozgrywki, obliczane jest prawdopodobieństwo wygranej. W rezultacie powstaje lista 10 ruchów o największym oszacowanym prawdopodobieństwie wygranej.

Na potrzeby artykułu pominięte zostanie omówienie fazy generacji ruchu. Jej wynikiem jest lista wszystkich poprawnych ruchów dla określonej kombinacji liter i ułożenia klocków na planszy.

5.1. Statyczne oszacowanie Do statycznego oszacowania wykorzystywana jest funkcja celu postaci $F(x) = P(x)+LV(x)$, gdzie $F(x)$ to wartość oszacowania dla zagrania x, $P(x)$ to wartość punktowa tego zagrania, a $LV(x)$ to współczynnik *leave value* dla klocków pozostałych po zagraniu.

Definicja 2. *Leave value* to funkcja, która zestawowi klocków pozostałych po zagraniu przypisuje wartość punktową odzwierciedlającą „perspektywiczność" pod kątem wysoko punktowanych zagrań w przyszłości. Funkcja przyjmuje wartości należące do zbioru $\mathbb{R}$ liczb rzeczywistych.

Przykładowo jeśli płytki pozostałe po ułożeniu czteroliterowego wyrazu to **Ą**, **Ź** oraz **Ń**, wtedy ich wartość *leave value* będzie ujemna, gdyż prawdopodobieństwo siedmioliterowego zagrania w kolejnym ruchu jest bardzo niskie. Wartości funkcji *leave value* wyznaczane są na podstawie bazy danych wcześniejszych gier, którą można zbudować wykorzystując zapisy partii z turniejów.

Statyczne oszacowanie przeprowadza się dla wszystkich dozwolonych ruchów wyznaczonych na etapie generacji. Do kolejnego etapu wybierane są 23 ruchy z najwyższą wartością funkcji celu.

5.2. Symulacja W fazie symulacji dla każdego z pozostałych 23 ruchów szacuje się jaki może być ich skutek na dwa zagrania w przód. Symulacja przeprowadzana jest według poniższego algorytmu:

1. Gracz G_1 wykonuje zagranie P_1.
2. Ze zbioru pozostałych płytek wybierana jest losowa zawartość stojaka gracza G_2.
3. Generowane są wszystkie możliwości ruchu dla gracza G_2.
4. Gracz G_2 wykonuje zagranie P_2 z najwyższą wartością statycznego oszacowania.
5. Gracz G_1 uzupełnia płytki.

[2]Przy założeniu, że wśród liter początkowych nie znajdują się blanki.

6. Generowane są wszystkie możliwości ruchu dla gracza G_1.

7. Gracz G_1 wykonuje zagranie P_3 z najwyższą wartością statycznego oszacowania.

8. Dla zagrania P_1 obliczana jest wartość symulacji PV_1. Jest to liczba punktów G_1 po zagraniu P_3 pomniejszona o liczbę punktów G_2 po zagraniu P_2.

9. Wartość PV_1 jest powiększana o *leave value* płytek gracza G_1 po zagraniu P_3.

5.3. Szacowanie prawdopodobieństwa wygranej Dla każdego z 23 wyników symulacji szacowane jest następnie prawdopodobieństwo wygranej. Prawdopodobieństwo wygranej jest funkcją $W : PV, TR \rightarrow [0; 1]$, gdzie PV jest wartością zagrania wyznaczoną w trakcie symulacji, a TR jest liczbą klocków pozostałych do wykorzystania w partii.

Wartości funkcji W, podobnie jak wartości funkcji celu dla fazy statycznego oszacowania, są wyznaczane na podstawie bazy danych gier. W wyniku etapu oszacowania prawdopodobieństwa wygranej pozostawiane jest 10 zagrań o największym prawdopodobieństwie wygranej.

5.4. Usprawnienia fazy mid-game Na etapie symulacji pomijane są dwa istotne aspekty rozgrywki:

- Losowanie zawartości stojaka przeciwnika odbywa się z rozkładem jednostajnym. Obserwując poprzedni ruch przeciwnika można lepiej przybliżyć ten rozkład.

- Gracze korzystają z *łowienia* korzystnych wyrazów.

Jeżeli w poprzednim ruchu przeciwnik ułożył na planszy słowo **RADO** można wnioskować, że nie posiadał on liter **I** oraz **U**. W przeciwnym razie utworzyłby wyżej punktowane słowa - **RADIO** lub **URODA**. Korzystając z tej obserwacji możemy wylosować $\frac{3}{7}$ zawartości stojaka przeciwnika ze zbioru pozostałych klocków z pominięciem liter **U** oraz **I**.

Definicja 3. *Łowienie* w grze Scrabble to umyślny brak wykorzystania większości klocków do wykonania zagrania ze względu na duże prawdopodobieństwo wylosowania układu pozwalającego na wysoko punktowane zagranie w kolejnym ruchu.

Niestety, w praktyce trudno określić kryteria, które pozwalają rozpoznać *łowienie* przeciwnika. Umiarkowanie dobrym kryterium jest analiza ilości liter użytych do poprzedniego zagrania. Jeżeli przeciwnik wykorzystał tylko jedną literę bardzo prawdopodobne jest, że próbował *złowić* dobrą kombinację liter. Aby przeciwdziałać *łowieniu* należy wybrać specjalną estymatę zawartości stojaka przeciwnika na etapie symulacji. Dzięki temu symulacja preferować będzie zagrania blokujące *hot-spoty*, czyli otwarte punkty planszy umożliwiające wysoko punktowaną rozgrywkę.

Kolejnym usprawnieniem algorytmu może być wprowadzenie rozróżnienia na zagrania ofensywne oraz defensywne. Etap oszacowania prawdopodobieństwa wygranej bazuje na nieprecyzyjnych informacjach wejściowych. Zasymulowana pozycja „w przyszłości" może być nieadekwatna do rzeczywistości, na przykład gdy zostało pominięte możliwe *bingo* przeciwnika. Wprowadzając podział zagrań na defensywne i ofensywne możliwe jest wprowadzenie dodatkowego elementu algorytmu rozważającego opłacalność zagrania otwierającego planszę. Do rozważania opłacalności można wykorzystać sieć neuronową.

Podział zagrań na defensywne i ofensywne można przeprowadzić na podstawie wielu kryteriów. Pod uwagę można brać między innymi liczbę wykorzystanych podczas zagrania klocków, które znajdowały się na planszy przed rozegraniem. Innym przykładem takiego kryterium jest ilość nowych premii otwartych w wyniku zagrania.

6. End Game Algorytm operujący w fazie EG dysponuje pełną informacją o grze, gdyż wiadomo jakimi literami dysponuje przeciwnik. W tych warunkach naturalne jest wykorzystanie przeszukiwania przestrzeni stanów algorytmem typu $\alpha - \beta$. Niestety, nawet w końcowych etapach rozgrywki przestrzeń stanów jest zbyt duża i zbyt głęboka

aby wykonać takie przeszukiwane. Niezbędne jest wyznaczenie ograniczeń.

Do wyznaczania ograniczeń można wykorzystać programowanie dynamiczne. Na potrzeby oszacowania przyjęte zostanie założenie, że sytuacja na planszy jest statyczna i wartość możliwych do uzyskania zagrań zależy jedynie od stanu planszy i zawartości stojaka, a nie superpozycji kolejnych zagrań obu graczy. Zakładając, że partia zakończy się w N ruchach można ograniczyć przeszukiwaną przestrzeń tylko do N poziomów. Każdemu zagraniu zostanie przypisana jego szacowana wartość w postaci $F_N(x) = P_N(x) + LV_{N-1}$. P_N oznacza liczbę punktów uzyskanych za zagranie, a LV_{N-1} to oszacowanie wartości pozostałych po zagraniu płytek przy założeniu, że do końca rozgrywki pozostało $N-1$ tur.

Przykład oszacowania dla fazy EG:

1. Gracz 1 (G_1) zakończy grę w 8 ruchach. Wtedy gracz 2 (G_2) rozegra 7 ruchów. Powstaje ścieżka do przetworzenia algorytmem minimax.

2. G_2 zakończy grę w 7 ruchach. G_1 rozegra wtedy 7 ruchów. Jeżeli ta ścieżka będzie lepsza dla G_2, $G2$ wybierze właśnie ją.

3. G_1 będzie miał okazję do poprawy jeżeli zakończy grę w 7 ruchach...

Algorytm ten jest kontynuowany do momentu, gdy żaden z graczy nie może się poprawić.

Analiza przestrzeni stanów może być dalej usprawniana. Nie zawsze istnieje bowiem jedna optymalna ścieżka zagrania dla przeciwnika. Przykładowo, gdy może on zagrać słowo **ALE** w dwóch różnych pozycjach (za 25 i 28 punktów odpowiednio) korzyść z zablokowania drugiej na rzecz pierwszej pozycji wynosi tylko trzy punkty. Spostrzeżenie to można zawrzeć w algorytmie przez wprowadzenie dwóch progów oszacowania: optymistycznego i pesymistycznego. Niestety, algorytm minimax nie wspiera przedziałów. Do przeszukiwania przestrzeni stanów należy zatem wykorzystać inny algorytm. Przykładem przeszukiwania wspierającego przedziały jest algorytm B^*. [7]

7. Pre-end Game W fazie PEG obliczenia wykonywane przez algorytm dla fazy MG pozostaną poprawne. Pod uwagę należy wziąć również scenariusz końca gry, gdyż wraz z rozpoczęciem fazy PEG możliwe jest zakończenie rozgrywki w dwóch turach. Rozwiązaniem jest wykorzystywanie hybrydowego podejścia, które najpierw wyznacza najkorzystniejsze ruchy algorytmem dla fazy MG, a potem dokonuje dla nich ograniczonego przeszukiwania przestrzeni stanów.

8. Wyniki testów Na potrzeby testowania usprawnień algorytmów przedstawionych w artykule zostały wykonane dwie implementacje sztucznej inteligencji do Scrabble. Jedna, nazywana dalej algorytmem referencyjnym, bazuje bezpośrednio na specyfikacji aplikacji Quackle. [6] Druga, tak zwany algorytm zmodyfikowany, nanosi do algorytmu referencyjnego wszystkie modyfikacje opisane w artykule. Oba algorytmy rozegrały przeciwko sobie 1523 partie. Podsumowanie wyników zostało zamieszczone w tabeli 6.

	Algorytm zmodyfikowany	Algorytm referencyjny
Wygrane	987	536
% wygranych	**64,8%**	35,2%

Rysunek 6: Wyniki testów modyfikacji algorytmu referencyjnego.

Rozkład procentowy wygranych znacząco odbiega od rozkładu $50\%-50\%$, którego można byłoby oczekiwać dla równorzędnych przeciwników. Na podstawie przedstawionej próby można wnioskować, że modyfikacje wprowadzone do algorytmu przyczyniły się do zwiększenia jego skuteczności.

Ze względu na brak odpowiedniej klasy przeciwników algorytm nie został należycie przetestowany przeciwko ludziom. Okazało się, że przewaga, którą daje znajomość całego słownika jest zbyt duża, aby gracze-amatorzy mogli wygrać chociaż jedną partię z algorytmem. W dotychczasowych testach zmodyfikowany algorytm odniósł 100% zwycięstw.

9. Podsumowanie Artykuł prezentuje informacje, które mogą posłużyć do implementacji zaawansowanego algorytmu sztucz-

nej inteligencji grającego w Scrabble. Artykuł przedstawia podstawy teoretyczne niezbędne do jego implementacji. Ponadto opisuje szczegółowo strategie, które mogą być stosowane na różnym stopniu zaawansowania rozgrywki.

Artykuł koncentruje się na wskazaniu słabych ogniw algorytmów dla poszczególnych etapów rozgrywki. Autor prezentuje modyfikacje, które należy wprowadzić celem eliminacji omawianych wad. Zaprezentowane w artykule wyniki testów pozwalają sądzić, że wprowadzone modyfikacje pozytywnie wpływają na skuteczność całego algorytmu.

Dalszym kierunkiem rozwoju algorytmu może być badanie sposobu na wykrycie próby *złowienia* dobrych liter przez przeciwnika. Testy algorytmu referencyjnego wskazują, że wiele przegranych wynika z powszechnego wykorzystywania *łowienia* przez zawodników klasy mistrzowskiej. Zaproponowany w artykule sposób wykrywania tego typu zagrań nie jest wyczerpujący i może być przedmiotem dalszych usprawnień. Dalszemu badaniu może podlegać również efektywność wprowadzonych modyfikacji dla innych języków rozgrywki.

Literatura

[1] *Wielki słownik ortograficzny PWN*, pod red. Edwarda Polańskiego, Wyd. 3 popr. i uzup., Warszawa, Wydawnictwo Naukowe PWN, 2012, ISBN 978-83-01-16405-8.

[2] *Zasady dopuszczalności słów* [online], Polska Federacja Scrabble [dostęp: 25 kwietnia 2014], Dostępny w Internecie: `<http://www.pfs.org.pl/zds.php>`.

[3] M. Lampis, V. Mitsou, K. Sołtys, *Scrabble is PSPACE-Complete*, 2012.

[4] B. Sheppard, *World-championship-caliber Scrabble*, „Artificial Intelligence", vol. 134, p. 241-275, 2002.

[5] E. Freeman, B. Bates, K. Sierra, *Rusz głową! Wzorce projektowe*, Wyd. 2, Gliwice, Wydawnictwo Helion, 2011, ISBN 978-83-246-2803-2, strona 56.

[6] Jason Katz-Brown, John O'Laughlin, *How Quackle Plays Scrabble* [online], Quackle [dostęp: 25 kwietnia 2014], Dostępny w Internecie: `<http://people.csail.mit.edu/jasonkb/quackle/doc/how_quackle_plays_scrabble.html>`.

[7] H. J. Berliner, C. McConnell, *B^* Probability Based Search*, Carnegie Mellon University, 1995.