Zaawansowana sztuczna inteligencja do Scrabble

Podstawy teoretyczne i opis implementacji algorytmu

Jakub Turek

26 kwietnia 2014

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.

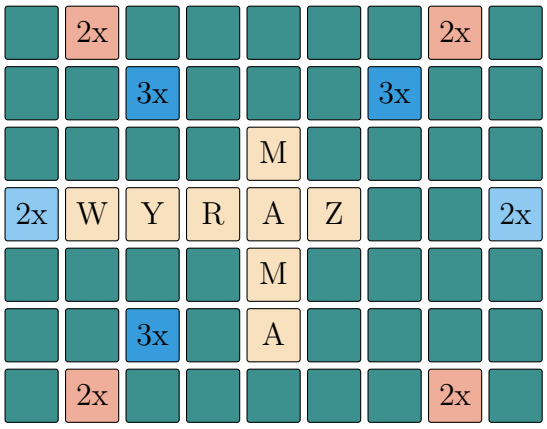
Wstęp

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:

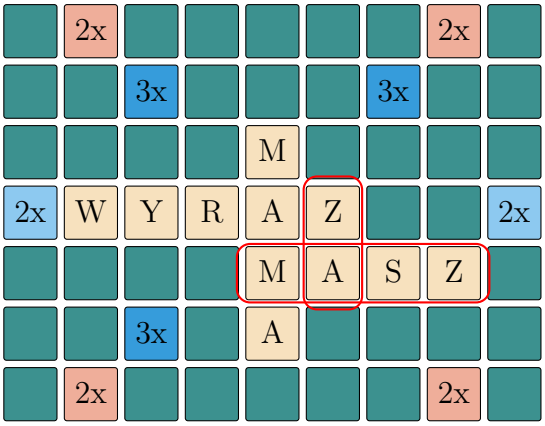
Rysunek 1 Pokazuje sytuację początkową obrazującą pewien moment rozgrywki.

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".

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



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

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.

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.

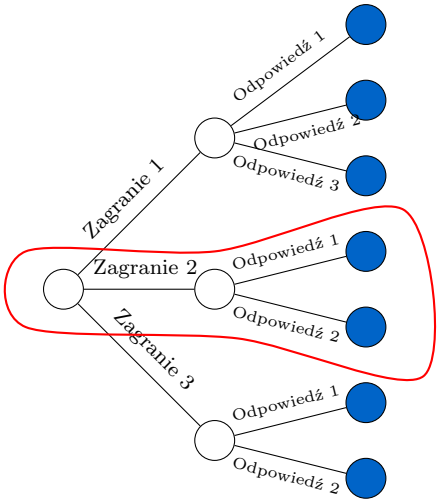
Aby dowieść twierdzenie 1 należy uprzednio zdefiniować przestrzeń stanów w Scrabble.

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.

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*.



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

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 poziomów (gdy gracze wykładają klocki pojedynczo) lub więcej (dochodzi możliwość pasowania). Wartości te są zbyt duże, by 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, kiedy rozgrywka może zakończyć się w dwóch turach, a więc kiedy w worku pozostaje siedem lub mniej klocków. Kończy się wraz z rozpoczęciem fazy EG.

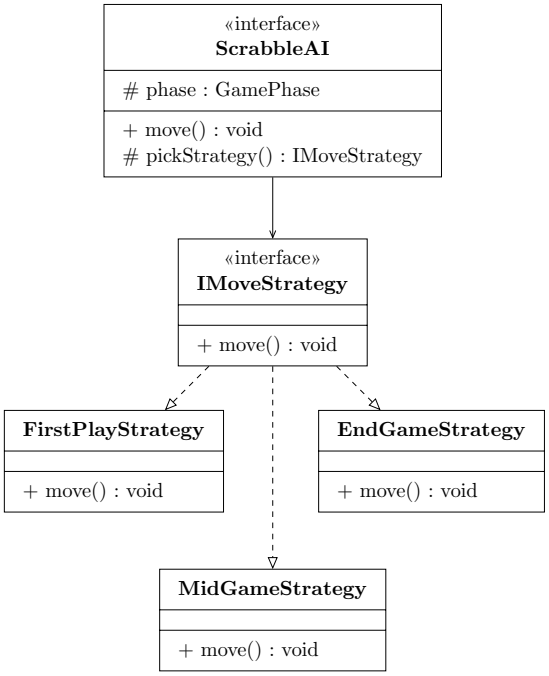
End Game faza zakończenia. Rozpoczyna się, gdy w worku nie ma już klocków i trwa aż do końca gry. Charakteryzuje się tym, że w trakcie jej trwania zawodnicy dysponują pełną informacją o grze, a więc również o literach posiadanych przez przeciwnika.

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.

Opening Play

Celem fazy OP jest wybranie najwyżej punktowanego zagrania. W algorytmie Quackle wybór ten jest dokonywany poprzez

[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



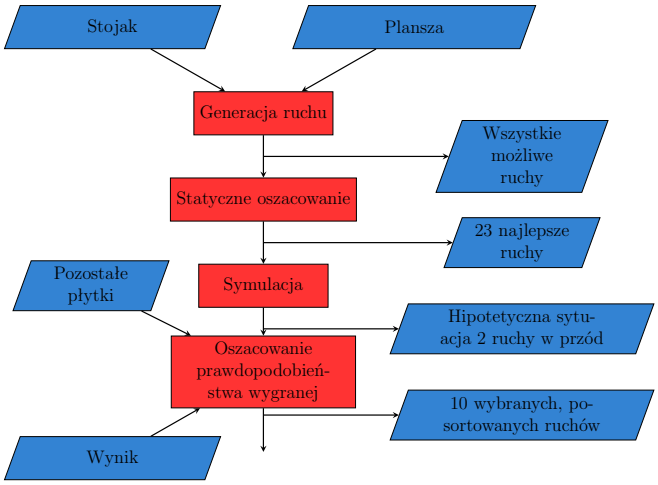
Rysunek 4: Przykład użycia wzorca projektowego Strategia w kontekście implementacji algorytmu.

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 jednostkowym[2].

Mid Game

W fazie MG w aplikacji Quackle wykorzystywany jest algorytm z heurystyczną funkcją statyczną oraz symulacją dwóch zagrań w przyszłość. 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 ru-

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

chów dla określonej kombinacji liter i ułożenia klocków na planszy.

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, który 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**, 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 na podstawie zapisów turniejowych.

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

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.
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.

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.

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 sieci neuronowe.

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.

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 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, więc do przeszukiwania przestrzeni stanów należy wykorzystać inny algorytm. Przykładem przeszukiwania wspierającego przedziały jest algorytm B^*. [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.

Słowniki do gier

Reguły Scrabble dopuszczają układanie „wszystkich słów występujących w słownikach języka polskiego oraz wszystkich ich prawidłowych form gramatycznych, z wyjątkiem takich słów, które rozpoczynają się od wielkiej litery, są skrótami, bądź słowami wymagającymi cudzysłowu lub łącznika". [2]

Brak zamkniętej listy słów możliwych do wykorzystania w grze prowadzi do konfliktu interesów, ponieważ gracze sami muszą rozstrzygnąć między sobą, czy dany wyraz jest dopuszczalny, czy nie. Aby umożliwić uczciwą rozgrywkę konieczne było stworzenie zbioru, który zawiera listę słów, wraz ze wszystkimi poprawnymi odmianami, dopuszczalnych do wykorzystania w grach słownych. W taki sposób zaczęły powstawać słowniki wyrazów do gier. [8]

W chwili obecnej istnieją dwa duże polskie słowniki do gier:

Oficjalny Słownik Polskiego Scrabblisty Przygotowany przez wydawnictwo naukowe PWN. Dopuszcza użycie wyrazów znajdujących się w słownikach PWN wydanych po roku 1980.

Słownik alternatywny Przygotowany na potrzeby serwisu z grami internetowymi `http://kurnik.pl`. Jest rozwijany przez administratorów serwisu przy współpracy z internautami. Dopuszcza użycie wyrazów znajdujących się w słownikach dowolnego wydawnictwa wydanych po roku 1980, z wyłączeniem czarnej listy słowników[3].

[3] Jest to lista słowników, które nie zostały dopuszczone jako źródło wyrazów ze względu na nie-

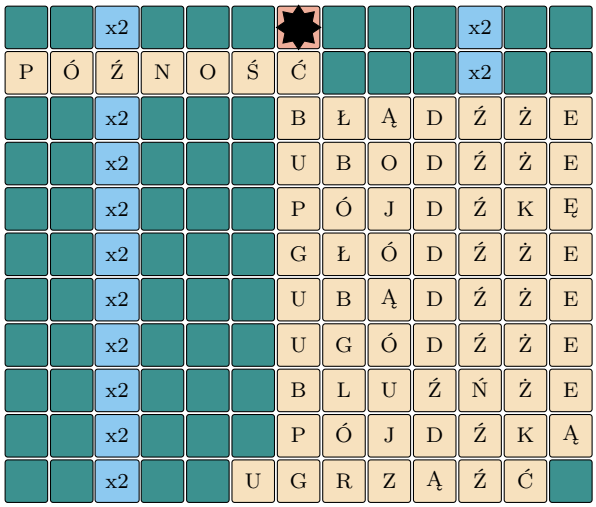
Poza dopuszczalnymi źródłami informacji słowniki posiadają jeszcze jedną znaczącą różnicę. Oficjalny Słownik Polskiego Scrabblisty jest dystrybuowany jako program umieszczony na płycie CD, który nie udostępnia listy wszystkich wyrazów zawartych w słowniku. Program pozwala jedynie sprawdzić, czy dane słowo jest dopuszczalne. Słownik alternatywny dostępny jest w Internecie również w postaci listy wszystkich dopuszczalnych wyrazów zebranych w pliku tekstowym. Pozwala to na przeprowadzenie analiz statystycznych, stąd w dalszej części artykułu autor rozważa wyłącznie Słownik alternatywny.

Analiza statystyczna

W celu wyznaczenia heurystyk mogących wspomóc sztuczną inteligencję grającą w Scrabble autor przeanalizował dane zawarte w Słowniku alternatywnym. Pierwszą analizowaną informacją jest częstotliwość występowania poszczególnych liter w słowniku. Wynika to z faktu, że zasady gry Scrabble są oparte na rozkładzie prawdopodobieństwa występowania liter w danym języku. Możliwe jest, że badanie statystyczne słownika przyniesie informacje o literach, których warto używać w pierwszej kolejności.

Wykres przedstawiony na rysunku 69 dostarcza nam informacji, że autorzy polskich reguł do gry Scrabble nie oszacowali poprawnie częstotliwości występowania wszystkich liter. Przykładowo litera **o** występuje w słowniku częściej niż litera **e**, natomiast prawdopodobieństwo wylosowania odpowiadających tym literom płytek w grze kształtuje się odwrotnie. Wynika z tego, że porównując dwa podobnie punktowane zagrania, warto wybrać to, które wykorzystuje więcej samogłosek **e** niż **o**. Podobną zależność można stwierdzić w przypadku litery **h**, której częstotliwość występowania została przez autorów zasad znacząco zawyżona. Korzytnie jest wybierać zagrania, które powodują wyłożenie płytek z literą **h**.

Interesującą statystyką są najlepsze otwarcia (czyli pierwsze zagrania w grze). Pozwalają one, przy wylosowaniu szczęśliwej kombinacji płytek, zagrać w najbardziej opłacalny sposób (na początku rozgrywki nie ma ryzyka pozostawienia przeciwnikowi „otwartej gry"). Statystyka ta jest szczególnie przydatna dla ludzi, chociaż również sztuczna inteligencja może wykorzystać ją do pominięcia zbędnych obliczeń. Najlepsze otwarcia dla słownika alternatywnego zebrano na rysunku 6.



Rysunek 6: Najlepsze otwarcia dla Słownika alternatywnego. Słowo *późność* jest warte 126 punktów, pozostałe po 124 punkty.

Dokonując statystycznej analizy słownika warto wziąć pod uwagę najbardziej opłacalne kombinacje liter. W trakcie rozgrywki gracze dążą do układania wyrazów, które wymagają użycia wszystkich siedmiu płytek znajdujących się na stojaku. Zagranie takie jest bowiem nagradzane dodatkową premią w wysokości 50 punktów. Stąd przez najbardziej opłacalną kombinację liter należy rozumieć taką zawartość stojaka, która pozwala ułożyć jak najwięcej wyrazów

dostateczną jakość opracowania.

siedmioliterowych. Duża liczba kombinacji sprawia, że z większym prawdopodobieństwem uda nam się dopasować wyraz do istniejącego układu płytek na planszy. Autor przeanalizował najlepsze kombinacje liter dla Słownika alternatywnego. Wyniki zebrane są w tabeli 7.

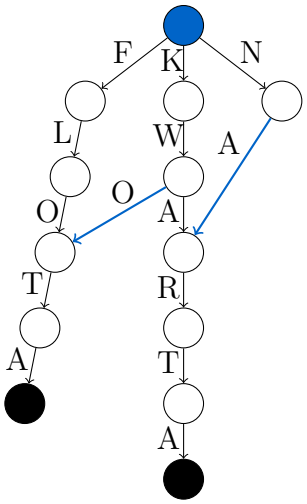
Litery	Liczba możliwych do ułożenia wyrazów
E, I, K, L, N, O, W	12 słów
A, E, I, K, P, R, S	12 słów
A, E, I, K, L, N, P	12 słów
A, E, K, N, R, T, Y	11 słów
A, I, K, M, O, P, S	11 słów
A, I, K, M, O, R, W	11 słów
A, A, I, K, L, M, S	10 słów
A, I, K, M, O, S, T	10 słów
A, I, K, L, N, O, W	10 słów
A, I, K, L, M, N, O	10 słów

Rysunek 7: Najlepsze kombinacje liter dla Słownika alternatywnego.

Najbardziej opłacalne kombinacje liter można wykorzystać do strategicznego planowania zagrań. Wykonując poszczególne ruchy korzystnie jest dążyć do uzyskania jednego z przedstawionych układów liter na stojaku. Można tego dokonać poprzez usuwanie płytek, które nie zostały uwzględnione w tych kombinacjach, jak również usuwanie płytek zduplikowanych (częstą sytuacją jest posiadanie kilku płytek z tą samą samogłoską).

Wyznaczanie wszystkich dopuszczalnych ruchów

Podstawowym zadaniem sztucznej inteligencji w grze Scrabble jest wyznaczenie (na podstawie zadanego słownika) wszystkich dopuszczalnych ruchów w danej turze. Odpowiedni algorytm został opracowany przez A. Appela i G. Jacobsona. [9] Autorzy proponują opisanie słownika za pomocą grafowej stuktury o nazwie DAWG[4]. Jest to odmiana drzewa trie[5], która charakteryzuje się silną kompresją danych uzyskiwaną dzięki łączeniu krawędziami identycznych sufiksów wyrazów. Przykładowy słownik DAWG został przedstawiony na rysunku 8.



Rysunek 8: Struktura DAWG, która opisuje wyrazy *flota*, *kwota*, *kwarta* oraz *narta*.

Autorzy wybrali tę strukturę ze względu na bardzo szybkie wyszukiwanie słów po ich prefiksie, a także na dobry współczynnik kompresji słownika. Pełen słownik do gier dla języka angielskiego opisany przy pomocy DAWG zajmuje około 2.5 MB pamięci.

Autorzy przedstawili czterokrokowy algorytm (zwany dalej algorytmem Appela-Jacobsona) wykorzystujący słownik DAWG, który umożliwia wyznaczenie wszystkich możliwych ruchów przy danym stanie planszy i dostępnych do wykorzystania literach:

1. Redukcja złożoności problemu do jednego wymiaru. Algorytm rozpatruje ruchy wyłącznie w jednym kierunku (pionowo lub poziomo), ograniczając

[4]Z angielskiego Directed Acyclic Word Graph - skierowany, niecykliczny graf słów.

[5]Od angielskiego słowa *retrieval*, które oznacza odczyt.

się do jednego wiersza lub kolumny. Rozumowanie to jest powtarzane dla każdego wiersza (lub kolumny), a następnie plansza do gry jest transponowana i algorytm wykonywany jest dla drugiego kierunku. W dalszym opisie autor artykułu zakłada, że rozpatrujemy wyłącznie wiersze, a nie kolumny, planszy.

2. Ograniczanie zbioru znaków, które można legalnie wstawić w daną komórkę. W tym kroku wykorzystuje się fakt, że tworzenie wyrazów w danym kierunku może skutkować utworzeniem nowych słów w kierunku przeciwnym, ale tylko poprzez dodanie jednego znaku. Jest to trywialny do sprawdzenia przypadek, który znacząco ogranicza liczbę możliwych do wykonania legalnych ruchów.

3. Wyznaczenie kotwic[6]. Kotwica jest to najbardziej wysunięta na lewo płytka nowego wyrazu, która przylega do innej, znajdującej się na planszy płytki. Zgodnie z zasadami gry w Scrabble, każdy nowo utworzony wyraz musi posiadać dokładnie jedną kotwicę.

4. Rozwijanie możliwych do utworzenia wyrazów poprzez wyjście od wyznaczonych w punkcie trzecim kotwic, a także uwzględnienie wyznaczonych w punkcie drugim ograniczeń.

Rozwijanie możliwych wyrazów rozpoczyna się od płytek znajdujących się po lewej stronie kotwicy. Rozpatrywane są trzy przypadki:

- Trywialny - na lewo od kotwicy nie ma żadnych płytek.
- Trywialny - część wyrazu na lewo od kotwicy składa się wyłącznie z płytek znajdujących się już na planszy.
- Lewa część wyrazu składa się z liter znajdujących się na stojaku i dostępnych do zagrania. Należy zbadać wszystkie możliwe kombinacje.

Następnie rozwijana jest prawa część wyrazu, która uwzględnia kotwicę oraz wszystkie płytki na prawo od niej. W tym celu wyszukuje się w słowniku poprawne sufiksy, które składają się z dostępnych na stojaku liter oraz spełniają wyznaczone ograniczenia.

Algorytm Appela-Jacobsona jest wydajny, posiada jednak narzut obliczeniowy związany z wyznaczaniem lewostronnych dopełnień wyrazów. Zakładając, że kotwica jest również ostatnią literą nowego wyrazu, należy zbadać $6! = 720$ lewostronnych kombinacji. W pesymistycznym przypadku, kiedy na stojaku znajdują się dwa blanki[7], liczba badanych kombinacji wzrasta do $\frac{4! \times 32^2}{2} = 12288$. Biorąc pod uwagę fakt, że większość wyznaczonych prefiksów w ogóle nie będzie występować w słowniku, jest to znaczący narzut.

Powyższa obserwacja stała się podstawą zmodyfikowanego algorytmu zaprezentowanego przez Stevena Gordona. [10] Autor nie zmienia kroków algorytmu wykorzystywanych do generacji wszystkich kombinacji ruchów. Zaproponowana modyfikacja obejmuje wykorzystanie struktury GADDAG[8], w miejsce słownika DAWG.

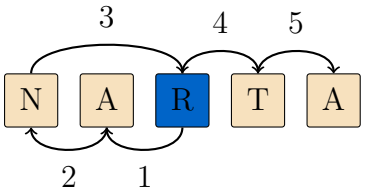
GADDAG jest odmianą drzewa trie, która jest nastawiona na szybkie prefiksowanie wyrazów. Każdy opisany wyraz podzielony jest na dwie części: prefiks oraz sufiks. Wychodząc od korzenia drzewa kolejne wierzchołki opisują prefiks wyrazu czy-

[6] W oryginalnej pracy termin ten został wprowadzony przy pomocy angielskiego zwrotu *anchor*.

[7] Blank - płytka, która nie posiada wartości punktowej, ale może być zastąpiona dowolną literą.

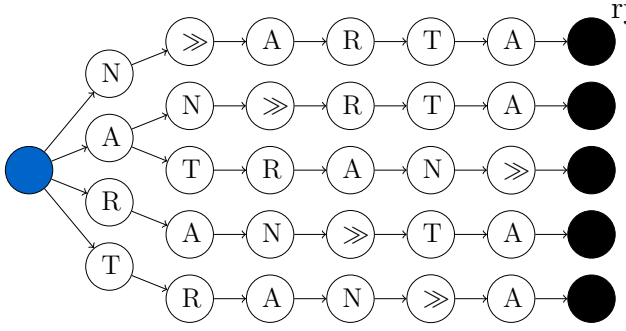
[8] Autor nie podaje rozwinięcia tego skrótu, jednak ze względu na postać przechowywanych w strukturze danych można domyślać się, że jest ona wariacją nazwy DAG (z angielskiego Directed Acyclic Graph - skierowany, acykliczny graf).

tany od tyłu. Dalej występuje znak zakończenia prefiksu (oznaczany ≫), a następnie sufiks wyrazu czytany w normalnej kolejności. Kształt struktury jest silnie powiązany z algorytmem Appela-Jacobsona. Wychodząc od kotwicy w kierunku lewej strony wyrazu analizowane są kolejne litery prefiksu w odwróconej kolejności. Dalej następuje przejście do kotwicy i rozpatrywany jest sufiks w normalnej kolejności. Ilustruje to rysunek 9.



Rysunek 9: Kolejność analizowania pól w algorytmie Appela-Jacobsona. Kotwica oznaczona jest na niebiesko.

Przykładowa struktura GADDAG opisująca wyraz „narta” została zaprezentowana na rysunku 10. Należy zauważyć, że sprawdzenie występowania wyrazu w słowniku wymaga dodatkowego kroku - słowo należy odwrócić. Dla podanego przykładu sprawdzenie poprawności wyrazu narta wymaga poszukiwania frazy *ATRAN* ≫.



Rysunek 10: Przykład struktury GADDAG dla wyrazu *narta*.

Dobór strategii w zależności od fazy gry

Rozgrywkę w Scrabble można podzielić na trzy fazy gry:

MG (mid-game) trwa od rozpoczęcia rozgrywki do rozpoczęcia fazy *PEG*,

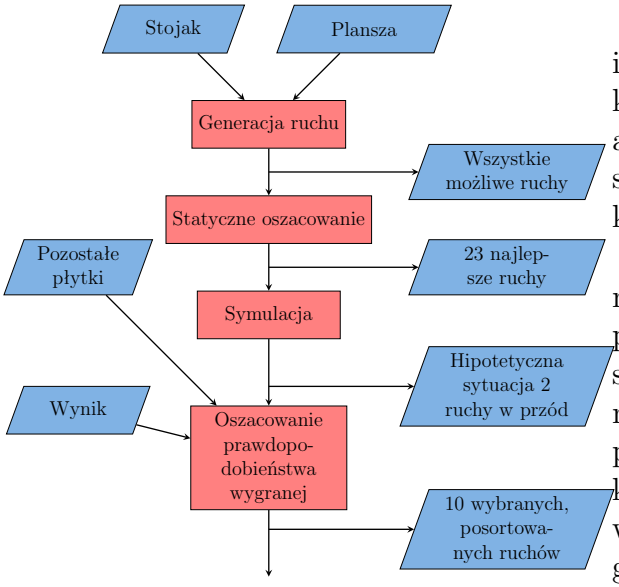
PEG (pre-endgame) rozpoczyna się, kiedy do pobrania pozostają wyłącznie jedna (lub, w zależności od definicji) dwie płytki,

EG (end-game) rozpoczyna się, kiedy nie ma już płytek do pobrania, a więc gracze znają wzajemnie własne płytki.

W fazie *MG* liczba możliwych kombinacji płytek przeciwnika oraz ruchów jest tak duża, że nie jest możliwa analiza przestrzeni stanów. Podejściem stosowanym w tej fazie rozgrywki jest wykorzystanie metod heurystycznych i symulacji do wyboru najlepszego ruchu. Schemat postępowania zostanie opisany na przykładzie algorytmu wykorzystywanego w programie *Quackle* - uznawanego za najlepszego komputerowego gracza Scrabble na świecie. [6] Diagram ilustrujący algorytm został przedstawiony na rysunku 11.

Quackle wykorzystuje następujący algorytm:

1. Generacja wszystkich dopuszczalnych możliwości ruchu.
2. Przypisanie każdej z możliwości statycznego oszacowania. Funkcja wykorzystywana do szacowania może być tak prosta, jak suma punktów uzyskanych za dane zagranie.
3. Sortowanie ruchów od najlepszego do najgorszego, względem wartości statycznego oszacowania.



Rysunek 11: Schemat blokowy algorytmu wykorzystywanego w aplikacji *Quackle*.

4. Wylosowanie prawdopodobnego układu płytek, którymi może dysponować przeciwnik.

5. Symulacja przeprowadzana dla 23 najlepszych ruchów. Dla wylosowanego układu płytek przeciwnika symulowana jest rozgrywka na dwa kroki w przód. Do oceny wykorzystuje się stan punktów po przeprowadzeniu symulacji.

6. Na podstawie aktualnego wyniku rozgrywki i liczby płytek pozostałych do pobrania szacowane jest prawdopodobieństwo wygranej dla każdego z zasymulowanych ruchów.

7. Wybieranych jest 10 ruchów dających największe prawdopodobieństwo wygranej.

8. Na podstawie dodatkowych czynników (przykładowo: zestawu wykorzystanych w danym ruchu liter) wybierany jest najlepszy ruch.

W fazach *PEG* oraz *EG* stosowane jest inne podejście. Liczba scenariuszy rozgrywki znacząco spada, dlatego możliwe staje się analizowanie przestrzeni stanów, która została poglądowo zaprezentowana na rysunku 3.

Na etapach *PEG* oraz *EG* wciąż nie jest możliwe dokonanie pełnego przeszukiwania przestrzeni stanów, nie można więc skorzystać z algorytmów $\alpha - \beta$, czy A^*. Użyteczne są algorytmy pozwalające przeszukiwać przestrzeń stanów z ograniczeniami (przykładowo: B^*) istnieje bowiem sposób na wyznaczenie dobrego przybliżenia zarówno górnego, jak i dolnego przedziału wartości punktowych, które pozwolą odnieść zwycięstwo w rozgrywce. Dodatkowo przeszukiwanie powinno być prowadzone w sposób progresywny, a więc rozpoczynając od miejsc, w których najszybciej możemy zakończyć przeszukiwanie. Miejsca takie związane są z możliwie małą ilością zagrań, a tym samym możliwie długimi układanymi wyrazami.

Przeszukiwanie przestrzeni stanów jest podejściem, które pozwala wygrać każdą rozgrywkę, o ile warunki początkowe przy wchodzeniu w fazę *EG* pozwalają na zwycięstwo. Świadczą o tym wyniki uzyskiwane przez komputerowych graczy Scrabble przeciwko najlepszym ludziom. Podczas turnieju, w którym najlepsze algorytmy sztucznej inteligencji rywalizowały przeciwko ludziom, *Quackle* uzyskał rezultat 32 wygranych i 4 porażek, a następnie pokonał ówczesnego mistrza świata - Davida Boysa - rezultatem 3 do 2. [11]

Podsumowanie

Artykuł przedstawia wyniki wykonanych przez autora badań statystycznych na polskim słowniku wyrazów do gier. Wyniki tych badań mogą zostać wykorzystane nie tylko przez graczy Scrabble, ale również

w szerokim zakresie nauk związanych z eksploracją danych tekstowych. Ponadto w artykule zostały zebrane informacje niezbędne do zbudowania sztucznej inteligencji grającej w Scrabble. Informacje te pokazują, że możliwe jest zbudowanie zaawansowanej sztucznej inteligencji, która będzie zdolna pokonywać oponentów klasy mistrzowskiej.

Na podstawie zaprezentowanych informacji autor artykułu zbuduje algorytm sztucznej inteligencji dostosowany do polskiego słownika wyrazów do gier. W trakcie prac szczególny nacisk zostanie położony na wykorzystanie wyników badań statystycznych zaprezentowanych w artykule. W ramach budowania algorytmu autor zbada możliwość wprowadzenia poprawek do metod wykorzystywanych obecnie do realizacji automatów grających w Scrabble.

Przedstawione w artykule informacje posłużą także do rozbudowy oprogramowania o elementy analizujące daną partię. Analizie będzie można poddać dowolną rozgrywkę tak, aby oprogramowanie mogło zostać wykorzystane jako osobisty trener.

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.

[8] Anna Andrzejczuk, *Słowniki do gier słownych jako nowy typ wydawnictw leksykograficznych*, pod kierownictwem dr hab. R. Pawelec, Warszawa, 2006.

[9] A. Appel, G. Jacobson, *The world's fastest Scrabble program*, „Communications of the ACM" 1988, 31(5), strony 572-585.

[10] S. Gordon, *A Faster Scrabble Move Generation Algorithm*, „Software - Practice and Experience" 1994, 24(2), strony 219-232.

[11] Jacek Mańdziuk, *Knowledge-Free and Learning-Based Methods in Intelligent Game Playing*, Wydawnictwo Springer, 2010, ISBN 978-3-642-11677-3.