Politechnika Wrocławska

Wydział Podstawowych Problemów Techniki

PRACA DYPLOMOWA

Tytuł pracy dyplomowej:

MASZYNOWE UCZENIE GRAMATYCZNYCH DESKRYPTORÓW SEKWENCJI BIAŁKOWYCH

Autor: Robert Kowalski
Opiekun: dr inż. Witold Dyrka

słowa kluczowe: sekwencje aminokwasów, algorytmy ewolucyjne, języki formalne, gramatyki, probabilistyczne gramatyki bezkontekstowe, analizatory składniowe

krótkie streszczenie: Celem niniejszej pracy jest implementacja wariantu metody maszynowego uczenia probabilistycznego modelu gramatycznego sekwencji białkowych. W pracy opisano pierwszorzędową strukturę białek, podstawy algorytmów ewolucyjne (z uwzględnieniem ich mocnych i słabych stron), procesy uczenia maszynowego, podstawowe pojęcia związane z gramatykami takie jak alfabet, łańcuch oraz język. W celu zbadania wpływu parametrów zaimplementowanego algorytmu na proces uczenia przeprowadzono szereg symulacji dla języków testowych. W celu przetestowania skuteczności zaimplementowanego algorytmu wykorzystano język opisujący dobrze poznaną grupę sekwencji aminokwasów.

Z podziękowaniami dla dr inż. Witolda Dyrki,
za cierpliwość, zaangażowanie
oraz nieocenioną pomoc.

Oraz dla Wrocławskiego Centrum Sieciowo-Superkomputerowego,
za udostępnienie zasobów umożliwiających wykonanie obliczeń.

SPIS TREŚCI

WPROWADZENIE

Struktura pierwszorzędowa białek, to najbardziej podstawowy i najmniej skomplikowany poziom organizacji strukturalnej białka, opisujący w formie liniowej układ aminokwasów tworzących rozpatrywany łańcuch polipeptydowy. Struktura pierwszorzędowa białka warunkuje jego budowę przestrzenną, która ma wpływ na jego właściwości fizykochemiczne. Badanie struktury pierwszorzędowej jest prostsze i tańsze niż badanie budowy przestrzennej białka, w związku z czym metody umożliwiające uzyskanie informacji pozwalających na zakwalifikowanie białek do określonych grup (np. o podobnej funkcji) na podstawie ich struktury pierwszorzędowej mogą być bardzo przydatne w biologii i medycynie molekularnej. Ze względu na to, że na poziomie struktury pierwszorzędowej białka mogą zostać przedstawione jako ciąg następujących po sobie aminokwasów, możliwy jest ich opis w postaci języka formalnego, w którym skończony zbiór monomerów (20 aminokwasów, które można traktować jako alfabet) tworzy zbiór łańcuchów (białka, które mogą być traktowane jako słowa) [1].

Algorytmy wykorzystujące założenia koncepcji języka białek mogą być wykorzystywane jako narzędzia bioinformatyczne umożliwiające, na przykład, stwierdzenie, z dużym prawdopodobieństwem, czy sprawdzany polipeptyd należy do określonej rodziny białek lub czy występuje w nim określone miejsce wiążące na podstawie jego struktury pierwszorzędowej [2].

CEL I ZAKRES PRACY

Celem niniejszej pracy jest wprowadzenie ulepszeń do istniejącej metody probabilistycznego modelu gramatycznego sekwencji białkowych [3, 4] w zakresie maszynowego uczenia modelu.

W ramach pracy opisałem algorytmy ewolucyjne na przykładzie prostego algorytmu genetycznego (ang. Simple Genetic Algorithm, SGA) w kontekście nauki probabilistycznych gramatyk bezkontekstowych (rozdział 1). Scharakteryzowałem wykorzystane w doświadczeniach języki (rozdział 2). Zaimplementowałem algorytm SGA w środowisku MATLAB (rozdział 3, aplikacja w załączniku). Wykorzystując stworzoną aplikację przetestowałem wpływ poszczególnych parametrów algorytmu (rozdział 4, podrozdział 2) oraz liczby symboli nieterminalnych wykorzystywanych przez reguły gramatyki (rozdział 4, podrozdział 3) na przebieg i wyniki uczenia. Sprawdziłem jak opracowany algorytm radzi sobie z uczeniem probabilistycznej gramatyki bezkontekstowej z wykorzystaniem próbki dobrze poznanych sekwencji aminokwasów z rodziny motywów białkowych (rozdział 4, podrozdział 4).

Obliczenia wykonano przy użyciu zasobów udostępnionych przez Wrocławskie Centrum Sieciowo-Superkomputerowe (http://wcss.pl), grant obliczeniowy Nr 98.

1. ALGORYTMY EWOLUCYJNE

1.1. WPROWADZENIE DO ALGORYTMÓW EWOLUCYJNYCH

Algorytmy ewolucyjne służą do przeszukiwania przestrzeni alternatywnych rozwiązań w celu wyszukania rozwiązania optymalnego lub rozwiązania, które będzie do niego w wystarczającym stopniu zbliżone w określonym, rozsądnym czasie [5].

Terminologia wykorzystywana przy opisywaniu algorytmów ewolucyjnych jest inspirowana terminologią wykorzystywaną do opisywania procesów ewolucyjnych w przyrodzie. Zbiór rozwiązań rozpatrywanego problemu nazywamy populacją, która składa się z pojedynczych osobników (będących pojedynczymi rozwiązaniami problemu). Rozwiązywany problem określamy mianem środowiska. W zależności od jakości danego rozwiązania, każdemu osobnikowi przypisywana jest wartość liczbowa, która określana jest mianem przystosowania. Każdy osobnik określany jest przez zestaw parametrów określany jako fenotyp, który z kolei jest kodowany przez zestaw wartości nazywany genotypem. Dla rozpatrywanego środowiska można określić funkcję przystosowania, która każdemu osobnikowi przypisuje konkretną wartość przystosowania w zależności od opisującego danego osobnika fenotypu. Genotyp osobnika składa się z chromosomów (lub pojedynczego chromosomu), które składają się z elementarnych jednostek nazywanych genami [6].

1.2. DZIAŁANIE ALGORYTMÓW EWOLUCYJNYCH

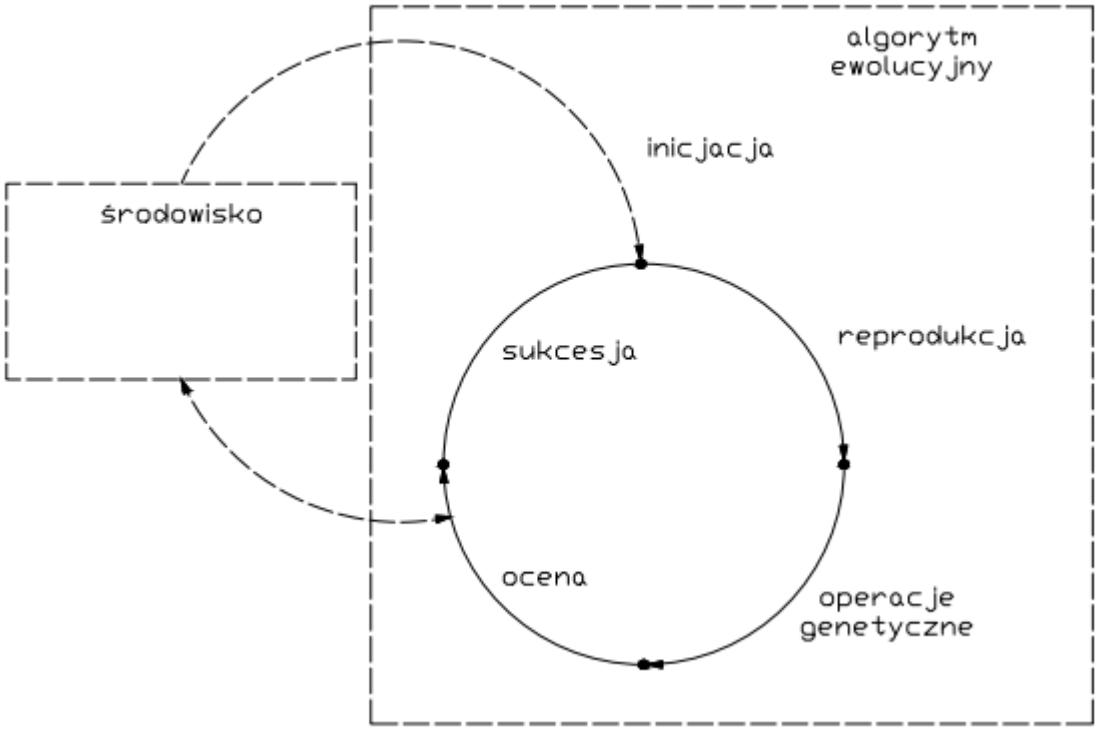
Działanie algorytmu ewolucyjnego (zilustrowane na Rys. 1.1) polega na kolejnym przeprowadzaniu procesów reprodukcji, operacji genetycznych, oceny oraz sukcesji.

Proces reprodukcji polega na losowym powielaniu poszczególnych osobników składających się na rozpatrywaną populację (przy czym prawdopodobieństwo powielenia konkretnego osobnika jest tym większe im lepsze jest jego przystosowanie do rozpatrywanego środowiska). Podczas procesu reprodukcji każdy osobnik może zostać powielony zarówno wielokrotnie, jak i ani razu.

Powstały w wyniku reprodukcji zbiór osobników jest nazywany osobnikami rodzicielskimi, są one poddawane operacjom genetycznym takim jak mutacje, które polegają na losowym modyfikowaniu opisującego danego osobnika genotypu (niekiedy przyjmuje się, że niewielkie mutacje są bardziej prawdopodobne niż duże). Krzyżowanie to operacja polegająca na utworzeniu osobnika potomnego, którego chromosomy są efektem wymieszania odpowiednich chromosomów osobników rodzicielskich (co najmniej dwóch), uzyskana w ten sposób grupa osobników nazywana jest populacją potomną.

W etapie oceny środowiska każdemu osobnikowi z populacji potomnej przypisywana jest konkretna wartość przystosowania (na podstawie funkcji przystosowania).

W etapie sukcesji tworzona jest nowa populacja bazowa (w skład której mogą wchodzić osobniki wyłącznie z populacji potomnej, jak i osobniki z populacji potomnej wraz z osobnikami z poprzedniej populacji bazowej) [6].



Rysunek 1.1 - Schemat działania algorytmu ewolucyjnego [na podstawie 4].

1.3. PROSTY ALGORYTM GENETYCZNY

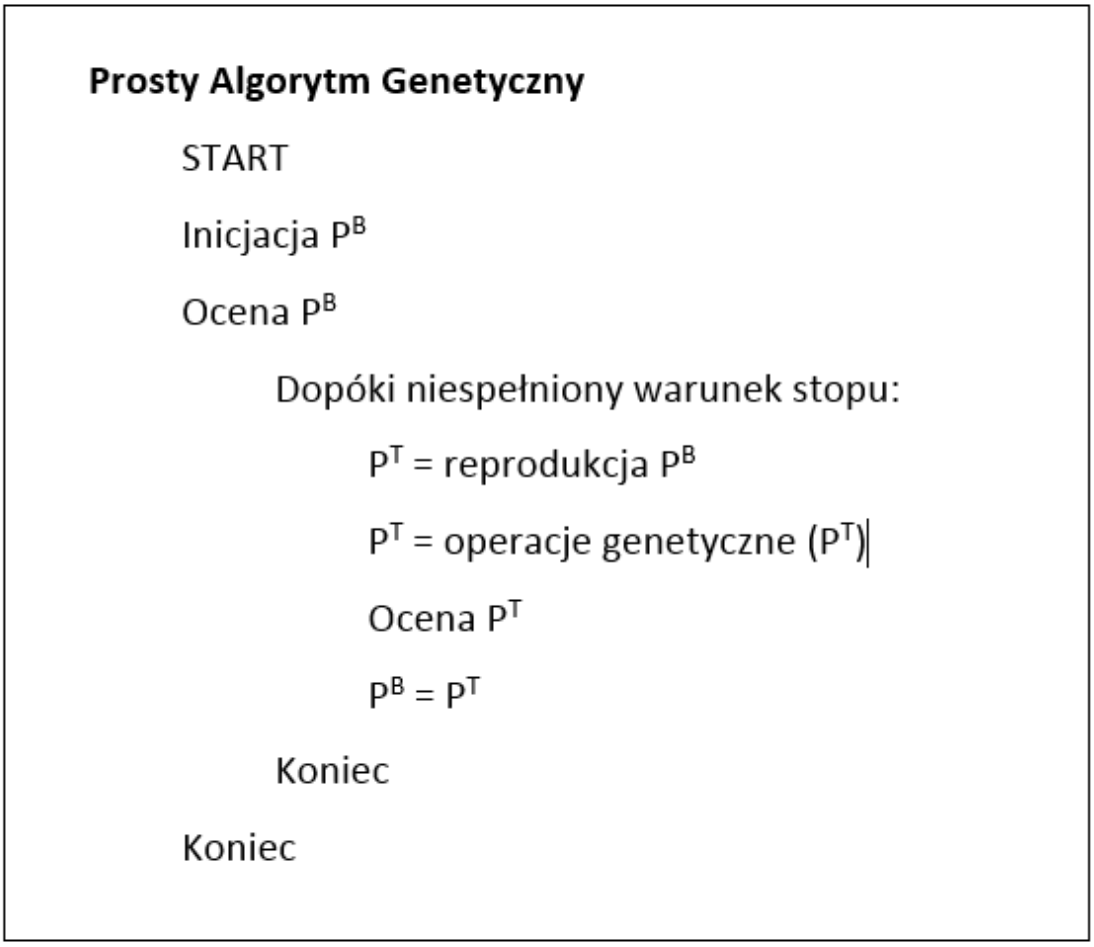
1.3.1. Właściwości prostego algorytmu genetycznego

Jako prosty algorytm genetyczny określa się zaproponowany w 1975 roku przez Johna Hollanda [8] algorytm mający początkowo za zadanie modelowanie procesu ewolucji. Jego działanie zostało zilustrowane na Rys. 1.2.

W procesie inicjacji tworzona jest pierwsza populacja bazowa (P^B) (osobniki należące do niej są generowane w sposób losowy). Populacja bazowa jest następnie poddawana ocenie, w wyniku której każdemu osobnikowi przypisywana jest wartość przystosowania.

Po inicjacji rozpoczyna się pętla, w której wykonywane są kolejno procesy: reprodukcji, operacji genetycznych oraz oceny. Procesy te powtarzane są aż do momentu, w którym spełniony zostanie warunek stopu, którym może być np. określona liczba obiegów pętli.

Proces reprodukcji polega na losowaniu osobników (z prawdopodobieństwem proporcjonalnym do określonego podczas oceny przystosowania) z obecnej populacji bazowej, wylosowany osobnik jest kopiowany, zaś stworzona w ten sposób kopia jest przenoszona do populacji tymczasowej (P^T). Proces losowania i kopiowania osobników jest powtarzany do momentu, w którym populacja tymczasowa osiąga wielkość populacji bazowej. Następnie populacja tymczasowa poddawana jest operacjom genetycznym, po których następuje ocena populacji tymczasowej, która staje się od tej poty kolejną populacją bazową [6].



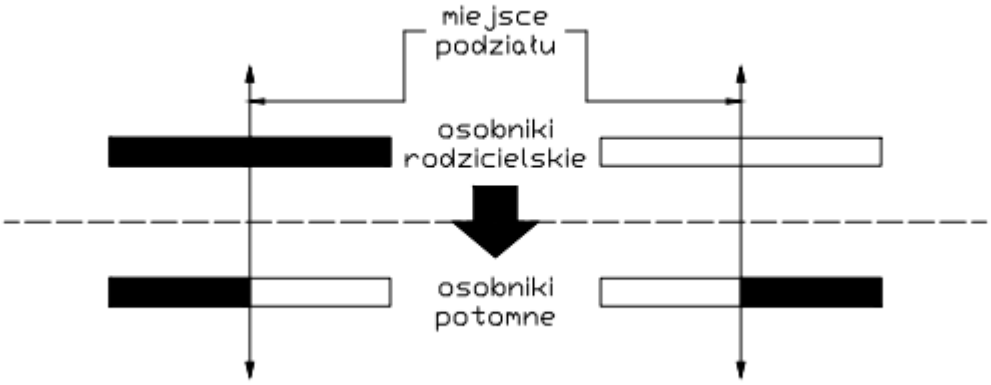
Rysunek 1.2 - Schemat działania algorytmu SGA[na podstawie 4].

1.3.2. Kodowanie osobników i operacje genetyczne w algorytmie SGA

Najprostszym kodowaniem osobników stosowanym w algorytmach genetycznych jest kodowanie binarne, zgodnie z którym genotyp każdego osobnika wchodzącego w skład populacji składa się z wektorów wypełnionych 0 lub 1 (wektory te pełnią funkcję chromosomów, zaś każdy element wektora jest pojedynczym genem).

Podczas mutacji każdy gen w genotypie danego osobnika jest rozpatrywany niezależnie, jeżeli dla konkretnego genu zostanie wylosowana decyzja o mutacji (prawdopodobieństwo takiego zdarzenia jest parametrem algorytmu SGA), to wartość genu zostaje zanegowana.

Podczas procesu krzyżowania (zilustrowanego na Rys. 1.5) losowane jest konkretne miejsce na chromosomie (z rozkładu równomiernego), następnie chromosomy obu krzyżowanych osobników rozcinane są w wylosowanym miejscu i pierwszy fragment chromosomu osobnika pierwszego łączy się z drugim fragmentem chromosomu osobnika drugiego, zaś drugi fragment chromosomu osobnika pierwszego łączy się z pierwszym fragmentem chromosomu osobnika drugiego [9].



Rysunek 1.3 - Schemat procesu krzyżowania [na podstawie 6].

Prawdopodobieństwo powielenia konkretnego osobnika z populacji bazowej w fazie reprodukcji wyraża się wzorem:

$$\Pr(X) = \frac{\Phi(X)}{\sum Y \in P^T \Phi(Y)} \quad (1)$$

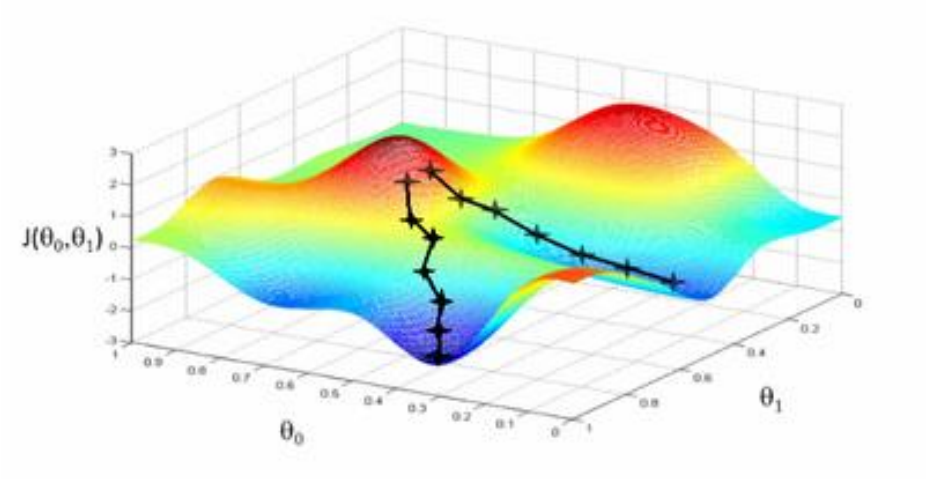
Gdzie X oznacza osobnika, $\Phi(X)$ oznacza wartość przystosowania. Metoda ta jest nazwana reprodukcją proporcjonalną, bądź reprodukcją ruletkową [6].

Najczęściej stosowane w praktyce warunki przerwania algorytmu zostały opisane już wcześniej. Ze względu na to, że algorytm SGA opiera się na procesach losowych, nie jesteśmy w stanie uzyskać gwarancji, że otrzymany wynik jest wynikiem optymalnym. Jednakże, wraz ze wzrostem liczby generacji, rośnie prawdopodobieństwo zbliżenia się do wyniku optymalnego [5].

1.3.3. Słabości algorytmów ewolucyjnych

Ze względu na uwzględnienie wartości przystosowania podczas procesu reprodukcji, geny osobników wykazujących niskie przystosowanie do rozpatrywanego środowiska, zostaną z dużym prawdopodobieństwem wykluczone z populacji bazowej po pewnej liczbie obiegów pętli algorytmu. Ponadto wszelkie mutacje powodujące zmniejszenie przystosowania zostaną najprawdopodobniej odrzucone (podczas gdy mutacje pozwalające poprawić wartość przystosowania do środowiska zostaną zachowane i powielone w kolejnych populacjach bazowych). Ze względu na to, zachodzi wysokie prawdopodobieństwo, że wraz ze wzrostem liczby obiegów pętli algorytmu będzie rosnąć poziom przystosowania kolejnych populacji bazowych. Kierunek rozwoju kolejnych populacji bazowych jest jednak mocno uzależniony od przebiegu funkcji przystosowania, parametrów pierwszej populacji bazowej oraz od rodzaju zachodzących mutacji. Jeżeli kolejna populacja bazowa znajdzie się w lokalnym maksimum funkcji przystosowania, to każda kolejna zmiana fenotypu będzie powodowała zmniejszenie przystosowania tej populacji do danego środowiska, w konsekwencji

uniemożliwiając opuszczenie lokalnego maksimum funkcji przystosowania (które może być o wiele mniejsze od globalnego maksimum tejże funkcji, zostało to zilustrowane na Rys 1.4.Algorytmy genetyczne są więc, jak wiele innych metod optymalizacji, podatne na osiadanie w maksimach lokalnych. Istnieją jednak metody umożliwiające zwiększenie szansy na opuszczenie takich pułapek ewolucyjnych, np. poprzez odpowiednie sterowanie skalą i prawdopodobieństwem mutacji oraz zwiększanie prawdopodobieństwa reprodukcji osobników słabiej przystosowanych.



Rysunek 1.4 - przedstawienie prawdopodobnego procesu zwiększania przystosowania populacji bazowej wraz z kolejnymi iteracjami algorytmu ewolucyjnego, θ_0 oraz θ_1 oznaczają parametry funkcji, zaś $J(\theta_0, \theta_1)$ oznacza zależą od parametrów funkcję przystosowania, czarnymi krzyżami oznaczono proces poszukiwania optymalnego rozwiązania przez algorytm (początkowo program znajduje się w obszarach o najniższej wartości $J(\theta_0, \theta_1)$ by wraz z kolejnymi iteracjami dojść do obszarów o wyższej wartości $J(\theta_0, \theta_1)$). Źródło: [10].

1.4. GRAMATYKI

1.4.1. Podstawowe pojęcia związane z gramatykami

Podstawowe pojęcia związane z gramatykami to: alfabety, łańcuchy oraz języki [11].

Alfabetem nazywamy konkretny, skończony, niepusty zbiór symboli. Przykładami alfabetu mogą być:

- zbiór wszystkich bądź niektórych liter alfabetu łacińskiego,
- zbiór symboli ASCII,
- zbiór złożony z jednego symbolu,
- zbiór liczb należących do pewnego przedziału,
- stosowany w informatyce alfabet binarny,
- zbiór 20 symboli przypisanych do konkretnych aminokwasów (stosowany w bioinformatyce).

Alfabety zwyczajowo oznaczamy symbolem $\sum$.

Łańcuchem nazywamy skończony zbiór symboli należących do konkretnego alfabetu. W przypadku, w którym wszystkie znaki danego łańcucha należą do konkretnego alfabetu mówimy, że jest to łańcuch nad tym alfabetem. Łańcuch nad danym alfabetem nie musi wykorzystywać wszystkich symboli które należą do tego alfabetu (np. „ab” to łańcuch nad

alfabetem: $\sum=\{a, b\}$ oraz $\sum=\{a, b, c\}$). Szczególnym przypadkiem jest łańcuch pusty, czyli łańcuch o zerowej liczbie wystąpień symboli, który oznaczamy symbolem ε i jest on łańcuchem nad dowolnym alfabetem.

Językiem nazywamy zbiór łańcuchów, którego wszystkie elementy zostały wybrane ze zbioru wszystkich łańcuchów nad danym alfabetem. Jeżeli wszystkie symbole, wykorzystywane w łańcuchach należących do danego języka, należą do danego alfabetu, to mówimy, że jest to język nad tym alfabetem. Język nie musi wykorzystywać wszystkich symboli zawartych w danym alfabecie, w związku z czym możemy stwierdzić, że jeżeli dany język jest językiem nad alfabetem $\sum$, to jest on również językiem nad dowolnym alfabetem, który jest nadzbiorem alfabetu $\sum$ [11].

1.4.2. Gramatyki formalne

Gramatyki są sposobem opisu języków formalnych. Gramatyka formalna składa się z:

- symboli terminalnych (zbiór symboli alfabetu na którym zbudowany jest język),
- symboli nieterminalnych (dowolny skończony zbiór składający się z symboli pomocniczych, rozłączny ze zbiorem symboli terminalnych),
- symbolu startowego (należący do zbioru symboli nieterminalnych),
- skończonego zbioru reguł przepisywania (produkcji), gdzie produkcja to para słów (pierwsze słowo, nazywane poprzednikiem, może przechodzić w drugie, nazywane następnikiem), w których mogą występować zarówno symbole nieterminalne jak i terminalne.

Aby sprawdzić, czy konkretne słowo należy do języka opisanego przez tak zdefiniowaną gramatykę, należy sprawdzić, czy możliwe jest wyprowadzenie tego słowa od symbolu startowego za pomocą zawartych w danej gramatyce produkcji.

Przykładowo, jeżeli rozpatrywany alfabet to alfabet binarny składający się z symboli 0 i 1, natomiast opisująca go gramatyka zawiera symbole pomocnicze: „A”, „B” i „C”, gdzie „A” jest symbolem startowym, natomiast zbiór produkcji tej gramatyki to: A→BC, B→10 oraz C→1 to możemy stwierdzić, że słowo: „101” należy do tej gramatyki ponieważ możliwe są takie przekształcenia symbolu startowego, że uzyskamy sprawdzane słowo (A→BC, BC→10C, 10C→101).

1.4.3. Probabilistyczne gramatyki bezkontekstowe

Gramatyki formalne, w zależności od postaci jaką przyjmują należące do niej reguły można podzielić na różne klasy. Przykładem takiej klasy gramatyk są gramatyki bezkontekstowe, w których wszystkie produkcje przyjmują postać $A\rightarrow\Gamma$, gdzie A to dowolny symbol nieterminalny, zaś Γ jest dowolnym zbiorem symboli nieterminalnych i terminalnych.

Probabilistyczna gramatyka bezkontekstowa jest rozszerzeniem gramatyki bezkontekstowej, które każdej produkcji przypisuje odpowiadającą jej wartość prawdopodobieństwa wystąpienia danej produkcji względem jej poprzednika. Prawdopodobieństwa dla konkretnych produkcji są przypisywane w ten sposób, aby suma prawdopodobieństw wszystkich produkcji o wspólnym poprzedniku wynosiła 1.

Jeżeli dany łańcuch może zostać wyprowadzony z symbolu startowego danej gramatyki probabilistycznej przy pomocy jej produkcji, to prawdopodobieństwo z jakim łańcuch ten jest

generowany przez tę gramatykę jest równe iloczynowi prawdopodobieństw reguł użytych przy tym wyprowadzeniu. W przypadku, w którym możliwe jest wyprowadzenie danego łańcucha przy użyciu różnych zestawów reguł, prawdopodobieństwo z jakim łańcuch ten jest generowany przez tę gramatykę jest równe sumie prawdopodobieństw poszczególnych wyprowadzeń.

1.5. ANALIZATORY SKŁADNIOWE

Analizatory składniowe, nazywane również parserami, to programy służące do analizy składniowej konkretnych danych wejściowych i określenia ich struktury gramatycznej.

W celu sprawdzenia, czy konkretne słowo należy do danej gramatyki bezkontekstowej niegenerującej symbolu pustego można użyć parsera korzystającego z algorytmu Cocke'a-Youngera-Kasamiego (algorytm CKY lub CYK) [12]. Schemat działania algorytmu CKY został przedstawiony na Rys. 1.3. Sprawdzana gramatyka musi zostać sprowadzona do postaci normalnej Chomsky'ego (forma, w której wszystkie produkcje danej gramatyki zostały sprowadzone do postaci: A→a lub B→CD, gdzie duże litery oznaczają symbole terminalne, natomiast małe litery oznaczają symbole nieterminalne) [13]. Każda gramatyka bezkontekstowa która w wyniku swoich reguł nie produkuje symbolu pustego może zostać sprowadzona do postaci normalnej Chomsky'ego.

```
function [prob] = CYK_Probabilistic(grammar, input, rulesProb)

%Initialize
P=zeros(length(input),length(input),length(grammar.nonTerminals));
%First
for i=1:length(input)
    for r=1:length(grammar.rules.Lex)
        if strcmp(input(i),grammar.rules.Lex{r,2})
            P(1,i,grammar.rules.Lex{r,1})=rulesProb(r+length(grammar.rules.NonLex));
        end
    end
end

n=length(input);
for i=2:n
    for j=1:n-i+1
        for k=1:i-1
            for r=1:length(grammar.rules.NonLex)
                 %left hand side (left rule symbol)
                lhs = grammar.rules.NonLex{r,1};
                 %right hand side (first right symbol of the rule)
                rhs1 = grammar.rules.NonLex{r,2};
                 %right hand side2 (second right symbol of the rule)
                rhs2 = grammar.rules.NonLex{r,3};
                if P(k,j,rhs1) && P(i-k,j+k,rhs2)
                    prob_splitting = rulesProb(r)*P(k,j,rhs1)*P(i-k,j+k,rhs2);
                        P(i,j,lhs) = P(i,j,lhs) + prob_splitting;
                end
            end
        end
    end
end
    prob = P(n,1,1);
end
```

Rysunek 1.5 – Implementacja probabilistycznej wersji algorytmu CKY w programie MATLAB. Funkcja jako parametry wejściowe przyjmuje: strukturę grammar - zawierającą reguły, input - sprawdzane zdanie oraz rulesProb - prawdopodobieństwa przypisane do poszczególnych reguł. Funkcja zwraca prawdopodobieństwo wygenerowania sprawdzanego zdania przez podaną gramatykę.

Przy analizie składniowej z wykorzystaniem parsera CKY występują 3 zagnieżdżone w sobie pętle, których liczba obiegów jest zależna od długości sprawdzanego słowa. Sprawia to, że zależność czasowej złożoności obliczeniowej tego algorytmu od długości sprawdzanego zdania jest równa $O(n^3)$, co oznacza, że n-krotne zwiększenie długości sprawdzanego zdania może spowodować wydłużenie czasu potrzebnego na przeprowadzenie analizy składniowej rzędu n^3 (Rys. 1.5).

1.6. UCZENIE GRAMATYK

Uczenie gramatyk może skupić się na różnych aspektach. Uczenie reguł gramatyki nieprobabilistycznej G polega na odnalezieniu minimalnej liczby reguł, umożliwiającej wyprowadzenie wszystkich sekwencji ze zbioru uczącego. W przypadku nauki gramatyki probabilistycznej G' = <G, Θ> celem staje się przyporządkowanie każdej regule $r_i \in R$ takiego prawdopodobieństwa $\theta_i \in \Theta$, aby zmaksymalizować iloczyn prawdopodobieństw z jakim uzyskana gramatyka generuje sekwencje z próbki uczącej.

Gramatyki probabilistyczne mogą być także wykorzystane jako klasyfikatory. Zadaniem gramatyki pełniącej funkcję klasyfikatora, jest określenie, czy dana sekwencja należy do określonej grupy na podstawie prawdopodobieństwa z jakim sekwencja ta jest wyprowadzana przez tę gramatykę. Klasyfikatory charakteryzuje określona wartość liczbowa, zwana progiem odcięcia, wszystkie sekwencje których prawdopodobieństwo wyprowadzenia jest większe niż próg odcięcia zostają uznane za należące do danej grupy, zaś sekwencje o prawdopodobieństwie niższym niż próg odcięcia są klasyfikowane jako nienależące do tej grupy. Przy nauce gramatyki probabilistycznej, w kontekście pełnienia funkcji klasyfikatora, konieczne jest takie rozłożenie prawdopodobieństw pomiędzy jej regułami, aby możliwe było wyznaczenie punktu odcięcia względem którego blisko 100% próbek pozytywnych będzie klasyfikowane jako należące do sprawdzanej grupy zaś blisko 100% próbek negatywnych będzie klasyfikowane jako nienależące do sprawdzanej grupy. W celu wyznaczenia optymalnego progu odcięcia dla danej gramatyki można posłużyć się krzywą ROC (ang. Receiver Operating Characteristic), czyli krzywej przedstawiającej zależność czułości (stosunek pozytywnych próbek zakwalifikowanych jako należące do sprawdzanej grupy do wszystkich próbek pozytywnych) od specyficzności (stosunek próbek negatywnych zakwalifikowanych do sprawdzanej grupy do wszystkich próbek negatywnych) dla różnych progów odcięcia. Pole pod krzywą ROC (AUC ROC, ang. Area Under Curve ROC) również może zostać wykorzystane jako wskaźnik efektywności modelu predykcyjnego – im lepszy model tym bardziej wartość AUC ROC zbliżona do 1 (wartość AUC ROC równa 1 oznacza klasyfikator idealny, czyli taki dla którego próg odcięcia klasyfikuje 100% próbek pozytywnych jako należące do sprawdzanej grupy i 100% próbek negatywnych jako nienależące do sprawdzanej grupy) [7].

2. BADANE JĘZYKI

W eksperymentach wykorzystałem dwa przykładowe języki przedstawione w artykule „*GA-based Learning of Context-Free Grammars using Tabular Representations*" [15] oraz jeden język obejmujący sekwencje aminokwasowe rodziny motywów białkowych.

2.1. PIERWSZY JĘZYK TESTOWY

Pierwszy wykorzystany język jest opisany wzorem: $\{\boldsymbol{a^n b^n c^m \mid m, n \geq 1}\}$ nad alfabetem: $\{a, b, c\}$.

Oznacza to, że wszystkie słowa należące do tego języka składają się z symboli: *a*, *b* oraz *c*, ułożonych w następującej sekwencji:

- na początku każdego słowa znajduje się *n* powtórzeń symbolu *a*,
- po symbolach „a" znajduje się taka sama liczba symboli *b*,
- każde słowo zakończone jest dowolną, niezerową liczbą powtórzeń symbolu *c*.

Gramatyka opisująca ten język, w postaci normalnej Chomsky'ego, składa się z:

- 3 symboli terminalnych: $\{a, b, c\}$,
- 6 symboli nieterminalnych: $\{S, T, V, A, B, C\}$, spośród których symbol *S* jest symbolem startowym,
- 8 reguł: $\{S \rightarrow TC,\ S \rightarrow SC,\ T \rightarrow AV,\ T \rightarrow AB,\ V \rightarrow TB,\ a \rightarrow a,\ B \rightarrow b,\ C \rightarrow c\}$.

Można zauważyć, że w przyjętej gramatyce, żaden z symboli nieterminalnych, zdolnych do generowania symbolu terminalnego nie jest zdolny do generowania innego symbolu nieterminalnego, jak również żaden z symboli nieterminalnych zdolnych do generowania innego symbolu nieterminalnego nie jest zdolny do generowania jakiegokolwiek symbolu terminalnego.

Ze względu na to, symbole nieterminalne można podzielić na:

- Symbole nieterminalne leksykalne (symbole nieterminalne generujące symbole terminalne): $\{A, B, C\}$,
- symbole nieterminalne strukturalne (symbole nieterminalne generujące symbole nieterminalne): $\{S, T, V\}$.

Analogiczne nazewnictwo stosuję w przypadku reguł, które zostały podzielone na reguły leksykalne (generujące symbole terminalne) oraz reguły strukturalne (generujące symbole nieterminalne).

Na potrzeby testowania maszynowego uczenia, dla powyższej gramatyki, wygenerowano gramatykę pokrywającą, czyli gramatykę zawierającą reguły umożliwiające generowanie dowolnego zestawu symboli nieterminalnych z każdego symbolu strukturalnego oraz każdego symbolu terminalnego z dowolnego symbolu leksykalnego [16].

Gramatyka pokrywająca została dodatkowo rozszerzona o jeden dodatkowy symbol strukturalny „U", w związku z czym zawierała 9 reguł leksykalnych oraz 196 reguł strukturalnych.

Gramatyka ta będzie w dalszej części pracy nazywana gramatyką testową 1, natomiast opisana wcześniej gramatyka przez nią rozszerzana: gramatyką docelową 1.

Podczas wszystkich testów wykonywanych na gramatyce testowej 1 lub gramatyce docelowej 1, jako zbiór uczący wykorzystano minimalny, wystarczający do nauczenia pierwszej gramatyki docelowej w wersji nieprobabilistycznej, zbiór próbek pozytywnych (należących do języka opisywanego przez tę gramatykę). Zbiór ten został wygenerowany za pomocą narzędzia dostępnego na stronie www: http://lukasz.culer.staff.iiar.pwr.edu.pl/gencreator.php i zawierał 7 sekwencji: {*abc, aabbcc, aaabbbc, abcc, abccc, aabbcc, aaaabbbb*c} [17].

Przy pomocy tego samego narzędzia wygenerowałem zbiór 100 sekwencji należących do sprawdzanego języka (rozłączny ze zbiorem uczącym) oraz 100 sekwencji do niego nienależących. Zbiory te zostały wykorzystane jako zbiór walidacyjny przy ocenie poprawności nauki.

Średnia długość sekwencji, w przypadku zbioru sekwencji pozytywnych wynosi 16.7, natomiast w przypadku sekwencji negatywnych 12,4. Liczba wystąpień symboli: *a, b* oraz *c* wynosi odpowiednio 418, 418 i 794 w przypadku zbioru sekwencji pozytywnych oraz 412, 408 i 421 w przypadku zbioru sekwencji negatywnych. Oba zbiory znajdują się w załączniku jako: „zbiór pozytywny, pierwszy język testowy" oraz „zbiór negatywny, pierwszy język testowy".

2.2. DRUGI JĘZYK TESTOWY

Drugi wykorzystany język jest opisany wzorem: $\{\boldsymbol{ac^m} \mid \boldsymbol{m \geq 1}\} \cup \{\boldsymbol{bc^m} \mid \boldsymbol{m \geq 1}\}$ nad alfabetem: {*a, b, c*}.

Oznacza to, że wszystkie słowa należące do tego języka składają się z symboli: *a* oraz *c* lub *b* oraz *c*, ułożonych w następującej sekwencji:

- na początku każdego słowa znajduje się jedno powtórzenie symbolu *a* lub *b*,
- po pierwszym symbolu występuje dowolna, niezerowa liczba powtórzeń symbolu *c*.

Gramatyka opisująca ten język, w postaci normalnej Chomsky'ego, składa się z:

- 3 symboli terminalnych: {*a, b, c*},
- 3 symboli nieterminalnych: {*S, T, C*}, spośród których symbol *S* jest symbolem startowym,
- 5 reguł: {$S \rightarrow TC$, $S \rightarrow SC$, $T \rightarrow a$, $T \rightarrow b$, $C \rightarrow c$}.

Podobnie jak w przypadku pierwszej gramatyki testowej i docelowej, symbole nieterminalne oraz reguły zostały podzielone na strukturalne oraz leksykalne.

Dla powyższej gramatyki, nazywanej od tej pory gramatyką docelową 2, wygenerowano 4 rozszerzające ją gramatyki pokrywające:

- bez dodatkowych symboli nieterminalnych, składająca się z 6 reguł leksykalnych oraz 9 reguł strukturalnych,
- z dodatkowym nieterminalnym symbolem strukturalnym: *U*, składająca się z 9 reguł leksykalnych oraz 16 reguł strukturalnych,
- z dodatkowym nieterminalnym symbolem leksykalnym: *A*, składająca się z 6 reguł leksykalnych oraz 32 reguł strukturalnych,
- z dodatkowymi dwoma nieterminalnymi symbolami strukturalnymi: *U* oraz *A*, składająca się z 6 reguł leksykalnych oraz 75 reguł strukturalnych.

Gramatyki te będą od teraz nazywane kolejno: gramatyką testową 2.1, gramatyką testową 2.2, gramatyką testową 2.3 oraz gramatyką testową 2.4.

Podobnie jak przy pierwszym języku testowym, za pomocą tego samego narzędzia [17], wygenerowano zbiory uczące i walidujące wykorzystywane później we wszystkich doświadczeniach przeprowadzanych dla gramatyk drugiego języka testowego.

Zbiór uczący składał się z pięciu sekwencji: {*ac, bc, acc, bcc, accc*}. Zbiór walidujący składał się z 10 sekwencji należących do języka i 100 sekwencji do niego nienależących.

Średnia długość sekwencji, w przypadku zbioru sekwencji pozytywnych wynosi 6,5, natomiast w przypadku sekwencji negatywnych 11,8. Liczba wystąpień symboli: *a*, *b* oraz *c* wynosi odpowiednio 5, 5 i 55 w przypadku zbioru sekwencji pozytywnych oraz 400, 381 i 401 w przypadku zbioru sekwencji negatywnych. Oba zbiory dostępne w załączniku jako: „zbiór pozytywny, drugi język testowy" oraz „zbiór negatywny, drugi język testowy".

2.3. JĘZYK SEKWENCJI RODZINY MOTYWÓW BIAŁKOWYCH

Badany w doświadczeniu motyw białkowy jest odpowiedzialny za wiązanie wapnia i manganu. Białka należące do badanej grupy pochodzą z lektyn roślin strączkowych [18]. Próbka pozytywna składała się z 24 sekwencji zawierających ten motyw białkowy, zaś próbka negatywna składała się ze 100 sekwencji niezawierających tego motywu. Wszystkie sekwencje należące zarówno do próbki pozytywnej jak i negatywnej mają długość 27 [4].

Ze względu na niewielką liczbę sekwencji, które zawierają badany motyw białkowy oraz to, że proces nauki gramatyki wymaga dużego i różnorodnego zbioru uczącego zastosowane metodę walidacji krzyżowej. Próbka pozytywna została podzielona na 4 części zawierające po 6 sekwencji, następnie utworzono 4 zestawy zbiorów uczących i walidujących, w których jedna część próbki pozytywnej wraz z próbką negatywną utworzyła zbiór walidujący, natomiast pozostałe pozytywne sekwencje stanowiły zbiór uczący.

Ze względu na to, że aminokwasy zwyczajowo oznaczane są wielkimi literami, w przypadku tej gramatyki symbole terminalne były przedstawiane przez wielkie litery alfabetu łacińskiego, natomiast symbole nieterminalne przez cyfry.

Na potrzeby maszynowego uczenia utworzyłem gramatykę pokrywającą wykorzystując architekturę, która została zastosowana w artykule „Estimating probabilistic context-free grammars for proteins using contact map constraints"[4] (ponieważ jak wykazuje przeprowadzony w nim eksperyment, wykorzystywana w tej architekturze liczba symboli nieterminalnych jest wystarczająca do uzyskania dobrego poziomu dopasowania gramatyki probabilistycznej do danych uczących). Gramatyka pokrywająca zawierała:

- 20 symboli terminalnych oznaczonych przez litery odpowiadające standardowym jednoliterowym oznaczeniom aminokwasów,
- 7 symboli nieterminalnych oznaczonych jako cyfry od 0 do 6, spośród których symbol oznaczony jako 0 jest jednocześnie symbolem startowym.

Symbole nieterminalne od 0 do 3 zostały wykorzystane jako nieterminalne symbole strukturalne, natomiast symbole od 4 do 6 jako nieterminalne symbole leksykalne.

3. ZAIMPLEMENTOWANY ALGORYTM EWOLUCYJNY

Wykorzystany w maszynowym uczeniu algorytm ewolucyjny został przeze mnie zaimplementowany w środowisku MATLAB i jest oparty na opisanym wcześniej prostym algorytmie genetycznym. Wartości wyznaczane podczas procesu uczenia to prawdopodobieństwa poszczególnych reguł sprawdzanej gramatyki.

Program jako dane wejściowe przyjmuje:

- zestaw reguł sprawdzanej gramatyki,
- zbiór sekwencji które mają zostać użyte jako zbiór uczący,
- zbiór sekwencji pozytywnych (należących do języka) i negatywnych (nienależących do języka), które łącznie tworzą zbiór walidujący,
- listę należących do danej gramatyki symboli terminalnych wraz z częstotliwością ich występowania względem siebie (wykorzystywane przy wyznaczaniu prawdopodobieństwa wygenerowania danej sekwencji przez model zerowy podczas procesu walidacji)
- parametry funkcji uczenia maszynowego (nazywane dalej: „hiperparametrami") do których należą: liczba osobników w populacji (n), prawdopodobieństwo mutacji (pm), maksymalna skala mutacji (sm) oraz prawdopodobieństwo krzyżowania (pk),
- czas zegarowy (w godzinach) po upływie którego program zapisuje dotychczasowe wyniki (można podać kilka wartości).

Każdy osobnik jest wektorem liczb rzeczywistych z przedziału zamkniętego od 0 do 1, przy czym długość wektora odpowiada liczbie reguł w danej gramatyce. Podczas inicjowania populacji początkowej program losuje prawdopodobieństwa reguł dla wszystkich osobników jako losową liczbę z rozkładu równomiernego w przedziale od 0 do 1.

W przypadku probabilistycznych gramatyk bezkontekstowych konieczne jest zadbanie o to, aby suma prawdopodobieństw wszystkich reguł o tym samym poprzedniku wynosiła 1. Losowo przypisane prawdopodobieństwa reguł zazwyczaj nie spełniają tego warunku. Aby poradzić sobie z tym problemem, program po rozlosowaniu prawdopodobieństw odpowiednio je skaluje (poprzez dzielenie prawdopodobieństwa sprawdzanej reguły P(r) przez sumę prawdopodobieństw wszystkich reguł mających tego samego poprzednika).

Przystosowanie poszczególnych osobników jest wyznaczane jako logarytm naturalny ze średniego prawdopodobieństwa Pśr z jakim sekwencje ze zbioru uczącego są generowane przez gramatykę ln(Pśr). Analiza składniowa, w wyniku której wyznaczane są prawdopodobieństwa z jakimi każda sekwencja jest generowana przez gramatykę probabilistyczną, odbywa się według opisanego wcześniej algorytmu CKY. Ze względu na to, że analiza składniowa jest najbardziej czasochłonną częścią programu oraz to, że przystosowania poszczególnych osobników są od siebie całkowicie niezależne, program umożliwia równoległe wyznaczanie funkcji przystosowania dla wielu osobników jednocześnie przy wykorzystaniu kilku rdzeni procesora (wszystkie testy opisane w pracy zostały wykonane przy użyciu 20 rdzeni co umożliwiło równoległe wyznaczanie funkcji przystosowania dla 20 osobników).

Proces reprodukcji przebiega zgodnie z założeniami prostego algorytmu genetycznego, jednak tworzona w jego wyniku populacja (nazywana od tej pory „populacją T1") jest o połowę mniejsza od populacji bazowej. Populacja T1 jest następnie poddawana mutacjom (Rys. 3.1). Ze względu na to, że prawdopodobieństwa reguł nie są wartościami binarnymi, podczas procesu

mutacji, dla każdej cechy dla której doszło do mutacji, losowana jest liczba z przedziału od 0,01 do maksymalnej skali mutacji (jeden z hiperparametrów programu, Rys. 3.1). Następnie, z jednakowym prawdopodobieństwem, wylosowana liczba jest dodawana lub odejmowana od aktualnego prawdopodobieństwa danej reguły (Rys. 3.1, linia 8 - 17), jeżeli ma dojść do odjęcia wylosowanej liczby, która jest większa od aktualnego prawdopodobieństwa danej reguły to jest ona zerowana (Rys. 3.1, linia 15).

```
1   function [population] = mutations(population,mutationProb,mutationScale)
2   [n,m] = size(population);
3   for i=1:n
4       for j=1:m
5           x=rand;
6           if x<mutationProb
7               z=0.01+rand*(mutationScale-0.01);
8               y = randi([1 2]);
9               if y==1
10                  population(i,j)=population(i,j)+z;
11              else
12                  if population(i,j)>z
13                      population(i,j)=population(i,j)-z;
14                  else
15                      population(i,j)=0;
16                  end
17              end
18          end
19      end
20  end
21  end
```

Rysunek 3.1 – Funkcja odpowiadająca za przeprowadzanie mutacji na danej populacji. Funkcja jako dane wejściowe przyjmuje: population - populację na której przeprowadza mutacje, mutationProb – prawdopodobieństwo zajścia mutacji, mutationScale – maksymalną skalę mutacji. Funkcja zwraca populację po przeprowadzeniu mutacji.

W wyniku zerowania niektórych reguł może dojść do sytuacji, w której dany osobnik zaczyna generować jedno ze słów z próbki uczącej z prawdopodobieństwem równym 0, mutacje takie zostają uznane za krytycznie niekorzystne, a osobnik ten nie zostaje przeniesiony do kolejnej populacji (zostaje zastąpiony przez swoją kopię sprzed mutacji).

Po procesie reprodukcji i mutacji do powstałej w ten sposób populacji T1 dodawana jest lepiej przystosowana połowa poprzedniej populacji (w niezmienionej formie), a powstała w ten sposób populacja stanowi populację bazową w kolejnym cyklu nauki.

Proces nauki zostaje zakończony po przejściu przez zadaną jako parametr wejściowy algorytmu liczbę cykli.

Podczas nauki program zapisuje w pamięci takie parametry jak:

- średnie prawdopodobieństwo z jakim sekwencje z próbki uczącej są generowane przez gramatykę zdefiniowaną przez tego osobnika,
- różnorodność populacji (rozumiana jako średnia z odległości euklidesowych każdego osobnika względem wszystkich pozostałych osobników w populacji),
- czas zegarowy jaki upłynął w poszczególnych cyklach nauczania,

- liczba osobników u których doszło do mutacji krytycznie niekorzystnych,
- najlepiej przystosowany osobnik z każdej populacji (tworzą one historię najlepiej przystosowanych osobników),
- co 100 cykli zapisywana jest cała populacja (zapisane w ten sposób populacje tworzą historię populacji).

Po zakończeniu procesu nauczania przeprowadzany jest proces oceny efektów nauczania (walidacji). W tym celu, dla co dziesiątego pokolenia z historii najlepszych osobników, wyznaczane są prawdopodobieństwa z jakimi sekwencje ze zbioru walidacyjnego (utworzonego z połącznia próbki pozytywnej i negatywnej, które są parametrami wejściowymi programu) są generowane przez gramatykę zdefiniowaną przez tego osobnika. Na podstawie uzyskanych w ten sposób prawdopodobieństw wykreślana jest krzywa ROC dla której wyznaczane jest ograniczone przez nią pole pod krzywą.

4. PRZEPROWADZONE EKSPERYMENTY

4.1. METODOLOGIA PRZEPROWADZANYCH EKSPERYMENTÓW

We wszystkich przeprowadzonych eksperymentach obliczeniowych, podstawowym parametrem, sprawdzanym podczas nauki, w celu określenia jak dobrze dopasowana do próbki uczącej jest gramatyka w danym momencie, jest przystosowanie najlepszego osobnika w populacji, czyli: „**best(ln(Pśr))**". Maksymalna wartość przystosowania osiągana podczas całego testu jest oznaczana jako: „**max(best(ln(Pśr)))**". Ze względu na to, że szybkość nauki spada wraz ze zbliżaniem się do rozwiązania optymalnego, w celu określenia tempa nauki sprawdzano czas (zegarowy oraz cyklach nauki) po którym best(ln(Pśr)) osiągało wartość 95% max(best(ln(Pśr))), czas ten został określony jako: „**czas efektywnego uczenia**".

Przy określaniu jakości uzyskanych podczas uczenia gramatyk w kontekście pełnienia funkcji klasyfikatora skorzystano z modelu zerowego. Oznacza to, że dla każdej sekwencji ze zbioru walidacyjnego wyznaczano stosunek prawdopodobieństwa z jakim sekwencja ta jest wyprowadzana przez gramatykę, do prawdopodobieństwa z jakim sekwencja ta może zostać wygenerowana przez model losowy. Zastosowany model losowy może uwzględniać różnice w prawdopodobieństwie występowania poszczególnych znaków tworzących sprawdzane łańcuchy (dla gramatyk testowych przyjęto, że prawdopodobieństwa wszystkich symboli są jednakowe, zaś dla aminokwasów wykorzystano częstość z jaką poszczególne aminokwasy występują w bazie danych UniProtKB/Swiss-Prot) [19].
Zastosowanie modelu zerowego umożliwia przeprowadzanie walidacji z wykorzystaniem próbek o różnych długościach (prawdopodobieństwo dłuższych sekwencji bez zastosowania modelu zerowego byłoby zazwyczaj o wiele mniejsze niż prawdopodobieństwo wyprowadzenia krótkich sekwencji, ze względu na konieczność zastosowania większej liczby reguł).

4.2. WPŁYW HIPERPARAMETRÓW ALGORYTMU EWOLUCYJNEGO NA PRZEBIEG I WYNIKI NAUKI

W pierwszej części eksperymentów zbadałem, w jaki sposób zmiana wartości hiperparametrów algorytmu uczącego wpływa na przebieg nauki i jej końcowe wyniki. W tym celu wykonałem szereg testów (po 3 dla każdego zestawu) dla gramatyki testowej 1, z różnymi zestawami hiperparametrów, przedstawionymi w Tabeli 4.1.

Tabela 4.1 - Wykorzystane zestawy hiperparametrów.

l.p.	liczba osobników	prawdopodobieństwo mutacji	maksymalna skala mutacji	prawdopodobieństwo krzyżowania
1	40	0,001	0,5	0,9
2				0,1
3				0,5
4			0,9	
5			0,1	
6		0,0001	0,5	
7		0,01		
8	80	0,001	0,5	0,9
9				0,1
10				0,5
11			0,9	
12			0,1	
13		0,0001	0,5	
14		0,01		
15	160	0,001	0,5	0,9
16				0,1
17				0,5
18			0,9	
19			0,1	
20		0,0001	0,5	
21		0,01		

Przed przeprowadzeniem testów dla pierwszej gramatyki testowej, w celu poznania wartości prawdopodobieństw poszczególnych reguł jak najbardziej zbliżonych do wartości optymalnych, przeprowadzono 3 próby uczenia dla pierwszej gramatyki docelowej (hiperparametry ustawione jak w zestawie nr 3, liczba cykli po której zakończono naukę ustawiona na 1000).

We wszystkich próbach, końcowa wartość średniego prawdopodobieństwa z jakim zdania ze zbioru uczącego należą do sprawdzanego języka (Pśr), była praktycznie identyczna i po zlogarytmowaniu wynosiła: ln(best(Pśr)) = -2.55.

Końcowe wartości prawdopodobieństw dla poszczególnych reguł we wszystkich testach również ostatecznie były do siebie bardzo podobne (Tabela 4.2).

Tabela 4.2 - Końcowe prawdopodobieństwa przypisane do poszczególnych reguł.

reguła	test 1	test 2	test 3
S→TC	0,616	0,615	0,616
S→SC	0,384	0,385	0,384
T→AV	0,530	0,529	0,528
T→AB	0,470	0,471	0,472
V→TB	1,000	1,000	1,000
A→a	1,000	1,000	1,000
B→b	1,000	1,000	1,000
C→c	1,000	1,000	1,000

Dla wszystkich przeprowadzonych testów sprawdzano:

- maksymalną wartość best(ln(Pśr))- w danym teście,
- czas efektywnego uczenia [cykle],
- czas efektywnego uczenia [minuty],
- po jak wielu cyklach klasyfikator staje się bliski idealnemu (pole pod krzywą ROC większe lub równe 0,95),
- po ilu minutach klasyfikator staje się bliski idealnemu,
- po ilu cyklach każdy nieterminalny symbol leksykalny posiada tylko jedną regułę o niezerowym prawdopodobieństwie,
- jak wiele reguł, u najlepiej przystosowanego osobnika populacji, zostało wyzerowane w cyklu w którym best(ln(Pśr)) osiąga 95% maksymalnej wartości.

Wyniki dla wszystkich testów można znaleźć w Tabeli 1 w załączniku.

Testy dla których maksymalna wartość best(ln(Pśr)) jest większa lub równa -2.8 uznałem za udane, ponieważ jest to wynik zbliżony do uzyskanego w testach z wykorzystaniem gramatyki docelowej 1. Część testów osiągnęła jednak maksymalną wartość best(ln(Pśr))

mniejszą niż -2.8, jest to spowodowane tym, że podczas procesu uczenia populacja osiadła w lokalnym maksimum, które było znacznie poniżej maksimum globalnego, takie testy uznałem za nieudane.

W populacjach początkowych średnia wartość best(ln(Pśr)) wynosiła około -9,5, najniższa końcowa wartość best(ln(Pśr)) uzyskana w teście uznanym za nieudany to -4,28 (uzyskana w teście nr 3 dla 6 zestawu hiperparametrów), zaś 18 z 23 nieudanych testów osiągnęła wartość best(ln(Pśr)) nie mniejszą niż -3.9. Pokazuje to, że nawet pomimo utknięcia populacji w maksimum lokalnym, nauka spowodowała zwiększenie wartości best(ln(Pśr)) o kilka rzędów.

Warto również zwrócić uwagę, że w zdecydowanej większości przypadków (21 spośród 23), gramatyki uzyskane w wyniku testów uznanych za nieudane mogą wciąż pełnić rolę bardzo dobrych klasyfikatorów przynależności do języka. Świadczy o tym to, że po pewnej liczbie cykli pole pod krzywą ROC osiąga wartość co najmniej 0,95.

W Tabeli 2 znajdującej się w załączniku, przedstawiono ile testów dla każdego zestawu parametrów, zostało uznane za udane oraz średnie wartości najważniejszych parametrów wraz z odchyleniem standardowym (przy liczeniu średniej i odchylenia standardowego brane pod uwagę były wyłącznie testy uznane za udane).

4.2.1. Wpływ prawdopodobieństwa mutacji

W celu zbadania wpływu prawdopodobieństwa mutacji na przebieg i wyniki maszynowego uczenia wydzieliłem spośród wszystkich sprawdzanych zestawów trzy grupy (ze względu na rozmiar populacji). W obrębie danej grupy wszystkie zestawy różniły się względem siebie tylko sprawdzanym parametrem. Utworzone grupy to:

- dla populacji liczącej 40 osobników – testy dla zestawów nr: 3, 6 oraz 7,
- dla populacji liczącej 80 osobników – testy dla zestawów nr: 10, 13 oraz 14,
- dla populacji liczącej 160 osobników – testy dla zestawów nr: 17, 20 oraz 21.

Parametrami porównywanymi we wszystkich testach były:

- liczba testów uznanych za udane,
- maksymalna wartości best(ln(Pśr)),
- czas efektywnej nauki

W Tabelach 4.3 – 4.6 przedstawiłem uśrednione wyniki dla testów pozytywnych z poszczególnych grup oraz wyniki zbiorcze dla zestawów o tym samym prawdopodobieństwie mutacji (z wyłączeniem czasu efektywnej nauki, ponieważ parametr ten jest ściśle powiązany z liczbą osobników w populacji, w związku z czym należy go analizować dla każdej liczebności populacji osobno).

Tabela 4.3 - Porównanie wyników testów dla zestawów o różnym prawdopodobieństwie mutacji, liczba osobników w populacji = 40.

prawdopodobieństwo mutacji	**0,01**	**0,001**	**0,0001**
liczba testów	3	3	3
liczba udanych testów	2	3	0
odsetek udanych testów	66,7%	100%	0%
max(best(ln(Pśr)))	-2,55	-2,48	-
czas efektywnej nauki [cykle]	820,0	1256,0	-
czas efektywnej nauki [min]	6,1	9,9	-

Tabela 4.4 - Porównanie wyników testów dla zestawów o różnym prawdopodobieństwie mutacji, liczba osobników w populacji = 80.

prawdopodobieństwo mutacji	**0,01**	**0,001**	**0,0001**
liczba testów	3	3	3
liczba udanych testów	1	1	1
odsetek udanych testów	33,3%	33,3%	33,3%
max(best(ln(Pśr)))	-2,48	-2,40	-2,68
czas efektywnej nauki [cykle]	529,0	1458,0	2559,0
czas efektywnej nauki [min]	6,7	17,3	32,4

Tabela 4.5 - Porównanie wyników testów dla zestawów o różnym prawdopodobieństwie mutacji, liczba osobników w populacji = 160.

prawdopodobieństwo mutacji	**0,01**	**0,001**	**0,0001**
liczba testów	3	3	3
liczba udanych testów	3	2	3
odsetek udanych testów	100%	66,7%	100%
max(best(ln(Pśr)))	-2,50	-2,50	-2,50
czas efektywnej nauki [cykle]	620,0	791,0	2138,3
czas efektywnej nauki [min]	13,1	18,0	44,1

Tabela 4.6 - Porównanie wyników testów dla zestawów o różnym prawdopodobieństwie mutacji, wyniki zbiorcze dla zestawów o tym samym prawdopodobieństwie mutacji ze wszystkich trzech grup.

prawdopodobieństwo mutacji	**0,01**	**0,001**	**0,0001**
liczba testów	9	9	9
liczba udanych testów	6	6	4
odsetek udanych testów	66,7%	66,7%	44,4%
max(best(ln(Pśr)))	-2,52	-2,47	-2,55

Odsetek udanych testów jest wyższy, gdy prawdopodobieństwo mutacji wynosiło 0,01 lub 0,001, co wskazuje, że zbyt niskie prawdopodobieństwo mutacji (0,0001) jest niekorzystne dla procesu uczenia i sprzyja osiadaniu w lokalnych maksimach. Populacja o wysokiej liczbie osobników (160) wydaje się radzić sobie lepiej, od mniej licznych populacji, z wysokim (0,01) oraz niskim (0,001) prawdopodobieństwem mutacji. Dla wszystkich rozmiarów populacji wyraźnie widać, że nauka przebiega tym szybciej, im prawdopodobieństwo mutacji jest większe. Średnie wyniki max(best(ln(Pśr))) są zbliżone dla wszystkich sprawdzanych prawdopodobieństw mutacji.

4.2.2. Wpływ maksymalnej skali mutacji

W celu zbadania wpływu maksymalnej skali mutacji na przebieg i wyniki maszynowego uczenia, analogicznie jak w punkcie 4.2.2, wydzieliłem spośród wszystkich testów trzy grupy:

- dla populacji liczącej 40 osobników – testy dla zestawów nr: 3, 4 oraz 5,
- dla populacji liczącej 80 osobników – testy dla zestawów nr: 10, 11 oraz 12,

- dla populacji liczącej 160 osobników – testy dla zestawów nr: 17, 18 oraz 19.

Wyniki wszystkich testów zostały zebrane w Tabelach 4.7 – 4.10 (w sposób analogiczny do tego z punktu 4.1.1).

Tabela 4.7 - Porównanie wyników testów dla zestawów o różnych maksymalnych skalach mutacji, liczba osobników w populacji = 40.

maksymalna skala mutacji	**0,1**	**0,5**	**0,9**
liczba testów	3	3	3
liczba udanych testów	2	3	1
odsetek udanych testów	66,7%	100%	33,3%
max(best(ln(Pśr)))	-2,46	-2,48	-2,55
czas efektywnej nauki [cykle]	2602,0	1256,0	767,0
czas efektywnej nauki [min]	19,5	9,9	6,7

Tabela 4.8 - Porównanie wyników testów dla zestawów o różnych maksymalnych skalach mutacji, liczba osobników w populacji = 80

maksymalna skala mutacji	**0,1**	**0,5**	**0,9**
liczba testów	3	3	3
liczba udanych testów	0	1	2
odsetek udanych testów	0%	33,3%	66,7%
max(best(ln(Pśr)))	-	-2,40	-2,48
czas efektywnej nauki [cykle]	-	1458,0	1131,0
czas efektywnej nauki [min]	-	17,3	13,9

Tabela 4.9 - Porównanie wyników testów dla zestawów o różnych maksymalnych skalach mutacji, liczba osobników w populacji = 160

maksymalna skala mutacji	**0,1**	**0,5**	**0,9**
liczba testów	3	3	3
liczba udanych testów	3	2	2
odsetek udanych testów	100%	66,7%	66,7%
max(best(ln(Pśr)))	-2,53	-2,50	-2,44
czas efektywnej nauki [cykle]	1840,3	791,0	876,0
czas efektywnej nauki [min]	38,0	18,0	18,3

Tabela 4.10 - Porównanie wyników testów dla zestawów o różnych maksymalnych skalach mutacji, wyniki zbiorcze dla zestawów o tej samej maksymalnej skali mutacji ze wszystkich trzech grup.

maksymalna skala mutacji	**0,1**	**0,5**	**0,9**
liczba testów	9	9	9
liczba udanych testów	5	6	5
odsetek udanych testów	55,6%	66,7%	55,6%
max(best(ln(Pśr)))	-2,50	-2,47	-2,48

Odsetek udanych testów i maksymalna wartość max(best(ln(Pśr))) są minimalnie wyższe w przypadku maksymalnej skali mutacji równej 0,5. W grupach 1 i 2 wyraźnie widać, że zwiększenie maksymalnej skali mutacji zmniejsza czas efektywnej nauki. W przypadku grupy trzeciej widać to przy zmianie skali mutacji z 0,1 na 0,5, jednak przy zmianie skali z 0,5 do 0,9 czas nauki wydłuża się. Bardzo niska maksymalna skala mutacji (0,1) wydaje się mieć niepożądany wpływ na proces uczenia wydłużając go i pogarszając ostateczne wyniki. Tempo nauki oraz maksymalna wartość z best(ln(Pśr)) uzyskiwane dla testów o maksymalnej skali mutacji 0,5 oraz 0,9 są do siebie zbliżone.

4.2.3. Wpływ prawdopodobieństwo krzyżowania

W celu zbadania wpływu maksymalnej skali mutacji na przebieg i wyniki maszynowego uczenia, analogicznie jak w punkcie 4.1.2, wydzieliłem spośród wszystkich testów trzy grupy:

- dla populacji liczącej 40 osobników – testy dla zestawów nr: 1, 2 oraz 3,
- dla populacji liczącej 80 osobników – testy dla zestawów nr: 8, 9 oraz 10,
- dla populacji liczącej 160 osobników – testy dla zestawów nr: 15, 16 oraz 17.

Wyniki wszystkich testów zostały zebrane w Tabelach 4.11 – 4.14 (w sposób analogiczny do tego z punktu 4.1.2).

Tabela 4.11 - porównanie wyników testów dla zestawów o różnym prawdopodobieństwie krzyżowania, liczba osobników w populacji = 40.

prawdopodobieństwo krzyżowania	**0,1**	**0,5**	**0,9**
liczba testów	3	3	3
liczba udanych testów	3	3	3
odsetek udanych testów	100%	100%	100%
max(best(ln(Pśr)))	-2,48	-2,48	-2,49
czas efektywnej nauki [cykle]	1372,7	1256,0	1356,7
czas efektywnej nauki [min]	10,4	9,9	10,7

Tabela 4.12 – Porównanie wyników testów dla zestawów o różnym prawdopodobieństwie krzyżowania, liczba osobników w populacji = 80.

prawdopodobieństwo krzyżowania	**0,1**	**0,5**	**0,9**
liczba testów	3	3	3
liczba udanych testów	2	1	3
odsetek udanych testów	66,7%	33,3%	100%
max(best(ln(Pśr)))	-2,44	-2,40	-2,50
czas efektywnej nauki [cykle]	1260,5	1458,0	954,3
czas efektywnej nauki [min]	15,1	17,3	12,3

Tabela 4.13 - Porównanie wyników testów dla zestawów o różnym prawdopodobieństwie krzyżowania, liczba osobników w populacji = 160.

prawdopodobieństwo krzyżowania	**0,1**	**0,5**	**0,9**
liczba testów	3	3	3
liczba udanych testów	1	2	2
odsetek udanych testów	33,3%	66,7%	66,7%
max(best(ln(Pśr)))	-2,55	-2,50	-2,51
czas efektywnej nauki [cykle]	817,0	791,0	1129,0
czas efektywnej nauki [min]	19,0	18,0	24,7

Tabela 4.14 - Porównanie wyników testów dla zestawów o różnym prawdopodobieństwie krzyżowania, wyniki zbiorcze dla zestawów o tym samym prawdopodobieństwie krzyżowania ze wszystkich trzech grup.

prawdopodobieństwo krzyżowania	**0,1**	**0,5**	**0,9**
liczba testów	9	9	9
liczba udanych testów	6	6	8
odsetek udanych testów	66,7%	66,7%	88,9%
max(best(ln(Pśr)))	-2,48	-2,47	-2,50

Testy z największym prawdopodobieństwem krzyżowania mają najwyższy odsetek udanych testów, jednak wartości best(ln(Pśr)) we wszystkich grupach są do siebie zbliżone, nie widać również żadnego widocznego powiązania pomiędzy prawdopodobieństwem krzyżowania a czasem efektywnej nauki.

4.2.4. Wpływ liczby osobników w populacji

Porównując wyniki odpowiadających sobie zestawów (czyli zestawów w których sprawdzany parametr miał tę samą wartość) z grup 1, 2 i 3 w doświadczeniach 4.1.1, 4.1.2 oraz 4.1.3 można uzyskać informacje na temat tego, jak zmiana liczby osobników w populacji wypływa na przebieg i wyniki nauki.

Dodatkowo, w Tabeli 4.15, przedstawiłem wyniki zbiorcze ze wszystkich symulacji wykonywanych dla danej liczebności populacji.

Tabela 4.15 - Porównanie wyników testów dla zestawów o różnej liczebności populacji.

liczba osobników	**40**	**80**	**160**
liczba testów	21	21	21
liczba udanych testów	14	10	16
odsetek udanych testów	66,67%	47,62%	76,19%
max(best(ln(Pśr)))	-2,50	-2,49	-2,50
czas efektywnej nauki [cykle]	1397,6	1219,2	1262,8
czas efektywnej nauki [min]	10,8	15,1	26,7

Odsetek udanych testów jest najwyższy dla testów, w których populacja liczyła 160 osobników, a najniższy przy populacjach liczących 80 osobników. Pokazuje to, że, w przypadku gramatyki testowej 1, skłonność wykorzystywanego narzędzia do osiadania w maksimach lokalnych nie jest silnie powiązana z liczbą osobników w populacji. Ponadto widać, że populacja licząca 40 osobników jest wystarczająco liczna, aby uzyskać wyniki, które będą, z dużym prawdopodobieństwem, bardzo zbliżone do optymalnych.

Średnia wartość max(best(ln(Pśr))) z udanych testów jest bardzo zbliżona dla wszystkich sprawdzanych liczebności populacji, co pokazuje, że zwiększanie liczby osobników nie ma wyraźnego wpływu na końcowy efekt procesu nauczania dla gramatyki testowej 1.

Liczba cykli czasu efektywnej nauki, jest przeważnie największa dla testów z populacjami liczącymi 40 osobników. Jednakże czas trwania każdego cyklu jest zależny od liczby osobników w populacji, w związku z czym, czas zegarowy, potrzebny, aby best(ln(Pśr)) osiągnęło 0,95 maksymalnej wartości, we wszystkich przeprowadzonych testach jest zauważalnie najmniejszy dla populacji liczącej 40 osobników.

4.2.5. Przykładowe gramatyki uzyskane dla pierwszego języka testowego

Spośród wszystkich wykonanych symulacji najwyższe wartości max(best(ln(Pśr))) uzyskana została w:

- teście nr 2 dla 1 zestawu parametrów,
- teście nr 2 dla 2 zestawu parametrów,
- teście nr 3 dla 5 zestawu parametrów,
- teście nr 1 dla 9 zestawu parametrów,
- teście nr 3 dla 18 zestawu parametrów.

Dla uzyskanych w tych testach gramatyk końcowych zebrano i zestawiono w Tabelach 4.16.

Tabela 4.16 – Gramatyki końcowe uzyskane w najlepszych testach.

zestaw 1 test 2		zestaw 2 test 2		zestaw 5 test 3		zestaw 9 test 3		zestaw 18 test 1	
reguła	P(r	reguła	P(r	reguła	P(r	reguła	P(r	reguła	P(r
S→SA	0,062	S→SC	0,112	S→SA	0,166	S→SB	0,126	S→SA	0,164
S→TA	0,146	S→TC	0,155	S→VA	0,834	S→TB	0,658	S→TA	0,223
S→VA	0,792	S→UC	0,733	T→CU	0,430	S→VB	0,216	S→UA	0,613
T→BU	0,442	T→AV	0,434	T→CB	0,570	T→TB	0,102	T→CV	0,439
T→BC	0,558	T→AB	0,566	U→TB	1,000	T→VB	0,477	T→CB	0,561
U→TC	1,000	U→TC	0,410	V→TA	0,327	T→AU	0,421	U→TA	0,495
V→TA	0,396	U→UC	0,118	V→VA	0,066	U→VC	1,000	U→UA	0,054
V→VA	0,158	U→AV	0,422	V→CU	0,439	V→AU	0,439	U→CV	0,451
V→BU	0,386	U→AB	0,050	V→CB	0,167	V→AC	0,561	V→TB	1,000
V→BC	0,060	V→TB	1,000	A→c	1,000	A→a	1,000	A→c	1,000
A→c	1,000	A→a	1,000	B→b	1,000	B→c	1,000	B→b	1,000
B→a	1,000	B→b	1,000	C→a	1,000	C→b	1,000	C→a	1,000
C→b	1,000	C→c	1,000	-		-		-	

Liczba reguł o prawdopodobieństwie większym niż 0% we wszystkich gramatykach jest podobna. We wszystkich gramatykach każdy nieterminalny symbol leksykalny wyprowadza tylko jeden symbol terminalny. Uzyskane gramatyki są do siebie podobne, jednak ich analiza jest utrudniona ze względu na to, że w poszczególnych gramatykach różne symbole nieterminalne pełnią analogiczne funkcje (przykładowo symbol terminalny *a* jest wyprowadzany w niektórych gramatykach przez *A*, natomiast w innych przez *B*).

W celu ułatwienia analizy podobieństwa poszczególnych gramatyk można zmienić symbole za pomocą których przedstawiane są konkretne symbole nieterminalne ze względu na pełnione przez nie funkcje w gramatyce. Przykładowo, dla gramatyk uzyskanych w wyniku testu 3 dla zestawu 9 i testu 2 dla zestawu 2 dokonałem następujących transformacji:

- symbol nieterminalny pełniący funkcję symbol startowego przedstawiłem jako 0,
- nieterminalny symbol leksykalny wyprowadzający terminal *a* przedstawiłem jako 1,
- nieterminalny symbol leksykalny wyprowadzający terminal *a* przedstawiłem jako 2,
- nieterminalny symbol leksykalny wyprowadzający terminal *a* przedstawiłem jako 3,
- nieterminalny symbol strukturalny wyprowadzany z symbolu startowego w regule z największym prawdopodobieństwem przedstawiłem jako 4,
- nieterminalny symbol strukturalny wyprowadzany z symbolu startowego w regule z drugim największym prawdopodobieństwem przedstawiłem jako 5,
- nieterminalny symbol strukturalny wyprowadzany z symbolu startowego w regule z drugim największym prawdopodobieństwem przedstawiłem jako 6.

Uzyskane w ten sposób gramatyki probabilistyczne zostały przedstawione w Tabeli 4.17.

Tabela 4.17 - Gramatyki końcowe uzyskane w najlepszych symulacjach z zestawów 2 i 9 po zmapowaniu.

zestaw 2 test 2		zestaw 9 test 3	
reguła	P(r)	reguła	P(r)
0→03	0,112	0→03	0,126
0→53	0,155	0→53	0,216
0→43	0,733	0→43	0,658
6→52	1,000	6→52	1,000
5→16	0,434	5→16	0,439
5→12	0,566	5→12	0,561
4→43	0,118	4→43	0,102
4→53	0,410	4→16	0,421
4→16	0,422	4→53	0,477
3→c	1,000	3→c	1,000
2→b	1,000	2→b	1,000
1→a	1,000	1→a	1,000
4→12	0,050	-	

Wszystkie sprawdzane gramatyki końcowe opisują język testowy 1. Pokazuje to, że stworzony algorytm bardzo dobrze radzi sobie z ograniczaniem bardzo obszernych gramatyk pokrywających (205 reguł) do postaci o bardzo małej liczbie produkcji.

4.3. WPŁYW LICZBY SYMBOLI NIETERMINALNYCH NA WYNIKI NAUKI

W drugiej części eksperymentów obliczeniowych zbadałem wpływ liczby nieterminalnych symboli strukturalnych oraz leksykalnych na złożoność gramatyki będącej wynikiem uczenia maszynowego.

Cechą która sprawiła, że do tego eksperymentu wybrano język testowy 2 jest niewielka minimalna liczba symboli nieterminalnych, wystarczająca do stworzenia gramatyki opisującej go. Ze względu na tą właściwość gramatyka docelowa 2 może być łatwo rozszerzana o dodatkowe symbole nieterminalne.

Po przeanalizowaniu wyników pierwszej części eksperymentu zdecydowałem się wykorzystać zestaw hiperparamterów, który został oznaczony w poprzedniej części doświadczenia jako zestaw nr 3.

Korzystając z tego zestawu hiperparametrów przeprowadziłem po 3 testy dla czterech gramatyk testowych:

- gramatyki testowej 2.1 (G2.1),
- gramatyki testowej 2.2 (G2.2),
- gramatyki testowej 2.3 (G2.3),
- gramatyki testowej 2.4 (G2.4).

Struktury wszystkich gramatyk testowych (liczba występujących w niej symboli i reguł) zostały przedstawione w Tabeli 4.20.

Tabela 4.18 – Struktura wykorzystanych gramatyk testowych.

	Terminale	Nieterminale	Nieterminale leks.	Nieterminale str.	Reguły	Reguły leks.	Reguły str.
G2.1	3	3	2	1	15	6	9
G2.2	3	4	3	1	25	9	16
G2.3	3	4	2	2	38	6	32
G2.4	3	5	2	3	81	6	75

Dla każdego testu sprawdzałem jaka jest maksymalna uzyskana wartość best(ln(Pśr)) oraz dla jak wielu reguł po zakończeniu nauki jest przypisane prawdopodobieństwo:

- równe 0%,
- mniejsze od 1%,
- mniejsze od 10%,
- większe od 10%,
- większe od 30%,
- równe 100%.

Wyniki tych testów zostały zestawione w Tabeli 4.16.

Tabela 4.19 - Zestawienie wyników wszystkich testów dla drugiego języka testowego.

gramatyka	nr testu	max (best(ln(Pśr)))	r(P=0%)	r(P<1%)	r(P<10%)	r(P>10%)	r(P>30%)	r(P=100%)
	1	-1,91	10	10	10	5	5	1
	2	-1,91	10	10	10	5	5	1
G2.1	3	-1,91	10	10	10	5	5	1
	1	-1,91	17	17	19	6	6	1
	2	-1,91	17	17	17	8	5	1
G2.2	3	-1,91	17	17	18	7	6	2
	1	-1,78	30	30	30	8	6	1
	2	-1,78	30	30	30	8	6	1
G2.3	3	-1,78	29	29	31	7	6	1
	1	-1,78	56	61	72	9	7	1
	2	-1,91	43	49	72	9	7	1
G2.4	3	-1,73	72	72	73	8	8	2

Wyniki max (best(ln(Pśr))) uzyskane dla gramatyki G2.1 są praktycznie co do wartości do wyników uzyskanych dla gramatyki G2.2 co świadczy o tym, że wprowadzenie dodatkowego nieterminalnego symbolu leksykalnego nie wpłynęło pozytywnie na wyniki nauki. Wyniki uzyskane dla gramatyk G2.3 oraz G2.4 bardzo podobne do siebie i zauważalnie lepsze niż dla gramatyk G2.1 i G2.2, pokazuje to, że rozszerzenie gramatyki o dodatkowy nieterminal

strukturalny jest korzystne dla procesu nauki, jednak przy większej liczbie nieterminalnych symboli strukturalnych zysk z dodania kolejnego staje się o wiele mniejszy.

We wszystkich testach udało się uzyskać gramatyki mogące pełnić rolę idealnych klasyfikatorów (AUC ROC równe 1) w bardzo krótkim czasie nauki (poniżej 50 cykli nauki). Jest to spowodowane bardzo prostą strukturą zdań tworzących język opisywany przez gramatykę docelową.

Gramatyki końcowe uzyskane w teście o największej wartości max (best(ln(Pśr))), dla każdej sprawdzanej gramatyki, zostały przedstawione w Tabeli 4.20.

Tabela 4.20 – Gramatyki końcowe uzyskane dla testów w których max(best(ln(Pśr))) osiągnęło największą wartość.

G2.1 test 1		G2.2 test 1		G2.3 test 2		G2.4 test 2	
reguła	P(r)	reguła	P(r)	reguła	P(r)	reguła	P(r)
S→ST	0,445	S→SC	0,444	S→SC	0,127	S→SA	0,445
S→CT	0,555	S→UC	0,556	S→TC	0,480	S→CA	0,555
T→c	1,000	T→c	0,840	S→AC	0,392	T→CA	0,408
C→a	0,599	U→a	0,600	T→TC	0,178	T→AA	0,289
C→b	0,401	U→b	0,400	T→AC	0,822	U→TA	0,320
		C→c	1,000	A→a	0,603	U→CA	0,139
				A→b	0,397	A→c	1,000
-		-		C→c	1,000	C→a	0,594
				-		C→b	0,406

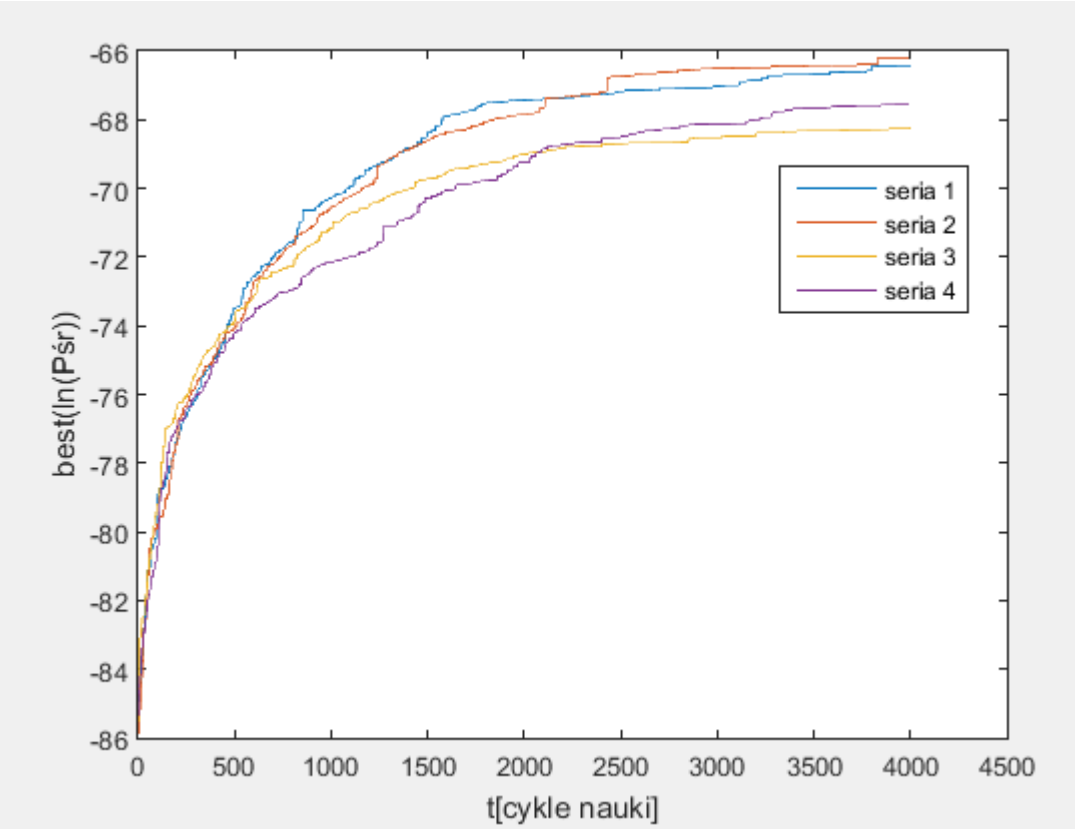
Jak widać w Tabeli 4.20, gramatyka końcowa uzyskana dla symulacji z wykorzystaniem G2.1 idealnie odzwierciedla gramatykę docelową 2. Wszystkie 4 uzyskane gramatyki końcowe dobrze opisują sprawdzany język.

4.4. Uczenie języka sekwencji aminokwasowych rodziny motywów białkowych

W ostatniej części badań, polegającej na przetestowaniu stworzonego algorytmu na języku, który opisuje rzeczywistą rodzinę białek, zdecydowałem się na ponowne użycie zestawu hiperparametrów wykorzystanych w doświadczeniu 4.2.

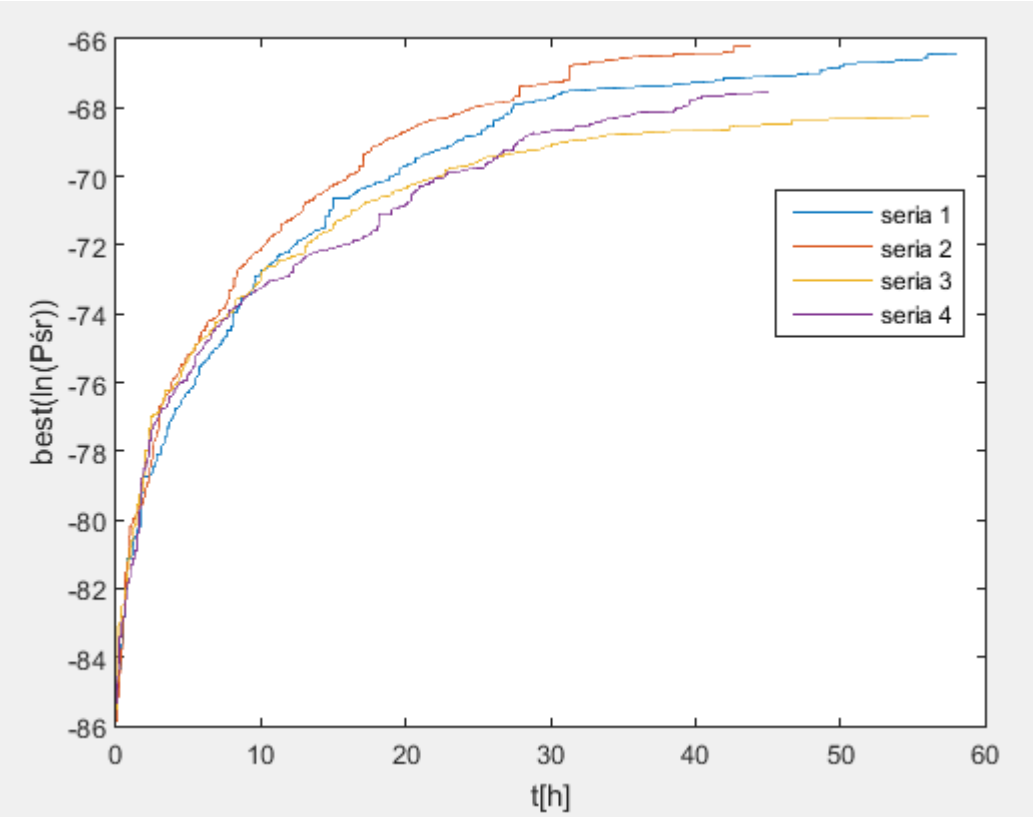
W doświadczeniu zastosowano 4-krotną walidację krzyżową. Dla każdego podzbioru wykonano jeden test (ze względu na czasochłonność obliczeń).

Na Rys. 4.1 widać jak zmieniała się wartość best(ln(Pśr)) w poszczególnych cyklach nauki dla wszystkich czterech serii.



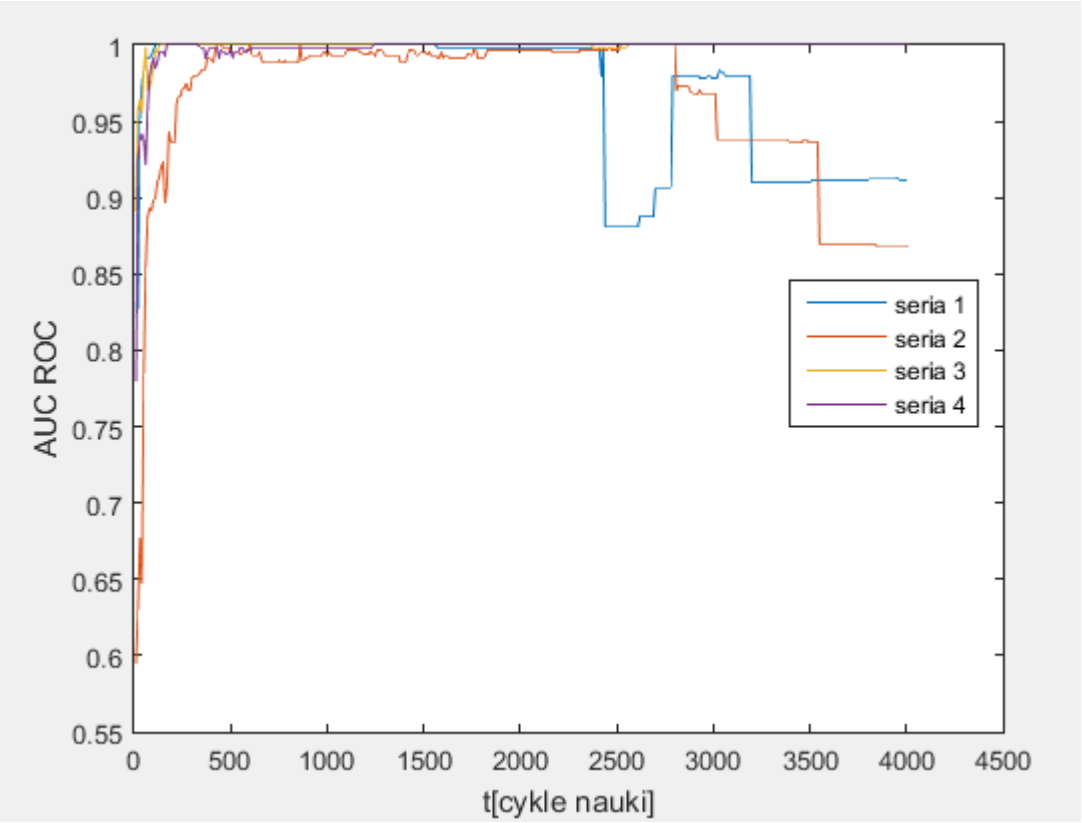
Rysunek 4.1 - Wyniki maszynowego uczenia dla czterech serii doświadczeń na sekwencjach białkowych (w cyklach nauki).

Na Rys. 4.2 przedstawiono zależność best(ln(Pśr)) od zegarowego czasu nauki (w godzinach).



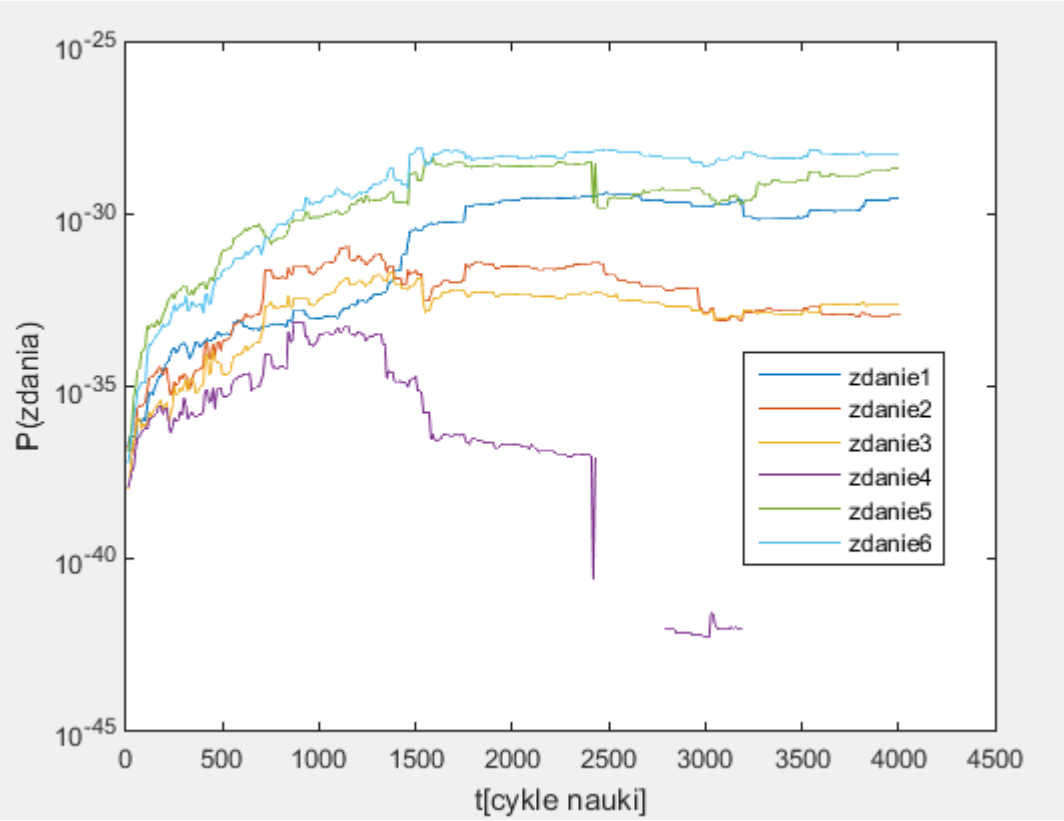
Rysunek 4.2 - Wyniki maszynowego uczenia dla czterech serii doświadczeń na sekwencjach białkowych (w godzinach).

Jak widać podczas trwania nauki wartość best(ln(Pśr)) cały czas rośnie, jeśli spojrzymy jednak na to, jak zmieniało się pole pod krzywą ROC podczas nauki możemy zauważyć, że w przypadku serii pierwszej oraz drugiej wykres w kilku momentach gwałtownie opada (Rys. 4.3).

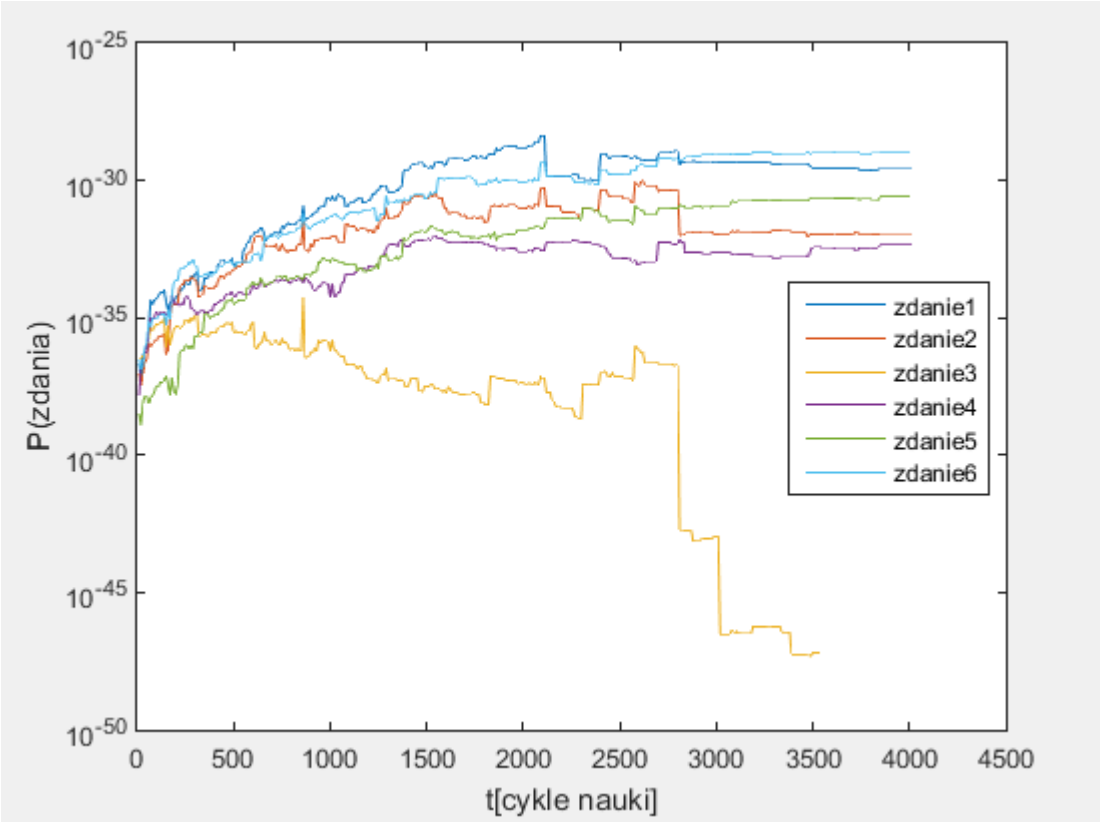


Rysunek 4.3 – Zmiana wartości pola pod krzywą ROC podczas maszynowego uczenia

Jest to wynik tak zwanego przeuczenia. Gramatyka zaczyna dopasowywać się tak dokładnie do próbki uczącej, że część pozytywnych próbek ze zbioru walidującego zaczyna być klasyfikowana jako nienależąca, lub należąca z bardzo małym prawdopodobieństwem, do testowanej gramatyki [20]. Analizując jak zmieniało się prawdopodobieństwo z jakim sekwencje pozytywne z próbki walidacyjnej były generowane przez testowaną gramatykę w seriach 1 i 2 (Rys. 4.4 oraz Rys.4.5) widać, że zarówno w przypadku serii 1 jak i serii 2 niektóre z sekwencji zaczynają być w pewnym momencie generowana z wyraźnie mniejszym, bądź zerowym prawdopodobieństwem. W przypadku serii 1 jest to najbardziej widczone dla zdania nr 4 („TVAVEFDTYENTVFTDPPYTHIGFDVN”), zaś w przypadku serii 2 jest to najbardziej widoczne dla zdania nr 3 („IFAVEFDVFANQEFNDINDNHVGVDVN”).

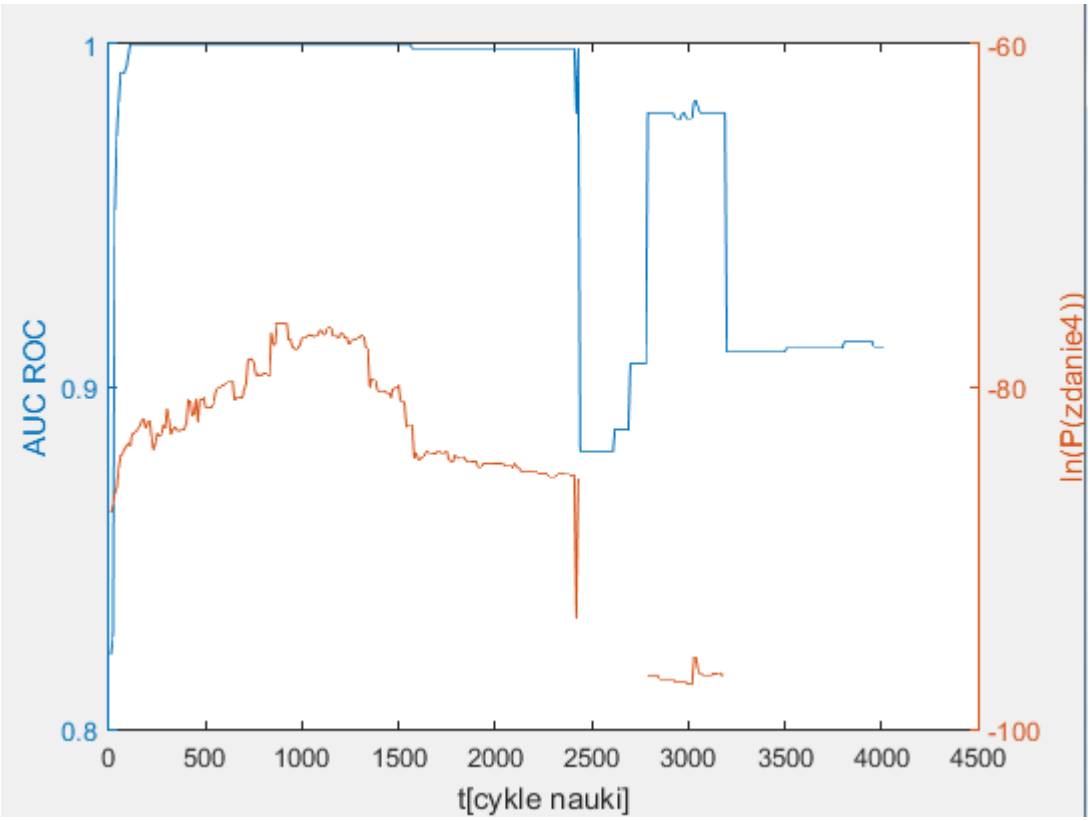


Rysunek 4.4 – Prawdopodobieństwa z jakimi pozytywne zdania z próbki walidacyjnej są generowane przez testowaną gramatykę, nieciągłości w przypadku zdania 4 oznaczają, że w pewnych momentach nauki prawdopodobieństwo z jakim zdanie 4 było generowane przez gramatykę wynosiło 0. Seria 1.

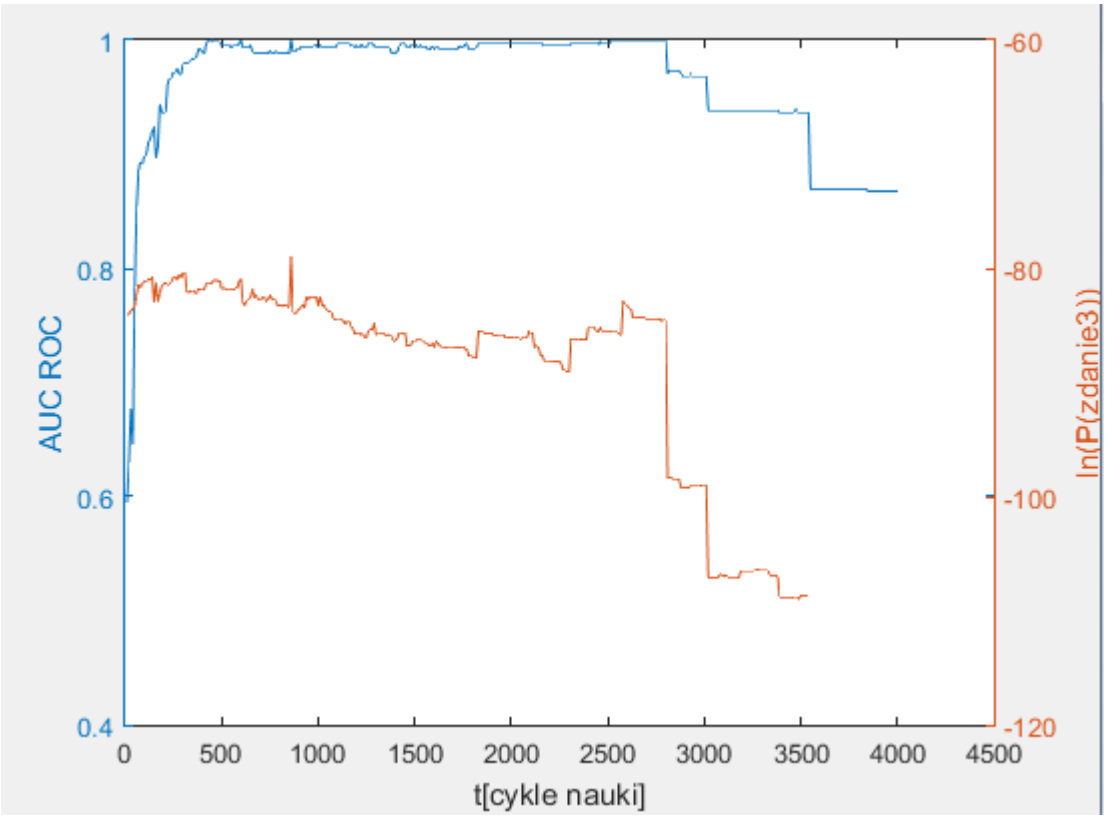


Rysunek 4.5 - Prawdopodobieństwa z jakimi pozytywne zdania z próbki walidacyjnej są generowane przez testowaną gramatykę. Seria 2.

Porównując zmiany P(zdania) ze zmianami AUC ROC podczas nauki (Rys. 4.6 oraz Rys 4.7) można zauważyć, że spadki prawdopodobieństwa z jakim zdania są generowane przez testowaną gramatyką pokrywają się ze spadkami wartości pola pod krzywą ROC.



Rysunek 4.6 – Zestawienie prawdopodobieństwa z jakim zdanie 4 z próbki walidacyjnej jest generowane przez sprawdzaną gramatykę z AUC ROC. Seria 1.



Rysunek 4.7 - Zestawienie prawdopodobieństwa z jakim zdanie 3 z próbki walidacyjnej jest generowane przez sprawdzaną gramatykę z AUC ROC. Seria 2.

Zmiany wartości best(ln(Pśr)) oraz pola pod krzywą ROC dla poszczególnych serii są przedstawione w Tabelach 4.22 oraz 4.23.

Tabela 4.21 – Zestawienie sprawdzanych parametrów dla serii 1 oraz 2.

	seria 1		seria 2	
t [h]	best(ln(Pśr))	AUC ROC	best(ln(Pśr))	AUC ROC
0	-85,03	0,82	-85,87	0,60
6	-75,43	1,00	-74,51	1,00
12	-72,14	1,00	-71,25	1,00
18	-70,19	1,00	-69,09	0,99
24	-68,92	1,00	-68,11	0,99
30	-67,73	1,00	-67,27	1,00
36	-67,39	1,00	-66,52	0,97
42	-67,14	0,88	-66,38	0,87
end	-66,42	0,91	-66,20	0,87

Tabela 4.22 – Zestawienie sprawdzanych parametrów dla serii 3 oraz 4.

	seria 3		seria 4	
t [h]	best(ln(Pśr))	AUC ROC	best(ln(Pśr))	AUC ROC
0	-85,52	0,89	-85,32	0,78
6	-74,72	1,00	-75,02	1,00
12	-72,37	1,00	-72,87	1,00
18	-70,70	1,00	-71,55	1,00
24	-69,71	1,00	-69,80	1,00
30	-69,14	1,00	-68,69	1,00
36	-68,75	1,00	-68,15	1,00
42	-68,65	1,00	-67,64	1,00
end	-68,26	1,00	-67,54	1,00

Jak widać na podstawie analizy Tabel 4.21 oraz 4.22 oraz Rys. 4.1 – 4.7 początkowe wartości best(ln(Pśr)) we wszystkich seriach są do siebie bardzo podobne i w początkowych etapach nauki zwiększają się w podobnym tempie. Obie serie, w których doszło do przeuczenia (seria 1 oraz seria 2), osiągają zauważalnie wyższą wartością max(best(ln(Pśr))).

Gramatyki końcowe dla wszystkich serii zostały przedstawione w Tabeli 4.23.

Tabela 4.23 - reguły o prawdopodobieństwie niemniejszym niż 5% w gramatykach końcowych uzyskanych w 3 eksperymencie obliczeniowym.

seria 1		seria 2		seria 3		seria 4	
reguła	P(r)	reguła	P(r)	reguła	P(r)	reguła	P(r)
0→15	0,681	0→06	0,725	0→06	0,333	0→04	0,148
0→40	0,216	0→30	0,128	0→41	0,118	0→11	0,186
0→60	0,103	1→23	0,179	0→42	0,111	0→50	0,665
1→06	0,096	1→45	0,713	0→51	0,107	1→34	0,603
1→16	0,493	1→65	0,197	0→62	0,282	1→51	0,312
1→41	0,151	2→34	0,090	1→24	0,125	2→24	0,254
1→43	0,113	2→45	0,253	1→43	0,741	2→25	0,069
1→61	0,146	2→53	0,180	1→53	0,053	2→56	0,135
2→24	0,432	2→54	0,387	2→16	0,749	2→62	0,289
2→25	0,092	2→65	0,090	2→50	0,130	2→64	0,252
2→42	0,305	3→31	0,276	2→52	0,106	3→16	0,499
2→55	0,118	3→41	0,092	3→36	0,134	3→26	0,151
3→25	0,102	3→62	0,632	3→50	0,694	3→35	0,181
3→35	0,314	4→D	0,351	3→56	0,111	3→52	0,134
3→36	0,208	4→L	0,125	4→D	0,100	4→N	0,383
3→43	0,342	4→F	0,331	4→E	0,278	4→I	0,213
4→A	0,113	4→S	0,090	4→L	0,077	4→P	0,051
4→D	0,160	4→Y	0,069	4→K	0,058	4→S	0,071
4→E	0,145	5→A	0,188	4→Y	0,072	4→V	0,157
4→L	0,099	5→R	0,062	4→V	0,277	5→A	0,099
4→F	0,146	5→Q	0,065	5→A	0,153	5→E	0,164
4→T	0,086	5→E	0,333	5→L	0,082	5→I	0,062
4→V	0,098	5→P	0,051	5→F	0,184	5→L	0,114
5→N	0,428	5→T	0,206	5→P	0,135	5→K	0,052
5→D	0,219	6→N	0,213	5→S	0,120	5→F	0,173
5→K	0,077	6→D	0,137	5→T	0,188	5→S	0,050
5→P	0,102	6→G	0,091	6→N	0,221	5→V	0,212
5→S	0,064	6→H	0,087	6→D	0,269	6→D	0,495
6→D	0,104	6→I	0,182	6→G	0,132	6→G	0,184
6→G	0,155	6→V	0,206	6→H	0,089	6→H	0,112
6→H	0,098	-		6→I	0,202	6→T	0,101
6→I	0,263			6→V	0,060	-	
6→L	0,067			-			
6→V	0,279						

W uzyskanych gramatykach nie widać wyraźnie powtarzających się wzorców wśród reguł strukturalnych, może to być spowodowane niedostatecznie długim czasem nauki. Analizując jak

zostały rozłożone prawdopodobieństwa wygenerowania poszczególnych aminokwasów przy pomocy dostępnych symboli terminalnych można zauważyć, że niektóre aminokwasy są generowane przez nieterminalne symbole leksykalne we wszystkich gramatykach ze znacznie większym prawdopodobieństwem niż inne. Aminokwas oznaczony symbolem terminalnym *D* (kwas asparaginowy) w seriach 1, 2 i 3 jest generowany z prawdopodobieństwem niemniejszym niż 10% przez aż dwa symbole terminalne, zaś w serii 4 jest generowany z prawdopodobieństwem bliskim 50% przez jeden symbol nieterminalny.

PODSUMOWANIE

W pracy udało się zaimplementować algorytm genetyczny oparty na założeniach algorytmu SGA, w którym zastosowano technikę przeprowadzania mutacji umożliwiającą zerowanie reguł. Testy stworzonego narzędzia przeprowadzone na językach testowych udowodniły, że jest ono w stanie efektywnie sprowadzić gramatykę pokrywającą zawierającą kilkaset reguł do postaci o kilkunastu regułach. W wyniku testów ustalono, że proces nauki przebiega szybciej wraz ze zwiększeniem prawdopodobieństwa mutacji, zastosowanie niskiej maksymalnej skali mutacji jest niekorzystne, zastosowany algorytm dobrze radzi sobie z symulacjami nawet z wykorzystaniem niewielkiej liczby osobników, oraz, że prawdopodobieństwo krzyżowania nie ma wyraźnego wpływu na tempo nauki. Przeprowadzone testy pokazały też, że uzyskanie gramatyki mogącej pełnić funkcję bardzo dobrego klasyfikatora (AUC ROC niemniejsze niż 0,95) jest zadaniem mniej czasochłonnym oraz niż optymalne rozdzielenie prawdopodobieństw pomiędzy regułami gramatyki. Większość serii z pierwszego eksperymentu obliczeniowego, w których populacja utknęła w lokalnym minimum wciąż mogła pełnić funkcję bardzo dobrego klasyfikatora, co wskazuje, że nawet daleki od optymalnego rozkład prawdopodobieństw reguł gramatyki może być wystarczający dla funkcji klasyfikacyjnych (zwiększając średnie prawdopodobieństwo z jakimi generowane są zdania należące do sprawdzanego języka jednocześnie zmniejszamy średnie prawdopodobieństwo z jakim generowane są sekwencje do tego języka nienależące). Doświadczenie wykonane dla gramatyki opisującej język obejmujący sekwencje białkowe pokazało, że z pomocą stworzonego narzędzia możliwe jest uzyskanie gramatyki, która może pełnić funkcję dobrego klasyfikatora dla sprawdzanej rodziny sekwencji białek.

Poruszona w pracy kwestia maszynowego uczenia gramatycznych deskryptorów sekwencji białkowej jest bardzo szerokim tematem, który może być w przyszłości rozwijany na wiele sposobów. Dla każdego, sprawdzanego w pracy, zestawu hiperparametrów przeprowadzone zostały 3 testy, w celu zwiększenia wiarygodności uzyskanych wyników można odtworzyć przeprowadzone eksperymenty, przeprowadzając więcej testów dla każdego zestawu i zwiększając liczbę sprawdzanych zestawów. Zastosowany w pracy sposób oceny jakości klasyfikatora (AUC ROC) może okazać się nieefektywny w przypadku, w którym w próbce walidacyjnej znajduje się wielokrotnie więcej sekwencji negatywnych niż pozytywnych. Ze względu na to, stworzony model może zostać rozwinięty o dodatkową formę oceny jakości klasyfikacji z wykorzystaniem krzywej PRC (ang. Precision-Recall Curves). Zaobserwowane w dwóch seriach eksperymentu 4.4 przeuczenie gramatyki do próbki uczącej pokazuje, że zbyt długie szukanie optymalnego rozdziału prawdopodobieństw może nie być korzystne. Porównanie jakości gramatyk uzyskanych tuż przed zajściem procesu przeuczenia z gramatyką końcową może być traktowane jako jedna z dróg rozwinięcia niniejszej pracy. Przy porównywaniu gramatyk uzyskanych w wyniku nauki dla języków testowych można zauważyć, że uzyskane języki są do siebie podobne. Analiza podobieństwa tych gramatyk jest jednak utrudniona ze względu na to, że poszczególne symbole nieterminalne mogą pełnić analogiczne funkcje w różnych seriach nauki. Stworzenie narzędzia umożliwiającego ułatwienie takiej analizy poprzez automatyczne mapowanie symboli ze względu na pełnione przez nie funkcje jest kolejnym możliwym rozwinięciem tej pracy.

LITERATURA

[1] D.B. Searl: A primer in macromolecular linguistics, Biopolymers. vol. 99 , s. 203–217, 2013.

[2] P.G. Higgs, T.K. Attwood: Bioinformatyka i ewolucja molekularna. PWN, Warszawa 2008.

[3] W. Dyrka, J. Nebel: A stochastic context free grammar based framework for analysis of protein sequences, BMC Bioinformatics, Volume 10, Number 1, 2009.

[4] W.Dyrka, F. Coste, J.e Talibart, Estimating probabilistic context-free grammars for proteins using contact map constraints, preprint arXiv:1805.08630, 2018.

[5] Ghaheri A, Shoar S, Naderan M, Hoseini SS. The Applications of Genetic Algorithms in Medicine. Oman Med J.; vol. 30(6), 2015, s.406-16, 2015.

[6] J. Arabas: Wykłady z algorytmów ewolucyjnych. WNT, Warszawa 2004.

[7] mathspace.pl/matematyka/receiver-operating-characteristic-krzywa-roc-czyli-ocena-jakosci-klasyfikacji-czesc-7/ [dostęp: 03.01.2019].

[8] Holland J. H.: Adaptation in natural and artificial system. Ann Arbor, University of Michigan Press 1975.

[9] A.Osyczka, S.Kundu: A New Method to Solve Generalized Multicriteria Optimization Problems Using the Simple Genetic Algorithm, Structural Optimization.vol. 10, 1995, s. 95.

[10] Regresja liniowa jednej zmiennej, https://knbit.edu.pl/pl/wiki/ai/kurs-machine-learning/02-regresja-liniowa-jednej-zmiennej/ [dostęp: 19.08.18].

[11] J. E. Hopcroft, R. Motwani, J.D. Ullman: Wprowadzenie do teorii automatów, języków i obliczeń. PWN Warszawa 2005.

[12] Cocke, John; Schwartz, Jacob T.; Programming languages and their compilers: Preliminary notes, 1970.

[13] N. Chomsky. On certain formal properties of grammars. Information and Control, 2(2):137 – 167, 1959.

[14] https://gawron.sdsu.edu/compling/course_core/assignments/prob_parsing_assignment.htm [dostęp: 11.12.18].

[15] Sakakibara, Yasubumi and Mitsuhiro Kondo. "GA-based Learning of Context-Free Grammars using Tabular Representations." ICML, 1999.

[16] George D. Smith, Nigel C. Steele, Rudolf F. Albrecht, Artificial Neural Nets and Genetic Algorithms: Proceedings of the International Conference in Norwich, U.K., 1997.

[17] Unold, O., Culer, Ł., Kaczmarek, A. (2018). Iterative method of generating artificial context-free grammars. The 14th International Conference on Grammatical Inference ICGI, Wrocław 2018.

[18] N. Sharon and H. Lis. Legume lectins–a large family of homologous proteins. The FASEB Journal, 4(14):3198–3208, 1990. PMID: 2227211.

[19] https://web.expasy.org/protscale/pscale/A.A.Swiss-Prot.html [dostęp: 11.12.18].

[20] Tetko, I.V.; Livingstone, D.J.; Luik, A.I. Neural network studies. 1. Comparison of Overfitting and Overtraining, J. Chem. Inf. Comput. Sci., 1995, 35, 826-833.