Maszynowe uczenie gramatycznych deskryptorów sekwencji białkowych

# **Machine learning of grammatical descriptors of protein sequences**

Robert Kowalski\*

Politechnika Wrocławska, Wydział Podstawowych Problemów Techniki, 50-370 Wrocław, Wybrzeże Wyspiańskiego 27

\* e-mail: 229715@student.pwr.edu.pl

Streszczenie

Praca polega na implementacji ulepszeń do istniejącej metody probabilistycznego modelu gramatycznego (opartego na gramatyce bezkontekstowej) sekwencji białkowych, obejmujących sam model i proces treningu (uczenie maszynowe). W pracy opisano wykorzystywaną w algorytmach pierwszorzędową strukturę białek, obecnie wykorzystywane w bioinformatyce algorytmy ewolucyjne (z uwzględnieniem ich mocnych i słabych stron), procesy uczenia maszynowego, podstawowe pojęcia związane z teorią automatów (takie jak alfabet, łańcuch, język oraz problem w kontekście teorii automatów) oraz języki i gramatyki ze szczególnym uwzględnieniem probabilistycznych gramatyk bezkontekstowych.

**Słowa kluczowe**: aminokwasy, algorytmy ewolucyjne, teoria automatów, język, gramatyka, probabilistyczna gramatyka bezkontekstowa, analizator składniowy

Abstract

The work consists in the implementation of improvements to the current methods of the probabilistic grammatical model (based on context-free grammar) of protein sequences, including the model itself and the training process (machine learning). The work describes the primary protein structure used in algorithms, currently used in bioinformatics, evolutionary algorithms (including their strengths and weaknesses), machine learning processes, basic concepts of automata theory (such as alphabet, chain, language and problem in the automation theory), and languages and grammars, with particular emphasis on probabilistic context-free grammars.

**Keywords:**amino acids, evolutionary algorithm, automata theory, language, grammar, probabilistic context-free grammar, pasring

1. Wprowadzenie

Struktura pierwszorzędowa białek, zwana również strukturą pierwotną białek, to najmniej skomplikowany poziom organizacji strukturalnej białka, opisujący w formie liniowej układ aminokwasów tworzących rozpatrywany łańcuch polipeptydowy.

Można zauważyć, że te kluczowe dla wszelkiego życia struktury wykazują podstawową charakterystykę języków: nieskończona liczba sekwencji może zostać wyrażona za pomocą skończonej liczby monomerów. W przypadku białek zaledwie 20 niepowtarzalnych aminokwasów (które mogą być traktowane jako litery) tworzy miliony sekwencji (które mogą być traktowane jako pojedyncze słowa lub całe zdania). Fizykochemiczne właściwości białek są zależne od ich struktury która jest warunkowana przez sekwencję aminokwasów tworzących dane białko, przy czym, podobnie jak w przypadku słów w języku naturalnym którego używamy na co dzień do komunikacji interpersonalnej, sekwencje aminokwasów mogą być niejednoznaczne (jedna sekwencja aminokwasów może odpowiadać za więcej niż jedną strukturę białka w zależności od kontekstu (środowiska w którym się znajduje)) [[[1]](#endnote-1)].

Ze względu na to, algorytmy wykorzystujące założenia koncepcji języka białek mogą być wykorzystywane jako narzędzia bioinformatyczne umożliwiające, z dużym prawdopodobieństwem, określenie struktury badanego białka (a co za tym idzie uzyskanie pełniejszego obrazu jego właściwości fizykochemicznych) na podstawie opisującej go struktury pierwszorzędowej (sekwencji aminokwasów tworzących dane białko).

1. Algorytmy ewolucyjne
   1. Wprowadzenie do algorytmów ewolucyjnych

Algorytmy ewolucyjne służą do przeszukiwania przestrzeni alternatywnych rozwiązań w celu wyszukiwania rozwiązania optymalnego, lub rozwiązania które będzie do niego w wystarczającym stopniu zbliżone.

Terminologia wykorzystywana przy opisywaniu algorytmów ewolucyjnych jest inspirowana terminologią wykorzystywaną do opisywania procesów ewolucyjnych. Tak więc: zbiór rozwiązań rozpatrywanego problemu nazywamy populacją, która to natomiast składa się z pojedynczych osobników (będących pojedynczymi rozwiązaniami problemu). Rozwiązywany problem określany jest 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 cech określany jako fenotyp, który z kolei jest kodowany przez zestaw parametrów nazywany genotypem. Dla rozpatrywanego środowiska można określić funkcję przystosowania, która dla każdego osobnika 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 [[[2]](#endnote-2)].

* 1. Działanie algorytmów ewolucyjnych

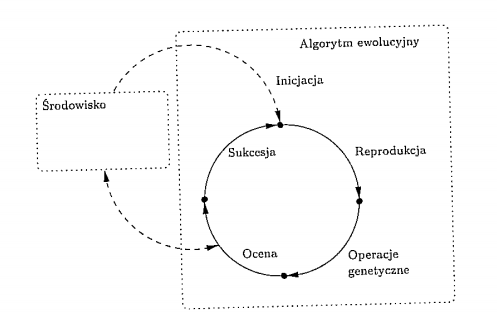
Działanie algorytmu ewolucyjnego (zilustrowane na Rysunek 1. polega na kolejnym przeprowadzaniu procesów reprodukcji, operacji genetycznych, oceny oraz sukcesji.

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 kilka razy, 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 (przyjmuje się, że niewielkie mutacje są bardziej prawdopodobne niż duże). Krzyżowanie to operacja polegająca na wygenerowaniu 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).



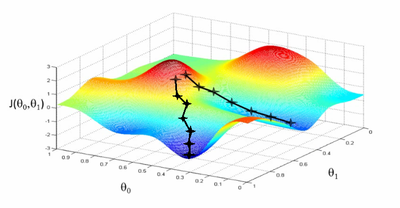
Rysunek 1-schemat działania algorytmu ewolucyjnego [[[3]](#endnote-3)]

* 1. **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 ilości obiegów pętli algorytmu, natomiast 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, można powiedzieć, że jest wysokie prawdopodobieństwo tego, że wraz  
ze wzrostem liczby obiegów pętli algorytmu będzie rosnąć poziom przystosowania kolejnych populacji bazowych.

Kierunek rozwoju kolejnych populacji bazowych jest jednak bardzo mocno uzależniony  
od przebiegu funkcji przystosowania, od 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 Rysunek 2). Opuszczenie takich pułapek ewolucyjnych jest tym bardziej prawdopodobne możliwe  
im większe mutacje podczas procesu operacji genetycznych oraz im większe prawdopodobieństwo reprodukcji osobników słabiej przystosowanych, zwiększanie tych parametrów skutkuje jednak wolniejszym zwiększaniem wartości przystosowania populacji, dlatego właśnie, algorytmy ewolucyjne są wrażliwe na lokalne maksima funkcji przystosowania)[2,2].



Rysunek 2 - przedstawienie prawdopodobnego procesu zwiększania przystosowania populacji bazowej wraz z kolejnymi iteracjami algorytmu ewolucyjnego[[[4]](#endnote-4)]

1. Teoria automatów i gramatyki
   1. Wprowadzenie do teorii automatów

Teoria automatów to badanie abstrakcyjnych urządzeń obliczeniowych, czyli „maszyn obliczeniowych”. Dzięki teorii automatów możliwe jest nakreślenie granicy pomiędzy tym, co może zrobić maszyna obliczeniowa (np. komputer), a tym, czego zrobić taka maszyna nie może. Problemy, które nie mogą zostać rozwiązane efektywnie za pomocą komputera to tak zwane problemy „niepodatne” bądź „NP-trudne”. W przypadku zakwalifikowania danego problemu do kategorii problemów niepodatnych uzyskujemy informację, że za pomocą metod obliczeniowych wykonywanych przez maszynę obliczeniową nie jesteśmy w stanie uzyskać jego rozwiązania,  
w związku z czym musimy skupić się na jego obejściu, bądź zadowolić się algorytmem umożliwiającym uzyskanie przybliżonego wyniku rozwiązania problemu.

* 1. Podstawowe pojęcia w teorii automatów

Podstawowe pojęcia w teorii automatów używane w celu opisu konkretnych problemów oraz w celu zakwalifikowania konkretnego problemu do kategorii problemów podatnych bądź niepodatnych  
to: alfabety, łańcuchy, języki oraz problemy.

Alfabetem nazywamy konkretny, skończony, niepsuty zbiór symboli, przykładami alfabetu mogą być:

* zbiór wszystkich bądź niektórych liter alfabetu łacińskiego
* zbiór symboli ASCII
* zbiór złożony z dokładnie jednego symbolu
* zbiór liczb należących do pewnego przedziału
* alfabet binarny (najczęściej stosowany w informatyce) składający się z dwóch symboli:  
  0 oraz 1
* zbiór 20 symboli przypisanych do konkretnych aminokwasów (stosowany  
  w bioinformatyce)

Alfabety zwyczajowo oznaczamy symbolem ∑.

Łańcuchem nazywamy skończony zbiór symboli należących do konkretnego alfabetu, w przypadku w którym znaki danego łańcucha należą do konkretnego alfabetu mówimy, że jest to łańcuch nad tym alfabetem (np. 010010 to łańcuch nad alfabetem binarnym). W przypadku łańcuchów możemy wyróżnić łańcuch pusty, czyli łańcuch o zerowej liczbie wystąpień symboli, łańcuch ten oznaczamy symbolem *ε* i jest on łańcuchem nad dowolnym alfabetem.

Językiem nazywamy zbiór łańcuchów, których wszystkie elementy zostały wybrane z określonego zbioru wszystkich łańcuchów nad danym alfabetem, jeżeli dany jeżyk jest opisywany przez konkretny alfabet, to mówimy, że jest to język nad tym alfabetem. Język nie musi wykorzystywać wszystkich elementów zawartych w danym alfabecie, w związku z czym możemy stwierdzić, że jeżeli dany język jest językiem nad alfabetem ∑, to jest on również językiem nad dowolnym alfabetem który jest nadzbiorem alfabetu ∑.

Problemem nazywamy kwestię rozstrzygnięcia, czy konkretny łańcuch należy do konkretnego języka [[[5]](#endnote-5)].

* 1. Gramatyki formalne

Gramatyki są sposobem opisu języków formalnych (czyli podzbiorów wszystkich słów nad danym alfabetem).

Najpopularniejszym sposobem aby opisać język formalny jest użycie gramatyki formalnej, 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)
* Symbolu startowego (który należy do symbolinieterminalnych)
* Skończonego zbioru reguł przepisywania (produkcji), każda produkcja to para słów (gdzie pierwsze słowo w ramach konkretnej gramatyki może przechodzić w drugie), w których mogą występować zarówno symbole nieterminalne jak i terminalne

Aby sprawdzić, czy konkretne słowo należy do języka opisanego przez taką 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) [2].

* 1. Probabilistyczne gramatyki bezkontekstowe

Rozstrzyganie, czy konkretne słowo należy do języka opisywanego przez daną gramatykę formalną należy do klasy problemów niepodatnych, w związku z czym w praktyce wykorzystuje się konkretne klasy gramatyk, przykładem takiej klasy gramatyk są gramatyki bezkontekstowe, w których wszystkie produkcje przyjmują postać (A, *Γ*), gdzie A to dowolny symbol nieterminalny, zaś *Γ* jest dowolnym zbiorem symboli nieterminalnych i terminalnych.

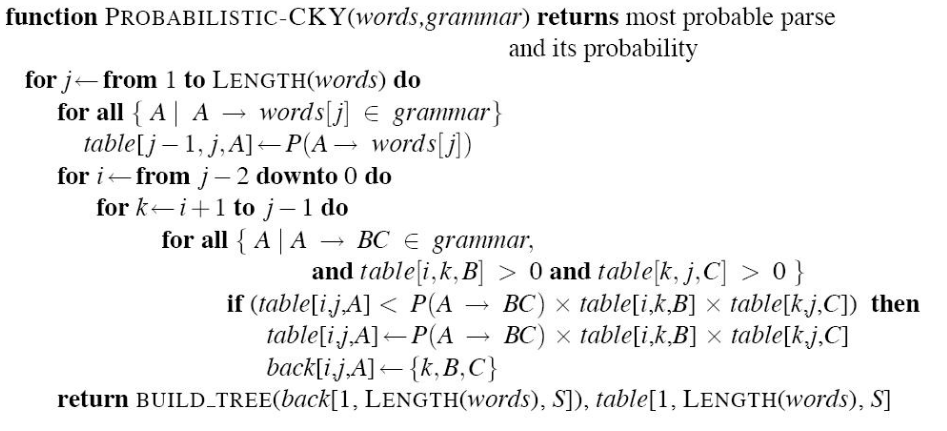
Probabilistyczne gramatyki bezkontekstowe to gramatyki bezkontekstowe, w których każdej produkcji przypisano odpowiadającą jej wartość prawdopodobieństwa wystąpienia danej produkcji. Prawdopodobieństwa dla konkretnych produkcji są przypisywane w ten sposób, aby suma prawdopodobieństw wszystkich produkcji o wspólnym poprzedniku wynosiła 1.

1. Analizatory składniowe

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

Aby sprawdzić, 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). Schemat działania algorytmu CKY został przedstawiony na Rysunek 3 - pseudokod probabilistycznego algorytmu CKY [] Sprawdzana gramatyka musi zostać sprowadzona do postaci normalnej Chomsky’ego (forma, w której wszystkie produkcje danej gramatyki zostały sprowadzone do postaci:

(A,a) lub (A,BC), gdzie duże litery symbolizują symbole terminalne, natomiast małe litery symbolizują symbole nieterminalne).[[[6]](#endnote-6),[[7]](#endnote-7)]



Rysunek 3 - pseudokod probabilistycznego algorytmu CKY [[[8]](#endnote-8)]

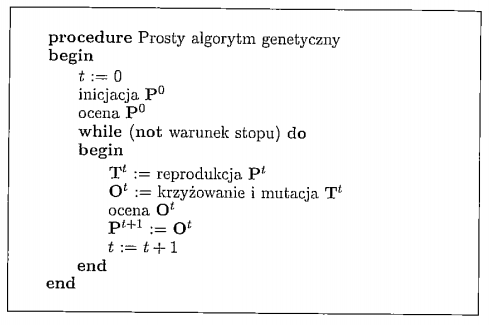
1. Prosty algorytm genetyczny
   1. Właściwości prostego algorytmu genetycznego

Prosty algorytm genetyczny (SGA) to zaproponowany w 1975 roku przez Johna Hollanda algorytm mający początkowo za zadanie modelowanie procesu ewolucji. Jego działanie zostało zilustrowane na Rysunek 4.

Algorytm przetwarza dwie populacje: Pt, którą nazywamy populacją bazową, oraz Ot, czyli populację potomną. W ramach działania algorytmu tworzona jest też populacja Tt, zwana populacją tymczasową, której zadaniem jest przechowywanie kopii osobników z populacji bazowej. Rozmiary tych trzech populacji są jednakowe.

W kroku inicjacji populacji bazowej, dochodzi do losowej generacji osobników tworzących pierwszą populację bazową, następnie obliczana jest wartość przystosowania osobników.

Osobniki z populacji bazowej są powielane z prawdopodobieństwem proporcjonalnym do ich wartości przystosowania, następnie osobniki z populacji tymczasowej są kojarzone w pary, dla każdej pary podejmuje się decyzję o krzyżowaniu (parametr określający prawdopodobieństwo decyzji pozytywnej jest atrybutem wejściowym funkcji korzystającej z algorytmu SGA), w przypadku decyzji pozytywnej osobniki z pary mieszają swoje geny, powstałe w ten sposób osobniki potomne zastępują swoich rodziców, następnie wszystkie osobniki z populacji tymczasowej są poddawane mutacjom,  
w ten sposób powstaje populacja potomna, która w następnym obiegu pętli staje się populacją bazową. Algorytm ten wykonuje się w pętli, której każdy obieg jest nazywany generacją, pętla ta może zostać przerwana po wykonaniu określonej liczby generacji, bądź wygenerowanie osobnika o określonej wartości przystosowania [2].



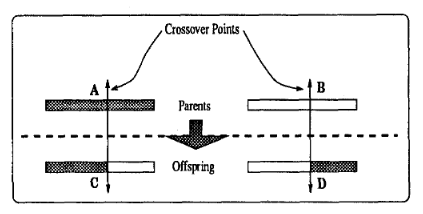
Rysunek 4-pseudokod algorytmu SGA [2]

* 1. Kodowanie osobników i operacje genetyczne w algorytmie SGA

Algorytm genetyczny najbardziej znany jest z kodowania binarnego, oznacza to, że 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 atrybutem wejściowym funkcji wykorzystującej algorytm SGA), to wartość genu zostaje zanegowana (zmieniana na przeciwną).

Podczas procesu krzyżowania (zilustrowanego na Rysunek 4.) losowane jest konkretne miejsce na chromosomie (z rozkładem równomiernym), 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]](#endnote-9)].



Rysunek 5-Schemat procesu krzyżowania[[[10]](#endnote-10)]

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

|  |  |  |
| --- | --- | --- |
|  |  |  |

Gdzie *X* oznacza osobnika, *Φ(X)*oznacza wartość przystosowania. Metoda ta jest nazwana reprodukcją proporcjonalną, bądź reprodukcją ruletkową.

Najczęściej stosowane w praktyce warunki przerwania algorytmu zostały opisane już wcześniej, należy jednak mieć na uwadze to, że algorytm genetyczny jest procesem losowym, w związku z czym nigdy nie jesteśmy w stanie uzyskać gwarancji, że otrzymany wynik jest wynikiem optymalnym, wiemy jedynie, że wraz ze wzrostem liczby generacji rośnie prawdopodobieństwo zbliżenia się do wyniku optymalnego[2].

Literatura

1. [] D.B. Searl: *A primer in macromolecular linguistics*, Biopolymers. vol. 99,2013, s. 203–217. [↑](#endnote-ref-1)
2. [] P.G. Higgs, T.K. Attwood: *Bioinformatyka i ewolucja molekularna*. PWN, Warszawa2008. [↑](#endnote-ref-2)
3. [] J. Arabas: *Wykłady z algorytmów ewolucyjnych*. WNT, Warszawa 2004. [↑](#endnote-ref-3)
4. [] Regresja liniowa jednej zmiennej, https://knbit.edu.pl/pl/wiki/ai/kurs-machine-learning/02-regresja-liniowa-jednej-zmiennej/ [↑](#endnote-ref-4)
5. [] E. Rivas, S.R. Eddy: *The language of RNA: a formal grammar that includes pseudoknots*, Bioinformatics.vol.16, 2000, s. 334–340. [↑](#endnote-ref-5)
6. [] M. Weir, S. Aggarwal, B. de Medeiros B. Glodek: *Password Cracking Using Probabilistic Context Free Grammars* Security and Privacy, 2009 30th IEEE Symposium on, 2009, s. 391–405. [↑](#endnote-ref-6)
7. [] E. Charniak,Statistical: *Parsing with a context-free grammar and word statistic*, Proceedings of the 14th NationalConference on Artificial Intelligence,1997s. 598–603. [↑](#endnote-ref-7)
8. [] Context Grammars, https://gawron.sdsu.edu/compling/course\_core/assignments/prob\_parsing\_assignment.htm [↑](#endnote-ref-8)
9. [] P.Korhonen, S.C.Narula: *An Evolutionary Approach to Support Decision Making with Linear Decision Models*, Multi-Criteria Decision Analysis.vol. 2, 1993, s. 111–119. [↑](#endnote-ref-9)
10. [] A.Osyczka, S.Kundu: *A New Method to Solve Generalized Multicriteria Optimization Problems Using the Simple Genetic Algorithm,* Structural Optimization.vol. 10, 1995, s. 94–99. [↑](#endnote-ref-10)