Uniwersytet im. A. Mickiewicza - Wydział Matematyki i Informatyki

Zbigniew Tenerowicz

nr albumu 302242

Zastosowanie obliczeń ewolucyjnych w przetwarzaniu języka naturalnego

Praca magisterska na kierunku:

INFORMATYKA

Promotor:

dr hab. Krzysztof Jassem

Poznań 2010

Oświadczenie

Ja, niżej podpisany Zbigniew Tenerowicz student Wydziału Matematyki i Informatyki Uniwersytetu im. Adama Mickiewicza w Poznaniu oświadczam, że przedkładaną pracę dyplomową pt.:

„Zastosowanie obliczeń ewolucyjnych w przetwarzaniu języka naturalnego”

napisałem samodzielnie. Oznacza to, że przy pisaniu pracy, poza niezbędnymi konsultacjami, nie korzystałem z pomocy innych osób, a w szczególności nie zlecałem opracowania rozprawy lub jej części innym osobom, ani nie odpisywałem tej rozprawy lub jej części od innych osób.

Oświadczam również, że egzemplarz pracy dyplomowej w formie wydruku komputerowego jest zgodny z egzemplarzem pracy dyplomowej w formie elektronicznej.

Jednocześnie przyjmuję do wiadomości, że gdyby powyższe oświadczenie okazało się nieprawdziwe, decyzja o wydaniu mi dyplomu zostanie cofnięta.

....

data podpis

Streszczenie

Niniejsza praca opisuje zastosowanie algorytmu genetycznego w ewoluowaniu wag reguł w gramatyce używanej w istniejącym parserze w systemie tłumaczenia automatycznego celem poprawienia jakości wykonywanych tłumaczeń.

Zaprezentowane zostały podstawy teoretyczne i opis realizacji eksperymentu oraz rezultaty. Podsumowanie pracy przedstawia wnioski płynące z eksperymentu i nakreśla kierunki dalszych działań.

Słowa kluczowe

algorytm genetyczny, parsing, automatyczne tłumaczenie regułowe, waga reguły

Spis treści

1 Wstęp

„Gdzieś we wnętrzu HEX-a zakręciły się koła zębate. W mrówczych farmach otworzyły się maleńkie klapki i miliony mrówek ruszyły szklanymi rurkami."

Terry Pratchett[1]

Rozwój myśli technicznej naszej cywilizacji w każdej dziedzinie prędzej czy później natrafia na problem, który *Matka Natura* rozwiązała już dużo wcześniej. To jeszcze można by *Jej* wybaczyć, gdyby nie fakt, że zawsze musi zrobić to lepiej niż nam się udaje, czasami nawet nie potrafimy powtórzyć jej osiągnięć. Dobitnym przykładem tej sytuacji jest historia pionierów awioniki. Od mitycznego Ikara, przez Leonarda da Vinci po osławionych braci Wright – niezliczone rzesze wynalazców próbowały zmierzyć się z problemem, którego rozwiązanie *Ona* znała od milionów lat. Z pewnością niejeden człowiek przed Karolem Darwinem zastanawiał się „Jak *Ona* to robi?"

Informatyka bardzo długo wydawała się być dziedziną nauki, która *Matce Naturze* się nie kłania. Metody analityczne, które nie miały sobie równych (czasami dlatego, że jeszcze nie wymyślono innych) prędzej czy później zawsze umożliwiały rozwiązanie kolejnych problemów, a wszystkie ich ograniczenia można było uważać za tymczasowe w obliczu wykładniczego w czasie wzrostu mocy obliczeniowej.

Z czasem jednak okazało się, że informatyka może się od *Matki Natury* wiele nauczyć. I choć nie powstały komputery zbudowane z mrowisk jak Pratchettowski *Hex*, to opracowano[2] algorytmy oparte na zachowaniach mrówek, sieci neuronowe wzorowane na mózgu, i wreszcie, algorytmy genetyczne realizujące zasady ewolucji. Tym razem nie tylko wzorujemy się na *Jej* dziełach, ale implementujemy jedyną metodę, której używa *Natura*.

Niniejsza praca opisuje zastosowanie algorytmu genetycznego w ewoluowaniu wag reguł w gramatyce używanej w istniejącym parserze w systemie tłumaczenia automatycznego celem poprawienia jakości wykonywanych tłumaczeń.

Eksperyment realizowany w ramach niniejszej pracy ma na celu sprawdzenie czy możliwe jest opracowanie na drodze ewolucji takiego zestawu wag dla gramatyki używanej w parsowaniu zdań źródłowych, który zwiększy częstość wybierania prawidłowych konstrukcji gramatycznych i w efekcie poprawi jakość automatycznego tłumaczenia regułowego.

Zagadnienie parsowania i związane z nim definicje wraz z przykładami omówione są w rozdziale 2. Zebrana tam wiedza obejmuje zagadnienia z dziedziny lingwistyki komputerowej, do których odnosi się dalsza część pracy.

1 HEX to komputer pojawiający się w książce „Hogfather". Autor oryginału: Terry Pratchett, tłumaczenie: Piotr W. Cholewa

2 Może trafniejsze byłoby stwierdzenie „odkryto"?

Rozdział 2 zawiera również omówienie problemu niejednoznaczności oraz roli wag reguł w gramatyce.

Rozdział 3 jest wstępem do zagadnienia algorytmów genetycznych, gdzie przedstawiłem definicje wszystkich bytów składających się na algorytm genetyczny. Dalej rozdział ten zawiera przegląd wybranych zastosowań algorytmów genetycznych, zarówno ogólnych, jak i w dziedzinie lingwistyki komputerowej. Podrozdział 3.2.2 wprowadza w tematykę związaną z analizą działania i optymalizowaniem algorytmów genetycznych.

W dalszej części pracy znajduje się opis wykorzystanych narzędzi i sposobu przeprowadzenia eksperymentu oraz użytego algorytmu genetycznego. Szczególny nacisk położyłem na przedstawienie procesu realizującego ocenę przystosowania osobników.

Ostatnie dwa rozdziały stanowią prezentację wyników eksperymentu i wniosków, do których doprowadził. Wnioski dotyczą zmian w rezultatach tłumaczenia automatycznego, ale również poruszają kwestię skuteczności systemu automatycznej ewaluacji jakości tłumaczenia. Pracę kończy wskazanie perspektyw dalszego rozwoju opisanego zastosowania algorytmu genetycznego.

2 Parsowanie w przetwarzaniu języka naturalnego

Streszczenie

W tym rozdziale przedstawię wybrane pojęcia potrzebne do opisania parsowania oraz czynności poprzedzających parsowanie. Następnie omówię dwa podejścia do zagadnienia parsowania i zaprezentuję najbardziej rozpowszechnione przykłady parserów języka naturalnego. W dalszej części postaram się pokazać, jaką rolę pełnią wagi reguł w parsowaniu języka naturalnego.

2.1 Wybrane definicje z dziedziny przetwarzania języka naturalnego

> Definicja
>
> **Język naturalny** to, w odróżnieniu od języka formalnego lub języka sztucznego, język stosowany przez ludzi w komunikacji i wytworzony bez odgórnego planu wraz z rozwojem pewnej kultury lub cywilizacji na przestrzeni wielu pokoleń.

> Definicja
>
> **Przetwarzanie języka naturalnego** jest dziedziną sztucznej inteligencji zajmującą się automatyczną analizą i generowaniem tekstu języka naturalnego.

Automatyczna analiza w przetwarzaniu języka naturalnego stanowi zazwyczaj zestawienie następujących procesów:

1. Segmentacja
2. Tokenizacja
3. Analiza leksykalna
4. Parsowanie

Wymienione procesy zostaną omówione w kolejnych podrozdziałach.

2.1.1 Segmentacja i tokenizacja

> Definicja
>
> **Segmentacja** jest procesem analizy tekstu polegającym na dzieleniu dłuższego tekstu na części (segmenty) będące jednostkami znaczeniowymi – najczęściej zdaniami

> Definicja
>
> **Token** jest ciągiem znaków z alfabetu języka ograniczonym separatorami, który stanowi podstawową niepodzielną jednostkę w łańcuchu tekstu.

> Definicja
>
> **Tokenizacja** to proces podziału łańcucha tekstu na tokeny w oparciu o używany w zadanym języku naturalnym zestaw separatorów.

Przykład 2.1

Wynikiem tokenizacji poniższego segmentu tekstu:

Usłyszał szczęk kluczy.

będą trzy tokeny:

usłyszał, *szczęk*, *kluczy*

2.1.2 Analiza leksykalna i POS-tagging

Zadaniem analizy leksykalnej jest przypisanie wyszczególnionym tokenom interpretacji. Ponieważ pojedynczy token języka naturalnego może być wieloznaczny, analiza leksykalna może zakończyć się zwróceniem listy możliwych interpretacji tokenu. W ramach analizy leksykalnej możliwe jest również dokonanie ujednoznacznienia i podanie tylko jednej interpretacji tokenu.

> Definicja
>
> **Tag** jest opisem znaczenia morfologicznego (części mowy oraz atrybutów) tokenu.

> Definicja
>
> **POS-tagging** (**part-of-speech tagging**), czyli tagowanie częściami mowy, jest procesem przypisania najodpowiedniejszego tagu ze zbioru dostępnych tagów do tokenu. Odpowiedniość tagu określa się na podstawie kontekstu występowania tokenu.

Przykład 2.2

Usłyszał szczęk kluczy

Wyraz „szczęk” może mieć dwie interpretacje:

szczęk[1] – dopełniacz liczby mnogiej od „szczęka”

szczęk[2] – biernik liczby pojedynczej od „szczęk” (mechaniczny odgłos)

Ujednoznacznienie polega w tym przypadku na wybraniu poprawnej interpretacji spośród tych dwóch i przypisanie jej do tokenu *szczęk*.

POS-tagger na podstawie zastanego kontekstu wybierze najbardziej prawdopodobną interpretację. Kontekstem są oczywiście wyrazy otaczające aktualnie rozpatrywany. Na podstawie ich tagów można zbadać wiele zależności występujących w języku, jak choćby fakt, że wystąpienie pary szczęk[1][rzeczownik w dopełniaczu] jest rzadsze niż pary szczęk[2][rzeczownik w dopełniaczu].

Wyraz „kluczy” może mieć dwie interpretacje:

kluczy[1] – czasownik „kluczyć” w trzeciej osobie liczby pojedynczej

kluczy[2] – dopełniacz liczby mnogiej od „klucz”

POS-tagger w tej sytuacji oprze się na częstotliwości występowania kombinacji tokenów „szczęk kluczy” w różnych znaczeniach i wybierze to bardziej prawdopodobne.

Wyjaśnienia wymaga również różnica między zbiorem znaczeń przypisywanych tokenom a faktycznie używanymi przez językoznawców częściami mowy. Nie są to te same zbiory. Tagi używane w automatycznej analizie języka naturalnego są o wiele liczniejsze.

Przykład 2.3

[Jurafsky Martin 2000] podaje następujący zbiór części mowy języka angielskiego: *noun, verb, adjective, adverb, preposition, determiner, pronoun, conjunction, auxiliary verb, particle, numeral*

Najpopularniejsze tagi używane dla języka angielskiego[3] podaje tabela 2.1:

Tag	Znaczenie
AT	article
BEZ	słowo *is*
IN	preposition
JJ	adjective
JJR	comparative adjective
MD	modal
NN	singular or mass noun
NNP	singular proper noun
NNS	plural noun
PERIOD	.:?!
PN	personal pronoun
RB	adverb
RBR	comparative adverb
TO	słowo *to*
VB	verb, base form
VBD	verb, past tense
VBG	verb, present participle, gerund
VBN	verb, past participle
VBP	verb, non-3rd person singular present
VBZ	verb, 3rd singular present
WDT	wh- determiner (what, which)

Tabela 2.1: Popularne tagi języka angielskiego

3 Za [Manning Schütze 1999]

Mnogość tagów jest podyktowana występowaniem w języku zależności, które pozwalają rozpoznać interpretacje tokenów. Zauważmy, że zadanie POS-taggera dla języków fleksyjnych jest bardziej zaawansowane, co obrazuje poniższy przykład:

Usłyszałem ***szczęk***[2] *kluczy.*

Poszukiwał ***szczęk***[1] *pacjentów.*

W obu powyższych zdaniach wyraz „szczęk" jest rzeczownikiem. POS-tagger powinien jednak przypisać wystąpieniom tego wyrazu różne tagi, odpowiednio: rzeczownik w bierniku liczby pojedynczej i rzeczownik w dopełniaczu liczby mnogiej.

2.1.3 Parsowanie – definicje

W parsowaniu używa się gramatyk formalnych przybliżających nieprecyzyjną gramatykę języka naturalnego. Zwykle gramatyka formalna używana w parsowaniu jest gramatyką bezkontekstową.

Definicja

Gramatyka bezkontekstowa (CFG) to czwórka uporządkowana $G=(V_T, V_N, P, S)$, gdzie:

V_T – skończony zbiór symboli terminalnych (słownik)

V_N – skończony zbiór symboli nieterminalnych

P – skończony zbiór reguł (projekcji) postaci $K \to R$ gdzie $K \in V_N$ i $R \in (V_T \cup V_N)^*$

K nazywamy głową a R ogonem reguły

S – symbol początkowy

W gramatyce bezkontekstowej modelującej język naturalny symbolami terminalnymi są tokeny, stąd zbiór V_T jest nazywany słownikiem. Symbol początkowy S odpowiada zdaniu języka. Reguły gramatyczne pozwalają wyprowadzić z symbolu S zdania języka złożone z tokenów.

Definicja

Weźmy gramatykę $G=(V_T, V_N, P, S)$.

Ciąg symboli $v_1 \dots v_k \in V_T \cup V_N \quad k \geq 1$ jest **bezpośrednio wyprowadzalny** z symbolu v. jeśli istnieje $p \in P$ takie, że $p = v_0 \to v_1 \dots v_k$

Definicja

Weźmy gramatykę $G=(V_T, V_N, P, S)$.

Wyprowadzenie W to ciąg list symboli postaci $W = l_1 \dots l_n$ gdzie $l_i = v_1 \dots v_{k_i} \in V_T \cup V_N \quad k_i \geq 1$, w którym dla dowolnego $i < n$ między l_i a l_{i+1} zachodzi następująca zależność:

lista l_{i+1} zawiera wszystkie elementy z l_i z wyjątkiem jednego elementu v_i, który jest zastąpiony ciągiem symboli bezpośrednio wyprowadzalnych z v_i.

Definicja

Zdanie generowane przez gramatykę $G=(V_T, V_N, P, S)$ to ciąg symboli terminalnych $v_1 \ldots v_k \in V_T \quad k \geq 1$ taki, że istnieje wyprowadzenie $W=l_1 \ldots l_n$ gdzie $l_1=S$ i $l_n=v_1 \ldots v_k$

Przykład 2.4 (gramatyka bezkontekstowa, zdania generowane)

Gramatyka bezkontekstowa $G1=(V_T, V_N, P, S)$ modelująca podzbiór języka naturalnego:

V_T = {usłyszał, szczęk, kluczy, Tomek}

V_N = {V, NP, NA, NG}

P = {S → V NP, S → NN V, NP → NA NG, V → usłyszał, V → kluczy, NA → szczęk, NG → szczęk, NG → kluczy, NN → Tomek }

Wyprowadzenie zdania w gramatyce *G1*:

1. S
2. V NP (z: S → V NP)
3. V NA NG (z: NP → NA NG)
4. usłyszał NA NG (z: V → usłyszał)
5. usłyszał szczęk NG (z: NA → szczęk)
6. usłyszał szczęk kluczy (z: NG → kluczy)

Zdania generowane przez gramatykę *G1* to:

Usłyszał szczęk szczęk. Usłyszał szczęk kluczy. Kluczy szczęk szczęk. Kluczy szczęk kluczy. Tomek kluczy. Tomek usłyszał.

Definicja

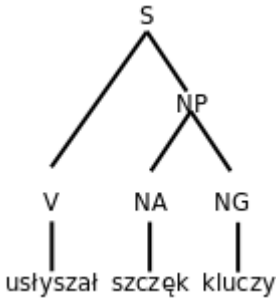
Weźmy gramatykę $G=(V_T, V_N, P, S)$.

Drzewo rozbioru zdania to drzewo uporządkowane, którego:

- korzeń zawiera symbol S
- każdy węzeł zawiera symbol ze zbioru $V_T \cup V_N$
- liście zawierają symbole ze zbioru V_T
- dowolny węzeł nie będącym liściem zawiera symbol $v \in V_N$ oraz ma tylko takie symbole $v_1 \ldots v_k \in V_T \cup V_N$ w węzłach potomnych, że istnieje projekcja $p \in P$ postaci $v \rightarrow v_1 \ldots v_k$

Przykład 2.5 (drzewo rozbioru zdania)

Drzewo rozbioru zdania *Usłyszał szczęk kluczy* przedstawia ilustracja 2.1.



Ilustracja 2.1: Drzewo rozbioru zdania

Definicja

W lingwistyce komputerowej **parsowanie (analiza składniowa)** oznacza proces analizy tekstu celem ustalenia jego struktury gramatycznej w kontekście zadanej gramatyki. Wynikiem parsowania jest pewna reprezentacja formalna struktury zdania, na przykład jego drzewo rozbioru.

Definicja

Parser to program realizujący proces parsowania w oparciu o pewien algorytm.

Parsowanie pozwala poznać składnię zdania poprzez dopasowanie tokenów zdania wejściowego jako liści do pewnego drzewa rozbioru możliwego do wygenerowania z zadanej gramatyki. W procesie parsowania może powstać wiele drzew pasujących do zdania. Sytuację taką nazywamy **niejednoznacznością** syntaktyczną.

Utworzenie drzewa rozbioru zdania może przebiegać w jednym z dwóch kierunków – od korzenia *S* do tokenów wchodzących w skład zdania, lub od tokenów składających się na zdanie do symbolu *S*. Przyjmując jako kryterium kierunek tworzenia drzewa rozbioru można podzielić parsowanie na dwie kategorie – parsowanie wstępujące i zstępujące.

2.1.3.1 Parsowanie wstępujące (Bottom-up)

Parsowanie wstępujące to historycznie pierwsza zaproponowana metoda komputerowej analizy syntaktycznej, która używana jest powszechnie w parsowaniu języków sztucznych (parsery typu „shift-reduce”). Parser wstępujący zaczyna od przyjęcia tokenów zdania wejściowego jako ciągu symboli terminalnych i buduje strukturę drzewa rozbioru zdania zaczynając od tego ciągu w roli liści. Węzły wewnętrzne drzewa są tworzone przez wyszukiwanie reguł, których ogon zawiera elementy występujące w konstruowanym drzewie nie mające jeszcze rodziców i przypisywanie im symbolu z głowy reguły jako rodzica. Drzewo rozbioru jest gotowe, gdy uda się doprowadzić do korzenia zawierającego symbol *S*. Jeżeli drzewo nie ma korzenia *S* i nie ma więcej reguł, które można na nim zastosować, to jest ono odrzucane.

Przykład 2.6 (parsowanie bottom-up)

Ilustracja 2.2 przedstawia cztery kroki parsowania przykładowego zdania metodą bottom-up.

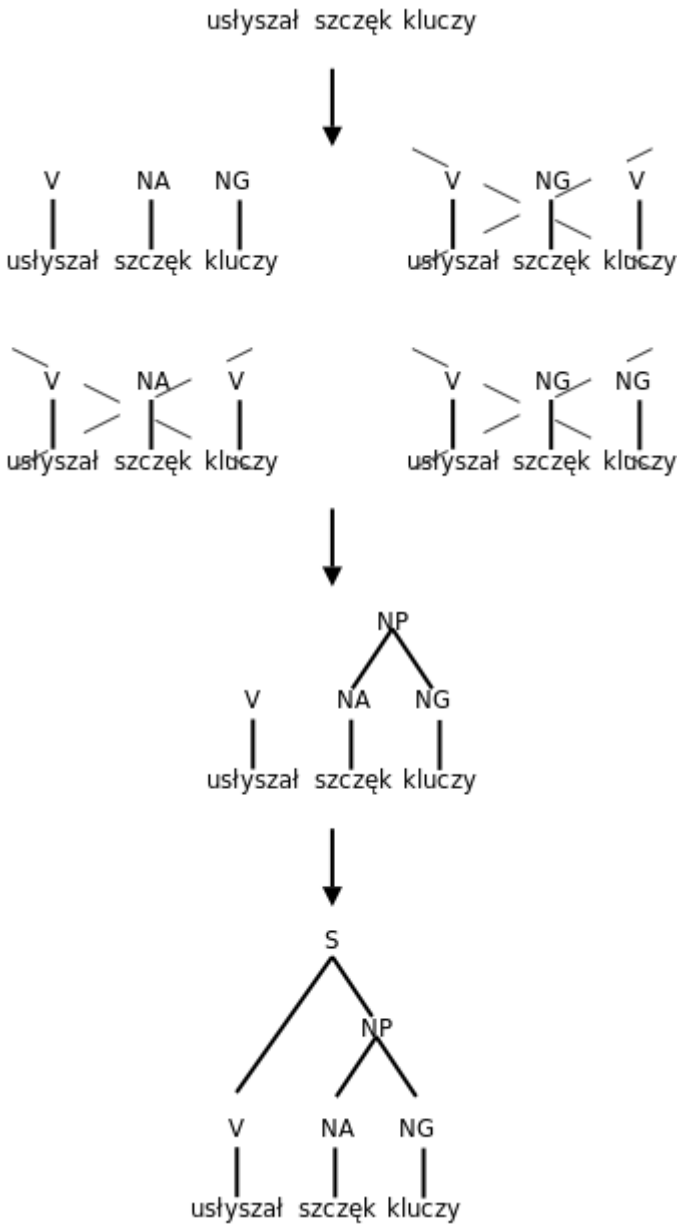
1. Tokeny przyjęte za liście powstającego drzewa.

2. Wszystkie możliwe rozwinięcia symboli terminalnych według reguł gramatyki *G1*.

   Niedokończone drzewa, których nie można dalej rozbudować z użyciem reguł gramatyki *G1* są przekreślone.

3. Rozbudowanie drzewa zgodnie z regułą NP → NA NG

4. Rozbudowanie drzewa zgodnie z regułą S → V NP i zakończenie parsowania.



Ilustracja 2.2: Tworzenie drzewa rozbioru w parsowaniu bottom-up

2.1.3.2 Parsowanie zstępujące (Top-down)

Parser zstępujący poszukuje drzewa rozbioru zdania poprzez budowanie możliwych drzew zaczynając od symbolu początkowego S i dochodząc do liści. Drzewo, z którego nie da się wyprowadzić zdania parsowanego jest odrzucane. W parsowaniu zstępującym przyjmuje się założenie, że zdanie wejściowe może być wyprowadzone z symbolu początkowego S w wyniku zastosowania reguł gramatyki.

Przykład 2.7 (parsowanie top-down)

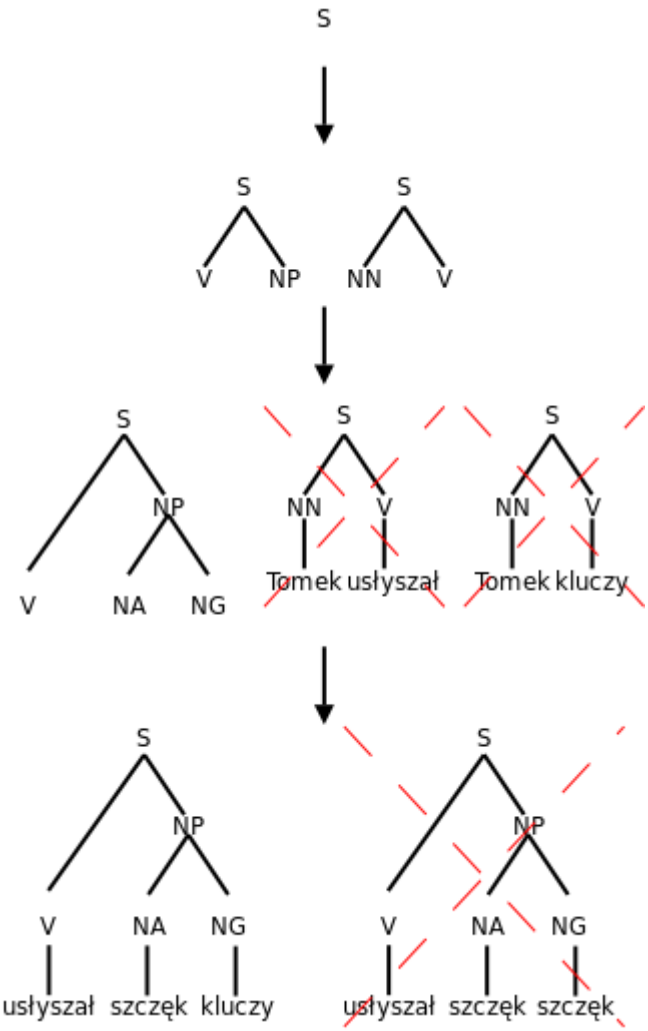
Ilustracja 2.2 przedstawia cztery kroki parsowania przykładowego zdania metodą top-down.

1. Rozpoczęcie od symbolu *S*

2. Rozwinięcie *S* regułami gramatyki *G1* kolejno: S → V NP i S → NN V

3. Kolejne rozwinięcie powstałych drzew. Symbol terminalny *Tomek* nie występuje wśród tokenów parsowanego zdania, więc następuje odrzucenie dwóch drzew.

4. Ostatnie rozwinięcie i odrzucenie niepoprawnego drzewa. Parsowanie zakończone zwróceniem drzewa rozbioru zdania.



Ilustracja 2.3: Tworzenie drzewa rozbioru w parsowaniu top-down

2.1.3.3 Niejednoznaczność syntaktyczna

Dla zadanej gramatyki *G* i pewnego zdania wejściowego może istnieć więcej niż jedno drzewo rozbioru zdania. Sytuację, gdy zdanie można sparsować na wiele sposobów nazywa się niejednoznacznością syntaktyczną. W językach naturalnych niejednoznaczność syntaktyczna jest częstym problemem.

Przykład 2.8 (Niejednoznaczność syntaktyczna)

Zdanie

Widziałem chłopca jedzącego zupę i bociana.

można zinterpretować na dwa sposoby, z których jeden jest wyraźnie korzystniejszy dla bociana. Ilustracja2.4 przedstawia dwa drzewa rozbioru tego zdania dla pewnej gramatyki.

Ilustracja 2.4: Niejednoznaczność syntaktyczna a żywotność bociana

W odróżnieniu od człowieka parser nie potrafi wybrać bardziej prawdopodobnej interpretacji zdania. Parser nie ma dostępu do wiedzy ani kontekstu pozwalających na przyjęcie założenia, że chłopcy sporadycznie jadają bociany. Wiele zastosowań wymaga jednak od parsera zwrócenia tylko jednego, najlepszego zdania. Zastosowania te są polem do użycia gramatyk zawierających dodatkową informację o wadze („ważności") reguł. Temat ten zostanie poruszony w podrozdziale 2.2.

2.2 Rola wag reguł w procesie parsowania

Wagi reguł w gramatyce umożliwiają rozwiązanie niejednoznaczności syntaktycznych podczas parsowania.

Przykład 2.9

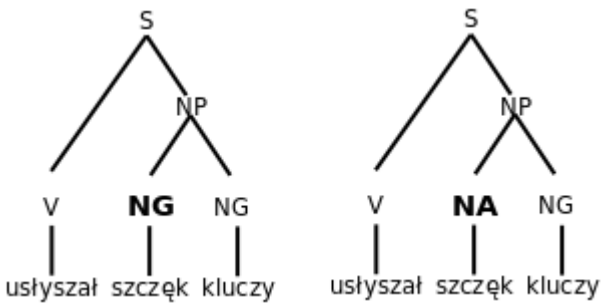
Załóżmy,że gramatyka bezkontekstowa $G2=(V_T, V_N, P, S)$ *została* utworzona w oparciu o gramatykę *G1* z przykładu 2.4 poprzez rozszerzenie o dodatkową regułę:

V_T = {usłyszał, szczęk, kluczy, Tomek}

V_N = {V, NP, NA, NG}

P = {S → V NP, S → NN V, **NP → NG NG**, NP → NA NG, V → usłyszał, V → kluczy, NA → szczęk, NG → szczęk, NG → kluczy, NN → Tomek }

Zdanie *Usłyszałem szczęk kluczy* może być w tej gramatyce sparsowane na dwa sposoby przedstawione na ilustracji 2.5.



Ilustracja 2.5: Dwa możliwe drzewa rozbioru jednego zdania

Aby umożliwić rozwiązanie tej niejednoznaczności, należy wprowadzić do gramatyki *G2* wagi reguł. Załóżmy, że ustalono je w następujący sposób:

Reguła	Waga
S → V NP	0.5
S → NN V	0.5
NP → NG NG	**0.1**
NP → NA NG	**0.9**
V → usłyszał	0.7
V → kluczy	0.3
NA → szczęk	0.7
NG → szczęk	0.3
NG → kluczy	1
NN → Tomek	1

Tabela 2.2: Wagi dla reguł gramatyki G2

Korzystając z informacji o wagach reguł odpowiednio zaimplementowany parser może obliczyć, że spośród dwóch interpretacji zdania *Usłyszałem szczęk kluczy* wyższy wynik uzyska ta interpretacja, która użyje reguły NP → NA NG, nie NP → NG NG. Obliczenia znajdują się w przykładzie 2.10.

Trudnością związaną z wykorzystywaniem gramatyk zawierających informację o wagach reguł jest pozyskanie wartości dla wag. Arbitralne ustalenie wag intuicyjnie lub metodą prób i błędów dla nietrywialnych gramatyk, które przybliżają język naturalny w satysfakcjonującym stopniu, byłoby wielce pracochłonne lub nawet niemożliwe. Jednym z rozwiązań jest zastosowanie gramatyki bezkontekstowej.

Definicja

Bezkontekstowa gramatyka probabilistyczna (PCFG) to piątka uporządkowana $G=(V_T, V_N, P, S, W)$, gdzie:

V_T – skończony zbiór symboli terminalnych (słownik)

V_N – skończony zbiór symboli nieterminalnych

P – skończony zbiór reguł (projekcji) postaci $K \to R$ gdzie $K \in V_N$ i $R \in (V_T \cup V_N)^*$

K nazywamy głową a R ogonem reguły

S – symbol początkowy

$W : P \rightarrow [0,1]$ – funkcja przyporządkowująca każdej projekcji ze zbioru P wartość liczbową z przedziału [0,1] spełniającą warunek:

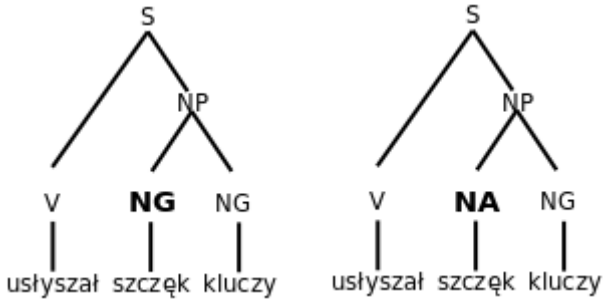
(*) Suma wartości przyporządkowanych przez funkcję W regułom o tej samej głowie wynosi 1.

W gramatyce probabilistycznej wartości zwracane przez funkcję W są traktowane jako prawdopodobieństwo zastosowania danej reguły. Wyznacza się je najczęściej przez sprawdzenie częstości występowania danej reguły w pewnym korpusie poprawnych drzew parsowania.

Gramatyka bezkontekstowa z wagami przypisanymi regułom jest więc gramatyką probabilistyczną tylko jeśli spełnia warunek **(*)**. Gramatyka *G2* w przykładzie 2.9 jest bezkontekstową gramatyką probabilistyczną.

Przykład 2.10

Załóżmy, że parser oblicza wagę całego drzewa rozbioru jako iloczyn wag reguł użytych do jego utworzenia.



0.5 (S → V NP)	0.5 (S → V NP)
0.1 (NP → NG NG)	0.9 (NP → NG NG)
0.7 (V → usłyszał)	0.7 (V → usłyszał)
0.3 (NG → szczęk)	0.7 (NA → szczęk)
1.0 (NG → kluczy)	1.0 (NG → kluczy)
0,0105	**0,2205**

Ilustracja 2.6: Obliczenia prowadzące do wyboru lepszej interpretacji zdania

Na podstawie tych iloczynów parser jest w stanie wybrać lepszą interpretację zdania. W przypadku zastosowania gramatyki probabilistycznej iloczyn prawdopodobieństw reguł należy traktować jako prawdopodobieństwo wystąpienia danego drzewa rozbioru zdania.

Alternatywnym rozwiązaniem jest zautomatyzowanie przydziału arbitralnego. Eksperyment realizujący takie rozwiązanie stanowi część projektową niniejszej pracy i jest opisany w rozdziale 4.

3 Algorytmy genetyczne

Streszczenie

W tym rozdziale opiszę zagadnienie algorytmów genetycznych w sposób ścisły, posługując się autorskimi definicjami opracowanymi w oparciu o mniej sformalizowaną literaturę[4]. W pierwszej kolejności przedstawię znaczenie pojęć z genetyki w kontekście algorytmów genetycznych. Następnie wyjaśnię przebieg algorytmu genetycznego i wprowadzę teorię schematów. W dalszej części rozdziału streszczę wybrane istniejące zastosowania algorytmów genetycznych.

Rys historyczny

Różne koncepcje komputerowej symulacji biologicznych procesów ewolucyjnych powstawały od lat 50' XX wieku i zaowocowały opracowaniem wielu mechanizmów opartych na ewolucji, których działanie bardzo przypominało współczesne algorytmy genetyczne. Same algorytmy genetyczne jako mechanizm stosowany do rozwiązania problemu zostały spopularyzowane przez Johna Hollanda dzięki jego pracy „Adaptation in Natural and Artificial Systems" (1975r). Jednym z celów podjętych przez Hollanda badań było stworzenie oprogramowania, które odtwarzałoby mechanizmy zaobserwowane w systemach biologicznych i pozwalało skorzystać z nich w systemach informatycznych. Tym sposobem schemat działania procesów ewolucji, który sprawdza się od tysięcy lat w świecie biologii, znalazł zastosowanie również w problemach obliczeniowych poprzez zaobserwowanie analogii między definicją osobnika zapisaną w genach a definicją rozwiązania problemu przez atrybuty tego rozwiązania.

Również w 1975 roku De Jong opublikował swoją pracę „An Analysis of the Behaviour of a Class of Genetic Adaptive Systems", w której połączył teorie Hollanda ze starannie zaprojektowanymi eksperymentami badającymi wpływ parametrów algorytmu genetycznego na jego zbieżność w różnych przestrzeniach przeszukiwania.

Intuicja

Dla problemu, którego każde potencjalne rozwiązanie można opisać jako n-elementową listę atrybutów (każdy atrybut o skończonej liczbie wartości) tego rozwiązania można znaleźć rozwiązanie optymalne sprawdzając wszystkie możliwe kombinacje atrybutów w pewnej kolejności. Algorytm genetyczny jest optymalizacją tego procesu realizowaną przez sprawdzanie niewielu rozwiązań oraz pomijanie wielu rozwiązań nie rokujących na bycie tym optymalnym.

> Definicja
>
> **Algorytm genetyczny** to algorytm przeszukiwania skończonej, dyskretnej, n-wymiarowej ($n<\infty$) przestrzeni rozwiązań zadanego problemu w poszukiwaniu rozwiązań optymalnych, którego działanie wzorowane jest na mechanizmie doboru naturalnego oraz dziedziczenia z uwzględnieniem czynnika losowego. Algorytm genetyczny operuje na zbiorze punktów z przeszukiwanej przestrzeni i w kolejnych iteracjach przetwarzania poddaje ocenie ich jakość.

4 [Goldberg 1995], [Eiben Smith 2003]

Przestrzeń przeszukiwania jest na potrzeby algorytmu genetycznego definiowana jako słownik wszystkich słów długości n (n-elementowych ciągów kodowych) nad pewnym alfabetem A.

3.1 Podstawowe definicje z dziedziny algorytmów genetycznych

W dziedzinie algorytmów genetycznych używa się wielu pojęć z genetyki do opisu bytów ze świata problemów obliczeniowych. Tabela 3.1 zawiera definicje pojęć potrzebnych do późniejszego analizowania działania algorytmów genetycznych i opisu ich działania.

Pojęcie	Znaczenie w genetyce	Znaczenie w algorytmach genetycznych
Allel	wersja określonego genu – kombinacja nukleotydów	wariant cechy, możliwa wartość genu, czyli znak z rozpatrywanego alfabetu
Gen	fragment DNA kodujący jedno białko	pewna cecha rozwiązania, zmienna przechowująca znak z rozpatrywanego alfabetu
Locus	umiejscowienie genu w łańcuchu, czyli jego funkcja	określenie cechy rozwiązania, na jaką wpływa dany gen – pozycja genu w ciągu kodowym
Chromosom	forma organizacji materiału genetycznego	ciąg kodowy składający się z pojedynczych genów
Genotyp	zespół genów danego osobnika warunkujących jego cechy	pełna specyfikacja jednego rozwiązania; genotyp składa się z jednego lub więcej chromosomów
Osobnik	pojedynczy organizm, zdefiniowany przez genotyp	element populacji, obiekt posiadający genotyp
Fenotyp	zespół cech organizmu – manifestacja jego kodu genetycznego	rozwiązanie wyznaczane przez genotyp; osobnika ocenia się poddając ocenie jego fenotyp
Przystosowanie	stopień dopasowania cech osobnika do środowiska	funkcja mierząca jakość rozwiązania reprezentowanego przez danego osobnika
Populacja	zbiór osobników jednego gatunku żyjących równocześnie w określonym środowisku	zbiór rozwiązań poddawanych pod działanie operatorów przeszukiwania w jednej iteracji

Pojęcie	Znaczenie w genetyce	Znaczenie w algorytmach genetycznych
Reprodukcja	wytwarzanie nowych osobników przez organizmy rodzicielskie	losowe tworzenie nowej populacji z puli aktualnej populacji z prawdopodobieństwami wynikającymi z oceny przystosowania osobników

Tabela 3.1: Pojęcia z genetyki używane do opisu algorytmów genetycznych

3.1.1 Operatory przeszukiwania

Przedstawię tu operatory przeszukiwania przy założeniu, że nie zdefiniowano dodatkowych zależności między genami w genotypie, których utrzymanie jest wymagane, by rozwiązanie było dopuszczalne. Dla reprezentacji, w których między genami występują zależności, stosuje się indywidualnie zaprojektowane operatory przeszukiwania.

3.1.1.1 Krzyżowanie

W genetyce krzyżowanie to proces wymiany odcinków kodu genetycznego w ramach chromosomu pochodzącego z jednego rodzica na analogiczny fragment pochodzący z chromosomu drugiego rodzica. Krzyżowanie zachodzi na etapie powstawania komórki potomnej.

W dziedzinie algorytmów genetycznych krzyżowanie to proces wymiany fragmentów informacji.

> Definicja
>
> **Krzyżowanie** w algorytmie genetycznym to proces złożony z dwóch etapów:
>
> 1. Dobranie osobników z populacji w pary w sposób losowy
>
> 2. Wybranie w sposób losowy jednej spośród *l*-1 (*l* to długość każdego chromosomu) pozycji (punktu krzyżowania) w każdym chromosomie jednego z osobników w parze, a następnie utworzenie dwóch osobników, których chromosomy będą się składały odpowiednio: w pierwszej części z genów pierwszego osobnika i w drugiej drugiego, oraz w pierwszej części z genów drugiego osobnika i w drugiej pierwszego.

Objaśnienie wizualne do etapu drugiego:

genotyp osobnika1: *1111111111*

genotyp osobnika2: *2222222222*

możliwe miejsca podziału i **wylosowane miejsce**:

1|1|1|1|1|1|1|1|1|1

genotyp potomka1: *1111222222*

genotyp potomka2: *2222111111*

Możliwe jest również krzyżowanie wielopunktowe, w którym losuje się wiele punktów krzyżowania i tworzy dwa genotypy potomne wybierając do nich odcinki naprzemiennie.

genotyp osobnika1:	genotyp osobnika2:
1111111111	*2222222222*

możliwe miejsca podziału i **wylosowane miejsca**:

1|1|1|1|1|1|1|1|1|1

genotyp potomka1:	genotyp potomka2:
1112211222	*2221122111*

Krzyżowanie jest w algorytmie genetycznym podstawowym mechanizmem odkrywania rozwiązań. Wraz z reprodukcją umożliwia odnajdywanie najlepszych kombinacji alleli (parametrów rozwiązań) na odcinkach chromosomów (ciągów kodowych).

3.1.1.2 Mutacja

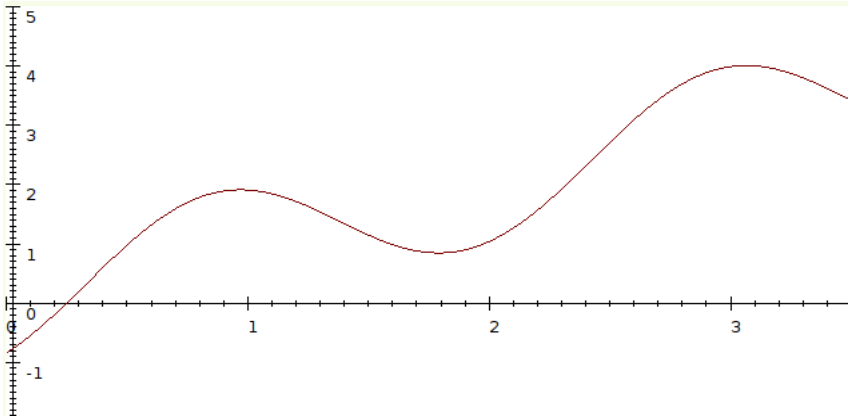
W procesach biologicznych mutacją jest każda przypadkowa nieprawidłowość w skopiowaniu fragmentu kodu genetycznego. Mutacje występują rzadko i zwykle nie prowadzą do osiągnięcia lepszego przystosowania. Pomimo tego są pożyteczne w procesie ewolucji.

> Definicja
>
> W algorytmie genetycznym **mutacja** jest to losowe (zwykle o niskim prawdopodobieństwie) zaburzenie wartości wybranych genów w genotypie następujące po reprodukcji i krzyżowaniu lub w trakcie ich przeprowadzania.

Mutacja służy umożliwieniu wprowadzenia do populacji genotypów, które w pewnym locus posiadają wartość, która w tym locus nie występowała dotychczas u żadnego osobnika w populacji. Oznacza to w praktyce rozpatrzenie rozwiązania, którego nie ma możliwości otrzymania poprzez wymienianie fragmentów istniejących rozwiązań w procesie krzyżowania. Mutacja zapewnia większą odporność algorytmu genetycznego na problemy posiadające lokalne maksima optymalności rozwiązania.

Przykład 3.1

Rozważmy problem poszukiwania maksimum funkcji $f(x)=\sin(3x-1)+x$ na przedziale [0,3] przedstawionej na ilustracji 3.1.



Ilustracja 3.1: Wykres funkcji f(x)

Weźmy trywialny algorytm genetyczny o dwuelementowej populacji poszukujący optymalnej wartości x, dla której f ma najwyższą wartość w tym przedziale. Argument x jest kodowany binarnie na 2 bitach.

Dysponujemy więc wyłącznie następującymi wartościami:

$00_2=0_{10}; 01_2=1_{10}; 10_2=2_{10}; 11_2=3_{10}$

Rozpoczynamy z populacją A:**00**, B:**01**

Operator krzyżowania nie pozwala wyjść poza zbiór początkowy, więc spośród tych dwóch wartości wyższy wynik daje **01**, ale nie jest to maksimum na przedziale [0,3]. Dopiero mutacja na starszym bicie liczby pozwoli znaleźć poprawne rozwiązanie.

Załóżmy, że losowa mutacja nastąpiła na starszym bicie osobnika B, wtedy populacja wygląda następująco: A:**00**, B:**11**

Osobnik B jest poprawnym rozwiązaniem, więc algorytm nie popełni już błędu.

Uwaga

Przykład zawiera pewne uproszczenia i pomija jeszcze nie omówione elementy m.in. etap reprodukcji.

Dokładna definicja operatora mutacji zależy od reprezentacji rozwiązania. W reprezentacji binarnej mutacja ma tylko jedną oczywistą postać negacji bitów odpowiadających wybranym genom. Istnieją jednak inne operatory mutacji dostosowane do reprezentacji.

Mutacja w reprezentacji całkowitoliczbowej i zmiennoprzecinkowej oraz innych reprezentacjach o allelach pochodzących ze zbiorów uporządkowanych o liczności większej niż 2 może przybierać dwie powszechnie stosowane postacie:

- Losowe resetowanie – podlegający jej gen ma przypisywaną nową losowo wybraną dopuszczalną wartość.
- Postępująca (pełzająca) mutacja – do podlegającego jej genu dodawana jest liczba dodatnia lub ujemna, której wartość jest wybierana losowo z prawdopodobieństwem wynikającym z rozkładu symetrycznego względem zera i malejącego wraz ze wzrostem wartości bezwzględnej, np. rozkład normalny. W reprezentacji zmiennoprzecinkowej rozkład jest ciągły, w reprezentacji całkowitoliczbowej dyskretny.

Warto zauważyć, że omówione trzy postacie mutacji zastosowane na genotypach z allelami pochodzącymi ze zbioru dwuelementowego są równoważne.

3.2 Zasady działania oraz cechy algorytmów genetycznych

Zastosowanie algorytmu genetycznego do pewnego problemu P należy poprzedzić wprowadzeniem dwóch definicji.

Definicja

Rozwiązanie *R* problemu *P* jest słowem długości *n* nad pewnym alfabetem *A*, przy czym dla zadanego problemu *P* alfabet *A* oraz długość *n* są takie same dla każdego istniejącego rozwiązania.

Z powyższego wynika, że rozwiązanie problemu musi zostać tak zdefiniowane, żeby można je było przedstawić jako pewien uporządkowany zbiór cech je opisujących. Przyjęty alfabet musi umożliwiać wyrażenie wszystkich możliwych rozwiązań zadanego problemu jako n-elementowych słów nad tym alfabetem.

Definicja

Słowo długości *n* nad alfabetem *A*, które nie jest akceptowalnym rozwiązaniem problemu, nazywamy **rozwiązaniem niedopuszczalnym**.

Implementacja algorytmu genetycznego dla zadanego problemu musi gwarantować, żeby rozwiązania niedopuszczalne nie mogły znaleźć się w populacji.

Definicja

Funkcja przystosowania (nazywana również jakością rozwiązania) to funkcja $f : L \to \mathbb{R}$ gdzie *L* jest słownikiem wszystkich wyrazów długości *n* nad alfabetem *A*. Funkcja ta może nie być zdefiniowana dla argumentów będących rozwiązaniami niedopuszczalnymi.

Jeśli obrazem *O* zbioru *L* przez funkcję *f* jest zbiór jednoelementowy to zdefiniowany funkcją *f* problem nie ma rozwiązania lub jest problemem trywialnym (dowolny ciąg jest poprawnym rozwiązaniem) zależnie od interpretacji jaką nadamy wartości funkcji *f*.

Pożądane jest, by obrazem zbioru *L* przez funkcję *f* był zbiór o znaczącej liczbie elementów.

3.2.1 Działanie algorytmu genetycznego

Podstawowy algorytm genetyczny ma następujący przebieg:

1. Losowanie populacji początkowej.
   - Dla zadanej wielkości populacji *n* i wyznaczonego genotypu *G* (sposobu reprezentacji) osobnika, wszystkim genom wchodzącym w skład genotypu każdego z osobników przypisuje się losową wartość ze zbioru wartości dopuszczalnych dla danego genu.
1. Poddanie każdego osobnika w populacji ocenie.
   - Niech *G* będzie genotypem osobnika. Wtedy $f(G)$ jest oceną przystosowania osobnika przez funkcję przystosowania *f*.
1. Reprodukcja – tworzenie osobników do nowego pokolenia.
   - Reprodukcja jest odpowiednikiem darwinowskiego doboru naturalnego. Polega na wybraniu do nowego pokolenia populacji w taki sposób, by tych o wyższym wyniku funkcji przystosowania było więcej.

- Najprostszą realizacją reprodukcji jest „ruletka”. *Realizacja ta polega na przydzieleniu każdemu osobnikowi wycinka koła ruletki proporcjonalnego do jego oceny przystosowania, a następnie przeprowadzenie losowania n razy, co daje nową n*-elementową populację.

1. Krzyżowanie osobników w nowym pokoleniu i losowe mutacje.

   - W tym kroku dokonuje się krzyżowanie na losowo dobranych parach z nowej populacji. Następnie (lub, zależnie od implementacji, jednocześnie) wprowadza się mutacje z zadanym prawdopodobieństwem

1. Przyjęcie nowego pokolenia jako aktualnej populacji.

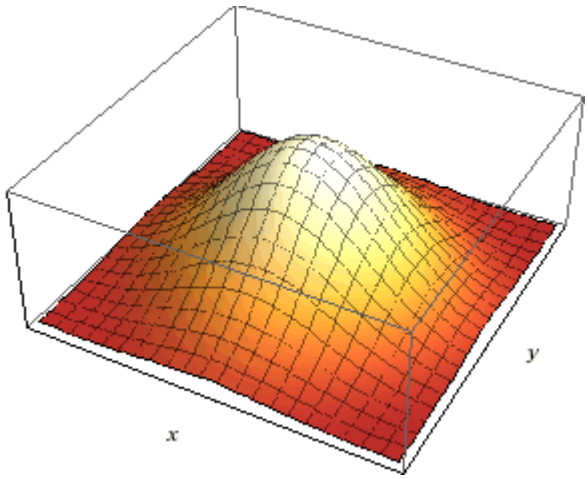
2. Jeśli populacja nie jest zdominowana przez osobniki o tym samym genotypie ani nie skończył się zadany czas – skok do punktu 2.

3. Zakończenie działania i zwrócenie wyniku.

   - Wynikiem jest oczywiście genotyp osobnika, który zdominował populację.

Przykład 3.2

Rozważmy prosty problem poszukiwania maksimum funkcji $f(x,y)=(\sin(x)+\sin(y))^4$ na przedziale $(0,\pi)x(0,\pi)$ przedstawionej na ilustracji 3.2.



Ilustracja 3.2: Wykres funkcji f(x,y)

Funkcja *f* jest w tym przypadku funkcją przystosowania.

Rozwiązanie natomiast zdefiniujemy jako $(a*\pi, b*\pi)$ *gdzie* a i b są wielkościami $0{,}1*k\ dla\ k\in\{0,1,\ldots,10\}$ i przeprowadzimy obliczenia na 4-elementowej populacji.

Oczywiście od początku wiadomo, że najlepszy wynik otrzymamy dla pary (0,5;0,5)

1. Losujemy populację

	a	b
osobnik1	0,1	0,5
osobnik2	0,8	0,1
osobnik3	0,6	0,2
osobnik4	0,3	0,0

Tabela 3.2: Populacja początkowa

2. Poddajemy osobniki ocenie, poprzez obliczenie wartości funkcji $f(a*\pi, b*\pi)$

0,1	0,5	**2,94**
0,8	0,1	**0,65**
0,6	0,2	**5,61**
0,3	0,0	**0,43**

Tabela 3.3: Oceny osobników

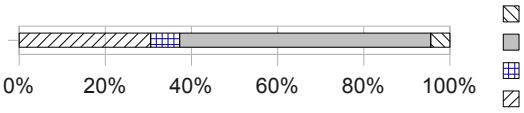
3. Dokonujemy reprodukcji

   - Aby ustawić parametry losowania w ruletce wyznaczamy procentowy udział wyników osobników w sumie wszystkich ocen.

0,1	0,5	**2,94**	31%
0,8	0,1	**0,65**	7%
0,6	0,2	**5,61**	58%
0,3	0,0	**0,43**	4%
		9,62	100,00%

Tabela 3.4: Udział ocen osobników w sumie wszystkich ocen

   - Następnie tworzymy „ruletkę”:



Ilustracja 3.3: Rozkład ruletki

   - losujemy n=4 liczb, które wskażą wybrane osobniki.

Przykładowe liczby zwrócone przez generator liczb pseudolosowych:

7, 15, 57, 79

   - Wynik reprodukcji:

0,1	0,5
0,1	0,5
0,6	0,2
0,6	0,2

Tabela 3.5: Nowe pokolenie wybrane za pomocą ruletki

4. Dobieramy osobniki w pary – w naszym przypadku będą to dwie identyczne pary (0,1; 0,5) (0,6; 0,2). Wykonujemy na nich operację krzyżowania, która

może zajść tylko na jednej pozycji – pomiędzy *a* i *b*. Zapisując wyniki krzyżowania dokonujemy jednej losowej mutacji (załóżmy, że trafia ona na gen *a* pierwszego osobnika).

0,5	0,5
0,1	0,5
0,6	0,5
0,1	0,5

Tabela 3.6: Rezultat mutacji

5. Zapisujemy wynik przedstawiony w tabeli 3.6 jako nową populację.
6. Populacja nie została zdominowana, więc przeprowadzamy dalsze obliczenia.

Oceniamy populację i kontynuujemy.

0,5	0,5	**16**	44%
0,1	0,5	**2,94**	8%
0,6	0,5	**14,49**	40%
0,1	0,5	**2,94**	8%
		36,36	100,00%

Tabela 3.7: Oceny kolejnego pokolenia

Wyraźnie już widać, że gen *b* zachowa poprawną wartość, dla genu *a* po reprodukcji najprawdopodobniej zostaną już tylko dwie wartości, między którymi o zwycięstwie rozstrzygnie kolejny cykl.

Wreszcie rozwiązanie (0,5; 0,5) zdominuje populację i działanie algorytmu się zakończy.

3.2.2 Odporność, uniwersalność i skuteczność algorytmów genetycznych

Algorytmy genetyczne znalazły wiele interesujących zastosowań w różnych dziedzinach i pod względem zróżnicowania zastosowań przewyższają inne rozwiązania optymalizacyjne. Istnieje wiele algorytmów, które dużo lepiej radzą sobie z konkretnymi typami zadań, ale tylko algorytmy genetyczne mogą być zastosowane z powodzeniem w tak wielu typach.

Znane metody przeszukiwania można podzielić na: analityczne, enumeratywne i losowe[5].

- Metody analityczne opierają się na matematycznej optymalizacji z wykorzystaniem układów równań nieliniowych, pochodnych (np. w metodzie gradientu) oraz wielu innych narzędzi dostępnych dzięki analizie matematycznej i dlatego są metodami rozwiązującymi niewielki zbiór problemów - tzw. metodami wyspecjalizowanymi. Każda z takich metod

5 Według [Goldberg 1995]

jest bardzo wydajna i skuteczna dla problemu, pod który była przygotowana, natomiast dla innych problemów okazuje się niewydajna lub nawet zwraca wyniki niepoprawne (np. maksimum lokalne zamiast globalnego).

- Metody enumeratywne, czyli polegające na sprawdzaniu kolejnych rozwiązań w pewnym porządku, radzą sobie z większą klasą zadań, ale ich wydajność jest niska zwłaszcza dla rozleglejszych przestrzeni przeszukiwania.

- Metody losowe również mają szersze zastosowania niż analityczne, ale ich skuteczność często za bardzo zależy od czynnika losowego i w ogólnym przypadku mogą wymagać zbyt wiele czasu na znalezienie rozwiązania.

W tym kontekście widać atut algorytmów genetycznych – potrafią rozwiązać każde zadanie, którego warunki da się wyrazić w postaci pary: lista atrybutów rozwiązania i funkcja oceny rozwiązania. Dla takich problemów wykazują podobną wydajność i ta cecha świadczy o wysokiej odporności metody. Algorytmy genetyczne można z powodzeniem zastosować do problemów, które z większą wydajnością mogłyby rozwiązać tylko metody wyspecjalizowane (użyteczne tylko dla jednego zadania).

Swoją odporność i uniwersalność algorytmy genetyczne zawdzięczają przede wszystkim temu, że nie wprowadzają żadnych założeń do rozpatrywanego problemu, wręcz abstrahują od definicji samego problemu. W odróżnieniu od metod klasycznych nie wymagają, by sposób osiągnięcia celu był znany. Wystarczy, że potrafimy zdefiniować reprezentację rozwiązania i ocenić jego przydatność. Co więcej, funkcja przystosowania może być względem algorytmu genetycznego zupełnie niezależnym tworem – traktuje się ją jak czarną skrzynkę.

3.2.2.1 Pojęcie schematu

Proces zachodzący na populacji w algorytmie genetycznym przejawia pewne nie wspomniane jeszcze cechy. W przykładzie 3.2 algorytm genetyczny na genotypie opisującym dwie cechy doprowadził w pierwszej kolejności do ustalenia pożądanej wartości dla jednej z nich. W większej skali, czyli dla problemów charakteryzowanych większą liczbą cech, można powiedzieć, że w ogólności praca algorytmu genetycznego polega na rozpoznawaniu fragmentów ciągu kodowego, które dają poprawę wyniku funkcji przystosowania przez wyszukiwanie podobieństw pomiędzy najlepszymi osobnikami. W procesie krzyżowania powstają między innymi genotypy, które zawierają korzystne fragmenty genotypów rodziców a wtedy poprawa wyniku jest jeszcze bardziej widoczna.

W rzeczywistości więc algorytm genetyczny nie przeszukuje po prostu kolejnych wartości w nieokreślonej kolejności, ale zestawem bardzo prostych metod przetwarza zbiory cech (genotypy) zwracając uwagę na korelacje występowania określonej cechy lub grupy cech z poprawą wyniku. Takie podejście do problemu jest kluczem do skuteczności algorytmów genetycznych

Można wyróżnić klasy złożone z grup cech, które w określony sposób wpływają na wynik funkcji przystosowania. Zauważamy, że pojedynczy genotyp reprezentuje wiele takich klas.

Przykład 3.3

Dla zapisu binarnego liczby na 5 pozycjach liczba 11001 reprezentuje m.in. dwie klasy:

1100**1** - liczb nieparzystych dzięki cyfrze 1 na najniższej pozycji

11001 – liczb większych niż $(15)_{10}$, dzięki cyfrze 1 na najwyższej pozycji.

Do analizy zachowania algorytmu genetycznego w kontekście poszukiwania korzystnych cech służy pojęcie schematu wprowadzone pierwszy raz przez Johna Hollanda.

Definicja

Schemat S zdefiniowany dla rozwiązań problemu P jest słowem długości *n* nad alfabetem $A' = A \cup \{*\}$, gdzie *n* jest długością rozwiązania, a *A* alfabetem, nad którym zdefiniowane są rozwiązania. Znak * jest metasymbolem reprezentującym w schemacie dowolną wartość z alfabetu *A*.

Pozycją ustaloną w schemacie S jest każda pozycja o wartości z alfabetu *A*, czyli każda z symbolem innym niż *.

Klasy z przykładu 3.3 można wyrazić jako schematy:

****1 – liczby nieparzyste

1**** - liczby większe od 15

Schematy umożliwiają zwarte i przejrzyste wyrażanie podobieństw pomiędzy rozwiązaniami.

Definicja

Schemat S **pasuje do** rozwiązania (genotypu) R jeśli dla każdego znaku rozwiązania R w schemacie S na odpowiadającej mu pozycji występuje ten sam symbol lub metasymbol *.

Analogicznie można powiedzieć, że rozwiązanie R **reprezentuje** schemat S

Jeśli schemat *S* pasuje do R_1 i do R_2, to *S* opisuje **podobieństwo** między R_1 i R_2.

Jeden genotyp reprezentuje 2^n schematów, gdyż pasujący schemat może mieć na każdej z *n* pozycji jedną z dwóch wartości: * lub wartość z odpowiadającej pozycji w genotypie. Z tego wynika, że *k*-elementowa populacja będzie reprezentować nie więcej niż $k2^n$ schematów.

Przyjrzyjmy się teraz wykonaniu algorytmu genetycznego z wykorzystaniem wiedzy o schematach. Na zbiór schematów reprezentowanych w populacji najbardziej interesujący wpływ ma operacja krzyżowania, natomiast operacja reprodukcji może wpłynąć tylko na liczebność reprezentantów danego schematu a mutacja pozwala rozszerzyć zbiór unikatowych schematów występujących w populacji, co jakkolwiek istotne, nie jest trudne do prześledzenia. Można powiedzieć, że zastosowanie schematów w analizie przebiegu algorytmu genetycznego służy tylko uwypukleniu interesujących następstw operacji krzyżowania.

Wpływ krzyżowania na zadany schemat jest wynikiem przeprowadzenia operacji krzyżowania na pewnym jego reprezentancie. Oznacza to, że tak jak jego reprezentant, schemat zostanie rozerwany i rozerwanie może nastąpić w dowolnie (losowo) wybranym miejscu. Z tego wynika, że rozerwanie schematu *1**1 jest dużo bardziej prawdopodobne niż rozerwanie schematu **11*, ponieważ pierwszy z nich nie zostanie rozerwany tylko gdy podział nastąpi na pierwszej od początku ciągu pozycji, natomiast rozerwanie drugiego nastąpi tylko gdy podział przebiegnie między dwoma jedynkami. Na potrzeby analizowania rozrywania schematów wprowadzono pojęcie rozpiętości schematu.

Definicja

Rozpiętością schematu S nazywamy odległość między dwoma skrajnymi pozycjami ustalonymi i oznaczamy ją przez $\delta(S)$

Przykład 3.4

S_1 = *1**1; $\delta(S_1)=3$

S_2 = **11*; $\delta(S_2)=1$

Schematy o minimalnej rozpiętości nazywane są „cegiełkami”. W procesie krzyżowania istnieje duże prawdopodobieństwo nie tylko przetrwania cegiełek w następnym pokoleniu, ale również połączenia się cegiełek w wyniku skrzyżowania dwóch różnych schematów o minimalnej rozpiętości.

Definicja

Rzędem schematu S nazywamy liczbę pozycji ustalonych w schemacie S i oznaczamy go przez $o(S)$

Warto zauważyć, że w miarę postępu w odnajdywaniu cech sprzyjających wysokim wartościom funkcji przystosowania rośnie liczba schematów wyższych rzędów reprezentowanych przez większość osobników w populacji.

Przykład 3.5

S_1 = *11**; S_2 = ***11;

Załóżmy, że krzyżowanie rozrywa genotypy w miejscu między pozycją 3 i 4. Wtedy po skrzyżowaniu dwóch genotypów reprezentujących schematy S_1 i S_2 otrzymamy jeden genotyp potomny reprezentujący schemat S_3 = *1111

Reprezentowanie schematu rzędu *k* przez wszystkie osobniki w populacji oznacza, że *k* cech odpowiadających pozycjom ustalonym schematu nie ulegnie już zmianie na drodze krzyżowania.

Uwidacznia się tu również ukryta równoległość przetwarzania w algorytmie genetycznym. Skoro jeden osobnik reprezentuje wiele schematów, to wszystkie one są analizowane równolegle gdy ten osobnik podlega operacji reprodukcji i krzyżowania.

Jako podsumowanie wyjaśnienia teorii schematów przedstawiam przykład z opisem schematów przetwarzanych w jego toku. Dla uproszczenia w przykładzie będę nadal rozpatrywał przypadek pięciocyfrowych liczb binarnych i pominę kroki, które nie mają wpływu na zestaw schematów reprezentowanych w populacji.

Przykład 3.6

Problem - znalezienie największej liczby 5-cyfrowej w systemie dwójkowym. Funkcją przystosowania jest w tym przypadku funkcja zwracająca wartość liczby.

- Populacja początkowa i lista reprezentowanych schematów (wypisane te bardziej interesujące)

10000	1****, *0***, ..., ****0, 10***, ...
00100	0****, *0***, **1**, ..., 00***, ...
01010	0****, *1***, **0**, ***1*, ..., *1*1*, ...
01100	0****, *1***, **1**, ..., *11***, ...

Tabela 3.8: Schematy reprezentowane w populacji

- Populacja po reprodukcji podlega krzyżowaniu:

10|000	00|100	**010|10**	011|00
10100	00**000**	**010**00	011**10**

Ilustracja 3.4: Schematy w procesie krzyżowania

Podkreśleniem wyróżniony jest proces budowania z „cegiełek". Schemat o rozpiętości $\delta(S)=2$ został zachowany. Ciąg 01110 nadal reprezentuje m. in. schemat *11**.

- Otrzymujemy nową populację z nowymi schematami:

10100	1****, ..., 1*1**, ...
00000	0****, *0***, ..., 00***, ...
01000	0****, *1***, **0**, ...
01110	0****, *1***, **1**, ..., *11***, ..., *111*, ...

Tabela 3.9: Schematy reprezentowane w populacji

Wśród reprezentowanych schematów zawierających na pozycjach ustalonych tylko jedynki pojawił się pierwszy schemat rzędu $o(S_i)=3$

- Populacja po reprodukcji - najsłabszy osobnik przestaje występować w populacji

10100	1****, ..., 1*1**, ...
10100	1****, ..., 1*1**, ...
01000	0****, *1***, **0**, ...
01110	0****, *1***, **1**, ..., *11**, ..., *111*, ...

Tabela 3.10: Schematy reprezentowane w populacji

- Kolejny etap krzyżowania:

1|0100	0|1000	**1|0100**	0|1110
11000	0**0100**	**1**1110	0**0100**

Ilustracja 3.5: Schematy w procesie krzyżowania

- Po następnej reprodukcji zostanie nam populacja:

11000	1****, ..., 11***, ...
11000	1****, ..., 11***, ...
11110	1****, ..., 11***, ..., 111**, ..., 1111*, ...
11110	1****, ..., 11***, ..., 111**, ..., 1111*, ...

Tabela 3.11: Schematy reprezentowane w populacji w kolejnym pokoleniu

- (...)

Widać już, że schematy 1**** i 11*** są reprezentowane w całej populacji i jako dominujące cechy rozprzestrzeniły się z dużą łatwością. Maksymalny rząd schematów o pozytywnych cechach również znacznie wzrósł.

Rozwiązanie 11111 reprezentuje wszystkie schematy złożone tylko z jedynek i metasymboli *. W przykładzie udało się pokazać, że algorytm genetyczny w każdej iteracji zwiększa liczbę i rząd reprezentowanych dobrych schematów.

Powyższy przykład pozwala zauważyć sposób w jaki krzyżowanie wyizolowuje pewne cechy z dwóch genotypów, a dzięki późniejszej ocenie i reprodukcji w populacji zostaje osobnik potomny z lepszym zestawem cech pochodzących od rodziców, a co za tym idzie – reprezentujący więcej schematów, których reprezentantem jest również poprawne rozwiązanie. Powołując się więc na fakt, że wspólny schemat oznacza podobieństwo, dochodzimy do oczywistego wniosku, że w kolejnych iteracjach algorytmu genetycznego obserwujemy rosnące podobieństwo osobników w populacji do poprawnego rozwiązania.

Uwaga

Teoria schematów wprowadzona przez Hollanda oraz obserwowane dzięki niej procesy zachodzące na „cegiełkach” implikują zalecenie, by alfabet *A* był jak najmniejszy.

Istnieją inne podejścia do kwestii liczności alfabetu, których ideą jest, by pojedyncze geny odpowiadały konkretnym cechom rozwiązania (nie tak jak w przypadku powyższego przykładu, gdzie w zasadzie cecha rozwiązania była tylko jedna). Zastosowanie bardziej skomplikowanych alfabetów i struktur genotypu implikuje ograniczone możliwości stosowania teorii schematów, jak i pewne komplikacje np. w definicji operatora mutacji.

W tym miejscu koniecznie należy zwrócić uwagę na konsekwencje nadmiernego rozszerzania alfabetu – nadmierne rozbudowanie alfabetu redukuje ukrytą równoległość, czyli efekt wynikający z analizowania wielu schematów reprezentowanych przez jednego osobnika.

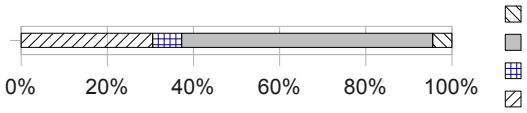
3.2.2.2 Plany reprodukcyjne w eksperymentach De Jonga

Wielu badaczy algorytmów genetycznych zauważyło problem „zatłoczonej

niszy ekologicznej” w swoich badaniach. Problem ten polega na nadmiernym nagromadzeniu bardzo podobnych do siebie osobników. W rzeczywistym świecie nadmiar bardzo podobnych do siebie osobników prowadzi do silnego wzrostu konkurencyjności i zmniejszenia przeżywalności i rozrodczości organizmów. Algorytmy genetyczne nie symulowały tej cechy środowiska. De Jong wymusił podobną presję w algorytmie genetycznym wprowadzając zmianę w operacji reprodukcji. W swoim eksperymencie analizował różne plany reprodukcyjne również pod względem fluktuacji losowej, czyli odchyleń liczby reprodukowanych osobników od wartości oczekiwanej wynikającej z rozkładu prawdopodobieństwa ich reprodukcji. Wynikało to z obserwacji, że w klasycznej ruletce może się zdarzyć, że losowe wybieranie osobników doprowadzi do utworzenia populacji, w której liczebności osobników po reprodukcji będą miały inne proporcje niż ich wartości funkcji przystosowania.

Przykład 3.7

Wróćmy do rozważanego w przykładzie 3.2 przypadku ruletki, w której ustalamy przedziały na podstawie wyników funkcji przystosowania



Ilustracja 3.6: Rozkład ruletki

Następnie losujemy n=4 liczb, które wskażą wybrane osobniki:

60, 77, 36, 53

W wyniku takiego nieszczęśliwego losowania otrzymaliśmy populację, w której drugi najlepiej przystosowany przodek nie ma w ogóle reprezentantów, natomiast słabo przystosowany osobnik doczekał się jednego potomka.

Metoda ruletki jest bardzo prosta w zastosowaniu, ale z powodu całkowitej niezależności wszystkich losowań może zbyt często prowadzić do reprodukcji w proporcjach niezgodnych z oczekiwaniami, co pokazał powyższy przykład, oraz w skrajnych przypadkach do utraty cennych osobników.

Osobnik słabo przystosowany może być cenny ze względu na posiadanie pewnego allelu na pewnym locus, który nie występuje więcej w populacji. Usunięcie takiego osobnika spowoduje, że krzyżowanie nie będzie mogło doprowadzić do stworzenia innego osobnika z takim allelem w tym locus.

Z tego powodu w badaniach nad algorytmami genetycznymi opracowano wiele rożnych planów reprodukcji.

De Jong zaproponował między innymi następujące warianty zastąpienia planu reprodukcji R1 (opartego na ruletce):

R2. Model elitarystyczny polega na zadbaniu o to, by najlepiej przystosowany osobnik zawsze miał przynajmniej jednego potomka. Oznacza to, że po wykonaniu planu R1 jeśli najlepiej przystosowany osobnik z poprzedniego pokolenia nie występuje w pokoleniu potomnym, to jest dołączany do populacji. To podejście zabezpiecza przed pechowymi losowaniami, w których „zgubimy” najlepsze dotychczas rozwiązanie. Niestety dla funkcji przystosowania o wielu maksimach lokalnych taki plan

reprodukcji obniża efektywność algorytmu.

R3. Model wartości oczekiwanej polega na obliczeniu kredytu, czyli oczekiwanej liczby kopii każdego osobnika i zmniejszaniu tej wartości za każdym razem kiedy ten osobnik zostanie wylosowany. Osobnik, którego kredyt spadł poniżej zera nie jest już brany pod uwagę podczas reprodukcji. To rozwiązanie eliminuje problem gubienia nadmiernej liczby alleli.

R5. Model ze ściskiem („crowding model") został wprowadzony jako analogia do „zatłoczonej niszy ekologicznej" i polega na wprowadzaniu czynnika ścisku CF. Podczas reprodukcji, gdy powstaje nowy osobnik, zostaje on wprowadzony do populacji poprzez zastąpienie najbardziej podobnego do niego osobnika z grupy liczącej CF losowo wybranych z populacji osobników.

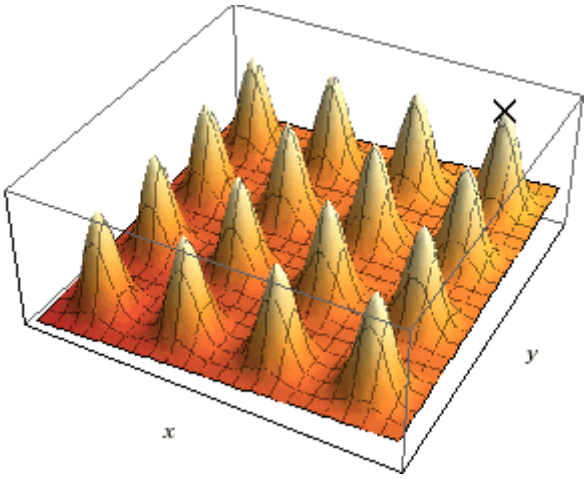
Model R5 zwiększa efektywność algorytmu genetycznego w problemach o wielu lokalnych maksimach funkcji przystosowania.

Przykład 3.8

Problem poszukiwania maksimum funkcji f o następującej postaci:

$$f(x,y)=3\sin^4(x)\sin^4(y)+\frac{(x+y)}{20}$$

na przedziale $(-2\pi,2\pi)x(-2\pi,2\pi)$



Ilustracja 3.7: Wykres funkcji f. Najwyższy wierzchołek oznaczono symbolem X

Problem ten będzie „pułapką" dla algorytmu genetycznego jeśli nie zastosuje się planu reprodukcji podobnego do R5. Algorytm genetyczny (podobnie jak większość metod przeszukiwania) w tym przypadku trafiwszy na pierwsze wysokie wierzchołki może mieć trudności z wyjściem z nich, by dotrzeć do najwyżej położonego punktu. Wprowadzenie współczynnika ścisku uniemożliwi szybkie skupienie całej populacji w niewielu wierzchołkach (lub nawet w tylko jednym), przez co algorytm dużo szybciej dotrze do najwyższego punktu.

3.3 Ogólne zastosowania algorytmów genetycznych

3.3.1 Symulowanie procesów biologicznych

Algorytmy genetyczne powstały w wyniku eksperymentów dotyczących modelowania ewolucji w systemie komputerowym.

Rosenberg w swojej rozprawie doktorskiej „Simulation of genetic populations with biochemical properties" opublikowanej w 1967 zajął się symulacją populacji organizmów jednokomórkowych. Osobniki w populacji nie były skomplikowane, ponieważ Rosenberg główny nacisk położył na dobre określenie biochemii osobników i opisał model kinetyki biochemicznej. Dla przyszłych badań nad algorytmami genetycznymi wartość przedstawiał jednak model genetyczny użyty przez Rosenberga[6].

Genotyp składał się z dwóch chromosomów, których długość ograniczona była do 20 genów o nie więcej niż 16 allelach. Każdy gen zawierał dodatkowo współczynnik sprzężenia, który decydował o prawdopodobieństwie wyznaczenia punktu krzyżowania w sąsiedztwie tego genu.

Funkcja przystosowania w eksperymencie Rosenberga dotyczyła stężeń składników chemicznych i zwracała wartość opisującą ich odchylenie od pożądanych poziomów. Zadaniem symulacji było ewoluowanie osobników do postaci o najlepszej równowadze biochemicznej, czyli o stężeniach składników chemicznych najbardziej zbliżonych do pożądanych.

W szerszej perspektywie algorytm genetyczny Rosenberga poszukiwał minimum funkcji przystosowania, co było działaniem równoważnym z rozwiązaniem skomplikowanego nieliniowego układu równań.

Krzyżowanie w algorytmie Rosenberga nie odbywało się w całkowicie losowy sposób, ale zgodnie z rozkładem prawdopodobieństwa określonym przez współczynniki sprzężenia w genotypie.

Weinberg w rozprawie o podobnym charakterze (1970) „Computer simulation of a Living Cell" wprowadził wielowarstwowy algorytm genetyczny. Parametry algorytmu ewoluującej populacji organizmów jednokomórkowych były ustalane przez inny algorytm genetyczny. Ewolucja populacji kontrolowana była przez 15 stałych dobieranych arbitralnie z liczb całkowitych z zakresu $10^{-6} - 10^{6}$. Weinberg zakodował je w postaci jednego ciągu i ewoluował w celu maksymalizacji tempa poprawy wyników symulacji.

3.3.2 Rozpoznawanie postaci metodą detektorów

Cavicchio w pracy „Adaptive Search Using Simulated Evolution" z 1970 roku zastosował metodę rozpoznawania postaci używającą detektorów[7].

Dla monochromatycznego cyfrowego obrazu wejściowego będącego wektorem pikseli detektor jest tożsamy z pewnym podzbiorem pikseli. Sumaryczny stan pikseli należących do detektora jest stanem detektora. Odpowiednio zdefiniowany zbiór detektorów jest urządzeniem rozpoznającym pozwalającym na przypisanie obrazu do pewnej klasy na podstawie stanów detektorów.

6 Informacje o pracy Rosenberga zaczerpnięte z [Goldberg 1995]
7 Informacje o pracy Cavicchio zaczerpnięte z [Goldberg 1995]

Metoda polegała na trenowaniu urządzenia rozpoznającego na zaklasyfikowanych obrazach i kojarzeniu stanów detektorów z klasami obrazów ze zbioru trenującego. Następnie dla nowego obrazu wybierano klasę na podstawie podobieństwa stanów detektorów do tych zarejestrowanych podczas trenowania.

Metoda ta jest skuteczna o ile wybierze się odpowiednie zbiory pikseli jako detektory. Cavicchio użył do tego celu algorytmu genetycznego.

Genotyp opisywał średnio 110 detektorów zawierających od 2 do 6 pikseli. Został zakodowany jako ciąg złożonych z podciągów liczb dodatnich i ujemnych postaci: *a a a, -b -b, c c c c c, -d -d -d ...* Do jednego detektora należały piksele o numerach wskazywanych przez wartości bezwzględne liczb w podciągu dodatnim lub ujemnym. Zmiana znaku w ciągu oznaczała początek definicji następnego detektora.

Funkcja przystosowania mierzyła skuteczność urządzenia rozpoznającego na stałym zbiorze przypadków testowych.

Krzyżowanie przebiegało tak jak w klasycznym algorytmie genetycznym, ale z zastrzeżeniem, że punkt krzyżowania może znajdować się wyłącznie na styku definicji detektorów. Cavicchio dopuszczał również krzyżowanie dwupunktowe.

Mutacja występowała w trzech wariantach:

- zmiana jednego piksela w detektorze
- zmiana wszystkich pikseli w detektorze
- wymiana pikseli w sąsiadujących detektorach

3.3.3 Projekt *GOLEM*[8]

Projekt GOLEM jest próbą rozszerzenia zastosowań technik ewolucyjnych o obszar świata materialnego poprzez ewoluowanie form elektromechanicznych (robotów), które mogą być zbudowane w sposób zautomatyzowany[9]

Projekt GOLEM był pierwszym zastosowaniem algorytmów genetycznych w robotyce nie do ewolucji cyfrowych symulacji robotów lub sterowania w realnych robotach, ale do ewolucji robotów jako całości. Roboty budowane były z podstawowych, prostych „klocków". Autorzy zadbali o pełną automatyzację procesu transferu projektów robotów do materialnego świata.

Fenotyp

Osobnik – robot – składał się z prostych belek połączonych stawami oraz zawierał neurony mogące łączyć się ze sobą jak i z belkami jako efektorami („mięśniami"), sterując ich długością. Tak skonstruowany robot miał możliwość poruszania się.

Genotyp

Genotyp składał się z opisu linii i łączących je węzłów oraz neuronów sieci i ich połączeń.

8 Na podstawie [Pollack Lipson 2000]
9 Cytat za [Pollack Lipson 2000], tłumaczenie własne.

Funkcja przystosowania

W procesie ewolucji do obliczenia funkcji przystosowania używano środowiska do symulacji ruchu robota. Wynikiem funkcji przystosowania był dystans przebyty przez robota w założonym czasie.

Rezultat

W wyniku przeprowadzenia ewolucji powstały bardzo proste roboty, które wyprodukowano w sposób automatyczny. Tak utworzone roboty rzeczywiście potrafiły się przemieszczać.

Przykładowe roboty wraz ze zdjęciami i nagraniami wideo ich materialnych wersji znajdują się pod adresem:

http://www.demo.cs.brandeis.edu/golem/results.html

3.3.4 Inżynieria i mechanika

Algorytmy genetyczne znalazły wiele zastosowań w optymalizacji konstrukcji.

Goldberg[10] podaje przykłady użycia algorytmu genetycznego do optymalizacji przepływu w gazociągu oraz optymalizacji strukturalnej konstrukcji kratownicy.

Znane są również zastosowania w:

- optymalizacji krzywizn karoserii samochodów wyścigowych
- maksymalizowaniu wytrzymałości elementów konstrukcji
- projektowaniu nowych układów elektronicznych i optymalizowaniu istniejących rozwiązań

3.3.5 Ekonomia i politologia

W ekonomii i politologii zastosowania algorytmów genetycznych opierają się w większości na rozwiązywaniu problemów konfliktu i kooperacji, czyli stanowią przypadki zbliżone do *iterowanego dylematu więźnia*.

Iterowany dylemat więźnia jest zagadnieniem z dziedziny teorii gier polegającym na podejmowaniu decyzji przez dwóch uczestników. Obaj wybierają pomiędzy dwiema opcjami: „współpracą” i „zdradą”. Od ich wspólnego wyboru zależy zysk lub kara każdego z nich. Poniższa tabela przedstawia zależność zysku (liczby całkowite) od decyzji graczy.

		Gracz 2	
		Współpraca	Zdrada
Gracz 1	Współpraca	R R	S T
	Zdrada	T S	P P

Tabela 3.12: Zysk graczy w zależności od zachowań obu z nich

10 [Goldberg 1995]

T określa nagrodę za zdradę, *R* – nagrodę za współpracę, *P* – karę za zdradę gdy obaj zdradzili, a *S* – wypłatę oszukanego.

Liczby te pozostają w zależności: $T > R > P > S$ oraz $2R > T + S$

Warunki te muszą być spełnione, żeby współpraca obu graczy była bardziej opłacalna niż naprzemienne zdradzanie się. Gracz, który w *n* iteracjach zdobędzie większą sumaryczną nagrodę zostaje zwycięzcą.

Axelrod użył algorytmu genetycznego do wyznaczenia nowych strategii wygrywania w *iterowanym dylemacie więźnia*[11]. Przyjął, że reguły decyzyjne mogą opierać się na 3 ostatnich posunięciach obu stron.

Genotyp był ciągiem symboli oznaczających różne zachowania przypisanych wszystkim możliwym kombinacjom trzech ostatnich ruchów dwóch graczy. Genotyp był zatem zestawem reguł postępowania dla wszystkich możliwych sytuacji.

Funkcja przystosowania obliczana była jako średnia ważona wyników pojedynków z 8 przeciwnikami (programami) reprezentującymi 98% znanych przed eksperymentem zachowań obserwowanych w rozgrywkach komputerowych. Pojedynki złożone były z 151 iteracji.

W poprzedzających eksperyment Axelroda potyczkach programów grających w *iterowany dylemat więźnia* ogólnym zwycięzcą okazała się prosta strategia nazywana „wet-za-wet" polegająca na powtarzaniu ostatniego zachowania przeciwnika.

Algorytm genetyczny zastosowany przez Axelroda odkrył nowe strategie, które w klasyfikacji ogólnej wygrywały ze strategią „wet-za-wet".

3.4 Znane zastosowania algorytmów genetycznych w lingwistyce

3.4.1 Generowanie utworów w języku naturalnym

Algorytmy genetyczne znalazły zastosowanie w dziedzinie symulowania kreatywności przez sztuczną inteligencję. Jednym ze znanych zastosowań w tej dziedzinie jest system generujący poezję o nazwie *McGonagall* opisany w pracy „An evolutionary algorithm approach to poetry generation"[12].

System *McGonagall* generuje za pomocą algorytmu genetycznego rymowane wiersze na podstawie utworzonego wcześniej semantycznego sensu. Przykładowe wejście do etapu przetwarzania ewolucyjnego może być postaci:

John walked. John slept.

Ewolucyjne przetwarzanie ma na celu zwrócenie tekstu postaci np.:

Into the bookshop John did slowly creep,

inside he fell into a peaceful sleep.

11 Eksperyment opisany w [Goldberg 1995]
12 [Manurung 2003]

Przeszukiwana przez algorytm genetyczny przestrzeń jest przestrzenią możliwych stanów tekstu wytworzonych z tekstu źródłowego poprzez realizację operacji takich jak:

- zmiany semantyczne dodające nową treść,
- podstawienie wyrazu synonimem,
- zmiana struktury gramatycznej zdań, np. inwersja,
- zastąpienie podmiotu podmiotem domyślnym.

Przeszukiwanie tak złożonej przestrzeni wymaga zastosowania wielu oddzielnych ewaluatorów składających się na funkcję przystosowania. Funkcja przystosowania jest zatem liniową kombinacją wyników poszczególnych ewaluatorów. Ustalenie wag dla tej kombinacji liniowej samo w sobie okazuje się być rozległym problemem.

Ostatecznym wynikiem możliwym do przyjęcia jako rozwiązanie jest osobnik, który spełnia trzy kryteria:

- „meaningfulness" - posiadanie znaczenia,
- „grammaticality" - poprawność gramatyczna,
- „poeticness" - „poetyckość", nietrywialna forma.

System *McGonagall* z powodzeniem generuje krótkie formy poetyckie, takie jak limeryk, czy haiku.

3.4.2 POS-tagging z wykorzystaniem algorytmów genetycznych

Tagowanie części mowy z wykorzystaniem algorytmu genetycznego zostało opisane w artykułach L.Araujo z 2002 roku oraz tegoż we współpracy z E. Alba i G.Luque z 2006 roku.

Pierwszy z artykułów[13] opisuje sposób zastosowania algorytmu genetycznego do zadania tagowania części mowy. Tagowanie algorytmem genetycznym zostało przez autora zaliczone do metod stochastycznych. Opiera się na wcześniej otagowanym korpusie języka angielskiego (*Brown Corpus*) i użytym w nim zestawie tagów. Podobnie jak inne stochastyczne metody posługuje się prawdopodobieństwami kontekstów w jakich występują wyrazy i tagi.

Genotyp w tym zadaniu jest zdefiniowany jako ciąg o długości równej liczbie wyrazów w zdaniu. Elementami ciągu są tagi wraz z dodatkową informacją dotyczącą kontekstu.

Populacja początkowa składa się z osobników wygenerowanych przez losowe wybranie dla każdego z wyrazów zdania jednego tagu spośród możliwych tagów danego wyrazu. (Umieszczenie w genotypie osobnika tagu, który nie jest możliwym tagiem danego wyrazu oznaczałoby wygenerowanie rozwiązania niedopuszczalnego.)

Funkcja przystosowania jest sumą ewaluacji tagów wszystkich wyrazów w zdaniu. Pojedynczy wyraz podlega ewaluacji przez sprawdzanie częstotliwości występowania kontekstów tagu począwszy od najszerszego kontekstu, a

13 [Araujo 2002]

skończywszy na sprawdzeniu prawdopodobieństwa samego tagu jako zerowego kontekstu. Informacje o średniej wartości funkcji przystosowania dla całej populacji są zachowywane i służą później do kontrolowania postępów algorytmu.

Krzyżowanie ma dwie wersje. Jako pierwsza stosowana jest wersja identyczna z klasycznym krzyżowaniem opisanym w sekcji 4.1.1.1. Gdy średnia wartość funkcji przystosowania populacji osiągnie wartość dwukrotnie większą niż początkowa, stosowana jest druga, bardziej zachowawcza, wersja operatora krzyżowania. Wymianie między osobnikami podlega tylko jeden allel na losowo wskazanej pozycji. W kontekście klasycznego krzyżowania można powiedzieć, że wymianie podlega podciąg długości jeden zaczynający się w miejscu krzyżowania.

Mutacja wykonuje z zadanym niskim prawdopodobieństwem na każdym genie podmianę wartości na inny poprawny dla odpowiadającego wyrazu tag.

W wyniku zastosowania algorytmu genetycznego ze stosunkowo małą populacją (16 osobników) z ograniczeniem do 100 iteracji algorytmu udało się osiągnąć poprawność tagowania rzędu 96%. Korpus trenujący składał się z 350000 wyrazów. Wskaźnik poprawności jest porównywalny z wcześniejszymi podejściami do tagowania, np. metodami regułowymi lub statystycznymi. Dzięki zastosowaniu algorytmu genetycznego i niewielkiej populacji rezultat jest osiągany w bardzo krótkim czasie.

Artykuł zawiera również podsumowanie najczęściej występujących błędów w tagowaniu tą metodą.

- Zgodnie z przewidywaniami, wyrazy bywają źle oznaczane, gdy powinny być oznaczone tagiem, który nie jest najbardziej prawdopodobny dla danego wyrazu w ogólności.
- Jeśli tag występuje w nietypowym kontekście, to prawdopodobnie nie zostanie rozpoznany poprawnie.
- Im dłuższe jest zdanie tym lepsze rezultaty, bo tagger dysponuje większym kontekstem pozwalającym poprawnie rozpoznać tag, który nie jest najbardziej prawdopodobnym dla danego wyrazu w ogólności.
- Dla przypadków zdań, w których wyraz występuje w rzadko spotykanej dla niego roli, wielkość korpusu trenującego wpływa negatywnie na poprawność tagowania.
- Zauważono, że analiza skomplikowanego tekstu wymaga większej populacji dla szybkiego osiągnięcia prawidłowego wyniku.

Drugi z artykułów[14] opisuje szczegółowo badania porównawcze taggerów używających różnych wariantów algorytmów genetycznych i technik stochastycznych oraz wpływu przetwarzania równoległego na skuteczność tych taggerów.

14 [Alba Luque 2006]

3.4.3 Ważenie ograniczeń w gramatyce WCDG[15]

Gramatyka WCDG (Weighted Constraint Dependency Grammar) jest formalizmem reprezentującym struktury gramatyczne jako przypisanie relacji *zależności* (*dependency*) między tokenami składającymi się na zdanie. Zbiór możliwych relacji *zależności* dla zdania podlega *ograniczeniom* (*constraint*), które pozwalają odrzucić niepoprawne struktury reprezentacji zdania. W gramatyce WCDG zdefiniowanych jest wiele takich *ograniczeń* i są one traktowane jako niezależne. Istnieje możliwość, że dla pewnego zdania wejściowego ograniczenia mogą okazać się sprzeczne. Gramatyka dopuszcza łamanie wybranych *ograniczeń* (*soft constraints*), a kara za ich łamanie jest zależna od *wag* (*weight*) przypisanych *ograniczeniom.*

Jak zauważają autorzy eksperymentu ważenia ograniczeń w WCDG algorytmem genetycznym, ustalanie wag, wcześniej dokonywane przez twórcę gramatyki, było zadaniem bardzo pracochłonnym i podatnym na błędy.

Genotyp w eksperymencie stanowił listę wag będących liczbami z przedziału [0,1] przypisanych ograniczeniom.

Populacja początkowa w pierwszym eksperymencie została zainicjowana gramatyką ważoną tradycyjną metodą.

Funkcja przystosowania odzwierciedlała w eksperymentach poprawność lub szybkość parsowania.

Operator krzyżowania zdefiniowano w dwóch wariantach:

- Tradycyjne krzyżowanie z wieloma punktami krzyżowania
- Krzyżowanie arytmetyczne – dla każdego genu obliczana jest średnia wartość z obu rodziców.

Operator mutacji występuje w trzech wariantach:

- Mutacja postępująca pojedynczego genu
- Wybranie losowej wartości genu
- Resetowanie – przypisanie wartości 0

W eksperymentach udało się wykazać skuteczność algorytmu genetycznego w dobieraniu wag zarówno pod kątem poprawienia skuteczności parsowania, jak i jego szybkości.

> Uwaga
>
> Wagi w gramatyce WCDG są zdefiniowane inaczej niż wagi zastosowane w gramatyce CFG lub PCFG.

15 Na podstawie [SHMF 2001]

4 Opis projektu i prowadzenia eksperymentu

Streszczenie

W tym rozdziale przedstawię wszystkie elementy składające się na środowisko potrzebne do przeprowadzenia eksperymentu oraz sposób jego przeprowadzenia. Najobszerniej opiszę złożony mechanizm realizujący funkcję przystosowania. Następnie przedstawię parametry i operatory algorytmu genetycznego użytego w eksperymencie.

4.1 Podmiot eksperymentu

System *Translatica* jest systemem tłumaczenia automatycznego, w którym zastosowana jest metoda regułowa. Oznacza to, że tłumaczenie w systemie *Translatica* bazuje na procesie tworzenia reprezentacji gramatycznej zdania źródłowego w oparciu o reguły pewnej gramatyki formalnej i konwersji jej na odpowiadającą strukturę języka docelowego. *Translatica* dostępna jest pod adresem: http://www.translatica.pl/

System *Meteor* jest systemem automatycznej ewaluacji jakości tłumaczenia automatycznego bazującym na statystyce podobieństwa pomiędzy tłumaczeniem automatycznym a tłumaczeniem ludzkim dla tego samego zdania źródłowego. *Meteor* jest dostępny pod adresem http://www.cs.cmu.edu/~alavie/METEOR/

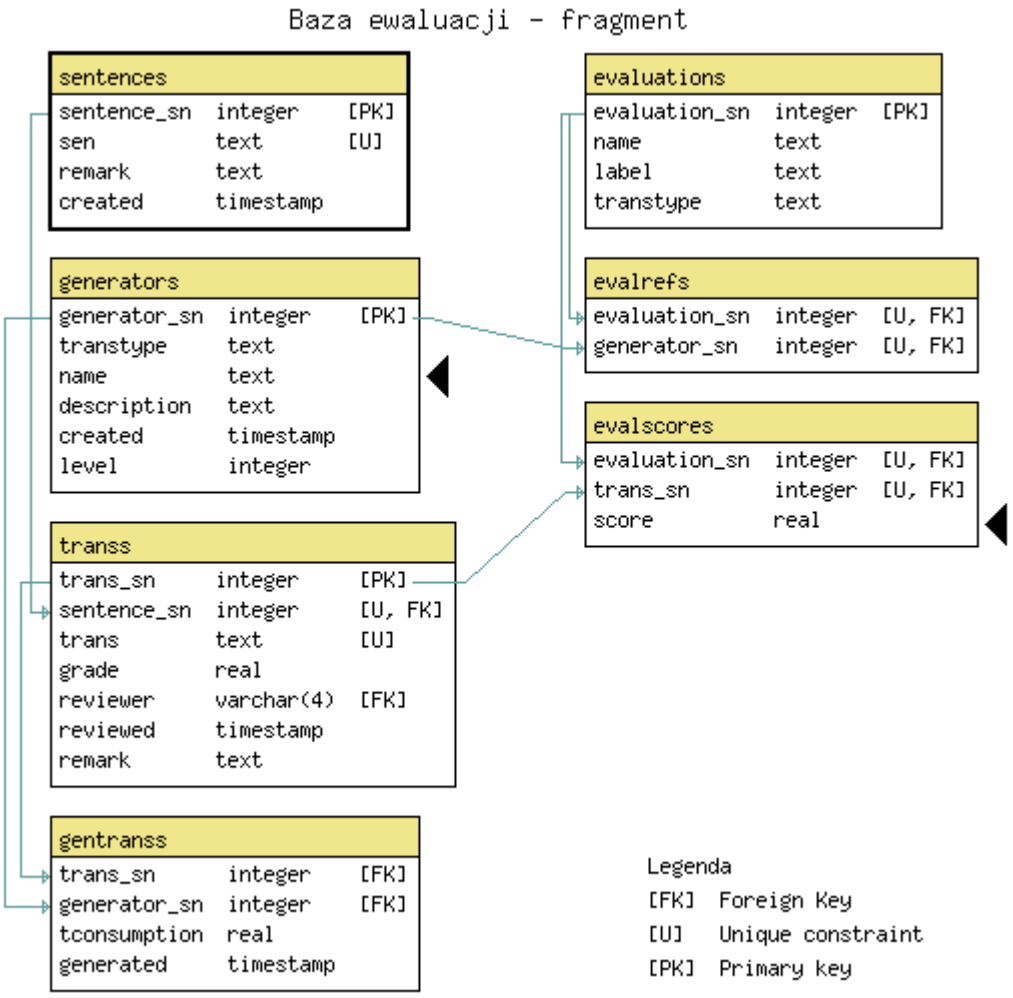
W eksperymencie ewolucji podlegają wagi reguł gramatyki języka niemieckiego używanej w systemie *Translatica*. Dotychczas dla języka niemieckiego używana była gramatyka probabilistyczna. W formalizmie używanym przez system *Translatica* w roli wag zachowano wartości logarytmu prawdopodobieństw gramatyki probabilistycznej. Oznacza to, że wagi należą do przedziału [0,-1]. Zadaniem algorytmu genetycznego w eksperymencie jest uzyskanie zestawu wag dającego wyższy wynik ewaluacji tłumaczenia niż dotychczasowy.

Ocenianie jakości wyewoluowanych wag odbywa się poprzez wykonanie tłumaczeń na zestawie liczącym 1992 zdania w języku niemieckim, a następnie ocenienie ich jakości systemem ewaluacji Meteor.

4.2 Baza poleng-evaldb - opis zastosowania w projekcie

W procesie ewolucji wag reguł funkcja oceny zwraca wynik w oparciu o ewaluację przetłumaczonych zdań ze zbioru testowego. Aby proces oceniania zdań był wydajny wymagane jest, by każde unikalne zdanie przechodziło ewaluację tylko raz. Zatem konieczne jest przechowanie informacji o zdaniu i jego ocenie w pewnej bazie danych.

W projekcie użyłem bazy ewaluacji *evaldb* wraz z towarzyszącymi jej narzędziami opracowanej jako oprogramowanie do testowania tłumaczenia automatycznego systemu *Translatica*.



Ilustracja 4.1: Diagram przedstawiający interesujące z punktu widzenia eksperymentu tabele w bazie evaldb

Baza danych *evaldb* przechowuje zbiór zdań testowych w tabeli *sentences* zaznaczonej na ilustracji pogrubieniem ramki. Tabela *sentences* jest powiązana z tabelą *transs* przechowującą tłumaczenia zdań. Tłumaczenia mogą pochodzić z rożnych źródeł. Przy dodawaniu tłumaczeń do bazy następuje przypisanie ich do generatora utworzonego w tabeli *generators*. Jedno zadanie tłumaczenia wykonane przez *Translatikę* odpowiada pojedynczemu generatorowi. Pole *name* generatora (zaznaczone strzałką) zawiera nazwę podaną przy uruchamianiu tłumaczenia. Funkcja przystosowania uruchamia tłumaczenie z ustaloną nazwą generatora.

W tabeli *evalscores* znajdują się wyniki ewaluacji dla pojedynczego tłumaczenia. Pole *score* zawiera ocenę zdania według systemu *Meteor*. Średnia ze wszystkich wartości pola *score* dla tłumaczeń pochodzących z jednego generatora jest oceną danego generatora. Na podstawie tej oceny funkcja przystosowania zwraca wynik.

4.2.1 Narzędzia do obsługi bazy ewaluacji

4.2.1.1 Polecenie tx

Polecenie *tx* uruchamia tłumaczenie zgromadzonych w tabeli *sentences* zdań z pomocą wskazanej instancji systemu *Translatica* i wyniki tłumaczenia umieszcza w bazie dbając o nie występowanie duplikatów. Jeśli tłumaczenie zdania istniało już wcześniej, zostaje tylko utworzona asocjacja do nowego generatora. Wybrane parametry wywołania metody *tx* zostały opisane w tabeli 4.1.

Parametr	Znaczenie
`--evaldb <nazwa bazy>`	wskazuje bazę, z którą będzie pracować *tx*
`--new-generator`	tworzy nowy generator
`--name <nazwa generatora>`	przyjmuje nazwę generatora

`--soap`
`--soap-host <IP>`
`--soap-port <numer portu>`
	konfiguruje połączenia z *Translaticą*

Tabela 4.1: Wybrane parametry wywołania polecenia tx

4.2.1.2 Polecenie meteoreval

Polecenie *meteoreval* uruchamia ewaluację tłumaczeń zdań z bazy przypisanych do wybranego generatora, które nie były wcześniej ocenione. Ewaluacja jest wykonywana przez system automatycznej oceny jakości tłumaczenia *Meteor*. Wynik ewaluacji każdego zdania trafia do bazy. Istotne parametry wywołania polecenia *meteoreval* zostały przedstawione w tabeli 4.2.

Parametr	Znaczenie
`--db <nazwa bazy>`	wskazuje bazę, z którą będzie pracować *meteoreval*
`--generators-test <nr+>`	wskazuje numer lub numery generatorów, które mają być ocenione; dopuszczalnymi wartościami są identyfikatory generators.*generator_sn* z bazy
`--evaluation-sn <nr>`	wskazuje numer metody oceny, jaką ma stosować *meteor;* dopuszczalnymi wartościami są identyfikatory *evaluations.evaluation_sn* z bazy

Tabela 4.2: Wybrane parametry wywołania polecenia meteoreval

Ponieważ *meteoreval* wymaga podania numeru generatora, trzeba wydobyć ten numer z bazy przed użyciem polecenia. W implementacji funkcji przystosowania użyłem do tego celu metody `findGenerator` z modułu `Poleng::Evaldb`

4.2.1.3 Metoda `Poleng::Evaldb ::getAvgEvaluation`

Uzyskanie wyników ewaluacji dla danego generatora wymaga skorzystania z wielu tabel dla ustalenia średniej wartości oceny wszystkich tłumaczeń powiązanych z tym generatorem i użyta metodą oceny stosowaną w ewaluacji *meteorem*. Dla uproszczenia tego procesu rozszerzyłem moduł `Poleng::Evaldb` o metodę `getAvgEvaluation`, która wykonuje odpowiednie zapytanie do bazy. Argumenty metody `getAvgEvaluation` zostały opisane w tabeli 4.3.

Argument	Znaczenie
`eid`	Id metody oceny. Dopuszczalnymi wartościami są identyfikatory *evaluations.evaluation_sn* z bazy.
`gid`	Identyfikator lub nazwa generatora, którego średnią ocenę chcemy pobrać. Jeśli podano nawę generatora to wybrany zostanie najnowszy generator o tej nazwie. Dopuszczalnymi wartościami są identyfikatory *generators.generator_sn* oraz nazwy *generators.name* z bazy.

Tabela 4.3: Argumenty metody getAvgEvaluation

4.3 Zastosowany algorytm genetyczny

4.3.1 Definicja przeszukiwanej przestrzeni

Genotyp G reprezentujący osobnika został zdefiniowany jako lista wag, w której waga reguły gramatyki w postaci liczby zmiennoprzecinkowej jest identyfikowana z regułą gramatyczną przez identyfikator reguły oraz numer instrukcji wewnątrz reguły. Identyfikator reguły obliczany jest funkcją skrótu MD5. Identyfikacja numeru instrukcji jest konieczna dla reguł zawierających rozgałęzienia warunkowe. W takim przypadku wewnątrz jednej reguły może istnieć wiele wag o różnych wartościach.

Fragment listy wag w postaci tekstowej:

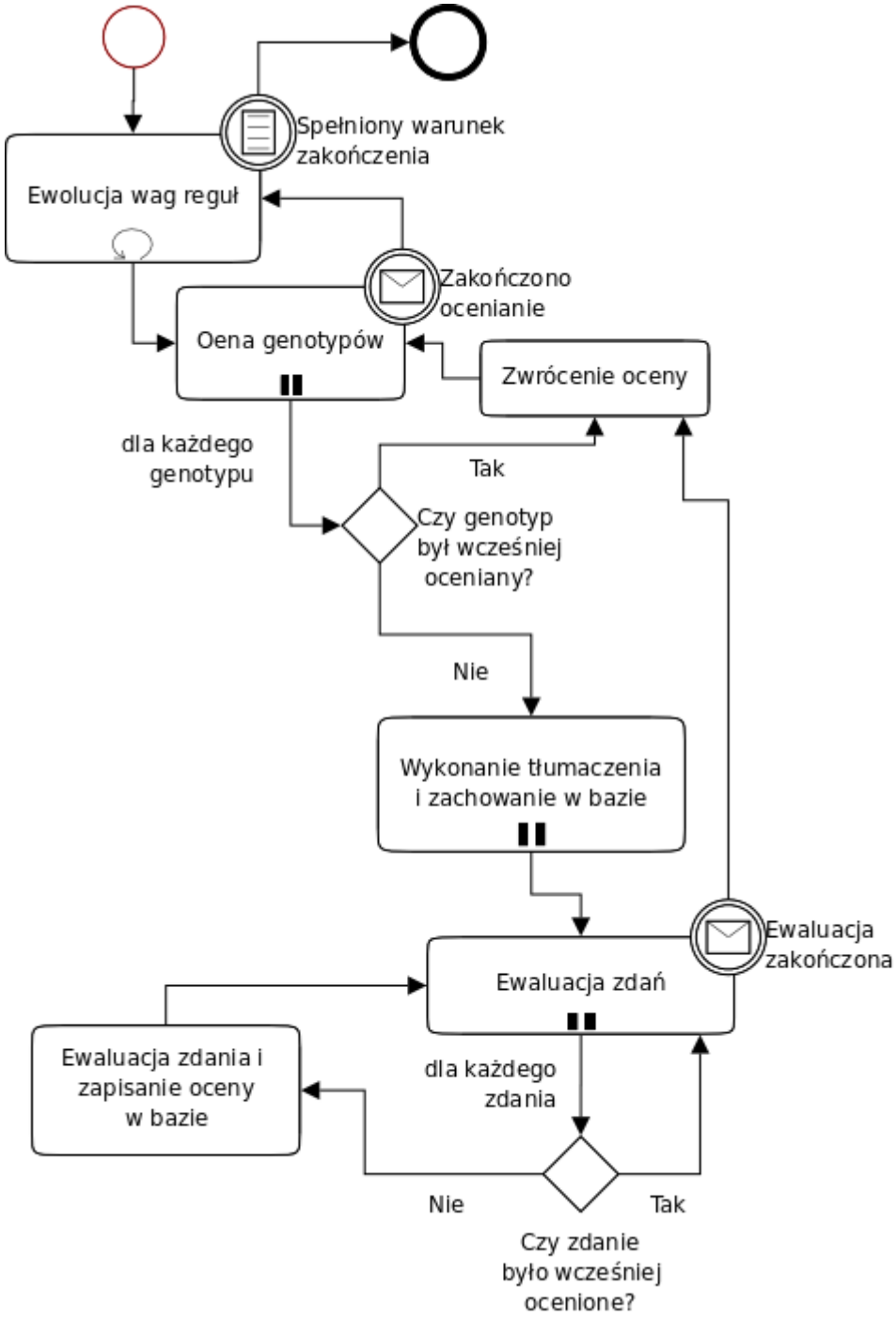
```
      identyfikator reguły      |nr|waga
0022bef90be613ef81e1db5f6f75df18|-1|0
0066b0d3220faab2f3ff993c78fe7038|6 |-2.52812
00f98e52d89b3f214c583f22e3ad75a5|5 |-8.39533
020b482fb34e0b31f607f0a83b0e24d4|2 |-5.82062
02620104c0f4e58b1f3d92e25e561ebf|4 |-5.14932
0282c46b7829ccaf363d5fd799079176|-1|0
036bffe118bae92b42ed0587e26614cf|13|-3.11028
```

Jeśli oryginalna reguła nie miała instrukcji nadającej wagę, to numer instrukcji jest równy -1 i nowa waga ustalona w czasie ewolucji jest ustawiana przez instrukcję dołączaną na końcu reguły.

Lista wag jest zawsze posortowana według identyfikatorów reguł i numerów. Wszystkie osobniki w populacji początkowej generowane są z prototypu – zrzutu aktualnie używanych wag reguł zdefiniowanych arbitralnie podczas tworzenia gramatyki.

4.3.2 Definicja funkcji przystosowania

Diagram na ilustracji 4.2 obrazuje przebieg eksperymentu z uwypukleniem metody uzyskiwania wyników funkcji przystosowania.



Ilustracja 4.2: Diagram BPMN eksperymentu

Dla genotypu *G* funkcja przystosowania zwraca wynik ewaluacji tłumaczenia automatycznego zrealizowanego w oparciu o reguły ważone wartościami z genotypu *G*.

Funkcja przystosowania jest obliczana następująco:

1. Genotyp *G* zostaje zapisany w postaci tekstowej w pliku *G.gen*
2. *Plik z tekstową postacią wag gramatyki w Translatice zostaje nadpisany przez G.gen*
3. Poleceniem *tx* przeprowadzane jest tłumaczenie zbioru zdań testowych z bazy *evaldb*.
4. Poleceniem *meteoreval* uruchamiany jest proces ewaluacji przetłumaczonych zdań. Każde zdanie jest poddawane ewaluacji tylko raz. Zdania, które już wcześniej wystąpiły mają wynik ewaluacji zapisany w bazie *evaldb*.
5. Dla zdań wygenerowanych przez tłumaczenie z genotypem *G* liczona jest średnia wartość wyników ewaluacji.
6. Średni wynik ewaluacji jest skalowany przez stałą będącą parametrem algorytmu genetycznego.

Skalowanie funkcji przystosowania jest bardzo ważne dla rozwoju populacji gdy reprodukcja jest realizowana metodą ruletki lub metodą pochodną. Wartości funkcji przystosowania są w metodzie ruletki używane jako prawdopodobieństwa reprodukcji osobników. Skuteczność algorytmu genetycznego opiera się w znacznej mierze na przeżywaniu najlepszych osobników, więc przy zbyt małych różnicach wartości funkcji przystosowania prawdopodobieństwa przeżycia osobników dobrze i słabo przystosowanych jest zbliżone.

4.3.3 Populacja i metoda reprodukcji

Populacja liczy 15 osobników. Do reprodukowania populacji użyłem metody wywodzącej się ze standardowej ruletki. W stosunku do ruletki podstawową zmianą jest zastosowanie pomysłu z modelu reprodukcji R2 u De Jonga, czyli zapewnianie co najmniej jednego potomka dla genotypu będącego aktualnie najlepszym rozwiązaniem. Ponadto wymianie ulega tylko określona część populacji. W przeprowadzonym eksperymencie 8 osobników z populacji jest zastępowana nowymi osobnikami. Nowe osobniki zajmują miejsce gorzej przystosowanych.

Obie zmiany zostały wprowadzone ze względu na wysoki koszt oceniania nowych rozwiązań. Ważne jest, żeby nie tracić w nieszczęśliwych losowaniach optymalnych osobników.

4.3.4 Definicje operatorów mutacji i krzyżowania

Krzyżowanie

W eksperymencie genotyp nie ma wielu chromosomów, ani nie ma dodatkowych warunków spójności, dlatego można zastosować klasyczny mechanizm krzyżowania jednopunktowego.

Genotypy poddawane są krzyżowaniu jak zwykłe listy atrybutów. Listy są posortowane według identyfikatorów. Dzięki temu nie występuje ryzyko wygenerowania nieprawidłowego genotypu zawierającego więcej niż jedno wystąpienie identyfikatora jednej reguły.

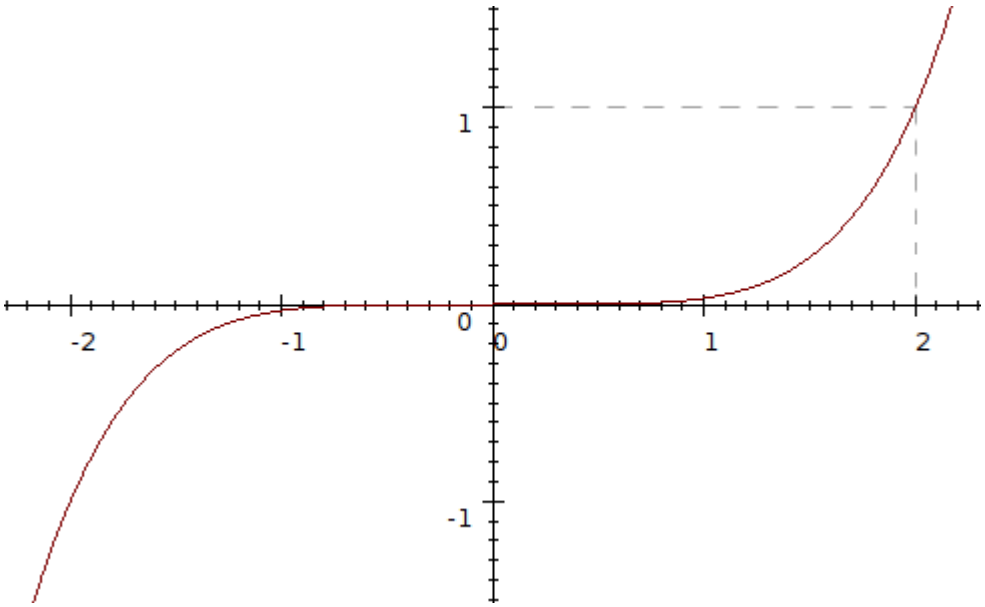
Mutacja

Zastosowany operator mutacji jest wariantem mutacji postępującej.

Mutacji podlega podzbiór całego genotypu. Każdy gen trafia do tego podzbioru z równym prawdopodobieństwem. Prawdopodobieństwo mutacji jest parametrem algorytmu genetycznego i poprzez jego zmianę można sterować średnią skalą mutacji, np. ustawiając prawdopodobieństwo mutacji na 50% uzyskamy operator mutacji, w którym średnio połowa genów będzie poddawana mutacji.

Pojedynczy gen jest mutowany poprzez dodanie do niego pewnej wartości *m* dodatniej lub ujemnej. Wartość bezwzględna liczby *m* jest ograniczona z góry przez amplitudę mutacji. Amplituda mutacji to parametr algorytmu, który definiuje najwyższą możliwą wartość skoku pojedynczej mutacji.

Aby zachować ograniczenie na amplitudę i zapewnić, żeby wartości bliskie zeru były bardziej prawdopodobne liczba *m* generowana jest przez podanie losowego argumentu z przedziału [-2,2] do funkcji $f(x)=\frac{x^5 * amplituda}{32}$



Ilustracja 4.3: funkcja f na przedziale [-2,2] *przy amplituda*=1

W ten sposób wartość mutacji z przedziału [-amplituda,amplituda] będzie mniejsza niż $amplituda * 0.03125$ z prawdopodobieństwem 0.5.

4.3.5 Analiza czasu przebiegu etapów algorytmu

Działania wykonywane przez algorytm genetyczny użyty w eksperymencie ze względu na swoją prostotę zabierały całkowicie zaniedbywalny czas. Ewaluacja systemem *Meteor* również nie wpłynęła na ogólny czas przetwarzania, ponieważ wprowadzane przez każdego osobnika zmiany wpływały jedynie na wybrany podzbiór zdań. Jak opisałem wcześniej każde unikalne zdanie było poddawane ewaluacji tylko raz – gdy pierwszy raz wystąpiło. Stąd wszystkie działania poza przeprowadzaniem tłumaczenia zajmowały czas mierzony w sekundach.

Tłumaczenie zbioru zdań trenujących dla jednego osobnika trwało średnio około 44 minut. Zatem w ciągu jednej doby algorytm przetwarzał 4 pokolenia.

5 Prezentacja wyników ewolucji

Streszczenie

W tym rozdziale przedstawię, na wybranych zdaniach wchodzących w skład zbioru trenującego, przykłady zmian wynikłych z zastosowania wyewoluowanych reguł. Omówię również zmianę wyniku automatycznej ewaluacji i jej porównanie z oceną wykonaną przez lingwistę.

5.1 Wyniki sumaryczne

W populacji złożonej z 15 osobników w każdym pokoleniu 8 osobników było zastępowane nowymi. Algorytm odbył 50 iteracji, zatem ewaluacji poddano w sumie 415 osobników.

Wynik ewaluacji osiągany przez tłumaczenie z wagami dla gramatyki probabilistycznej:

0.131525404175302

Wynik ewaluacji osiągany przez tłumaczenie z wagami osiągniętymi w eksperymencie:

0.134243944613544

Wynik osiągany przez tłumaczenie z wagami ustalonymi na 0:

0.119034461566503

5.2 Przykłady zmiany zachowania systemu Translatica dzięki nowym regułom

Przedstawię tu wybrane przykłady zdań, które zostały zmienione w wyniku zmiany wag reguł. Zdania w zestawieniu pochodzą ze zbioru 127 zdań arbitralnie wybranych spośród 1992 zdań treningowych, który został poddany ludzkiej ewaluacji. Zdania zostały wybrane tak, by zmiany w zdaniu wnosiły cokolwiek do potencjalnej oceny. Ewaluacja przebiegała w dwóch etapach:

1. Ocena poprawek na podstawie kontekstu.
2. Ocena w oparciu o zdania źródłowe wykonana przez tłumacza języka niemieckiego.

Zestawienie podzieliłem na kategorie według wyników automatycznej ewaluacji i jej poprawności.

5.2.1 Zmiana na lepsze oceniona wyżej (true positive)

Następujące zdania są przykładami zdań poprawionych i ocenionych wyżej przez system *Meteor*

1. Relację do polityki ekologicznej ~~o~~ 2004 państwo **znajdują** ~~sądzą~~ w **Internecie** ~~Internet~~ poniżej adresu: http:
2. ...jeśli w położeniu chcemy być, **wspierać naszych emerytów,** ~~nasi~~

~~emeryci podpierają,~~ przenosić nowe miejsca pracy i nasze środowisko oczyszczać.

3. Projekt-naukowiec **wzywają** ~~żądają~~ jednak nadal **do współpracy.** ~~współpracy na.~~ '...
4. **Ponieważ społeczeństwo zmienia się,** ~~Tam się przyjęcie odmienia, wchodzą~~ poza tym **oczekiwania wchodzą,** ~~oczekiwania,~~ które są stawiane pod nauczycielami.
5. Organizatorzy mają nadzieję na udział przez badaczy, **dostawców** ~~dostawcy~~ technologii, **administracyjnych członków** ~~administracyjni członkowie~~ i **politycznych decydentów** ~~polityczni decydenci~~ z odejść obszarów i państw.
6. Dalej inicjatywa **obejmuje programy stypendialne,** ~~programów stypendialnych obejmuje,~~ doktoranckie stypendia, gratyfikacje i warsztaty.
7. Mamy kilka bardzo dobre ludzie, które **zasiadają w** ~~siedzą do~~ amerykańskich **instytucjach** ~~wyposażeń~~ na strategicznie ważne miejsca.
8. **interaktywna** '~~interaktywną~~ i dobrze **udokumentowana witryna internetowa była rozstrzygającym czynnikiem',** ~~udokumentowaną witryną internetową rozstrzygający czynnik był~~
9. Rozpoznawać zainteresowane partie, że przemysł Europy ~~przodującej pozycji~~ może **zachowywać przodującą pozycję** ~~utrzymywać~~ tylko **poprzez** innowację.
10. On prorokował, że ten kierunek **utrzymywałby się** ~~zatrzymywałby~~ też **na** do przewidzenia przyszłość.
11. Cały doraźny przydział środków **wynosi** 12.250.000 **euro** ~~euro wynosi~~.

Zdania 5 i 6 są przykładem na zwiększenie zgodności atrybutów wyrazów, takich jak liczba i rodzaj. Zdania od 8 do 11 obrazują tendencję do poprawiania błędów szyku zdania polegających na zachowaniu szyku charakterystycznego dla języka źródłowego – niemieckiego.

5.2.2 Zmiana na gorsze oceniona niżej (true negative)

Następujące zdania są przykładami zdań zmienionych na gorsze i ocenionych niżej przez system *Meteor*.

1. To **sprzedaje się** ~~chodzi~~ wokół wspólnej inicjatywy **brytyjski** ~~brytyjskich~~ i **irlandzki narodowe punkty kontaktowe** ~~irlandzkich narodowych punktów kontaktowych~~ (NKS) dla Marie-Curie-Programm.
2. Terapia genetyczna znaczy albo **wymiana, manipulacja** ~~zamianę, manipulację~~ lub uzupełnienie nie **działający geny** ~~działających genów~~ poprzez zdrowe geny.
3. **Wynalezienie** ~~Rozwój~~ strategii unijnej ~~wymaga~~ na wspólną gwarancję **konkurencyjny przemysł obronny wymaga** ~~konkurencyjnego przemysłu obronnego~~ oraz **nowatorski** ~~nowatorskie~~ i technicznie **rozwinięty** ~~rozwinięte~~ rozwiązania bezpieczeństwa, który wyjaśniał dla przedsiębiorstwa i przemysł odpowiedzialny UE-komisarz Günter

Verheugen.

4. Zaliczać do dziewięciu **priorytetowym obszarom technologicznym** ~~priorytetowych obszarów technologicznych~~ strategii też biobasierte produkty przemysłowe.

5. Europejska Agencja Kosmiczna ma nadzieję, aż 2020 móc **wykładać nowe pokolenie** ~~wynaleźć nową generację~~ od rakiet nośnych.

6. Verheugen: RP7 powinna **bawić się w** ~~grać~~ istotną rolę w strategii zapewniania bezpieczeństwa UE

Zdania 2 i 4 stanowią przykłady niepoprawnego rozpoznania przypadku rzeczownika. Zmiany w wagach, które doprowadziły do poprawek prezentowanych w 5.2.1 (przykłady 8 - 11) tutaj okazały się niekorzystne. Algorytm genetyczny osiągnął w tej sytuacji kompromis polegający na poprawieniu szyku zdania kosztem wprowadzenia błędów w pewnych konstrukcjach gramatycznych.

5.2.3 Zmiana na gorsze oceniona wyżej (false positive)

Następujące zdania są przykładami zdań zmienionych na gorsze, ale ocenionych wyżej przez system *Meteor*. Występowanie takich przypadków jest szkodliwe dla skuteczności algorytmu genetycznego.

1. Przy innym projekcie interaktywne gry wideo sporządzać, **wymagali** ~~które wymagały~~ mozolnych wysiłków fizycznych.

2. Dla Mariann rybaka Boel, UE-komisarz dla rolnictwa i wiejskiego rozwoju, heit hasło na przyszłość polityk w obszar rolnictwa i **wiejskie wynalezienie** ~~wiejski rozwój~~ innowacji.

3. Poza tym on żądał **wydobywania** ~~wspieranie~~ interdyscyplinarnych **osadów** ~~podejść~~ poprzez integrację aktualnego gniazdo-program **(nowe** ~~(nowych~~ i się przerysowujący naukowy i technologiczny **rozwoje)** ~~rozwijań)~~ w priorytetowe zakresy tematyczne...

4. **'eksperyment wyznał** ~~'znany eksperyment~~ wskazuje, że wzrost **przyjął** od **gazów cieplarnianych** ~~gazom cieplarnianym~~ dużo **silniejsze skutki** ~~silniejszych skutków~~ na **klimat** ~~klimacie przypuszczano ma~~ niż dotychczas', prowadzący naukowiec ~~objaśniał~~ projektu **uznawał Davida** ~~Davidowi~~ Stainforth od uniwersytetu Oxford.

5. **Szersze informacje udzielają organizatorów konferencji:** ~~Szerszych informacji organizatorzy konferencji udzielają:~~

6. Też brak może przedstawiać na niezależność problem, w szczególności jeśli etyka-komisja jest przyłączona prosto ~~do~~ ministerstwa, na przykład **przy ministerstwo** ~~do ministerstwa~~ Zdrowia...

5.2.4 Zmiana na lepsze oceniona niżej (false negative)

Następujące zdania są przykładami zdań zmienionych na lepsze, ale ocenionych lepiej przez system *Meteor*. Występowanie takich przypadków jest szkodliwe dla skuteczności algorytmu genetycznego.

1. W USA **'człowiek stoi naprzeciwko ryzyk** ~~'kontrastujcie z człowieka ryzykami~~ całkowicie inaczej', Heinz **objaśniał Zourek,** ~~Zourek objaśniał,~~ w zastępstwie EU-Generaldirektor dla przedsiębiorstwa i przemysłu.

2. **Naukowy plan** ~~O naukowym planie~~ strategii od Enterprise państwo **znajdują** ~~sądzą~~ w **Internecie** ~~Internet~~ poniżej adresu: tutaj klikać

3. Chorwacji BIP **na** ~~pro~~ głowa wynosi przynajmniej podwójny porównywali z najwięcej **ten wcześniejszych** ~~wczesny~~ republik **do** ~~w~~ Jugosławii, i państwo **ma** relatywnie dobrze **rozwiniętą** ~~rozwinięta~~ się **bazę naukową.** ~~baza naukowa ma.~~

4. ...podwojenie wydatków stawałoby się za poszukiwania i **wynalezienie** ~~rozwój~~ (FuE) dla HIV-AIDS odkrycie szczepionki na AIDS około trzech lat przyśpieszać i ratują **miliony żyć.** ~~milionów życie.~~

5. Podczas gdy dużo platform technologicznych było **urządzane** ~~nastawiane~~ dopiero **w** najbardziej **niedawnym czasie,** ~~niedawnemu czasowi,~~ pracować trzy przez nich już od roku.

W powyższym zestawieniu występują przykłady zmian, które w innych zdaniach prowadziły do podwyższenia oceny automatycznego ewaluatora.

5.2.5 Zmiany zignorowane przez ocenę automatyczną

Następujące zdania są przykładami zdań zmienionych na gorsze, dla których nie występuje zmiana w ocenie zwracanej przez system *Meteor*.

1. **W oprawę** ~~Do ramy~~ posiedzenia **ku tematowi badaniu podstawowemu** ~~na temat badania podstawowego~~ zastępca Komisji Europejskiej będzie prezentować aktualne plany w względzie na możliwe modele finansowania.

2. Oddalanie weterynaryjnych lekarstw z **pokarmowe substancje** ~~pokarmowych substancji~~

3. Wokół długofalowego **wynalezienia** ~~rozwijania~~ Nanowissenschaften i -technologie w UE wspierać, Komisja Europejska zapewnia 2,2 Mio.

Następujące zdania są przykładami zdań zmienionych na lepsze, dla których nie występuje zmiana w ocenie zwracanej przez system *Meteor*.

1. AAAS-Jahresversammlung **jako** ~~niż~~ ogniwo między nauką i społeczeństwo

2. Wszystkie państwa członkowskie **dysponują** ~~dysponować~~ narodową etyka-komisja lub są powierzeni obecnie z tym, taki zakładać.

3. Pierwszy **posiedzenie** ~~pozowanie~~ grupy znajdowało 25. Styczeń zamiast.

4. Jak w **innych obszarach** ~~inne obszary~~ musimy **zamieszczać dobre poszukiwania** ~~sprowadzać dobrą naukę~~ i rozwój laboratorium w przemysł spożywczy i do gospodarstw rolnych ", komisarz sądził.

5. Pierwszy **przykładem** ~~podnosić przykład~~ na to konieczność byłaby, **intensyfikować** opracowywanie produktów i **podnosić** zdolności **produkcji.** ~~produkcji intensyfikują.~~ ...

6 Podsumowanie

Streszczenie

Eksperyment wykazał, że można poprawić wyniki nawet wychodząc od gramatyki probabilistycznej. Niestety automatyczna ewaluacja jest zbyt mało skuteczna, by dobrze realizować funkcję przystosowania. Rozdział wskazuje możliwy kierunek dalszych prac.

6.1 Wnioski z eksperymentu

Udało się wykazać, że modyfikując wagi reguł gramatycznych w sposób automatyczny można uzyskać poprawę szyku zdania. Możliwa jest poprawa konkretnych aspektów tłumaczenia automatycznego mimo że (co warto przypomnieć) modyfikowane są wagi reguł używanych nie do tłumaczenia, ale do parsowania zdania źródłowego. Algorytm genetyczny może zmodyfikować wagi tak, żeby osiągnąć skuteczność większą nawet niż gramatyka probabilistyczna. Zostało to wykazane pomimo wykorzystania niereprezentacyjnego zbioru testowego, ponieważ poprawa szyku zdania wystąpiła w sposób powtarzalny.

Dla gramatyk opatrzonych wagami ustalanymi arbitralnie metodą prób i błędów metoda zastosowana w eksperymencie powinna przynieść o wiele lepsze rezultaty. Powtórzenie eksperymentu dla gramatyki języka angielskiego będzie przedmiotem dalszych prac.

Jeżeli zmiana owocuje poprawieniem tłumaczenia w niektórych przypadkach, a pogorszeniem w innych to algorytm genetyczny jest zdolny do wypracowania kompromisu optymalnego dla ogólnej oceny. Trzeba jednak dobrze dobrać zbiór treningowy, by utrzymywał proporcje występowania konstrukcji gramatycznych w danym języku. Dobre modelowanie języka przez zbiór trenujący jest konieczne, żeby wynik ewaluacji wprowadzanych zmian był jak najlepiej skorelowany z faktycznym stopniem poprawy ogólnej jakości tłumaczenia.

Jednym z głównych wniosków z eksperymentu przeprowadzonego w ramach niniejszej pracy jest obserwacja, że automatyczny system ewaluacji jakości tłumaczenia jest narzędziem niedoskonałym. Ewaluacja przez porównanie z tłumaczeniem referencyjnym oparta jest na założeniu, że wszystkie poprawne tłumaczenia danego zdania są do siebie podobne – składają się w większości z tych samych wyrazów. Jest to założenie, którego nie spełniają wszystkie języki, w których występują synonimy.

Ponadto metoda ewaluacji stosowana w systemie *Meteor* pozwala rozpoznać poprawność gramatyczną zdania tylko, jeśli w tłumaczeniu referencyjnym użyto analogicznej konstrukcji gramatycznej. Jeśli, na przykład, tłumacz użył inwersji, a w poprawnym tłumaczeniu automatycznym zachował się prosty szyk zdania, to automatyczna ewaluacja uzna taki szyk zdania za nieprawidłowy.

Dalsze badania warto rozwinąć o zastosowanie metryki oceniającej poprawność gramatyczną niezależnie od zgodności ocenianego tłumaczenia z tłumaczeniem referencyjnym.

6.2 Możliwe rozszerzenie metody

Algorytm genetyczny doprowadził w eksperymencie do zauważalnej poprawy w pewnych aspektach oraz pogorszenia w niektórych innych. Ze względu na niewielki zbiór zdań trenujących (w porównaniu ze zbiorem wszystkich sensownych zdań możliwych do wyprowadzenia z używanej gramatyki) oraz niedoskonałość metod automatycznej ewaluacji algorytm genetyczny zastosowany wprost do rozwiązania tego problemu inny rezultat jest mało prawdopodobny. Wynika z tego potrzeba wyizolowania i zachowania tylko tych spośród modyfikacji wprowadzonych w wyniku eksperymentu do zestawu wag reguł, które powodują występowanie zmian na lepsze.

Eksperyment można zatem rozszerzyć o drugą fazę, w której z dotychczasowych wyników wydobyte zostaną tylko pożądane zmiany. Zamiast automatycznej ewaluacji funkcję przystosowania należy zrealizować przez następującą strategię:

1. Dla zdań, w których zauważono poprawę w pierwszej fazie należy przygotować reguły dające punkty jeśli poprawa jest nadal obecna. (najważniejsza metryka)
2. Dla zdań, w których zauważono pogorszenie w pierwszej fazie należy przygotować reguły zabierające punkty jeśli błędy są nadal obecne.
3. Osobniki należy karać proporcjonalnie do odchylenia od postaci wag używanej przed pierwszą fazą eksperymentu.
4. Dla zdań pozostałych można prowadzić ewaluację jak dotychczas, co powinno zminimalizować niechciane poboczne zmiany.

W ten sposób algorytm genetyczny znajdzie minimalny zestaw zmian w wagach reguł w stosunku do stanu sprzed eksperymentów, który wprowadza w tłumaczeniach pożądane zmiany zauważone po pierwszej fazie. Powinien jednocześnie zminimalizować liczbę przypadkowych błędów.

Populacja początkowa mogłaby składać się z osobników osiągających wysokie wyniki w instancjach pierwszej fazy przeprowadzonej z ograniczeniami możliwości modyfikowania wag tylko do pewnych obszarów.

7 Bibliografia

Jurafsky Martin 2000: D. Jurafsky, J. Martin, "Speech and Language Processing", 2000, roz. 8,9,10,12

Manning Schütze 1999: C. Manning, H. Schütze, "Foundations of Statistical Natural Language Processing", 1999, roz. 3,10

Goldberg 1995: D. Goldberg, "Algorytmy genetyczne i ich zastosowania", 1995, roz. 1 - 5

Eiben Smith 2003: A. Eiben, J. Smith, "Introduction to Evolutionary Computing", 2003, roz. 1 - 3

Pollack Lipson 2000: J. Pollack, H. Lipson, "The GOLEM Project: Evolving Hardware Bodies and Brains", 2000 [w:] "The Second NASA/DoD Workshop on Evolvable Hardware"

Manurung 2003: H. Manurung, "An evolutionary algorithm approach to poetry generation", 2003

Araujo 2002: L. Araujo, "Part-of-Speech Tagging with Evolutionary Algorithms", 2002 [w:] "Lecture Notes In Computer Science"2276 230 - 239

Alba Luque 2006: E. Alba, G. Luque, L. Araujo, "Natural Language Tagging with Parallel Genetic Algorithms", 2006 [w:] "Information Processing Letters"100, i5 173 - 182

SHMF 2001: I. Schröder, H. Pop, W. Menzel, K. Foth , "Learning Grammar Weights Using Genetic Algorithms", 2001 [w:] "Proceedings Recent Advances in Natural Language Processing" 235–239