Streszczenie

Rozdział 1

Wstęp i cele pracy

Rozdział 2

Stan wiedzy

2.1 Teoria ewolucji

Teoria zaprezentowana przez uczonego przyrodnika i geologa Charlesa Darwina w 1859 w pierwszym wydaniu jego książki, „*The Origin of Species*", podsumowującej lata pracy i zebrane doświadczenia na temat rozwoju gatunków. W ogólności teoria ewolucji głosi, że wszystkie organizmy żywe są ze sobą spokrewnione i pochodzą od jednego wspólnego przodka. Świat istot żywych podlega ciągłym i stopniowym zmianom, dążącym do adaptacji organizmów, a wszystkie zmiany te są wynikiem doboru naturalnego. [1]
Teoria Darwina opiera się na pewnym zbiorze zasad [1], [2]:

- Prawo zmienności powszechnej i bezkierunkowej
  Wyjaśnia, iż jedynie zmienność dziedziczna ma wpływ na ewolucję. Zmienność niedziedziczna, nie wpływa na jej przebieg.

- Prawo różnorodności gatunków
  Głosi, że gatunki dzielą się na podgatunki potomne, lub w procesie pączkowania wytwarzają innego rodzaju organizmy potomne.

- Prawo walki o byt
  Jest to mechanizm redukujący nadmiar populacji, będący czynnikiem napędzającym proces ewolucji. Walka o byt może się odbywać między różnymi gatunkami w układzie ofiara - drapieżnik lub w obrębie jednego gatunku w wyniku konkurencji o tę samą niszę ekologiczną.

- Prawo doboru naturalnego
  Przeżywają jedynie osobniki najlepiej przystosowane, a formy pośrednie wymierają, co prowadzi do coraz większej rozbieżności cech w następnych pokoleniach i powstania z czasem form bardzo różniących się od praprzodka i powstawania nowych gatunków.

- Prawo dziedziczenia
  Bezpośrednio łączy się z powyższym prawem. Osobniki słabsze, częściej padające ofiarą, mają mniejsze szanse na rozmnażanie, a co za tym idzie na przekazanie swojego zestawu cech. Powstają organizmy potomne, dziedziczące jedynie cechy od silnych osobników, które przetrwały.

Wszystkie tezy zostały potwierdzone badaniami z dziedziny biologii molekularnej, ekologii oraz biogeografii.

2.2 Ewolucja Organiczna

Ewolucja to proces stopniowej przemiany osobników (zarówno zwierząt jak i roślin), który w ostateczności może doprowadzić do powstania nowych gatunków. Przemiana ta może dotyczyć zarówno cech morfologicznych jak i fizjologicznych. Jej istotą jest zmiana składu materiału genetycznego organizmów potomnym w stosunku do organizmów rodzicielskich. [3] Zmiany te mogą być wynikiem różnych mechanizmów:

2.2.1 Mutacje i zmienność rekombinacyjna

Występowanie tych dwóch mechanizmów ma charakter losowy. Zmienność rekombinacyjna jest wynikiem mieszania się materiału genetycznego, natomiast mutacje spowodowane są zmianami w obrębie jednego organizmu. Mutacje mogą być typu punktowego (dotyczące jednego nukleotydu) lub obejmować większy odcinek DNA (chromosomowe). Wśród nich wyróżniamy:

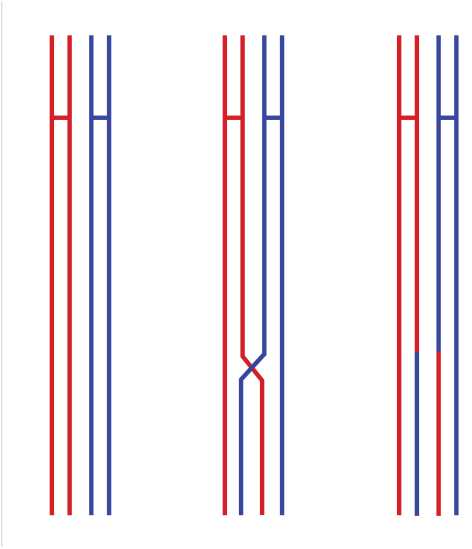
- substytucję,
- delecję,
- insercję,
- tranzycję,
- transwersję
- inwersję,

- deficjencję,
- translokację. [4]

Zmienność rekombinacyjna zachodzi dzięki zjawiskom takim jak: crossing-over, niezależna segregacja chromosomów i połączenie gamet.

Crossing-over

Crossing-over, inaczej krzyżowanie, jest zjawiskiem wymiany materiału genetycznego między chromatydami niesiostrzanymi chromosomów homologicznych podczas procesu mejozy. [5] Schemat zachodzenia crossing-over zaprezentowano na grafice rys. 2.1. Chromosomami homologicznymi nazywamy parę chromosomów pochodzących, po jednym, od osobników rodzicielskich.



Rys. 2.1 Profaza mejozy - crossing-over

2.2.2 Dobór naturalny

To czynnik nadający ewolucji kierunkowy i przystosowawczy charakter. Ma na celu zwiększenie stopnia przystosowania (adaptacji) do warunków środowiskowych zarówno na poziomie osobniczym jak i genowym. Organizmy posiadające korzystne cechy mają większą szansę na przeżycie i rozmnażanie, co prowadzi do zwiększania częstości występowania korzystnych genów w populacji.

2.2.3 Dryft genetyczny (zjawisko Wrighta)

Dryftem genetycznym nazywa się wahania częstotliwości występowania genu nie wynikające z działania doboru naturalnego, migracji, czy mutacji. Jest efektem losowych zmian w ilości alleli w kolejnych pokoleniach. [6]

2.2.4 Hybrydyzacja (krzyżowanie)

Hybrydyzacja to proces polegający na krzyżowaniu się osobników, będących przedstawicielami różnych genetycznie populacji, w wyniku którego może powstać potomstwo mieszańcowe. Może to doprowadzić do powstania nowych gatunków, lub przyczynić się do zwiększenia różnorodności genetycznej populacji, bądź pojawienia się w populacji nowych korzystnych cech. [7]

2.3 Strategie ewolucyjne (ES)

Pojęcie strategii ewolucyjnych powstało w latach pięćdziesiątych XX wieku, gdy naukowcy postawili sobie za cel wykorzystanie teorii ewolucji Darwina oraz zasady doboru naturalnego na zbiorze potencjalnych wyników do ich optymalizacji. [8] W 1975 roku profesor J.H. Holland jako pierwszy opracował koncept algorytmów genetycznych, które zaprezentowano w książce „*Adaption in Natural and Artificial Systems*". Zaproponował on, by zamodelować chromosomy w postaci ciągów zer i jedynek. Tak przygotowany zbiór wejściowy z łatwością ulegać może „ewolucji" poprzez mutację, selekcję, czy też crossing-over. [8] Słownik pojęć niezbędnych do poruszania się po temacie zaprezentowano w tabeli 2.1

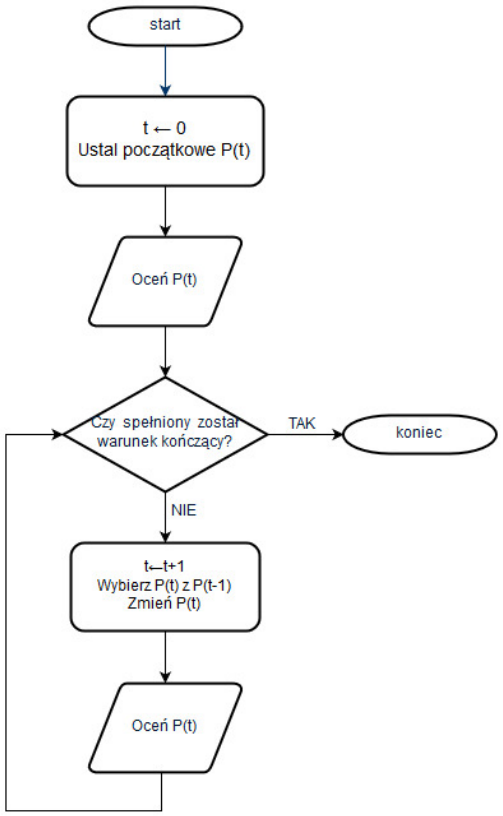
2.3.1 Algorytm ewolucyjny (EA), Algorytm Genetyczny (GA)

Algorytmem ewolucyjnym nazywamy algorytm probabilistyczny, opierający się na zasadach obowiązujących w ewolucji organicznej [9], dla którego generowany jest zbiór osobników $P(t) = \{x_1^t, ..., x_n^t\}$ w każdej iteracji t. Każdy osobnik przedstawia potencjalne rozwiązanie zadanego problemu i posiada swoją reprezentację jako struktura danych S. Obiekty zbioru oceniane są w oparciu o ich „dopasowanie". W iteracji $t+1$ tworzy się nową populację osobników. Jest ona wynikiem selekcji najlepiej „dopasowanych" obiektów z iteracji t. Niektóre z wybranych podlegają transformacji (mutacja / crossing-over) dając nowe rozwiązania. Po zakończeniu działania algorytmu oczekuje się, iż najlepsze możliwe osobniki znajdą się w zbiorze końcowym i reprezentują rozwiązanie znajdujące

Pojęcie	Objaśnienie
Chromosom	Zakodowana forma potencjalnego rozwiązania zadanego problemu. Ciąg uporządkowanych genów.
Gen	Element składowy chromosomu.
Osobnik	Dla algorytmów genetycznych, równoważny z pojęciem chromosomu. Niekiedy jednak prezentowany jako zespół chromosomów (genotyp).
Fenotyp	Odpowiednik genotypu w przestrzeni odkodowanej.
Populacja	Zbiór osobników o określonej liczebności.
Przystosowanie	Przystosowanie osobników do zadanego problemu. Oceniane za pomocą funkcji przystosowania. Im większy stopień przystosowania, tym lepsze rozwiązanie.
Selekcja	Proces filtracji najlepiej dopasowanych osobników spośród populacji. Wybrane chromosomy trafiają do populacji rodzicielskiej, przygotowywanej do rekombinacji genów.
Krzyżowanie	Rekombinacja genów chromosomów rodzicielskich, której wynikiem jest chromosom potomny o zmienionym składzie. Patrz 2.2.4
Rodzic	Chromosom wybrany do krzyżowania.
Potomek	Wynik krzyżowania pary rodziców.
Mutacja	Proces zamiany genów w obrębie jednego chromosomu bez wpływu chromosomów rodzicielskich. Patrz 2.2.1

Tab. 2.1 Słownik pojęć podstawowych

się blisko optymalnego (rozwiązanie rozsądne).[[10]] W ten sposób unika się przeszukiwania całej przestrzeni w poszukiwaniu rozwiązania, a wybierana zostaje jedynie niewielka populacja jej przedstawicieli. A dzięki mutacjom otrzymuje się rozwiązania coraz lepsze, bliskie optimum. Ogólny schemat blokowy działania algorytmu przedstawiono na rysunku 2.2.



Rys. 2.2 Struktura programu ewolucyjnego

Podczas analizy literatury stwierdzono, iż nazwy „algorytm ewolucyjny" oraz „algorytm genetyczny" stosuje się zamiennie. W poniższym tekście przyjęto również taką koncepcję.

2.3.2 Algorytm genetyczny a program ewolucyjny

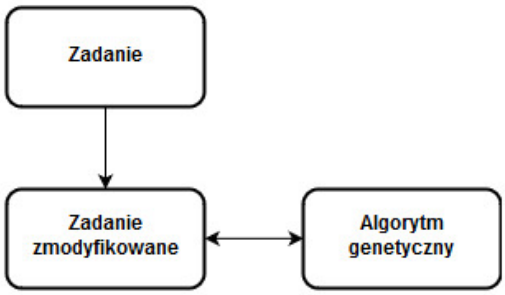
Na podstawie algorytmów ewolucyjnych powstały programy ewolucyjne. Ich struktura pozostaje taka sama, jednak różnice widać na niższym poziomie. Dla algorytmów przyjęto zapis w postaci skończonego, uporządkowanego ciągu jasno zdefiniowanych czynności, koniecznych do wykonania pewnego zadania. Konieczny do rozszyfrowania tego zapisu jest specjalny parser, który zamienia ciąg

w wykonalną funkcję oraz rozpoznaje ewentualne zmiany stanu (wywołane mutacją, bądź crossing-over), które mogłyby zagrażać jego działaniu. W porównaniu do tego program ewolucyjny jest przedstawiony jako drzewiasta struktura czynności i wartości. Również niezbędny jest parser, jednak pomniejszony o świadomość stanów (te ukryte są wewnątrz struktury).
Poza tym znaczącą różnicę stanowi reprezentacja chromosomów. Dla algorytmów ewolucyjnych/genetycznych chromosomy muszą być w formie binarnej, natomiast program pozwala nam na zdefiniowanie dowolnych struktur. Z tym związane jest również zapotrzebowanie na wprowadzenie spersonalizowanych operatorów genetycznych, odpowiednich dla zadanej struktury i zadania, podczas gdy algorytmy korzystają z podstawowych operatorów.
Algorytmy genetyczne wymagają modyfikacji zadania (przetworzenie na łańcuch binarny). Nie jest to zadaniem łatwym i niekiedy może wymagać użycia parserów, czy też algorytmów naprawy, np. reprezentacja indeksów liczby z zakresu od 1 - 5, możliwa jest dzięki 3 bitom. Jednak podczas procesu mutacji mogą powstać indeksy wykraczające poza zakres (6-8). Zmienienie ich wartości do zgodnych z zakresem wymaga użycia specjalnego algorytmu naprawy. Programy ewolucyjne, natomiast, wymagają zmiany reprezentacji chromosomowej potencjalnych rozwiązań oraz wytworzenia odpowiednich operatorów genetycznych do działania na wytworzonych strukturach. Zależności te w sposób schematyczny przedstawiono na rysunkach (rys. 2.3, rys. 2.4).

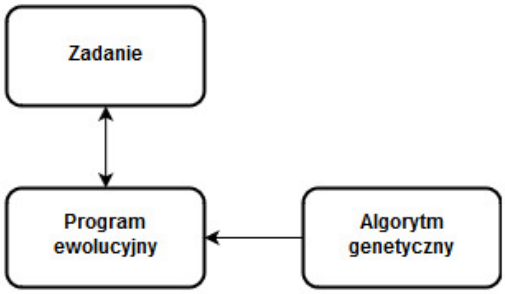
2.3.3 Wymagania

Zarówno program jak i algorytm posiadają listę wymagań, które muszą zostać spełnione, by zapewnić ich poprawne działanie. [10]
Musi istnieć:

- zbiór z reprezentacją możliwych rozwiązań problemu,
- metoda generowania początkowej populacji potencjalnych rozwiązań,
- funkcja oceniająca, do oceny „dopasowania” rozwiązań,
- operator „genetyczny” wpływający na populację,
- parametr populacji niezbędny algorytmowi (np. rozmiar populacji, prawdopodobieństwo mutacji, długość wykonywania się algorytmu itp.)



Rys. 2.3 Schemat działania algorytmu genetycznego.



Rys. 2.4 Schemat działania programu ewolucyjnego.

2.4 Ewolucyjne generowanie kodów źródłowych

Tematem pracy jest implementacja programu, którego istotą jest generowanie kodów źródłowych programów w sposób ewolucyjny, kierując się przy tym zadanym parametrem np. czasem wykonywania, czy długością kodu. Zagadnienie to w literaturze nazywane jest „*automatic programming*", czy też „*program synthesis*" i określa wszelkie metody generowania kodów bez ingerencji człowieka mające na celu wytworzenie sprawnego oprogramowania na bazie określonej specyfikacji.
Zakłada się, iż istnieje następujący podział automatycznego programowania [11]

- Programowanie generatywne
  Powszechny rodzaj programowania, w którym biblioteki standardowe są wykorzystywane do poprawy jakości i prędkości oprogramowania. Programista nie musi samodzielnie implementować, ani znać sposobów działania niektórych funkcji, a jedynie skorzystać z gotowych rozwiązań biblioteki.

- Generowanie kodów źródłowych
  Kod źródłowy jest generowany na podstawie predefiniowanego modelu, czy wzorca, powstałego dzięki wykorzystaniu narzędzi programistycznych i odpowiedniego środowiska (IDE). Przykładem działania takiego programu

jest *Google/MIT App Inventor*, gdzie generuje się kod na podstawie wzorca stworzonego przez użytkownika z bloków funkcji. Mechanizm ten nie wymaga od użytkownika napisania ani jednej linii kodu.

2.4.1 Dotychczasowe rozwiązania

Podczas analizy literatury znaleziono kilka rozwiązań podobnych problemów, do zadanego w tej pracy.

Co-evolutionary automatic programming for software development

Jednym z ciekawszych rozwiązań, które udało się znaleźć jest podejście przedstawione w artykule [12]. Zakłada ono wykorzystanie programowania genetycznego do ewolucji programów, podczas gdy testy jednostkowe, badające stopień dopasowania chromosomu wynikowego, ulegają ko-ewolucji na zasadzie konkurencji. Dopasowanie programu (chromosomu) określane jest na podstawie ilości testów, które zwróciły wynik negatywny, natomiast dopasowanie testu wyliczane zostaje z ilości programów, które go nie przeszły. Zastosowano model drapieżnik - ofiara i przyjęto, że w roli drapieżnika stoją testy jednostkowe, natomiast ofiarą są programy. Jeśli program będzie zbyt „wolny" padnie ofiarą drapieżnika i nie przejdzie testu. Takie programy nie nadają się do dalszej ewolucji. Krzyżować się mogą jedynie osobnik „najszybsze", w wyniku dając osobniki potomne, których zestaw cech powinien również gwarantować bezpieczeństwo przed drapieżnikiem. Gdyby testy (drapieżnicy) nie były objęte procesem ewolucji, działanie programu ewoluującego szybko by się skończyło. By osiągnąć lepsze wyniki postanowiono ewoluować również populację testów. Testy o dużej skuteczności (tzn. wyszukujące dużą ilość niedziałających programów) przekazują swoje cechy dalej i pomagają tworzyć nowe testy, o ulepszonym działaniu. Podczas działania algorytmu nie są porównywane wyniki otrzymanych programów z wynikiem przewidywanym (poprawnym), lecz jedynie funkcja podobieństwa obu wyników, to jest odległość między wynikami. Jeśli jej wartość jest duża, to znaczy, że wyniki znacząco się miedzy sobą różnią i przeciwnie, w przypadku gdy odległość jest bliska zeru, wyniki można uznać za identyczne. Dzięki temu nie jest konieczne przetrzymywanie informacji o tym, jaki wynik zwróci działanie programu, a jedynie wiadomość o jego względnej wartości w stosunku do wyniku poprawnego.
O ile istnieje możliwość wytworzenia formalnej specyfikacji, by wygenerować ciągłą funkcję dopasowania, to możliwe jest automatyczne wytworzenie programu. Przy czym wytworzenie poprawnie działającego programu do rozwiązywania problemu stopu [13] nie jest możliwe.

Automatic Generation of Programs using Genetic Network Programming

Praca [14], w porównaniu do powyższego podejścia, nie skupiła się na ewoluowaniu testów podczas generowania programów, lecz na przetwarzaniu problemu za pomocą struktury sieci skierowanej. Udowodniono poprawę jakości w stosunku do zwykłego programowania genetycznego.
Sieć oparta jest na dwóch rodzajach węzłów : oceniających oraz przetwarzających. Prócz tego istnieją również węzeł startu do inicjalizacji programu, pamięć współdzielona między węzłami oraz połączenia między węzłami przetwarzającymi. Program działa według następującego algorytmu:

1. Inicjalizuj losowo populację wejściową
2. Wykonaj i oceń programy
3. Ewoluuj programy
   - Crossover - Wybierz dwóch rodziców i dowolny węzeł, dokonaj zamiany.
   - Mutacja - Wybierz jednego rodzica i losowo zamień połączenia.
   - Zachowanie elity - Wybierz najlepszego osobnika i zachowaj jako potomka.
4. Zastąp rodziców potomstwem.
5. Powtarzaj kroki 2-4 aż nastąpi spełnienie warunku końcowego.

   Węzeł przetwarzający na podstawie danych zawartych w pamięci wytwarza kod, który zapisuje do pamięci. Węzły oceniające, korzystając z funkcji dopasowania, oceniają węzły przetwarzające i koordynują, do którego z nich należy przejść w kolejnym kroku.

Automatic Programming Using Genetic Programming

Inne podejście można było zaobserwować w [15]. Autorzy stworzyli własny pseudo język programowania, który miał uprościć działanie programu genetycznego, a także zapewnić uniwersalność, tzn. wygenerowany kod może zostać przetłumaczony na dowolny język programowania. Liczba funkcji została ograniczona i zdefiniowana (tabla 2.2). Samo działanie programu zachodzi natomiast w następujących krokach:

1. Generowanie populacji początkowej
   Każdy z osobników przedstawiony jest jako drzewo wyprowadzenia [16].

Typ operacji	Funkcja	Ilość przyjmowanych parametrów
Arytmetyczna	$+, -, *, /$	2
String	strequal, strnotequal	2
Logiczna	==,!=,<=,>=,<,>	2
Działania na pamięci	Write, Change, Read	3,2,1
Warunkowa	If, switchc,switchi	3, $3 \leqslant n \leqslant 8$
Iteracja	For, While	3, 2
Wyrażenia złożone	block, n=2,3 , combn	2 lub 3 ,$1 \leqslant n \leqslant 5$

Tab. 2.2 Pseudo-język stworzony w artykule [15]

.

2. Selekcja i ewaluacja
   Wybierana jest para rodziców w procederze turnieju, tzn. z losowego zbioru kandydatów wybierane zostają osobniki o najlepszym dopasowaniu. Tak jak w artykule 2.4.1 dopasowanie jest miarą odległości otrzymanego wyniku, do wyniku przewidywanego. Dla typu string sprawdzana jest ilość pokrywających się znaków (*„ hits ratio”*).

3. Regeneracja
   Dzięki operatorom genetycznym tworzeni są potomkowie wybranych rodziców. Mutacje polegają na wygenerowaniu nowego poddrzewa w wylosowanym węźle. Dla crossing-over zachodzi zamiana poddrzew miedzy dwoma rodzicami w wybranym węźle.

4. Modularyzacja
   Modularyzacja jest koncepcją mającą za zadanie podział drzewa na takie bloki, by wymiana informacji miedzy nimi była znacznie rzadsza niż wewnątrz. Poza tym zauważa się kompresję drzewa, przy jednoczesnym zachowaniu struktury. Poddrzewa są łączone, a w ich miejsca są dodawane etykiety. Dobre bloki funkcji (tzn. dające dobre wyniki) dodawane są do zestawu funkcji, co powoduje stopniowe uczenie się programu genetycznego. Głównym celem jest wiec rozłożenie zadanego problemu na prostsze - mniejsze, łatwiejsze do rozwiązania.

Autorzy artykułu przetestowali program dla 10 problemów i otrzymali zadowalające wyniki, zarówno jeśli chodzi o poprawność wyników, jak i czas pracy

programów. W dwóch problemach wyjątkowo skuteczna i niezbędna okazała się modularyzacja.

2.5 Podsumowanie

Rozdział 3

Analiza projektowa

3.1 Wybrane technologie

3.1.1 C#

3.1.2 GeneticSharp

3.1.3 QT

3.2 Architektura systemu

3.3 Wymagania

3.4 Ograniczenia

3.5 Przypadki użycia

3.6 Propozycja rozwiązania

3.6.1 Prototyp

Organizacja danych - struktury

Algorytm ewolucyjny

3.6.2 Projekt testów

3.7 Podsumowanie

Rozdział 4

Implementacja

4.1 Funkcjonalność

4.1.1 Inicjalizacja populacji wejściowej

4.1.2 Operator krzyżowania

4.1.3 Operator mutacji

4.1.4 Funkcja dopasowania

4.2 Klasy

4.2.1 Diagram klas

4.2.2 Funkcjonalność klas

4.3 Algorytmy

4.4 GUI

4.5 Podsumowanie

Rozdział 5

Testy i wyniki

5.1 Testy

5.1.1 Warunki początkowe

5.2 Wyniki

5.3 Wnioski

Rozdział 6

Podsumowanie

6.1 Wyniki końcowe

6.2 Podsumowanie wyników pracy

6.3 Możliwości dalszego rozwoju

Spis treści

Bibliografia

[1] A. Alzohairy. Darwin's theory of evolution. 04 2009.

[2] J. Kominek. Prawa ewolucji Darwina. `http://wiw.org/~jkominek/lojban/9402/msg00074.html`, 1994. [Online; dostęp 09.02.19].

[3] *Encyklopedia Powszechna PWN*. PWN, 1973.

[4] Typy mutacji. `https://evolution.berkeley.edu/evolibrary/article/%3C?%20echo%20$baseURL;%20?%3E/mutations_03l`. [Online; dostęp 09.02.19].

[5] Definicja crossing-over. `https://biologydictionary.net/crossing-over/`. [Online; dostęp 09.02.19].

[6] *Encyklopedia Biologiczna*. GREG Krakow, 2017. Wydanie drugie, poprawione.

[7] K.A. Jadwiszczak A.Chrzanowska. Rola hybrydyzacji międzygatunkowej w kształtowaniu zmienności genetycznej oraz morfologicznej brzóz (betula l.). 2015.

[8] B. Kanber L. Jackobson. *Genetic Algorithms in Java Basics*. Apress, 2015.

[9] O algorytmach genetycznych. `https://www.toptal.com/algorithms/genetic-algorithms`. [Online; dostęp 09.02.19].

[10] Z.Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Wydawnictwo Naukowo-Techniczne, 1999.

[11] Podział Automatycznego Programowania na kategorie. `https://www.techopedia.com/definition/5638/automatic-programming`. [Online; dostęp 10.02.19].

[12] X. Yao A. Arcuri. Co-evolutionary atomatic programming for software development. *Information Sciences*, pages 412–432, 2014.

[13] R. Janczewski. *Złożonosć Obliczeniowa Algorytmów*. Gdańsk, Luty 2002.

[14] J. Hu Y.Matsuya, K. Hirasawa and J. Murata. Automatic Geneartion of Programs using Netework Programming. *Proceedings of the 41st SICE Annual Conference. SICE 2002.*, pages 1269–1274, 2002.

[15] N. Pillay K. Igwe. Atomatic Programming Using Genetic Programming. *2013 Third World Congress on Information and Communication Technologies (WICT 2013)*, pages 337–342, 2013.

[16] Drzewa wyporwadzenia. `http://interactivepython.org/courselib/static/pythonds/Trees/ParseTree.html`.