Politechnika Wrocławska

Wydział Informatyki i Zarządzania
Kierunek studiów: Informatyka
Specjalność: Inteligentne Systemy Informatyczne

Praca dyplomowa - magisterska

Algorytmy ewolucyjne z uwzględnieniem płci w rozwiązywaniu wybranych problemów

Filip Malczak

słowa kluczowe:
algorytmy ewolucyjne
płeć
operator selekcji

krótkie streszczenie:
Bardzo krótkie streszczenie w którym powinno się znaleźć omówienie tematu pracy i poruszanych terminów. Tekst ten nie może być zbyt długi.

Promotor:	.....	.....	.....
	imię i nazwisko	*ocena*	*podpis*

Wrocław 2015

Niniejszy dokument został złożony w systemie LaTeX.

Spis treści

Streszczenie

Abstrakt

Abstract

Abstrakt

Rozdział 1

Wprowadzenie

Optymalizacja to zadanie wyboru najlepszego elementu z danego zbioru (nazywanego przestrzenią rozwiązań), gdzie przez najlepszy rozumiemy taki, dla którego tzw. funkcja oceny, czy też kryterium przyjmuje najwyższą (w zadaniu maksymalizacji) lub najniższą (w zadaniu minimalizacji) wartość. Z takim problemem spotykamy się wszędzie tam, gdzie chcemy zwiększyć lub zmniejszyć jakieś wskaźniki,np. zminimalizować koszta produkcji lub transportu, albo zmaksymalizować zyski ze sprzedaży lub jakość klasyfikacji.

Jeżeli funkcję oceny można zapisać w postaci analitycznej (np. w formie układu równań różniczkowych) to zazwyczaj możemy ją optymalizować metodami numerycznymi, analitycznymi lub algebraicznymi. Istnieją sytuacje, w których nie możemy skorzystać z żadnej z tych metod. Powody tego mogą być różne, m.in. możemy nie znać postaci funkcji (bo np. reprezentuje ona obserwacje jakiegoś procesu lub zjawiska), funkcja zapisana analitycznie może nie spełniać pewnych kryteriów (np. nie być różniczkowalna), lub obliczenia mogą zajmować zbyt wiele czasu (np. pełne przeszukiwanie przestrzeni rozwiązań będącej iloczynem kartezjańskim dużej liczby długich wektorów). Sytuacje takie często charakteryzują się tym, że *de facto* nie jest nam potrzebne globalne optimum (tzn najlepsze rozwiązanie ze wszystkich możliwych), a wystarczy rozwiązanie jak najlepsze (tzn. optimum lokalne, lub punkt o wartości kryterium zbliżonej do wartości kryterium optimum globalnego).

Do takich zastosowań przeznaczone są metody nazywane heurystykami. Heurystyką nazywamy skończony ciąg kroków, który może (ale nie musi) prowadzić do odnalezienia właściwego rozwiązania (czyli, w przypadku optymalizacji - znalezienia optimum).

1.1. Algorytmy ewolucyjne

Ewolucja to proces zachodzący w naturze odpowiedzialny za dopasowywanie się osobników danego gatunku do środowiska w jakim żyją. Podstawą tego procesu jest przetrwanie lepiej przystosowanych osobników, dziedziczenie i mutacja.

Przetrwanie lepiej przystosowanych osobników to zasada zgodnie z którą osobniki lepiej dopasowane do środowiska mają większą szansę na przeżycie, a co za tym idzie, na

wydanie potomstwa. Oznacza to, że rodzice większości osobników z kolejnego pokolenia będą radzić sobie w tym środowisku lepiej niż pozostałe osobniki z ich pokolenia.

Dziedziczenie to zjawisko przekazywania cech rodziców dzieciom. Odbywa się ono podczas rozmnażania, a więc zachodzi między dwojgiem rodziców, a potomstwem. Kod genetyczny potomstwa tworzony jest przez losowe łączenie odpowiednich części kodu genetycznego rodziców, przez co kolejne pokolenie dzieli ich cechy. W ten sposób losowe osobniki przejmą od rodziców te cechy, które pozwalały im się dopasować do środowiska i w niektórych przypadkach pozwoli im to na jeszcze lepsze dopasowanie się do otoczenia. Część osobników przejmie jednak nie tylko cechy poprawiające ich szansę przetrwania, ale również cechy negatywne, co przełoży się na ich gorsze dopasowanie.

Mutacja to zjawisko zachodzenia losowych zmian w kodzie genetycznym osobnika, przez które ma on szansę zyskać nowe cechy, które w niektórych przypadkach doprowadzą do lepszego dopasowania. Osobniki z przypadkowymi zmianami, które poprawiają ich dopasowanie mają większe szanse na przeżycie i wydanie potomstwa, *ergo* przypadkowe pozytywne zmiany powinny zostać rozpropagowane wśród osobników przyszłych pokoleń.

Algorytmy ewolucyjne to rodzina heurystyk naśladujących proces ewolucji w celu optymalizacji [1]. Pojedynczy punkt w przestrzeni rozwiązań jest w nich nazywany osobnikiem. Osobniki możemy między sobą porównywać pod względem wartości optymalizowanej funkcji dla nich, a relacja mniejszości (dla problemów minimalizacji) lub większości (dla problemów maksymalizacji) reprezentuje relację bycia lepiej przystosowanym do środowiska. Ponadto, na osobnikach określone są operatory mutacji i krzyżowania, które mają na celu kolejno symulację losowych zmian w osobniku i tworzenie nowych osobników na podstawie starych. Heurystyka polega na wielokrotnym przetworzeniu populacji (czyli zbioru osobników) poprzez zastosowanie każdego z operatorów z pewnym prawdopodobieństwem. W każdym kroku (nazywanym w nomeklaturze algorytmów ewolucyjnych pokoleniem) do dotychczasowej populacji dołączane są wyniki działania tych operatorów (czyli zbiory osobników zmutowanych i potomstwa), a następnie wybierana jest nowa populacja, używana w kolejnym kroku. Aby odwzorować zasadę przetrwania najlepiej dopasowanych osobników do kolejnej populacji wybierane są z wyższym prawdopodobieństwem osobniki lepiej przystosowane.

W naturze rozmnażanie się osobników wielu gatunków jest ściśle związane ze zjawiskiej podziału gatunku na płcie. Bardziej szczegółowy opis tego zjawiska znajduje się w rozdziale 4. Dotychczasowe rozwiązania rzadko (patrz: rozdział 3) uwzględniały ten aspekt procesu ewolucji. Powodem tego jest raczej uproszczenie działania samej heurystyki niż lepsza jakość wyników uzyskiwanych z pominięciem tego aspektu ([2], [3]).

1.2. Cele pracy

Cele tej pracy zostały opisane w tabeli 1.1.

Tabela 1.1. Cele pracy opisanej w tym dokumencie

Nr.	Cel	Opis
1	Formalizacja ogólnego schematu algorytmu ewolucyjnego korzystającego ze zjawiska płci	Schemat algorytmu ewolucyjnego jest w pewnym stopniu dowolny, a stosunkowo mała ilość publikacji w temacie używa różnych formalizmów (patrz: rozdział 3). Powoduje to brak jednej standardowej implementacji, a co za tym idzie utrudnia porównywanie rozwiązań między sobą. Spełnienie tego celu powinno doprowadzić do określenia konkretnego schematu, który pozwoli zaimplementować wybrane istniejące rozwiązania oraz zaproponowane rozwiązanie autorskie.
2	Realizacja i implementacja w zaproponowanym schemacie wybranych rozwiązań literaturowych	Spełnienie tego celu powinno doprowadzić do zdefiniowania wybranych rozwiązań literaturowych w schemacie określonym w celu nr. 1 i implementacji frameworku badawczego opartego o ten schemat, wraz z implementacją tych rozwiązań.
3	Propozycja, realizacja i implementacja nowego rozwiązania w proponowanym schemacie	Jest to główny cel niniejszej pracy, którego efektem powinno być stworzenie nowego podejścia do zagadnienia płci, zbadanie go i porównanie z rozwiązaniami literaturowymi. Realizacja i implementacja powinna się odbyć we frameworku badawczym stworzonym przy celu 2.
4	Zbadanie działania różnych rozwiązań dla wybranych problemów	Podczas realizacji tego celu powinny zostać zebrane wybrane miary jakości działania algorytmu ewolucyjnego dla każdego z podejść, tak aby możliwe było miarodajne porównanie ich ze sobą.

Rozdział 2

Algorytmy ewolucyjne

Algorytmy ewolucyjne to rodzina heurystyk populacyjnych, tzn. heurystyk które operują i zwracają zbiór rozwiązań zamiast pojedynczego wyniku. Podstawowe pojęcia w tej dziedzinie to osobnik, populacja, ocena (nazywana też funkcją oceny, kryterium lub funkcją dopasowania), operator krzyżowania, operator mutacji, operator selekcji i warunek stopu (nazywany też warunkiem zakończenia lub przerwania). Poniższe rozdziały opisują szczegółowe znaczenie wymienionych pojęć, sposób zastosowania heurystyki do rozwiązania problemu oraz sposoby oceny działania heurystyki.

2.1. Działanie

Wymienione wyżej pojęcia określają elementy heurystyki które musimy zdefiniować aby ją zastosować w celu rozwiązania problemu. Każde z tych pojęć niesie ze sobą nie tylko znaczenie i zastosowanie wzorowane na przyrodzie, ale również ograniczenia nałożone na realizację danego elementu, również wynikające z natury.

W tabeli 2.1 zostały pokazane konwencje dotyczące znaczenia czcionek używane w tej pracy.

2.1.1. Pojęcie operatora

W dalszych rozdziałach będziemy używać pojęcia „**operator**", które jest zbliżone do pojęcia funkcji. Różnica między tymi dwoma terminami jest taka, że funkcja musi zawsze zwrócić tą samą wartość dla tego samego argumentu, podczas gdy takie ograniczenie nie musi być spełnione dla operatora. Innymi słowy pojęcie operatora pokrywa

Tabela 2.1. Konwencje dotyczące czcionek

`operator`, `n`	operatory i parametry heurystyki
T, p	zbiory i parametry pomocnicze
$\mathcal{S}$	zbiory używane w definicjach
$\mathbf{R}$	relacje
$\mathbb{N}$	zbiory liczbowe

się z pojęciem funkcji z imperatywnych języków programowania, ale nie pokrywa się z funkcją w rozumieniu matematycznym.

Operator możemy też rozumieć jako zmienną losową opisaną rozkładem parametryzowanym argumentami operatora.

Warto pamiętać, że każda funkcja (w rozumieniu matematycznym) jest operatorem.

Do operatorów możemy stosować niektóre pojęcia używane w stosunku do funkcji. Przez *dziedzinę* operatora rozumiemy przestrzeń dozwolonych argumentów operatora, a przez *przeciwdziedzinę* - zbiór wartości które może zwrócić. Aby uzyskać *wynik* operatora (czyli wartość przez niego zwracaną) *stosujemy* ten operator na *argumentach*.

Dodatkowo, operator możemy *sparametryzować* aby uzyskać inny operator. Przez *parametry* rozumiemy argumenty które w ramach danego ciągu obliczeń są stałe. Przykładowo, operator sąsiedztwa o określonym rozmiarze d dla punktu *(x, y)*:

$$sasiedztwo((x,y),d) \in [x-d, x+d] \times [y-d, y+d]$$

możemy sparametryzować rozmiarem sąsiedztwa, aby uzyskać operator sąsiedztwa o konkretnym rozmiarze:

$$sasiedztwo_5(P) = sasiedztwo(P,5)$$

De facto każdy operator opisywany w tej pracy może być parametryzowany. W kolejnych rozdziałach przyjęto konwencję według której definiując nowy operator zapisywana jest jego minimalna dziedzina, co nie oznacza, że podczas realizacji nie może on być parametryzowany. Analogicznie stwierdzenie, że operator powinien przyjmować daną liczbę argumentów nie oznacza, że nie może on przyjmować ich więcej. Innymi słowy sygnatura postaci:

$$operator : D \rightarrow C$$

gdzie D oznacza dziedzinę, a C przeciwdziedzinę, jest równoważna z:

$$operator : D \times P \rightarrow C$$

gdzie P oznacza przestrzeń parametrów, zależnych od realizacji operatora.

2.1.2. Osobnik, populacja i ocena

Podstawowym pojęciem używanym w opisywanej heurystyce jest **osobnik**. Jest to abstrakcja pojedynczego rozwiązania, nawiązująca do pojedynczego żywego stworzenia charakteryzującego się pewnymi cechami które wpływają na prawdopodobieństwo jego przeżycia i wydania potomstwa. Pojęciem symulującym wpływ cech na prawdopodobieństwo przeżycia jest **funkcja oceny**. Dodatkowo wprowadzamy pojęcie **populacji**, czyli zbioru osobników istniejących w danym momencie w procesie ewolucji.

Realizacją osobnika w heurystyce jest reprezentacja rozwiązania problemu, a realizacją populacji - zbiór rozwiązań, czyli podzbiór przestrzeni rozwiązań.

Ważniejszym pojęciem jest funkcja oceny. Wbrew nazwie nie musi być to funkcja w rozumieniu matematycznym, a operator (patrz: rozdział 2.1.1) [1]. Jego dziedziną powinna być przestrzeń możliwych osobników, a przeciwdziedziną dowolny zbiór na którym możemy określić relację porządku. Relacja ta odpowiada relacji bycia lepiej przystosowanym do środowiska w rzeczywistym procesie ewolucji.

Standardową realizacją osobnika (zaproponowaną w pierwszym artykule dot. EA cytować toto) jest wektor binarny. Algorytmy ewolucyjne korzytające z takiej reprezentacji nazywane są algorytmami genetycznymi. Reprezentacja jest jednak uzależniona od rozwiązywanego problemu. Często wykorzystuje się bardziej skomplikowane przedstawienia rozwiązania, jak wektor wartości z pewnego zbioru (skończonego, lub nieskończonego, jak liczby), czy drzewo (np. dla problemów aproksymacji funkcji) może znajde źródła z czymś takim .

Istnieją implementacje i frameworki do obliczeń ewolucyjnych które rozróżniają pojęcie genotypu osobnika (czyli reprezentacji na której stosujemy operatory genetyczne, jak krzyżowanie i mutacja) i jego fenotypu (czyli reprezentacji, na podstawie której oceniamy osobnika), jednak jest to jedynie abstrakcja umożliwiająca czytelniejsze zapisanie realizacji odpowiednich pojęć w języku programowania. Niniejsza praca nie wprowadza takiego rozróżnienia.

framework - czy to się da przetłumaczyć?

Sygnatura 1 Osobnik

$$osobnik \in \mathcal{S} \tag{2.1}$$

Zbiór $\mathcal{S}$ to przestrzeń rozwiązań.

Sygnatura 2 Funkcja oceny

$$\texttt{funkcjaOceny} : \mathcal{S} \to \mathcal{E} \tag{2.2}$$

$$\exists \mathbf{R}_{\prec} \subset \mathcal{E} \times \mathcal{E} \tag{2.3}$$

$$\mathbf{R}_{\lhd} \leftarrow \{(x, y) : (\texttt{funkcjaOceny}(x), \texttt{funkcjaOceny}(y)) \in \mathbf{R}_{\prec}\} \tag{2.4}$$

Zbiór $\mathcal{E}$ to zbiór możliwych ocen osobnika, a $\mathbf{R}_{\prec}$ to relacja porządku na nim określona. Ponadto określamy relację lepszego dopasowania $\mathbf{R}_{\lhd}$, porządkującą przestrzeń osobników według porządku określonego na ich ocenach.

2.1.3. Operatory genetyczne

Dwa podstawowe operatory używane w algorytmach ewolucyjnych to **operator mutacji** i **operator krzyżowania**. Są one stosowane na osobnikach z populacji w danym pokoleniu w celu przeszukania lokalnej przestrzeni rozwiązań. W zależności od

1. Stochastyczną funkcję oceny możemy zastosować na przykład w sytuacji, w której za pomocą algorytmu ewolucyjnego szukamy konfiguracji innej heurystyki. Osobnikiem w takiej sytuacji może być zestaw parametrów konfigurowanej heurystyki, a oceną - średnia jakość działania dla kilku uruchomień.

implementacji, kolejność stosowania operatorów jest różna, a wyniki ich zastosowania są dołączane do całej populacji w danym pokoleniu, lub składają się na nową populację, używaną w kolejnym pokoleniu. W niniejszej pracy stosowane będzie podejście zgodnie z którym osobniki są dołączane do obecnej populacji, która pod koniec obecnego pokolenia jest zmniejszana za pomocą operatora selekcji (opisanego w rozdziale 2.1.4).

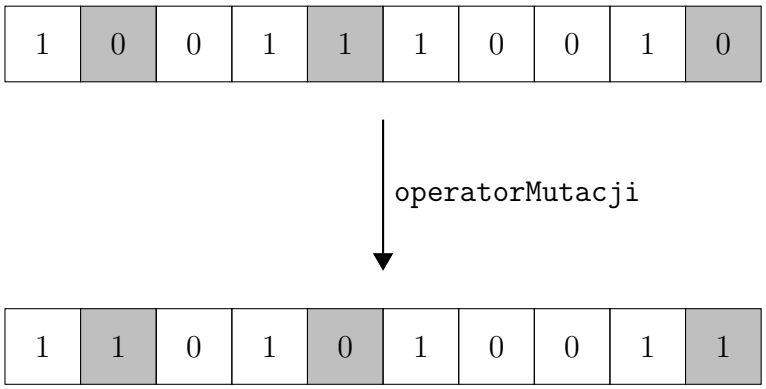
Operator mutacji

Operator mutacji odpowiada za symulowanie losowych zmian zachodzących w kodzie genetycznym. Jego zadaniem jest sterowanie eksploracją przestrzeni rozwiązań.

Dziedziną tego operatora jest przestrzeń rozwiązań, a przeciwdziedziną - $k$krotny iloczyn kartezjański przestrzeni rozwiązań. Oznacza to, że jednokrotne zastosowanie operatora mutacji daje w wyniku k osobników powstałych przez modyfikację osobnika pierwotnego. Każdy ze zwróconych osobników powinien być nieznacznie różny od argumentu operatora.

Jedną z popularnych realizacji tego operatora dla osobnika reprezentowanego jako wektor bitów jest negacja losowych wartości, w wyniku czego otrzymujemy osobniki różniące się od argumentu operatora tylko kilkoma elementami. Na rysunku 2.1 zobrazowano działanie takiego podejścia. Wektor $(1, 0, 0, 1, 1, 1, 0, 0, 1, 0)$ to argument operatora. Kolorem zaznaczono losowe pozycje w wektorze, które zostaną zanegowane, w wyniku czego powstanie wektor wyjściowy $(1, \mathbf{1}, 0, 1, \mathbf{0}, 1, 0, 0, 1, \mathbf{1})$.

Rys. 2.1. Działanie przykładowej realizacji operatora mutacji



Z operatorem mutacji ściśle związany jest jeden z parametrów algorytmu ewolucyjnego - prawdopodobieństwo mutacji. Jest to wartość określająca prawdopodobieństwo zastosowania operatora mutacji do osobnika w danym pokoleniu. Jeśli wartość ta jest równa 0, to operator mutacji nie jest w ogóle używany, a jeśli jest równa 1, to operator zostanie zastosowany do każdego osobnika.

Sygnatura 3 Operator mutacji

$$\texttt{operatorMutacji} : \mathcal{S} \rightarrow \mathcal{S}^k \tag{2.5}$$

$$\texttt{prawdopodobienstwoMutacji} \in [0, 1] \tag{2.6}$$

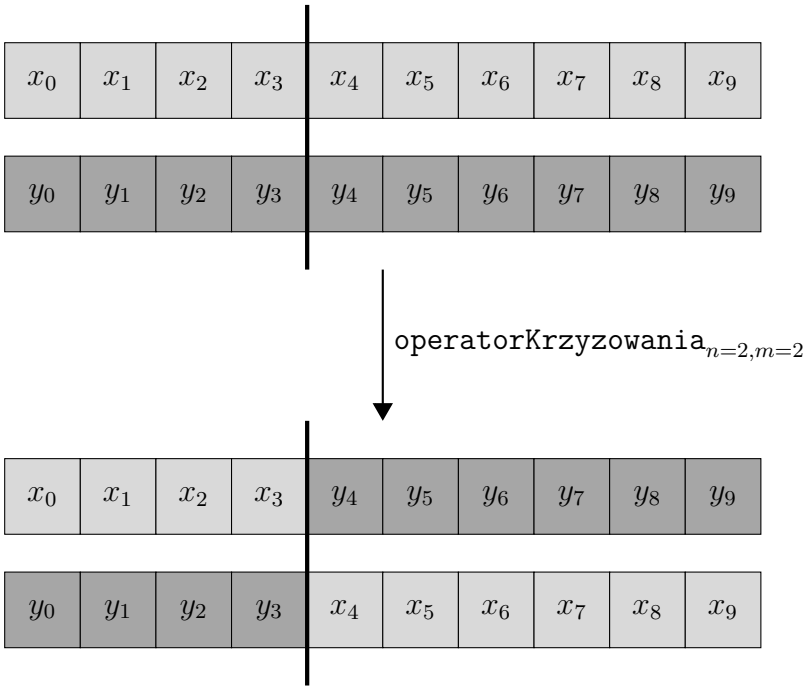
Operator krzyżowania

Operator krzyżowania symuluje proces krzyżowania się osobników w naturze. Jego zadaniem jest sterowanie eksploatacją przestrzeni rozwiązań.

Dziedziną tego operatora jest $n$krotny iloczyn kartezjański przestrzeni rozwiązań, a przeciwdziedziną - $m$krotny iloczyn tej przestrzeni. Oznacza to, że operator ten przyjmuje n osobników (nazywanych *rodzicami*) jako argument, a zwraca m osobników (nazywanych *potomstwem*). Zazwyczaj przyjmuje się $n = 2$ i $m \in \{1, 2\}$. Zwracane osobniki powinny być podobne (w takim sensie, że mają podobną reprezentację) do osobników wejściowych (argumentów operatora).

Jedną z popularnych realizacji dla osobnika reprezentowanego jako wektor dla $n = 2$ jest wybranie losowej pozycji w wektorze (tzw. punktu przecięcia) i zwrócenie 2 osobników powstałych przez zamianę podwektorów od wylosowanej pozycji wzwyż. Na rysunku 2.2 zobrazowano działanie takiego podejścia. Wektory $(x_0, \ldots, x_9)$ i $(y_0, \ldots, y_9)$ to wektory wejściowe. Grubą linią zaznaczono wylosowany punkt przecięcia, wyznaczający podwektory które zostają zamienione. Wynikiem zastosowania operatora są wektory $(x_0, \ldots, x_3, y_4, \ldots y_9)$ i $(y_0, \ldots, y_3, x_4, \ldots x_9)$.

Rys. 2.2. Działanie przykładowej realizacji operatora krzyżowania



Z operatorem krzyżowania ściśle związany jest jeden z parametrów algorytmu ewolucyjnego - prawdopodobieństwo krzyżowania. Jest to wartość określająca prawdopodobieństwo zastosowania operatora krzyżowania do osobnika w danym pokoleniu. Jeśli wartość ta jest równa 0, to operator krzyżowania nie jest w ogóle używany, a jeśli jest równa 1, to operator zostanie zastosowany do każdego osobnika.

Sygnatura 4 Operator krzyżowania

$$\texttt{operatorKrzyzowania} : \mathcal{S}^n \to \mathcal{S}^m \tag{2.7}$$
$$\texttt{prawdopodobienstwoKrzyzowania} \in [0, 1] \tag{2.8}$$

n to liczba rodziców, a m to liczba potomków.

2.1.4. Operator selekcji

Zadaniem **operatora selekcji** jest symulacja zjawiska przeżycia najsilniejszych (czyli najlepiej dopasowanych) osobników. Jest on stosowany pod koniec każdego pokolenia w celu usunięcia z niej osobników gorzej dopasowanych, poprzez wykorzystanie populacji wyjściowej jako używanej w kolejnym pokoleniu.

Dziedziną tego operatora jest $p$krotny iloczyn kartezjański przestrzeni rozwiązań, co oznacza, że przyjmuje on populację p osobników. Wartość p to ilość osobników w populacji pod koniec danej generacji, więc można ją przybliżać za pomocą równać z linii 2.11-2.13 sygnatury 5, ponieważ w populacji znajdą się osobniki z populacji początkowej tego pokolenia oraz wyniki operatorów mutacji i krzyżowania.

Przeciwdziedziną operatora selekcji jest `rozmiarPopulacji`krotny iloczyn kartezjański przestrzeni rozwiązań, co oznacza, że wynikiem działania operatora powinien być zbiór `rozmiarPopulacji` osobników. Wartość `rozmiarPopulacji` to parametr całej heurystyki, określająca jak liczna powinna być populacja na początku każdego pokolenia. Im większą wartość przyjmuje ten parametr, tym lepsze przeszukiwanie przestrzeni rozwiązań, ale też tym dłużej trwa sama heurystyka (ponieważ trzeba ewaluować więcej osobników i do większej liczby z nich zastosować operatory genetyczne).

Dodatkowo na operator nakłada się wymóg opisany w liniach 2.14-2.17 sygnatury 5. Mówią one o tym, że prawdopodobieństwo tego, że osobnik z populacji wejściowej znajdzie się w populacji wyjściowej powinno być tym większe im lepiej dopasowany jest osobnik. Ma to symulować zasadę przetrwania osobników najlepiej przystosowanych do środowiska.

jakiś przykład

Sygnatura 5 Operator selekcji

$$\texttt{operatorSelekcji} : \mathcal{S}^p \to \mathcal{S}^{\texttt{rozmiarPopulacji}} \tag{2.9}$$
$$\texttt{rozmiarPopulacji} \in \mathbb{N}_+ \tag{2.10}$$
$$p \approx (1 + \texttt{prawdopodobienstwoMutacji} \times k \tag{2.11}$$
$$+ \texttt{prawdopodobienstwoKrzyzowania} \times m) \tag{2.12}$$
$$\times \texttt{rozmiarPopulacji} \tag{2.13}$$
$$T \leftarrow \{t_0, t_1, \ldots t_{\texttt{rozmiarPopulacji}-1}\} \tag{2.14}$$
$$s \in \texttt{operatorSelekcji}(T) \Leftrightarrow \exists_{t \in T} t \equiv s \tag{2.15}$$
$$P(i) \leftarrow P(t_i \in \texttt{operatorSelekcji}(T)) \tag{2.16}$$
$$P(i) < P(j) \Leftrightarrow (t_i, t_j) \in \mathbf{R}_\lhd \tag{2.17}$$

2.1.5. Warunek stopu

Warunek stopu, nazywany też warunkiem zatrzymania lub przerwania to pojęcie określające warunek zakończenia heurystyki. Jest to odpowiedź na pytanie „kiedy należy przestać przetwarzanie kolejnych pokoleń?". Często stosuje się takie kryteria jak przekroczenie jakiejś arbitralnej liczby pokoleń, lub zaobserwowanie tzw. stagnacji. Zjawisko takie polega na znacznym zmniejszeniu różnorodności ocen w populacji, co oznacza, że znaleźliśmy optimum lokalne. Jeśli wariancja ocen przez kilka pokoleń się nie zmienia (lub zmienia, ale nie znacząco), to można uznać, że operatory genetyczne nie pozwolą na wyjście z tego optimum i przerwać działanie heurystyki. Warunek zatrzymania możemy traktować jako dodatkowy operator, jednak trudno jest określić jego jednoznaczną sygnaturę, ze względu na dowolność realizacji. W sygnaturze 6 przedstawiono ogólny zapis operatora (z dowolną dziedziną, linia 2.18) i sygnaturę jego przykładowej realizacji (opartej o stałą liczbę pokoleń, linia 2.19).

Sygnatura 6 Warunek zatrzymania i jego przykładowa realizacja

$$\texttt{warunekStopu} : \ldots \rightarrow \{0, 1\} \tag{2.18}$$

$$\texttt{warunekStopu} : \mathbb{N} \rightarrow \{0, 1\} \tag{2.19}$$

$$\texttt{warunekStopu}(g) = 1 \Leftrightarrow g \leqslant \overline{g} \tag{2.20}$$

Zwrócenie wartości 1 oznacza, że należy zakończyć działanie heurystyki, a 0 - sytuację odwrotną.
Wartość g oznacza numer obecnego pokolenia (zaczynając od 1), a $\overline{g}$ - maksymalną dopuszczalną liczbę pokoleń w ramach jednego przebiegu algorytmu ewolucyjnego.

2.1.6. Schemat algorytmu

Na rysunku 2.3 zobrazowane zostały różne elementy schematów blokowych używane w tej pract, opisane w tabeli 2.2. Ponadto, w algorytmach używany jest operator $random(X)$ opisany sygnaturą 7, zwracający losowy element zbioru X (z rozkładem równomiernym).

Sygnatura 7 Operator $random(S)$

$$random : A^n \rightarrow A \tag{2.21}$$

$$n \in \mathbb{N}_+ \tag{2.22}$$

$$X \leftarrow \{x_0, x_1, \ldots x_{n-1}\} \tag{2.23}$$

$$\forall_{x_i, x_j \in X} P(random(X) = x_i) = P(random(X) = x_j) \tag{2.24}$$

A to dowolny zbiór, a n jego rozmiar.

Rys. 2.3. Elementy schematów blokowych używane w tej pracy

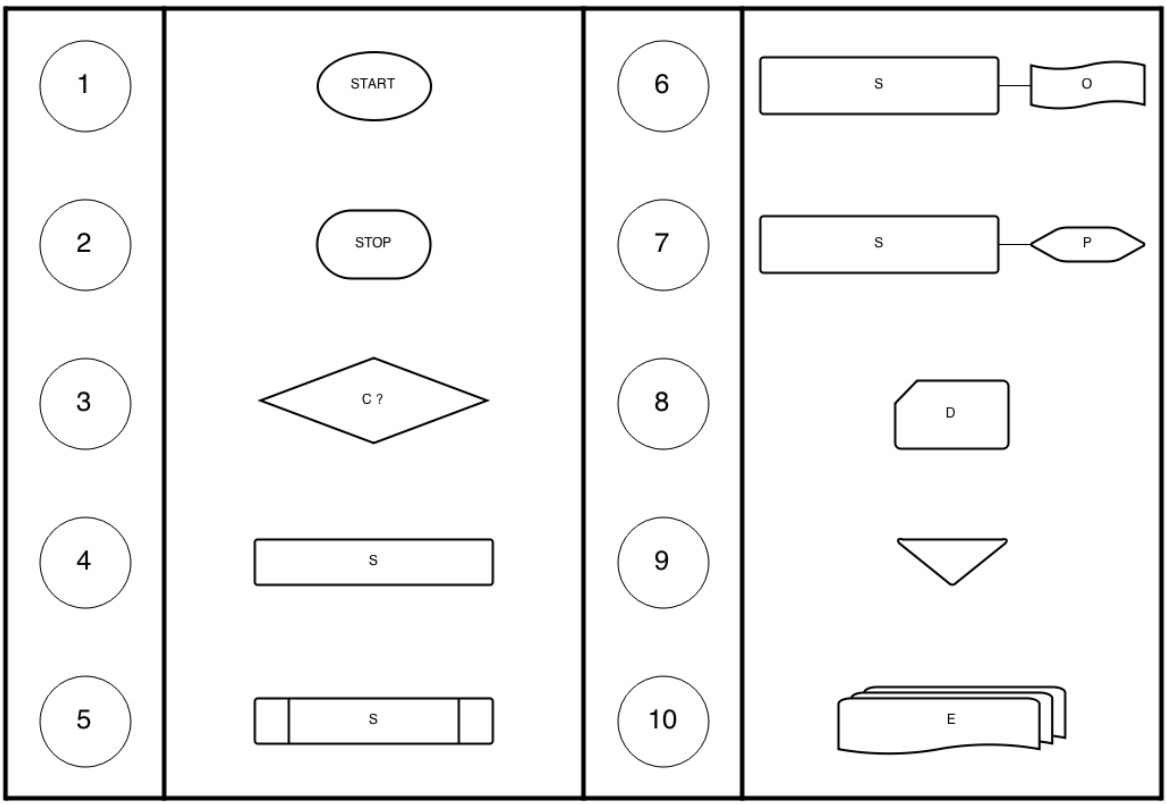
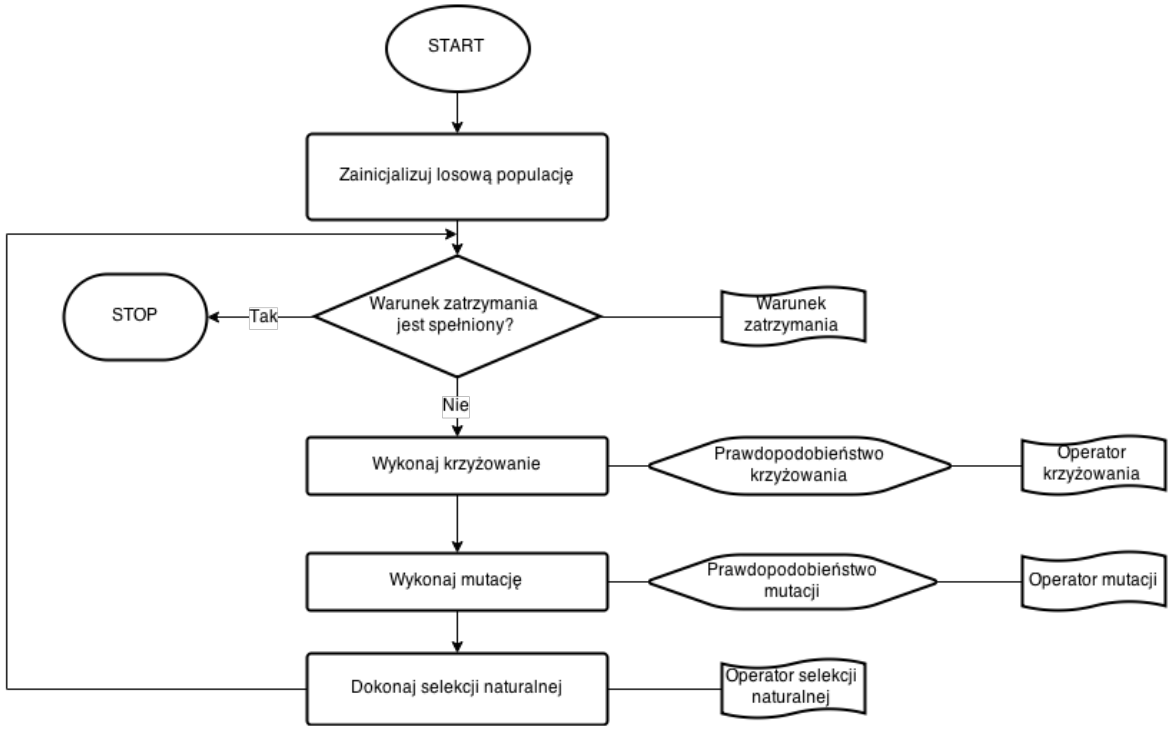


Tabela 2.2. Opis konwencji dotyczących elementów schematów blokowych

①	Symbol rozpoczęcia heurystyki.
②	Symbol rozpoczęcia heurystyki.
③	Symbol oznaczający ewaluację warunku C i kontynuację procesu zgodnie z jego wynikiem.
④	Symbol oznaczający wykonanie kroku opisanego przez S.
⑤	Symbol tożsamy z symbolem ④ , jednak stosowany w opisie proponowanego podejścia dla zaznaczenia różnic ze standardowym schematem (używane w rozdziale 4).
⑥	Taka kombinacja symboli oznacza, że podczas wykonywania kroku S używany jest operator O.
⑦	Taka kombinacja symboli oznacza, że podczas wykonywania kroku S używany jest parametr P.
⑧	Symbol oznaczający, że dany schemat wyjaśnia szczegółowe działanie kroku D.
⑨	Symbol oznaczający rozpoczęcie lub zakończenie szczegółowo opisywanego kroku.
⑩	Symbol oznaczający wykonanie kolejnego symbolu dla każdego z elementów opisanych przez E.

Rys. 2.4. Ogólny schemat działania algorytmów ewolucyjnych



Na rysunku 2.4 zobrazowany został ogólny schemat działania algorytmów ewolucyjnych.

Cały proces rozpoczyna się od inicjalizacji populacji przy pomocy losowych osobników. Następnie, póki warunek zatrzymania nie jest spełniony, powtarzane jest kilka kroków wykonywanych w ramach jednego pokolenia. Są to kolejno: krzyżowanie, mutacja i selekcja naturalna.

Algorytm 2.1 opisuje szczegóły kroku "Wykonaj krzyżowanie". Krok ten rozpoczyna się od inicjalizacji zmiennej *noweOsobniki* na pusty zbiór (w linii 2.), który w linii 10. zostanie dołączony do ogólnej populacji. Dla każdego osobnika w populacji (linia 3) sprawdzane jest, czy należy wykonać krzyżowanie z jego udziałem (linia 4). Jeżeli tak, to losowo dobierany jest jego partner (linia 5), a następnie na tak określonej parze osobników stosowany jest operator krzyżowania, którego wynik jest dołączany do zbioru *noweOsobniki*. Schemat ten przewiduje, że operator krzyżowania przyjmuje 2 osobniki jako argument (tzn. $n = 2$, patrz: rozdział 2.1.3), przez co wartość, z którą porównujemy losowo wybraną liczbę z przedziału $[0, 1]$ to $\texttt{prawdopodobienstwoKrzyzowania}/2$. W ogólności, dla dowolnego n wartość ta wynosi $\texttt{prawdopodobienstwoKrzyzowania}/n$, a zamiast jednego partnera należy wybrać ich $n - 1$.

Algorytm 2.2 opisuje szczegóły kroku "Wykonaj mutację". Krok rozpoczyna się od inicjalizacji zmiennej *noweOsobniki* na pusty zbiór (w linii 2.), który w linii 9. zostanie dołączony do ogólnej populacji. Dla każdego osobnika w populacji (linia 3) sprawdzane jest, czy należy wykonać na nim mutację (linia 4), a jeśli tak, to stosowany jest do niego operator mutacji, którego wynik zostaje dołączony do zbioru *noweOsobniki* (linie 5 i 6).

Algorytm 2.3 opisuje szczegóły kroku "Dokonaj selekcji naturalnej", który *de facto*

sprowadza się do zastąpienia dotychczasowej populacji wynikiem zastosowania operatora selekcji na niej (linia 1).

Algorytm 2.1 Szczegółowy schemat działania kroku "Wykonaj krzyżowanie"

Potrzebne zmienne, parametry i operatory:
populacja ▷ Obecna populacja, tj. zbiór osobników
`prawdopodobienstwoKrzyzowania` ▷ Prawd. krzyżowania
`operatorKrzyzowania` ▷ Operator krzyżowania

1: **procedure** CROSSINGOVER
2: **var** $noweOsobniki \leftarrow \emptyset$
3: **for all** $osobnik \in populacja$ **do**
4: **if** $random([0,1]) \leqslant$ `prawdopodobienstwoKrzyzowania`$/2$ **then**
5: **var** $partner \leftarrow random(populacja \setminus \{osobnik\})$
6: **var** $potomek \leftarrow$ `operatorKrzyzowania`$(osobnik, partner)$
7: $noweOsobniki \leftarrow noweOsobniki \cup \{potomek\}$
8: **end if**
9: **end for**
10: $populacja \leftarrow populacja \cup noweOsobniki$
11: **end procedure**

Algorytm 2.2 Szczegółowy schemat działania kroku "Wykonaj mutację"

Potrzebne zmienne, parametry i operatory:
populacja ▷ Obecna populacja, tj. zbiór osobników
`prawdopodobienstwoMutacji` ▷ Prawd. mutacji
`operatorMutacji` ▷ Operator mutacji

1: **procedure** MUTATION
2: **var** $noweOsobniki \leftarrow \emptyset$
3: **for all** $osobnik \in populacja$ **do**
4: **if** $random([0,1]) \leqslant$ `prawdopodobienstwoMutacji` **then**
5: **var** $mutant \leftarrow$ `operatorMutacji`$(osobnik)$
6: $noweOsobniki \leftarrow noweOsobniki \cup \{mutant\}$
7: **end if**
8: **end for**
9: $populacja \leftarrow populacja \cup noweOsobniki$
10: **end procedure**

Tak zdefiniowany proces służy do efektywnego przeszukiwania przestrzeni rozwiązań, tymczasem zadanie optymalizacji ma na celu znalezienie jednego, jak najlepszego rozwiązania. Oznacza to, że jedno z rozwiązań, które zostanie zbadane w trakcie działania heurystyki musi być zapamiętane i zwrócone jako wynik. Naiwnym podejściem jest analiza populacji po ostatnim pokoleniu i zwrócenie najlepszego z osobników. Jako że operator selekcji nie gwarantuje tego, że w zbiorze wynikowym znajdzie się najlepszy osobnik ze zbioru wejściowego, to istnieje szansa, że najlepsze rozwiązanie z ostatniej populacji nie jest najlepszym rozwiązaniem znalezionym w trakcie działania heurystyki. Zamiast tego podczas działania heurystyki zapamiętujemy globalnie najlepsze rozwią-

Algorytm 2.3 Szczegółowy schemat działania kroku "Dokonaj selekcji naturalnej"

Potrzebne zmienne, parametry i operatory:
populacja ▷ Obecna populacja, tj. zbiór osobników
`rozmiarPopulacji` ▷ Rozmiar populacji
`operatorSelekcji` ▷ Operator selekcji

1: **procedure** NATURALSELECTION
2: $populacja \leftarrow \texttt{operatorSelekcji}_{\texttt{rozmiarPopulacji}}(populacja)$
3: **end procedure**

zanie na jakie trafiliśmy i właśnie je traktujemy jako rozwiązanie problemu optymalizacji. Zazwyczaj realizuje się to w ten sposób, że definiuje się zmienną przechowującą globalne optimum, która początkowa ma wartość pustą. Po zakończeniu pierwszego pokolenia, ale przed zastosowaniem operatora selekcji zapisuje się w niej najlepsze rozwiązanie w tym pokoleniu. W dalszych pokoleniach, również przed zastosowaniem operatora selekcji, porównuje się ocenę obecnego globalnego optimum i najlepszego osobnika z danego pokolenia (optimum lokalnego). Jeśli optimum lokalne jest lepsze niż globalne, to zastępuje je ono.

2.2. Analiza jakości działania

Algorytmy ewolucyjne to rodzina losowych, iteracyjnych heurystyk populacyjnych. Określenie „losowych" oznacza, że każde uruchomienie procesu optymalizacji może zwrócić inny wynik. Co za tym idzie, ocena działania heurystyki dla danego zestawu parametrów jest trudna i wymaga wielokrotnego zebrania wyników. W następnych rozdziałach opisane zostaną sposoby analizy jakości działania pojedynczego przebiegu heurystyki, jak i wielu przebiegów o tych samych parametrach.

2.2.1. Analiza jednego przebiegu heurystyki

Jak było wspomniane wyżej, algorytmy ewolucyjne to iteracyjne heurystyki populacyjne. Oznacza to, że w ramach jednego procesu optymalizacji wielokrotnie powtarzamy pewien krok (stąd określenie „iteracyjne") w którym badamy wiele rozwiązań (stąd określenie „populacyjne"). polskie cudzysłowie

Aby ocenić pojedynczy przebieg, możemy analizować pewne statystyki oceny w skali populacji na przestrzeni wielu pokoleń.

Podstawowe statystyki używane w tym celu, to m.in.:

- najlepsza i najgorsza ocena osobnika z populacji,
- średnia i odchylenie standardowe (lub wariancja [2]) oceny osobników w populacji,
- mediana i kwantyle oceny osobników w populacji.

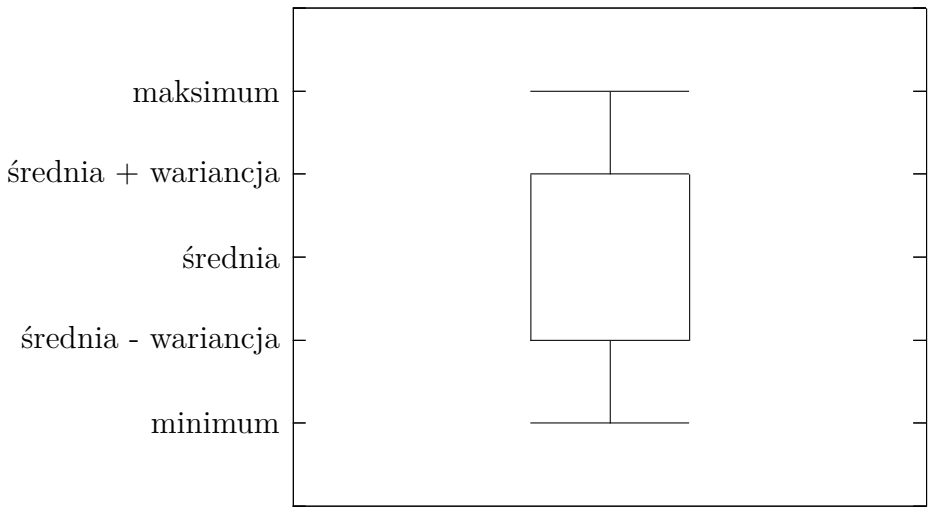
Wartości te można wygodnie zobrazować na wykresie pudełkowym, na osi odciętych umieszczając numery pokoleń, a na osi rzędnych - wartości odpowiednich statystyk. Takie zobrazowanie przebiegu heurystyki pozwala na wyciągnięcie wniosków i stosowne

2. Często wariancji używa się do automatyzacji oceny, ponieważ obliczenie jej nie wymaga czasochłonnego pierwiastkowania. Znając wariancję w dowolnym momencie możemy obliczyć odchylenie standardowe, np. w celu prezentacji oceny.

dopasowanie parametrów heurystyki (np. jeśli w problemie minimalizacji wszystkie wartości stopniowo maleją, to być może warto zwiększyć rozmiar populacji lub zmienić kryterium zatrzymania, aby cały proces trwał dłużej, ponieważ taka obserwacja wskazuje na efektywne działanie heurystyki).

W tej pracy do analizy wykorzystano tylko 4 z wyżej wymienionych statystyk: średnią, wariancję[3], minimum i maksimum. Są one w pełni wystarczające do analizy porównawczej między uruchomieniami. Na rysunku 2.5 przedstawiono konwencje dotyczące rysowania wykresów przyjęte w tym dokumencie.

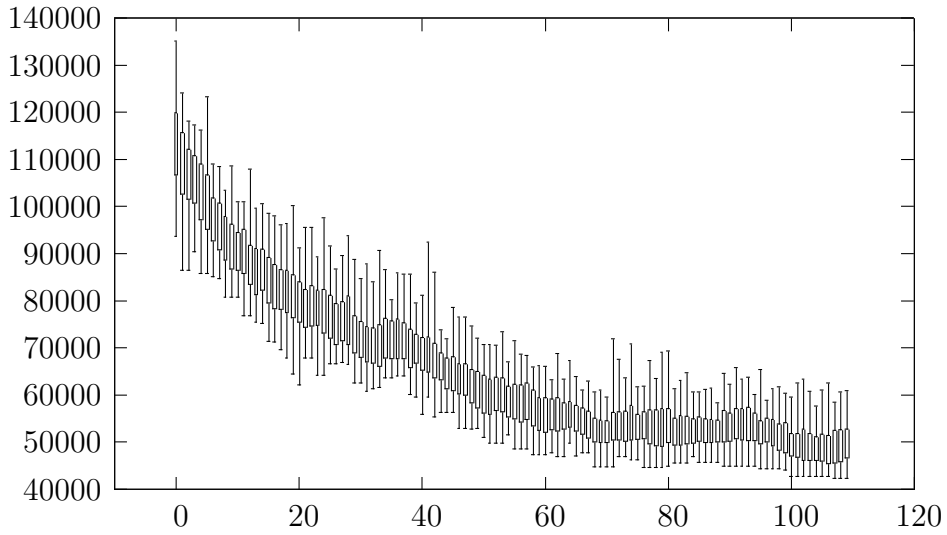
Rys. 2.5. Konwencje dotyczące rysowania wykresów



Na rysunku 2.6 przedstawiony jest przykładowy wykres statystyk populacji na przestrzeni wielu pokoleń. Wykresy takie, nazywane w skrócie *wykresami przebiegów*, będą wykorzystywane w tej pracy do analizy pojedynczych uruchomień heurystyki.

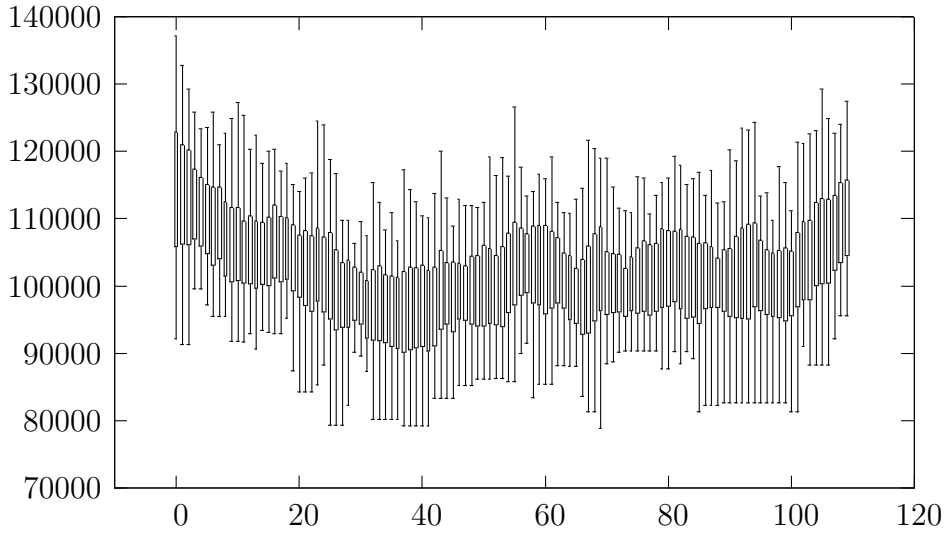
3. Zdecydowano się na wykorzystanie wariancji zamiast odchylenia standardowego ponieważ wartości odchylenia okazały się być zbyt małe, aby były dostrzegalne na wykresach.

Rys. 2.6. Przykładowy wykres przebiegu



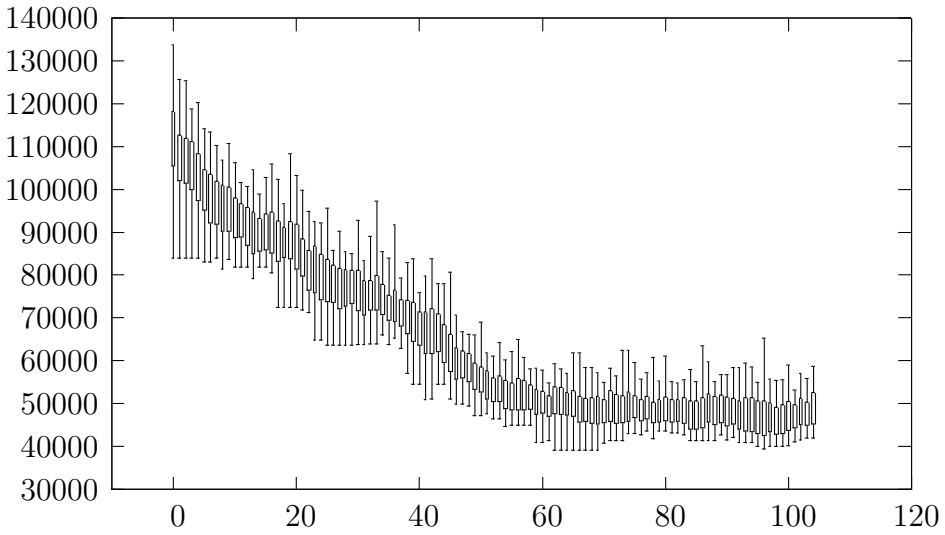
Na rysunku 2.7 przedstawiony jest wykres przebiegu na którym możemy zaobserwować sytuację opisaną na końcu rozdziału 2.1.6, tzn. taką, w której optimum globalne jest znalezione w innym pokoleniu niż ostatnie.

Rys. 2.7. Wykres przebiegu, w którym globalne optimum nie znajduje się w ostatniej populacji



Na rysunku 2.8 przedstawiono sytuację, w której obserwujemy tzw. stagnację (wspominaną już w rozdziale 2.1.5). Możemy zaobserwować, że od pewnego momentu (około sześćdziesiątej generacji) kolejne pokolenia nie przynoszą znaczącej zmiany wyniku, co mogłoby być powodem do przerwania działania heurystyki.

Rys. 2.8. Wykres przebiegu w którym obserwujemy stagnację



2.2.2. Analiza wielu przebiegów heurystyki

Jak zostało wspomniane na początku tego rozdziału, algorytmy ewolucyjne to heurystyki losowe, przez co za każdym uruchomieniem zwracają różne wartości. Aby ocenić wyniki procesu optymalizacji dla różnych zestawów parametrów należy kilkukrotnie powtórzyć proces i porównywać statystyki wyników. Jeśli jesteśmy w trakcie dostrajania heurystyki (czyli dobierania najlepszych parametrów), to taką statystyką może być najlepszy wynik z kilku powtórzeń, jednak jeśli chcemy przeprowadzić miarodajne badania, to najprostszym podejściem dającym wgląd w jakość działania jest obliczenie średniej i wariancji (lub odchylenia standardowego, patrz: przypis 2) wyników wielu przebiegów dla różnych konfiguracji i porównanie ich testem statystycznym, takim jak np. test t-studenta.

Rozdział 3

Przegląd literatury

Rozdział 4

Proponowane rozwiązania

This stuff gotta be fixed - patrz: notatki od promotora

Dotychczasowe implementacje algorytmów ewolucyjnych zazwyczaj (choć nie zawsze [2], [3]) pomijały ważny aspekt procesu ewolucji, który w przyrodzie okazuje się mieć duży wpływ na dopasowywanie się gatunków do środowiska - podział gatunku na płcie. W rzeczywistości większa część istniejących gatunków, zaczynając od dość prostych (jak owady, czy rośliny), a kończąc na złożonych (takich jak ssaki), do rozmnażania potrzebują dwóch rodziców różniących się konkretnym chromosomem. Różnica ta jest powodem istnienia całego zespołu cech, które pozwalają podzielić osobniki na żeńskie i męskie, a w ogólności na osobniki różnych płci. Mimo, że nie jest to spotykane w naturze, to w ramach eksperymentu myślowego można założyć dowolną liczbę płci, a nie tylko dwie.

Jak zostało pokazane w rozdziale 3, istnieją rozwiązania które nie ignorują podziału populacji na płcie. Aby skutecznie je porównać i zaproponować nowe podejście, zdefiniowano schemat działania heurystki, opisany w algorytmie 4.1 (jego parametry są opisane w tabeli 4.1). Ujmuje on wyżej opisany aspekt biologii organizmów w ramach nowego operatora genetycznego.

Tabela 4.1. Parametry i zmienne użyte w algorytmie 4.1

Parametr	**Znaczenie**
$\|populacja\|$	rozmiar populacji
$kryteriumStopu$	kryterium zakończenia działania heurystyki
$operatorSelekcjiPlciowej$	operator selekcji płciowej (patrz: rozdział 4.3)
cp	prawdopodobieństwo krzyżowania (patrz: rozdział 4.3)
$operatorKrzyzowania$	operator krzyżowania
mp	prawdopodobieństwo mutacji
$operatorMutacji$	operator mutacji
$operatorSelekcjiNaturalnej$	operator selekcji naturalnej (patrz: rozdział 4.1)

Algorytm 4.1 Proponowany schemat działania algorytmu ewolucyjnego

```
1: procedure ALGORYTMEWOLUCYJNY
2:     var populacja ← |populacja| losowych osobników
3:     while nie zachodzi kryteriumStopu do
4:         var zbioryRodzicow ← operatorSelekcjiPlciowej(populacja, cp)
5:         for all rodzice ∈ zbioryRodzicow do
6:             populacja ← populacja ∪ operatorKrzyzowania(rodzice)
7:         end for
8:         var mutanty ← ∅
9:         for all kandydat ∈ populacja do
10:            if zachodzi losowe zdarzenie z prawdopodobieństwem mp then
11:                mutanty ← mutanty ∪ operatorMutacji(kandydat)
12:            end if
13:        end for
14:        populacja ← populacja ∪ mutanty
15:        populacja ← operatorSelekcjiNaturalnej(populacja)
16:    end while
17: end procedure
```

Kryterium stopu, operatory krzyżowania i mutacji oraz prawdopobieństwo mutacji zachowują znaczenie zgodne z przedstawionym w rozdziale 2.

W liniach 2-3 procedura optymalizacji zaczyna się tak, jak w standardowym schemacie - od wygenerowania populacji początkowej i reiteracji (tj. przechodzeniu przez kolejne pokolenia), aż do osiągnięcia kryterium zatrzymania. Znacząca zmiana zachodzi w liniach 4-7, opisujących tworzenie krzyżówek, tj. potomstwa wybranych osobników z populacji. Zamiast, jak dotychczas, wybierać z pewnym prawdopodobieństwem losowych rodziców, kwestię wyboru osobników krzyżujących się przekazujemy do operatora selekcji naturalnej. Do każdego z wybranych zbiorów przykładamy operator krzyżowania, a jego wynik dodajemy do populacji. Linie 8-14 opisują niezmienione podejście do tworzenia osobników zmutowanych. Zbiór mutantów inicjalizujemy na pusty, a następnie z populacji wybieramy z pewnym prawdopodobieństwem losowe osobniki i przykładamy do nich operator mutacji. Jego wynik dodajemy do zbioru mutantów. Po zmutowaniu wybranych osobników zbiór mutantów dołączamy do populacji. Linia 15 jest taka sama jak w standardowym schemacie - opisuje przyłożenie operatora selekcji naturalnej do populacji w celu uzyskania zbioru osobników, które mają przeżyć do kolejnej generacji.

4.1. Operator selekcji naturalnej

Operator selekcji naturalnej jest zbliżony w swojej roli do dotychczasowego operatora selekcji. Ma on symulować przetrwanie osobników między kolejnymi generacjami, z większym prawdopodobieństwem pozwalając na przeżycie jednostkom lepiej przystosowanym. Przyłożenie tego operatora do zbioru osobników będącego sumą poprzedniej populacji oraz zbiorów krzyżówek i mutantów z obecnego pokolenia powinno zwró-

cić populację używaną w kolejnej generacji. W ogólności standardowe implementacje operatora selekcji da się zastosować jako operatory selekcji naturalnej.

4.2. Operator selekcji naturalnej, a operator wyboru

Operator selekcji naturalnej można zaimplementować jako powtórzenie wyboru osobnika ze zbioru (bez zwracania). Komponent odpowiedzialny za pojedyncze wybranie osobnika nazywany jest operatorem wyboru, z nazwą taką samą jak schemat selekcji, np. wybór osobnika przez turniej o rozmiarze 2 nazywa się operatorem wyboru przez turniej o rozmiarze 2.

Poza standardowymi operatorami wyboru takimi jak operator elitarny (wybierający najlepiej dopasowanego osobnika), turniejowy (losujący określoną liczbę osobników i wybierający najlepszego z nich; liczbę porównywanych instancji nazywamy rozmiarem turnieju) lub ruletkowy (losujący osobnika z puli dostępnych z prawdopodobieństwem proporcjonalnym do jego dopasowania lub pozycji w rankingu) możemy mówić również o operatorach takich jak:

- *losowy operator wyboru*
  losujący osobnika z puli z równym prawdopodobieństwem,
- *operator wyboru najbardziej odmiennego osobnika*
  mający sens tylko wtedy, gdy wybraliśmy osobnika referencyjnego. Polega na wybraniu osobnika który jest najbardziej odmienny od osobnika referencyjnego. Stosując ten operator wyboru musimy określić funkcję odległości między osobnikami.

4.3. Operator selekcji płciowej i prawdopodobieństwo krzyżowania

Operator selekcji płciowej jest nowym elementem schematu heurystyki. Jego zadaniem jest symulowanie zachowań osobników prowadzących do dobierania się w pary w celu wydania potomstwa. Osobniki lepiej dopasowane do środowiska powinny mieć większą szansę na zostanie rodzicami niż te dopasowane gorzej. Przyłożenie operatora do populacji z poprzedniego pokolenia powinno zwrócić zestaw par (lub w ogólności zbiorów) osobników-rodziców, z których każda para zostanie dalej przekazana do operatora krzyżowania.

Zmienia się znaczenie prawdopodobieństwa krzyżowania. Nazwa tego parametru zostaje bez zmian, aby nie komplikować nomeklatury, jednak sama wartość nie przekłada się na matematyczne prawdopodobieństwo tego, że losowy osobnik zostanie rodzicem. Zamiast tego może być rozumiana jako stosunek liczby zdarzeń krzyżowania w każdym pokoleniu do rozmiaru populacji. Jest to ściśle związane z liczbą potomków tworzonych w jednej generacji, jednak przez to, że krzyżowanie może skutkować utworzeniem więcej niż jednego potomka, nie należy rozumieć tego parametru jako stosunku liczby krzyżówek do rozmiaru populacji.

Tabela 4.2. Parametry i zmnienne użyte w algorytmie 4.2

Parametr	**Znaczenie**
$kandydaciNaRodzicow$	zbiór osobników spośród których wybieramy rodziców
cp	prawdopodobieństwo krzyżowania
$liczbaRodzicow$	liczba osobników wymagana przez operator krzyżowania
$plecMaZnaczenie$	zmienna logiczna opisana w rozdziale 4.4
$plec(x)$	funkcja zwracająca płeć osobnika jako liczbę z zakresu $[1, liczbaRodzicow]$
$\|populacja\|$	rozmiar populacji
$operatorWyboru[i]$	operator wyboru $i$tej płci

4.4. Schemat implementacji wybranych operatorów selekcji płciowej

Mimo, że dotychczas w literaturze nie był używany termin „operator selekcji płciowej", to istniejące rozwiązania da się zaimplementować w modelu uwzględniającym podział na różne operatory selekcji. W tym celu definiujemy rodzinę operatorów parametryzowanych przez:

- tyle operatorów wyboru ile rodziców jest wymaganych przez operator krzyżowania (zazwyczaj są to 2 operatory), oraz
- zmienną logiczną określającą, czy rodzice muszą być różnej płci; jeśli zmienna przyjmuje wartość „fałsz", to w rzeczywistości płeć jest ignorowana.

Wybór wszystkich zestawów rodziców jest opisany pseudokodem algorytmu 4.2. Jego parametry są zdefiniowane w tabeli 4.2.

Algorytm 4.2 Schemat działania rodziny operatorów selekcji płciowej opisywanej w rozdziale 4.4

1: **function** STANDARDOWASELEKCJAPLCIOWA($kandydaciNaRodzicow, cp$)
2: **var** $zbioryRodzicow \leftarrow \emptyset$
3: **var** $kandydaci \leftarrow$*tablica zbiorów o rozmiarze* $liczbaRodzicow$
4: **for** $i \leftarrow 1$ **to** $liczbaRodzicow$ **do**
5: **if** $plecMaZnaczenie$ **then**
6: $kandydaci[i] \leftarrow \{x \in kandydaciNaRodzicow : plec(x) = i\}$
7: **else**
8: $kandydaci[i] \leftarrow kandydaciNaRodzicow$
9: **end if**
10: **end for**
11: **while** $|zbioryRodzicow| < \lceil cp \times |populacja| \rceil$ **do**
12: **var** $rodzice \leftarrow \bigcup_{i=1}^{liczbaRodzicow} \{operatorWyboru[i](kandydaci[i])\}$
13: $zbioryRodzicow \leftarrow zbioryRodzicow \cup \{rodzice\}$
14: **end while**
15: **return** $zbioryRodzicow$
16: **end function**

W linii 2. inicjalizowana jako pusty zbiór jest zmienna, która zostanie zwrócona w linii 15. W liniach 3-10 zainicjalizowana zostaje tablica zbiorów kandydatów na rodziców, która następnie jest wypełniana w zależności od wartości zmiennej logicznej sterującej podejściem do płci tak, aby na pozycji i znajdował się zbiór osobników nadających się na $i$tego rodzica. Linie 11-14 tworzą odpowiednią liczbę zbiorów rodziców. Każdy z nich powstaje przez przyłożenie odpowiedniego operatoru wyboru do kolejnego elementu tablicy kandydatów.

Jeśli za wszystkie operatory wyboru przyjmiemy operator losowy i będziemy ignorować płeć, to *de facto* sprowadzamy heurystykę do standardowego algorytmu ewolucyjnego, w którym wydanie potomstwa przez osobnika zależy jedynie od operatora selekcji naturalnej (tożsamego z operatorem selekcji z rozdziału 2.). Jeżeli jednak wymusimy różne płci dla rodziców, dalej korzystając z losowych operatorów wyboru, to zrealizujemy metodę GGA [2]. Aby sprowadzić heurystykę do SexualGA [3] należy ignorować płeć osobników, każdego rodzica wybierać innym operatorem wyboru, a wewnątrz operatora selekcji naturalnej używać losowego operatora wyboru.

4.5. Proponowany nowy operator selekcji płciowej

Proponowany operator selekcji płciowej jest inspirowany zachowaniem gatunków, u których niewiele osobników danej płci (tzw. osobników alfa) jest rodzicami większości potomstwa. Wokół tych osobników tworzą się grupy osobników płci przeciwnej, nazywane haremami. Rodzicami pozostałej części potomstwa są osobniki, którym udaje się rozmnożyć kiedy osobniki alfa są zajęte czymś innym. W dalszej części pracy będziemy nazywać je osobnikami beta, jeśli są tej samej płci co osobniki alfa i partnerami, jeśli są odmiennej płci.

Algorytm 4.3 przedstawia schemat działania tego operatora, a tabela 4.3 opisuje jego parametry.

W linii 2. następuje inicjalizacja zmiennej zwracanej w linii 32. Przypisywany jest jej zbiór pusty. W liniach 3-6 populacja osobników jest dzielona względem płci tak, że tablica *perPlec* zawiera na pozycji i kandydatów na rodziców $i$tej płci. Linia 7 inicjalizuje zmienną określającą rozmiar zwracanego zbioru, tj. liczbę generowanych zestawów rodziców. Linie 8 i 9 obliczają liczby kolejno osobników alfa i beta. Wartości te niekoniecznie sumują się do wartości z linii 7, ponieważ są zaokrąglane w górę. Z tego powodu faktyczny rozmiar zwracanego zbioru może być większy niż wcześniej obliczony, jednak jedyną konsekwencją tego może być sprawdzenie nieznacznie większej liczby osobników w każdej populacji (konkretnie *liczbaAlfa+1*). Linie 10-22 opisują pętlę, w której najpierw przy pomocy odpowiedniego operatora wybierany jest osobnik alfa (linia 11), usuwany następnie ze zbioru kandydatów tej płci (aby każdy osobnik alfa był różny oraz aby osobniki alfa nie były brane pod uwagę przy wyborze osobników beta). Kolejne linie (11-20) opisują dobranie partnerów dla osobników alfa, przy użyciu odpowiednich operatorów wyboru. Na końcu ciała pętli nowo utworzony zestaw rodziców jest dołączany do zwracanego zbioru. Linie 22-32 opisują pętlę tworząca rodziców związanych z osobnikami beta, jednak zamiast wybierać osobnika płci alfa (czyli osobnika beta) dla grupy partnerów pozostałych płci, jest on wybierany do każdego zestawu rodziców oddzielnie (z dopuszczeniem powtórzeń).

Algorytm 4.3 Schemat działania haremowego operatora selekcji płciowej (opisywanego w rozdziale 4.5)

```
1: function haremowaSelekcjaPlciowa(kandydaciNaRodzicow, cp)
2:     var zbioryRodzicow ← ∅
3:     var perPlec ←tablica zbiorów o rozmiarze liczbaPlci
4:     for i ← 1 to liczbaPlci do
5:         perPlec[i] ← {x ∈ kandydaciNaRodzicow : plec(x) = i}
6:     end for
7:     var liczbaZbiorowRodzicow ← ⌈cp × |populacja|⌉
8:     var liczbaBeta ← ⌈wspolczynnikBeta × liczbaZbiorowRodzicow⌉
9:     var perAlfa ← ⌈(liczbaZbiorowRodzicow − liczbaBeta)/liczbaAlfa⌉
10:    for i ← 1 to liczbaAlfa do
11:        var alfa ← operatorWyboruAlfa(perPlec[plecAlfa])
12:        perPlec[plecAlfa] ← perPlec[plecAlfa] \ {alfa}
13:        for j ← 1 to perAlfa do
14:            var rodzice ← {alfa}
15:            for k ← 1 to liczbaPlci do
16:                if k ≠ plecAlfa then
17:                    rodzice ← rodzice ∪ {operatorWyboru[k](perPlec[k])}
18:                end if
19:            end for
20:            zbioryRodzicow ← zbioryRodzicow ∪ {rodzice}
21:        end for
22:    end for
23:    for i ← 1 to liczbaBeta do
24:        var beta ← operatorWyboruBeta(perPlec[plecAlfa])
25:        var rodzice ← {beta}
26:        for j ← 1 to liczbaPlci do
27:            if j ≠ plecAlfa then
28:                rodzice ← rodzice ∪ {operatorWyboru[j](perPlec[j])}
29:            end if
30:        end for
31:        zbioryRodzicow ← zbioryRodzicow ∪ {rodzice}
32:    end for
33:    return zbioryRodzicow
34: end function
```

Tabela 4.3. Parametry i zmnienne użyte w algorytmie 4.3

Parametr	**Znaczenie**
$kandydaciNaRodzicow$	zbiór osobników spośród których wybieramy rodziców
cp	prawdopodobieństwo krzyżowania
$liczbaPlci$	liczba płci branych pod uwagę (zazwyczaj 2), równa liczbie rodziców wymaganej przez operator krzyżowania
$plec(x)$	funkcja zwracająca płeć osobnika jako liczbę z zakresu [1, $liczbaRodzicow$]
$\|populacja\|$	rozmiar populacji
$wspolczynnikBeta$	stosunek liczby zdarzeń krzyżowania w których biorą udział osobniki beta, do liczby wszystkich zdarzeń krzyżowania
$liczbaAlfa$	liczba osobników alfa, wokół których tworzone są „haremy”
$operatorWyboruAlfa$	operator wyboru używany do określenia osobników alfa
$plecAlfa$	numer płci z której pochodzą osobniki alfa
$operatorWyboruBeta$	operator wyboru używany do określenia osobników beta
$operatorWyboru[i]$	operator wyboru $i$tej płci

Innymi słowy, w liniach 10-22 wybierane są osobniki alfa i ich haremy, a w liniach 23-32 - osobniki beta i ich partnerzy.

Rozdział 5

Eksperymenty

5.1. Implementacja

5.1.1. Komponenty niezależne od problemu

5.1.2. TSP

działanie operatorów, etc

5.1.3. Knapsack

...

5.1.4. (?)

...

5.2. Procedury eksperymentów

eksploracja z nawrotami/czysty przegląd/inne procedury z etapu 2

5.3. Przeprowadzone eksperymenty

Wyjaśnić strukture

5.3.1. TSP

Initial

Konfiguracja zakresy parametrów, parametry początkowe, ilość nawrotów i powtórzeń

Przebieg Kolejno znajdowane konfiguracje

Wyniki 10 najlepszych max, 10 najlepszych avg, zestawić w tabelki, opisać 1 najlepszy pokazać i opisać przebieg

Tweak

...

Compare

...

SexualGA

... - co tu będzie?

GGA

... - jw?

5.3.2. Knapsack

...

5.3.3. (?)

...

Rozdział 6

Wnioski i spostrzeżenia

Rozdział 7

Dalsze drogi rozwoju

wywalić listy (?), poradzić sobie z podwójną bibliografią (patrz: spis treści)

Spis rysunków

Spis tablic

Spis sygnatur

Bibliografia

[1] Davis L. et al. *Handbook of genetic algorithms*, volume 115. Van Nostrand Reinhold New York, 1991.

[2] Rejeb J., AbuElhaij M. New gender genetic algorithm for solving graph partitioning problems. In *Circuits and Systems, 2000. Proceedings of the 43rd IEEE Midwest Symposium on*, volume 1, pages 444–446 vol.1, 2000.

[3] Wagner S., Affenzeller M. SexualGA: Gender-specific selection for genetic algorithms. In *WMSCI 2005 - The 9th World Multi-Conference on Systemics, Cybernetics and Informatics, Proceedings*, volume 4, pages 76–81, Institute for Formal Models and Verification, Johannes Kepler University, 2005.