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). 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 dużych, czy wręcz nieskończonych zbiorów). Często 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. W zależności od źródła definicje tego pojęcia są różne. W dziedzinie informatyki, a w szczególności obliczeń miękkich i sztucznej inteligencji, termin ten możemy nieformalnie opisać jako algorytm który nie daje gwarancji uzyskania poprawnego wyniku, ale z większym prawdopodobieństwe zwróci wyniki „lepsze". Heurystyki stosowane są wszędzie tam, gdzie nie mamy rzeczywistej możliwości uzyskania wyniku, więc dowolne jego przybliżenie będzie lepsze niż brak jakichkolwiek rezultatów.

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

zasada przetrwania lepiej przystosowanych osobników oraz zjawiska dziedziczenia i mutacji.

Zgodnie z powyższą zasadą, jednostki 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 pozostała część populacji.

Zjawisko dziedziczenia polega na przekazywaniu cech rodziców dzieciom. Zachodzi ono podczas rozmnażania, a więc między dwojgiem rodziców i ich potomstwem. Kod genetyczny potomstwa tworzony jest przez losowe łączenie odpowiednich części kodu genetycznego rodziców, dzięki czemu kolejne pokolenie dzieli ich cechy. W ten sposób niektóre osobniki przejmą od rodziców te cechy, które pozwalały im się dopasować do środowiska. 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, dzięki którym ma on szansę zyskać nowe cechy, co w niektórych przypadkach doprowadzi do lepszego dopasowania. Osobniki z przypadkowymi zmianami, które poprawiają ich dopasowanie mają większe szanse na przeżycie i wydanie potomstwa. Zmiany te więc zostać rozpropagowane wśród osobników przyszłych pokoleń.

Algorytmy ewolucyjne to rodzina heurystyk naśladujących proces ewolucji w celu optymalizacji [2]. Pojedynczy element 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 symulację odpowiednich zjawisk występujących w przyrodzie. Heurystyka polega na wielokrotnym przetworzeniu populacji (czyli zbioru osobników) poprzez zastosowanie każdego z operatorów z pewnym prawdopodobieństwem. W każdym kroku (nazywanym w tym przypadku pokoleniem) do dotychczasowej populacji dołączane są wyniki ich działania tych operatorów, 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. W dostępnej literaturze dotyczącej tematu algorytmów ewolucyjnych rzadko można znaleźć prace, w których uwzględnia się ten aspekt procesu ewolucji (patrz: rozdział 3). Powodem tego jest raczej chęć uproszczenia działania samej heurystyki niż lepsza jakość wyników uzyskiwanych z pominięciem tego aspektu ([3], [5]).

1.2. Cele pracy

Pierwszym celem niniejszej pracy jest opracowanie formalnego opisu algorytmu ewolucyjnego uwzględniającego płeć. Schemat heurystyki jest w pewnym stopniu dowolny, a publikacje z tej dziedziny nie używają jednego wspólnego formalizmu, co utrudnia jednoznaczne porównywanie ich działania. Spełnienie tego celu powinno doprowadzić

do określenia konkretnego schematu, który pozwoli zaimplementować wybrane istniejące rozwiązania oraz zaproponowane rozwiązanie autorskie.

Kolejnym celem jest opis i implementacja wybranych istniejących rozwiązań wykorzystujących płeć w ramach wcześniej opisanego schematu. Poza implementacją tych metod, powinno powstać narzędzie badawcze, które pozwoli na klarowne porównanie ich działania.

Cel trzeci to opracowanie nowego podejścia uwzględniającego płeć. Jest to główny cel tej pracy, mający wnieść coś nowego do dziedziny algorytmów ewolucyjnych.

Ostatnim celem jest zbadanie wcześniej opisanych i zbadanych podejść. Polega to na zebraniu miar jakości działania algorytmów ewolucyjnych dla każdego z nich i porównaniu ich ze sobą.

Rozdział 2

Algorytmy ewolucyjne

Ten rozdział ma na celu opisanie zagadnienia algorytmów ewolucyjnych, zaczynając od ogólnych informacji, stopniowo przechodząc do szczegółów. W pierwszej jego części opisana jest ogólna idea tych heurystyk. Kolejna sekcja skupia się na formalnym opisie poszczególnych pojęć. Po niej znajduje się część w której znajdziemy szczegóły idei opisanej na początku, zapisane z użyciem pojęć przedstawionych w sekcji drugiej. Całość zostanie zamknięta opisem sposobów oceny i analizy działania algorytmów ewolucyjnych.

2.1. Idea algorytmów ewolucyjnych

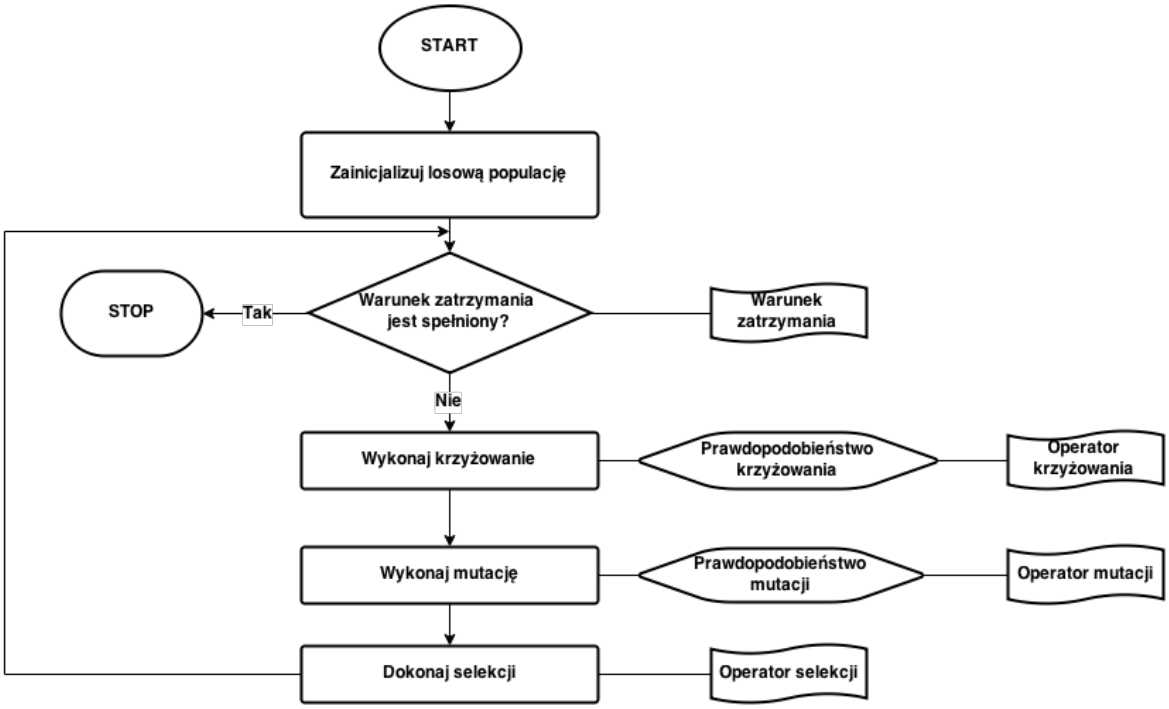
Podstawowym pojęciem używanym w opisywanej heurystyce jest *osobnik*. Jest to abstrakcja pojedynczego rozwiązania, nawiązująca do pojedynczego żywego stworzenia charakteryzującego się pewnymi cechami które wpływają na prawdopodobieństwo jego przeżycia i wydania potomstwa. Pojęciem symulującym wpływ cech 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. Populację możemy też rozumieć jako zbiór próbek z pewnej podprzestrzeni zbioru rozwiązań.

Przez pojęcie funkcji oceny (nazywanej też kryterium) rozumiemy pewne przyporządkowanie (w idealnej sytuacji - funkcję, jednak nie zawsze jest to możliwe; patrz: rozdział 2.2.3) każdemu osobnikowi, czyli rozwiązaniu problemu, jakiejś wartości, na której możemy określić porządek. Taka relacja porządku powinna określać czy dane rozwiązanie jest równie dobre lub lepsze niż inne, czy nie.

Standardową realizacją osobnika jest wektor binarny, który sprawdza się w wielu zastosowaniach i jest prosty w realizacji programowej lub sprzętowej. 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 [1] (np. dla problemów aproksymacji funkcji [4]).

Rys. 2.1. Ogólny schemat działania algorytmów ewolucyjnych



Konwencje dotyczące rysowania schematów blokowych zostały wyjaśnione w podsekcji 2.2.1.

Istnieją implementacje i narzędzia do obliczeń ewolucyjnych które rozróżniają pojęcie genotypu osobnika (czyli wewnętrznej reprezentacji rozwiązania, na której stosujemy operatory genetyczne takie jak krzyżowanie i mutacja) i jego fenotypu (czyli reprezentacji zewnętrznej, 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, ponieważ nie zmienia ono w żadnym stopniu jakości działania heurystyki i nie jest związane z tematem badań.

Na rysunku 2.1 zobrazowany został ogólny schemat działania algorytmów ewolucyjnych. Formalny opis używanych parametrów i operatorów znajduje się w kolejnej sekcji, wraz z definicją takich pojęć jak np. operator.

Współczesna nauka dalej nie jest w stanie dać nam odpowiedzi na pytanie „skąd wzięło się życie", a co za tym idzie, nie jesteśmy w stanie zasymulować momentu rozpoczęcia ewolucji. Zamiast tego, w heurystykach naśladujących ten proces działanie zaczyna się od wygenerowania losowej populacji, czyli zbioru osobników. Symuluje to w uproszczony sposób analizę ewolucji od przypadkowego momentu w czasie istnienia gatunku. Biorąc pod uwagę, że celem heurystyki nie jest idealnie wierne odtworzenie warunków naturalnych, a optymalizacja, nie wydaje się to być problemem.

Mając zbiór początkowych osobników (czyli rozwiązań naszego problemu) rozpoczynamy proces przeszukiwania ich przestrzeni. Dzieje się to iteracyjnie, co znaczy, że pewien zestaw kroków jest wielokrotnie powtarzany. Jedno takie powtórzenie jest nazywane *generacją* lub *pokoleniem*, a składa się na nie kolejno krzyżowanie, mutacja i selekcja naturalna. Moment w którym należy przerwać proces optymalizacji jest

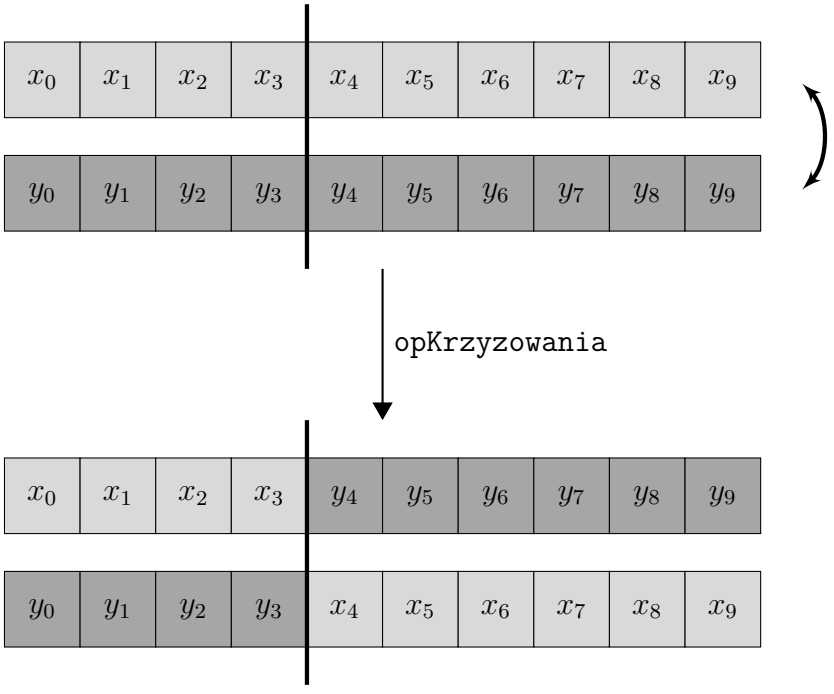
wybierany przez kryterium przerwania. Nie ma to przełożenia na naturę (jako, że ewolucja teoretycznie się nigdy nie kończy) i jest jedynie narzędziem które ma sprawić, że heurystyka nie będzie działać wiecznie, tylko w skończonym czasie zwróci interesujące nas wyniki.

Na pojedyncze pokolenie składają się 3 główne kroki: krzyżowanie, mutacja i selekcja. Mają one symulować odpowiednie zjawiska występujące w przyrodzie - kolejno dziedziczenie, mutację i przetrwanie lepiej dopasowanych osobników. Pierwsze dwa z nich kontrolują tempo przeszukiwania przestrzeni rozwiązań, a ostatnie steruje przeszukiwaniem tak, aby odnajdywać jak najlepsze rozwiązania.

Krzyżowanie odbywa się przy pomocy dedykowanego operatora, z częstością określoną odpowiednim prawdopodobieństwem. Polega ono na wybraniu losowych par osobników nazywanych rodzicami i dołączeniu do populacji nowych rozwiązań (nazywanych potomstwem), o reprezentacji zbliżonej do reprezentacji rodziców. Ma to symulować dziedziczenie cech rodziców przez ich potomków. Zadaniem krzyżowania jest eksploatacja przestrzeni rozwiązań, poprzez zbadanie punktów, które znajdują się pomiędzy znanymi nam już rozwiązaniami. Dzięki temu zagęszczamy próbkowanie podprzestrzeni, którą już zbadaliśmy, czyli dotychczasowej populacji, co ma na celu znalezienie potencjalnych optimów, które mogły zostać przeoczone przez zbyt rzadkie próbkowanie.

Jedną z popularnych realizacji operatora krzyżowania dla osobnika reprezentowanego jako wektor jest wybranie losowej pozycji w wektorach opisujących rodziców (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 przykładowi rodzice. Grubą linią zaznaczono wylosowany punkt przecięcia, wyznaczający podwektory które zostają zamienione. Wektory $(x_0, \ldots, x_3, y_4, \ldots y_9)$ i $(y_0, \ldots, y_3, x_4, \ldots x_9)$ są wtedy traktowane jako potomstwo.

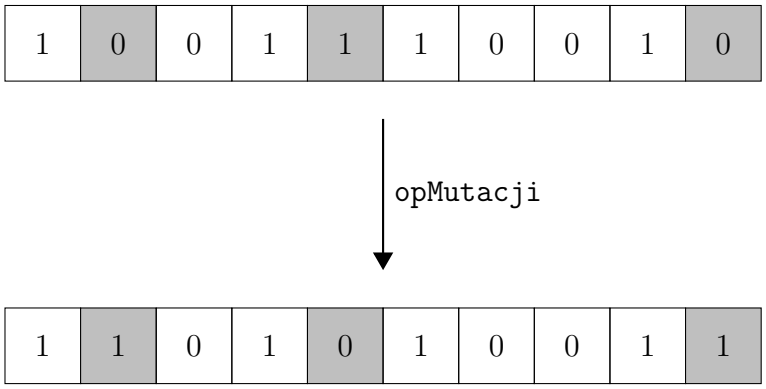
Rys. 2.2. Działanie przykładowej realizacji operatora krzyżowania



Kolejnym krokiem w każdym pokoleniu jest mutacja, czyli wprowadzenie losowych zmian w losowo wybranych osobnikach. Jest to wykonywane przy pomocy odpowiedniego operatora, z prawdopodobieństwem określającym szansę osobnika na zmutowanie. Zmodyfikowane osobniki w zależności od implementacji są dołączane do populacji, albo zastępują niezmodyfikowane rozwiązania. W tej pracy wykorzystano pierwszą z tych opcji, aby nie doprowadzić do sytuacji, w której mutacja pogorszy znane rozwiązanie, które zostanie odrzucone z populacji (ponieważ na jego miejsce wejdzie rozwiązanie zmodyfikowane). Zadaniem tego kroku jest eksploracja przestrzeni rozwiązań, czyli zbadanie rozwiązań spoza dotychczas spróbkowanej podprzestrzeni. W ten sposób heurystyka przeszukuje nowe obszary, które potencjalnie zawierają optimum globalne, a przynajmniej optima lokalne lepsze od dotychczas znalezionych.

Jedną z popularnych realizacji operatora mutacji 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 2.3 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. 2.3. Działanie przykładowej realizacji operatora mutacji



Zmiany zachodzące w jednym osobniku nie powinny być zbyt drastyczne (tak, aby był on zbliżony w reprezentacji do pierwotnego osobnika), ale też nie mogą być zbyt mało znaczące, ponieważ znacząco spowolniłoby to proces eksploracji, a co za tym idzie, obniżyłoby efektywność heurystyki. Gdyby przykładowy operator mutacji (z rysunku 2.3)) operował na wektorach o długości rzędu kilku tysięcy, to zamiana pojedynczego bitu byłaby zbyt mało znacząca, ponieważ rozwiązania przed i po jego zastosowaniu praktycznie by się od siebie nie różniły. Jeśli jednak operator zmieniałby około połowy bitów, to nowe rozwiązanie byłoby zbyt dalekie od pierwotnego, przez co heurystyka sprowadziłaby się do kompletnie losowego przeszukiwania przestrzeni rozwiązań.

W zależności od implementacji oba operatory opisane powyżej mogą być stosowane w różnej kolejności. Ponadto, od realizacji zależy to, czy mutacja jest stosowana jedynie na populacji z poprzedniego pokolenia, czy również na potomstwie. W tej pracy przyjmuje się, że krzyżowanie zostaje wykonane przed mutacją, a jego wyniki również mogą być przez nią modyfikowane. Dzięki temu próbkowanie przestrzeni rozwiązań jest gęstsze, niż gdybyśmy postępowali w inny sposób.

Te dwa kroki mają na celu losowe przeszukiwanie przestrzeni rozwiązań. Gdyby ograniczyć się jedynie do nich, to heurystyka nie dawałaby pożądanych wyników. W celu dojścia do optimów należy tą losowość jakoś ukierunkować. Właśnie w tym celu na końcu każdego pokolenia do całej populacji (a więc osobników z początku pokolenia, potomstwa i osobników zmutowanych) stosuje się operator selekcji naturalnej. Jego zadaniem jest wybranie ze znanych rozwiązań tych, które są lepsze w sensie rozwiązywanego problemu. Zostaną one wykorzystane jako populacja używana w kolejnym pokoleniu. Intuicyjnie, operator ten pełni rolę negatywnych czynników środowiskowych, które sprawiają, że nie wszystkie osobniki danego gatunku przeżywają. Zgodnie z prawami rządzącymi naturą, większe prawdopodobieństwo przeżycia powinny mieć jednostki lepiej dopasowane do swojego środowiska, co w heurystyce przekłada się na lepszą (mniejszą, lub większą, w zależności od rozwiązywanego zadania) funkcję oceny.

Najprostszym operatorem selekcji jest tzw. operator elitarny (nazywany też elitystycznym), którego działanie polega na zwróceniu określonej liczby bezwzględnie najlepszych osobników z dotychczasowej populacji. Mimo, że wydaje się to intuicyjne, to nie jest to rozwiązanie idealne. Praktyka pokazuje, że rozwiązania gorsze często mają cechy, które w dalszych pokoleniach, po kolejnych modyfikacjach, dają rozwiązania lepsze od dotychczasowych. Jest to powodem istnienia takich realizacji jak np. selekcja turniejowa, czy ruletkowa (opisane w rozdziale 5 w ramach opisu implementacji).

Tak sformułowany proces służy do efektywnego przeszukiwania przestrzeni rozwiązań, tymczasem zadanie optymalizacji ma na celu znalezienie jak najlepszego z nich. 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 w trakcie symulacji zapamiętujemy globalnie najlepsze rozwiązanie 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 tym samym momencie w dalszych pokoleniach porównuje się ocenę obecnego globalnego optimum i najlepszego osobnika z danego pokolenia (optimum lokalnego). Jeśli optimum globalne jest gorsze niż lokalne, to zostaje nim zastąpione.

2.2. Formalny opis operatorów i parametrów

Niniejsza sekcja jest przeznaczona na formalny opis elementów algorytmu ewolucyjnego. Jak zostało zaznaczone w poprzedniej sekcji, sam schemat działania heurystyki zależy od implementacji, jednak pokazane tu definicje i sygnatury powinny być zgodne z większością istniejących rozwiązań. Każde z opisywanych pojęć niesie ze sobą nie tylko znaczenie i zastosowanie wzorowane na przyrodzie, ale również ograniczenia nałożone na realizację danego elementu, również wynikające z natury. Wszystkie te aspekty zostaną podane i zapisane w formie, która ma zapobiec występowaniu nieścisłości.

W pierwszej podsekcji zajduje się zestaw konwencji związanych z diagramami, czcionkami, itd. używanymi w tej pracy. Normy te obowiązują w całym dokumencie, również w diagramach z poprzednich sekcji i rozdziałów.

Druga podsekcja opisuje pojęcie operatora, które jest często stosowane przy formalnych opisach elementu heurystyki. Dodatkowo, znajdziemy tam definicję operatora $random(X)$, który jest bardzo przydatny w dalszych opisach.

Kolejne pięć podsekcji skupia się już na konkretnych częściach heurystyki, które należy określić w celu jej wykorzystania.

2.2.1. Konwencje

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

Na rysunku 2.4 zobrazowane zostały różne elementy schematów blokowych używane w tej pracy, opisane w tabeli 2.2.

Na rysunku 2.5 przedstawiono konwencje dotyczące rysowania wykresów przyjęte w tym dokumencie.

Ponadto, w sygnaturach operatorów i funkcji będą często wykorzystywane zbiory o konkretnym rozmiarze. Jako, że nie istnieje a przynajmniej ja nie znam powszechnie przyjęta konwencja zapisu takiego faktu, w niniejszej pracy przyjęto, że zapis $A^{\{n\}}$ oznacza klasę zbiorów $n$elementowych, w której każdy element należy do zbioru A. Zapis A^n, zgodnie ze standardową konwencją oznacza $n$krotny iloczyn kartezjański zbioru A, czyli klasę wektorów o długości n, których składowe pochodzą ze zbioru A.

$$A^{\{n\}} \equiv \{a : a \subset A, |a| = n\} \tag{2.1}$$
$$A^{\{0\}} \equiv \emptyset \tag{2.2}$$
$$A^n \equiv \underbrace{A \times A \times \ldots \times A}_{n} \tag{2.3}$$
$$A^0 \equiv \emptyset \tag{2.4}$$
$$A \times B \equiv \{(a, b) : a \in A, b \in B\} \tag{2.5}$$

2.2.2. 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 ograni-

Tabela 2.1. Konwencje dotyczące czcionek

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

Rys. 2.4. Konwencje dotyczące elementów schematów blokowych

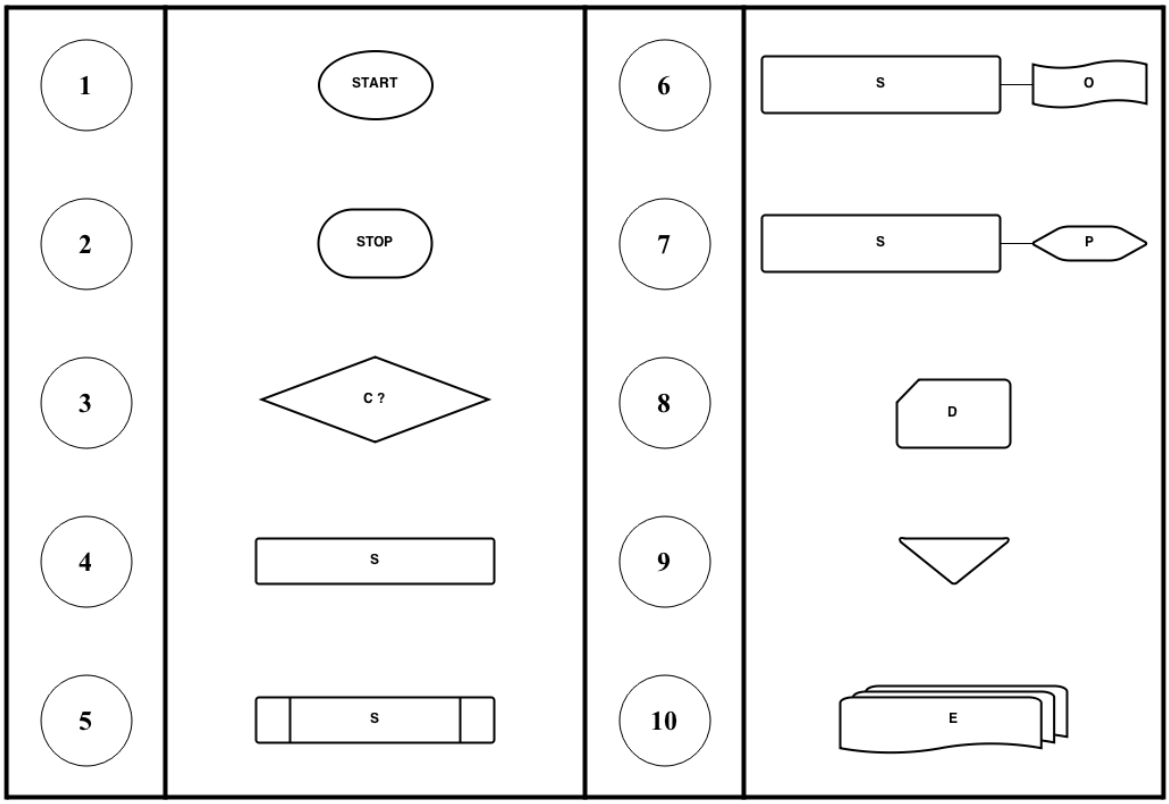
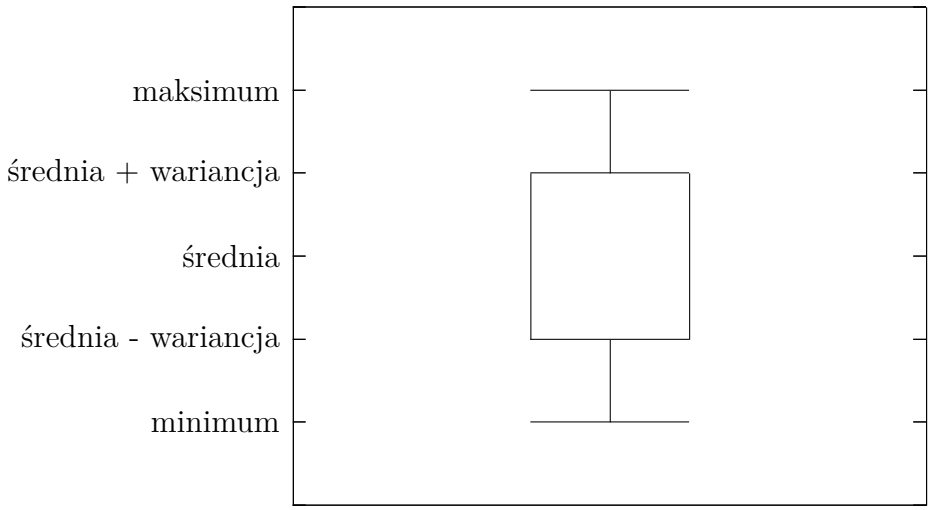


Tabela 2.2. Opis konwencji dotyczących elementów schematów blokowych

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	Kombinacja symboli oznaczająca, że podczas wykonywania kroku S używany jest operator O.
⑦	Kombinacja symboli oznaczająca, 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. 2.5. Konwencje dotyczące rysowania wykresów



czenie nie musi być spełnione dla operatora. Innymi słowy pojęcie operatora pokrywa 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.

Ponadto, w algorytmach znajdujących się w tej pracy używany jest operator $random(X)$ opisany sygnaturą 1, zwracający losowy element zbioru X (z rozkładem równomiernym).

Sygnatura 1 Operator $random(S)$

$$random : A^{\{n\}} \rightarrow A \tag{2.6}$$
$$X \leftarrow \{x_0, x_1, \ldots x_{n-1}\} \tag{2.7}$$
$$\forall_{x_i, x_j \in X} P(random(X) = x_i) = P(random(X) = x_j) \tag{2.8}$$

A to dowolny zbiór, a n jego rozmiar.

2.2.3. Osobnik, populacja i ocena

Podstawowym pojęciem w dziedzinie algorytmów ewolucyjnych jest osobnik, czyli rozwiązanie. Definiujemy go jako element przestrzeni rozwiązań, którą również musimy zdefiniować. Na te pojęcia nie ma nałożonych żadnych ograniczeń, jednak od ich realizacji zależy prostota i efektywność pozostałych pojęć, opisanych w dalszych podsekcjach.

Ważniejszym pojęciem jest funkcja oceny. Wbrew nazwie nie musi być to funkcja w rozumieniu matematycznym, a operator (patrz: rozdział 2.2.2) [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.

Sygnatura 2 Osobnik

$$osobnik \in \mathcal{S} \tag{2.9}$$

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

2.2.4. Operator mutacji

Dziedziną tego operatora jest przestrzeń rozwiązań, a przeciwdziedziną - mkrotny iloczyn kartezjański przestrzeni rozwiązań. Oznacza to, że jednokrotne zastosowanie

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.

Sygnatura 3 Funkcja oceny

$$\texttt{funkcjaOceny} : \mathcal{S} \rightarrow \mathcal{E} \tag{2.10}$$
$$\exists \mathbf{R}_{\prec} \subset \mathcal{E} \times \mathcal{E} \tag{2.11}$$
$$\mathbf{R}_{\lhd} \leftarrow \{(x, y) : (\texttt{funkcjaOceny}(x), \texttt{funkcjaOceny}(y)) \in \mathbf{R}_{\prec}\} \tag{2.12}$$

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.

operatora mutacji daje w wyniku m osobników powstałych przez modyfikację osobnika pierwotnego. Każdy ze zwróconych osobników powinien być nieznacznie różny od argumentu operatora.

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 4 Operator mutacji

$$\texttt{opMutacji} : \mathcal{S} \rightarrow \mathcal{S}^{\{\mathsf{m}\}} \tag{2.13}$$
$$\texttt{prawdMutacji} \in \langle 0, 1 \rangle \tag{2.14}$$

m to liczba zwracanych wyników mutacji.

2.2.5. Operator krzyżowania

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

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.

2.2.6. Operator selekcji

Zadaniem **operatora selekcji** jest symulacja zjawiska przeżycia silniejszych (czyli lepiej dopasowanych) osobników. Jest on stosowany pod koniec każdego pokolenia w celu usunięcia z niej osobników gorzej dopasowanych, poprzez wykorzystanie populacji

Sygnatura 5 Operator krzyżowania

$$\texttt{opKrzyzowania} : \mathcal{S}^{\{\overline{\texttt{c}}\}} \rightarrow \mathcal{S}^{\{\underline{\texttt{c}}\}} \tag{2.15}$$

$$\texttt{prawdKrzyzowania} \in \langle 0, 1 \rangle \tag{2.16}$$

$\overline{\texttt{c}}$ to liczba rodziców, a $\underline{\texttt{c}}$ to liczba potomków.

wyjściowej jako używanej w kolejnym pokoleniu. Zmniejsza on różnorodność genetyczną w obrębie populacji, co w ogólności jest negatywnym efektem, ponieważ zahamowywuje proces eksplorację przestrzeni rozwiązań. Operator mutacji powinien eksplorować ją na tyle efektywnie, żeby operator selekcji odrzucał mało obiecujące kierunki eksploracji zamiast całkiem zatrzymywać jej proces.

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 2.19-2.21 sygnatury 6, 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 2.22-2.25 sygnatury 6. 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.

Sygnatura 6 Operator selekcji

$$\texttt{opSelekcji} : \mathcal{S}^{\{p\}} \rightarrow \mathcal{S}^{\{\texttt{rozmiarPopulacji}\}} \tag{2.17}$$

$$\texttt{rozmiarPopulacji} \in \mathbb{N}_+ \tag{2.18}$$

$$p \approx (1 + \texttt{prawdMutacji} \times \texttt{m} \tag{2.19}$$

$$+ \texttt{prawdKrzyzowania} \times \underline{\texttt{c}}) \tag{2.20}$$

$$\times \texttt{rozmiarPopulacji} \tag{2.21}$$

$$T \leftarrow \{t_0, t_1, \ldots t_{\texttt{rozmiarPopulacji}-1}\} \tag{2.22}$$

$$s \in \texttt{opSelekcji}(T) \Rightarrow s \in T \tag{2.23}$$

$$P(i) \leftarrow P(t_i \in \texttt{opSelekcji}(T)) \tag{2.24}$$

$$P(i) < P(j) \Leftrightarrow (t_i, t_j) \in \mathbf{R}_{\lhd} \tag{2.25}$$

2.2.7. 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 7 przedstawiono ogólny zapis operatora (z dowolną dziedziną, linia 2.26) i sygnaturę jego przykładowej realizacji (opartej o stałą liczbę pokoleń, linie 2.27-2.28).

Sygnatura 7 Warunek zatrzymania i jego przykładowa realizacja

$$\texttt{warunekStopu} : \ldots \rightarrow \{0, 1\} \tag{2.26}$$

$$\texttt{warunekStopu} : \mathbb{N} \rightarrow \{0, 1\} \tag{2.27}$$

$$\texttt{warunekStopu}(g) = 1 \Leftrightarrow g \leqslant \overline{\texttt{g}} \tag{2.28}$$

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

2.3. Szczegóły działania heurystyki

Ogólny schemat działania algorytmu ewolucyjnego został przedstawiony w sekcji 2.1. Kolejne kroki przedstawionego tam diagramu zostaną opisane ze szczegółami w kolejnych akapitach.

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 (linia 6), którego wynik jest dołączany do zbioru *noweOsobniki* (linia 7). Schemat ten przewiduje, że operator krzyżowania przyjmuje 2 osobniki jako argument (tzn. $\overline{\texttt{c}} = 2$, patrz: rozdział 2.2.5), przez co wartość, z którą porównujemy losowo wybraną liczbę z przedziału $\langle 0, 1\rangle$ to $\texttt{prawdKrzyzowania}/2$. W

Algorytm 2.1 Szczegółowy schemat działania kroku "Wykonaj krzyżowanie"

```
    Dostępne zmienne:
        populacja                    ▷ Obecna populacja, tj. zbiór osobników
    Parametry:
        prawdKrzyzowania             ▷ Prawd. krzyżowania
        opKrzyzowania                ▷ Operator krzyżowania
 1: procedure CROSSINGOVER
 2:     var noweOsobniki ← ∅
 3:     for all osobnik ∈ populacja do
 4:         if random([0, 1]) ⩽ prawdKrzyzowania/2 then
 5:             var partner ← random(populacja \ {osobnik})
 6:             var potomek ← opKrzyzowania(osobnik, partner)
 7:             noweOsobniki ← noweOsobniki ∪ {potomek}
 8:         end if
 9:     end for
10:     populacja ← populacja ∪ noweOsobniki
11: end procedure
```

ogólności, dla dowolnego $\bar{c}$ wartość ta wynosi $\texttt{prawdKrzyzowania}/\bar{c}$, a zamiast jednego partnera należy wybrać ich $\bar{c} - 1$.

Algorytm 2.2 Szczegółowy schemat działania kroku "Wykonaj mutację"

```
    Dostępne zmienne:
        populacja                    ▷ Obecna populacja, tj. zbiór osobników
    Parametry:
        prawdMutacji                 ▷ Prawd. mutacji
        opMutacji                    ▷ Operator mutacji
 1: procedure MUTATION
 2:     var noweOsobniki ← ∅
 3:     for all osobnik ∈ populacja do
 4:         if random([0, 1]) ⩽ prawdMutacji then
 5:             var mutant ← opMutacji(osobnik)
 6:             noweOsobniki ← noweOsobniki ∪ {mutant}
 7:         end if
 8:     end for
 9:     populacja ← populacja ∪ noweOsobniki
10: end procedure
```

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 (linia 5), którego wynik zostaje dołączony do zbioru *noweOsobniki* (linia 6).

Algorytm 2.3 Szczegółowy schemat działania kroku "Dokonaj selekcji naturalnej"

Dostępne zmienne:
populacja ▷ Obecna populacja, tj. zbiór osobników
Parametry:
`rozmiarPopulacji` ▷ Rozmiar populacji
`opSelekcji` ▷ Operator selekcji

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

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 2).

2.4. 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 podsekcjach opisane zostaną sposoby analizy jakości działania pojedynczego przebiegu heurystyki, jak i wielu przebiegów o tych samych parametrach.

2.4.1. 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").

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

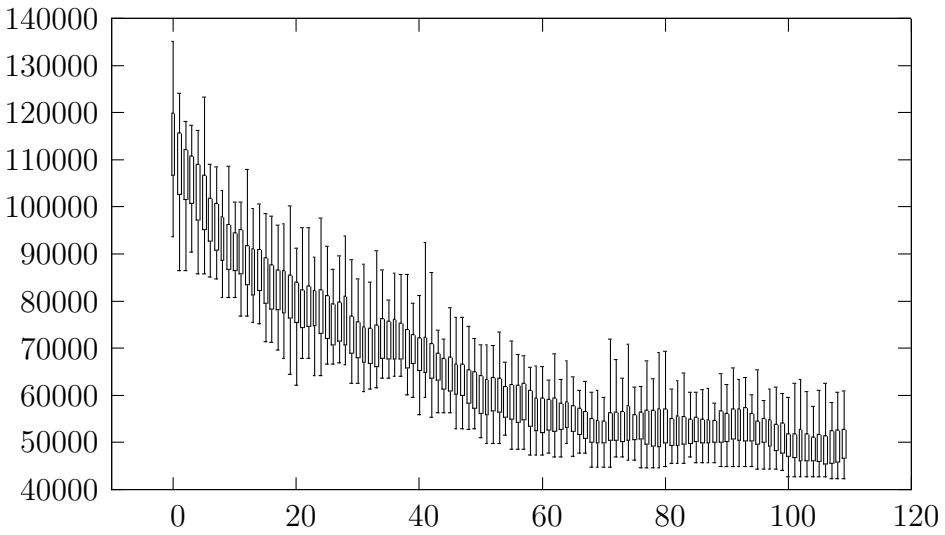
Podstawowe statystyki używane w tym celu, to m.in.:
- najlepsza i najgorsza ocena osobnika z populacji,
- średnia i odchylenie standardowe (lub wariancja [2]) oceny osobników w populacji,
- mediana i kwantyle oceny osobników w populacji.

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 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).

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.

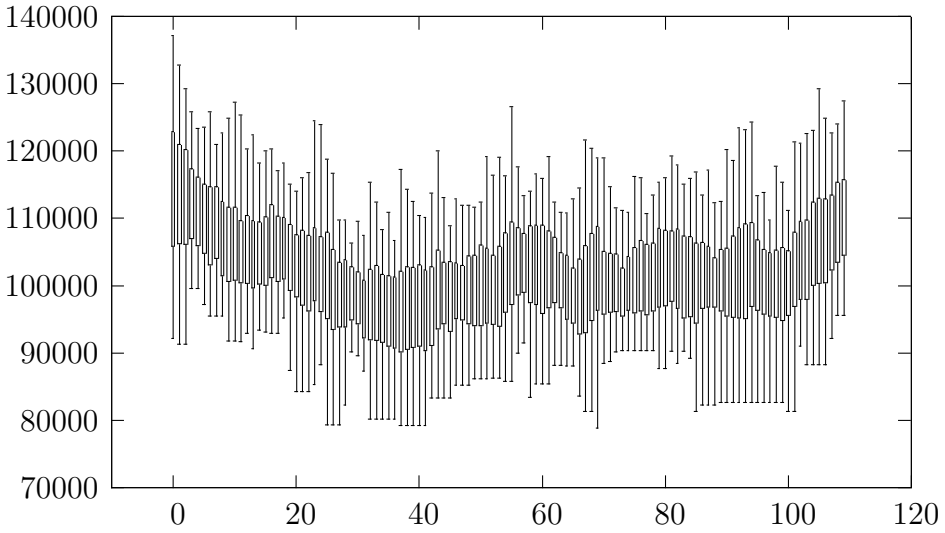
W tej pracy do analizy wykorzystano tylko 4 z wyżej wymienionych statystyk: średnią, wariancję[3], minimum i maksimum. Są one w pełni wystarczające do analizy porównawczej między uruchomieniami.

Rys. 2.6. Przykładowy wykres przebiegu



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.

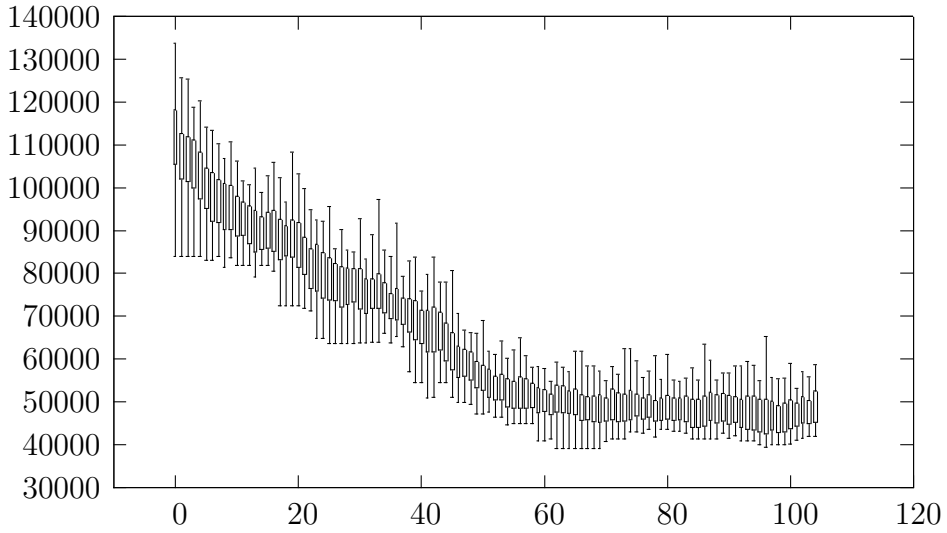
Rys. 2.7. Wykres przebiegu, w którym globalne optimum nie znajduje się w ostatniej populacji



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

Na rysunku 2.7 przedstawiony jest wykres przebiegu na którym możemy zaobserwować sytuację opisaną na końcu rozdziału 2.1, tzn. taką, w której optimum globalne jest znalezione w innym pokoleniu niż ostatnie.

Rys. 2.8. Wykres przebiegu w którym obserwujemy stagnację



Na rysunku 2.8 przedstawiono sytuację, w której obserwujemy tzw. stagnację (wspominaną już w rozdziale 2.2.7). Możemy zaobserwować, że od pewnego momentu (około sześćdziesiątej generacji) kolejne pokolenia nie przynoszą znaczącej zmiany wyniku, co mogłoby być powodem do przerwania działania heurystyki.

2.4.2. 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

Dotychczasowe implementacje algorytmów ewolucyjnych zazwyczaj (choć nie zawsze [3], [5]) 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.

Przy opisywaniu operatorów wprowadzanych w tym rozdziale, używana będzie funkcja $plec(osobnik)$, przypisująca osobnikowi jego płeć.

Sygnatura 8 Funkcja $plec(osobnik)$

$$plec : \mathcal{S} \rightarrow \mathcal{G} \tag{4.1}$$

$$\mathcal{G} \neq \emptyset \tag{4.2}$$

$\mathcal{G}$ to zbiór możliwych płci.
W niniejszej pracy przyjęto co najwyżej 2 płcie, ze zbiorem $\mathcal{G}$ określonym jako $\mathcal{G} = \{♀, ♂\}$

4.1. Heurystyka DSEA

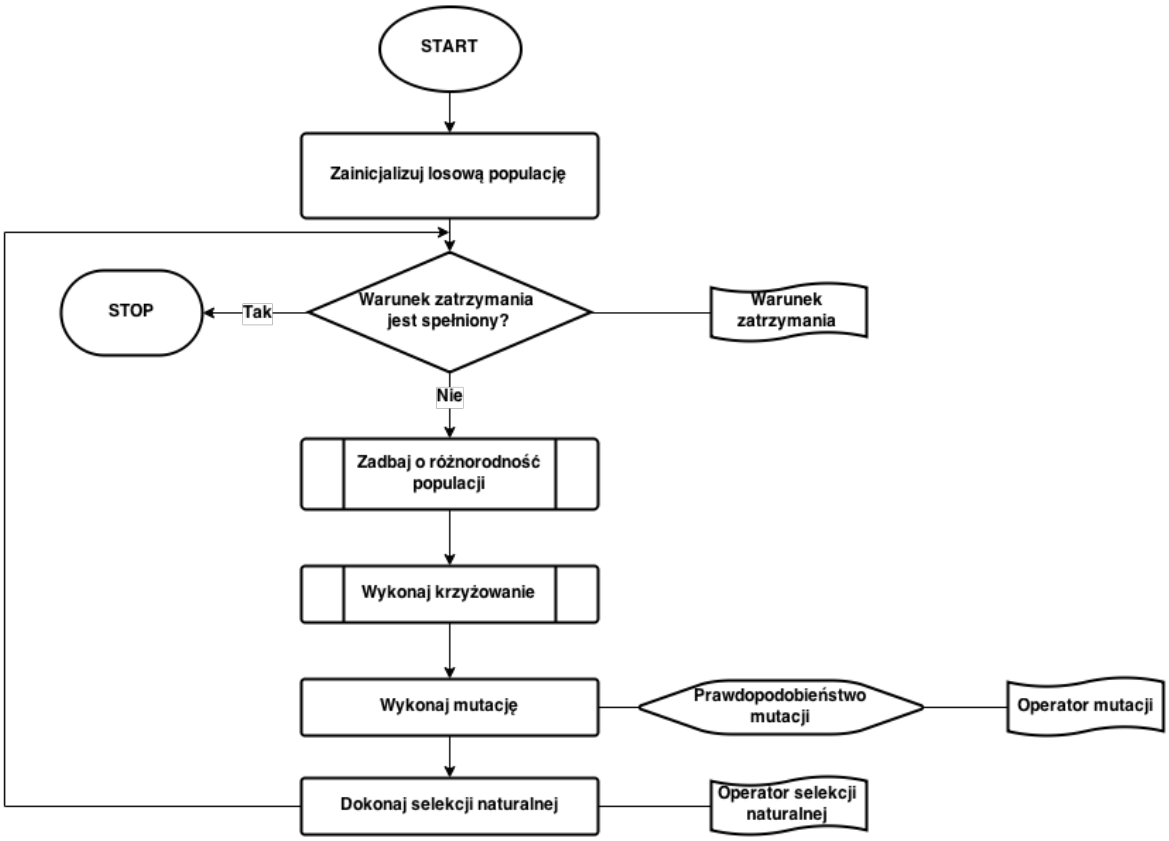
Jak zostało pokazane w rozdziale 3, istnieją rozwiązania które nie ignorują podziału populacji na płcie. Aby skutecznie je porównać i zaproponować nowe podejście, zdefiniowano schemat działania heurystki, opisany schematem blokowym znajdującym się na rysunku 4.1. Ujmuje on wyżej opisany aspekt biologii organizmów w ramach nowego operatora selekcji. W dalszej części pracy tak zdefiniowaną heurystykę nazywać

będziemy **algorytmem ewolucyjnym o podwójnej selekcji** (ang. *double selection evolutionary algorithm*) i odnosić się do niej przy pomocy skrótu **DSEA**, będącego akronimem nazwy angielskiej.

4.1.1. Działanie

Na rysunku 4.1 przedstawiono schemat działania heurystyki DSEA w postaci schematu blokowego.

Rys. 4.1. Schemat działania heurystyki DSEA



W większej części diagram zgadza się z diagramem 2.1 przedstawionym w sekcji 2.1. Istotne różnice między diagramami to zmiana działania kroku "Wykonaj krzyżowanie oraz dodanie nowego kroku Źadbaj o różnorodność populacjiźaraz przed nim. Zostały one zdefiniowane w paragrafach kolejno 4.1.1 i i 4.3. Ponadto, zmieniła się nazwa operatora używanego w kroku "Dokonaj selekcji naturalnej", co zostanie opisane w podsekcji 4.1.2.

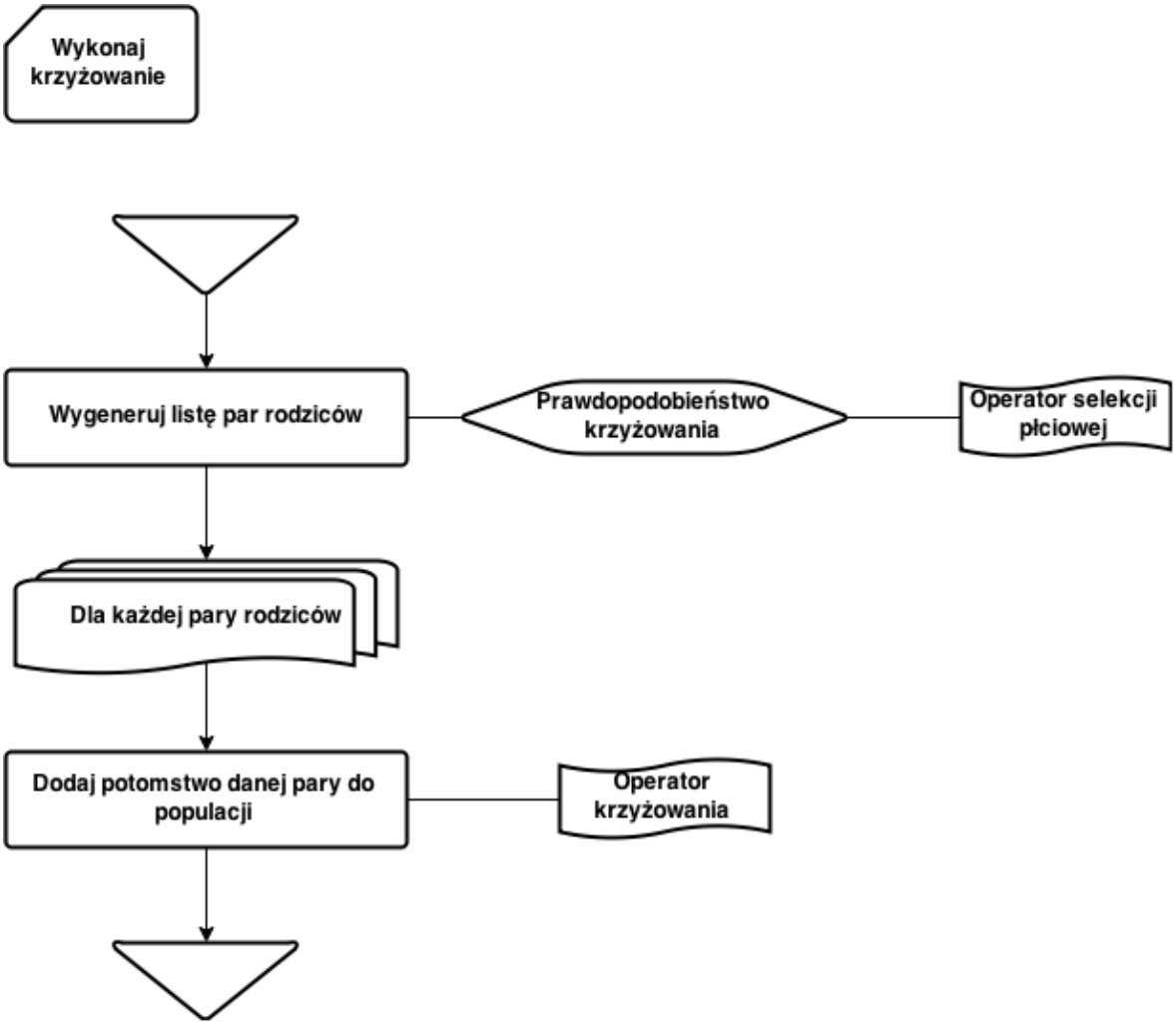
Krok „Wykonaj krzyżowanie"

Na rysunku 4.2 znajduje się diagram przedstawiający krok "Wykonaj krzyżowanie"jako sekwencję mniejszych kroków. Za pomocą operatora selekcji płciowej[1] gene-

1. W podsekcji 4.1.3 opisane zostanie znaczenie i sygnatura tego operatora, wyjaśnione jest uproszczenie polegające na zwracaniu par rodziców oraz różnica w interpretacji parametru `prawdKrzyzowania`.

rowana jest lista par rodziców. Następnie, dla każdej z nich wykonuje się krzyżowanie, traktując oba osobniki z pary jako argumenty operatora krzyżowania, a jego wynik dołączając do populacji. W przeciwieństwie do realizacji opisanej w sekcji 2.3 przy pomocy algorytmu 2.1 wyniki operatora krzyżowania nie są dołączane do osobnego zbioru, a od razu do całości populacji. Dzieje się tak, ponieważ zadanie wyboru rodziców jest oddelegowane do operatora, który zawsze zostanie użyty przed operatorem krzyżowania.

Rys. 4.2. Szczegóły działania kroku „Wykonaj krzyżowanie”

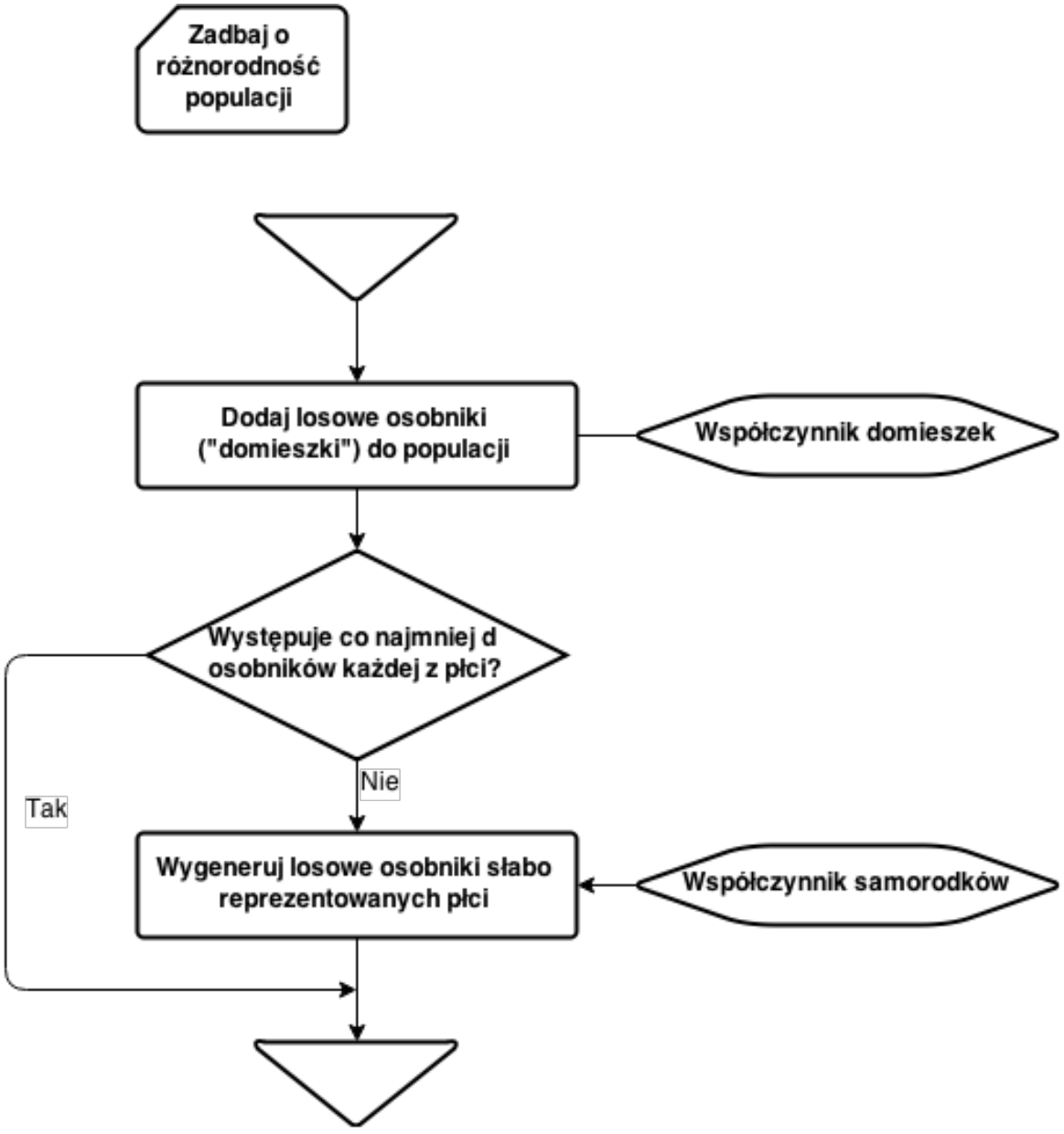


Krok „Zadbaj o różnorodność populacji”

Wprowadzenie podziału populacji na płcie powinno skutkować odmiennym traktowaniem osobników w zależności od tej cechy. Co za tym idzie, obie podgrupy powinny różnić się tym bardziej, im więcej generacji zostało przeprowadzonych. Efekt taki ma zarówno wady, jak i zalety. Dzięki niemu potomstwo (czyli wynik operatora krzyżowania) w danej populacji powinno również zachować większą różnorodność. Negatywną stroną tego zjawiska jest fakt, że operator selekcji naturalnej, który nie powinien brać pod uwagę płci, może zaburzyć stosunek liczności osobników danych płci względem siebie. W skrajnych sytuacjach może dojść do tego, że w całej populacji zabraknie

osobników którejś z płci, co uniemożliwi dalsze działanie heurystyki (ponieważ niemożliwe byłoby zastosowanie operatora krzyżowania). Ponadto, im dysproporcje między różnymi płciami będą większe, tym mniejsza będzie różnorodność całej populacji, co szybko prowadzi do stagnacji.

Rys. 4.3. Szczegóły działania kroku „Zadbaj o różnorodność populacji"



Na rysunku 4.3 przedstawiono krok mający ograniczyć negatywne efekty podziału na płcie, opisane wyżej.

W pierwszej jego części do populacji dodajemy tzw. „domieszki" (ang. *mixins*), czyli losowe osobniki (tworzone w ten sam sposób co populacja początkowa). Ma to na celu regularne uzupełnianie puli genów obecnych w populacji, na wypadek gdyby geny odpowiedzialne za pozytywną cechę zanikły z powodu losowości operatora selekcji naturalnej lub płciowej. Ilość domieszek jest kontrolowana przez współczynnik domieszek

(`wspDomieszek`), będący stosunkiem pożądanej ilości nowych osobników do wartości `rozmiarPopulacji`.

Druga część kroku ma zapobiec sytuacjom, w których któraś płeć nie występuje w populacji, lub jest zbyt słabo reprezentowana. Jeśli liczność którejś z płci w populacji spadnie poniżej arbitralnie dobranej liczby `d`, to do populacji dołączane jest $\lceil$`wspSamorodków` $\times$ `rozmiarPopulacji`$\rceil$ (lub d, jeśli d jest większe) losowych osobników tej płci[2]. W praktyce, w ramach tej pracy użyto arbitralnie dobranych wartości `d` $= 5$ i `wspSamorodków` $= 0.05$.

4.1.2. Operator selekcji naturalnej

Heurystyka DSEA, jak nazwa wskazuje, korzysta z dwóch operatorów selekcji, które należy rózróżnić. Z tego powodu dotychczasowy operator selekcji zmienił nazwę na **operator selekcji naturalnej**. Zachowuje on to samo znaczenie i zastosowanie, opisane w rozdziale 2.2.6.

Dodatkowo, wymaga się, aby prawdopodobieństwo znalezienia się osobnika w zbiorze zwracanym przez ten operator było niezależne od płci osobnika. Zostało to zapisane w linii 4.12 sygnatury 9, korzystając z twierdzenia zgodnie z którym prawdopodobieństwo łączne dwóch zdarzeń jest równe iloczynowi ich prawdopodobieństw wtedy i tylko wtedy, gdy są to zdarzenia niezależne.

Sygnatura 9 Operator selekcji naturalnej

$$\texttt{opSelNat} : \mathcal{S}^{\{p\}} \rightarrow \mathcal{S}^{\{\texttt{rozmiarPopulacji}\}} \tag{4.3}$$

$$\texttt{rozmiarPopulacji} \in \mathbb{N}_+ \tag{4.4}$$

$$p \approx (1 + \texttt{prawdMutacji} \times \texttt{m} \tag{4.5}$$

$$+ \texttt{prawdKrzyzowania} \times \underline{\texttt{c}}) \tag{4.6}$$

$$\times \texttt{rozmiarPopulacji} \tag{4.7}$$

$$T \leftarrow \{t_0, t_1, \ldots t_{\texttt{rozmiarPopulacji}-1}\} \tag{4.8}$$

$$s \in \texttt{opSelNat}(T) \Leftrightarrow \exists_{t \in T} t \equiv s \tag{4.9}$$

$$P(i) \leftarrow P(t_i \in \texttt{opSelNat}(T)) \tag{4.10}$$

$$\forall_{i,j=0,1,\ldots,\texttt{rozmiarPopulacji}-1} P(i) < P(j) \Leftrightarrow (t_i, t_j) \in \mathbf{R}_{\lhd} \tag{4.11}$$

$$\forall_{i=0,1,\ldots,\texttt{rozmiarPopulacji}-1, g \in \mathcal{G}} P(i \wedge plec(t) = g) = P(i) \times P(plec(t_i) = g) \tag{4.12}$$

Standardowo korzysta się z operatorów selekcji naturalnej które odpowiednią ilość razy powtarzają wybór pojedynczego osobnika (bez powtórzeń) z populacji wejściowej, zbierając w ten sposób populację wyjściową. O ile sam sposób wyboru różni się między realizacjami operatora, to kolejne osobniki dołączane do populacji wyjściowej są zazwyczaj wybierane przy pomocy tego samego podejścia, które dalej będzie nazywane operatorem wyboru. Został on zdefiniowany w sygnaturze 10. Rzecz jasna istnieją realizacje, które korzystają z wielu operatorów wyboru, ale nie są one tematem tej pracy i nie będą w niej poruszane.

2. Alternatywnym rozwiązaniem mogłaby być zmiana płci losowych osobników pozostałych płci.

Sygnatura 10 Operator wyboru

$$\texttt{opWyboru} : \mathcal{S}^{\{p\}} \rightarrow \mathcal{S} \tag{4.13}$$

Realizacje operatorów wyboru nazywamy zgodnie z nazwami operatorów selekcji które z nich korzystają. Przykładowo, turniejowy operator selekcji korzysta z turniejowego operatora wyboru. Operatory wykorzystane w tej pracy zostały opisane w odnieść się do eksperymentów.

Wydzielenie abstrakcji wyboru jednego rozwiązania pozwala na zapisanie uogólnionego schematu działania większości popularnie stosowanych operatorów selekcji. Jest on opisany w algorytmie 4.1.

Algorytm 4.1 Schemat działania operatora selekcji naturalnej korzystającego z operatora wyboru

Wejście:
populacja ▷ Populacja wyjściowa
Wyjście:
wynik ▷ Populacja wyjściowa
Parametry:
`rozmiarPopulacji` ▷ Rozmiar populacji
`opWyboru` ▷ Operator wyboru

```
1: operator opSelNat(populacja)
2:     var wynik ← ∅
3:     while |wynik| < rozmiarPopulacji do
4:         wynik ← wynik ∪ {opWyboru(populacja \ wynik)}
5:     end while
6:     return wynik
7: end operator
```

Działanie takiego operatora zaczyna się od inicjalizacji zmiennej *wynik*, zwracanej jako rezultat (linia 6). Początkowa jej wartość to zbiór pusty (linia 2), jednak w kolejnym kroku rozpoczyna się pętla, powtarzana tak długo, aż zbiór ten będzie miał odpowiedni rozmiar (linia 3). W ciele tej pętli za pomocą operatora wyboru `opWyboru` dokonuje się selekcji jednego nie wybranego jeszcze rozwiązania i dołącza sie je do zbioru *wynik* (linia 4).

4.1.3. Operator selekcji płciowej

W przeciwieństwie do tego jak działają klasyczne algorytmy ewolucyjne, w przyrodzie fakt dobierania się osobników w pary w celu wydania potomstwa nie jest losowy. W zależności od gatunku i sytuacji środowiskowej przedstawiciele jednej lub obu płci muszą przekonać swoich przyszłych partnerów o tym, że mają cechy, które przekazane potomstwu dałyby mu większą szansę na przetrwanie. Sposoby na to są różne - samce różnych gatunków rywalizują o samice poprzez walkę, okrzyki i śpiew, taniec, itd. U innych gatunków to samice zabiegają o względy partnerów. Ponadto, różne gatunki

stosują różne strategie wiązania się w pary. Niektóre z nich starają się krzyżować tak często jak to możliwe z możliwie dużą liczbą partnerów, co powoduje dużą liczbę potomstwa, z których „odsiewana" jest słabo przystosowana do środowiska większość. Inne wiążą się w pary na całe życie (jak np. niektóre gatunki ptaków), albo przynajmniej na jakiś czas, dłuższy niż jedno pokolenie.

Operator selekcji płciowej to nowy element schematu heurystyki, który ma symulować te zjawiska i strategie. W ogólności osobniki lepiej dopasowane do środowiska powinny mieć większą szansę na zostanie rodzicami niż te dopasowane gorzej. Zastosowanie tego operatora do populacji z poprzedniego pokolenia powinno zwrócić zestaw par osobników-rodziców, z których każda para zostanie dalej przekazana do operatora krzyżowania.

W naturze, podobnie jak w podejściach porównywanych w tej pracy, rodziców jest dwoje. W sztucznym środowisku symulacyjnym liczba ta może być w gruncie rzeczy dowolna. Oznacza to, że wspomniane wyżej pary rodziców mogą być zbiorami o dowolnym rozmiarze.

Dotychczasowe algorytmy ewolucyjne można traktować jako DSEA z losowym operatorem selekcji płciowej.

Zmienia się znaczenie prawdopodobieństwa krzyżowania. Nazwa tego parametru zostaje bez zmian, aby nie komplikować nazewnictwa, 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.

Sygnatura 11 Operator selekcji płciowej

$$\texttt{opSelPlciowej} : \mathcal{S}^{\{\texttt{rozmiarPopulacji}\}} \to \left(\mathcal{S}^{\{\overline{\texttt{c}}\}}\right)^{\{q\}} \tag{4.14}$$

$$q = \texttt{prawdKrzyzowania} \times \texttt{rozmiarPopulacji} \tag{4.15}$$

$$T \leftarrow \{t_0, t_1, \ldots t_{\texttt{rozmiarPopulacji}-1}\} \tag{4.16}$$

$$\exists_{P \in \texttt{opSelekcji}(T)} t \in P \Rightarrow t \in T \tag{4.17}$$

$\overline{\texttt{c}}$ to ilość osobników potrzebna do wykonania krzyżowania (patrz: sygnatura 5, rozdział 2.2.5).
Operator zwraca zbiór q zbiorów, z których każdy zawiera $\overline{\texttt{c}}$ osobników.

4.1.4. Realizacja wybranych rozwiązań literaturowych (?) w modelu DSEA

troche prozy, wspólna realizacja W rozdziale 3 przedstawiono przegląd prac naukowych o tematyce zgodnej z tematem tej pracy, tj. z uwzględnianiem zjawiska płci w algorytmach ewolucyjnych. Spośród przedstawionych tam pozycji wybrano dwie, które prezentowały nieco odmienne podejścia do zagadnienia: GGA [3], które opierało się o wymuszenie różnych płci rodziców, wybierając ich losowo, oraz SexualGA [5],

w którym nie wprowadzano rozróżnienia osobników na płcie, ale każdego z rodziców wybierano w inny sposób, tj. za pomocą innego operatora wyboru.

Oba te podejścia można opisać przy pomocy tego samego schematu, jednak odmiennie parametryzowanego dla każdej z tych metod. Algorytm 4.2 opisuje jego działanie.

Algorytm 4.2 Schemat działania wybranych rozwiązań literaturowych

Wejście:
populacja ▷ Populacja wyjściowa
Wyjście:
rodzice ▷ Zbiór par rodziców
Parametry:
`opWyboru1` ▷ Operator wyboru pierwszego z rodziców
`opWyboru2` ▷ Operator wyboru drugiego z rodziców
`plecMaZnaczenie` ▷ Zmienna logiczna określająca, czy rodzice muszą różnić się płcią
`prawdKrzyzowania` ▷ Prawdopodobieństwo krzyżowania

1: **operator** `opSelPlciowej`(*populacja*)
2: **var** $rodzice \leftarrow \emptyset$
3: **var** $kandydaci1 \leftarrow \begin{cases} \{s : plec(s) = ♀, s \in populacja\} & : \texttt{plecMaZnaczenie} = \top \\ populacja & : \texttt{plecMaZnaczenie} = \bot \end{cases}$
4: **var** $kandydaci2 \leftarrow \begin{cases} \{s : plec(s) = ♂, s \in populacja\} & : \texttt{plecMaZnaczenie} = \top \\ populacja & : \texttt{plecMaZnaczenie} = \bot \end{cases}$
5: **while** $|rodzice| < \texttt{rozmiarPopulacji} \times \texttt{prawdKrzyzowania}]$ **do**
6: **var** $rodzic1 \leftarrow opWyboru1(kandydaci1)$
7: **var** $rodzic2 \leftarrow opWyboru1(kandydaci1 \setminus \{rodzic1\})$
8: $rodzice \leftarrow rodzice \cup \{\{rodzic1, rodzic2\}\}$
9: **end while**
10: **return** *rodzice*
11: **end operator**

Symbole $\top$ i $\bot$ oznaczają kolejno prawdę i fałsz.

GGA

Działanie, realizacja genSel, tuning innych elementów, żeby pasowało

SexualGA

Działanie, realizacja genSel, tuning innych elementów, żeby pasowało

4.2. Proponowany operator selekcji płciowej

Motywacja, działanie

4.2.1. Realizacja

algorytm

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] Cramer N. L. A representation for the adaptive generation of simple sequential programs. In *Proceedings of the First International Conference on Genetic Algorithms*, pages 183–187, 1985.

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

[3] 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.

[4] Streeter M., Becker L. Automated discovery of numerical approximation formulae via genetic programming. *Genetic Programming and Evolvable Machines*, 4(3):255–286, 2003.

[5] 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.