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

Spis treści

Streszczenie

Abstrakt

Abstract

Abstrakt

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.

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 [5]. 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 3. W dostępnej literaturze dotyczącej tematu algorytmów ewolucyjnych rzadko można znaleźć prace, w których uwzględnia się ten aspekt procesu ewolucji (patrz: rozdział 2). Powodem tego jest raczej chęć uproszczenia działania samej heurystyki niż lepsza jakość wyników uzyskiwanych z pominięciem tego aspektu ([7], [9]).

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 i implementacja 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 1

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.

1.1. Idea algorytmów ewolucyjnych

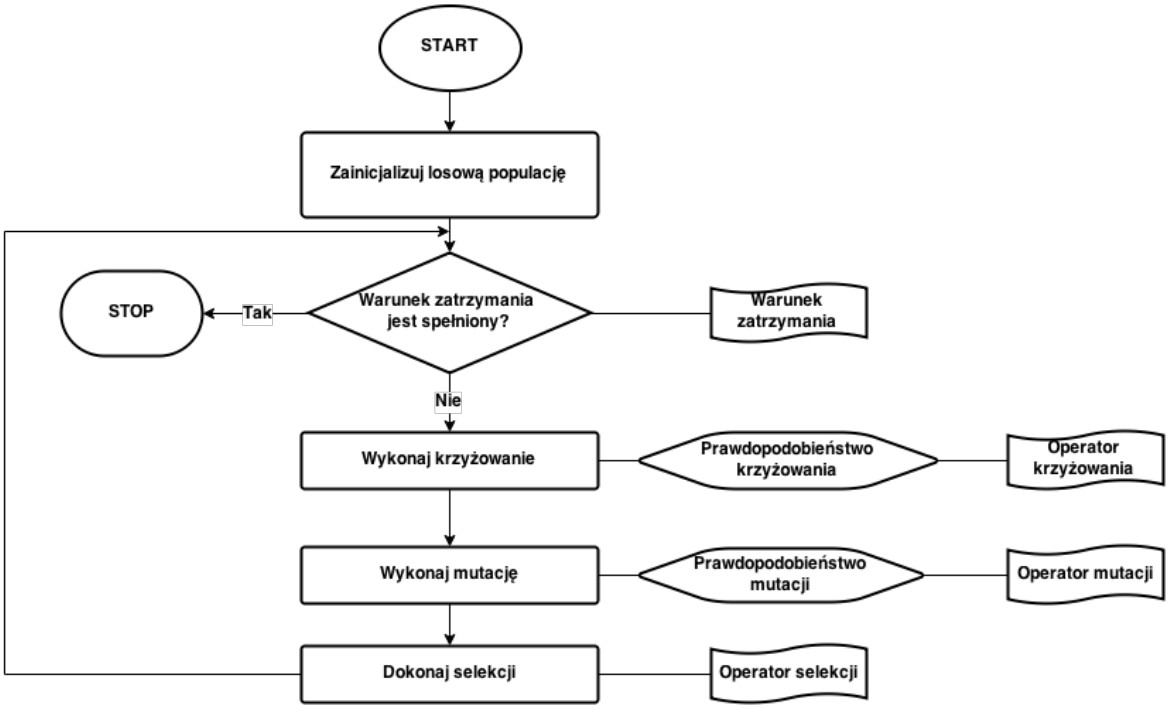
Podstawowym pojęciem używanym w opisywanej heurystyce jest *osobnik*. Jest to abstrakcja pojedynczego rozwiązania, nawiązująca do pojedynczego żywego stworzenia charakteryzującego się pewnymi cechami które wpływają na prawdopodobieństwo jego przeżycia i wydania potomstwa. Pojęciem symulującym wpływ cech 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 1.2.3) 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 [4] (np. dla problemów

Rys. 1.1. Ogólny schemat działania algorytmów ewolucyjnych



Konwencje dotyczące rysowania schematów blokowych zostały wyjaśnione w podsekcji 1.2.1.

aproksymacji funkcji, przedstawionych jako zadanie minimalizacji błędu przybliżenia [8]).

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 1.1 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 nigdy się 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.

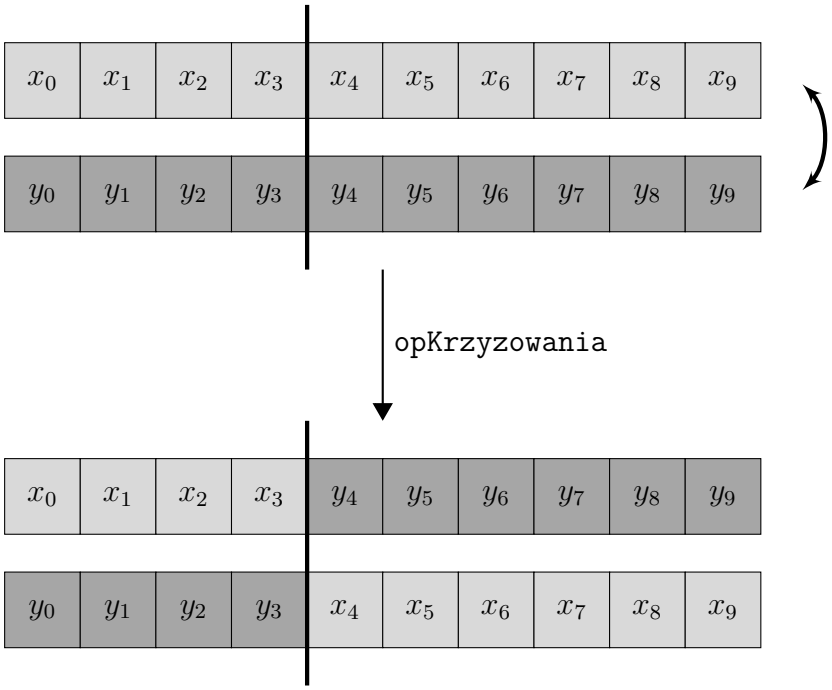
Jedną z popularnych realizacji operatora krzyżowania dla osobnika reprezentowanego jako wektor jest wybranie losowej pozycji w wektorach opisujących rodziców (tzw. punktu przecięcia) i zwrócenie 2 osobników powstałych przez zamianę podwektorów od wylosowanej pozycji wzwyż. Na rysunku 1.2 zobrazowano 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.

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 1.3 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 $(1, \mathbf{1}, 0, 1, \mathbf{0}, 1, 0, 0, 1, \mathbf{1})$.

Zmiany zachodzące w jednym osobniku nie powinny być zbyt drastyczne (tak, aby

Rys. 1.2. Działanie przykładowej realizacji operatora krzyżowania

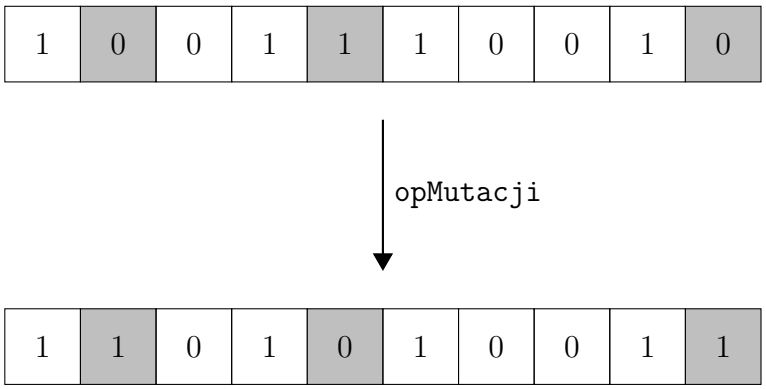


był on zbliżony w reprezentacji do pierwotnego osobnika), ale też nie mogą być zbyt mało znaczące, ponieważ znacząco spowolniłoby to proces eksploracji, a co za tym idzie, obniżyłoby efektywność heurystyki. Gdyby przykładowy operator mutacji (z rysunku 1.3)) operował na wektorach o długości rzędu kilku 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. Jest to powód, dla którego 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

Rys. 1.3. Działanie przykładowej realizacji operatora mutacji



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 4 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[1], 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.

1.2. 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

1. W tym kontekście optima globalne i lokalne oznaczają najlepsze dotychczas znalezione rozwiązanie i najlepsze rozwiązanie w danej populacji.

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 poprzednich sekcjach i rozdziałach.

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.

1.2.1. Konwencje

W tabeli 1.1 zostały pokazane konwencje dotyczące znaczenia czcionek używane w tej pracy.

Na rysunku 1.4 zobrazowane zostały różne elementy schematów blokowych używane w tej pracy, opisane w tabeli 1.2.

Na rysunku 1.5 przedstawiono konwencje dotyczące rysowania wykresów przyjęte w tym dokumencie.

Ponadto, w sygnaturach operatorów i funkcji będą często wykorzystywane klasy zbiorów o konkretnym rozmiarze. Jako, że nie istnieje powszechnie przyjęta konwencja zapisu takiego bytu, 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.

$$A^{\{n\}} \equiv \{a : a \subset A, |a| = n\} \tag{1.1}$$

$$A^{\{0\}} \equiv \emptyset \tag{1.2}$$

$$A^n \equiv \underbrace{A \times A \times \ldots \times A}_{n} \tag{1.3}$$

$$A^0 \equiv \emptyset \tag{1.4}$$

$$A \times B \equiv \{(a, b) : a \in A, b \in B\} \tag{1.5}$$

Tabela 1.1. 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. 1.4. Konwencje dotyczące elementów schematów blokowych

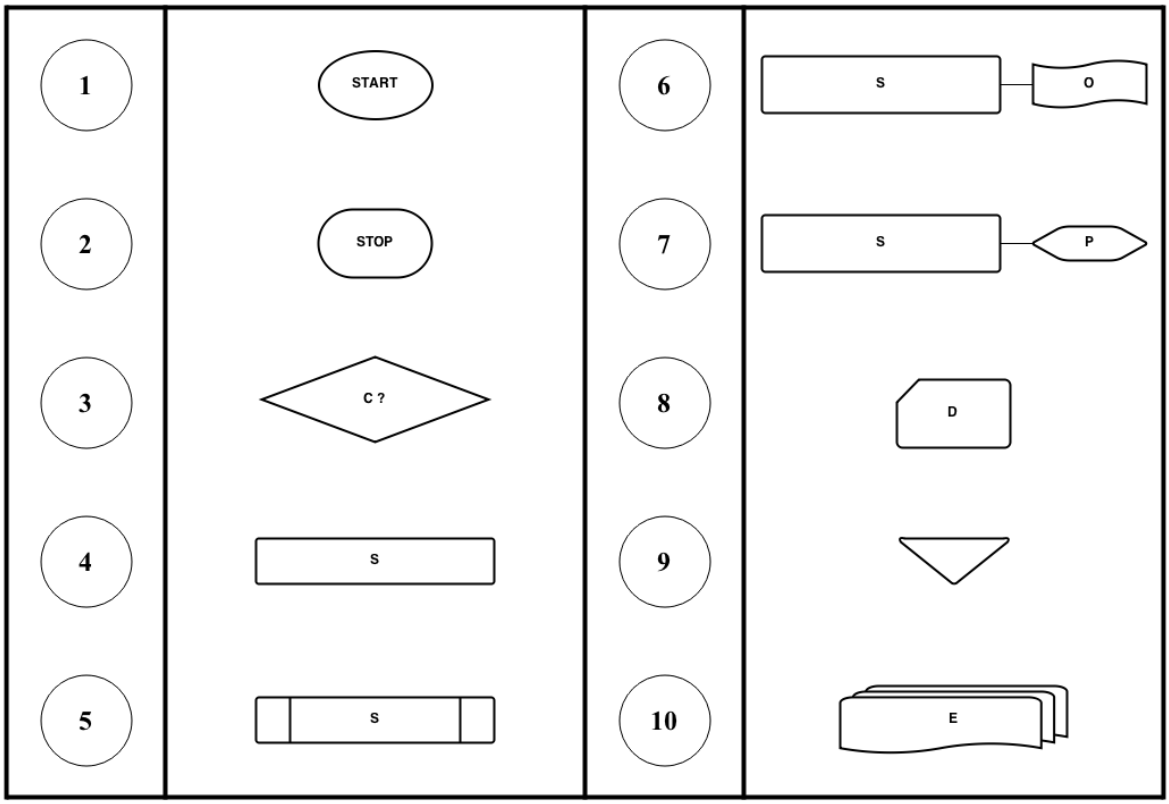
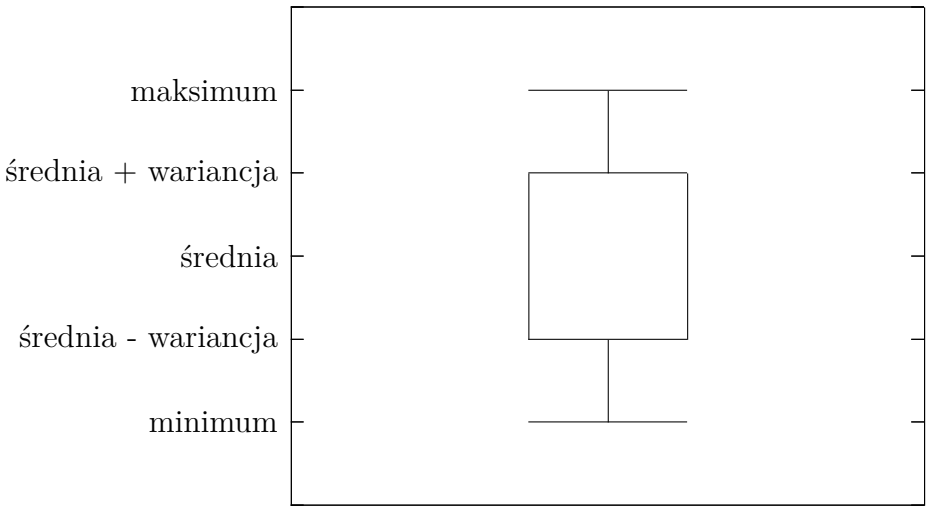


Tabela 1.2. Opis konwencji dotyczących elementów schematów blokowych

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 3).
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. 1.5. Konwencje dotyczące rysowania wykresów



1.2.2. 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 ograniczenie nie musi być spełnione dla operatora. Innymi słowy pojęcie operatora pokrywa się z pojęciem funkcji z imperatywnych języków programowania, ale nie jest tożsame z funkcją w rozumieniu matematycznym.

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

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).

Definicja 1 Operator $random(S)$

$$random : A^{\{n\}} \rightarrow A \tag{1.6}$$
$$X \leftarrow \{x_0, x_1, \ldots x_{n-1}\} \tag{1.7}$$
$$\forall_{x_i, x_j \in X} P(random(X) = x_i) = P(random(X) = x_j) \tag{1.8}$$

A to dowolny zbiór, a n jego rozmiar.

1.2.3. 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 łatwość realizacji i efektywność działania 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 1.2.2) [2]. 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.

Definicja 2 Osobnik

$$osobnik \in \mathcal{S} \tag{1.9}$$

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

2. 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.

Definicja 3 Funkcja oceny

$$\texttt{funkcjaOceny} : \mathcal{S} \rightarrow \mathcal{E} \tag{1.10}$$

$$\exists \mathbf{R}_{\prec} \subset \mathcal{E} \times \mathcal{E} \tag{1.11}$$

$$\mathbf{R}_{\lhd} \leftarrow \{(x, y) : (\texttt{funkcjaOceny}(x), \texttt{funkcjaOceny}(y)) \in \mathbf{R}_{\prec}\} \tag{1.12}$$

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.

1.2.4. Operator mutacji

Dziedziną tego operatora jest przestrzeń rozwiązań, a przeciwdziedziną - klasa $\mathsf{m}$elementowych podzbiorów przestrzeni rozwiązań. Oznacza to, że jednokrotne zastosowanie operatora mutacji daje w wyniku m osobników powstałych przez modyfikację osobnika pierwotnego. Każdy ze zwróconych osobników powinien być nieznacznie różny od argumentu operatora. Zazwyczaj przyjmuje się $\mathsf{m} = 1$

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.

Definicja 4 Operator mutacji

$$\texttt{opMutacji} : \mathcal{S} \rightarrow \mathcal{S}^{\{\mathsf{m}\}} \tag{1.13}$$

$$\texttt{prawdMutacji} \in \langle 0, 1 \rangle \tag{1.14}$$

m to liczba zwracanych wyników mutacji.

1.2.5. Operator krzyżowania

Dziedziną tego operatora jest klasa $\overline{\mathsf{c}}$elementowych podzbiorów przestrzeni rozwiązań, a przeciwdziedziną - klasa $\underline{\mathsf{c}}$elementowych pozdbiorów 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.

Definicja 5 Operator krzyżowania

$$\texttt{opKrzyzowania} : \mathcal{S}^{\{\overline{\texttt{c}}\}} \rightarrow \mathcal{S}^{\{\underline{\texttt{c}}\}} \tag{1.15}$$

$$\texttt{prawdKrzyzowania} \in \langle 0, 1 \rangle \tag{1.16}$$

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

1.2.6. 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 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 klasa $p$elementowych pozdbiorów 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 1.19-1.21 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 1.22-1.25 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.

1.2.7. 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 przed-

Definicja 6 Operator selekcji

$$\texttt{opSelekcji} : \mathcal{S}^{\{p\}} \rightarrow \mathcal{S}^{\{\texttt{rozmiarPopulacji}\}} \tag{1.17}$$
$$\texttt{rozmiarPopulacji} \in \mathbb{N}_+ \tag{1.18}$$
$$p \approx (1 + \texttt{prawdMutacji} \times \texttt{m} \tag{1.19}$$
$$+ \texttt{prawdKrzyzowania} \times \underline{\texttt{c}}) \tag{1.20}$$
$$\times \texttt{rozmiarPopulacji} \tag{1.21}$$
$$T \leftarrow \{t_0, t_1, \ldots t_{\texttt{rozmiarPopulacji}-1}\} \tag{1.22}$$
$$s \in \texttt{opSelekcji}(T) \Rightarrow s \in T \tag{1.23}$$
$$P(i) \leftarrow P(t_i \in \texttt{opSelekcji}(T)) \tag{1.24}$$
$$P(i) < P(j) \Leftrightarrow (t_i, t_j) \in \mathbf{R}_{\lhd} \tag{1.25}$$

stawiono ogólny zapis operatora (z dowolną dziedziną, linia 1.26) i sygnaturę jego przykładowej realizacji (opartej o stałą liczbę pokoleń, linie 1.27-1.28).

Definicja 7 Warunek zatrzymania i jego przykładowa realizacja

$$\texttt{warunekStopu} : \ldots \rightarrow \{0, 1\} \tag{1.26}$$

$$\texttt{warunekStopu} : \mathbb{N} \rightarrow \{0, 1\} \tag{1.27}$$
$$\texttt{warunekStopu}(g) = 1 \Leftrightarrow g \leqslant \overline{\texttt{g}} \tag{1.28}$$

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.

1.3. Szczegóły działania heurystyki

Ogólny schemat działania algorytmu ewolucyjnego został przedstawiony w sekcji 1.1. 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 1.1 opisuje szczegóły kroku "Wykonaj krzyżowanie". Krok ten rozpoczyna się od inicjalizacji zmiennej *noweOsobniki* na pusty zbiór (w linii 1), który w linii 9. zostanie dołączony do ogólnej populacji. Dla każdego osobnika w populacji (linia 2) sprawdzane jest, czy należy wykonać krzyżowanie z jego udziałem (linia 3). Jeżeli tak, to losowo dobierany jest jego partner (linia 4), a następnie na tak określonej parze

Algorytm 1.1 Szczegółowy schemat działania kroku "Wykonaj krzyżowanie"

Dostępne zmienne:
populacja ▷ Obecna populacja, tj. zbiór osobników
Parametry:
`prawdKrzyzowania` ▷ Prawdopodobieństwo krzyżowania
`opKrzyzowania` ▷ Operator krzyżowania

```
0: procedure WYKONAJKRZYZOWANIE
1:     var noweOsobniki ← ∅
2:     for all osobnik ∈ populacja do
3:         if random([0, 1]) ⩽ prawdKrzyzowania/2 then
4:             var partner ← random(populacja \ {osobnik})
5:             var potomek ← opKrzyzowania({osobnik, partner})
6:             noweOsobniki ← noweOsobniki ∪ {potomek}
7:         end if
8:     end for
9:     populacja ← populacja ∪ noweOsobniki
10: end procedure
```

osobników stosowany jest operator krzyżowania (linia 5), którego wynik jest dołączany do zbioru *noweOsobniki* (linia 6). Schemat ten przewiduje, że operator krzyżowania przyjmuje 2 osobniki jako argument (tzn. $\bar{c} = 2$, patrz: rozdział 1.2.5), przez co wartość, z którą porównujemy losowo wybraną liczbę z przedziału $\langle 0, 1\rangle$ to $\texttt{prawdKrzyzowania}/2$. W ogólności, dla dowolnego $\bar{c}$ wartość ta wynosi $\texttt{prawdKrzyzowania}/\bar{c}$, a zamiast jednego partnera należy wybrać ich $\bar{c} - 1$.

Algorytm 1.2 Szczegółowy schemat działania kroku "Wykonaj mutację"

Dostępne zmienne:
populacja ▷ Obecna populacja, tj. zbiór osobników
Parametry:
`prawdMutacji` ▷ Prawdopodobieństwo mutacji
`opMutacji` ▷ Operator mutacji

```
0: procedure WYKONAJMUTACJE
1:     var noweOsobniki ← ∅
2:     for all osobnik ∈ populacja do
3:         if random([0, 1]) ⩽ prawdMutacji then
4:             var mutant ← opMutacji(osobnik)
5:             noweOsobniki ← noweOsobniki ∪ {mutant}
6:         end if
7:     end for
8:     populacja ← populacja ∪ noweOsobniki
9: end procedure
```

Algorytm 1.2 opisuje szczegóły kroku "Wykonaj mutację". Krok rozpoczyna się od inicjalizacji zmiennej *noweOsobniki* na pusty zbiór (w linii 1), który w linii 8 zostanie dołączony do ogólnej populacji. Dla każdego osobnika w populacji (linia 2) sprawdzane jest, czy należy wykonać na nim mutację (linia 3), a jeśli tak, to stosowany jest do niego

operator mutacji (linia 4), którego wynik zostaje dołączony do zbioru *noweOsobniki* (linia 5).

Algorytm 1.3 Szczegółowy schemat działania kroku "Dokonaj selekcji naturalnej"

```
   Dostępne zmienne:
      populacja                ▷ Obecna populacja, tj. zbiór osobników
   Parametry:
      rozmiarPopulacji                          ▷ Rozmiar populacji
      opSelekcji                                ▷ Operator selekcji
0: procedure DOKONAJSELEKCJINATURALNEJ
1:    populacja ← opSelekcji_rozmiarPopulacji(populacja)
2: end procedure
```

Algorytm 1.3 opisuje szczegóły kroku "Dokonaj selekcji naturalnej", który *de facto* sprowadza się do zastąpienia dotychczasowej populacji wynikiem zastosowania operatora selekcji na niej (linia 1).

1.4. 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.

1.4.1. 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 [3]) oceny osobników w populacji,
- mediana i kwantyle oceny osobników w populacji.

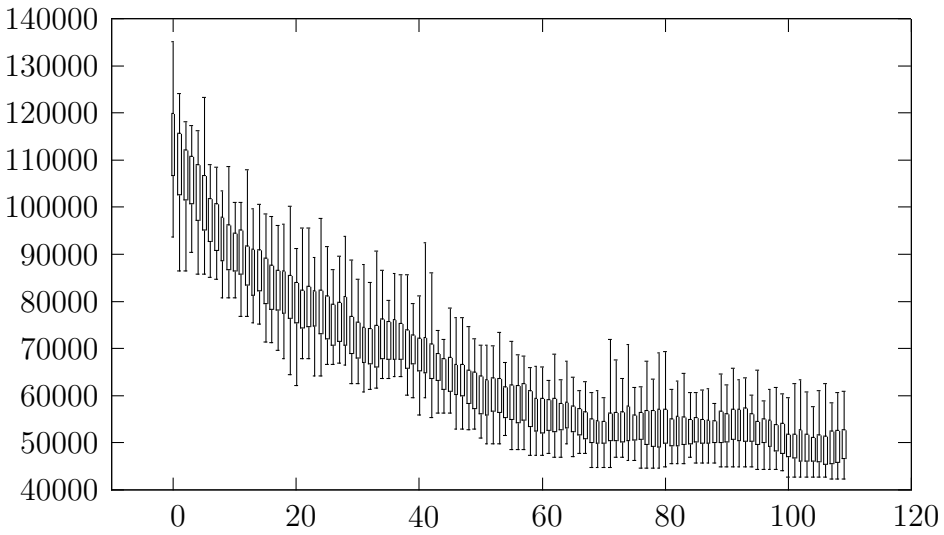
Wartości te można wygodnie zobrazować na wykresie pudełkowym, na osi odciętych umieszczając numery pokoleń, a na osi rzędnych - wartości odpowiednich statystyk. Takie zobrazowanie przebiegu heurystyki pozwala na wyciągnięcie wniosków i stosowne dopasowanie parametrów heurystyki (np. jeśli w problemie minimalizacji wszystkie

3. 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.

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[4], minimum i maksimum. Są one w pełni wystarczające do analizy porównawczej między uruchomieniami.

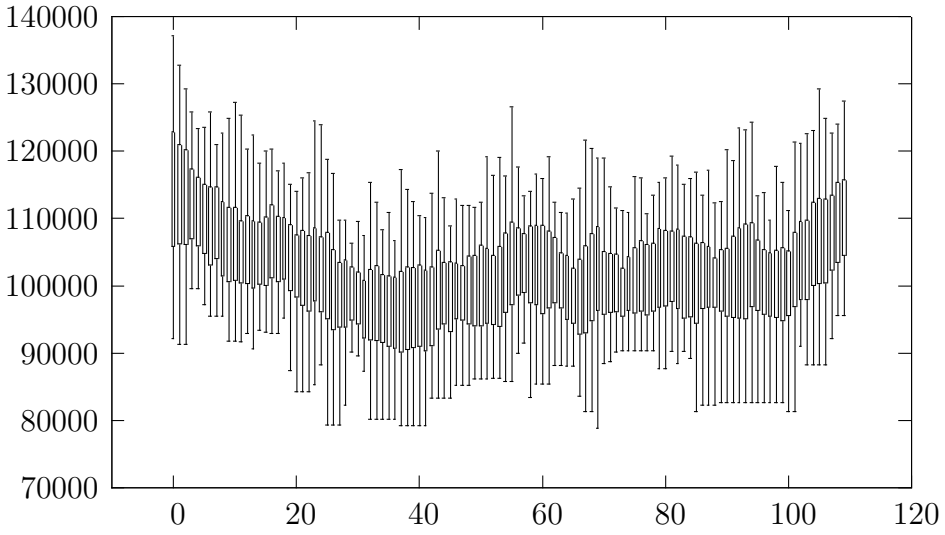
Rys. 1.6. Przykładowy wykres przebiegu



Na rysunku 1.6 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.

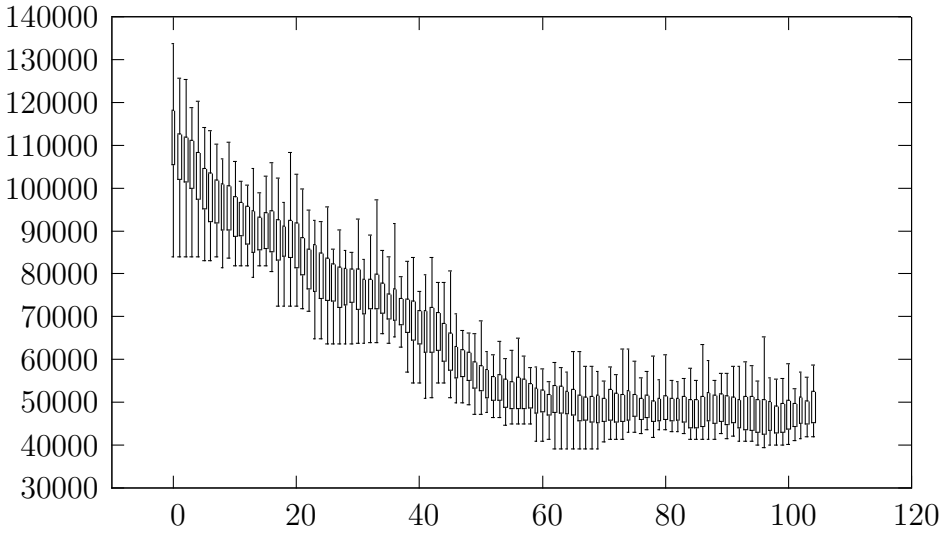
4. Zdecydowano się na wykorzystanie wariancji zamiast odchylenia standardowego ponieważ wartości odchylenia okazały się być zbyt małe, aby były dostrzegalne na wykresach.

Rys. 1.7. Wykres przebiegu, w którym globalne optimum nie znajduje się w ostatniej populacji



Na rysunku 1.7 przedstawiony jest wykres przebiegu na którym możemy zaobserwować sytuację opisaną na końcu rozdziału 1.1, tzn. taką, w której optimum globalne jest znalezione w innym pokoleniu niż ostatnie.

Rys. 1.8. Wykres przebiegu w którym obserwujemy stagnację



Na rysunku 1.8 przedstawiono sytuację, w której obserwujemy tzw. stagnację (wspominaną już w rozdziale 1.2.7). Możemy zaobserwować, że od pewnego momentu (około sześćdziesiątej generacji) kolejne pokolenia nie przynoszą znaczącej zmiany wyniku, co mogłoby być powodem do przerwania działania heurystyki.

1.4.2. Analiza wielu przebiegów heurystyki

Jak zostało wspomniane na początku tej sekcji, 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 3) wyników wielu przebiegów dla różnych konfiguracji i porównanie ich testem statystycznym, takim jak np. test t-studenta.

Rozdział 2

Przegląd literatury

Rozdział 3

Proponowane rozwiązania - algorytm ewolucyjny DSEA (*ang. double selection evolutionary algorithm*)

Dotychczasowe implementacje algorytmów ewolucyjnych zazwyczaj (choć nie zawsze [7], [9]) pomijały ważny aspekt rzeczywistości, który w przyrodzie okazuje się mieć duży wpływ na dopasowywanie się gatunków do środowiska - podział gatunku na płcie. W naturze 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.

W tym rozdziale zostanie zaproponowana metaheurystyka DSEA, która uwzględnia aspekt płci jako integralną część swojego działania, a nie jak w istniejących rozwiązaniach jako modyfikację powszechnie używanej metody. Różni się ona od zwyczajnych algorytmów ewolucyjnych tym, że wyróżnia się w niej 2 osobne operatory selekcji naturalnej i płciowej. Pierwszy z nich to w uproszczeniu operator selekcji z klasycznych algorytmów ewolucyjnych. Drugi ma za zadanie symulowanie dobierania się osobników w pary. Taki podział może prowadzić do stanu w których dalsze działanie algorytmu jest niemożliwe lub nieefektywne. W metaheurystyce DSEA zaproponowano sposoby zapobiegania takim sytuacjom.

Przy opisywaniu pojęć i operatorów wprowadzanych w tym rozdziale, używana będzie funkcja $plec(osobnik)$, przypisująca osobnikowi jego płeć. Jedyne wymaganie nałożone na jej przeciwdziedzinę to to, żeby była skończonym, niepustym zbiorem. Oznacza to, że reprezentację rozwiązania musimy powiększyć o odpowiednią cechę, określającą wartość tej funkcji.

Definicja 8 Osobnik w metaheurystyce DSEA

$$osobnik \in \mathcal{S} \tag{3.1}$$
$$osobnik_{DSEA} = osobnik_{EA}|g \tag{3.2}$$
$$g \in \mathcal{G} \tag{3.3}$$
$$\mathcal{G} \neq \emptyset \tag{3.4}$$
$$|\mathcal{G}| < \infty \tag{3.5}$$

$\mathcal{S}$ to przestrzeń rozwiązań, czyli zbiór wszystkich możliwych osobników *osobnik*. $osobnik_{DSEA}$ oznacza reprezentację wykorzystywaną w metaheurystyce DSEA, a $osobnik_{EA}$ - wykorzystywaną w klasycznych algorytmach ewolucyjnych. $g \in \mathcal{G}$ to cecha określająca płeć.
$\mathcal{G}$ t zbiór możliwych płci.
W niniejszej pracy przyjęto istnienie co najwyżej 2 płci, ze zbiorem płci określonym jako $\mathcal{G} = \{\odot\}$ lub $\mathcal{G} = \{♀, ♂\}$. Symbol $\odot$ to arbitralne oznaczenie na płeć w sytuacji, w której nie jest ona ważna do działania metahuerystyki.

Definicja 9 Funkcja *plec*(*osobnik*)

$$plec : \mathcal{S} \rightarrow \mathcal{G} \tag{3.6}$$

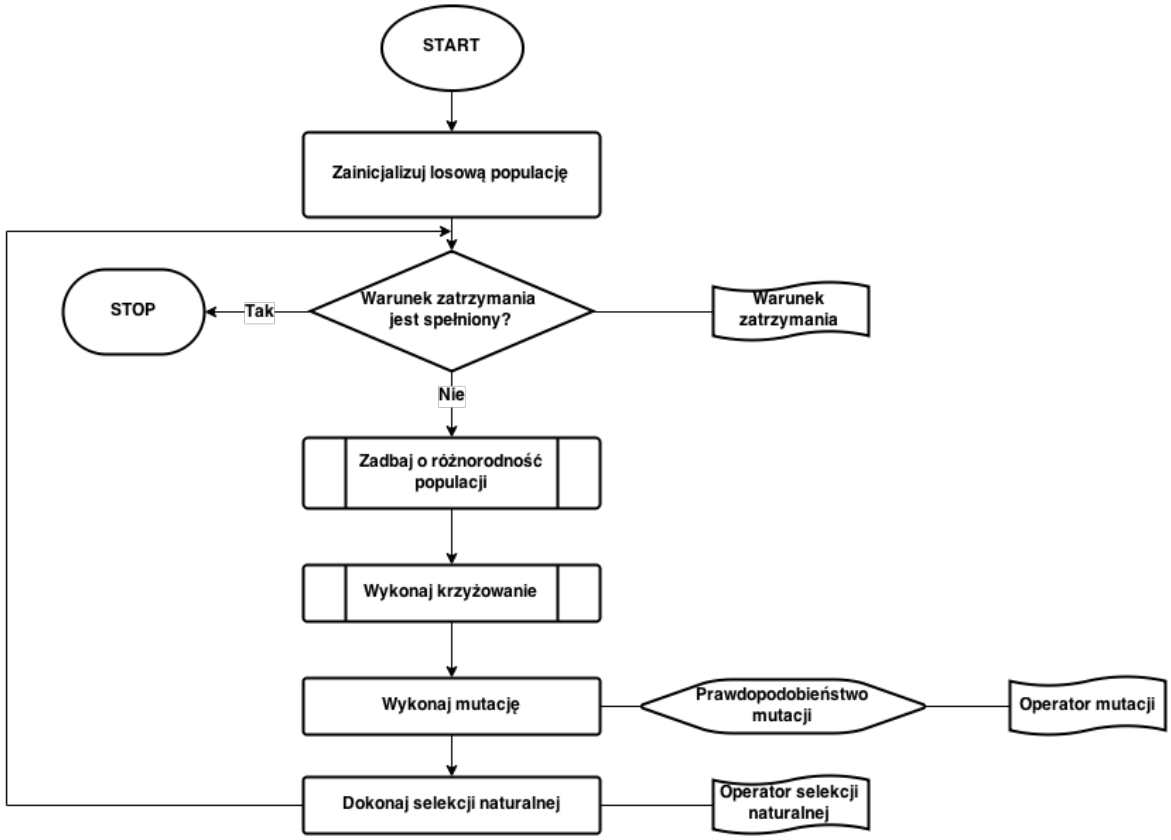
Funkcja *plec*(*osobnik*) przyporządkowywuje swojemu argumentowi jego płeć ze zbioru $\mathcal{G}$.

3.1. Algorytm DSEA w szczegółach

Jak zostało pokazane w rozdziale 2, istnieją rozwiązania które nie ignorują podziału populacji na płcie. Aby skutecznie je porównać i zaproponować nowe podejście, zdefiniowano schemat działania metaheurystki, opisany schematem blokowym znajdującym się na rysunku 3.1. Ujmuje on opisany aspekt płci organizmów w ramach nowego operatora selekcji. W dalszej części pracy tak zdefiniowaną metaheurystykę nazywać będziemy **algorytmem ewolucyjnym o podwójnej selekcji** (ang. *double selection evolutionary algorithm*) i odnosić się do niej za pomocą skrótu **DSEA**, będącego akronimem nazwy angielskiej.

Na rysunku 3.1 przedstawiono schemat działania metaheurystyki DSEA w postaci schematu blokowego.

Rys. 3.1. Schemat działania metaheurystyki DSEA



W większej części diagram zgadza się z diagramem 1.1 przedstawionym w podrozdziale 1.1. Istotne różnice między diagramami to zmiana działania kroku „Wykonaj krzyżowanie" oraz dodanie nowego kroku „Zadbaj o różnorodność populacji" zaraz przed nim.

Dodatkowo, DSEA wykorzystuje dwa operatory selekcji zamiast jednego - operator selekcji płciowej i naturalnej. Pierwszy z nich wraz z operatorem krzyżowania jest używany w kroku „Wykonaj krzyżowanie",a drugi w kroku „Dokonaj selekcji naturalnej". W następnych dwóch podrozdziałach zostaną one szczegółowo opisane. W kolejnym podrozdziale znajdują się opisy zmian działania odpowiednich kroków. Całość zamknięta jest opisem związku między DSEA, a innymi odmianami algorytmu ewolucyjnego.

3.1.1. Operator selekcji naturalnej

Algorytm DSEA, jak sugeruje nazwa w języku angielskim, 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, jak zostało opisane w rozdziale 1.2.6.

Dodatkowo, wymaga się, aby prawdopodobieństwo znalezienia się osobnika w zbiorze zwracanym przez ten operator było niezależne od płci osobnika. Zostało to zapisane za pomocą równania 3.16, korzystając z twierdzenia zgodnie z którym prawdopodobień-

stwo łączne dwóch zdarzeń jest równe iloczynowi ich prawdopodobieństw wtedy i tylko wtedy, gdy są to zdarzenia niezależne.

Definicja 10 Operator selekcji naturalnej

$$\texttt{opSelNat} : \mathcal{S}^{\{p\}} \rightarrow \mathcal{S}^{\{\texttt{rozmiarPopulacji}\}} \tag{3.7}$$

$$\texttt{rozmiarPopulacji} \in \mathbb{N}_+ \tag{3.8}$$

$$p \approx (1 + \texttt{prawdMutacji} \times \texttt{m} \tag{3.9}$$

$$+ \texttt{prawdKrzyzowania} \times \underline{\texttt{c}}) \tag{3.10}$$

$$\times \texttt{rozmiarPopulacji} \tag{3.11}$$

$$S' \leftarrow \{s_1, \ldots s_p\} \tag{3.12}$$

$$s \in \texttt{opSelNat}(S') \Rightarrow \exists_{s_i \in S'} s_i = s \tag{3.13}$$

$$P(i) \leftarrow P(s_i \in \texttt{opSelNat}(S')) \tag{3.14}$$

$$\forall_{i=1\ldots p} P(i) < P(j) \Leftrightarrow (s_i, s_j) \in \mathbf{R}_{\lhd} \tag{3.15}$$

$$\forall_{i=1\ldots p, g \in \mathcal{G}} P(s_i \in \texttt{opSelNat}(S') \wedge plec(t) = g) = P(i) \times P(plec(s_i) = g) \tag{3.16}$$

`opSelNat` to operator selekcji naturalnej. Jako wejście przyjmuje on populację na końcu każdego pokolenia, a zwraca populację używaną w kolejnym pokoleniu. p to przybliżenie rozmiaru zbioru wejściowego za pomocą prawdopodobieństw zastosowania operatorów oraz rozmiaru populacji.
S' to pomocniczy zbiór przedstawiający przykładową populację na końcu pokolenia, zawierający osobniki s_i.
$P()$ to funkcja prawdopodobieństwa. W szczególności $P(i)$ oznacza prawdopodobieństwo tego, że osobnik s_i znajdzie się w zbiorze będącym wynikiem zastosowania opisywanego operatora do populacji S'.
$\mathbf{R}_{\lhd}$ to relacja porządku określona na osobnikach na podstawie ich ocen. Zapis $(s_i, s_j) \in \mathbf{R}_{\lhd}$ oznacza, że osobnik s_i jest gorzej dopasowany niż s_j.

Operator wyboru

Standardowo korzysta się z operatorów selekcji naturalnej, które odpowiednią liczbę 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 za pomocą tej samej metody, która dalej będzie nazywana operatorem wyboru. Definicja 11 opisuje taki operator. Rzecz jasna istnieją realizacje, które korzystają z wielu operatorów wyboru, ale nie są one tematem tej pracy i nie będą w niej poruszane.

Intuicyjnie, operator selekcji to przekształcenie jednego zbioru w inny, którego elementy będą albo elementami zbioru wejściowego, albo jego podzbiorami. Pierwsza z tych możliwości opisuje operator selekcji naturalnej, przyjmujący populację z końca pokolenia, a zwracający populację z początku następnej generacji. Druga z nich mówi o operatorze selekcji płciowej, przyjmującym populację na początku pokolenia, a zwraca zbiór zestawów rodziców, wykorzystywany później w krzyżowaniu. Operator wyboru natomiast służy do wyboru pojedynczego elementu z tego zbioru.

Definicja 11 Operator wyboru

$$\texttt{opWyboru} : \mathcal{S}^{\{k\}} \rightarrow \mathcal{S} \tag{3.17}$$

$$S' \leftarrow \{s_1, \ldots s_k\} \tag{3.18}$$

$$s \in \texttt{opSelNat}(S') \Rightarrow \exists_{s_i \in S'} s_i = s \tag{3.19}$$

`opWyboru` to operator wyboru. Przyjmuje on k osobników, spośród których wybiera jednego.
S' to pomocniczy zbiór osobników s_i wykorzystywany w kolejnych równaniach.

Realizacje operatorów wyboru nazywane są 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 dalszej części pracy przy okazji opisu implementacji wykorzystanej do badań.

Warto zauważyć, że wcześniej zdefiniowany operator $random(X)$ może być traktowany jako operator wyboru. Nie nadaje się on do zastosowań mających na celu poprawę wyników algorytmu, jednak zostanie on wykorzystany do opisu istniejących podejść w modelu DSEA.

Operator selekcji naturalnej wykorzystujący operator wyboru

Wydzielenie abstrakcji wyboru jednego rozwiązania pozwala na zapisanie uogólnionego schematu działania większości popularnie stosowanych operatorów selekcji. Jest on opisany w algorytmie 3.1.

Algorytm 3.1 Schemat działania operatora selekcji naturalnej korzystającego z operatora wyboru

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

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

Działanie takiego operatora zaczyna się od inicjalizacji zmiennej *wynik*, zwracanej jako efekt końcowy algorytmu (linia 5). Początkowa jej wartość to zbiór pusty (linia 1), jednak w kolejnym kroku rozpoczyna się pętla, powtarzana tak długo, aż zbiór ten będzie miał odpowiedni rozmiar (linia 2). 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 3).

3.1.2. 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 wprowadzany metaheurystyki DSEA, 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.

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.

3.1.3. Różnice w działaniu kroków algorytmu DSEA względem klasycznych algorytmów ewolucyjnych

Metaheurystyka DSEA różni się od innych nie tylko wykorzystywanymi przez nią operatorami. Niektóre kroki jej działania również są niestandardowe. Zostanie to opisane w kolejnych podrozdziałach.

Definicja 12 Operator selekcji płciowej

$$\texttt{opSelPlciowej} : \mathcal{S}^{\{\texttt{rozmiarPopulacji}\}} \rightarrow \left(\mathcal{S}^{\{\overline{\texttt{c}}\}}\right)^{\{q\}} \tag{3.20}$$

$$q = \lceil \texttt{prawdKrzyzowania} \times \texttt{rozmiarPopulacji} \rceil \tag{3.21}$$

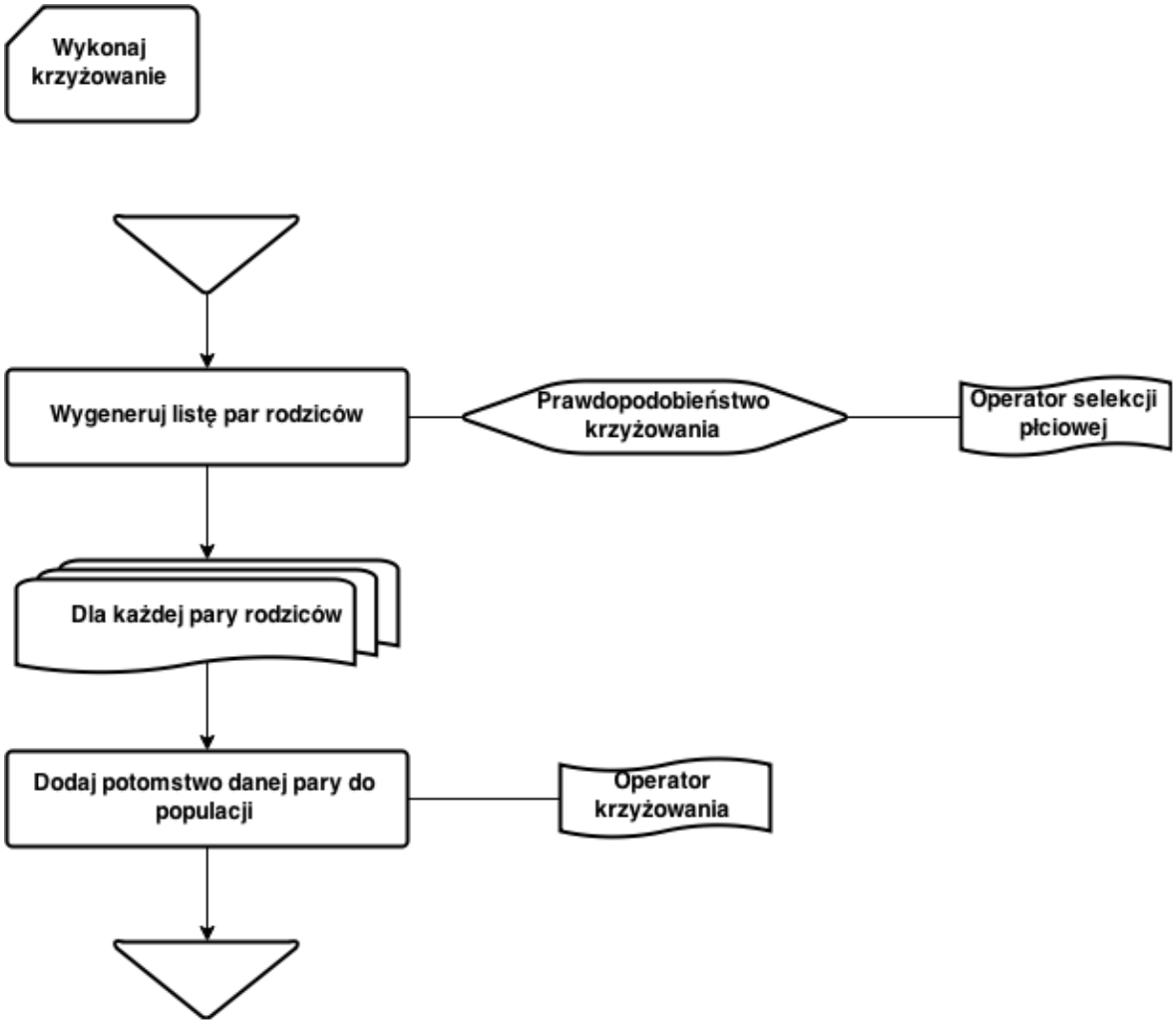
$$S' \leftarrow \{s_1, s_2, \ldots s_{\texttt{rozmiarPopulacji}}\} \tag{3.22}$$

$$\exists_{P \in \texttt{opSelekcji}(S')} s \in P \Rightarrow s \in S' \tag{3.23}$$

$$\tag{3.24}$$

`opSelPlciowej` to operator selekcji płciowej. Zwraca on zbiór q zbiorów, z których każdy zawiera $\overline{\texttt{c}}$ osobników.
$\overline{\texttt{c}}$ to ilość osobników potrzebna do wykonania krzyżowania (patrz: definicja 5 w rozdziale 1.2.5).
S' to pomocniczy zbiór osobników s_i, a P to oznaczenie jednego ze zbiorów rodziców.

Krok „Wykonaj krzyżowanie”



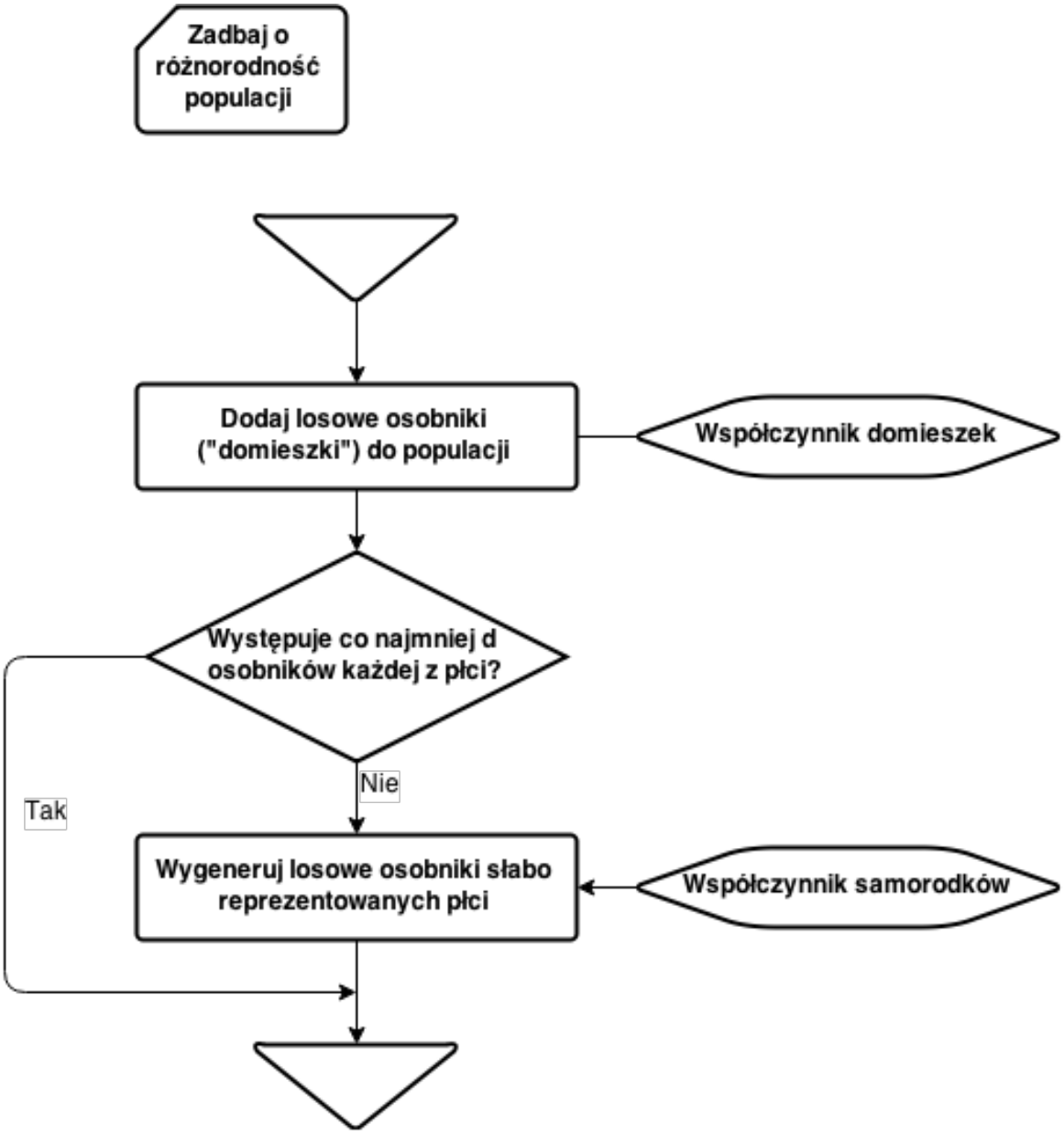
Rys. 3.2. Szczegóły działania kroku „Wykonaj krzyżowanie”

Na rysunku 3.2 znajduje się diagram przedstawiający krok „Wykonaj krzyżowanie" w postaci sekwencji kolejnych kroków. Za pomocą operatora selekcji płciowej[1] generowana 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 realizacji opisanej w podrozdziale 1.3 za pomocą algorytmu 1.1 wyniki operatora krzyżowania są dołączane najpierw do osobnego zbioru, a dopiero po wystarczającej liczbie powtórzeń operacji krzyżowania. Ma to zapobiegać sytuacjom, w których potomek wygenerowany w tym pokoleniu zostałby w tym samym pokoleniu rodzicem. W heurystyce potomstwo jest od razu dołączane do całości populacji. Dzieje się tak, ponieważ zadanie wyboru rodziców jest oddelegowane do operatora, który zawsze zostanie użyty przed operatorem krzyżowania.

Krok „Zadbaj o różnorodność populacji"

Wprowadzenie podziału populacji na płcie powinno skutkować odmiennym traktowaniem osobników w zależności od tej cechy. Negatywną stroną tego zjawiska jest fakt, że operator selekcji naturalnej, który nie powinien brać pod uwagę płci, może zaburzyć stosunek liczności osobników danej płci względem siebie. W skrajnych sytuacjach może dojść do tego, że w całej populacji zabraknie osobników którejś z płci, co uniemożliwi dalsze działanie metaheurystyki (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.

1. W jednym z następnych podrozdziałów opisane zostało znaczenie i definicja tego operatora, uproszczenie polegające na zwracaniu par rodziców oraz różnica w interpretacji parametru `prawdKrzyzowania`.



Rys. 3.3. Szczegóły działania kroku „Zadbaj o różnorodność populacji"

Na rysunku 3.3 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 liczby nowych osobników do wartości `rozmiarPopulacji`.

Druga procedura w tym 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` $= 0.05$.

3.1.4. Realizacja wybranych operatorów selekcji płciowej

W rozdziale 2 przedstawiono kilka prac naukowych uwzględniających zjawisko płci w algorytmach ewolucyjnych. Spośród przedstawionych tam pozycji wybrano dwie, które prezentowały interesujące podejścia do zagadnienia: GGA [7], które opierało się na wymuszeniu różnych płci rodziców, wybierając ich losowo, SexualGA [9], 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 za pomocą tego samego schematu, jednak odmiennie parametryzowanego dla każdej z tych metod. Jego działanie jest wyjaśnione w algorytmie 3.2.

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

Algorytm 3.2 Schemat działania wybranych operatorów selekcji płciowej

Wejście:
populacja ▷ Populacja wejściowa, z początku aktualnego pokolenia
Wyjście:
rodzice ▷ Zbiór par rodziców
Parametry:
`opWyboru1` ▷ Operator wyboru pierwszego z rodziców
`opWyboru2` ▷ Operator wyboru drugiego z rodziców
`plecMaZnaczenie` ▷ Wartość logiczna określająca, czy rodzice muszą różnić się płcią
`prawdKrzyzowania` ▷ Prawdopodobieństwo krzyżowania
`rozmiarPopulacji` ▷ Ilość osobników na początku pokolenia

0: **operator** `opSelPlciowej`(*populacja*)
1: **var** $rodzice \leftarrow \emptyset$
2: **var** $kandydaci1 \leftarrow \begin{cases} \{s : plec(s) = ♀, s \in populacja\} & : \texttt{plecMaZnaczenie} = \top \\ populacja & : \texttt{plecMaZnaczenie} = \bot \end{cases}$
3: **var** $kandydaci2 \leftarrow \begin{cases} \{s : plec(s) = ♂, s \in populacja\} & : \texttt{plecMaZnaczenie} = \top \\ populacja & : \texttt{plecMaZnaczenie} = \bot \end{cases}$
4: **while** $|rodzice| < \texttt{rozmiarPopulacji} \times \texttt{prawdKrzyzowania}$ **do**
5: **var** $rodzic1 \leftarrow opWyboru1(kandydaci1)$
6: **var** $rodzic2 \leftarrow opWyboru2(kandydaci1 \setminus \{rodzic1\})$
7: $rodzice \leftarrow rodzice \cup \{\{rodzic1, rodzic2\}\}$
8: **end while**
9: **return** *rodzice*
10: **end operator**

Symbole $\top$ i $\bot$ oznaczają kolejno prawdę i fałsz.
Operator wyboru przyjmuje zbiór osobników, a zwraca jego element.

Działanie operatora rozpoczyna się od deklaracji i inicjalizacji zmiennych pomocniczych. Zbiór *rodzice* początkowo jest pusty (linia 1). Dodatkowo, deklarujemy dwa zbiory przechowujące kandydatów na każdego z rodziców (linie 2-3). W zależności od parametru `plecMaZnaczenie` przechowują one albo całą populację (jeżeli zachodzi $\texttt{plecMaZnaczenie} = \bot$, czyli rodzice mogą być tej samej płci), albo odflitrowane osobniki odpowiedniej płci (jeżeli $\texttt{plecMaZnaczenie} = \top$, czyli wymagane jest, aby rodzice byli różnych płci).

Następnie, póki nie zostanie wybrana odpowiednia liczba par rodziców (linia 4) powtarzany jest zestaw operacji określający kolejną z nich. Za pomocą operatora `opWyboru1` ze zbioru `kandydaci1` wybierany jest pierwszy rodzic (linia 5). Drugi z rodziców jest wybierany ze zbioru `kandydaci2` za pomocą operatora `opWyboru2` (linia 6), nie dopuszczając do sytuacji w której oboje rodziców byłoby tym samym osobnikiem (mogłoby się tak zdarzyć, jeżeli $\texttt{plecMaZnaczenie} = \bot$, więc oboje z nich byłoby wybieranych z całej populacji). Dwuelementowy zbiór tak wybranych rodziców jest dołączany do zbioru *rodzice*.

Kiedy określimy odpowiednio dużo par, zmienna *rodzice* jest zwracana jako wynik operatora (linia 9).

Tak określona realizacja spełnia założenie tej pracy, zgodnie z którym płci jest co najwyżej 2 ($|\mathcal{G}| \in \{1, 2\}$; jeżeli `plecMaZnaczenie` $= \bot$ to liczność tego zbioru nie ma znaczenia, ponieważ funkcja $plec(s)$ nie jest używana, jednak w przeciwnej sytuacji musi ona wynosić 2. Dodatkowo zakłada się liczbę rodziców wymaganą przez operator krzyżowania $\overline{\mathtt{c}} = 2$. Algorytm można dość łatwo uogólnić do postaci, w której liczby te są większe, pamiętając, że jeżeli `plecMaZnaczenie` $= \top$, to rodzice muszą być różnych płci, więc rozmiar argumentu operatora krzyżowania musi być równy ich liczbie ($\overline{\mathtt{c}} = |\mathcal{G}|$).

Metaheurystyka DSEA jest w pewien sposób ogólniejsza od zwykłych algorytmów ewolucyjnych. Jeżeli dobierze się odpowiednie operatory i parametry, to można za jej pomocą zasymulowanie istniejących rozwiązań. Następne 3 podrozdziały opisują sposoby parametryzacji DSEA potrzebne do odtworzenia innych metageurystyk znanych z literatury.

Standardowe algorytmy ewolucyjne

Jeżeli:

- użyto operatora selekcji płciowej opisanego powyżej, sparametryzowanego za pomocą:
  - `plecMaZnaczenie` $= \bot$,
  - `opWyboru1` $= random$,
  - `opWyboru2` $= random$,
- użyto wybranego operatora selekcji naturalnej (`opSelNat` $\neq random$),
- zignorowano liczbę płci ($|\mathcal{G}| \in \mathbb{N}_+$) , ponieważ `plecMaZnaczenie` $= \bot$,

to DSEA sprowadza się do standardowego algorytmu ewolucyjnego, w którym na prawdopodobieństwo wydania potomstwa ma wpływ jedynie operator selekcji naturalnej.

GGA

GGA [7] to odmiana algorytmu genetycznego (co oznacza, że jej autorzy wykorzystywali reprezentację osobnika przez wektor bitów), w której wyróżnia się cechę osobnika określającej jego płeć i wymaga, aby rodzice różnili się nią. Każdy z rodziców jest wybierany losowo. Można to zrealizować spełniając następujące warunki:

- użyto operatora selekcji płciowej opisanego powyżej, sparametryzowanego za pomocą:
  - `plecMaZnaczenie` $= \top$,
  - `opWyboru1` $= random$,
  - `opWyboru2` $= random$,
- użyto wybranego operatora selekcji naturalnej (`opSelNat` $\neq random$),
- liczba płci jest większa niż 1($|\mathcal{G}| > 1$).

SexualGA

SexualGA [9] to inna odmiana na temat algorytmu genetycznego uwzględniającego płeć. Autorzy nie wyróżniali w niej cechy określającej płeć osobnika, proponując inne podejście do tego zjawiska. Przedstawione rozwiązanie wykorzystywało operator selekcji do wybrania rodziców, a nie osobników przechodzących do kolejnego pokolenia, a

każdy z nich był wybierany za pomocą innego operatora wyboru[3]. Miało to symulować różne zachowania samców i samic związane z wyborem partnera.

Jeżeli poniższe warunki będą spełnione, to DSEA sprowadza się do SexualGA:

- użyto operatora selekcji płciowej opisanego powyżej, sparametryzowanego za pomocą:
  - `plecMaZnaczenie` $= \perp$,
  - `opWyboru1` $\neq random \vee$ `opWyboru2` $\neq random$,
- użyto losowego operatora selekcji naturalnej (`opSelNat` $= random$),
- zignorowano liczbę płci ($|\mathcal{G}| \in \mathbb{N}_+$) , ponieważ `plecMaZnaczenie` $= \perp$).

Podany wyżej warunek, mówiący o tym, że co najmniej jeden operator wyboru musi być różny od losowego (*random*) wynika z tego, że gdyby oba operatory działały losowo, to metaheurystyka polegałaby na kompletnie przypadkowym przeszukiwaniu przestrzeni rozwiązań. Według autorów najlepsze wyniki osiągano używając operatora losowego w połączeniu z innym, różnym od niego, jednak sytuacja w której użyto tego samego operatora dla obu płci jest warta zbadania.

3.2. Proponowany operator selekcji płciowej

U wielu gatunków występujących w przyrodzie (np. u większości płetwonogich[6], w tym morsów i fok) występuje zjawisko formowania się haremów. Polega ono na tym, że tylko mała liczba osobników jednej z płci (zazwyczaj samców[4]) nazywanych dalej *osobnikami alfa*, przyczynia się do wydania na świat potomstwa. Oznacza to, że dostęp do grupy samic nazywanej *haremem* ma bardzo ograniczona grupa samców. Dzieje się tak, ponieważ osobniki alfa są w ogólności silniejsze od pozostałych i bronią dostępu do samic przed resztą samców. Zakaz ten nie zawsze jest przestrzegany. W momencie, w którym osobniki alfa są czymś zajęte, pewien odsetek samców ma szansę na rozmnożenie się. Osobniki którym się to uda nazywane są *osobnikami beta*.

Mimo, że takie zjawisko w dużym stopniu ogranicza różnorodność genetyczną, to w naturze wydaje się przynosić pozytywne rezultaty. Z tego powodu w niniejszej pracy zdecydowano się zbadać haremowe podejście do dobierania się w pary w celu rozmnażania. Zrealizowano to definiując operator selekcji płciowej, nazywany operatorem haremowym. Jest on opisany w algorytmie 3.3. Tak przedstawiona realizacja wymaga, aby w populacji występowały dokładnie 2 płcie ($|\mathcal{G}| = 2$).

Działanie takiego operatora zależy od pięciu parametrów, poza parametrami samej metaheurystyki. Są to trzy operatory wyboru: `opWyboruAlfa` służący do wyboru osobników alfa, `opWyboruBeta` używany do wyboru osobników beta oraz `opWyboruPartnerów` wykorzystywany do wyboru partnerów dla osobników alfa i beta. Ponadto, wykorzystywane są dwa parametry liczbowe: `liczbaAlfa` określający liczbę osobników alfa wybieranych w pojedynczym pokoleniu i `wspBeta`, czyli współczynnik określający stosunek liczby par rodziców z których jeden jest osobnikiem beta do liczby wszystkich par rodziców

Działanie operatora zaczyna się od inicjalizacji zmiennej *rodzice* na pusty zbiór (linia 1) oraz rozdzielenia populacji na zbiory *kandydaci*1 i *kandydaci*2, zawierające osob-

3. W oryginalnej pracy nie używano określenia „operator wyboru", a „operator selekcji".
4. Mimo, że nie zawsze osobniki alfa są płci męskiej, to przyjęte zostanie takie uproszczenie. Osobniki alfa i beta będą nazywane samcami, a ich parterki - samicami.

Algorytm 3.3 Schemat działania operatora haremowego

Wejście:
populacja ▷ Populacja wejściowa, z początku aktualnego pokolenia
Wyjście:
rodzice ▷ Zbiór par rodziców
Parametry:
`liczbaAlfa` ▷ Liczba osobników alfa
`wspBeta` ▷ Stosunek liczby par rodziców zawierających osobnika alfa do liczby wszystkich par
`opWyboruAlfa` ▷ Operator wyboru osobników alfa
`opWyboruBeta` ▷ Operator wyboru osobników beta
`opWyboruPartnerow` ▷ Operator wyboru partnerów przeciwnej płci
`prawdKrzyzowania` ▷ Prawdopodobieństwo krzyżowania
`rozmiarPopulacji` ▷ Liczba osobników na początku pokolenia

0: **operator** opHaremowy(*populacja*)
1: **var** $rodzice \leftarrow \emptyset$
2: **var** $kandydaci1 \leftarrow \{s : plec(s) = \male, s \in populacja\}$
3: **var** $kandydaci2 \leftarrow \{s : plec(s) = \female, s \in populacja\}$
4: **var** $liczbaZbiorowRodzicow \leftarrow \lceil$`prawdKrzyzowania` $\times$ `rozmiarPopulacji`$\rceil$
5: **var** $liczbaBeta \leftarrow \lceil$`wspBeta` $\times liczbaZbiorowRodzicow\rceil$
6: **var** $perAlfa \leftarrow \lceil(liczbaZbiorowRodzicow - liczbaBeta)/$`liczbaAlfa`$\rceil$
7: ▷ Liczba zbiorów rodziców w których znajdzie się dany osobnik alfa
8: **for** $i \leftarrow 1$ **to** `liczbaAfa` **do**
9: **var** $alfa \leftarrow$ `opWyboruAlfa`$(kandydaci1)$
10: $kandydaci1 \leftarrow kandydaci1 \setminus \{alfa\}$
11: **for** $j \leftarrow 1$ **to** $perAlfa$ **do**
12: **var** $partner \leftarrow$ `opWyboruPartnerów`$(kandydaci2)$
13: $zbioryRodzicow \leftarrow zbioryRodzicow \cup \{\{alfa, partner\}\}$
14: **end for**
15: **end for**
16: **for** $i \leftarrow 1$ **to** $liczbaBeta$ **do**
17: **var** $beta \leftarrow$ `opWyboruBeta`$(kandydaci1)$
18: **var** $partner \leftarrow$ `opWyboruPartnerow`$(kandydaci2)$
19: $rodzice \leftarrow rodzice \cup \{\{beta, partner\}\}$
20: **end for**
21: **return** $rodzice$
22: **end operator**

niki różnych płci (linie 2 - 3). Następnie obliczane są wartości *liczbaZbiorowRodzicow*, czyli całkowita liczba par rodziców zwracanych przez operator (linia 4), *liczbaBeta*, czyli liczba osobników beta biorących udział w rozmnażaniu (linia 5), oraz $perAlfa$, czyli liczba par rodziców przypadających na jednego osobnika alfa (linia 7).

Kolejnym krokiem jest wygenerowanie par rodziców zawierających osobnika alfa. `liczbaAfa` razy powtarzane jest (w pętli w linii 8) wybranie kolejnego takiego osobnika $alfa$ (w linii 9) i usunięcie go ze zbioru kandydatów danej płci (w linii 10). Dzięki temu żaden osobnik alfa nie zostanie wybrany dwukrotnie, ani nie pokryje się z którymś z osobników beta. Dla danego osobnika alfa wybierany jest $perAlfa$ (w pętli w linii 11) partnerów *partner* (linia 12), a tak określone pary rodziców są dołączane do zbioru *rodzice* (w linii 13).

W dalszej kolejności wybierane jest *liczbaBeta* osobników beta (linie 16-17), dla których dobierani są partnerzy *partner* (w linii 18). Tak wybrane pary również dołączane są do zbioru *rodzice* (linia 19).

Operator zwraca zmienną *rodzice* (w linii 21).

Przedstawiony schemat nie zgadza się z definicją operatora selekcji płciowej przedstawioną w jednym z poprzednich podrozdziałów, ponieważ w niektórych sytuacjach zwraca więcej par rodziców niż powinien.

Oczekiwany rozmiar zbioru zwracanego przez operator to *liczbaZbiorowRodzicow*, czyli $\lceil$`prawdKrzyzowania` $\times$ `rozmiarPopulacji`$\rceil$, jednak ponieważ zaokrąglamy wartość $perAlfa$ w górę, to często generowane jest `liczbaAlfa` nadmiarowych par. Można temu zaradzić modyfikując ostatnią pętlę, generującą pary z osobnikami beta, w sposób opisany algorytmem 3.4.

Algorytm 3.4 Sposób zaradzenia nadmiarowi zwracanych zbiorów

```
 0: operator opHaremowy(populacja)
 1:     (...)
15:     while |rodzice| < liczbaZbiorowRodzicow do
16:         var beta ← opWyboruBeta(kandydaci1)
17:         var partner ← opWyboruPartnerow(kandydaci2)
18:         rodzice ← rodzice ∪ {{beta, partner}}
19:     end while
20:     return rodzice
21: end operator
```

Takie podejście powoduje, że parametr `wspBeta` traci swoje dosłowne znaczenie i określa przybliżony stosunek liczby par rodziców zawierających osobniki beta do liczby wszystkich zwracanych par.

W implementacji wykonanej w niniejszej pracy nie zastosowano tej modyfikacji. Umotywowane było to małą różnicą między zakładanym i rzeczywistym rozmiarem zbioru par rodziców, oraz faktem, że *de facto* generowanie nadmiarowych par powoduje minimalnie lepszą eksploatację przestrzeni rozwiązań (ponieważ jest ona gęściej próbkowana).

Jeżeli w populacji występuje więcej niż 2 płcie ($|\mathcal{G}| > 2$), to jedną z nich należy wyróżnić jako tą, z której pochodzą osobniki alfa i beta. Ponadto, operator ma wtedy więcej parametrów (tyle operatorów wyboru partnerów ile jest niewyróżnionych płci),

a zamiast jednego partnera (linie 12 i 18) wybiera się ich odpowiednio więcej, każdego z wykorzystaniem przeznaczonego do tego operatora wyboru.

Rozdział 4

Eksperymenty

Niniejsza praca ma na celu zbadanie i porównanie różnych podejść do zagadnienia płci wykorzystywanych w algorytmach ewolucyjnych. Aby to osiągnąć zaimplementowano metody opisane w poprzednim rozdziale i przeprowadzono zestaw eksperymentów, których wyniki zostały porównane.

Ten rozdział zaczyna się od prezentacji wybranych problemów optymalizacji, na których przeprowadzone zostały badania. Kolejna sekcja zawiera opis implementacji badanych metod. Następnie omówione są procedury wykorzystywane przy eksperymentach. Całość zamknięta jest podsumowaniem ich przebiegu i przedstawieniem wyników.

4.1. Wybrane problemy optymalizacji

Aby zbadać jakość działania wybranych metod zaimplementowano i podjęto próbę rozwiązania dwóch problemów optymalizacji. Były to problem komiwojażera (*ang. traveling salesman problem*, w skrócie *TSP*) oraz binarny problem plecakowy.

Problem komiwojażera to zadanie znalezienia minimalnego cyklu Hamiltona w pełnym grafie ważonym. Cykl Hamiltona to cykl, w którym każdy wierzchołek jest odwiedzany tylko raz (z pominięciem pierwszego wierzchołka, który jest taki sam jak ostatni). Graf pełny, to taki, w którym między dowolnymi dwoma wierzchołkami istnieje krawędź, a graf ważony to taki, w którym krawędzie opisane są pewnymi wartościami, nazywanymi wagami. Minimalnym cyklem nazywamy taki, dla którego suma wag krawędzi jest najmniejsza.

Problem ten można przełożyć na rzeczywistą sytuację, w której mamy zbiór punktów na mapie (np. miast, reprezentowanych przez wierzchołki grafu) leżących w pewnych odległościach od siebie[1] (które są reprezentowane przez wagi krawędzi). Rozwiązanie problemu komiwojażera sprowadza się do minimalizacji długości drogi przechodzącej przez wszystkie punkty. Wynik optymalizacji jest cenny w rzeczywistych

1. Geograficzne odległości między punktami to tylko jedna z możliwych interpretacji. Zamiast tego między punktami może być określony czas przejazdu, jego koszt, itd. W ogólności, znane są pewne wartości związane z każdymi dwoma punktami, które oznaczają tym gorszą sytuację z punktu widzenia osoby rozwiązującej problem, im są wyższe.

zastosowaniach, ponieważ pozwala m.in. zaoszczędzić zasoby w firmach zajmujących się transportem.

Binarny problem plecakowy (nazywany czasem dyskretnym problemem plecakowym) to zadanie optymalizacji, w którym ze zbioru n par $\{(k_1, w_1), (k_2, w_2), \ldots, (k_n, w_n)\}$ należy wybrać pozdbiór, dla którego suma pierwszych elementów z pary będzie jak największa, a suma drugich elementów nie przekroczy zadanej wartości W. Problem ten przekłada się na sytuację, w której mamy pewien pojemnik (popularnie nazywany plecakiem, skąd bierze się nazwa problemu) o określonej pojemności W i zestaw przedmiotów, które możemy do niego włożyć. Każdy przedmiot jest opisamy parą (k, w), gdzie k reprezentuje jego wartość, a w jego rozmiar, czy też objętość. Rozwiązaniem problemu jest taki zestaw przedmiotów jaki zmieści się w plecaku i będzie miał jak największą wartość.

Ogólna wersja problemu plecakowego polega na przypisaniu każdemu z przedmiotów (par) liczby naturalnej, określającej ile jego egzemplarzy wkładamy do plecaka, jednak w ramach niniejszej pracy zdecydowano się na binarną wariację tego problemu, która dopuszcza wykorzystanie co najwyżej jednego egzemplarza każdego obiektu.

Opisane problemy zostały wybrane, ponieważ reprezentują dwie klasy zadań powszechnie rozwiązywane za pomocą algorytmów ewolucyjnych. Binarny problem plecakowy to popularne zadanie w którym rozwiązania są reprezentowane jako wektory bitów, a problem komiwojażera to znane zagadnienie, powszechnie wykorzystywane do badania jakości działania heurystyk. Ponadto, problem plecakowy to zadanie optymalizacji z ograniczeniami, których nie da się łatwo zachować w ramach operatorów genetycznych (w przeciwieństwie np. do problemu komiwojażera, w którym można zastosować operatory, które gwarantują, że zwrócą rozwiązania nie przekraczające ograniczeń, jeżeli osobniki wejściowe ich nie przekraczały).

4.2. Implementacja

W tej sekcji opisane są operatory i ich implementacje wykorzystywane podczas przeprowadzania eksperymentów. Pierwsza podsekcja zawiera opis elementów heurystyki niezależnych od problemu, czyli operatorów wyboru i kryterium zarzymania, a kolejne dwie - reprezentacji osobników i operatorów genetycznych wykorzystanych w badaniu wybranych problemów.

4.2.1. Komponenty niezależne od problemu

Niezależnie od tego jaki problem jest rozwiązywany za pomocą algorytmów ewolucyjnych, pewne elementy są niemalże uniwersalne. W tym rozdziale opisane zostaną operatory wyboru oraz kryterium stopu, oraz zestawienie oznaczeń wykorzystywanych w dalszych rozdziałach.

Operatory wyboru

Jak zostało zdefiniowane w podsekcji 3.1.1, zadaniem operatora wyboru jest zwrócenie pojedynczego elementu ze zbioru osobników. W standardowych algorytmach ewolucyjnych powszechnie używane są operatory selekcji, które można zaimplementować zgodnie z algorytmem 3.1, korzystające z tego operatora w celu wybrania każdego

ze zwracanych osobników. Ponadto, operatory wykorzystywane w heurystyce DSEA również korzystają z tej abstrakcji.

Jedną z popularnych implementacji operatora wyboru jest **operator ruletkowy** (*ang. roulette select*), wykorzystujący pewną dodatnią funkcję (lub operator) przypisującą każdemu osobnikowi tzw. „wagę". Zwraca on rozwiązanie z prawdopodobieństwem wprost proporcjonalnym do wartości tej funkcji. Działanie tego operatora standardowo tłumaczy się w następujący sposób: Wyobraźmy sobie koło ruletki. Na całym tym kole zaznaczmy obszary (wycinki koła) przypisane do osobników, w taki sposób, że odpowiedni obszar ma rozmiar wprost proporcjonalny do wagi rozwiązania. Następnie zakręćmy tym kołem, a osobnika, do którego obszar będzie tym, na którym zatrzyma się kulka ruletki, zwróćmy.

Istnieją co najmniej dwie standardowe wersje tego operatora, różniące się funkcją przypisującą wagę osobnikom. Jedną z nich jest ważenie osobników ich oceną. W przypadku problemu maksymalizacji przekłada się to bezpośrednio na wymóg nałożony na standardowy operator selekcji, mówiący o tym, że osobniki „lepsze" mają większą szanse na zostanie wylosowanymi. Odmiana ta jest jednak problematyczna w przypadku problemów minimalizacji, ponieważ „lepsze" osobniki mają wtedy niższą ocenę, a co za tym idzie - zajmują mniej miejsca na wirtualnym kole ruletki, co przekłada się na mniejszą szansę wylosowania. Jeżeli dla danego problemu znana jest górne ograniczenie funkcji oceny (czyli największa wartość pochodząca z jej przeciwdziedziny), to problem minimalizacji możemy przekształcić do problemu maksymalizacji, poprzez zdefiniowanie nowej funkcji oceny, zwracającą wartość oryginalnego kryterium odjętą od jego górnego ograniczenia[2]. W takiej sytuacji, osobniki możemy ważyć za pomocą ich „rangi", czyli pozycji po posortowaniu od najgorszych do najlepszych. Dzięki temu najgorszy osobnik zajmuje jedną jednostke obszaru koła ruletki, drugi - dwie jednostki, a najlepszy - n jednostek, gdzie n to rozmiar zbioru na którym stosuje się operator selekcji. Dzięki temu, że operator korzysta jedynie z relacji porządku między osobnikami $\mathbf{R}_{\lhd}$ (wynikającej z relacji porządku określonej na ich ocenach $\mathbf{R}_{\prec}$) nie występuje tu wcześniej opisany problem, więc operator jest użyteczny zarówno w problemach minimalizacji, jak i maksymalizacji.

Schemat działania operatora ruletkowego jest opisany za pomocą algorytmu 4.1.

Pierwszym krokiem działania operatora jest posortowanie populacji wejściowej tak, aby osobniki o większej wadze (czyli o większym prawdopodobieństwie wyboru) znalazły się bliżej początku posortowanego wektora *posortowane* (w linii 1). Następnie, obliczana jest suma wag wszystkich osobników, przechowywana w zmiennej $sumaWag$ (linia 2). W kolejnych krokach wybierana jest losowa wartość $wybrane$ z przedziału $\langle 0, sumaWag\rangle$ (w linii 3) i inicjalizowana jest zmienna pomocniczna $k = 0$ (w linii 4). W pętli, rozpoczynającej się w linii 5 przeszukiwany jest wektor *posortowane*, w poszukiwaniu takiego osobnika, którego waga jest mniejsza niż $wybrane - k$, ale większa lub równa k (warunek w linii 6). Jeżeli taki osobnik zostanie znaleziony, to zostaje on zwrócony jako wynik działania operatora (linia 7), a w przeciwnym wypadku jego waga zostaje dodana do wartości k (w linii 9). Jeśli żaden z osobników nie spełnia tych

2. Prosta zmiana znaku kryterium również sprowadza problem minimalizacji do problemu maksymalizacji, jednak nie pozwala na zastosowanie operatora ruletkowego, ponieważ waga osobnika musi być dodatnia. Alternatywne podejście, polegające na użyciu odwrotności wartości kryterium również nie zawsze jest możliwe do użycia, ponieważ wartość oceny może wynosić 0.

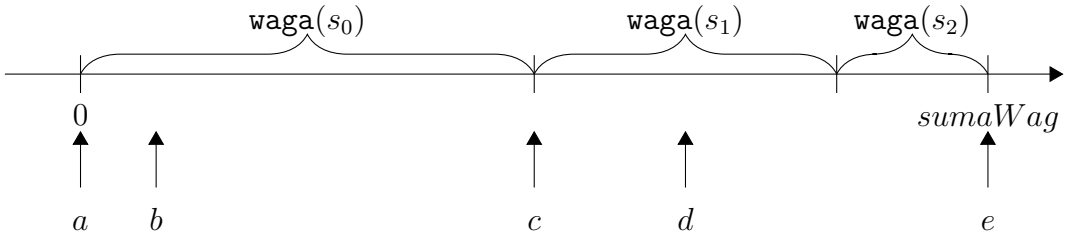
Algorytm 4.1 Schemat działania ruletkowego operatora wyboru

Wejście:
populacja ▷ Populacja z której wybierany jest osobnik
Wyjście:
wybrany osobnik
Parametry:
`waga` ▷ Funkcja wagi, np. funkcja oceny lub ranga

```
 0: operator opRuletkowy(populacja)
 1:     var posortowane ← populacja posortowana malejąco po wartości funkcji waga
 2:     var sumaWag ← ∑_{s∈populacja} waga(s)
 3:     var wybrane ← random(⟨0, sumaWag⟩)
 4:     var k ← 0
 5:     for s ∈ posortowane do                    ▷ w kolejności określonej w linii 1
 6:         if wybrane ⩾ k ∧ wybrane < k + waga(s) then
 7:             return s
 8:         else
 9:             k ← k + waga(s)
10:         end if
11:     end for
12:     return ostatni element posortowane
13: end operator
```

warunków, to zwracany jest ostatni osobnik, z najmniejszą wagą (linia 12).

Rys. 4.1. Działanie ruletkowego operatora wyboru



Działanie tak opisanego operatora obrazuje rysunek 4.1. Na osi zaznaczono przykładowe wagi 3 osobników, w sytuacji w której $posortowane = [s_0, s_1, s_2]$ oraz kilka przykładowych wartości (a, b, c, d, e) jakie może przyjąć $wybrane$. Jeżeli $wybrane = a$ lub $wybrane = b$, to zwrócony zostanie osobnik s_0, ponieważ k będzie równe 0 i $wybrane$ znajdzie się w odpowiednim przedziale. W przeciwnym wypadku k zostanie zwiększone do wartości $\texttt{waga}(s_0)$. Wtedy, jeżeli $wybrane = c$ lub $wybrane = d$, to zwrócony zostanie osobnik s_1, z tego samego powodu. Jeżeli jednak $wybrane = e$, to k przyjmie kolejno wartości $\texttt{waga}(s_0)+\texttt{waga}(s_1)$ i $\texttt{waga}(s_0)+\texttt{waga}(s_1)+\texttt{waga}(s_2)$. W takiej sytuacji warunek sprawdzany w pętli nie zostanie spełniony, więc zwrócony zostanie osobnik s_2.

Drugą powszechnie stosowaną metodą selekcji jest **operator turniejowy** (*ang. tourney select*). Jego działanie polega na wylosowaniu z równym pradopodobieństwem pewnej ilości osobników i zwróceniu najlepszego spośród nich. Zbiór kandydatów do

zwrócenia nazywa się *turniejem*, a jego rozmiar (czyli ilość rozwiązań które są między sobą porównywane) nazywany jest *rozmiarem turnieju* i jest parametrem operatora.

Algorytm 4.2 Schemat działania turniejowego operatora wyboru

```
    Wejście:
        populacja          ▷ Populacja z której wybierany jest osobnik
    Wyjście:
        wynik                                  ▷ Wybrany osobnik
    Parametry:
        n                                      ▷ Rozmiar turnieju
 0: operator opTurniejowy(populacja)
 1:     var turniej ← ∅
 2:     while |turniej| < n do
 3:         turniej ← turniej ∪ random(populacja \ turniej)
 4:     end while
 5:     var wynik ← NULL
 6:     for s ∈ turniej do
 7:         if wynik = NULL ∨ (wynik, s) ∈ R◁ then
 8:             wynik ← s
 9:         end if
10:     end for
11:     return wynik
12: end operator
```

Schemat działania operatora turniejowego jest opisany za pomocą algorytmu 4.1.

Działanie operatora rozpoczyna się od inicjalizacji zbioru *turniej* na zbiór pusty (w linii 1). Następnie, póki zbiór ten nie osiągnie rozmiaru n (linia 2) dodawane są do niego osobniki losowo wybierane (bez powtórzeń) z populacji (w linii 3). W kolejnym kroku *turniej* jest przeszukiwany pod kątem rozwiązania lepszego od pozostałych. Polega to na zainicjalizowaniu pomocniczej zmiennej *wynik* na wartość $NULL$ (czyli abstrakcyjną, pustą wartość, linia 5) i sprawdzeniu każdego z osobników ze zbioru *turniej* (linia 6). Jeżeli zmienna *wynik* ma wartość $NULL$ (czyli pętla dopiero się rozpoczęła) lub rozwiązanie s sprawdzane w danym momencie jest lepsze od wartości tej zmiennej (co sprowadza się do sprawdzenia, czy dana para osobników jest w relacji lepszego dopasowania $\mathbf{R}_{\lhd}$), to wartość *wynik* jest nadpisywana przez s (linie 7 i 8). Wynikiem działania operatora jest znaleziony w ten sposób najlepszy osobnik spośród losowo wybranych, czyli wartość *wynik*, zwracana w linii 11.

Kryterium stopu

Jak zostało opisane w podsekcji 1.2.7 zadaniem kryterium stopu jest przerwanie działania heurystyki w wybranym momencie. Ma to na celu ograniczenie czasu działania procesu do skończonej wartości.

Najprostszą implementacją tego warunku jest zatrzymanie heurystyki po przetworzeniu określonej liczby pokoleń. Działanie takiego operatora opisane jest za pomocą algorytmu 4.3

Algorytm 4.3 Warunek zatrzymania po określonej liczbie pokoleń

Dostępne zmienne:
pokolenie ▷ Numer obecnego pokolenia, liczony od 1
Parametry:
`max` ▷ Liczba pokoleń które należy przetworzyć
Wyjście:
1 oznaczające, że należy przerwać działanie heurystyki, lub 0 w przeciwnym wypadku

```
0: operator krytStopu
1:     if pokolenie ⩽ max then
2:         return 1
3:     else
4:         return 0
5:     end if
6: end operator
```

Jego działanie jest trywialne i sprowadza się do sprawdzenia odpowiedniego warunku (w linii 1) i zwrócenia 1 (linia 2) lub 0 (linia 4) w zależności od jego prawdziwości.

Wykorzystane implementacje i oznaczenia

W niniejszej pracy wykorzystano operator losowy, ruletkowy ważony rangą, oraz operatory turniejowe o rozmiarze 2 i 3. Wykorzystywane są one wraz z operatorem selekcji naturalnej opisanym algorytmem 3.1 oraz operatorami selekcji płciowej opisanymi za pomocą algorytmów 3.2 i 3.3. Ponadto, niezależnie od eksperymentu wykorzystano kryterium zatrzymania opisane w poprzednim paragrafie.

Tabela 4.1 przedstawia oznaczenia różnych elementów wraz z ich opisem. Przykładowo, zapis stdGenSel($\top$, R, TS(2)) oznacza standardowy operator selekcji naturanej, parametryzowany przez `plecMaZnaczenie` = $\top$, wykorzystujący operator losowy jako `opWyboru1` i operator turniejowy (z rozmiarem turnieju równym 2) jako `opWyboru2`.

W pracy przyjęto co najwyżej 2 płcie osobników. Jeżeli płec była tylko jedna, to $\mathcal{G} = \{\odot\}$, w przeciwnym wypadku $\mathcal{G} = \{\male, \female\}$.

4.2.2. Implementacja problemu komiwojażera

Model problemu komiwojażera może zostać przedstawiony w postaci macierzy incydencji grafu reprezentującego rozkład punktów na mapie. Macierz ta (oznaczana jako $\mathcal{M}_{TSP}$) ma wymiary $p \times p$ gdzie p to liczba punktów.

$$\mathcal{M}_{TSP} = \begin{pmatrix} m_{1,1} & m_{1,2} & \cdots & m_{1,p} \\ m_{2,1} & m_{2,2} & \cdots & m_{2,p} \\ \vdots & \vdots & \ddots & \vdots \\ m_{p,1} & m_{p,2} & \cdots & m_{p,p} \end{pmatrix}$$

Wartość $m_{x,y}$ oznacza wagę krawędzi łączącej $x$ty i $y$ty wierzchołek grafu, czyli odległość między punktami o numerach x i y.

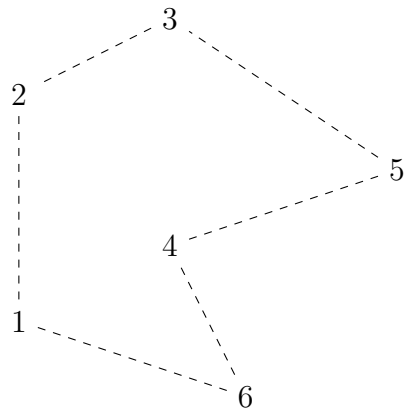
Tabela 4.1. Oznaczenia operatorów

Oznaczenie	**Znaczenie**
R	Losowy operator wyboru $random(X)$
RS	Ruletkowy operator wyboru
TS(n)	Turniejowy operator wyboru o rozmiarze turnieju n
natSel(X)	Operator selekcji naturalnej opisany algorytmem 3.1, korzystający z operatora wyboru X
stdGenSel(p, W1, W2)	Operator selekcji płciowej używany do realizacji dotychczasowych rozwiązań, opisany algorytmem 3.2, parametryzowany przez `plecMaZnaczenie` $= p$, `opWyboru1` $= W1$ oraz `opWyboru2` $= W2$
harem(a, b, WA, WB, WP)	Proponowany haremowy operator selekcji płciowej, opisany algorytmem 3.3 (bez zmian opisanych algorytmem 3.4), wykorzystujący `liczbaAlfa` $= a$, `wspBeta` $= b$, `opWyboruAlfa` $= WA$, `opWyboruBeta` $= WB$, `opWyboruPartnerow` $= WP$.

Wykorzystano reprezentację osobnika przez $p$elementowy wektor wartości ze zbioru $\{1, 2, \ldots, p\}$. Interpretacja tego, że na $i$tej pozycji w wektorze znajduje się wartość j jest taka, że jako numer $i$tego odwiedzanego punktu wynosi j.

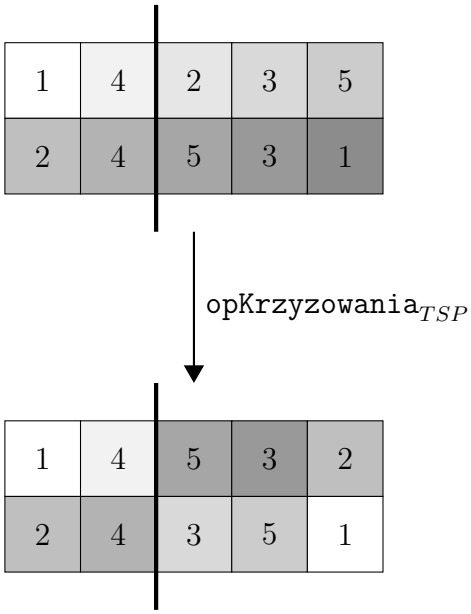
Przykładowo, dla punktów rozmieszczonych w sposób przedstawiony na rysunku 4.2, osobnik reprezentowany przez wektor $[1, 6, 4, 5, 3, 2]$ oznacza ścieżkę zaznaczoną na tym samym rysunku przerywaną linią, rozpoczynającą i kończącą się w punkcie 1.

Rys. 4.2. Przykładowe rozwiązanie problemu komiwojażera



Funkcją oceny w tym problemie jest długość ścieżki. Formalnie, $\texttt{funkcjaOceny}_{TSP}$ jest wyrażona jako suma wag krawędzi łączących punkty w kolejności określonej wektorem, co można zapisać w postaci równania 4.1. Pierwsza część równania, występująca przed sumowaniem po odpowiednich indeksach oznacza długość krawędzi między ostatnim, a pierwszym punktem zapisanym w wektorze, dzięki czemu rozwiązania opisujące ścieżki z tą samą kolejnością, ale rozpoczynające się w różnych punktach, np. $[1, 6, 4, 5, 3, 2]$ i $[5, 3, 2, 1, 6, 4]$ mają tą samą ocenę.

Rys. 4.3. Przykład działania operatora krzyżowania dla problemu komiwojażera



$$\texttt{funkcjaOceny}_{TSP}([x_1, x_2, \ldots, x_p]) = p_{x_p,x_1} + \sum_{i=1}^{p-1} m_{x_i,x_{i+1}} \tag{4.1}$$

W niniejszej pracy wykorzystano zbiór punktów *sahara* [1] opisujących położenie 29 miast leżących na zachodniej Saharze. Znane jest globalne optimum dla tego zadania, wynoszące 27603.

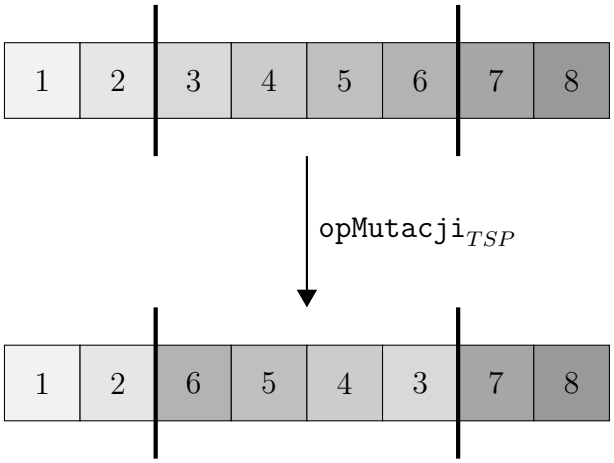
Wykorzystany operator krzyżowania jest zbliżony do tego opisanego w sekcji 1.1. Polega na wybraniu losowego punktu przecięcia wektorów i wymianie ich podwektorów. Różni się on podanego przykładu tym, że elementy wektorów nie powinny się w nich powtarzać. Aby spełnić to ograniczenie operator tworzy potomków poprzez wykorzystanie pierwszych części wektorów-rodziców i dołączenie do nich niewykorzystanych punktów. Punkty te dołączane są najpierw z podwektora drugiego rodzica za punktem przecięcia, a jeżeli po tym dalej brakuje niektórych z nich, są wybierane od jego początku. Zostało to zobrazowane na rysunku 4.3, gdzie na górze pokazano wektory wejściowe (rodziców), grubą linią zaznaczono punkt przecięcia, a na dole przedstawiono wynik krzyżowania (potomków).

Operator mutacji wykorzystany w tej pracy nosi angielską nazwę *Reverse Sequence Mutation* [2], co można przetłumaczyć na *mutację przez zamianę podciągów*. Jego działanie polega na wybraniu dwóch losowych punktów przecięcia, przez co dzielimy cały wektor na 3 części. Osobnik zmutowany powstaje poprzez zachowanie kolejności elementów w pierwszej i trzeciej części oraz odwrócenie kolejności elementów w drugiej części. Jest to zobrazowane na rysunku 4.4.

4.2.3. Implementacja problemu plecakowego

Model binarnego problemu plecakowego można przedstawić w postaci ciągu $\mathcal{M}_{PLECAK}$ p par (k, w) oraz wartości W określającej maksymalny rozmiar plecaka.

Rys. 4.4. Przykład działania operatora mutacji dla problemu komiwojażera



$$\mathcal{M}_{PLECAK} = (((k_1, w_1), (k_2, w_2), \ldots, (k_p, w_p)), W)$$

Wartość k_i oznacza koszt $i$tego przedmiotu, a w_i - jego wagę.

Wykorzystano reprezentację osobnika przez $p$elementowy wektor binarny (czyli taki, którego wartości to elementy zbioru $\{0, 1\}$). Interpretacja tego, że na $i$tej pozycji w wektorze występuje 1 jest taka, że $i$ty przedmiot zostaje zapakowany do plecaka, podczas gdy wartość 0 wskazuje na sytuację przeciwną.

Funkcja oceny dla tego problemu jest bardziej skomplikowana niż dla problemu komiwojażera. Główną jej składową jest łączy koszt wszystkich wybranych przedmiotów. Aby sprowadzić problem do problemu minimalizacji[3] jest ona mnożona przez -1. Dodatkowo, aby pokierować heurystyką tak, żeby ograniczenie na łączny rozmiar wybranych przedmiotów było spełnione, wprowadzono funkcję kary, polegającą na dodaniu do funkcji oceny iloczynu łącznego kosztu przedmiotów i różnicy między sumą ich obiętości, a maksymalnym rozmiarem plecaka (jeżeli ta różnica jest dodatnia). W ten sposób rozwiązania spełniające ograniczenia mają ocenę równą łącznemu kosztowi z ujemnym znakiem, te, które przekroczyły ograniczenie o 1 mają ocenę równą 0, itd. Zostało to zapisane za pomocą równań 4.2-4.4.

$$x = [x_1, x_2, \ldots, x_p] \tag{4.2}$$

$$\texttt{funkcjaOceny}_{PLECAK}(x) = (nadmiar(x) - 1) \times \sum_{i=1}^{p}(x_i \times k_i) \tag{4.3}$$

$$nadmiar(x) = \begin{cases} 0 & : \sum_{i=1}^{p}(x_i \times w_i) \leqslant W \\ (\sum_{i=1}^{p}(x_i \times w_i)) - W & : \sum_{i=1}^{p}(x_i \times w_i) > W \end{cases} \tag{4.4}$$

W tej pracy wykorzystano zbiór *medium* [3], opisujący 100 przedmiotów i maksymalny rozmiar plecaka wynoszący 27. Dla tak przedstawionego zagadnienia znane jest optimum, wynoszące -1137, co oznacza, że wybrane przedmioty mają łączny koszt 1137 i ograniczenie na rozmiar nie jest przekroczone.

3. Zdecydowany się na problem minimalizacji, aby uprościć implementację.

Zastosowano standardowy operator krzyżowania dla wektorów binarnych, opisany w rozdziale 1.1, zobrazowany rysunkiem 1.2. Działa on na zasadzie losowego rozcięcia wektorów na dwa podwektory i zamianie ich drugich części w celu uzyskania potomków.

Wykorzystany operator mutacji to adaptacyjna[4] wariacja na temat operatora przedstawionego w rozdziale 1.1 na rysunku 1.3. W zależności od tego, czy łączna objętość wybranych przedmiotów była większa, czy mniejsza, niż rozmiar plecaka, prawdopodobieństwa tego, że bit zostanie odwrócony (tzn. 0 zostanie zamienione na 1 lub na odwrót) były różne.

Zanim działanie tego operatora zostanie zdefiniowane formalnie, należy opisać pomocniczą funkcję $rzutuj(x, \langle a, b\rangle)$, zwracającą punkt leżący w przedziale $\langle a, b\rangle$ w odległości od obu punktów określonej przez argument x. Jej definicja jest zapisana w sygnaturze 13.

Definicja 13 Funkcja $rzutuj(\langle a, b\rangle, x)$

$$rzutuj : \langle 0, 1\rangle \times \mathbb{R}^2 \to \mathbb{R} \tag{4.5}$$

$$rzutuj(x, \langle a, b\rangle) = a + (b - a) \times x \tag{4.6}$$

Działanie operatora jest zasadniczo podobne do tego opisanego we wcześniejszym rozdziale - każdy z bitów wektora wejściowego może zostać odwrócony. Prawdopodobieństwo odwrócenia bitu jest zależne od tego, czy jest on równy 0, czy 1, oraz od tego, czy ograniczenie zostało przekroczone, czy też łączny rozmiar wybranych przedmiotów jest mniejszy niż rozmiar plecaka. Pierwsza z tych sytuacji jest umownie nazywana nadmiarem, a druga - niedomiarem.

W ogólności prawdopodbieństwo P odwrócenia bitu jest wyrażone wzorem 4.7.

$$P = rzutuj(czynnik, przedzial) \tag{4.7}$$

Sposób obliczania wartości *czynnik* jest zależny od sytuacji. Jeżeli obserwowany jest niedomiar, to jego wartość jest różnicą rozmiaru plecaka i sumy rozmiarów wybranych przedmiotów podzieloną przez rozmiar plecaka ($\frac{W-\sum_{i=1}^{p}(x_i \times w_i)}{W}$), czyli relatywną ilością niewykorzystanego miejsca. Wartość mianownika dla zbadanego problemu to 27 (ponieważ tyle wynosi rozmiar plecaka dla wykorzystanego zbioru danych). Jeżeli natomiast występuje nadmiar, to jest on tożsamy z wartością funkcji $nadmiar(x)$ (patrz: linia 4.4) podzieloną przez sumę różnicę sumy wszystkich wag i rozmiaru plecaka ($\frac{nadmiar(x)}{(\sum_{i=1}^{p} w_i)-W}$), czyli wartością nadmiaru znormalizowaną do przedziału $\langle 0, 1\rangle$. W tym przypadku wartość pod kreską ułamkową wynosi 1333, ponieważ suma wag przedmiotów w wybranym zbiorze danych wynosi 1360.

Sposób określania wartości *przedzial* również jest zależny od sytuacji. W tabeli 4.2 przedstawiony jest sposób wyboru odpowiedniego przedziału. Wartości w niej zawarte zostały arbitralnie dobrane tak, aby w sytuacji nadmiaru z większym prawdopodbieństwem odrzucać wybrane przedmioty, a w sytuacji niedomiaru - częściej dobierać nowe.

4. Określenie „adaptacyjny" oznacza w tym kontekście, że szczegóły działania operatora zmieniają się w zależności od sytuacji, w której jest wykorzystywany, dopasowując się do napotkanych warunków.

Tabela 4.2. Tablica wyboru wartości *przedzial*

		Sytuacja	
		Niedomiar	Nadmiar
Wartość bitu	0	$\langle 0.06, 0.1\rangle$	$\langle 0.05, 0.15\rangle$
	1	$\langle 0.05, 0.2\rangle$	$\langle 0.2, 0.7\rangle$

Intuicyjnie, tak zdefiniowany operator powinien z dużym prawdopodobieństwem zamieniać sytuacje nadmiaru na takie, które spełniają ograniczenia, a przynajmniej powodują znacznie mniejszy nadmiar, a w sytuacji niedomiaru powinien „dopełniać" plecak losowymi przedmiotami, aby uzyskać jak największy zysk (tzn. sumę kosztów przedmiotów).

4.3. Badania

W dalszej części pracy konkretny zestaw parametrów będzie nazywany konfiguracją. Wyrażenie „wynik eksperymentu dla konfiguracji wyniósł X" oznacza, że heurystyka została uruchomiona z parametrami (rozmiarem populacji, prawdopodobieństwami, itd.) określonymi przez konfigurację, a jej wynikiem było X. Należy pamiętać, że algorytm ewolucyjny to heurystyka losowa, więc wynik dla danej konfiguracji za każdym uruchomieniem może być inny.

2 etapy - jakie, każdy problem osobno

4.3.1. Określenie bazowej konfiguracji

krótko o eksploracji

Konfiguracja 1. Wartości wykorzystane podczas poszukiwania parametrów początkowych

Parametr	Zbiór wartości
`rozmiarPopulacji`	10, 20, 50
`wspDomieszek`	0, 0,1, 0,25, 0,5
(`prawdKrzyzowania`, `prawdMutacji`)	[0,6, 0,7, 0,8] × [0,1, 0,2, 0,3]
`max`	25, 50, 100
`opSelNat`	natSel(RS), natSel(TS(2)), natSel(TS(3))

Parametr	Wartość
`opSelPlciowej`	stdGenSel($\perp$, R, R)
Zbiór płci	$\mathcal{G} = \{\odot\}$

zakres, jak przy drugim powtórzeniu

wynikowe konfigi

Konfiguracja 2. Parametry używane w dalszych badaniach problemu komiwojażera

Parametr	**Wartość**
`opSelNat`	natSel(RS)
`opSelPlciowej`	stdGenSel($\perp$, R, R)
`rozmiarPopulacji`	50
`wspDomieszek`	0,0
`prawdKrzyzowania`	0,6
`prawdMutacji`	0,15
`max`	110

Konfiguracja 3. Parametry używane w dalszych badaniach problemu plecakowego

Parametr	**Wartość**
`opSelNat`	natSel(RS)
`rozmiarPopulacji`	60
`wspDomieszek`	0,1
`prawdKrzyzowania`	0,75
`prawdMutacji`	0,15
`max`	60

4.3.2. Badania porównawcze

co zbadano fallback do poprzednich tabel

Badane konfiguracje

plain; gga; sexual ga; uogólniony ; harem

Konfiguracja 4. Konfiguracja standardowego algorytmu ewolucyjnego

Operator selekcji płciowej	stdGenSel($\perp$, R, R)
Pozostałe parametry	Zgodne z konfiguracją 2 lub 3
Ilość powtórzeń	5
Zbiór płci	$\mathcal{G} = \{\odot\}$

Konfiguracja 5. Konfiguracja heurystyki GGA

Operator selekcji płciowej	stdGenSel($\top$, R, R)
Operator selekcji naturalnej	natSel(W)
Pozostałe parametry	Zgodne z konfiguracją 2 lub 3
Ilość powtórzeń	5
Zbiór płci	$\mathcal{G} = \{\male, \female\}$

Parametr	**Zbiór wartości**
W	RS, TS(2), TS(3)

Konfiguracja 6. Konfiguracja heurystyki SexualGA

Operator selekcji płciowej	stdGenSel($\perp$, `opWyboru1`, `opWyboru2`)
Operator selekcji naturalnej	natSel(R)
Pozostałe parametry	Zgodne z konfiguracją 2 lub 3
Ilość powtórzeń	5
Zbiór płci	$\mathcal{G} = \{\odot\}$

Parametr	Zbiór wartości
`opWyboru1`	RS, TS(2), TS(3)
`opWyboru2`	R, RS, TS(2), TS(3)

Konfiguracja 7. Konfiguracja heurystyki DSEA z uogólnionym operatorem selekcji płciowej

Operator selekcji płciowej	stdGenSel(p, `opWyboru1`, `opWyboru2`)
Pozostałe parametry	Zgodne z konfiguracją 2 lub 3
Ilość powtórzeń	5

Parametr	Zbiór wartości
$p =$ `plecMaZnaczenie`	$\top, \perp$
`opWyboru1`	RS, TS(2), TS(3)
`opWyboru2`	R, RS, TS(2), TS(3)
Zbiór płci	$\lvert\mathcal{G}\rvert \in \{1, 2\}$, zależnie od `plecMaZnaczenie`

Konfiguracja 8. Konfiguracja heurystyki DSEA z haremowym operatorem selekcji płciowej

Operator selekcji płciowej	harem(a, b, WA, WB, WP)
Pozostałe parametry	Zgodne z konfiguracją 2 lub 3
Ilość powtórzeń	5
Zbiór płci	$\mathcal{G} = \{\male, \female\}$

Parametr	Zbiór wartości
a	1, 3, 5
b	0, 0,1, 0,25
WA	R, RS, TS(2), TS(3)
WB	R, RS, TS(2), TS(3)

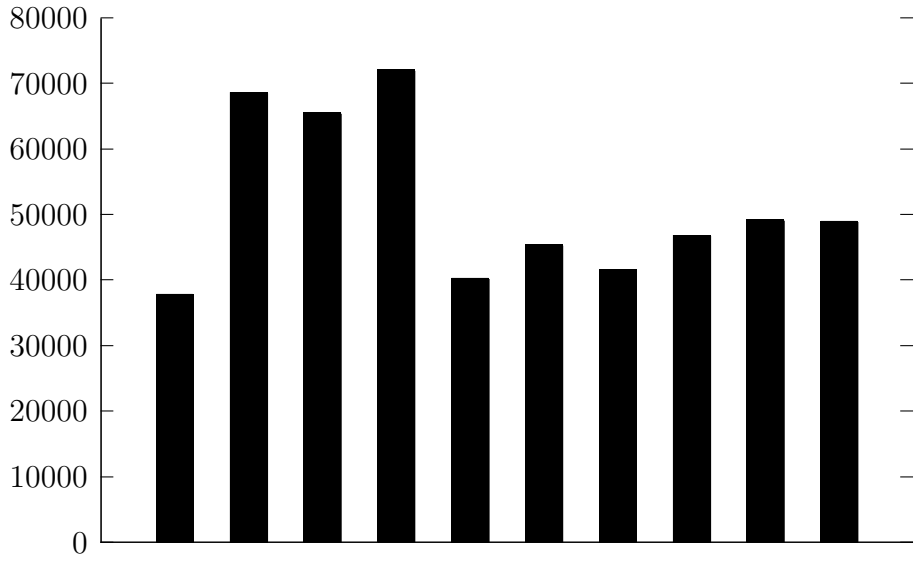
Parametr	Wartość
WP	R

Wyniki

Problem komiwojażera słupki i tabela z porównaniem plain/gga/sga/general true/false/harem

Tabela 4.3. My caption

Heurystyka	**Op. selekcji płciowej**	**Op. selekcji naturalnej**	**Ocena**
Alg. ewolucyjny	opNatSel(RS)	stdGenSel(⊥, R, R)	37830,7834±1865,7587
GGA	opNatSel(RS)	stdGenSel(⊤, R, R)	68582,5919±2240,5888
SexualGA	opNatSel(R)	stdGenSel(⊥, RS, RS)	65454,5598±4541,7191
		stdGenSel(⊥, RS, R)	72023,5817±5135,7596
DSEA	opNatSel(RS)	stdGenSel(⊥, RS, RS)	40199,6226±3708,8267
		stdGenSel(⊥, RS, R)	45328,6013±967,5844
		stdGenSel(⊤, RS, RS)	41555,6377±3146,2442
		stdGenSel(⊤, RS, R)	46760,4675±4193,2035
		harem(5, 0,1, RS, RS, R)	49180,7887±4249,2504
		harem(5, 0,1, RS, R, R)	48915,0643±799,6703



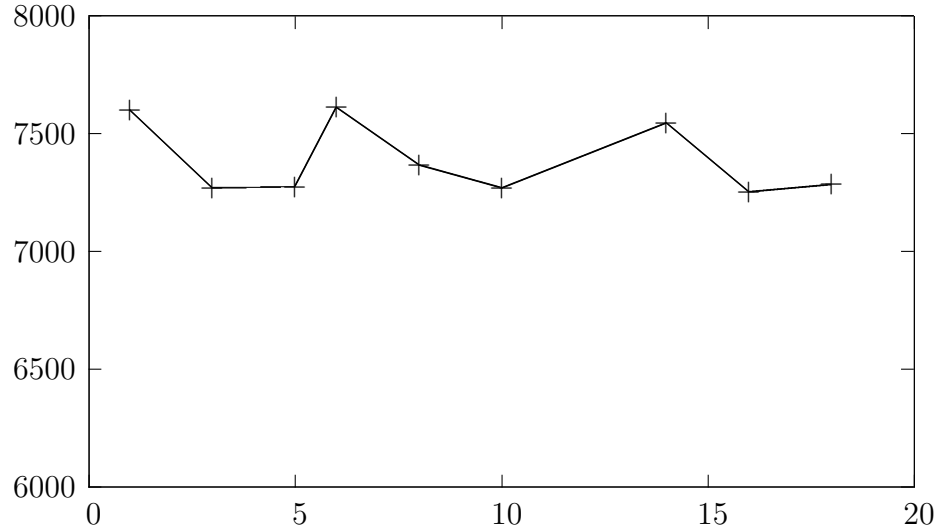
W tabeli 4.4 przedstawiono 12 najlepszych wyników uzyskanych przy użyciu konfiguracji 8. Zestawienie pełnych wyników można znaleźć w dodatku A.

Można zaobserwować, że heurystyka działała najskuteczniej, kiedy jeden lub oba operatory wyboru były takie same jak użyty w operatorze selekcji naturalnej. Ponadto, najlepsze wyniki uzyskano dla większych liczb osobników alfa i beta.

Warto zauważyć, że najlepszy i najbardziej stabilny wynik uzyskano wybierając osobniki alfa operatorem ruletkowym, a osobniki beta i partnerów losowym. Może to oznaczać, że drastyczna selekcja rodziców nie jest nieuzasadniona, a osobniki alfa mają najbardziej korzystne cechy.

Wykres 4.5 przedstawia średnią funkcję oceny dla różnych liczb samców. Liczba ta została obliczona jako $\lceil \texttt{liczbaAlfa} + \texttt{wspBeta} \times 50 \rceil$, ponieważ `rozmiarPopulacji` wynosił 50. Jak widać, nie ma żadnej jednoznacznej zależności między tą liczbą, a wynikami, z czego można wnioskować, że to operatory wyboru (losowy lub ruletkowy w większości przypadków) decydowały o skuteczności.

Rys. 4.5. Wykres średniej oceny od łącznej liczby osobników alfa i beta



Rys. 4.6. Wykres średniej oceny od łącznej liczby osobników alfa i beta

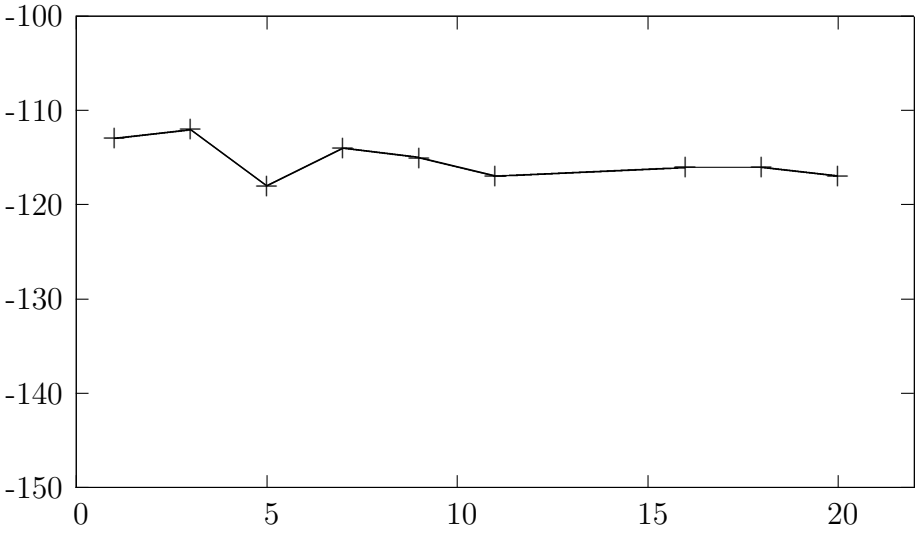


Tabela 4.4. Wyniki heurystyki DSEA z operatorem haremowym

a	**b**	**WA**	**WB**	**Ocena**
5	0,1	RS	R	48915,0643±799,6703
5	0,1	RS	RS	49180,7887±4249,2504
3	0,25	RS	RS	49744,9081±2768,5553
3	0	RS	R	49984,0394±3413,5860
5	0,1	RS	TS(3)	49985,4366±4012,8489
3	0,25	RS	R	50011,9993±3104,8017
3	0	RS	RS	50296,3335±4102,9637
5	0	RS	R	50317,9438±3977,9757
3	0,1	RS	TS(3)	50425,7944±2674,6857
5	0,25	RS	RS	50581,0518±3533,9353
5	0	RS	TS(3)	50788,1319±1542,8235
5	0	RS	TS(2)	51156,4782±3617,3294

Tabela 4.5. Wyniki heurystyki DSEA z operatorem haremowym

a	**b**	**WA**	**WB**	**Ocena**
5	0,25	RS	RS	-1144,2000±20,4294
3	0	RS	TS(2)	-1142±19,3080
5	0	RS	TS(2)	-1141,8000±22,3195
5	0	RS	TS(3)	-1134±27,4591
1	0,25	RS	RS	-1130,4000±16,2800
5	0,25	RS	R	-1129±11,3490
5	0	RS	R	-1128,2000±32,6827
5	0,1	RS	R	-1128±23,7571
3	0,1	RS	TS(2)	-1127,8000±26,4983
3	0,1	RS	RS	-1126,6000±8,0647
3	0,25	RS	TS(3)	-1125,4000±19,9660
3	0,25	RS	RS	-1125,4000±29,6554

Rozdział 5

Wnioski i spostrzeżenia

Rozdział 6

Dalsze drogi rozwoju

wywalić listy (?)
zmienić styl bibliografii, URL z online resource, uzupełnić linki
znormalizować referencje figure: i vector:

Spis rysunków

Spis tablic

Spis sygnatur

Spis algorytmów

Bibliografia

[1] Western sahara tsp dataset.

[2] Abdoun O., Abouchabaka J., Tajani C. Analyzing the performance of mutation operators to solve the travelling salesman problem. *arXiv preprint arXiv:1203.3099*, 2012.

[3] Barker W.

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

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

[6] Lindenfors P., Tullberg B., Biuw M. Phylogenetic analyses of sexual selection and sexual size dimorphism in pinnipeds. *Behavioral Ecology and Sociobiology*, 52(3):188–193, 2002.

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

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

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

Dodatek A

Procedura eksploracji

poprawić W ramach pierwszej części badań wykorzystano procedurę eksperymentalną, nazywaną procedurą eksploracji (lub krócej, eksploracją).

Jest ona wykorzystywana w sytuacji, w której pojawia się potrzeba przeszukania dużej przestrzeni parametrów. Jest to często niemożliwe, ponieważ pojedyncza ewaluacja zestawu parametrów trwa relatywnie długo, a ilość kombinacji wartości parametrów jest zbyt duża.

Aby obejść ten problem, w tej pracy zdecydowano się na iteracyjne podejście przeszukujące przestrzeń konfiguracji wzdłuż wymiarów. Zostało ono wyjaśnione przy pomocy algorytmu A.1.

Algorytm A.1 Procedura eksploracji

Parametry:

$X_1, X_2, \ldots, X_p$ ▷ Zestaw p uporządkowanych zbiorów wartości parametrów $X_i = [x_{i,1}, x_{i,2}, \ldots, x_{i,|X_i|}]$, gdzie $x_{i,j}$ to $j$ta wartość $i$tego parametru

R ▷ Ilość nawrotów procedury

I ▷ Ilość powtórzeń eksperymentu dla każdej konfiguracji

statystyka ▷ Operator, przyjmujący zbiór wyników, a zwracający jakąś ich statystykę, np. średnią, medianę wartość minimalną lub maksymalną

Wejście:

cialo ▷ Ciało eksperymentu, tj. operator wykonania pojedynczego eksperymentu, przyjmujący zestaw parametrów $(P_1, P_2, \ldots)$ $(P_i \in X_i)$, a zwracający jakąś miarę jakości działania; dla uproszczenia w algorytmie przyjęto, że zwracaną miarę należy zminimalizować.

Wyjście:

$konfiguracja$ ▷ Najlepsza konfiguracja znaleziona w procesie eksploracji, tj. p wartości $x_{i,j}$ o różnych indeksach i

```
 0: operator EKSPLORUJ(cialo)
 1:     var konfiguracja = [x_{1,1}, x_{2,1}, ..., x_{p,1}]
 2:     for nawrot ← 1 to R do
 3:         for parametr ← 1 to p do
 4:             var statystyki = [NULL, NULL, ...] ▷ Tablica o rozmiarze |X_parametr|
 5:             for i ← 1 to |X_parametr| do
 6:                 konfiguracja[parametr] ← x_{parametr,i}
 7:                 var wyniki ← ∅
 8:                 for powtorzenie ← 1 to I do
 9:                     wyniki ← wyniki ∪ {cialo(konfiguracja)}
10:                 end for
11:                 statystyki[i] ← statystyka(wyniki)
12:             end for
13:             var najlepszaWartosc ← arg min_i statystyki[i]
14:             konfiguracja[parametr] ← x_{parametr,najlepszaWartosc}
15:         end for
16:     end for
17:     return konfiguracja
18: end operator
```

Zapis $a[b]$ oznacza element tablicy a o indeksie b

Procedura eksploracji jest zapisana jako operator, ponieważ zwraca pewną wartość (więc nie jest procedurą), która za każdym razem może być inna (więc nie jest funkcją).

Jej działanie zaczyna się od określenia początkowej konfiguracji, czyli zestawu parametrów. W tym celu wykorzystywane są pierwsze elementy każdego z zestawów wartości. Wybór konkretnie tych wartości jest zupełnie arbitralny.

Następnie wykonywane jest R iteracji nazywanych *nawrotami*. Ma to na celu upew-

nienie się, że dla parametrów wybranych w dalszej części każdego nawrotu wartości wybrane wcześniej faktycznie są jak najlepsze.

W każdym nawrocie badane są wszystkie zestawy wartości parametrów, w kolejności określonej przez parametry procedury. Dla każdego z nich sprawdzane są wszystkie elementy. Dzieje się to przez zmianę odpowiedniej wartości w konfiguracji na element badany w danym momencie.

Po takim podstawieniu ciało eksperymentu wykonywane jest I razy, a wyniki tych wykonań zbierane są do zbioru *wyniki*. Po odpowiedniej liczbie powtórzeń obliczana jest statystyka tego zbioru. W tym celu wykorzystywany jest parametr procedury *statystyka*. Jej wartość jest zapisywana tak, żeby można było określić dla której wartości parametru została ona określona.

Kiedy wszystkie wartości parametru zostaną zbadane, wybiera się najlepszą z nich, oceniając je po wartości statystyk. Wartość ta zostaje zapisana w konfiguracji używanej do badania kolejnego z parametrów.

referencje do linii

Dodatek A

Przebieg procedury eksploracji

A.1. Problem komiwojażera

A.1.1. Znalezienie konfiguracji początkowej

Przebieg

Tabela A.1 przedstawia przebieg wykorzystanej procedury. Lewa kolumna grupuje znalezione parametry w ramach jednego nawrotu eksploracji. Kolejne dwie, czytane od góry, określają jakie wartości zostały kolejno znalezione.

Jak widać, w pierwszym nawrocie znaleziono prawie najlepszą konfigurację. W drugim, udało się odnaleźć prawdopodobieństwa mutacji i krzyżowania które dawały lepsze wyniki. Trzeci nawrót potwierdził, że znaleziona konfiguracja jest najlepsza z przeszukiwanych.

Tabela A.1. Przebieg procedury eksploracji poszukującej parametrów początkowych

	Parametr	Określona wartość
Nawrót 1	`rozmiarPopulacji`	50
	`wspDomieszek`	0,0
	(`prawdKrzyzowania`, `prawdMutacji`)	(0,8, 0,3)
	`max`	100
	`opSelNat`	natSel(RS)
Nawrót 2	`rozmiarPopulacji`	50
	`wspDomieszek`	0,0
	(`prawdKrzyzowania`, `prawdMutacji`)	(0,6, 0,1)
	`max`	100
	`opSelNat`	natSel(RS)
Nawrót 3	`rozmiarPopulacji`	50
	`wspDomieszek`	0,0
	(`prawdKrzyzowania`, `prawdMutacji`)	(0,6, 0,1)
	`max`	100
	`opSelNat`	natSel(RS)

Wyniki

Na tym etapie eksperymentów nie udało się znaleźć globalnego optimum. W tabeli A.2 przedstawiono wyniki wszystkich powtórzeń dla konfiguracji znalezionej w procesie eksploracji. Średnia różnica między znalezionym, a najlepszym możliwym rozwiązanie wynosiła około 13 tysięcy, podczas gdy optimum miało wartość około 27 i pół tysiąca. Oznacza to, że znajdowane rozwiązania były około półtora raza gorsze niż poszukiwane.

Heurystyka zachowywała się dość stabilnie, tzn. za każdym uruchomieniem zwracała podobny wynik. Odchylenie standardowe wyniosło około 2 tysiące, czyli mniej więcej 5% średniej.

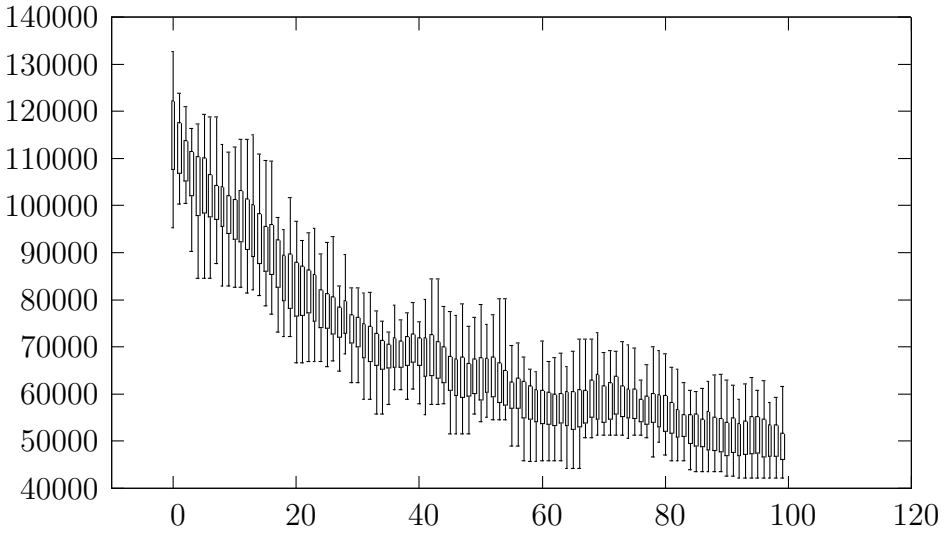
Na rysunku A.1 przedstawiono wykres trzeciego z pięciu wykonanych przebiegów. Można zaobserwować, że zarówno średnia, jak i najlepsze i najgorsze rozwiązanie spadają na przestrzeni wszystkich populacji, co oznacza, że wyniki sukcesywnie się polepszały.

Jednocześnie da się zauważyć, że wariancja wariancja utrzymywała się na dość stałym poziomie, z czego można wnioskować, że różnorodność genetyczna była zachowana.

Tabela A.2. Wyniki heurystyki parametryzowanej znalezioną konfiguracją

Numer powtórzenia	**Wynik**	**Różnica względem optimum**
1	42129,167	14526,167
2	42808,0792	15205,0792
3	36656,3316	9053,3316
4	42174,1093	14571,1093
5	41143,6598	13540,6598
Średnia	40982,2694	13379,2694
Odchylenie standardowe	2227,5198	
Optimum	**27603**	

Rys. A.1. Wykres przebiegu trzeciego powtórzenia znalezionej konfiguracji



A.1.2. Poprawa konfiguracji początkowej

Przebieg

W tabeli A.3 przedstawiono przebieg procedury eksploracji, korzystając z tej samej reprezentacji co na poprzednim etapie.

Rozmiar populacji i współczynnik domieszek nie zmienił się względem konfiguracji z pierwszego etapu. W pierwszym nawrocie określono, że nieznacznie mniejsze (0,58) prawdopodobieństwo krzyżowania i większa liczba pokoleń dają lepsze wyniki. Drugi nawrót wykazał, że pierwotne prawdopodobieństwo krzyżowania daje lepsze wyniki, za to zwiększone zostało prawdopodobieństwo mutacji.

Tabela A.3. Przebieg procedury eksploracji poszukującej parametrów początkowych

	Parametr	Określona wartość
Nawrót 1	`rozmiarPopulacji`	50
	`wspDomieszek`	0,0
	(`prawdKrzyzowania`, `prawdMutacji`)	(0,58, 0,1)
	`max`	110
Nawrót 2	`rozmiarPopulacji`	50
	`wspDomieszek`	0,0
	(`prawdKrzyzowania`, `prawdMutacji`)	(0,6, 0,15)
	`max`	110

Wyniki

W tabeli A.4 przedstawiono wyniki działania heurystyki dla konfiguracji 2 znalezionej w tym etapie.

Średnia ocena jest niższa (a więc lepsza) od średniej z poprzedniego etapu. Przeciętna różnica między znalezionym rozwiązaniem, a optimum spadła do około 9,5 tysiąca, a odchylenie standardowe nieznacznie (o około pół tysiąca) wzrosło.

Oznacza to, że udało się uzyskać lepsze wyniki kosztem nieznacznego pogorszenia stabilności heurystyki.

Tabela A.4. Wyniki heurystyki parametryzowanej poprawioną konfiguracją

Numer powtórzenia	Wynik	Różnica względem optimum
1	39592,0623	11989,0623
2	34292,9277	6689,9277
3	37904,1809	10301,1809
Średnia	37263,0570	9660,0570
Odchylenie standardowe	2707,1178	
Optimum	**27603**	

A.2. Problem plecakowy

A.2.1. Znalezienie konfiguracji początkowej

Przebieg

Tabela A.5 przedstawia przebieg wykorzystanej procedury.

Jak widać, w pierwszym nawrocie znaleziono prawie najlepszą konfigurację, która tylko nieznacznie zmieniła się w kolejnym. Zmiany obejmowały prawdopodobieństwo krzyżowania oraz współczynnik domieszek. Trzeci nawrót nie zmienił odnalezionej konfiguracji.

Tabela A.5. Przebieg procedury eksploracji poszukującej parametrów początkowych

	Parametr	**Określona wartość**
Nawrót 1	`rozmiarPopulacji`	50
	`wspDomieszek`	0,25
	(`prawdKrzyzowania`, `prawdMutacji`)	(0,6, 0,1)
	`max`	50
	`opSelNat`	natSel(RS)
Nawrót 2	`rozmiarPopulacji`	50
	`wspDomieszek`	0,1
	(`prawdKrzyzowania`, `prawdMutacji`)	(0,8, 0,1)
	`max`	50
	`opSelNat`	natSel(RS)
Nawrót 3	`rozmiarPopulacji`	50
	`wspDomieszek`	0,1
	(`prawdKrzyzowania`, `prawdMutacji`)	(0,8, 0,1)
	`max`	50
	`opSelNat`	natSel(RS)

Wyniki

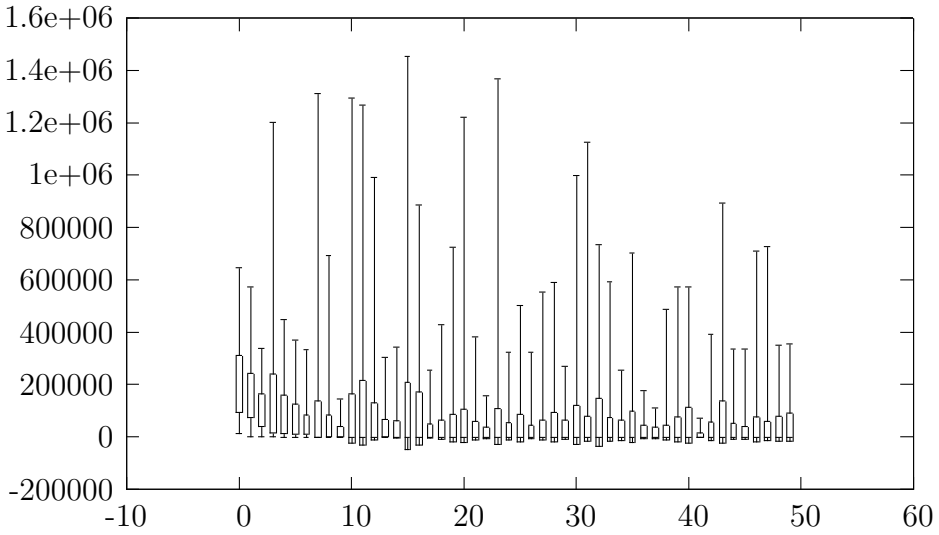
Jak zostało pokazane w tabeli A.6, wyniki dla tak określonego zestawu parametrów są bardzo dobre - najgorszy z nich był większy od optimum o niecałe 6 %. W ostatnim z przebiegów udało się znaleźć optimum globalne.

Na rysunku A.2 pokazany został wykres tego przebiegu. Można zaobserwować, że minimum zostało znalezione dość szybko, a dalsze działanie heurystyki przeszukiwało przestrzeń rozwiązań, często pogarszając wyniki.

Tabela A.6. Wyniki heurystyki parametryzowanej znalezioną konfiguracją

Numer powtórzenia	**Wynik**	**Różnica względem optimum**
1	-1107	66
2	-1156	17
3	-1115	58
4	-1120	53
5	-1173	0
Średnia	-1134,2000	38,8000
Odchylenie standardowe	28,6827	
Optimum	**-1173**	

Rys. A.2. Wykres przebiegu piątego powtórzenia znalezionej konfiguracji



A.2.2. Poprawa konfiguracji początkowej

Przebieg

skomentuj

Tabela A.7. Przebieg procedury eksploracji poszukującej parametrów początkowych

	Parametr	**Określona wartość**
Nawrót 1	`rozmiarPopulacji`	60
	`wspDomieszek`	0,1
	(`prawdKrzyzowania`, `prawdMutacji`)	(0,75, 0,1)
	`max`	50
Nawrót 2	`rozmiarPopulacji`	60
	`wspDomieszek`	0,1
	(`prawdKrzyzowania`, `prawdMutacji`)	(0,75, 0,15)
	`max`	60

Wyniki

Blah, blah, here go results

Tabela A.8. Wyniki heurystyki parametryzowanej poprawioną konfiguracją

Numer powtórzenia	**Wynik**	**Różnica względem optimum**
1	-1127	46
2	-1161	12
3	-1131	42
Średnia	-1139,6667	33,3334
Odchylenie standardowe	18,5831	
Optimum	**-1173**	

poszło lepiej niż init

Dodatek A

Pełne wyniki badań