Uniwersytet Ekonomiczny we Wrocławiu
Wydział Zarządzania, Informatyki i Finansów

Paweł Wicher

„Rozwiązywanie problemu komiwojażera za pomocą algorytmu genetycznego”

Praca magisterska

prof. dr hab. Juliusz Siedlecki

Wrocław 2008

Oświadczam, że pracę niniejszą przygotowałem samodzielnie. Wszystkie dane, istotne myśli i sformułowania pochodzące z literatury (przytoczone dosłownie lub niedosłownie) są opatrzone odpowiednimi odsyłaczami. Praca ta nie była w całości ani w części, która zawierałaby znaczne fragmenty przedstawione w pracy jako oryginalne (wyniki badań empirycznych, obliczenia, spostrzeżenia, oceny, wnioski, propozycje itp.), przez nikogo przedłożona do żadnej oceny i nie była publikowana.

Spis treści

Wrocław University of Economics
Faculty of Management, Computer Science and Finance

Paweł Wicher

„Solving Travelling Salesman Problem using genetic algorithm”

Master's thesis

prof. dr hab. Juliusz Siedlecki

Wrocław 2008

Table of contents

Wstęp

Celem pracy jest zbudowanie modelu algorytmu genetycznego do rozwiązywania problemu komiwojażera oraz jego implementacja komputerowa. Kolejnym etapem będzie przeprowadzenie szeregu symulacji, z wykorzystaniem stworzonej aplikacji, na podstawie 3 przykładowych problemów komiwojażera.

Problem komiwojażera należy do grupy klasycznych zadań optymalizacyjnych, często spotykanych w praktyce. Z problemem tym można się spotkać w wielu dziedzinach, takich jak logistyka czy projektowanie układów elektronicznych. Powstało wiele metod rozwiązywania tego zadania, które przybliżają rozwiązanie optymalne. Jedną z takich metod jest wykorzystanie algorytmów genetycznych, które będą omówione w tej pracy w kontekście zastosowania w problemie komiwojażera.

Algorytmy genetyczne w sposób naturalny realizują proces optymalizacji, zgodnie z zasadą ewolucji, która była bezpośrednią inspiracją dla twórców algorytmów genetycznych. Metody oparte na algorytmach genetycznych dają dobre rezultaty w złożonych problemach, gdzie zastosowanie innych metod nie jest możliwe lub daje słabe wyniki. Przykładem takiego problemu jest zadanie komiwojażera, stąd w pracy będzie podjęta próba zastosowania oraz zbadania efektywności algorytmu genetycznego dla tego problemu.

Algorytmy genetyczne powstały stosunkowo niedawno. Ich rosnąca popularność wynika ze wzrostu mocy obliczeniowej komputerów, które służą do przeprowadzenia symulacji algorytmów. W takcie prac nad algorytmami genetycznymi powstało wiele modyfikacji i usprawnień ideowego „prostego algorytmu genetycznego" Johna Hollanda. Dalsze prace na zagadnieniem algorytmów genetycznych owocowały powstawaniem nowych operatorów genetycznych, metod selekcji i innych modyfikacji. Obecnie za najbardziej efektywne w dziedzinie algorytmów genetycznych, uznaje się metody hybrydowe, które są rozwinięciem metod genetycznych o pewne heurystyki, co ma na celu wykorzystanie dostępnej wiedzy o konkretnym problemie.

W rozdziale I pracy zostaną przedstawione podstawowe pojęcia związane z algorytmami genetycznymi. Oprócz terminów i definicji przedstawione zostaną również podstawowe typy operatorów genetycznych, metod selekcji a także prosty przykład działania algorytmu genetycznego.

Rozdział II, w swojej pierwszej części, przedstawi charakterystykę problemu komiwojażera jako zadania optymalizacyjnego. Druga część rozdziału zajmą kolejne etapy budowy modelu algorytmu genetycznego do rozwiązywania problemu komiwojażera. Omówione zostaną różne warianty dla poszczególnych składników modelu, z których część znajdzie się w implementacji modelu.

W rozdziale III, zarazem wieńczącym pracę, omówione pokrótce zostaną szczegóły związane z implementacja modelu oraz zasadnicza jego część – eksperyment. Eksperyment zostanie przeprowadzony za pomocą stworzonej aplikacji, która stanowić będzie implementację modelu opisanego w rozdziale II. Eksperyment zostanie podzielony na kilka etapów, z których każdy będzie badał inny element modelu genetycznego. Podsumowaniem każdego z etapów będzie przedstawienie odpowiednich wniosków.

1. Wprowadzenie do algorytmów genetycznych

1.1 Inspiracja biologiczna

Podwaliny pod stan dzisiejszej wiedzy na temat algorytmów genetycznych położyli wybitni uczeni: Karol Darwin – twórca teorii ewolucji, Grzegorz Mendel – autor teorii dziedziczenia, Hugo de Varis – twórca teorii mutacji genów oraz Walter Sutton i Teodor Boveri – autorzy teorii chromosomu. Prace tych uczonych stanowiły inspirację dla twórców algorytmów genetycznych w postaci, jaką znamy teraz.

Zasada działania algorytmów genetycznych jest prawie dokładną analogią do zjawisk występujących w naturze. Wykorzystują one mechanizmy podobne do naturalnych, takich jak: ewolucyjna zasada doboru naturalnego, dziedziczenie i mutację. W tym miejscu posłużę się przykładem, zawartym w pozycji [3], pokazującym jak przebiega proces ewolucyjny dla pewnej populacji osobników.

Dana jest populacja królików. Jak w każdej większej grupie osobników, tak i w przypadku populacji królików, występują egzemplarze szybsze i sprytniejsze od innych. Szybkość i spryt pomaga im uciec przed lisem a co za tym idzie po prostu przeżyć. Zatem dzięki swoim cechom, grupa królików szybkich i sprytnych, w większości przypadków przeżywa. Nie bez szans są oczywiście króliki głupie i wolne, które mogą przeżyć dzięki szczęśliwemu przypadkowi. Grupa królików, która przeżyła, a są to zarówno króliki „szybkie" jak i „wolne" (oczywiste jest, że odsetek „szybkich" jest większy niż „wolnych"), wydaje potomstwo. Potomstwo to stanowi mieszaninę materiału genetycznego, ponieważ mogło się zdarzyć, że króliki „szybkie" krzyżowały się z „wolnym", tak samo jak „szybkie" z „szybkimi" czy „wolne" z „wolnymi". Populacja potomna może być wzbogacona o genetyczną „dziką kartę", czyli mutację. Przykład ten pokazuje ewolucyjną zasadę przystosowania. Osobniki populacji potomnej będą średnio szybsze i sprytniejsze od swoich poprzedników. Można powiedzieć, że w kolejnych pokoleniach następuje adaptacja populacji królików do panujących warunków środowiska, czyli w tym przykładzie zagrożeń ze strony lisów.

Algorytmy genetyczne, jak wiemy wzorowane na zjawiskach ewolucyjnych, realizują podobny schemat, co populacja królików czy każda inna populacja żywych organizmów. Algorytmy genetyczne czerpią nie tylko ogólną zasadę działania z genetyki czy teorii ewolucji, ale również z pewnych pojęć z tych dziedzin. Dzięki temu

zrozumienie sposobu ich działania nie stanowi większego problemu, gdyż terminologia, służąca opisowi algorytmów genetycznych jest bardzo intuicyjna. Mowa tu o takich terminach jak populacja czy osobnik, których interpretacja nie budzi żadnych wątpliwości.

Algorytm genetyczny przetwarza populację osobników. Odbywa się to w określonym środowisku. Środowisko to jest określane przez charakterystykę problemu, rozwiązywanego przez algorytm genetyczny. Każdy z osobników zawiera pewną informację, na podstawie której jest oceniany przez środowisko. Ocena środowiska polega na określeniu wartości przystosowania osobnika na podstawie informacji o tym osobniku. Mówiąc o ocenie środowiska rozumiemy pewne obiektywne przyporządkowanie osobników, na podstawie zawartej w nich informacji, do określonej wartości (liczbowej), dzięki której możliwe jest porównywanie osobników. Przyporządkowanie (pod pewnymi warunkami) można nazwać funkcją, a dla algorytmów genetycznych zwykło się mówić o funkcji przystosowania. Informacja o osobniku zawarta jest w jego chromosomie. Chromosom z kolei składa się z elementarnych jednostek składowych – genów. Geny można określić jako składnik opisujący pewną cechę osobnika, np. kolor oczu. Geny, możemy traktować jako zmienne, natomiast konkretne wartości tych zmiennych (nazywane allelami) determinują odmianę cechy u danego osobnika, np. zielony kolor oczu. Algorytmy genetyczne działają na populacji osobników, a konkretnie na ich chromosomach. Chromosomy są strukturą przechowującą informacje o osobnikach, służą do oceny ich przystosowania oraz podlegają operacjom genetycznym, co można łącznie określić jako kreowanie nowej populacji na podstawie populacji poprzedniej. Spodziewamy się, że każdy z osobników nowej populacji będzie średnio lepiej przystosowany do środowiska (tak jak w wypadku przykładu z królikami). To właśnie zasada ewolucji, poprawianie przystosowania jest cechą natury, która przyciągnęła uwagę badaczy innych dziedzin niż tylko biologia czy genetyka.

Dzięki swoim własnościom algorytmy genetyczne są chętnie wykorzystywane w różnych dziedzinach. Niewątpliwie jest to zasługą właśnie wspomnianych własności ale także dzięki intuicyjnemu i klarownemu opisowi. Co ciekawe algorytmy genetyczne są dziedziną mającą swój początek w naukach przyrodniczych jednak w obecnej formie mają z nimi niewiele wspólnego. Algorytmy genetyczne czerpią terminologię zarówno z genetyki, jak i informatyki. Ponieważ algorytmy genetyczne rozwiązują pewne problemy, które są następnie realizowane komputerowo, to terminy zaczerpnięte z

genetyki poszerzają swoje znaczenie lub nabierają nowych. Przedstawię w tym miejscu tabelę podstawowych terminów z genetyki oraz ich odpowiedników dla algorytmów genetycznych.

Genetyka	**Algorytmy genetyczne**
chromosom	ciąg kodowy
gen	cecha
allel	wariant cechy
genotyp	struktura
fenotyp	zdekodowana struktura

Tabela 1.1. Terminy genetyczne i ich odpowiedniki dla algorytmów genetycznych (na podstawie [2])

1.2. Prosty algorytm genetyczny

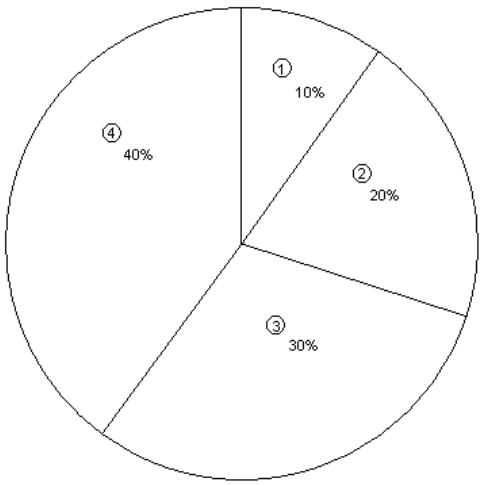
Algorytmy genetyczne wzorują się na procesie ewolucji występującym w naturze. Obserwacja przyrody zrodziła chęć wykorzystania mechanizmów adaptacyjnych żywych organizmów do rozwiązywania różnych problemów. Zaowocowało to szeregiem różnych dokonań badaczy, z których za najważniejsze lub wręcz przełomowe uznaje się „prosty algorytm genetyczny", zaproponowany w roku 1975 przez Johna Hollanda. Algorytm Hollanda jest zdumiewająco prosty. Prostota jest niewątpliwie przyczyną atrakcyjności stosowania algorytmów genetycznych.

Prosty algorytm genetyczny jest zbudowany z trzech podstawowych operacji na populacji osobników: reprodukcji, krzyżowaniu oraz mutacji. Chromosomy są kodowane binarnie, innymi słowy chromosomy są liczbami w systemie dwójkowym. Przykładowy chromosom może być ciągiem zerojedynkowym:
01101110010101011.

Kodowanie binarne jest najczęściej spotykanym rodzajem kodowania w zadaniach rozwiązywanych za pomocą algorytmów genetycznych. Istnieją jednak również inne sposoby kodowania chromosomów, uwzględniające specyfikę rozwiązywanego zagadnienia.

Na początku losowo generowana jest populacja bazowa. Polega to generowaniu losowych ciągów binarnych w liczbie odpowiadającej liczbie osobników populacji.

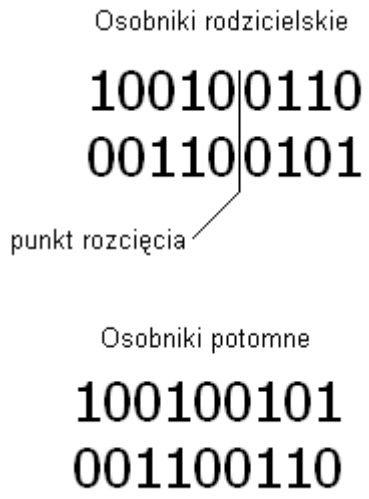
Kolejnym etapem jest reprodukcja. Polega ona na skopiowaniu osobników z populacji bazowej do populacji tymczasowej. Jednak nie jest to zwykłe kopiowanie chromosomu po chromosomie. Kopiowanie albo powielanie osobników musi się odbywać na wzór darwinowskiej zasady doboru naturalnego. Osobniki obiektywnie lepsze, o większym przystosowaniu, mają większą szansę na znalezienie się w populacji tymczasowej. Analogicznie osobniki o niskim przystosowaniu mają nikłe szanse na znalezienie się w niej, ale nie jest to niemożliwe, tylko mało prawdopodobne. Powielanie osobników odbywa się losowo, ale z uwzględnieniem ich przystosowania. Prawdopodobieństwo wylosowania osobnika o wysokim przystosowaniu jest większe od prawdopodobieństwa wylosowania osobnika o przystosowaniu mniejszym. Wartości prawdopodobieństw wylosowania dla poszczególnych osobników są proporcjonalne do wartości ich przystosowania. Przystosowanie jest liczone na podstawie informacji zawartej w chromosomie. W przypadku kodowania binarnego, informację zawartą w chromosomie możemy odczytać jako liczbę i na jej podstawie obliczyć wartość funkcji przystosowania.



Rysunek 1.1. Metoda ruletki (opracowanie własne)

Oczywiście, nie jest to jedyny sposób wykorzystania kodowania binarnego (inne podejście pokażę w ostatnim punkcie tego rozdziału), jednak jest on najczęściej spotykany. Funkcja przystosowania jest funkcją celu, którą algorytm maksymalizuje, tak jak króliki maksymalizują w kolejnych pokoleniach swoje przystosowanie do otaczającego środowiska. Sposób reprodukcji w prostym algorytmie genetycznym określa się jako reprodukcję proporcjonalną lub ruletkową, albo po prostu metodą ruletki. Przykładowo zakładając, że w populacji mamy tylko cztery osobniki, a ich przystosowania wynoszą kolejno: 1, 2, 3 i 4, to prawdopodobieństwa wylosowania do

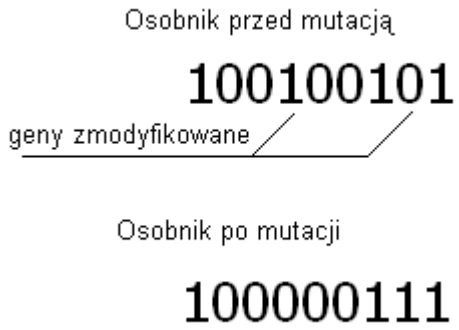
populacji tymczasowej są równe ilorazowi przystosowania konkretnego osobnika i sumy przystosowań wszystkich osobników. Zatem będą one wynosić odpowiednio: 10%, 20%, 30% oraz 40%. Załóżmy teraz, że wylosowano 4 liczby z przedziału (0;100): 25, 44, 71, 92. Wynika z tego, że do populacji tymczasowej trafi osobnik 2, 3 oraz dwie kopie osobnika 4. Jasnym jest, że średnie przystosowanie osobników populacji tymczasowej jest wyższe od tego z populacji bazowej. Jak widać reprodukcja proporcjonalna posługuje się zmodyfikowaną wersją ruletki, w której trafienie na pewne obszary jest bardziej prawdopodobne a na inne mniej. Kolejnym krokiem algorytmu jest przeprowadzenie operacji genetycznych na chromosomach. Najpierw przeprowadzane jest krzyżowanie.



Rysunek 1.2. Krzyżowanie jednopunktowe (opracowanie własne)

Aby możliwe było przeprowadzenie operacji krzyżowania konieczne jest skojarzenie osobników populacji tymczasowej w rozłączne pary. Następnie dla każdej pary z osobna podejmowana jest decyzja (losowa) o krzyżowaniu. Gdy decyzja jest negatywna para chromosomów pozostaje bez zmian, w przeciwnym przypadku następuje krzyżowanie. Prosty algorytm genetyczny przewiduje najprostszy z możliwych operatorów krzyżowania – krzyżowanie jednopunktowe (w literaturze występuje ogromna liczba różnych operatorów genetycznych, projektowanych specjalnie pod dane zagadnienie). Przy krzyżowaniu jednopunktowym należy określić losowo punkt przecięcia – jest to liczba naturalna ze zbioru $\{1,2,..n-1\}$, gdzie n jest długością chromosomu. Osobniki potomne tworzone są poprzez wymianę podciągów z osobników rodzicielskich wokół punktu krzyżowania. Po wykonanym cyklu krzyżowania, populacja poddawana jest drugiej klasycznej operacji genetycznej –

mutacji. Przyjmuje się, że mutacja zachodzi dla znikomej liczby genów (podobnie jak w naturze). Mutacja operuje na pojedynczych genach. Decyzja o mutacji podejmowana jest niezależnie, dla wszystkich genów każdego osobnika osobno. Przyjęło się, że prawdopodobieństwo mutacji powinno być stosunkowo małe (rzędu 0.001).



Rysunek 1.3. Mutacja (opracowanie własne)

Mutacja polega na zamianie wartości wybranego bitu na przeciwną. Podobnie jak w przypadku operatora krzyżowania, istnieje wiele różnych wariatów mutacji, uwzględniających specyfikę problemów, dla których były projektowane. Zakończenie operacji krzyżowania i mutacji finalizuje utworzenie populacji potomnej, która zostaje poddana ocenie, podobnie jak populacja bazowa na początku algorytmu. Następnie populacja potomna zostaje przypisana populacji bazowej (przypisanie w tym miejscu jest rozumiane jako instrukcja języka programowania), celem zapewnienia poprawności dalszego działania algorytmu. W tym miejscu kończy się pierwszy cykl algorytmu. Cykl ten nazywamy generacją, a stan populacji w danym momencie – pokoleniem. Algorytm powtarza wszystkie wymienione operacje, aż do wystąpienia warunku stopu. Warunek stopu jest, na tle całej genetycznej inspiracji, czymś zupełnie sztucznym. W przyrodzie ewolucja jest procesem ciągłym, który nie ma końca. Jednak pętla nieskończona w wypadku implementacji komputerowej nie ma zupełnie sensu. Poza tym możemy sobie wyobrazić sytuację, gdy nasza populacja będzie już na tyle przystosowana, że będzie nas to już w zupełności satysfakcjonowało, więc nie będzie potrzeby kontynuowania algorytmu.

Prosty algorytm genetyczny jest uściśleniem, ale i uproszczeniem procesu ewolucji w przyrodzie. Korzysta z zasady adaptacji osobników lepiej przystosowanych oraz zamieraniem osobników o niskim przystosowaniu. Podczas krzyżowania zachodzi

```
procedure Prosty algorytm genetyczny
//P – populacja bazowa, T – populacja tymczasowa, O – populacja potomna
begin
        t:=0
        inicjacja P^0
        ocena P^0
        while (not warunek stopu) do
        begin
                T^t:= reprodukcja P^t
                O^t:= krzyżowanie i mutacja T^t
                ocena O^t
                P^(t+1):= O^t
                t:=t+1
        end
end
```

Rysunek 1.4. Prosty algorytm genetyczny w pseudokodzie (źródło [1])

łączenie informacji genetycznej osobników, co w kolejnych krokach algorytmu powoduje zwiększenie średniej wartości przystosowania dla populacji. Algorytm z czasem był udoskonalany i modyfikowany tak, aby jeszcze lepiej radził sobie ze specyficznymi problemami, jednak ogólna zasada działania pozostała bez zmian.

1.3. Algorytmy genetyczne a metody tradycyjne

Algorytmy genetyczne stanowią alternatywę dla metod tradycyjnych. Reprezentują zupełnie inną filozofię, jeżeli chodzi o podejście do rozwiązywania stawianych im problemów. Szczególnie popularnym i szerokim obszarem zastosowań algorytmów genetycznych są zagadnienia optymalizacji (temat optymalizacji poruszany jest w rozdziale drugim pracy). Istnieje wiele klasycznych metod rozwiązywania zadań optymalizacji, jednak powstanie algorytmów genetycznych sprawiło, że przybyła im licząca się konkurencja. Mimo, że algorytmy genetyczne mają zastosowanie w innych dziedzinach niż optymalizacja, to sposób przetwarzania algorytmów genetycznych sam w sobie ma charakter optymalizacyjny. Zatem za każdym razem, gdy będzie mowa o

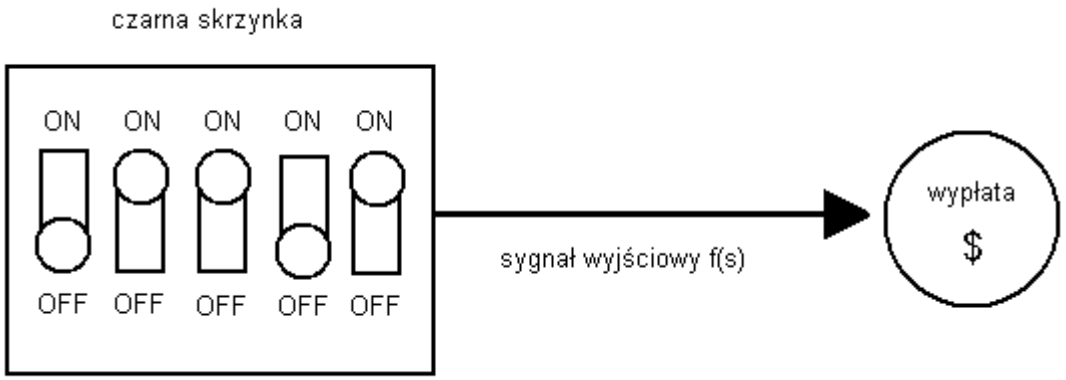
zastosowaniu algorytmu genetycznego, rozumiane przez to będzie zastosowanie w optymalizacji.

Algorytmy genetyczne należą do rodziny algorytmów probabilistycznych. Probabilistyczny w tym przypadku nie oznacza czysto losowy. W przeciwieństwie do metod deterministycznych, algorytmy genetyczne korzystają z minimum informacji o problemie. Metody deterministyczne, jak na przykład rachunek pochodnych znany z analizy matematycznej, są obarczone pewnymi ograniczeniami (warunek ciągłości i różniczkowalności rozpatrywanej funkcji). Z kolei algorytm genetyczny nie potrzebuje żadnych dodatkowych informacji (np. znajomości pochodnej) do rozwiązania zadania, korzysta wyłącznie z funkcji przystosowania, która nie musi być w postaci analitycznej. Kolejną różnicą w stosunku do metod tradycyjnych jest operowanie na zakodowanej wersji wartości zmiennych zadania. Kodowanie jest podyktowane koniecznością przeprowadzania operacji genetycznych, co byłoby niemożliwe na postaci niezakodowanej. Następna różnica polega na sposobie poszukiwania rozwiązań. W metodach tradycyjnych (np. metody gradientowe) kolejne lepsze rozwiązania generowane są na podstawie wcześniejszego rozwiązania, zatem jest to przejście z punktu do punktu. W algorytmach genetycznych przetwarzana jest cała populacja punktów (rozwiązań zadania) . Algorytmy genetyczne posiadają następujące cechy, odróżniające je od metod tradycyjnych:

- przetwarzanie zakodowanej postaci parametrów zadania
- korzystanie wyłącznie z funkcji celu
- losowa reguła wyboru
- przetwarzanie całej populacji rozwiązań zadania

Wymienione własności dobrze ilustruje przykład optymalizacji czarnej skrzynki przytoczony w pozycji [2]. Dana jest czarna skrzynka z pięcioma przełącznikami ON/OFF. Dla sygnałów wyjściowych skrzynki zdefiniowana jest funkcja wypłat. Rozwiązania tego zadania są zatem kodowane poprzez wektor stanów przełączników. Jako, że przełączniki posiadają tylko dwa ustawienia, jest to kodowanie binarne. Tradycyjne podejście do optymalizacji tak postawionego problemu polegałoby na przestawianiu przełączników z jednej pozycji na drugą w określonej kolejności. Sposób przejścia z jednej pozycji do drugiej byłby określany przez konkretną metodę. Natomiast algorytm genetyczny przetwarza jednocześnie wiele ustawień przełączników naraz, wybierając do kolejnych pokoleń ustawienia o średnio coraz większej wypłacie.

Oczywiście, aby metoda tradycyjna była w stanie rozwiązać problem czarnej skrzynki, musi znać regułę przejścia z jednego ustawienia do kolejnego. Taka wiedza może się znajdować w czarnej skrzynce – jest więc niedostępna. Ponieważ algorytm genetyczny nie musi znać specyfiki rozwiązywanego problemu, zawartość czarnej skrzynki jest bez znaczenia.



Rysunek 1.5. Optymalizacja czarnej skrzynki (opracowanie własne na podstawie [2])

Własności algorytmów genetycznych pozwalają efektywnie rozwiązywać pewne klasy zadań. Nie jest domeną algorytmów genetycznych rozwiązywanie problemów dobrze określonych, takich na przykład jak zadanie, w którym mamy określoną wypukłą, różniczkowalną funkcję w postaci analitycznej. W takiej sytuacji lepszym rozwiązaniem jest wybranie metody specjalizowanej. Natomiast, gdy wiedza o problemie jest znikoma i niemożliwe jest zastosowanie na przykład metod analitycznych, zastosowanie algorytmu genetycznego da z pewnością dobre rezultaty. Możliwość stosowania algorytmów genetycznych niezależnie od natury problemu jest ich zaletą, ale i jednocześnie wadą. Zaletą jest niewątpliwie to, że niezależnie od zadania niezmienne jest podejście, sposób jego rozwiązywania. Pozwala to na rozwiązywania różnych klas zadań w ten sam sposób (co ma sens wtedy, gdy nie istnieją specjalizowane metody rozwiązywania tych zadań). Może zdarzyć się jednak tak, że algorytm genetycznych z powodu swojej uniwersalności, nie będzie generował dobrych rozwiązań lub będzie je generował nieefektywnie. W takiej sytuacji możliwe jest „zaszycie" informacji o problemie, poprzez zaprojektowanie odpowiednich operatorów genetycznych oraz sposobu kodowania. Niebagatelną zaletą algorytmów genetycznych jest prostota ich implementacji, głównie dzięki prostej zasadzie działania a także dzięki prostym, elementarnym operacjom na strukturach danych.

1.4. Podstawy matematyczne algorytmów genetycznych

Algorytmy genetyczne, mimo prostej zasady działania, stanowią efektywne narzędzie optymalizacyjne. Kolejne pokolenia charakteryzują się coraz wyższym średnim przystosowaniem, co niewątpliwie jest korzystne. Intuicja podpowiada, że ten schemat działania jest korzystny i prowadzi do poprawy wartości funkcji celu, w zadaniu rozwiązywanym za pomocą algorytmu genetycznego. Należałoby jednak zadać pytanie czy w istocie jest to korzystne i czy intuicja jest w tym wypadku poprawna. Pytanie to można postawić w sposób bardziej konkretny, mianowicie, jaka informacja, zawarta w populacji chromosomów, odpowiada za prawidłowy kierunek poszukiwań algorytmu genetycznego? W przykładzie zaprezentowanym w pozycji [2] – optymalizacji funkcji jednej zmiennej $f(x) = x^2$ została przeprowadzona następująca obserwacja:

Ciąg	Przystosowanie
01101	169
11000	576
01000	64
10011	361

Tabela 1.2. Przykładowe ciągi kodowe i odpowiadające im przystosowania (źródło [2])

wysokie wartości przystosowania odpowiadają ciągom o podobnej strukturze, podobnie jak ciągom o niskim przystosowaniu. W przypadku wysokiego przystosowania ciągi mają na najstarszej pozycji „jedynkę". Z kolei gorzej przystosowane mają „zero". W tym miejscu da się zauważyć, że istnieje zależność wartości funkcji przystosowania od struktury ciągu. W tym miejscu należałoby uściślić nieco ten jakościowy opis. Do opisu podobieństw strukturalnych ciągów służą schematy.

Schemat jest podzbiorem ciągów kodowych podobnych ze sobą ze względu na ustalone pozycje. Do opisu klasycznego sposobu kodowania problemów w zastosowaniach algorytmów genetycznych używa się alfabetu dwójkowego {0,1}. W opisie za pomocą schematów rozszerza się ten alfabet o znak „*", który oznacza „nieistotne", jest to zatem znak uniwersalny, może oznaczać zarówno 0 jak i 1. Przykładowo schemat:

10*11

oznacza następujący zbiór ciągów:

{10011,10111}

schemat:

10**1

oznacza:

{10001,10011,10101,10111}

a schemat:

10101

oznacza po prostu:

{10101}.

Liczba możliwych schematów dla ciągu kodowego o długości l wynosi 3^l. Dla schematu o długości $l = 5$ wynosi $3^5 = 243$, gdyż na każdej pozycji może występować 0, 1 lub *. Dla konkretnego ciągu o długości l istnieje 2^l pasujących do niego schematów. Dla przykładowego ciągu 00000 istnieje 2^5 schematów pasujących do niego, bo na każdej pozycji może być „0" lub „*". Wynika stąd, że populacja składająca się z n osobników może zawiera od 2^l do $n \cdot 2^l$ schematów, gdzie l jest długością ciągu kodowego (chromosomu). Liczba tych schematów wacha się między 2^l a $n \cdot 2^l$, ponieważ może się zdarzyć sytuacja, w której ciągi w populacji będą się powtarzać.

Schematy pozwalają na przeprowadzenie analizy skutków reprodukcji oraz operacji genetycznych na stan populacji w kolejnych krokach algorytmu genetycznego. Dla opisu schematów zdefiniowano pojęcia rzędu i rozpiętości schematu.

Definicja 1.1.

Rzędem schematu H, oznaczonym przez $o(H)$ nazywa się liczbę ustalonych pozycji (pozycji różnych od znaku „nieistotne") we wzorcu.

Przykładowo:

$o(10**1*01) = 5$

$o(1*******) = 1$

Definicja 1.2.

Długością schematu H, oznaczoną przez $\delta(H)$ nazywa się odległość między dwiema skrajnymi ustalonymi pozycjami we wzorcu.

Przykładowo:

$\delta(*0**101*) = 7 - 2 = 5$

$\delta(1*******) = 1 - 1 = 0$

Posiadając narzędzia opisu ciągów kodowych w postaci schematów oraz ich własności można przeprowadzić analizę wpływu reprodukcji i operacji genetycznych na populację, tym samym przybliżyć, w ściślejszy sposób, zasadę działania algorytmu genetycznego. Analiza taka opiera się na badaniu liczby wystąpień określonego schematu, poddanemu działaniu wymienionych operacji.

Badanie oczekiwanej liczby reprezentantów danego schematu rozpocznie się od określenie wpływu reprodukcji. Zakłada się, że w populacji w pewnej chwili t znajduje się $m = m(H,t)$ reprezentantów schematu H. Ponieważ analiza jest przeprowadzana zgodnie z założeniami klasycznego algorytmu genetycznego, do reprodukcji wykorzystywana jest reguła ruletki (proporcjonalna), zgodnie z którą prawdopodobieństwo znalezienia się osobnika w populacji potomnej jest ilorazem jego przystosowania i sumy przystosowań wszystkich osobników:

$$p_i = \frac{f_i}{\sum f_i}$$

Liczebność populacji wynosi n. Mając te informacje należy określić oczekiwaną liczbę reprezentantów schematu H, czyli:

$$m(H,t+1) = m(H,t) \cdot n \cdot \frac{f(H)}{\sum f_i}$$

gdzie $f(H)$ oznacza średnie przystosowanie ciągów, reprezentujących schemat H.

Wartość średniego przystosowania całej populacji wyraża się wzorem:

$$\bar{f} = \frac{\sum f_i}{n}$$

więc wzór na oczekiwaną liczbę reprezentantów schematu w kolejnej populacji przyjmuje postać:

$$m(H,t+1) = m(H,t) \cdot \frac{f(H)}{\bar{f}}$$

Wynika z tego, że liczba reprezentantów schematu H jest proporcjonalna do średniego przystosowania tego schematu. Wniosek ten jest całkowicie zgodny intuicyjnym przewidywaniem. Ze względu na samą reprodukcję schematy propagują się w następnym pokoleniu odpowiednio do swojego średniego przystosowania. Oznacza to, że schematy „dobre" (ponadprzeciętne) rozprzestrzeniają się, a „gorsze" zanikają. Znając liczbę reprezentantów schematu w kolejnym pokoleniu, można wyprowadzić wzór zależność na ich liczbę w kolejnych pokoleniach. Jeżeli założy się, że przystosowanie schematu H jest większe od przystosowania średniego o pewną stałą wartość $c\bar{f}$ (c - dowolna liczba dodatnia), to liczebność reprezentantów tego schematu można określić następująco:

$$m(H,t+1) = m(H,t) \cdot \frac{(\bar{f} + c\bar{f})}{\bar{f}} = m(H,t) \cdot (1+c)$$

a zaczynając od $t = 0$:

$$m(H,t) = m(H,0) \cdot (1+c)^t$$

Wzory te stanowią ilościową ocenę działania reprodukcji na schemat. Wynika z niej, że lepsze schematy rozprzestrzeniają się w rosnącej wykładniczo liczbie reprezentantów, analogicznie gorsze – w malejącej.

Oczywiście sama reprodukcja nie przyczynia się do przeszukiwania nowych rozwiązań, a jedynie przegląda i wybiera rozwiązania z populacji początkowej. Za dostarczanie nowej informacji, a konkretnie (w opisie jakościowym) składaniu znanych poglądów w nowe idee, odpowiedzialne są operacje genetyczne – krzyżowanie i mutacja. Krzyżowanie zaburza wzorzec reprodukcji. Operacja krzyżowania przeprowadza uporządkowaną, ale zawierającą element przypadku, wymianę informacji między ciągami kodowymi. Kojarzenie osobników wywołuje zmiany w średnim przystosowaniu populacji. Działanie na schematy operacji krzyżowania (rozpatrywany jest klasyczny operator - krzyżowanie proste jednopunktowe) może być dwojakiego rodzaju: albo je niszczy albo nie narusza. Przykładowo ciąg i dwa schematy (na podstawie [2]):

A = 0111000

H_1 = *1****0

H_2 = ***10**

Ciąg **A** jest reprezentantem schematów H_1 i H_2. Aby pokazać działanie operatora krzyżowania na schemat, należy określić ten operator. Wcześniej zostało powiedziane, że rozpatrywanym operatorem krzyżowania jest klasyczny operator – proste krzyżowanie jednopunktowe dla kodowania binarnego. Do krzyżowania został wybrany przykładowy ciąg **A**, określony wcześniej. Następnie losowo wybrany został punkt krzyżowania - powiedzmy, że wybrany został punkt między 3 a 4 pozycją w ciągu:

$A = 011|1000$

$H_1 = *1*|***0$

$H_2 = ***|10**$

Z tej prostej analizy wynika, że schemat H_1 ulegnie rozerwaniu na skutek krzyżowania a schemat H_2 przetrwa. Zauważyć można ponadto, że rozpiętość schematu H_2 jest mniejsza od rozpiętości H_1. Aby sprawdzić zależność szans przetrwania schematu od jego rozpiętości należy przeprowadzić analizę przypadku ogólnego. W przykładzie z ciągiem A jasnym było, że prawdopodobieństwo tego, że schemat ten zostanie przerwany wynosi 1/6 (przetrwa w 5/6 przypadków). Wynika to z tego, że dla ciągu kodowego o długości 7, istnieje 6 możliwych punktów krzyżowania, z kolei rozpiętość tego schematu wynosząca 1 powoduje istnienie tylko jednej możliwości przerwania schematu H_2 (między pozycją 4 i 5). Analogicznie dla schematu H_1 prawdopodobieństwo przetrwania wynosi 1/6. Używając opisu słownego to, że schemat przetrwa można określić następująco: schemat przetrwa, jeżeli punkt krzyżowania wypadnie poza jego „wnętrzem", czyli poza obszarem wyznaczanym przez pierwszą i ostatnią ustaloną pozycją. Ostatecznie prawdopodobieństwo przeżycia schematu określa następującą nierówność:

$$p_S \geq 1 - p_C \cdot \frac{\delta(H)}{l-1}$$

p_S - prawdopodobieństwo przeżycia

p_C - prawdopodobieństwo krzyżowania (parametr algorytmu)

$\delta(H)$ - rozpiętość schematu

$l-1$-liczba możliwych położeń punktu krzyżowania, gdzie l to długość ciągu.

Obecność nierówności w tym wzorze wynika z tego, że prawdopodobieństwo krzyżowania jest nie większe od 1 (w praktyce jest zawsze mniejsze od 1), czyli może się zdarzyć, że dla wybranej pary rodziców algorytm nie przeprowadzi krzyżowania – w

tej sytuacji schematy na pewno przetrwają.

Wyprowadziwszy zależność dotyczącą krzyżowania a wcześniej reprodukcji, można określić ich łączny wpływ na liczbę schematów w danym pokoleniu. Oczekiwana liczba reprezentantów schematu H po uwzględnieniu reprodukcji i krzyżowania jest iloczynem oczekiwanej liczby schematów po reprodukcji i prawdopodobieństwa przetrwania operacji krzyżowania schematu :

$$m(H,t+1) = m(H,t) \cdot \frac{f(H)}{\bar{f}} \cdot \left[1 - p_C \cdot \frac{\delta(H)}{l-1}\right]$$

Widać, że większą liczbę reprezentantów będą miały schematy o wyższym przystosowaniu (co zostało już pokazane) i małej rozpiętości.

Ostatnim czynnikiem wpływającym na oczekiwaną liczbę reprezentantów schematu jest mutacja. Definicja mutacji, podobnie jak krzyżowania, w tej analizie jest tożsama z klasyczną wersją mutacji binarnej, polegającej na zamianie bitów w ciągu na przeciwne z określonym prawdopodobieństwem p_M. Określony schemat przetrwa operację mutacji, jeżeli nie zmieniony zostanie każdy bit o ustalonej wartości. Prawdopodobieństwo przetrwania bitu na określonej pozycji jest równe $1 - p_M$. Zakładając, że ustalonych pozycji w schemacie jest $o(H)$ to prawdopodobieństwo przeżycia mutacji dla schematu H wynosi:

$$p_S = (1 - p_M)^{o(H)}$$

Dla małych wartości p_M (a w praktyce takie właśnie się wykorzystuje) wzór ten można przybliżyć wyrażeniem:

$p_S = 1 - o(H) \cdot p_M$, dla małych wartości p_M (rzędu 0.1).

Ostatecznie łączny wpływ reprodukcji, krzyżowania i mutacji na oczekiwaną liczbę reprezentantów schematu można określić wzorem:

$$m(H,t+1) = m(H,t) \cdot \frac{f(H)}{\bar{f}} \cdot \left[1 - p_C \cdot \frac{\delta(H)}{l-1} - o(H) \cdot p_M\right]$$

Ponieważ wyrażenie $o(H) \cdot p_M$ ma relatywnie małą wartość, nie zmienia ono znacząco oczekiwanej liczby schematów w porównaniu z działaniem samej reprodukcji i krzyżowania. Wyprowadzony wzór stanowi algebraiczny zapis fundamentalnego twierdzenia z dziedziny algorytmów genetycznych – twierdzenia o schematach. Twierdzenie to zostało zaproponowane przy pewnych założeniach, ograniczeniach:

- populacja bazowa zawiera nieskończoną liczbę osobników

- populacja bazowa zawiera chromosomy należące do wszystkich możliwych schematów
- reprodukcja metodą ruletki
- zastosowanie binarnego kodowania, krzyżowania jednopunktowego oraz mutacji bitowej z prawdopodobieństwem bliskim zeru.

Twierdzenie o schematach:
Liczba osobników, cechujących się schematami o małej długości i przystosowaniem przewyższającym średnią, rośnie wykładniczo w kolejnych pokoleniach.

Twierdzenie o schematach pozwala na sformułowanie dalszych wniosków. Pozwala określić, co jest kluczowym czynnikiem w przeszukiwaniu przestrzeni rozwiązań, dokonywanym przez algorytm genetyczny. Wniosek ten jest znany jako hipoteza cegiełek (lub hipoteza bloków budujących). Hipoteza ta zakłada, że w kolejnych pokoleniach osobniki będą składane z coraz lepszych cegiełek (bloków budujących), czyli krótkich schematów. Rozwiązanie generowane przez algorytm genetyczny otrzymuje się przyrostowo, poprzez składanie cegiełek. Składanie odbywa się poprzez użycie operatora krzyżowania, co z kolei podkreśla jego ogromne znaczenie. Idąc dalej, hipoteza cegiełek, dała podwaliny nowemu podejściu do metodyki algorytmów genetycznych. Dotychczas zakładano, że działanie algorytmu odbywa się w całkowitym oderwaniu od natury problemu, jaki rozwiązuje. Jest to ciekawa własność, gdyż elastyczność pozwala jednej metodzie na znajdowanie rozwiązań różnych problemów. Jednak nie zawsze generowane rozwiązania są na tyle dobre, aby konkurować z innymi, często deterministycznymi metodami, podpartymi wiedzą o problemie. Przykładem takiego zagadnienia jest problem komiwojażera (TSP, ang. *Travelling salesman problem*), którego rozwiązanie za pomocą algorytmów genetycznych jest przedmiotem tej pracy. Zastosowanie kodowania binarnego i klasycznych operatorów (szczegółowe omówienie modeli genetycznych dla problemu komiwojażera znajduje się w rozdziale II pracy) daje słabe rezultaty. Z tego powodu powstały inne podejścia do tego problemu, których inspiracją była hipoteza cegiełek. Hipoteza cegiełek, zakłada istnienie bloków (dobrej jakości), z których składane jest dobre, bardzo dobre lub nawet optymalne rozwiązanie. Problem polega na takim określeniu składowych algorytmu (kodowania, operatorów), aby poprzez przetwarzanie genetyczne wyodrębniane były cegiełki, specyficzne dla danego problemu. TSP jest przykładem problemu, w którym rezygnacja

z ogólnego podejścia, reprezentowanego przez klasyczny algorytm genetyczny, daje dobre rezultaty w postaci rozwiązań optymalnych dla problemów o małej i średniej wielkości. Algorytm genetyczny, w swojej klasycznej postaci, rezygnuje z wiedzy o problemie. Jest to niewątpliwie zaletą w przypadku, gdy wiedza taka nie jest dostępna. Mówi się wtedy o algorytmach genetycznych jako metodach ostatniej szansy. Ale skoro dysponujemy taką wiedzą to, dlaczego by z niej nie skorzystać celem poprawienia jakości rozwiązania, pomimo utraty ogólności metody.

1.5. Selekcja

O ile dla trywialnych problemów wystarczające będzie użycie klasycznego schematu algorytmu genetycznego, to dla niektórych problemów może się on okazać nieefektywny. Prace nad algorytmami genetycznymi doprowadziły do powstania szeregu modyfikacji klasycznego algorytmu. Modyfikacje te dotyczyły metod selekcji, operatorów genetycznych oraz sposobu kodowania zadań. Mimo, że ogólność i prostota klasycznego podejścia nie uniemożliwiały rozwiązywania zadań, to czyniła to rozwiązywanie nieefektywnym dla pewnych problemów. Stąd właśnie wzięły się modyfikacje algorytmu. Jedną z grup modyfikacji są różne warianty selekcji.

Selekcja jest często mylona z reprodukcją, gdyż w prostym algorytmie genetycznym pojęcia te są tożsame. Jednak należy ściśle rozdzielić te pojęcia. Selekcja jest pojęciem szerszym. Selekcja przebiega dwuetapowo: najpierw odbywa się reprodukcja, a następnie sukcesja. Reprodukcja powiela osobniki o wyższej wartości funkcji przystosowania z większym prawdopodobieństwem niż te gorzej przystosowane. Dzięki temu algorytm ewoluuje w stronę coraz lepszych rozwiązań (kolejne pokolenia mają coraz wyższe średnie przystosowanie). Reprodukcja pełni w selekcji rolę pierwszoplanową, jednak jej zadaniem jest przygotowanie populacji tymczasowej, która jest poddawana operacjom genetycznym. Populacja tymczasowa po przeprowadzeniu krzyżowania i mutacji jest poddana drugiemu etapowi selekcji – sukcesji. Sukcesja określa sposób przejścia z przetworzonej populacji tymczasowej do kolejnej populacji bazowej. W tym kontekście o reprodukcję można określić jako preselekcję, a sukcesję – postselekcją.

Kluczowym określeniem charakteryzującym metody selekcji jest nacisk selektywny. Nacisk selektywny stanowi tendencję algorytmu do poprawy średniej

wartości przystosowania w populacji. Nacisk jest tym większy im więcej egzemplarzy dobrze przystosowanych osobników znajdzie się w kolejnej populacji. Im większy nacisk selektywny tym algorytm szybciej lokalizuje okolice rozwiązania, jednocześnie jest wtedy wrażliwy na znajdowanie minimów lokalnych. Wzmocnienie nacisku selektywnego ma generalnie na celu zwiększenie efektywności algorytmu, czyli w tym przypadku zmniejszenie czasu wykonywania. Znalezienie dobrego rozwiązania zadania metodyką algorytmów genetycznych jest często kompromisem między naciskiem selektywnym, prowadzącym niekiedy do rozwiązań suboptymalnych, a efektywnością algorytmu przy zbyt długim czasie jego wykonania.

1.5.1. Reprodukcja

Reprodukcja proporcjonalna. Reprodukcja proporcjonalna (inaczej metoda ruletki) jest najbardziej znaną, klasyczną metodą reprodukcji, wywodzącą się z klasycznego prostego algorytmu genetycznego. W metodzie ruletki prawdopodobieństwo wyboru określonego osobnika z populacji P^t wyraża się następującym wzorem:

$$p_r = \frac{\Phi(X)}{\sum_{Y \in P^t} \Phi(Y)}$$

p_r - prawdopodobieństwo reprodukcji osobnika

$\Phi(X)$ - przystosowanie osobnika

$\sum_{Y \in P^t} \Phi(Y)$ - suma przystosowań wszystkich osobników w populacji

Prawdopodobieństwo reprodukcji pewnego osobnika jest równe ilorazowi jego przystosowania i sumy przystosowań wszystkich osobników w populacji. Dzięki znajomości prawdopodobieństwa reprodukcji osobnika można określić oczekiwaną liczbę jego kopii po reprodukcji:

$$e(X) = p_r(X) \cdot n$$

gdzie n jest liczbą osobników w populacji.

Reprodukcja proporcjonalna, mimo swojej popularności i znaczenia dla rozwoju algorytmów genetycznych, nie jest wolna od wad. Jedną z nich jest wrażliwość na dodanie wartości stałej do funkcji przystosowania. Wyniki eksperymentalne zawarte w pozycji [1] pokazują, że im większa jest wartość stałej dodanej do funkcji przystosowania, tym gorsze wyniki są generowane przez algorytm. Kolejną wadą reprodukcji proporcjonalnej jest założenie o dodatnich wartościach funkcji

przystosowania. Mankament ten da się obejść korzystając ze zmodyfikowanej wersji metody ruletki. Aby zapobiec uwzględnianiu ujemnych wartości funkcji przystosowania przy obliczaniu prawdopodobieństw reprodukcji stosuje się pewne skalowanie tej funkcji. Polega to na utworzeniu zmodyfikowanej wersji funkcji przystosowania:

$$\Phi'(X) = \Phi(X) - \Phi_0$$

gdzie

$$\underset{X \in D}{\forall} : \Phi_0 \leq \Phi(X)$$

Czasem nie jest możliwe znalezienie wartości Φ_t, która jest globalnym minimum funkcji przystosowania, wtedy zastępuje się tę wartość najmniejszą wartością funkcji przystosowania w danym pokoleniu.

Reprodukcja rangowa.

Odmiennym podejściem do reprodukcji w stosunku do metody ruletki jest metoda rangowa (rankingowa). Reprodukcja oparta na rankingu jest pozbawiona wady metody ruletki, związanej z ujemnymi wartościami funkcji przystosowania. W metodzie tej osobniki są szeregowane według wartości przystosowania – od najlepiej przystosowanego do najgorzej przystosowanego. Następnie osobnikom nadawane są rangi, które są po prostu ich numerami w wcześniej dokonanym uporządkowaniu. I tak najlepszy osobnik ma rangę równą 1, najgorszy – równą liczbie osobników w populacji. Na podstawie rang określane jest prawdopodobieństwo reprodukcji osobnika. Tworzy się je na podstawie funkcji, którą jest dobierana arbitralnie. Popularna jest funkcja liniowa:

$$p_r = a + k \cdot \left(1 - \frac{r(X)}{r_{\max}}\right)$$

Gdzie p_r jest prawdopodobieństwem reprodukcji osobnika, $r_{\max}$ jest maksymalną co do wartości rangą, natomiast a i k są parametrami dobieranymi w taki sposób, aby spełnione były następujące warunki:

$$\sum_{i=1}^{N} p_r(X) = 1$$

$$0 \leq p_r(X) \leq 1$$

Innym sposobem określenia prawdopodobieństw reprodukcji w metodzie rangowej jest wykorzystanie funkcji potęgowej (parametry funkcji są ustalane analogicznie jak dla

funkcji liniowej):

$$p_r = a + k \cdot \left(r_{\max} - r(X)\right)^b$$

Metoda oparta na rankingu jest też wolna od drugiej głównej wady metody ruletki – wrażliwości na dodanie stałej do funkcji przystosowania. Jednak w przypadku zastosowania rankingu jako metody reprodukcji, głównym problemem może się okazać dobranie odpowiedniej postaci funkcji wyznaczającej prawdopodobieństwa reprodukcji.

Reprodukcja turniejowa

Kolejnym ciekawym sposobem reprodukcji jest metoda turniejowa. Jak wskazuje nazwa metody, osobniki konkurują ze sobą, które z nich zostaną zreprodukowane. W kolejnych krokach tej metody wybiera się losowo podpopulację, liczącą q osobników. Następnie z tak wybranej grupy osobników wybiera się osobnika najlepiej przystosowanego w swojej grupie – zwycięzcę turnieju. Osobnik ten jest kopiowany do populacji tymczasowej. Proces trwa do momentu wypełnienia całej populacji tymczasowej. Głównym parametrem tej metody jest wielkość grupy turniejowej q. W praktyce najlepsze rezultaty dają małe grupy turniejowe (2, 3 elementowe).

Reprodukcja progowa

Metoda ta jest szczególnym przypadkiem reprodukcji rangowej. Prawdopodobieństwo reprodukcji jest określone wzorem:

$$p_r = \begin{cases} \dfrac{1}{\rho \cdot n} & dla \quad 0 \le r(X) \le \rho \cdot n \\ 0 & w \quad przeciwnym \quad razie \end{cases}$$

gdzie:

n - wielkość populacji

ρ - wskaźnik nacisku selektywnego

$r(X)$ - ranga nadana tak jak w metodzie rangowej

Kluczowym parametrem metody jest wskaźnik nacisku selektywnego. Zmniejszanie wartości tego wskaźnika powoduje zwiększanie nacisku selektywnego. W praktyce najlepsze rezultaty dają wartości wskaźnika w okolicach 0.5 (na podstawie [1]).

1.5.2. Sukcesja

Sukcesja określa sposób tworzenia populacji potomnej, na podstawie wcześniej utworzonej, populacji tymczasowej. Wyróżnia się dwa rodzaje sukcesji: z całkowitym zastępowanie oraz z częściowym zastępowaniem.

Sukcesja z całkowitym zastępowaniem występuje w prostym algorytmie genetycznym (stąd bierze się utożsamianie reprodukcji z szerszym pojęciem selekcji w tym przypadku). Ten schemat sukcesji (zwany też trywialnym) polega tym, że utworzona populacja tymczasowa w procesie reprodukcji staje się aktualną populacją bazową, bez jakichkolwiek modyfikacji. Sukcesja z całkowitym zastępowaniem nie wprowadza dodatkowego nacisku selektywnego.

Dużo ciekawszym schematem sukcesji jest jej wariant z częściowym zastępowaniem. W przeciwieństwie do sukcesji trywialnej w tym przypadku nowa populacja bazowa jest tworzona zarówno z osobników populacji tymczasowej jak i tych z poprzedniej populacji bazowej. Istnieją trzy warianty sukcesji z częściowym zastępowaniem:

- do populacji bazowej wchodzą najlepsze osobniki ze starej populacji bazowej
- metoda ze ściskiem
- wybór losowych osobników

Z wymienionych wariantów zdecydowanie najpopularniejszym i dającym dobre rezultaty jest wybór najlepszych osobników do populacji bazowej. Metodę tą określa się mianem sukcesji elitarnej.

Sukcesja elitarna

Procedura sukcesji elitarnej jest następująca: osobniki starej populacji bazowej sortuje się według wartości funkcji przystosowania. Następnie z tego posortowanego zbioru wybiera się η najlepszych osobników (parametr sukcesji). W kolejnym kroku wybiera się μ (liczebność populacji bazowej) najlepszych osobników spośród osobników populacji tymczasowej oraz zbioru wybranych η najlepszych osobników (zatem dokonuje się wyboru μ osobników spośród $\mu+\eta$ osobników). Zastosowanie sukcesji elitarnej wzmacnia nacisk selektywny przeszukiwania. Może niestety powodować przedwczesną zbieżność, czyli spowodować osiadanie algorytmu w minimach lokalnych. Aby zapobiec temu zjawisku należy odpowiednio dobrać liczebność elity η. W szczególnym przypadku, gdy $\eta = 0$ sukcesja elitarna jest tożsama z sukcesją

trywialną. Oczywiste jest, że należy poszukiwać niezerowych wartości tego parametru. Wyniki badań zaprezentowane w pozycji [1] pokazują, że najlepsze wyniki daje mała liczebność elity. Zastosowanie sukcesji elitarnej znacznie poprawia jakość rozwiązania. Średnia uzyskanych rozwiązań jest zdecydowanie większa od średniej dla sukcesji trywialnej, a do tego sukcesja elitarna generuje rozwiązania przy średnio dużo niższym koszcie niż trywialna. Co do wielkości elity to najlepsze rezultaty uzyskuje nieliczna elita. W [1] była to elita wielkości 1 i 2.

1.6. Kodowanie

Klasycznym sposobem kodowania zmiennych zadania, rozwiązywanego za pomocą algorytmu genetycznego, jest zapisywanie ich w postaci wektorów binarnych. Kodowanie binarne jest uniwersalne – za jego pomocą można opisać różne zadania. Niekiedy jednak działanie algorytmu z kodowaniem binarnym chromosomów może być nieefektywne. W związku z tym rozpoczęto badania nad innymi sposobami kodowania problemów. Modyfikacje klasycznego algorytmu genetyczne nie ominęły sposobów kodowania, operatorów genetycznych, co jest wyrazem uszczegóławiania algorytmów genetycznych, pod kątem danego problemu.

1.6.1. Cechy dobrego kodowania

Kodowanie w algorytmach genetycznych przekształca przestrzeń rozwiązań zadania w postać, która może być poddana operacjom genetycznym. Ważne aby wszelkie własności środowiska (fenotyp) miały swoje odzwierciedlenie w sferze algorytmu genetycznego (genotyp). Dzięki dobremu odwzorowaniu fenotypu na genotyp możliwe jest efektywne przeszukiwanie przestrzeni rozwiązań. Formalnie cechy dobrego kodowania można zapisać następująco:

1. $\underset{x}{\forall}\,\underset{X}{\exists} : F(X) = x$

2. $|x_1 - x_2| \geq |x_1 - x_3| \Rightarrow |X_1 - X_2| \geq |X_1 - X_3|$

x - punkt w przestrzeni problemu (fenotypu)

X - punkt w przestrzeni genotypu

F - funkcja dekodująca genotyp na fenotyp, czyli $x = F(X)$

Pierwszy z postulatów oznacza tyle, że każde rozwiązanie zadania jest możliwe do przedstawienia jako genotyp. Niespełnienie tego postulatu oznaczałoby sytuację, w

której pewne punkty dziedziny zadania nie miały by swojego genotypowego odpowiednika. Być może któryś z takich punktów mógłby zawierać rozwiązanie określonego zadania, tym samym algorytm nie miałby możliwości dotrzeć do tego rozwiązania. Przykładowo chcemy znaleźć maksimum (w dziedzinie liczb całkowitych) funkcji $f(x) = x$ w przedziale $[0;8]$ (przykład trywialny). Problem kodujemy binarnie, długość chromosomów wynosi 3. Oczywistym jest, że maksimum to wynosi 8 (wynika to z własności funkcji liniowej). Jednak czy byłoby możliwe osiągnięcie tego rozwiązania przy tak przyjętym kodowaniu? Odpowiedzi udzieli poniższa tabela.

0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111
8	**1000**

Tabela 1.3. Liczby w systemie dziesiętnym i ich dwójkowe odpowiedniki (opracowanie własne)

Jak widać, okazuje się, że nie istnieje w przestrzeni genotypu odpowiednik dla wartości 8 w przestrzeni fenotypu. Jak wiemy punkt ten jest rozwiązaniem zadania, zatem okazało się, że tak sformułowane kodowanie uniemożliwiłoby algorytmowi znalezienie optymalnego rozwiązania.

Drugi postulat dobrego kodowania mówi o zachowaniu proporcji w odległościach miedzy odpowiednimi punktami w przestrzeni fenotypu i ich genotypowych odpowiednikach. Realizacja tego postulatu gwarantuje, że algorytm nie będzie musiał się borykać z dodatkowymi minimami lokalnymi (każde lokalne minimum może stanowić „pułapkę" dla algorytmu) poza tymi, które są odpowiednikami minimów z przestrzeni fenotypu.

1.6.2. Rodzaje kodowania

Odpowiedni sposób kodowania problemu (jak i dobór odpowiednich operatorów

genetycznych) jest kluczem do efektywności algorytmu genetycznego. Kodowanie powinno być odzwierciedleniem natury problemu. Mimo, że tak naprawdę wszystkie problemy da się opisać za pomocą łańcuchów binarnych, to często okazuje się to nieefektywne oraz utrudnia implementacje. Poszukiwane są alternatywne sposoby przedstawiania zadań, często okazujące się lepszymi od klasycznego binarnego chromosomu. W bieżącym paragrafie zostaną przedstawione trzy sposoby kodowania: za pomocą ciągów binarnych, liczb zmiennoprzecinkowych oraz liczb całkowitych.

Kodowanie binarne

W kodowaniu tym chromosomy są przedstawione jako łańcuchy binarne – ciągi zer i jedynek. Taki łańcuch można potraktować jako liczbę w systemie dwójkowym. Na przykład dla długości chromosomu 5 można za pomocą ciągów binarnych przedstawić wszystkie liczby całkowite od 0 do 31. Jest to pewne ograniczenie w przypadku gdy chcemy podać rozwiązanie z większą dokładnością. Aby uzyskać pożądaną dokładność należy użyć większej ilości bitów w chromosomie, tak aby każdej wartości liczby całkowitej, którą reprezentuje określony chromosom, odpowiadał przedział wartości o szerokości równej postulowanej dokładności. Problem pojawia się, gdy chcemy użyć naprawdę dużej dokładności, przedziału o relatywnie dużej szerokości oraz gdy wymiar problemu jest duży (duża liczba zmiennych). Duża dokładność, szeroki zakres oraz wielowymiarowość powodują dramatyczny wzrost potrzebnej liczby bitów i co za tym idzie obniżenie efektywności algorytmu. Przykładowo dla 100 zmiennych, których wartości należą do dziedziny o zakresie od –500 do 500 oraz dokładności 6 miejsc po przecinku potrzebny jest chromosom o długości 3000 bitów. Łańcuchy binarne umożliwiają kodowanie problemów optymalizacji zarówno numerycznej jaki i kombinatorycznej, jednak algorytmy genetyczne z kodowaniem binarnym w tych przypadkach są nieefektywne.

Kodowanie zmiennoprzecinkowe

Możliwe jest przedstawienie chromosomu jako wektora liczb rzeczywistych. Reprezentacja taka wydaje się bardziej naturalna dla problemów gdzie wartości zmiennych należą do zbioru liczb rzeczywistych. Zastosowanie takiego kodowania uniemożliwia skorzystanie z klasycznych operatorów genetycznych z oczywistych względów. Kodowanie zmiennoprzecinkowe wymusza zastosowanie odpowiednich operatorów, właściwych dla specyfiki liczb rzeczywistych. Zastosowanie tego schematu

kodowania umożliwia przeprowadzać przeszukiwanie w relatywnie szerokich dziedzinach, w przeciwieństwie do kodowania binarnego, gdzie przy stałej liczbie bitów wraz ze wzrostem wymiaru zadania, rezygnuje się z dokładności. Wyniki eksperymentów świadczą o większej szybkości algorytmu z kodowaniem zmiennoprzecinkowym. Nieocenioną zaletą tego kodowania jest jego naturalny intuicyjny charakter dla określonych problemów, co pozwala na łatwiejsze projektowanie wyspecjalizowanych operatorów genetycznych.

Kodowanie całkowitoliczbowe

Podobnie jak kodowanie za pomocą wektora liczb rzeczywistych, kodowanie za pomocą wektorów liczb całkowitych, jest wskazane dla zadań optymalizacji dyskretnej. W problemach takich zmienne przyjmują wartości z pewnego skończonego zbioru (przykładem takiego problemu jest TSP). Analogicznie do reprezentacji zmiennoprzecinkowej, w kodowaniu liczbami całkowitymi, należy określić wyspecjalizowane operatory genetyczne.

Idea tworzenia nowych sposobów kodowania kłóci się z ogólnością podejścia charakterystycznego dla algorytmów genetycznych. Ich niewątpliwą zaletą jest niezależność od specyfiki rozpatrywanego problemu. Niestety rezultaty, jakie generuje prosty algorytm genetyczny, nie stanowią silnej konkurencji dla wyspecjalizowanych metod optymalizacji, które korzystają z wiedzy specyficznej dla rozpatrywanego problemu.

1.7. Operatory genetyczne

Modyfikacje klasycznego prostego algorytmu genetycznego odnoszą się również do operatorów genetycznych. Postaci operatorów są związane z typem kodowania, jakiego użyto dla konkretnego problemu. Przy określonym typie kodowania operatory powinny korzystać z wiedzy o problemie oraz być projektowane tak by operowały na cegiełkach, specyficznych dla problemu (w przypadku operatora krzyżowania). Podobnie jak w przypadku kodowania, ważne jest aby operatory genetyczne posiadały określone własności.

1.7.1. Cechy dobrych operatorów genetycznych

Operatory genetyczne działają w przestrzeni genotypu. Operując na łańcuchach kodowych – chromosomach. Zasadniczo wyróżnia się dwie grupy operatorów: krzyżowanie i mutację. Aby algorytm genetyczny funkcjonował poprawnie i wydajnie muszą one posiadać pewne własności.

Pierwszym warunkiem jaki musi spełniać operator genetyczny jest utrzymywanie spójności przestrzeni genotypów. Spełnienie tego postulatu oznacza, że algorytm musi zawierać co najmniej jeden taki operator, dzięki któremu będzie możliwe wygenerowanie dowolnego chromosomu z dowolnego innego chromosomu. Własność ta musi być spełniona dla każdego chromosomu w populacji. Dzięki jej spełnieniu nie może dojść do sytuacji, w której część chromosomów będzie z góry niedostępna (a zatem i część rozwiązań zadania). Z reguły, za spełnienie postulatu spójności odpowiedzialny jest operator mutacji. Prosty algorytm genetyczny z jednym tylko operatorem – krzyżowaniem jednopunktowym, nie zapewnia spójności. Jeśli w populacji znalazłyby się wyłącznie kopie jednego osobnika, to operator krzyżowania generowałby jedynie kolejne kopie tego osobnika. Z kolei gdyby algorytm zawierał samą mutację warunek spójności byłby spełniony. Pod tym względem mutacja jest operatorem wystarczającym.

Drugim warunkiem jaki powinny spełniać operatory genetyczne jest brak obciążeń. Operator jest obciążony, gdy przy braku nacisku selektywnego, niektóre chromosomy będą osiągane częściej. W takim wypadku powinna występować sytuacja odwrotna – wszystkie chromosomy powinny być jednakowo osiągalne. W przypadku spełnienia warunku braku obciążeń dla mutacji, osiągniecie dowolnego chromosomu powinno być możliwe z jednakowym prawdopodobieństwem. Dla operatora krzyżowania warunek braku obciążeń powoduje to, że nawet wtedy, gdy w wyniku działania tego operatora wykreowane zostaną słabo przystosowane osobniki, to dalsze krzyżowanie będzie w stanie poprawić średnie przystosowanie populacji.

1.7.2. Operator krzyżowania

W toku badań nad algorytmami genetycznymi powstało wiele operatorów krzyżowania, specyficznych dla danego problemu. O mnogości operatorów krzyżowania świadczy m.in. zawartość pozycji [5], która stanowi zbiór sklasyfikowanych operatorów.

Operatory krzyżowania, które będą teraz przedstawione, są podzielone na dwie

grupy. Pierwsza jest charakterystyczna dla kodowania binarnego – grupa operatorów krzyżowania wymieniającego, druga – dla kodowania zmiennoprzecinkowego – operatory krzyżowania uśredniającego.

Krzyżowanie dwupunktowe

Operator ten jest próbą usprawnienia klasycznego krzyżowania jednopunktowego, występującego w prostym algorytmie genetycznym. Różnica w stosunku do krzyżowania jednopunktowego polega na tym, że losowane są dwa punkty rozcięcia. Wymianie podlega fragment chromosomu między punktami rozcięcia. Dla dwóch chromosomów X^1 oraz X^2 krzyżowanie wygląda następująco:

osobniki rodzicielskie:

$$X^1 = [X_1^1 \dots X_n^1]$$

$$X^2 = [X_1^2 \dots X_n^2]$$

osobniki potomne:

$$Y^1 = [X_1^1 \dots X_{C_1}^1, X_{C_1+1}^2 \dots X_{C_2}^2, X_{C_2+1}^1 \dots X_n^1]$$

$$Y^2 = [X_1^2 \dots X_{C_1}^2, X_{C_1+1}^1 \dots X_{C_2}^1, X_{C_2+1}^2 \dots X_n^2]$$

Krzyżowanie równomierne

Chromosom potomny za pomocą tego operatora jest tworzony następująco:

$$Y_i = \begin{cases} X_i^1 & gdy \quad \xi_{U(0,1),i} < p_e \\ X_i^2 & w \quad przeciwnym \quad wypadku \end{cases}$$

gdzie

$\xi_{U(0,1),i}$ oznacza niezależną realizację dla każdego i zmiennej losowej rozkładu równomiernego ze zbioru {0,1}; w tym przypadku zmienna losowa ma wartości dyskretne

p_e jest parametrem krzyżowania (zwykle przyjmuje się $p_e = \frac{1}{2}$)

Operator krzyżowania równomiernego, w przeciwieństwie do operatorów krzyżowania jedno- i dwupunktowego, jest nieobciążony.

Krzyżowanie uśredniające

Jest przykładem krzyżowania dla kodowania zmiennoprzecinkowego. Jego cechą jest to, że wartość każdego genu osobnika potomnego zawiera się miedzy minimalną a maksymalną wartością genu osobników rodzicielskich. Krzyżowanie uśredniające przebiega według schematu:

$$Y^1 = X^1 + \xi_{U(0,1)} \cdot (X^2 - X^1)$$

$$Y^2 = X^2 + X^1 - Y^1$$

Operator krzyżowania uśredniającego gwarantuje spójność, generowane przez niego chromosomy będą zawsze zawierać się w przestrzeni genotypu.

1.7.3. Operator mutacji

Drugim, kluczowym operatorem, obok operatora krzyżowania, w algorytmie genetycznym jest mutacja. Często mutacja traktowana jest jako operator drugoplanowy, jednak to mutacja zapewnia spójność przestrzeni genotypów. Wynika z tego, że operator mutacji jest niezbędny.

Mutacja, niezależnie od zastosowanego sposobu kodowania, polega na losowej perturbacji chromosomu. W prostym algorytmie genetycznym (z reprezentacją binarną) mutacja polegała na zamianie losowych bitów na przeciwne:

$$Y_i = 1 - Y_i$$

Decyzja o zamianie konkretnego bitu zapadała niezależnie dla każdego bitu z pewnym prawdopodobieństwem p_M.

Dla kodowania zmiennopozycyjnego mutacja odbywa się według wzoru:

$$Y_i = X_i + \xi_{r,i}$$

gdzie $\xi_{r,i}$ jest realizacją zmiennej losowej o rozkładzie r.

1.8. Zastosowania algorytmów genetycznych

Algorytm genetyczny stanowi narzędzie przeszukiwania przestrzeni rozwiązań zadania w zakodowanej postaci. Przeszukiwanie ma charakter optymalizacyjny, gdyż poszukuje najlepszego, według pewnego kryterium, najlepszego rozwiązania. Własność ta została wykorzystana w różnych zadaniach optymalizacyjnych, często także w

rzeczywistych, praktycznych zastosowaniach. Z wiadomych względów algorytmy genetyczne są wykorzystywane w zadaniach optymalizacji sformułowanych wprost, zwłaszcza w optymalizacji kombinatorycznej (problem komiwojażera).
Przykładowe zastosowania algorytmów genetycznych w praktyce:

- oprogramowaniu wspomagającym projektowanie (CAD)
- projektowanie rozłożenia elementów elektronicznych na płytce krzemu
- algorytmy genetyczne jako narzędzie wspomagania decyzji
- optymalizacja pracy gazociągu
- obróbka medycznych obrazów rentgenowskich
- optymalizacja wielkości łączy w sieci łącznościowej
- projektowanie układu klawiatury

1.9. Przykład działania prostego algorytmu genetycznego

Rozważany w tym miejscu prosty przykład ma charakter poglądowy. Jego celem jest pokazanie mechanizmów przetwarzania genetycznego, głownie selekcji oraz działania operatorów genetycznych na konkretnych ciągach kodowych, w możliwie najprostszej postaci. Rozwiązywany problem jest całkowicie sztuczny i raczej bezużyteczny w praktyce. Przykład ten został przedstawiony w pozycji [4].

Zadanie polega na znalezieniu chromosomu z możliwie największą liczbą jedynek. Rozwiązanie jest oczywiste – przy ustalonej liczbie bitów (np. 12 bitów) – rozwiązanie stanowi chromosom złożony z 12 jedynek. Przeszukiwanie przebiega zgodnie z założeniami prostego algorytmu genetycznego – selekcja metodą ruletki (w zasadzie reprodukcja metodą ruletki oraz sukcesja z całkowitym zastępowaniem), krzyżowanie jednopunktowe i mutacja bitowa. Dalsze parametry to: długość chromosomu – 12 bitów, liczebność populacji – 8 osobników, prawdopodobieństwo krzyżowania – 1, mutacji – 0, wykonana zostanie jedna iteracja algorytmu.

Pierwszym krokiem algorytmu jest zainicjowanie populacji bazowej. Inicjacja sprowadza się to do serii losowań wartości {0;1}. Jako, że osobników ma być 8, każdy ma po 12 bitów (lub genów, terminy używane w tym kontekście zamiennie), daje to 96 rzutów monetą. Następnie populacja początkowa podlega ocenie na podstawie funkcji przystosowania, która w tym przypadku przyporządkowuje osobnikowi liczbę jedynek jakie on zawiera (wartość funkcji przystosowania jest sumą bitów chromosomu).

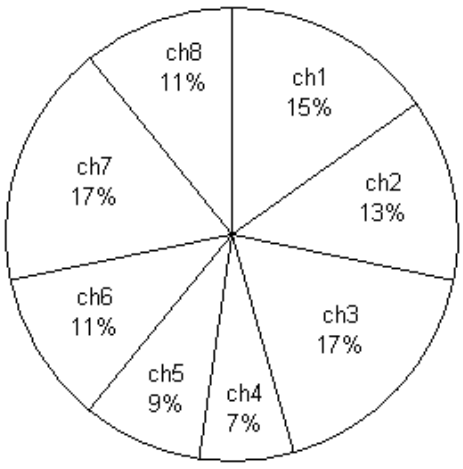
ch1	111001100101
ch2	001100111010
ch3	011101110011
ch4	001000101000
ch5	010001100100
ch6	010011000101
ch7	101011011011
ch8	000010111100

Tabela 1.4. Przykładowa populacja, kodowanie binarne (opracowanie własne na podstawie [4])

Liczona jest suma przystosowań oraz prawdopodobieństwa reprodukcji:

		F(ch)	v(ch)
ch1	111001100101	7	15%
ch2	001100111010	6	13%
ch3	011101110011	8	17%
ch4	001000101000	3	7%
ch5	010001100100	4	9%
ch6	010011000101	5	11%
ch7	101011011011	8	17%
ch8	000010111100	5	11%
	suma F(ch)	46	

Tabela 1.5. Przystosowanie osobników populacji (opracowanie własne na podstawie [4])



Rysunek 1.6. Koło ruletki dla danych z przykładu (opracowanie własne na podstawie [4])

Znając prawdopodobieństwa reprodukcji poszczególnych osobników, dokonuje się selekcji, losując 8 liczb z przedziału [0;1]. Załóżmy, że wylosowano następujące liczby:

79 44 9 74 44 86 48 23

W dla każdej wylosowanej liczby osobnik do reprodukcji jest wybierany na podstawie wartości dystrybuanty. Wybrany zostaje ten, którego wartość dystrybuanty jest największa, mniejsza od wylosowanej liczby. Zatem dla wylosowanych liczb oznacza to wybranie osobników:

ch7 ch3 ch1 ch7 ch3 ch7 ch4 ch2.

Kolejnym etapem jest losowe skojarzenie wybranych osobników w pary i przeprowadzenie operacji krzyżowania. Ponieważ prawdopodobieństwo krzyżowania zostało ustalone na 1, to wszystkie wylosowane pary będą poddane operacji krzyżowania. Załóżmy że wylosowano następujące pary osobników do krzyżowania:

ch2 i ch7 ch1 i ch7 ch3 i ch4 ch3 i ch7.

Aby przeprowadzić krzyżowanie konieczne jest wylosowanie punktu krzyżowania. Załóżmy, że dla pierwszej pary wylosowano 4, drugiej – 3, trzeciej – 11, czwartej – 5.

rodzice	punkt krzyżowania	potomkowie
001100111010		001111011011
101011011011	4	101000111010
111001100101		111011011011
101011011011	3	101001100101
011101110011		011101110010
001000101000	11	001000101001
011101110011		011101011011
101011011011	5	101011110011

Tabela 1.6. Wyniki krzyżowania (opracowanie własne na podstawie [4])

Potomkowie tworzą nową populację bazową, która podana zostaje ocenie i pętla

algorytmu się zamyka.

		F(ch)
ch1	001111011011	8
ch2	101000111010	6
ch3	111011011011	9
ch4	101001100101	6
ch5	011101110010	7
ch6	001000101001	4
ch7	011101011011	8
ch8	101011110011	8
	suma F(ch)	56

Tabela 1.7. Ponowna ocena przystosowania (opracowanie własne na podstawie [4])

Gdyby prawdopodobieństwo mutacji było większe od zera, należałoby właśnie z takim prawdopodobieństwem przeprowadzić losowe zamiany bitów dla wszystkich osobników w populacji.

Populacja potomków cechuje się wyższym średnim przystosowaniem niż ich rodzice. Dodatkowo, w wyniku krzyżowania, w populacji potomnej znalazł się osobnik o wyższej wartości przystosowania niż najwyższa wartość wśród rodziców. W kolejnych iteracjach należy się spodziewać dalszej poprawy średniego przystosowania, aż do znalezienia osobnika o przystosowaniu równym 12 - ciągu samych jedynek, czyli rozwiązania zadania.

2. Problem komiwojażera

2.1. Optymalizacja

W praktyce inżynierskiej, w ekonomii, czy wreszcie w życiu codziennym człowiek staje przed różnymi wyborami. Podejmowane decyzje z reguły nie są przypadkowe. Z każdą hipotetyczną decyzją związana jest pewna subiektywna ocena. Oczywistym jest, że chcemy podejmować, które będą korzystne. Często zbiór decyzji jest ograniczony a wybór korzystnego rozwiązania jest oczywisty. Jednak są sytuacje, gdy problem decyzyjny jest na tyle złożony, że niemożliwe jest podjęcie decyzji tylko na podstawie subiektywnej oceny.

Optymalizacja jest procesem poszukiwania najlepszego rozwiązania dla danego problemu (*Optimus* – łac. najlepszy). Jest to ścisłe rozumienie optymalizacji. W praktyce często zadowalające jest znalezienie rozwiązania lepszego niż znane dotychczas. Kluczowym składnikiem procesu optymalizacji jest kryterium (funkcja celu), na podstawie którego możliwa jest obiektywna ocena rozwiązań (decyzji) oraz ich porównywanie. Optymalizując poszukujemy rozwiązania o największej lub najmniejszej możliwej wartości funkcji celu.

Kluczowym dla optymalizacji problemu identyfikacja klasy zadania oraz wybór odpowiedniej metody, algorytmu bądź procedury postępowania, prowadzącej do osiągnięcia optimum. Powstało wiele metod optymalizacji dla konkretnych klas problemów.

2.1.1. Rodzaje zadań

Zadania ciągłe

Przestrzeń przeszukiwań w tego typu zadaniach jest podzbiorem iloczynu kartezjańskiego zbioru liczb rzeczywistych.

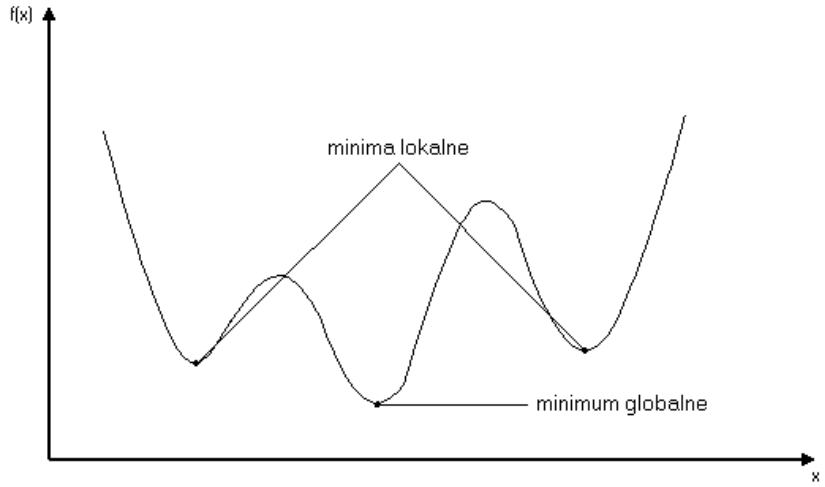
Zadania dyskretne

Wartości zmiennych należą do zbioru dyskretnego (skończonego lub przeliczalnego).

Zadania mieszane

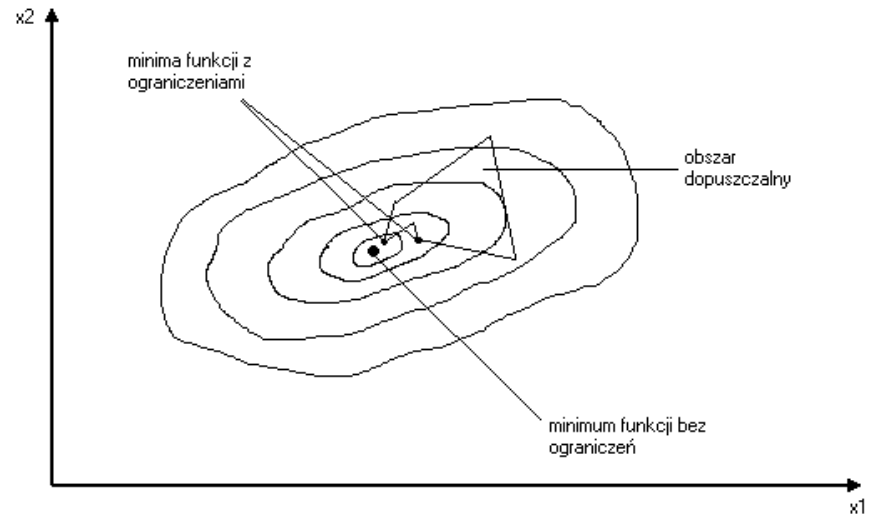
Część zmiennych przyjmuje wartości całkowite, część – rzeczywiste.

Stopień trudności każdego zadania wynika z postaci funkcji celu oraz zbioru dopuszczalnego. Pewne własności funkcji celu stanowią ułatwienie w rozwiązywaniu zadania, inne są przeszkodą. Ułatwieniami takimi są na przykład: liniowość, wypukłość, różniczkowalność funkcji celu.



Rysunek 2.1. Funkcja posiadająca wiele minimów lokalnych (opracowanie własne na podstawie [1])

Z kolei utrudnieniem jest na istnienie wielu minimów lokalnych funkcji celu. Jeżeli chodzi o zbiór dopuszczalny to utrudnieniem jest jego niewypukła lub niespójna postać.



Rysunek 2.2. Niewypukły zbiór dopuszczalny (opracowanie własne na podstawie [1])

2.1.2. Metody optymalizacji

Różnorodność zadań optymalizacji spowodowała, że powstało wiele metod, uwzględniających specyfikę problemu, jak na przykład własności funkcji celu. Trzy

główne grupy metod optymalizacji to: metody analityczne, kombinatoryczne oraz losowe.

Metody analityczne

Zastosowanie tych metod odnosi się do zadań ciągłych, gdy mamy do czynienia z wypukłą funkcją celu. Dzielą się one na dwie podgrupy: metody pośrednie i bezpośrednie. W metodach pośrednich gradient funkcji celu przyrównywany jest do zera – jest uogólnienie metody punktów krytycznych z rachunku różniczkowego dla przypadku wielowymiarowego. Metody te dotyczą niestety wąskiej grupy zadań, gdyż wymagają dodatkowych założeń dotyczących funkcji celu – funkcja celu w postaci analityczne, różniczkowalność. Druga grupa – metody bezpośrednie – wykorzystuje gradient funkcji celu do określenia kierunku przeszukiwania. Przeszukiwanie przebiega w kierunku, który zapewnia maksymalną poprawę wartości funkcji celu. Do grupy metod bezpośrednich zalicza się: metodę największego spadku, gradientów sprzężonych. Istnieje grupa metod nie korzystających z gradientu funkcji celu. Są nimi: metoda poszukiwań prostych oraz kierunków sprzężonych.

Metody kombinatoryczne

Cechą tej grupy metod jest to, że gwarantują uzyskanie rozwiązania optymalnego. Idea metod kombinatorycznych jest zupełnie oczywista. Polega ona na sprawdzaniu kolejno wszystkich punktów zbioru rozwiązań z pamiętaniem najlepszego rezultatu. Niestety podejście takie jest zupełnie nieefektywne przy dużych przestrzeniach a ich zastosowanie możliwe jest tylko dla małych problemów. Usprawnieniem pełnego przeglądu są metody podziału i ograniczeń, gdzie część rozwiązań jest pomijana przy przeszukiwaniu. Niestety metoda ta ma te same wady co pełny przegląd (efektywność tylko dla relatywnie niewielkich problemów).

Metody losowe

Metody te nie gwarantują znalezienia optimum, jednak nie dotyczą ich ograniczenia typowe dla metod analitycznych i przeglądowych. Najprostszym przykładem metody losowej jest błądzenie przypadkowe (metoda Monte Carlo), w którym losowane są kolejne punkty z przestrzeni rozwiązań i pamiętany jest najlepszy dotychczas znaleziony wynik. Jednak podejście to nie jest lepsze od pełnego przeglądu. Odmianą metod losowych są algorytmy zrandomizowane. Przykładem takiej metody są

algorytmy genetyczne. W tym przypadku losowy wybór wspomaga przeszukiwanie. Innym przykładem metody losowej jest symulowane wyżarzanie, której zasada działania jest podobna do błądzenia przypadkowego z tą różnicą, że najlepsze dotychczas znalezione rozwiązanie nie zawsze zostaje zapamiętane.

2.1.3. Ocena algorytmów optymalizacji.

W praktyce ścisła matematyczna definicja rozwiązania optymalnego często jest często weryfikowana przez różne czynniki zewnętrzne. Bywa tak, że za optymalne przyjmowane jest rozwiązanie wcale nie najlepsze. Wynika to z różnych względów. Można wyobrazić sobie sytuację gdy znalezienie optimum wymaga niesłychanych nakładów (na przykład wielkiej mocy obliczeniowej komputera, czasu) podczas gdy rozwiązanie różniące się od optymalnego nieznacznie jest dostępne przy małych nakładach (kilka sekund na domowym komputerze). W takiej sytuacji należałoby się zastanowić nad odejściem od ścisłej matematycznej definicji rozwiązania optymalnego, na rzecz jej bardziej pragmatycznej wersji. Ponieważ optymalizacja jest zagadnieniem szeroko spotykanym w praktyce, metody optymalizacji powinny być oceniane nie tylko pod względem skuteczności znajdowania optimum (w ścisłym sensie) lecz także pod względem innych, czysto praktycznych, czynników. Najważniejsze czynniki oceny metod optymalizacyjnych to: jakość rozwiązania, odporność oraz koszt.

Jakość rozwiązania

Dokładność rozwiązania określa się jako odległość od rozwiązania optymalnego:

$$\left|x^* - x\right|$$

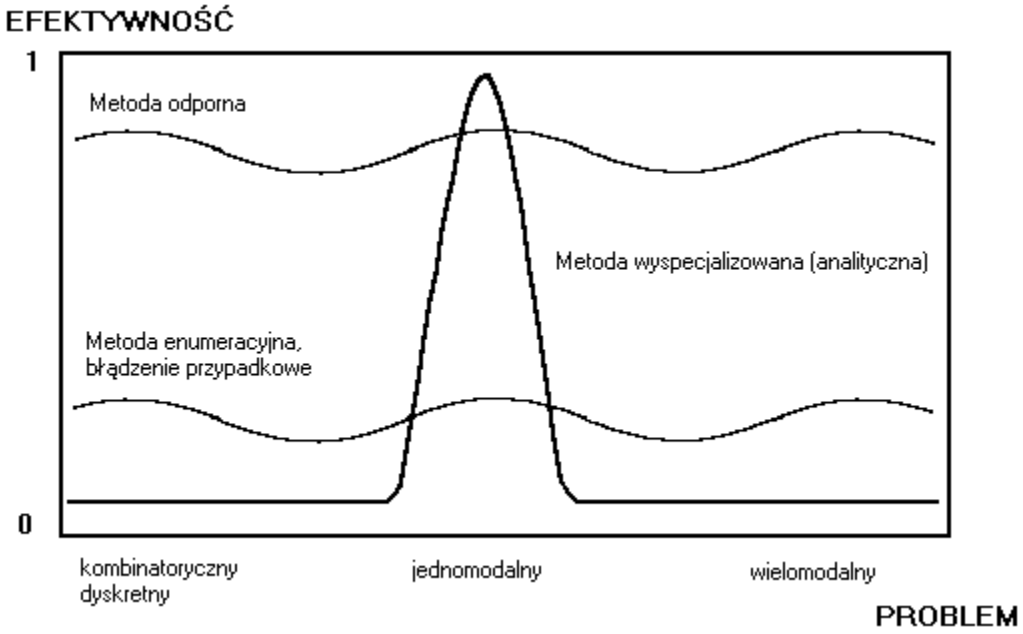
lub jako przybliżenie wartości funkcji celu w poszukiwanym rozwiązaniu

$$\left|f(x^*) - f(x)\right|$$

Oczywiście dokładne określenie jakości rozwiązania jest możliwe tylko wtedy gdy znane jest poszukiwane optimum. Jest to założenie niepraktyczne, gdyż właśnie po to rozwiązujemy zadanie aby to optimum uzyskać, zatem nie jest ono znane. Jednak jakość generowanego rozwiązania można określić pośrednio dla innego, znanego rozwiązania podobnego problemu. Jeżeli okaże się, że metoda daje dobre rezultaty dla podobnego problemu testowego, należy się spodziewać równie dobrego rozwiązania problemu rozważanego.

Odporność

Przykładowo w zadaniach z wypukłą, różniczkowalną funkcją celu nie ma problemu ze znalezieniem rozwiązania optymalnego. Stosuje się wtedy metodę analityczną, z wykorzystaniem pochodnych. Dla problemów małych rozmiarów skuteczną może się okazać metoda pełnego przeglądu lub proste błądzenie losowe. Problemy zaczynają się gdy przestrzeń rozwiązań jest relatywnie duża a funkcja celu posiada wiele ekstremów. Może się zdarzyć taka sytuacja, że przykładowo metoda największego spadku ominie minimum globalne, lokalizując lokalne minimum funkcji. Gdyż kierunek poszukiwań zależy bezpośrednio od wyboru punktu startowego. „Pułapki" związane z maksimami lokalnymi są największym problemem algorytmów optymalizacji. Odporność metody określa jej przydatność dla konkretnego zadania. Idealna odporna metoda powinna sobie radzić z szeroką gamą zadań, lokalizując ich optymalne rozwiązania. Oczywiście nie istnieje taka metoda. Cześć metod radzi sobie dobrze z jednym typem zadań, nie radząc sobie kompletnie z innymi. Przykładowo metody analityczne oparte na pochodnych dają rozwiązania optymalne ale tylko w bardzo wąskiej grupie problemów (dla funkcji celu bez minimów lokalnych). Metody analityczne dają bardzo dobre rezultaty (nawet optymalne) dla wąskiej grupy zadań (zadań, w których funkcja celu na odpowiednie własności). Z kolei metody przeglądowe są jednakowo nieefektywne dla wszystkich zadań. W tym kontekście metoda odporna byłaby wydajna dla wszystkich zadań. Rysunek 2.3. przedstawia schematycznie zależność między typem metody a jej efektywnością dla konkretnego rodzaju zadania. Orientacyjne wartości 0 i 1 na osi Efektywności oznaczają odpowiednio brak efektywności (tzn. że metoda nie nadaje się do określonego problemu) oraz pełną efektywność (metoda idealna dla danego problemu). Przykładowo metody analityczne są zdecydowanie najbardziej efektywne w grupie problemów, w których funkcja celu ma jedno minimum (maksimum). W innych przypadkach metody analityczne charakteryzują się praktycznie zerową efektywnością. Z kolei metoda odporna to taka, która w szerokiej klasie problemów gwarantuje stosunkowo wysoką efektywność.



Rysunek 2.3 Metoda odporna (opracowanie własne na podstawie [2])

Algorytm genetyczny jest metodą charakteryzującą się wysoką odpornością szerokiego zakresu problemów. W pewnych zadaniach nie radzi sobie tak dobrze ja na przykład metody analityczne, lecz w zamian za to daje dobre rezultaty w zadaniach gdzie metody analityczne w ogóle nie mogą być zastosowane.

Koszt

Określenie kosztu związanego z metodą optymalizacji w praktyce sprowadza się do określenia zasobów komputera (takich jak czas procesora czy ilość pamięci), potrzebnych do przeprowadzenia symulacji tego algorytmu. Wynika to z tego, że w praktyce większość metod posiada swoje implementacje komputerowe i nikt nie wyobraża sobie innego sposobu ich realizacji. Koszt to przede wszystkim czas procesora potrzebny do wykonania obliczeń. Koszt metody jest zdeterminowany przez złożoność algorytmu oraz liczbę elementarnych operacji w każdym jego kroku. Te czynniki stanowią o czasie wykonania algorytmu. Koszt często stanowi kryterium zatrzymania algorytmu. W praktyce jest to liczba iteracji lub z góry określony czas wykonania, po którym algorytm jest zatrzymywany. Ze względu na koszt często rezygnuje się z jakości rozwiązania aby uzyskać rozwiązanie dostatecznie dobre (według subiektywnej oceny) w rozsądnym czasie.

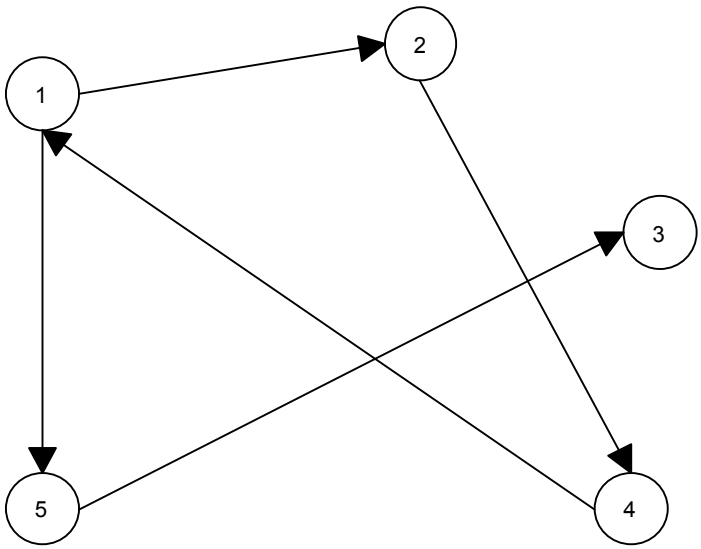
2.2. Charakterystyka problemu komiwojażera

Problem komiwojażera (TSP) jest klasycznym zadaniem optymalizacji dyskretnej. Zajmuje on szczególne miejsce wśród problemów tego typu ze względu na swoją prostą definicję oraz rzeczywiste, niebanalne zastosowania. Prace nad metodami rozwiązywania problemu trwają od bardzo dawna, czego rezultatem jest powstanie wielu różnych podejść do tego zadania, między innymi podejścia związanego z wykorzystaniem algorytmów genetycznych. Najefektywniejsze znane dzisiaj algorytmy są w stanie rozwiązywać przykłady zadania komiwojażera dla kilkudziesięciu tysięcy miast.

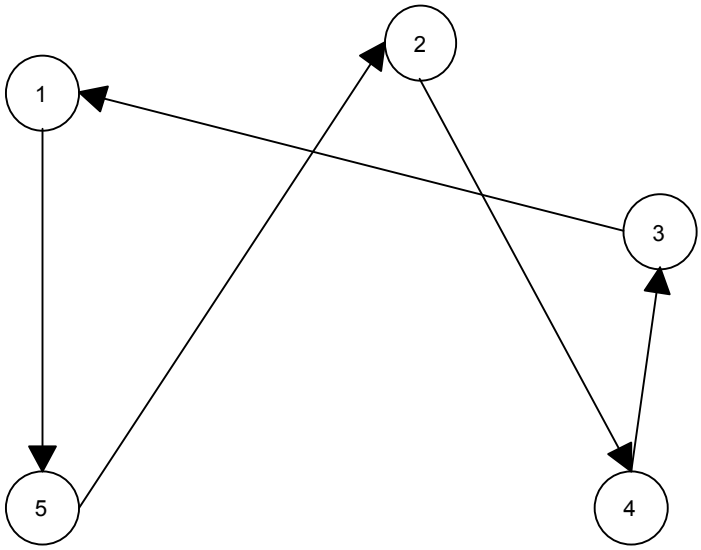
Problem komiwojażera po raz pierwszy został sformułowany w roku 1759 przez Eulera. Problem ten co prawda występował pod inną nazwą ale analogia do tego co obecnie rozumie się przez problem komiwojażera jest oczywista. Zadanie zaproponowane przez Eulera można określić jako zadanie ruchu konia szachowego po szachownicy. Polegało ono ta tym, aby konik szachowy, startując z pewnego wybranego pola, odwiedził wszystkie pozostałe pola na szachownicy, odwiedzając każde tylko raz i powrócił na pole startowe.

Pierwszy raz w literaturze sformułowanie „komiwojażer" w kontekście wyboru optymalnej trasy pojawiło się w roku 1832 w niemieckiej książce, która była swego rodzaju podręcznikiem dla osób zajmujących się handlem obwoźnym. Książka zawierała kilka tras wiodących przez miasta Niemiec i Szwajcarii, przy czym ostatnia z nich była trasą komiwojażera w ścisłym rozumieniu. Była to trasa wiodąca przez 47 niemieckich miast i charakteryzowała się bardzo dobrą jakością (rozwiązanie bliskie optymalnemu).

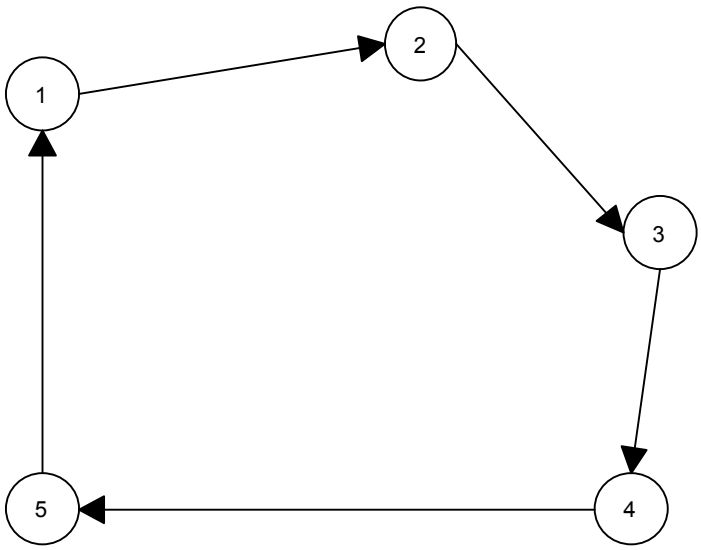
Definicja problemu komiwojażera jest bardzo prosta. Hipotetyczny komiwojażer ma do odwiedzenia pewną liczbę miast. Ogranicza go to, że każde z tych miast musi odwiedzić dokładnie raz. Zadaniem komiwojażera jest taki wybór trasy, aby była ona najkrótsza (trasa o najmniejszym koszcie) z wszystkich dopuszczalnych tras, czyli tras tworzących zamknięty cykl.



Rysunek 2.4. Trasa niedopuszczalna (opracowanie własne)



Rysunek 2.5. Przykładowa dopuszczalna trasa komiwojażera dla 5 miast (opracowanie własne)



Rysunek 2.6. Optymalna trasa komiwojażera dla 5 miast (opracowanie własne)

Problem komiwojażera można opisać także jako zagadnienie z teorii grafów. Zadanie polega na znalezienia cyklu Hamiltona (jest to cykl, w którym każdy wierzchołek występuje jeden raz) w grafie ważonym o minimalnej sumie wag krawędzi.

Problem komiwojażera mimo prostego, intuicyjnego opisu, stanowi nie lada wyzwanie dla twórców algorytmów. Jest to zadanie, dla którego pewność co do uzyskania rozwiązania optymalnego uzyskuje się po pełnym przeglądzie wszystkich tras. Pełnym przegląd jest podejściem gwarantującym (w teorii) znalezienie najkrótszej trasy, jednak jest to podejście całkowicie nieefektywne. Pełny przegląd w praktyce możliwy jest do zastosowania dla problemów o małych rozmiarach (do 10 miast). Każde kolejne miasto powoduje ogromny wzrost złożoności. Liczba możliwych tras wynosi $n!$ (uwzględniając powtarzające się trasy) czyli pełny przegląd polega na zbadaniu wszystkich $n!$ permutacji liczb $\{1,2...n\}$. O tym jak szybko rośnie złożoność problemu świadczą wartości $n!$ dla kolejnych n:

0! = 1
1! = 1
2! = 2
3! = 6
4! = 24
5! = 120
6! = 720
7! = 5040
8! = 40320
9! = 362880
10! = 3628800
11! = 39916800
12! = 479001600
13! = 6227020800
14! = 87178291200
15! = 1307674368000
...
49! = 608281864034267560872252163321295376887552831379210240000000000
50! = 30414093201713378043612608166064768844377641568960512000000000000

Jak widać dla problemów o relatywnie niewielkich rozmiarach (rzędu 50) liczba możliwych tras jest astronomiczna. Dla ścisłości należy dodać, że część z tych tras jest tożsama, na przykład dla problemu o wielkości 5 trasy:

1 2 3 4 5

(start w mieście nr 1, potem odwiedzenie miasta 2, potem 3 itd. na koniec powrót do 1)

oraz

2 3 4 5 1

są równoważne, mimo że odpowiadają innym permutacjom. Podobnie trasy:

1 2 3 4 5

oraz

5 4 3 2 1

są takie same.

Ostatecznie liczba wszystkich unikalnych tras komiwojażera wyraża się liczbą:

$$\frac{(n-1)!}{2}$$

gdzie n jest liczbą miast.

Różnica ma niestety tylko znaczenie formalne, gdyż nie zmienia wykładniczej postaci złożoności.

Problem komiwojażera nie jest wyłącznie zagadnieniem typowo akademickim. Ma szerokie zastosowania praktyczne, zwłaszcza w technice. Przykładowym zastosowaniem jest programowanie trasy ruchu ramienia robota wykonującego otwory w płytce, którą zapełnić mają układy scalone. Oczywistym jest, że przy danym rozkładzie otworów, optymalna trasa ruchu ramienia zagwarantuje wykonanie otworów w najkrótszym czasie i najmniejszym koszcie.

2.3. Algorytm genetyczny dla problemu komiwojażera - budowa modelu

Paragraf ten jest poświęcony opisowi modelu genetycznego do rozwiązywania problemu komiwojażera. Opis modelu stanowi punkt wyjścia do implementacji komputerowej problemu, która szerzej będzie omówiona w rozdziale 3 pracy. W tym

miejscu przedstawię różne warianty poszczególnych elementów algorytmu genetycznego w kontekście zadania komiwojażera, między innymi sposobu kodowania zadania oraz operatorów genetycznych.

2.3.1. Określenie sposobu kodowania zadania

Budowa modelu algorytmu genetycznego, mającego rozwiązać określone zadanie, polega na między innymi na wyborze odpowiedniego sposobu kodowania zadania. Wydaje się, że dobór odpowiedniego kodowania ma kluczowe znaczenie dla poprawności działania algorytmu oraz jego efektywności. Oczywiście z jednej strony ogólność algorytmów genetycznych oraz ich niezależność od specyfiki problemu jest niewątpliwą zaletą ale w praktyce dokonanie pewnego uszczegółowienia algorytmu daje lepszy efekt. W przypadku problemu komiwojażera możliwe jest przedstawienie chromosomu w różny sposób. Przedstawię kilka sposobów kodowania zadania komiwojażera, żeby zobrazować różne podejścia do zagadnienia reprezentacji chromosomu. Paragraf ten zaprezentuje różne sposoby kodowania problemu komiwojażera ale w modelu a w konsekwencji i w aplikacji komputerowej zostanie zastosowany jeden wybrany sposób kodowania. W kolejnych paragrafach zostaną omówione miedzy innymi operatory genetyczne odpowiednie dla wybranej reprezentacji.

Reprezentacja binarna

Problem komiwojażera można próbować rozwiązywać korzystając z reprezentacji binarnej. Możliwe jest to ze względu na to, że kodowanie binarne jest najbardziej ogólną i uniwersalną metodą kodowania. Jednak praktyka pokazuje, że najczęściej podejścia ogólne są mniej efektywne od tych, które wykorzystują pewną wiedzę o problemie. Jak było wspomniane wcześniej istnieje możliwość zastosowania kodowania binarnego dla problemu komiwojażera ale jest to zadanie dosyć karkołomne. Rozwiązanie zadania, czyli trasę przedstawia się jako ciąg liczb naturalnych – numerów miast. Chromosom binarny musiałby być podzielony na odpowiednią liczbę bloków, z których każdy (czyli każda liczba w systemie dwójkowy) odpowiadałby numerowi miasta. Przykładowo trasa wiodąca przez 5 miast:

1-2-3-4-5

zakodowana jako łańcuch binarny wyglądałaby tak:

001|010|011|100|101

(dla przejrzystości kolejne liczby dwójkowe odpowiadające numerom miast zostały odseparowane znakiem „|”),

Kodowanie binarne w przypadku problemu komiwojażera powoduje spore problemy. W pokazanym wcześniej przykładzie, chromosom binarny może reprezentować trasę niedopuszczalną:

001|010|011|100|111

odpowiadałoby trasie 1-2-3-4-7 co jest oczywiście pozbawione sensu.

Podobny problem, czyli chromosom przedstawiający trasę niedopuszczalną, może się zdarzyć po przeprowadzeniu krzyżowania lub mutacji. Kolejną wadą kodowania binarnego jest jego wielkość. Przykładowo dla 100 miast do zakodowania pojedynczego miasta potrzeba 7 bitów (2^7=128) długość chromosomu będzie wynosić 700 bitów. Przy jeszcze większym problemie i dużej populacji będzie to powodować wolne działanie realizacji komputerowej.

Kolejne reprezentacje: przyległościowa, porządkowa oraz ścieżkowa zostały zaprojektowane z uwzględnieniem specyfiki problemu komiwojażera, dzięki czemu są bardziej efektywne niż kodowanie binarne. W dalszej części zostaną przedstawione wymienione reprezentacje a dla reprezentacji ścieżkowej (która będzie zastosowana w modelu) dodatkowo omówione charakterystyczne dla niej operatory genetyczne.

Reprezentacja przyległościowa

W przypadku tego sposobu kodowania trasa jest reprezentowana jako ciąg n miast. Miasto j znajduje się na pozycji i gdy trasa przebiega z miasta i do miasta j. Przykładowo:

2 4 8 3 9 7 1 5 6

oznacza trasę

1-2-4-3-8-5-9-6-7.

Niestety niektóre ciągi reprezentacji przyległościowej reprezentują trasy niedopuszczalne. Niemożliwe jest również zastosowanie klasycznego krzyżowania jednopunktowego. Dla tej reprezentacji określono trzy operatory krzyżowania: z wydzielaniem krawędzi, z wydzielaniem podtras oraz heurystyczny. Opis tych operatorów zostanie pominięty, gdyż opisana reprezentacja nie będzie wykorzystana w modelu.

Reprezentacja porządkowa

Trasa jest reprezentowana jako ciąg n miast w taki sposób, że element o indeksie i jest liczbą z zakresu od 1 do $n-i+1$. Zasada tej reprezentacji jest następująca. Dany jest uporządkowany ciąg miast C, który jest punktem odniesienia dla tej reprezentacji:

C = (1 2 3 4 5 6 7 8 9)

trasa:

1-2-4-3-8-5-9

jest reprezentowana przez ciąg wskazań:

l = (1 1 2 1 4 1 3 1 1)

co jest interpretowane w następujący sposób:

pierwszą liczbą w ciągu l jest 1, zatem bierzemy pierwsze miasto z ciągu C - jest to 1, wstawiamy do trasy częściowej i usuwamy z C. Trasa częściowa to:

1.

Kolejną liczbą w l jest znowu 1, więc bierzemy pierwszą liczbę z C - 2, usuwamy i wstawiamy do trasy częściowej:

1 – 2.

Następnie bierzemy trzecią wartość z l czyli 2 – wstawiamy drugie miasto z C czyli 4, trasa częściowa:

1 – 2 – 4.

Kolejną liczbą w l jest 1, więc bierzemy znowu pierwszą liczbę z C - 3, usuwamy i wstawiamy do trasy częściowej:

1 – 2 – 4 – 3.

Kolejną liczbą w l jest 4, więc czwartą liczbę z C - 8, usuwamy i wstawiamy do trasy częściowej:

1 – 2 – 4 – 3 - 8.

W kolejnych krokach postępujemy analogicznie do momentu aż ciąg C będzie pusty. W tym przykładzie trasa reprezentowana przez ciąg wskazań l to:

1 – 2 – 4 – 3 – 8 – 5 – 9 – 6 – 7.

Kodowanie porządkowe pozwala na zastosowanie klasycznego krzyżowania. Każde przecięcie i krzyżowanie dowolnych dwóch chromosomów w reprezentacji porządkowej prowadzi do tras dopuszczalnych.

Reprezentacja ścieżkowa

Jest to najbardziej intuicyjny, naturalny sposób kodowania trasy w problemie komiwojażera.

Przykładowa trasa:

1 – 2 – 4 – 3 – 8 – 5 – 9 – 6 – 7

jest w tym przypadku reprezentowana przez ciąg:

(1 2 4 3 8 5 9 6 7).

Kodowanie ścieżkowe nie pozwala na zastosowanie klasycznego krzyżowania, gdyż wyniki takiego krzyżowania mogą dawać trasy niedopuszczalne. Do niewątpliwych zalet tej reprezentacji należy to, że dzięki swojej postaci w łatwy sposób można obliczać długość tras tak reprezentowanych. W przypadku poprzednich reprezentacji trasy należy najpierw odkodować, co oczywiście wiąże się z wydłużeniem czasu trwania symulacji komputerowej.

Dla reprezentacji ścieżkowej znanych jest wiele operatorów krzyżowania, do najpopularniejszych należą: krzyżowanie z częściowym odwzorowaniem (PMX – *partial mapped crossover*), z porządkowaniem (OX – *order crossover*) oraz cykliczne (CX – *cycle crossover*). Te trzy operatory będą omówione szerzej w jednym z kolejnych paragrafów, gdyż z racji wyboru reprezentacji ścieżkowej jako formy kodowania w modelu będą one również w nim umieszczone. Operatory te (a także operator zachłanny GSX - *greedy subtour crossover*) będą ponadto wykorzystane w implementacji komputerowej (rozdział 3) a także zbadane pod kontem efektywności.

2.3.2. Określenie metody selekcji

W rozdziale 1 pracy zaprezentowane zostały różne sposoby selekcji (rozdzielone na reprodukcję i sukcesję). Sukcesja jest kolejnym, po reprezentacji, elementem koniecznym do zdefiniowania w modelu algorytmu genetycznego.

Jako metoda reprodukcji arbitralnie została wybrana reprodukcja proporcjonalna (metoda ruletki). Metoda ta została omówiona w poprzednim rozdziale. Jednak w związku ze specyfiką problemu komiwojażera, a konkretnie z jego charakterem jako zadania optymalizacyjnego, jest pewien problem. Metoda ruletki zakłada, że funkcja przystosowania jest maksymalizowana a w problemie komiwojażera szuka się najkrótszej trasy, czyli minimalizuje. Problem ten można ominąć poprzez odpowiednie sformułowanie funkcji przystosowania. Jeżeli przystosowanie osobnika byłaby określone przez długość trasy to poszukiwanie rozwiązanie poległoby na znalezieniu

możliwie najdłuższej trasy, co jest oczywiście niepożądane. Chodzi o to, aby chromosomy o krótszych trasach były oceniane lepiej od tych o dłuższych trasach. W związku z tym zmodyfikowana funkcja przystosowania powinna mieć następującą postać:

$$Fit(CH) = \frac{dl^*}{dl_{CH}}$$

gdzie dl^* oznacza najdłuższą trasę w populacji – czyli najgorszego osobnika

a dl_{CH} długość trasy osobnika, dla którego obliczane jest przystosowanie.

Taka postać funkcji przystosowania gwarantuje, że krótsze trasy będą oceniane lepiej niż dłuższe. Dodatkowym warunkiem, dla tak sformułowanej funkcji przystosowania, jest konieczność znalezienia osobnika o najdłuższej trasie (co w praktyce można wybyło uprościć do znalezienia osobnika o najdłuższej trasie w danej populacji). Dopiero wtedy możliwe jest obliczanie przystosowania pozostałych osobników.

Drugim wątkiem przy określaniu sposobu selekcji jest sukcesja. W modelu zastosowany będzie model sukcesji elitarnej, gdzie w każdym kroku algorytmu pamiętany będzie najlepszy osobnik (elita). Osobnik ten będzie zawsze wchodził do populacji potomnej. Zabieg taki ma na celu zwiększenie nacisku selektywnego co będzie prowadzić do szybszej zbieżności algorytmu. Innymi słowy do zlokalizowania obiecujących obszarów (rozwiązań bliskich optimum) będzie potrzebna mniejsza liczba iteracji.

2.3.3. Określenie operatorów krzyżowania

Z racji wyboru kodowania ścieżkowego do modelu jako sposobu reprezentacji zadania, wybór operatorów krzyżowania ogranicza się do operatorów typowych dla tego kodowania. Operatorami tymi są wspomniane wcześniej OX, PMX, CX oraz GSX. Przykładowe dane dla opisywanych operatorów, które będą tutaj przytoczone, pochodzą z pozycji [3] oraz [7].

Krzyżowanie OX

W krzyżowaniu z porządkowaniem, potomek jest tworzony poprzez wybór podtrasy od jednego z rodziców. Wybrana podtrasa jest wstawiana w chromosomie potomka w tym samym miejscu, w jakim występowała u rodzica. Pozostałe geny uzupełnia się, biorąc je od drugiego rodzica, przy zachowaniu ich uporządkowania. Długość oraz pozycję

podtrasy określa się poprzez losowy wybór dwóch punktów cięcia. Drugi potomek jest tworzony identycznie jak pierwszy, z zamianą kolejności rodziców. Jeżeli wylosowana długość podtrasy będzie równa długości chromosomu, wtedy potomkowie będą identyczni ze swoimi rodzicami. Przykładowo dla danej pary osobników rodzicielskich:

p_1 = (1 2 3 4 5 6 7 8 9) oraz

p_2 = (4 5 2 1 8 7 6 9 3)

krzyżowanie będzie przebiegać następująco:

Załóżmy, że wylosowano punkty cięcia c_1=3 i c_2=7. Oznacza to, że podtrasa będzie się zaczynać po trzecim a kończyć przed ósmym genem:

p_1 = (1 2 3 | 4 5 6 7 | 8 9)

p_2 = (4 5 2 | 1 8 7 6 | 9 3).

W pierwszym etapie krzyżowania, trasy między punktami cięcia są kopiowane do chromosomów potomków:

o_1 = (x x x | 4 5 6 7 | x x)

o_2 = (x x x | 1 8 7 6 | x x).

Geny oznaczone przez „x” dla potomka o_1, są uzupełniane rozpoczynając od drugiego punktu cięcia rodzica p_2. Geny są uzupełniane w tej samej kolejności, w jakiej występują w p_2, z pominięciem takich, które już w o_1 występują. Ciąg genów p_2 rozpoczynając od drugiego punktu cięcia jest następujący:

9 – 3 – 4 – 5 – 2 – 1 – 8 – 7 – 6

ale geny o wartościach 4, 5, 6 i 7 już występują w o_1 zatem rozpatrywane są tylko:

9 – 3 – 2 – 1 – 8 . Ciąg ten jest przepisywany do o_1, również rozpoczynając od drugiego punktu cięcia. Ostatecznie pierwszy potomek będzie miał postać:

o_1 = (2 1 8 4 5 6 7 9 3) .

Analogicznie tworzy się drugiego potomka. Będzie on wyglądał tak:

o_2 = (3 4 5 1 8 7 6 9 2).

Krzyżowanie PMX

W krzyżowaniu z częściowym odwzorowaniem, potomka tworzy się, podobnie jak w przypadku krzyżowania OX, poprzez wybór podtrasy z jednego rodzica oraz pozostawiając porządek i pozycje genów drugiego rodzica o ile jest to możliwe.

Podobnie jak w OX, podtrasę tworzy się, wybierając losowo dwa punkty cięcia. Dla pary rodziców p_1 i p_2 z poprzedniego przykładu oraz dla tych samych co wcześniej punktów cięcia, parę osobników potomnych tworzy się następująco:

do chromosomu potomka o_1 zostaje przepisana podtrasa z p_2 (a do o_2 podtrasa z p_1):

o_1 = (x x x | 1 8 7 6 | x x)

o_2 = (x x x | 4 5 6 7 | x x)

taka wymiana określa ciąg odwzorowań:

1 – 4, 8 – 5, 7 – 6 oraz 6 – 7.

Następnie wstawia się dalsze wartości od rodziców, dla których nie zachodzi konflikt (czyli się nie powtarzają).

o_1 = (x 2 3 | 1 8 7 6 | x 9)

o_2 = (x x 2 | 4 5 6 7 | 9 3)

Ostatnim etapem jest zastąpienie wartości, dla których zachodził konflikt na podstawie ciągu odwzorowań. Przykładowo w potomku o_1 na pierwszym miejscu powinna być wartość 1, jednak powodowała ona konflikt, więc wykorzystuje się wartość wynikającą z odwzorowania 1 – 4, czyli 4. Drugie „x" w o_1 zajmuje 5 bo istnieje odwzorowanie 8 – 5. Postępowanie takie kontynuuje się aż do skompletowania chromosomów osobników potomnych. W tym przykładzie będą nimi:

o_1 = (4 2 3 | 1 8 7 6 | 5 9)

o_2 = (1 8 2 | 4 5 6 7 | 9 3).

Krzyżowanie CX

W krzyżowaniu cyklicznym potomków tworzy się w taki sposób, że każdy gen oraz jego pozycja pochodzi od jednego z osobników rodzicielskich. Krzyżowanie CX przebiega następująco:

dla danej pary rodziców

p_1 = (1 2 3 4 5 6 7 8 9)

p_2 = (4 1 2 8 7 6 9 3 5)

potomka tworzy się, biorąc pierwszy gen z pierwszego rodzica

o_1 = (1 x x x x x x x x).

Ponieważ każda wartość genu (oraz jego pozycja) w potomku powinna być wzięta od jednego z rodziców, w tym przypadku nie mamy wyboru. Jako kolejne miasto musi

zostać wybrane 4 – miasto od rodzica p_2, które występuje zaraz za wybranym miastem 1. W rodzicu p_1 miasto to jest na pozycji „4”, zatem

o_1 = (1 x x 4 x x x x x).

miasto 4 to wskazuje na miasto 8, czyli miasto z rodzica p_2 występujące zaraz za wybranym wcześniej miastem 4. Czyli:

o_1 = (1 x x 4 x x x 8 x).

Idąc dalej zgodnie z tą regułą, następnymi miastami, które należy wstawić do potomka o_1 są 3 i 2. Teraz widać, że wybór miasta 2 wymaga wyboru 1, które jest już w o_1. Zatem cykl jest kompletny:

o_1 = (1 2 3 4 x x x 8 x).

Pozostałe miasta są wstawiane od drugiego rodzica:

o_1 = (1 2 3 4 7 6 9 8 5).

W podobny sposób tworzy się drugiego potomka, którym będzie:

o_2 = (4 1 2 8 5 6 7 3 9).

Krzyżowanie GSX

Operator krzyżowania GSX (*greedy subtour crossover*) przy tworzeniu potomka stara się zachować możliwie jak najdłuższe poddtrasy z chromosomów rodzicielskich. Krzyżowanie GSX przebiega następująco:

dla danej pary rodziców

p_1 = (4 8 2 1 3 6 7 5)

p_2 = (2 3 4 7 8 6 5 1)

potomka tworzy się, wybierając losowo miasto początkowe i wstawiając je do chromosomu potomka. Zakładamy, że losowo wybrano miasto o numerze 3:

o = (x x x 3 x x x x).

Kolejne miasta są pobierane na zmianę od pierwszego i drugiego z rodziców, dopóki jest to możliwe (tzn. do momentu, w którym dodanie kolejnego miasta utworzy podtrasę w chromosomie potomka będącą cyklem). W tym konkretnym przykładzie dodawanie miast przebiega następująco:

miasto 3 w u pierwszego rodzica występuje na pozycji 5 a u drugiego na pozycji 2

p_1 = (4 8 2 1 **3** 6 7 5)

p_2 = (2 **3** 4 7 8 6 5 1)

są to punkty startowe, od których rozpoczyna się pobieranie miast, na zmianę od pierwszego i drugiego rodzica oraz wstawianie ich do potomka. Jako pierwsze dołącza się miasto z pierwszego rodzica, kierując się w lewo od punktu miasta 3. Jest to miasto o numerze 1*:*

o = (x x 1 3 x x x x).

Następnie dołącza się miasto od drugiego rodzica (idąc w prawo od punktu startowego):

o = (x x 1 3 4 x x x).

Kolejnymi krokami jest dołączanie miasta 2 (pierwszy rodzic), miasta 7 (drugi rodzic) oraz miasta 8 (pierwszy rodzic):

o = (8 2 1 3 4 7 x x).

Zgodnie z wcześniejszym postępowaniem, należałoby pobrać kolejne miasto od drugiego rodzica. Byłoby to miasto numer 8. Jednak miasto to zostało już wstawione do trasy potomka i dodanie go jest niemożliwe. Jeżeli niemożliwe jest dodanie miasta od jednego z rodziców wtedy wstawia się miasto od drugiego. W tym przykładzie jest to niemożliwe, ponieważ kolejnym miastem u pierwszego rodzica jest 4, które zostało już użyte. Krzyżowanie GSX zakłada, że jeżeli dodanie kolejnego miast jest niemożliwe to pozostałe miasta wstawiane są do potomka w losowym porządku. Ostatecznie chromosom potomny będzie miał następującą postać:

o = (8 2 1 3 4 7 5 6).

2.3.4. Określenie operatorów mutacji

Drugim, obok krzyżowania, kluczowym operatorem dla algorytmu genetycznego jest operator mutacji. W przypadku problemu komiwojażera kodowanego za pomocą reprezentacji ścieżkowej operator mutacji musi być skonstruowany w sposób, który uwzględnia specyfikę takiego kodowania. Operator ten musi zachowywać spójność chromosomów czyli nie tworzyć podtras. W paragrafie tym będą omówione 3 operatory mutacji – zamiana, inwersja i perturbacja.

Zamiana

Operator ten jest najprostszym z możliwych schematów mutacji. Polega na zamianie pozycji dwóch losowo wybranych miast w chromosomie. Przykładowo dla chromosomu o długości 9 wybrano do zamiany miasta na pozycjach 3 i 7.

1 2 **3** 4 5 6 **7** 8 9

po zastosowaniu operatora
1 2 **7** 4 5 6 **3** 8 9.

Inwersja

Operator inwersji jest nieco bardziej złożony od prostej zamiany. Podobnie jak w przypadku operatora mutacji, polegającego na zamianie dwóch miast, operator inwersji wymaga losowego wybrania dwóch miast w chromosomie. Posługując się tym samym przykładem co wcześniej operacja inwersji przebiega następująco:
wybrano losowo miasta na pozycjach 3 i 7
1 2 **3 4 5 6 7** 8 9.
Operator inwersji powoduje przepisanie podtrasy między wybranymi pozycjami w odwrotnej kolejności:
1 2 **7 6 5 4 3** 8 9.

Perturbacja

Podobnie jak w przypadku inwersji, losowo wybierane są dwie pozycje w chromosomie:
1 2 **3** 4 5 6 **7** 8 9.

Zmianie ulega podtrasa między wybranymi pozycjami. Perturbacja ustawia miasta między wybranymi pozycjami w sposób przypadkowy:
1 2 **5 3 7 4 6** 8 9.

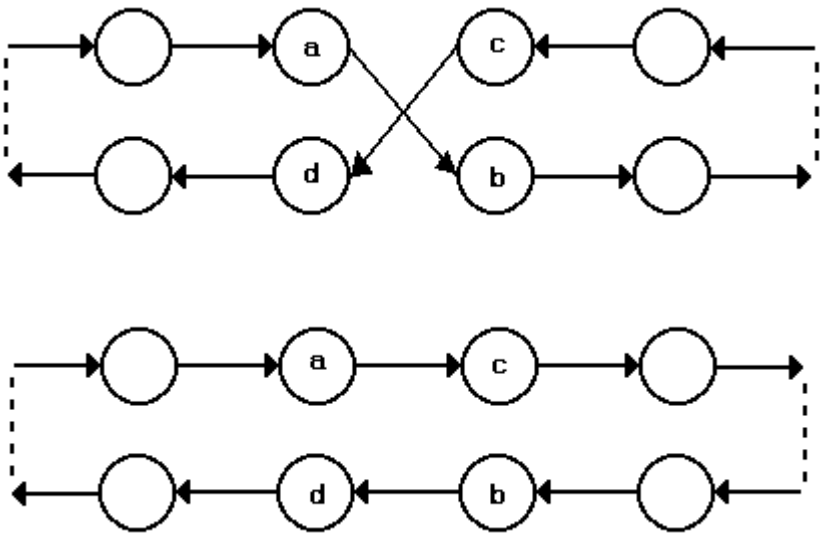
2.3.5. Heurystyki

Zastosowanie wiedzy, jaką posiadamy na temat konkretnego problemu, może znacznie poprawić efektywność działania algorytmu genetycznego. Mimo, że ideowy algorytm genetyczny nie korzysta z żadnych dodatkowych informacji na temat rozpatrywanego problemu, co niewątpliwie stanowi jego ogromną zaletę, to w praktyce wykorzystanie dodatkowej wiedzy zawsze poprawia efektywność. Algorytmy genetyczne mogą skutecznie działać bez dodatkowej wiedzy, ale gdy taka wiedza jest dostępna to warto ją wykorzystać.

Metoda 2-OPT

Lokalna optymalizacja metodą 2-OPT polega na przeszukiwaniu otoczenia rozwiązania

roboczego (czyli dla konkretnego chromosomu) w poszukiwaniu trasy lokalnie optymalnej. Otoczenie takiego punktu roboczego w przypadku metody 2-OPT określa się jako wszystkie trasy jakie mogą powstać poprzez zamianę dwóch krawędzi między dowolnymi dwoma miastami w bieżącym rozwiązaniu.



Rysunek 2.7. Idea metody 2-OPT (opracowanie własne na podstawie [7])

Metoda przeszukuje wszystkie trasy z lokalnego otoczenia bieżącej trasy, szukając tej o najmniejszej sumie odległości. Po znalezieniu trasy lokalnie optymalnej, trasa bieżąca zostaje nią zastąpiona.

Inicjacja populacji metodą najbliższego sąsiada

Najbliższe sąsiedztwo jest często podawane jako samodzielna metoda do rozwiązywania problemu komiwojażera. Niestety daje kiepskie rezultaty. Niemniej jednak może okazać się przydatna jako metoda pomocnicza dla zasadniczego algorytmu. Metoda najbliższego sąsiada polega na budowaniu trasy poprzez wybór kolejnych miast, w sposób który w danej chwili wydaje się najlepszy. Startując z pewnego miasta, jako kolejne miasto dołącza się do trasy to, które znajduje się najbliżej miasta startowego. Analogicznie postępuje się dla kolejnych miast. Inicjacja populacji tą metodą polega na wygenerowaniu trasy za pomocą najbliższego sąsiedztwa a następnie odpowiednie jej przekształcenie, tak aby na jej podstawie utworzonych zostało wiele tras. Przykładowo jeżeli metoda najbliższego sąsiada dała w rezultacie następującą trasę:

3 2 5 1 4.

To na jej podstawie możliwe jest utworzenie pięciu unikalnych chromosomów:

3 2 5 1 4

2 5 1 4 3

5 1 4 3 2

1 4 3 2 5

4 3 2 5 1.

Mimo, że utworzone chromosomy odpowiadają tej samej trasie to sam fakt, że różnią się jako ciągi uporządkowane wystarczy do poprawnego działania algorytmu genetycznego. Należy zaznaczyć, że w sytuacji gdy cała populacja bazowa byłaby wypełniona kopiami tego samego osobnika to nie jest to przeszkodą w poprawnym działaniu algorytmu. Przy dobrze zdefiniowanym operatorze mutacji możliwe jest otrzymanie dobrych rezultatów.

2.3.6. Szczegóły modelu

Po określeniu zasadniczych elementów modelu (sposobu kodowania, metody selekcji oraz operatorów genetycznych) należy zastanowić się nad pewnymi szczegółami o charakterze techniczno-implementacyjnym.

Inicjacja populacji

W poprzednim paragrafie omówiona została inicjacja populacji metodą najbliższego sąsiada. Oprócz tej metody w modelu zastosowana zostanie klasyczna forma inicjacji, polegająca na losowym generowaniu osobników. Losowa inicjacja populacji polega na przeprowadzeniu niezależnych losowań ciągów o długości równej rozmiarowi problemu i ilości równej zakładanej wielkości populacji. Jedynym aspektem wymagającym komentarza jest to, że ciągi muszą być losowane bez zwracania tak aby tworzone były dopuszczalne trasy komiwojażera.

Parametryzacja algorytmu

Efektywność działania algorytmu genetycznego może być uzależniona od wielu czynników. Do takich czynników z pewnością należą wielkość populacji oraz prawdopodobieństwa zaistnienia określonych operacji genetycznych. Zapewnienie parametryzacji nie jest cechą samego modelu lecz jego implementacji. Konieczne jest

zapewnienie mechanizmu, który pozwoli kontrolować operacje genetyczne. Zasadniczymi parametrami modelu będą zatem:

- wielkość populacji
- prawdopodobieństwo krzyżowania
- prawdopodobieństwo mutacji
- prawdopodobieństwo optymalizacji lokalnej

Warunek zatrzymania algorytmu

Jest to składnik typowy dla implementacji modelu. Ideowy algorytm genetyczny zakłada nieskończoną ilość kroków co jest niemożliwe do uzyskania w praktyce. Należy zatem określić kryterium, na podstawie którego algorytm będzie zatrzymywany. W implementacji warunek stopu zostanie zdefiniowany jako wykonanie pewnej z góry ustalonej liczby kroków. Ma to uzasadnienie praktyczne, gdyż pozwoli na badanie zbieżności algorytmu dla różnych kombinacji operatorów genetycznych w miarę zbliżonych warunkach.

3. Implementacja problemu komiwojażera. Eksperyment.

3.1. Budowa aplikacji

Język programowania

Aplikacja została napisana w jeżyku C++. C++ jest obiektowym rozwinięciem klasycznego języka C. C jest znany jako język doskonale nadający się do skomplikowanych obliczeń numerycznych dzięki swojej szybkości. C++ odziedziczył pod tym względem dobre cechy „czystego" C. Ponieważ obecnie dominującym podejściem do tworzenia oprogramowania jest obiektowość, C++ dostarcza narzędzi do tworzenia obiektowego kodu. W tym kontekście, C++ jest językiem uniwersalnym nadającym się do tworzenia różnego rodzaju oprogramowania.

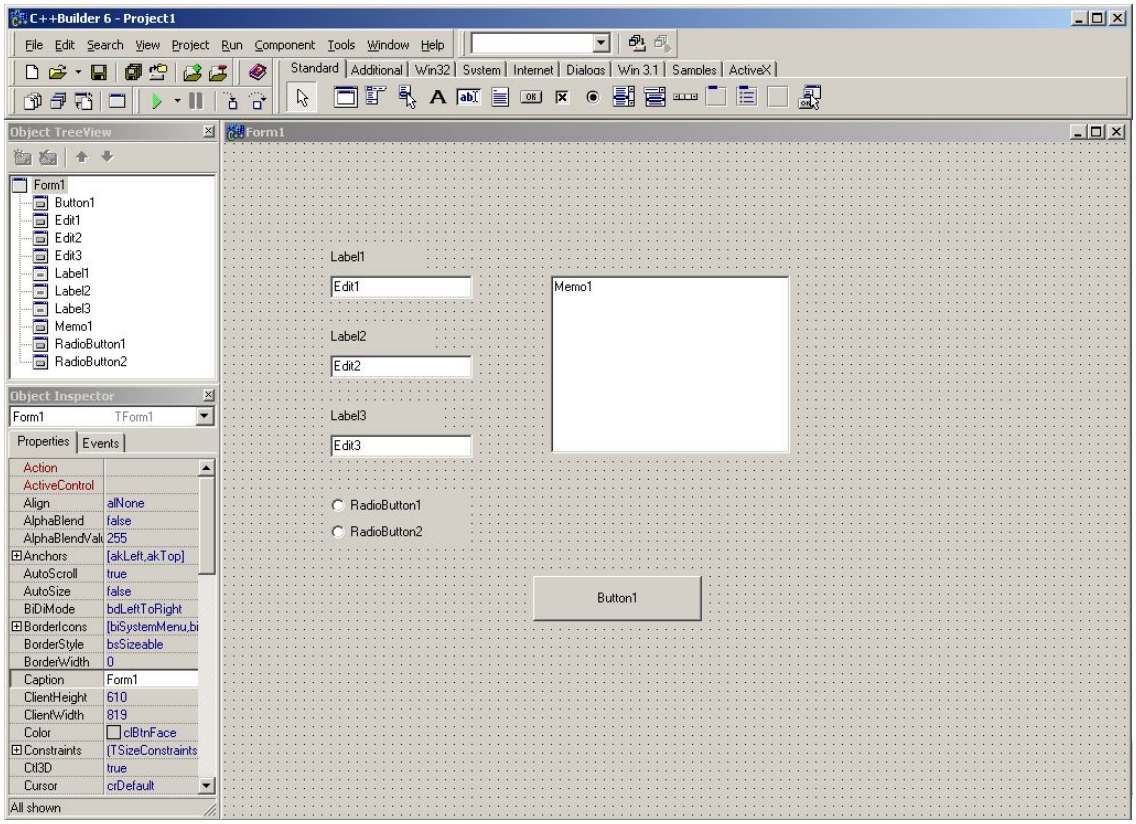
Środowisko implementacyjne

Stworzenie jakiegokolwiek programu ułatwia komunikacja z kompilatorem danego języka poprzez pewien interfejs – środowisko programistyczne. Możliwe jest oczywiście tworzenie kodu programu bez korzystania z konkretnego środowiska, jednak jest to nieefektywne a nawet uciążliwe. Do stworzenia aplikacji, realizującej zaproponowany w poprzednim rozdziale model genetyczny, wykorzystano środowisko C++ Builder 6 firmy Borland w wersji Personal. Środowisko to posiada wiele cech, które usprawniają tworzenie aplikacji. Oprócz podstawowych usprawnień takich jak kolorowanie składni, które posiadają nawet zwykłe edytory tekstu, śledzenia wykonania programu, organizowanie wielu plików z kodem programu w jeden spójny projekt. C++ Builder posiada graficzne narzędzie do tworzenia interfejsu użytkownika. Tworzenie interfejsu przebiega w sposób intuicyjny (ang. *Drag and Drop*). Głównie przez tę cechę środowisko to zostało wybrane do tworzenia aplikacji.

Implementacja

Aplikacja składa się zasadniczo z dwóch elementów: modułu odpowiedzialnego za ogólnie rozumiane obliczenia oraz interfejs użytkownika, poprzez który możliwe jest sterowanie częścią obliczeniową. Część obliczeniowa została podzielona na dwie klasy oraz zbiór metod pomocniczych. Zbiór metod pomocniczych zawiera pomocne

narzędzia m.in. do odczytywania danych z pliku tekstowego, obliczanie odległości między miastami czy inicjowanie macierzy odległości. Klasa Chromosom definiuje

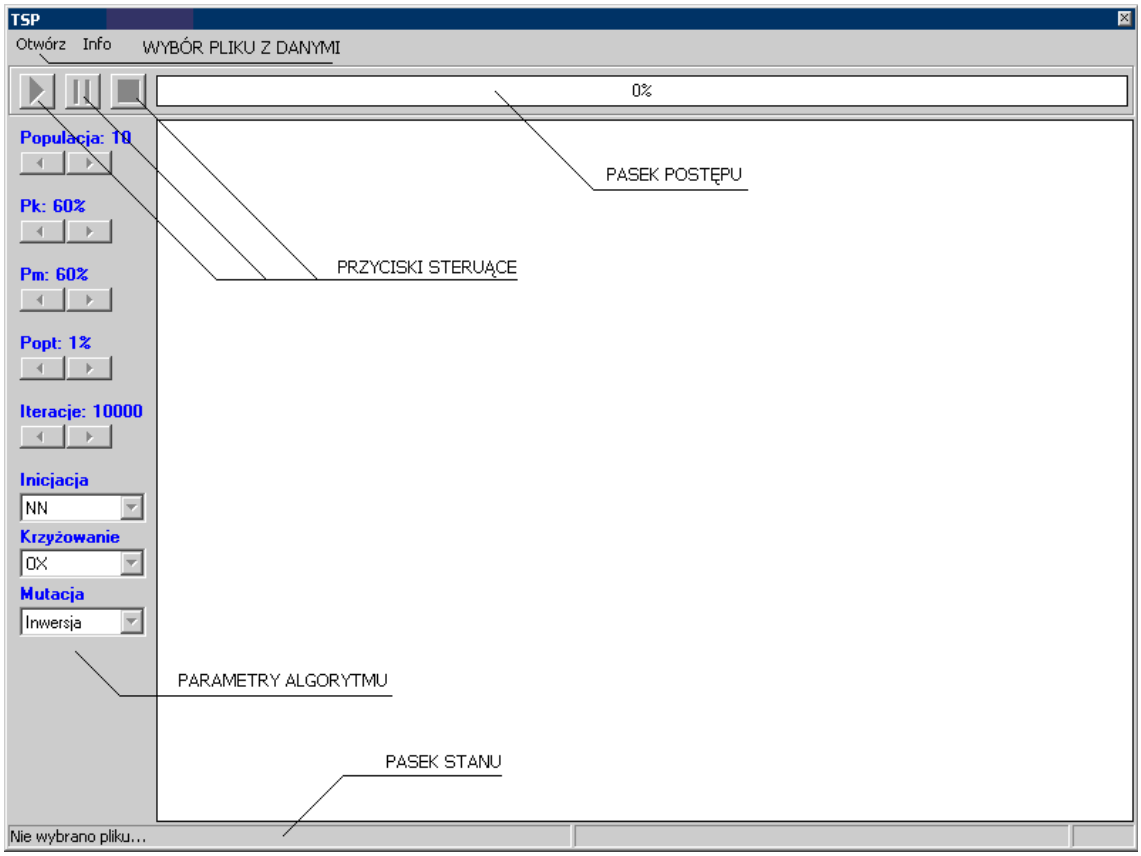


Rysunek 3.1. Budowanie interfejsu użytkownika w środowisku C++ Builder

i opisuje obiekt, realizujący pojedynczy chromosom. Klasa Populacja, która jest zasadniczą i najważniejszą częścią aplikacji, realizuje algorytm genetyczny na populacji chromosomów. Udostępnia metody do przeprowadzania inicjacji populacji, reprodukcji, oceny osobników, sukcesji oraz operacji genetycznych. Trzy wymienione moduły są dołączane do modułu zawierającego interfejs użytkownika. Interfejs komunikuje się z modułem obliczeniowym obsługując akcje użytkownika, takie jak na przykład naciśnięcie przycisku na formatce.

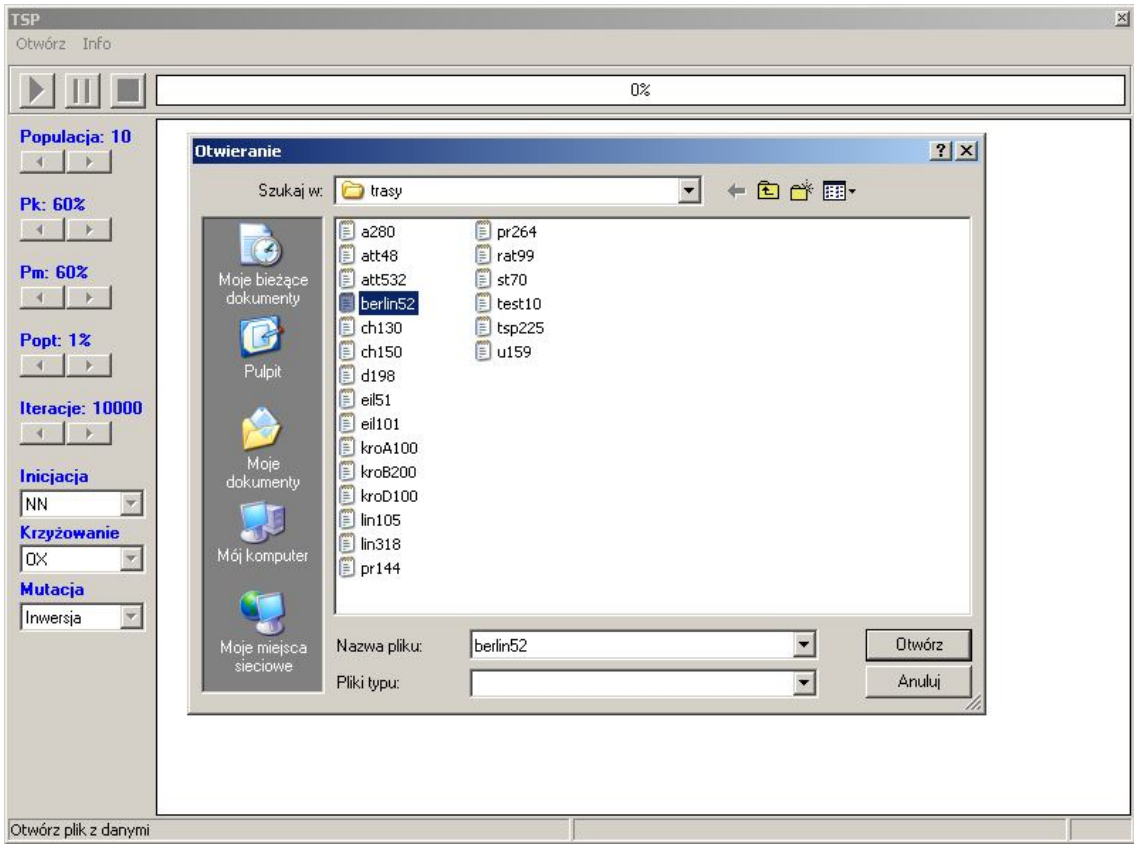
3.2. Opis aplikacji

Aplikacja od strony użytkownika składa się ona z jednego okienka, na którym największy obszar zajmuje pole, na którym pokazywana jest aktualnie najlepsza trasa komiwojażera. Po lewej stronie znajdują się kontrolki do określania parametrów algorytmu. W części górnej znajdują się przyciski sterujące przebiegiem samego

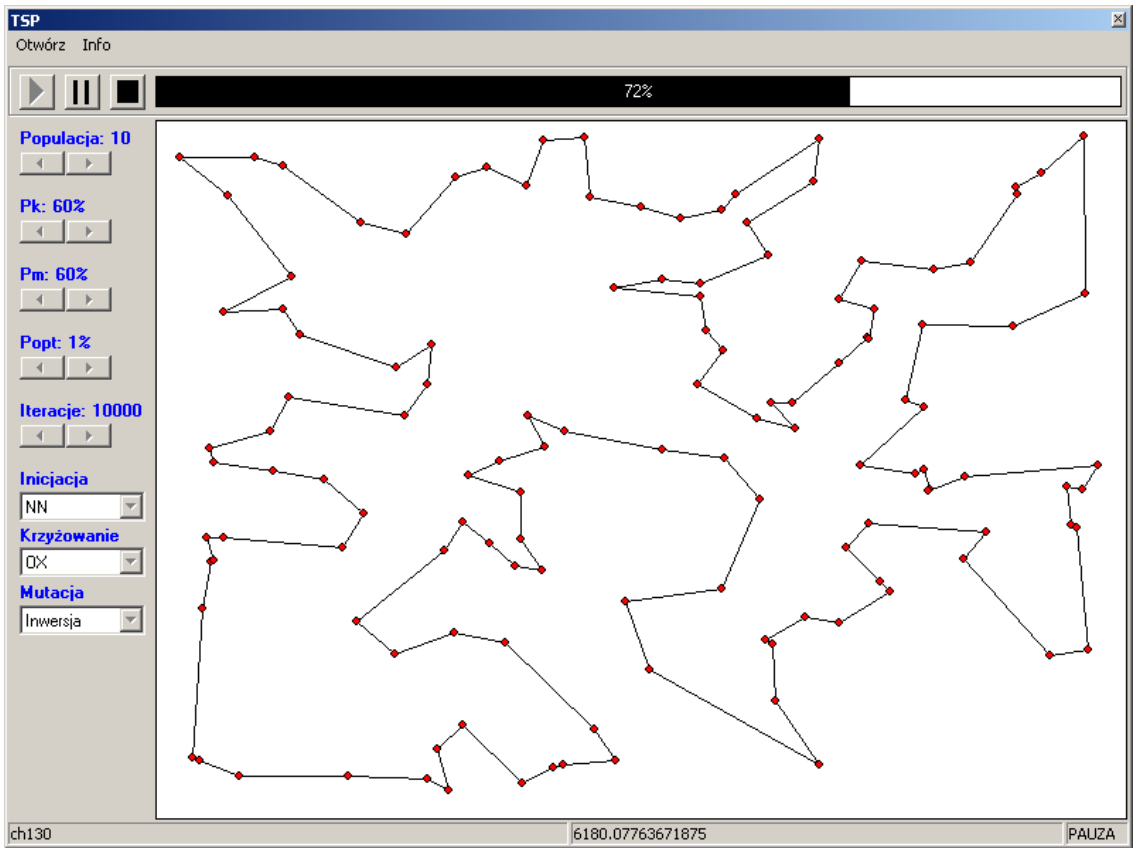


Rysunek 3.2. Okno główne aplikacji

algorytmu. Sterowanie daje możliwość zatrzymania algorytmu (pauza) oraz jego definitywne przerwanie. Stan zaawansowania algorytmu, czyli liczba dotychczas przeprowadzonych iteracji jest uwidoczniona na pasku postępu, po prawej od przycisków sterujących. Wybór pliku z danymi odbywa się poprzez wybranie opcji „Otwórz", co powoduje pojawienie się standardowego okna dialogowego do wyboru pliku. Na samym dole głównego okna znajduje się pasek stanu, na którym wyświetlane są informacje pomocnicze: nazwa aktualnie rozwiązywanego problemu, długość najkrótszej jak dotąd znalezionej trasy oraz informacja o ewentualnym wstrzymaniu pracy programu.



Rysunek 3.3. Wybór pliku z danymi



Rysunek 3.4. Aplikacja podczas pracy

3.3. Eksperyment

Eksperyment będzie polegał na zbadaniu wyników działania algorytmu genetycznego dla problemu komiwojażera, dla różnych konfiguracji parametrów modelu. Punktem odniesienia dla uzyskanych wyników będą rozwiązania problemów komiwojażera, o różnej wielkości, pochodzące z biblioteki TSPLIB [8]. Rozwiązania znajdujące się w TSPLIB są jak dotąd najlepszymi, jakie znaleziono dla poszczególnych problemów. Stąd w swoich rozważaniach rozwiązania z TSPLIB będę uznawał jako optymalne. Eksperyment będzie podzielony na kilka części – etapów. Każdy z nich będzie dotyczył pewnej składowej algorytmu. Badane będą operatory genetyczne – porównanie różnych krzyżowań i mutacji. Badany będzie również wpływ wielkości populacji oraz sposobu inicjacji populacji bazowej na wynik algorytmu. Na końcu zbadany zostanie wpływ zastosowania heurystyki w postaci metody 2-OPT na wynik działania algorytmu oraz próby zastosowania najlepszej znalezionej konfiguracji parametrów do rozwiązania kilku problemów komiwojażera, o różnej wielkości.

Każdy z etapów eksperymentu będzie przeprowadzony a następnie przedstawiony w podobny sposób. Po krótkim wstępie, zostaną podane warunki początkowe, czyli wejściowe wartości parametrów modelu. Wyniki zostaną przedstawione w odpowiednich tabelach i porównane z rozwiązaniami pochodzącymi z TSPLIB. Wnioski wyciągnięte na podstawie wyników zostaną przedstawione na końcu każdego z etapów.

Ponieważ algorytmy genetyczne należą do klasy metod zrandomizowanych, czyli takich gdzie czynnik losowy jest narzędziem przeszukiwania przestrzeni rozwiązań, dokładne określenie efektywności działania algorytmu jest możliwe poprzez wielokrotne powtarzanie eksperymentu. Im więcej prób zostanie przeprowadzonych tym średni wynik tych wszystkich prób będzie bardziej miarodajny. W swoim eksperymencie przyjąłem, że liczba powtórzeń w każdym teście będzie wynosić 10.

Etap I – badanie operatorów krzyżowania

Przeprowadzone badanie ma na celu porównanie czterech zaimplementowanych operatorów krzyżowania pod kątem efektywności. Założeniem badania są stałe wartości wszystkich pozostałych parametrów algorytmu.

Warunki początkowe

- populacja: 10 osobników, inicjacja losowa
- liczba iteracji: 5000
- operator mutacji: inwersja (15%)
- metoda 2-OPT: wyłączona (0%)
- każdy operator krzyżowania badany dla prawdopodobieństwa 0.15, 0.5 oraz 0.85
- rozpatrywane problemy komiwojażera: berlin52, kroA100, ch150

Wyniki

OPT	**BERLIN52**			
7542	**15%**			
	OX	**PMX**	**CX**	**GSX**
1	8421,302734	8405,756836	8028,939941	8072,695313
2	8597,614258	8035,425781	8425,716797	8213,791016
3	8412,542969	8969,119141	8055,35791	7968,657715
4	8466,083008	8517,469727	8130,586914	8499,686523
5	9010,619141	8845,056641	8744,212891	8150,377441
6	8153,306641	8380,887695	8858,430664	8186,197754
7	8647,522461	8330,930664	8380,724609	8710,530273
8	8307,163086	8051,437012	8715,444336	8335,981445
9	7930,169922	8319,844727	7619,152832	8562,689453
10	8188,462891	8292,185547	8880,420898	8457,167969
SREDNIA	8413,478711	8414,811377	8383,898779	8315,77749
%OPT	**89,642%**	**89,628%**	**89,958%**	**90,695%**
MIN	7930,169922	8035,425781	7619,152832	7968,657715
%OPT	**95,105%**	**93,859%**	**98,987%**	**94,646%**

OPT	**BERLIN52**			
7542	**50%**			
	OX	**PMX**	**CX**	**GSX**
1	8266.021484	8222.986328	8522.895508	8093.639648
2	8326.852539	8608.298828	8438.950195	8673.371094
3	8181.825684	8289.47168	8579.594727	8277.749023
4	8237.485352	8859.904297	8044.598633	8102.036133
5	7795.543945	8302.241211	8548.645508	8106.815918
6	8030.489746	8609.896484	8293.956055	8384.503906
7	8516.120117	8260.543945	8310.594727	8166.058105
8	8401.494141	8137.63916	8296.364258	8194.663086
9	7998.109375	8667.854492	8581.022461	8746.619141
10	8321.200195	8615.615234	9002.15332	8219.939453
SREDNIA	8207.514258	8457.445166	8461.877539	8296.539551
%OPT	**91.891%**	**89.176%**	**89.129%**	**90.905%**
MIN	7795.543945	8137.63916	8044.598633	8093.639648
%OPT	**96.748%**	**92.680%**	**93.752%**	**93.184%**

OPT	**BERLIN52**			
7542	**85%**			
	OX	**PMX**	**CX**	**GSX**
1	8264.581055	8511.298828	8212.711914	8399.600586
2	8391.951172	8937.59082	8346.77832	8370.172852
3	8434.389648	8033.773438	8139.742188	8689.041016
4	8027.622559	7990.765625	8692.206055	8324.390625
5	8460.65625	8807.71582	8589.289063	8417.587891
6	8306.244141	8558.214844	8562.842773	8757.911133
7	8405.775391	8766.731445	8270.849609	8254.09082
8	8554.140625	8311.348633	8404.073242	8348.379883
9	8168.05957	8402.09375	8509.180664	8843.583984
10	8072.619141	8757.708984	8559.447266	8537.303711
SREDNIA	8308.603955	8507.724219	8428.712109	8494.20625
%OPT	**90.773%**	**88.649%**	**89.480%**	**88.790%**
MIN	8027.622559	7990.765625	8139.742188	8254.09082
%OPT	**93.951%**	**94.384%**	**92.656%**	**91.373%**

OPT	**KROA100**			
21282	**15%**			
	OX	**PMX**	**CX**	**GSX**
1	30123.79883	33918.45313	31729.43945	33517.16797
2	32249.59961	33796.27734	31789.86719	32775.95313
3	28078.07227	32651.20117	34194.75	30936.0625
4	32456.6543	33201.53125	31301.98047	32225.78906
5	31576.80664	33525.78906	34824.4375	29475.58398
6	31586.27148	29341.63867	30482.33203	30702.52148
7	30653.98438	29326.82031	28092.42578	32096.50586
8	30283.20313	30129.25391	32988.37109	29066.64258
9	33696.125	32296.6582	33266.24219	30050.5918
10	28307.26758	31966.14063	30204.38086	31968.0918
SREDNIA	30901.17832	32015.37637	31887.42266	31281.49102
%OPT	**68.871%**	**66.474%**	**66.741%**	**68.034%**
MIN	28078.07227	29326.82031	28092.42578	29066.64258
%OPT	**75.796%**	**72.568%**	**75.757%**	**73.218%**

OPT	**KROA100**			
21282	**50%**			
	OX	**PMX**	**CX**	**GSX**
1	32120.47461	33235.67578	32047.64063	33385.75391
2	29998.28711	32704.80273	30464.55859	32063.36719
3	33528.54297	29114.71484	32281.80469	32560.27539
4	31324.51953	32116.76172	32539.87305	33001.17969
5	29089.42773	29037.94336	32363.95703	32052.29688
6	30655.84375	31414.77148	31704.54492	32729.19727
7	31346.21875	31629.27148	30953.41211	30971.64844
8	31813.11328	31679.23047	31898.25	34609.85156
9	33378.62891	32978.96484	29990.13477	31474.45117
10	29343.92773	28987.5918	31836.08203	32526.96289
SREDNIA	31259.89844	31289.97285	31608.02578	32537.49844
%OPT	**68.081%**	**68.015%**	**67.331%**	**65.408%**
MIN	29089.42773	28987.5918	29990.13477	30971.64844
%OPT	**73.161%**	**73.418%**	**70.963%**	**68.714%**

OPT	**KROA100**			
21282	**85%**			
	OX	**PMX**	**CX**	**GSX**
1	36705.57422	28456.77539	30328.57227	32587.27344
2	31329.28906	29551.48242	29689.83203	32179.85938
3	33160.21875	31608.01172	30398.58594	34978.30469
4	32447.67383	32601.82422	32664.57031	33442.55078
5	31344.01758	34458.45703	30526.36719	30170.77734
6	30676.91992	33340.89844	31308.79492	32000.5957
7	35605.15625	29984.35156	31766.57813	35591.75391
8	31460.52344	33148.23828	32516.38086	33075.57031
9	30781.69727	29189.13086	34062.96875	33256.07422
10	33689.23828	30060.62109	30348.60156	35080.45313
SREDNIA	32720.03086	31239.9791	31361.1252	33236.32129
%OPT	**65.043%**	**68.124%**	**67.861%**	**64.032%**
MIN	30676.91992	28456.77539	29689.83203	30170.77734
%OPT	**69.375%**	**74.787%**	**71.681%**	**70.538%**

OPT	**CH150**			
6528	**15%**			
	OX	**PMX**	**CX**	**GSX**
1	13203.5459	13398.95996	12986.91309	14051.10059
2	12614.03613	13803.07031	12432.09668	13394.49512
3	13113.95313	12805.2207	13706.76367	13494.70898
4	12116.96484	13336.40332	13677.33887	13078.70605
5	12853.55273	12827.61426	13135.80762	13015.86035
6	13629.65527	12296.31543	12052.10547	12680.03125
7	12848.8291	13037.07422	13203.5127	13434.66992
8	12633.46582	13478.85742	14139.43164	13502.67773
9	13094.97949	13463.2998	13128.36914	13053.8418
10	12504.42188	13216.66895	13120.11426	13044.91016
SREDNIA	12861.34043	13166.34844	13158.24531	13275.1002
%OPT	**50.757%**	**49.581%**	**49.611%**	**49.175%**
MIN	12116.96484	12296.31543	12052.10547	12680.03125
%OPT	**53.875%**	**53.089%**	**54.165%**	**51.483%**

OPT	**CH150**			
6528	**50%**			
	OX	**PMX**	**CX**	**GSX**
1	12553.40332	13403.79102	13720.03516	14455.69824
2	14572.86914	12816.98535	13039.73438	13023.90625
3	12692.45117	12944.81934	13208.09863	14308.90039
4	13513.04199	13276.31445	13185.81543	13276.92773
5	12526.91113	12880.01855	12708.57031	14253.99219
6	12646.77441	12597.29297	13634.80664	14069.86035
7	13362.68359	12189.51465	13374.55762	13935.09766
8	12579.14941	12745.1377	13038.88086	14004.01855
9	14866.15625	13152.5	12629.3457	13750.92871
10	13451.26953	13619.95117	13345.50195	13679.35449
SREDNIA	13276.471	12962.63252	13188.53467	13875.86846
%OPT	**49.170%**	**50.360%**	**49.498%**	**47.046%**
MIN	12526.91113	12189.51465	12629.3457	13023.90625
%OPT	**52.112%**	**53.554%**	**51.689%**	**50.123%**

OPT	**CH150**			
6528	**85%**			
	OX	**PMX**	**CX**	**GSX**
1	14830.41113	11774.27539	12623.7959	16253.96582
2	13920.28418	13220.14648	13412.31152	14114.01758
3	15260.92676	12077.69531	12861.44141	15046.18457
4	15496.48828	12780.10059	13299.45117	15847.50586
5	15157.52051	12867.89551	13960.04492	14435.23926
6	14228.48047	12645.32324	13146.89453	15179.33789
7	15033.20703	13272.01563	13133.15234	15315.4707
8	15199.05273	13511.12402	12897.50586	15505.36133
9	14733.19531	12569.5625	13242.93262	14959.82129
10	14179.30078	13622.67773	13184.04004	16146.71191
SREDNIA	14803.88672	12834.08164	13176.15703	15280.36162
%OPT	**44.097%**	**50.865%**	**49.544%**	**42.722%**
MIN	13920.28418	11774.27539	12623.7959	14114.01758
%OPT	**46.896%**	**55.443%**	**51.712%**	**46.252%**

Wnioski

Przeprowadzone badanie operatorów krzyżowania nie daje jednoznacznej odpowiedzi na pytanie, który z operatorów jest zdecydowanie najlepszy, gdyż wszystkie uzyskały dosyć podobne wyniki. Minimalnie lepszy od pozostałych operatorów wydaje się być OX, który w 6 na 9 testach uzyskał najlepszy rezultat (z reguły nieznacznie lepszy od pozostałych operatorów). Poziom prawdopodobieństwa krzyżowania wydaje się nie mieć znaczenia, gdyż wyniki uzyskiwane dla poszczególnych jego wartości (15%, 50% i 85%) niewiele się różnią.

Etap II – badanie operatorów mutacji

Przeprowadzone badanie ma na celu porównanie trzech zaimplementowanych operatorów mutacji pod kątem efektywności. Założeniem badania są stałe wartości wszystkich pozostałych parametrów algorytmu.

Warunki początkowe

- populacja: 10 osobników, inicjacja losowa
- liczba iteracji: 5000
- operator krzyżowania: wyłączony (0%)
- metoda 2-OPT: wyłączona (0%)
- każdy operator mutacji badany dla prawdopodobieństwa 0.15, 0.5 oraz 0.85
- rozpatrywane problemy komiwojażera: berlin52, kroA100, ch150

Wyniki

OPT	**BERLIN52**		
7542	**15%**		
	inwersja	**zamiana**	**perturbacja**
1	8524.94043	9969.864258	15896.02246
2	8623.87207	11500.07227	14193.53906
3	8893.678711	10901.62305	16859.07227
4	8853.433594	11054.89551	16547.43164
5	7996.977539	9789.783203	16369.13867
6	8415.545898	10691.80664	15800.7959
7	8417.769531	10840.29102	14610.13477
8	8557.764648	10354.91113	14626.86816
9	8167.194336	11387.35059	15740.52832
10	8438.425781	10103.01563	16042.91309
SREDNIA	8488.960254	10659.36133	15668.64443
%OPT	**88.845%**	**70.755%**	**48.134%**
MIN	7996.977539	9789.783203	14193.53906
%OPT	**94.311%**	**77.039%**	**53.137%**

OPT	**BERLIN52**		
7542	**50%**		
	inwersja	**zamiana**	**perturbacja**
1	8501.895508	11766.48633	13361.80859
2	8384.198242	11602.19043	15037.47656
3	8609.108398	9926.581055	16595.61328
4	8180.260254	11797.6748	17186.55664
5	8133.975098	11219.36914	14831.20508
6	8529.673828	9927.073242	13731.03906
7	8337.402344	9261.176758	14984.10449
8	8641.34375	10527.67285	14519.49121
9	7842.951172	10717.3877	15556.25586
10	8203.450195	10757.23926	15969.30957
SREDNIA	8336.425879	10750.28516	15177.28604
%OPT	**90.470%**	**70.156%**	**49.693%**
MIN	7842.951172	9261.176758	13361.80859
%OPT	**96.163%**	**81.437%**	**56.444%**

OPT	**BERLIN52**		
7542	**85%**		
	inwersja	**zamiana**	**perturbacja**
1	8472.547852	9986.619141	16839.63477
2	8582.816406	9951.335938	17227.9043
3	8233.552734	11043.4209	13629.26465
4	7951.189453	10544.18457	15489.43066
5	8098.960938	10542.96094	14316.66211
6	8737.836914	10187.56934	13036.5752
7	8377.026367	10929.84766	14671.83691
8	7935.169434	10760.75977	13500.1123
9	8225.598633	10659.25977	15208.4375
10	8387.363281	11252.60352	14143.87793
SREDNIA	8300.206201	10585.85615	14806.37363
%OPT	**90.865%**	**71.246%**	**50.938%**
MIN	7935.169434	9951.335938	13036.5752
%OPT	**95.045%**	**75.789%**	**57.853%**

OPT	**KROA100**		
21282	**15%**		
	inwersja	**zamiana**	**perturbacja**
1	32802.33984	57202.55078	95705.99219
2	31438.13086	55799.27734	85626.60156
3	33614.98047	56810.86719	80978.61719
4	33356.59766	56931.16016	81700.42969
5	31078.90039	53841.32031	79570.53125
6	32194.45703	54142.88281	97601.35938
7	32174.83594	54339.19141	80532
8	33388.67188	57804.90234	87166.26563
9	31708.91016	58149.33203	84487.73438
10	31957.57227	60850.35938	85616.53906
SREDNIA	32371.53965	56587.18438	85898.60703
%OPT	**65.743%**	**37.609%**	**24.776%**
MIN	31078.90039	53841.32031	79570.53125
%OPT	**68.477%**	**39.527%**	**26.746%**

OPT	**KROA100**		
21282	**50%**		
	inwersja	**zamiana**	**perturbacja**
1	30775.37305	61231.42969	82210.5
2	29288.07422	51078.03125	88637.92188
3	29472.23828	53542.10156	82153.52344
4	29306.14844	49720.48438	82831.4375
5	28878.70117	58908.03516	88763.49219
6	27530.98047	49901.96484	69438.5
7	30785.57813	54872.375	81704.65625
8	31411.19336	60776.13281	77375.48438
9	28746.14648	55583.17578	79955.22656
10	29403.91992	55734.90625	82932.5
SREDNIA	29559.83535	55134.86367	81600.32422
%OPT	**71.996%**	**38.600%**	**26.081%**
MIN	27530.98047	49720.48438	69438.5
%OPT	**77.302%**	**42.803%**	**30.649%**

OPT	**KROA100**		
21282	**85%**		
	inwersja	**zamiana**	**perturbacja**
1	28255.19531	54473.35156	85022.9375
2	28953.02148	55288.39063	74815.3125
3	29746.26758	55406.46094	85219.54688
4	29658.39844	54388.28516	70115.16406
5	29356.41797	48130.13672	85172.17188
6	29109.17578	55273.89844	83835.66406
7	27673.42773	55346.1875	85613.89844
8	28475.81445	51101.82031	88543.58594
9	29020.08203	51695.89063	77344.14063
10	27833.46875	49125.85938	79733.32031
SREDNIA	28808.12695	53023.02813	81541.57422
%OPT	**73.875%**	**40.137%**	**26.100%**
MIN	27673.42773	48130.13672	70115.16406
%OPT	**76.904%**	**44.218%**	**30.353%**

OPT	**CH150**		
6528	**15%**		
	inwersja	**zamiana**	**perturbacja**
1	12804.52148	20709.97656	32018.07227
2	12728.3623	20369.68945	32935.17969
3	14335.87207	20050.68164	32398.14453
4	13335.30762	20678.12109	30040.53711
5	13046.92773	21440.36719	33765.91797
6	12677.85547	20059.03125	31436.09766
7	14037.83691	19801.49219	31890.15625
8	13848.4082	21632.05078	32151.50977
9	12203.24023	20970.05469	33749.19141
10	13506.41016	19308.70117	31866.31836
SREDNIA	13252.47422	20502.0166	32225.1125
%OPT	**49.259%**	**31.841%**	**20.257%**
MIN	12203.24023	19308.70117	30040.53711
%OPT	**53.494%**	**33.809%**	**21.731%**

OPT	**CH150**		
6528	**50%**		
	inwersja	**zamiana**	**perturbacja**
1	12064.26172	20221.28125	28196.69727
2	12260.65918	20680.21875	31838.55859
3	12420.56836	19044.66016	31579.82227
4	11775.30371	19987.41016	29316.41602
5	11185.92285	19024.60742	30928.77539
6	12726.53027	21511.15039	30960.91797
7	11609.76953	18018.76367	30699.20117
8	12017.78809	18633.55664	30558.62109
9	12070.875	17534.89648	32286.36914
10	12309.94727	21566.92578	30733.37109
SREDNIA	12044.1626	19622.34707	30709.875
%OPT	**54.201%**	**33.268%**	**21.257%**
MIN	11185.92285	17534.89648	28196.69727
%OPT	**58.359%**	**37.229%**	**23.152%**

OPT	**CH150**		
6528	**85%**		
	inwersja	**zamiana**	**perturbacja**
1	11284.31445	19759.60156	30481.93359
2	12267.74023	19143.07617	33475.64453
3	11943.97754	18333.24023	33749.28516
4	11007.2041	18845.55469	32474.2793
5	10907.83496	19113.60156	31933.23633
6	11686.55566	18940.05273	30319.64453
7	11833.98438	19448.93164	33003.87891
8	11267.03809	18309.39453	30483.75586
9	11250.21094	18288.3457	31953.41602
10	12276.51074	20613.63086	31052.93359
SREDNIA	11572.53711	19079.54297	31892.80078
%OPT	**56.409%**	**34.215%**	**20.469%**
MIN	10907.83496	18288.3457	30319.64453
%OPT	**59.847%**	**35.695%**	**21.531%**

Wnioski

W przeciwieństwie do poprzedniego badania, w przypadku operatorów mutacji testy jednoznacznie pokazują wyższość jednego z operatorów nad pozostałymi. W każdym z przeprowadzonych testów najlepsza okazała się inwersja. Różnice między wynikami uzyskanymi przez inwersje a pozostałymi są znaczące. Okazało się, że lepszy średni wynik jest uzyskiwany dla wyższych wartości prawdopodobieństwa. Jest to całkowicie zrozumiałe i zgodne z oczekiwaniem, gdyż wyższe prawdopodobieństwo oznacza więcej wykonanych mutacji i co za tym idzie więcej szans do wygenerowania dobrego wyniku.

Etap III – badanie wpływu wielkości populacji

Przeprowadzone badanie ma na celu wyciągniecie wniosków dotyczących wielkości populacji. Założeniem badania są stałe wartości wszystkich pozostałych parametrów algorytmu.

Warunki początkowe

- populacja: 10, 20 i 50 osobników, inicjacja losowa
- liczba iteracji: 5000
- operator krzyżowania: OX (85%)
- operator mutacji: inwersja (15%)
- metoda 2-OPT: wyłączona (0%)
- badanie oparte o problemy: berlin52, kroA100, ch150

Wyniki

OPT	**BERLIN52**		
7542	**10**	**20**	**50**
1	8692.555664	8534.876953	9017.658203
2	8363.275391	8250.167969	8325.701172
3	7959.651855	8905.683594	8474.494141
4	8158.974609	8698.945313	8841.93457
5	8285.263672	8220.665039	9796.099609
6	8421.120117	8570.354492	8676.494141
7	8399.720703	8700.24707	9181.893555
8	8177.950195	8559.071289	9184.985352
9	8112.903809	8762.924805	8722.578125
10	7925.359863	8718.132813	8222.470703
SREDNIA	8249.677588	8592.106934	8844.430957
%OPT	**91.422%**	**87.778%**	**85.274%**
MIN	7925.359863	8220.665039	8222.470703
%OPT	**95.163%**	**91.744%**	**91.724%**

OPT	**KROA100**		
21282	**10**	**20**	**50**
1	29627.82813	40142.17578	46460.55078
2	33853.70703	44102.02734	49241.6875
3	33315.49609	39976.79688	45208.66797
4	32242.21484	38519.25	43397.99219
5	29449.39063	41430.64453	47325.86719
6	32512.45703	39075.94922	46899.00391
7	32150.25977	39163.49219	57458.16406
8	33530.625	40886.43359	48550.8125
9	30656.78711	39873.53125	45105.5625
10	33203.71875	41751.38672	52824.33594
SREDNIA	32054.24844	40492.16875	48247.26445
%OPT	**66.394%**	**52.558%**	**44.110%**
MIN	29449.39063	38519.25	43397.99219
%OPT	**72.266%**	**55.250%**	**49.039%**

OPT	**CH150**		
6528	**10**	**20**	**50**
1	13336.41992	18449.04492	21934.54492
2	14374.14844	18327.5625	22195.54492
3	13973.72852	18177.60547	21772.28516
4	15338.42969	16949.48242	19872.61914
5	14127.24609	18262.55078	22870.4375
6	13402.78418	17708.27344	21495.07227
7	14840.50586	17451.73047	21014.40039
8	15168.65039	17017.34961	21234.11133
9	14623.09863	17418.94922	21545.83008
10	14235.62207	17606.53711	23195.55273
SREDNIA	14342.06338	17736.90859	21713.03984
%OPT	**45.516%**	**36.805%**	**30.065%**
MIN	13336.41992	16949.48242	19872.61914
%OPT	**48.949%**	**38.514%**	**32.849%**

Wnioski

Dla wszystkich trzech problemów komiwojażera przy stałej liczbie iteracji najlepsze wyniki uzyskano na populacji o wielkości 10, czyli najmniejszej. Prawdopodobnie aby większe populacja uzyskiwała wyniki zbliżone do małej populacji, należałoby odpowiednio zwiększyć liczbę iteracji. Jednak zwiększanie populacji i liczby iteracji nie jest pożądane z racji rosnącego, dość gwałtownie, czasu wykonania algorytmu.

Etap IV – porównanie metod inicjacji populacji

Przeprowadzone badanie ma na celu porównanie losowej inicjacji populacji z inicjacją metodą najbliższego sąsiada. Założeniem badania są stałe wartości wszystkich pozostałych parametrów algorytmu.

Warunki początkowe

- populacja: 10 osobników, inicjacja losowa oraz NN
- liczba iteracji: 5000
- operator krzyżowania: OX (85%)
- operator mutacji: inwersja (15%)
- metoda 2-OPT: wyłączona (0%)
- badanie oparte o problemy: berlin52, kroA100, ch150

Wyniki

OPT	**BERLIN52**	
7542	**losowa**	**NN**
1	8097.56543	7941.312988
2	8053.024414	8014.731445
3	8677.129883	7953.397949
4	8953.837891	7944.5
5	8313.130859	8076.760254
6	8096.046875	7990.150391
7	8194.583984	8092.084473
8	8443.349609	7925.919434
9	8008.125488	7846.788086
10	8187.347168	7985.779297
SREDNIA	8302.41416	7977.142432
%OPT	**90.841%**	**94.545%**
MIN	**8008.125488**	**7846.788086**
%OPT	94.179%	96.116%

OPT	**KROA100**	
21282	**losowa**	**NN**
1	31345.70703	23662.0957
2	29309.78125	23745.95508
3	32270.88672	24250.60156
4	35777.58594	24051.63477
5	31988.01563	23470.68359
6	34370.00391	23109.92383
7	30210.53516	24796.99805
8	33475.47656	24063.15039
9	31153.10547	23192.17188
10	34958.36719	23208.96289
SREDNIA	32485.94648	23755.21777
%OPT	**65.511%**	**89.589%**
MIN	29309.78125	23109.92383
%OPT	**72.611%**	**92.090%**

OPT	**CH150**	
6528	**losowa**	**NN**
1	14859.66113	7908.858398
2	15112.2959	7573.254883
3	14079.74121	7371.702637
4	14346.74805	7924.23584
5	16124.16113	7580.701172
6	14500.38184	7820.689941
7	13807.33496	7693.938965
8	15170.76953	7889.411621
9	14319.10254	7804.557129
10	14204.14941	7732.086914
SREDNIA	14652.43457	7729.94375
%OPT	**44.552%**	**84.451%**
MIN	13807.33496	7371.702637
%OPT	**47.279%**	**88.555%**

Wnioski

Zastosowanie metody NN przy inicjacji populacji daje o wiele lepsze rezultaty niż inicjacja losowo. Jest to całkowicie zrozumiałe i zgodne z oczekiwaniami, gdyż dzięki NN algorytm już na wstępie otrzymuje bardzo dobre rozwiązania, które następnie dodatkowo poprawia. Przy inicjacji losowej algorytm musi najpierw wykorzystać dużą część z dostępnej puli iteracji aby dojść do rozwiązań, które przy NN są dostępne na wejściu. Inicjacja losowa dała w miarę dobry wynik dla problemu berlin52. Jest to problem o stosunkowo małym rozmiarze. Przy kolejnych problemach wyniki inicjacji losowej są coraz słabsze. Wynika z tego, że przy większych problemach aby uzyskać dobre rozwiązanie należy korzystać z usprawnień heurystycznych.

Etap V – badanie wpływu metody 2-OPT

Przeprowadzone badanie ma na celu określenie wpływu zastosowania metody 2-OPT na efektywność algorytmu. Założeniem badania są stałe wartości wszystkich pozostałych parametrów algorytmu.

Warunki początkowe

- populacja: 10 osobników, inicjacja NN
- liczba iteracji: 5000
- operator krzyżowania: OX (85%)
- operator mutacji: inwersja (15%)
- metoda 2-OPT: 0%, 1%, 2%, 5%
- badanie oparte o problemy: berlin52, kroA100, ch150

Wyniki

OPT	**BERLIN52**			
7542	**0%**	**1%**	**2%**	**5%**
1	8033.017578	7544.365234	7544.365234	7544.365234
2	7808.192871	7544.365234	7544.365234	7544.365234
3	8171.220703	7544.365234	7544.365234	7544.365234
4	7944.500488	7544.365234	7544.365234	7544.365234
5	7892.868164	7544.365234	7544.365234	7544.365234
6	8226.049805	7544.365234	7544.365234	7544.365234
7	7992.517578	7544.365234	7544.365234	7544.365234
8	7991.461914	7544.365234	7544.365234	7544.365234
9	8038.701172	7544.365234	7544.365234	7544.365234
10	8056.01123	7544.365234	7544.365234	7544.365234
SREDNIA	8015.45415	7544.365234	7544.365234	7544.365234
%OPT	**94.093%**	**99.969%**	**99.969%**	**99.969%**
MIN	7808.192871	7544.365234	7544.365234	7544.365234
%OPT	**96.591%**	**99.969%**	**99.969%**	**99.969%**

OPT	**KROA100**			
21282	**0%**	**1%**	**2%**	**5%**
1	23656.33398	21285.44727	21285.43945	21285.4375
2	22992.00586	21285.43555	21285.43945	21285.43555
3	23163.76172	21285.44141	21285.43945	21285.43945
4	23381.73633	21349.43945	21294.39453	21285.44531
5	23299.19531	21294.39453	21285.44336	21285.43555
6	24607.10938	21349.44141	21294.39063	21307.41797
7	24256.62109	21285.43555	21285.44141	21294.39258
8	23644.57227	21377.50586	21294.39258	21285.43555
9	23778.76172	21285.43945	21285.43555	21285.44336
10	23305.68945	21307.42383	21285.43555	21285.4375
SREDNIA	23608.57871	21310.54043	21288.1252	21288.53203
%OPT	**90.145%**	**99.866%**	**99.971%**	**99.969%**
MIN	22992.00586	21285.43555	21285.43555	21285.43555
%OPT	**92.563%**	**99.984%**	**99.984%**	**99.984%**

OPT	**CH150**			
6528	**0%**	**1%**	**2%**	**5%**
1	7843.949707	6595.372559	6553.79541	6566.414063
2	7802.419434	6578.366211	6589.644043	6564.825684
3	7300.830566	6556.075195	6576.263672	6563.161621
4	7646.062012	6580.344238	6576.263184	6530.900391
5	7664.833008	6553.101563	6553.101563	6584.061523
6	7455.23291	6600.343262	6579.904785	6530.899902
7	7666.598145	6614.739258	6564.591309	6552.295898
8	7930.032715	6629.60498	6595.514648	6530.900879
9	7668.691406	6562.016113	6591.376465	6576.263184
10	7819.797852	6595.961426	6530.900879	6552.29541
SREDNIA	7679.844775	6586.59248	6571.135596	6555.201855
%OPT	**85.002%**	**99.110%**	**99.344%**	**99.585%**
MIN	7300.830566	6553.101563	6530.900879	6530.899902
%OPT	**89.414%**	**99.617%**	**99.956%**	**99.956%**

Wnioski

Zastosowanie metody 2-OPT spowodowało znaczną poprawę efektywności. Przy wyłączonej metodzie 2-OPT (0%) najlepsze ze znalezionych rozwiązań dla wszystkich trzech problemów było odległe od rozwiązania optymalnego o 5%-10%. Dla testów gdzie metoda 2-OPT zastała zastosowana (w przypadkach 1%, 2% oraz 5%) jakość rozwiązań była na poziomie około 99%. Oznacza to, że zastosowanie metody 2-OPT podnosi efektywność algorytmu. Dodatkowo da się zauważyć, że większe prawdopodobieństwo 2-OPT daje minimalnie lepszy wynik niż w przypadku mniejszego. Jednak dalsze zwiększanie prawdopodobieństwa metody 2-OPT nie wydaje się uzasadnione, gdyż poprawa wyniku jest minimalna a zwiększa się wyraźnie czas wykonania.

Zakończenie

Celem tej pracy było przedstawienie możliwości algorytmów genetycznych w rozwiązywaniu niebanalnego zagadnienia, jakim jest problem komiwojażera. Zasadniczymi elementami umożliwiającymi realizację tego celu była budowa modelu algorytmu a następnie jego komputerowa implementacja. Kolejnym etapem było przeprowadzenie eksperymentu, czyli serii symulacji dla różnych warunków początkowych algorytmu, które miały na celu zbadanie własności modelu w zależności od wartości różnych parametrów.

Przeprowadzone badania pokazały sporą wrażliwość algorytmu na zmianę jego parametrów, czego przykładem może być na przykład to, że operator OX uzyskiwał średnio lepsze wyniki od pozostałych operatorów. Wynika z tego, że możliwe jest skonstruowanie operatora specyficznego dla zadania, który będzie uzyskiwał dobre wyniki dla tego zadania.

Drugim, drugim i chyba najważniejszym, wnioskiem wynikającym z przeprowadzonego badania jest potrzeba stosowania usprawnień heurystycznych w modelu celem osiągnięcia wyższej efektywności. Zarówno zastosowanie metody najbliższego sąsiada jako sposobu inicjacji populacji, jak i metody 2-OPT poprawiło znacznie efektywność. Średnie wyniki uzyskiwane przy zastosowaniu metody 2-OPT, dla wybranych problemów komiwojażera, za każdym razem były oddalone od rozwiązań optymalnych o około 1%.

Każde usprawnienie heurystyczne wiąże się z wykorzystaniem dostępnej wiedzy o rozpatrywanym problemie. W przypadku algorytmów genetycznych z wykorzystaniem usprawnień heurystycznych traci się, uznawaną za jedną z najważniejszych zalet metod genetycznych, czyli ogólność. Z założenia algorytmy genetyczne miały być uniwersalna metodą w zadaniach o nieznanej charakterystyce. Jednak, z drugiej strony, w problemach, o których naturze coś wiadomo, warto byłoby spróbować skorzystać z dostępnej wiedzy, poprzez zastosowanie heurystyk.

Spis Literatury

[1] Jarosław Arabas, Wykłady z algorytmów ewolucyjnych, WNT, Warszawa 2004

[2] David E. Goldberg, Algorytmy genetyczne i ich zastosowania, WNT Warszawa 2003

[3] Zbigniew Michalewicz, Algorytmy Genetyczne + Struktury Danych = Programy Ewolucyjne, WNT, Warszawa 2003

[4] Danuta Rutkowska, Maciej Piliński, Leszek Rutkowski, Sieci neuronowe, algorytmy genetyczne i systemy rozmyte, PWN, Warszawa 1997

[5] Tomasz Dominik Gwiazda, Algorytmy genetyczne : T.1, Operator krzyżowania dla problem ów numerycznych, PWN, Warszawa 2007

[6] David L. Applegate, Robert E. Bixby, Vasek Chvátal, William J. Cook, The Traveling Salesman Problem: A Computational Study, Princeton University Press, 2006

[7] http://www.gcd.org/sengoku/docs/arob98.pdf

[8] http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/

[9] http://www.tsp.gatech.edu/

[10] T.H. Cormen, Ch.E. Leiserson, R.L. Rivest, C. Stein, Wprowadzenie do algorytmów, WNT, Warszawa 2004

[11] Bjarne Stroustrup, Język C++, WNT, Warszawa 1994