

Wydział Informatyki

**Projekt zaliczeniowy**

*Algorytmy genetyczne o reprezentacji binarnej i zmiennopozycyjnej jako optymalizatory funkcji wielu zmiennych.*

**Wykonanie:** Krzysztof Dziemiańczuk, Łukasz Dworakowski, Marcin Joka

**Rodzaj studiów:** dzienne

**Kierunek studiów:** Informatyka

**Specjalność:** Inżynieria Oprogramowania

**Przedmiot:** Algorytmy Ewolucyjne

**Prowadzący:** dr inż. Wojciech Kwedlo

**Rok Akademicki:** 2014/2015

**Treść zadania:**

Algorytmy genetyczne o reprezentacji binarnej i zmiennopozycyjnej jako optymalizatory funkcji wielu zmiennych. Implementacja standardowego algorytmu genetycznego (Goldberg) oraz algorytmu wykorzystującego chromosomy z reprezentacją zmiennopozycyjną (Michalewicz). Zastosowanie obydwu algorytmów do optymalizacji co najmniej pięciu testowych funkcji (np. Rastrigina, Griewanka, Rosenbrocka, ... ). Badania symulacyjne i porównanie obydwu algorytmów.

**Zrozumienie i cel zadania:**

Zadanie polegało na zaimplementowaniu dwóch algorytmów genetycznych:

* Standardowego algorytmu genetycznego,
* Algorytmu wykorzystującego chromosomy z reprezentacją zmiennopozycyjną.

W późniejszej fazie przeprowadzone zostały badania symulacyjne, na podstawie których dokonana została analiza i porównanie obu algorytmów.

Pierwszy z wymienionych algorytmów genetycznych został opracowany w 1975 roku przez Johna H. Hollanda, a badania nad jego optymalizacją prowadzili jego doktoranci – Kenneth A. de Jong oraz David E. Goldberg. Algorytm bazuje na reprezentacji binarnej chromosomów tj. każdy gen to pojedynczy bit, a rozwiązanie jest ciągiem binarnym.

Drugi algorytm został opracowany przez Z. Michalewicza w 1996 roku. W odróżnieniu do standardowego algorytmu genetycznego, do reprezentacji wykorzystuje liczby rzeczywiste.

Celem zadania było przeprowadzenie badań prowadzących do porównania i oceny wyników działania obu stworzonych algorytmów z uwzględnieniem zastosowania różnych sposobów selekcji, operatorów mutacji i krzyżowania.

**Standardowy algorytm genetyczny (Goldberg):**

**W projekcie zaimplementowano 6 funkcji testowych:**

* funkcja Rastringa,
* funkcja Griewanka
* dolina Rosenbrocka,
* funkcja Easoma,
* funkcja Schaffera F6,
* funkcja De Jonga.

Ponadto umożliwiono wybór spośród następujących metod selekcji:

* selekcja turniejowa,
* selekcja proporcjonalna,
* selekcja rangowa.

Oraz operatorów krzyżowania:

* krzyżowanie jednopunktowe,
* krzyżowanie wielopunktowe,
* krzyżowanie jednostajne.

Warunkiem stopu dla działania funkcji jest wykonanie zadanej liczby iteracji.

W trakcie wykonywania badań przeprowadzono testy jakości osiąganych wyników pod kątem:

* zmiany wielkości populacji,
* zmian prawdopodobieństwa mutacji,
* zmiany wielkości elity,
* wyboru różnych metod selekcji wraz z edycją parametrów,
* wyboru różnych operatorów krzyżowania.

**Wpływ wielkości populacji:**

W trakcie obserwacji zauważono, iż wzrost populacji zwiększa szansę na znalezienie optymalnego rozwiązania. W związku z tym, iż na początku obliczeń osobniki rozmieszczane są losowo, większa populacja (większa ilość osobników) wpływa na wylosowanie osobnika w pobliżu rozwiązania optymalnego. Im większa jest liczba obiektów w populacji, tym mniejsze prawdopodobieństwo ugrzęźnięcia w minimum lub maksimum lokalnym, a większe prawdopodobieństwo odnalezienia globalnego. Ma to kluczowe znaczenie, korzystając z funkcji, które mają liczne optima lokalne i tylko jedno globalne.

Z tych obserwacji wynika, iż najlepsze rozwiązanie otrzymamy przy możliwie największej populacji. Należy jednak pamiętać, że każdy kolejny osobnik (liczba populacji) wydłuża czas obliczeń. Biorąc pod uwagę liczbę iteracji, którą musi wykonać program w trakcie obliczeń, wielkość populacji powinna stanowić kompromis pomiędzy czasem obliczeń a uzyskanym rezultatem. Wyniki uzyskiwane przez wykorzystanie różnych funkcji znacznie się od siebie różniły.

Warto mieć na uwadze, iż przy kombinacji duża wielkość populacji + niewielka ilość iteracji oraz mała wielkość populacji + duża ilość iteracji otrzymujemy wyniki niskiej jakości.

**Wpływ prawdopodobieństwa mutacji:**

Mutacja polega na losowej zmianie bitów poszczególnych chromosomów na przeciwny. Z określonym (niewielkim) prawdopodobieństwem może zajść dla dowolnego bitu. Wielkość tego prawdopodobieństwa charakteryzuje jak często, w osobnikach danej populacji, będą zachodziły mutacje.

W przeprowadzonych badaniach widać wyraźnie, że prawdopodobieństwo mutacji wpływa na zróżnicowanie wyników. Dzięki mutacji możliwe jest uniknięcie sytuacji, w której algorytm utknąłby w minimum lokalnym. Prowadzi to ze sobą również szansę, na pojawienie się dużo lepszego (lub gorszego) osobnika w populacji, który ją zdominuje.

**Wpływ wielkości elity:**

W procesie wykonywania się algorytmu genetycznego, może zdarzyć się taka sytuacja, że w kolejnej populacji, nie zostali uwzględnieni najlepsi osobnicy (np. kiedy cała populacja rodziców jest zastępowana populacją potomków). Aby zapewnić przetrwanie najlepszych, wykorzystuje się model sukcesji elitarnej. Najlepsi osobnicy (elita) z populacji rodziców są włączani do populacji potomnej w miejsce najsłabszych osobników.

Wprowadzenie elity znacząco poprawia jakość działania algorytmu genetycznego. Należy mieć na uwadze fakt, iż zbyt duża wielkość elity zwiększa presję selekcyjną, a co za tym idzie, zwiększa prawdopodobieństwo ugrzęźnięcia w minimum lub maksimum lokalnym. W związku z tym wielkość elity nie powinna przekraczać kilku osobników.

**Wpływ zmian metod selekcji:**

W programie zaimplementowano trzy metody selekcji (proporcjonalną, turniejową i rangową). W trakcie badań zauważono, iż selekcja turniejowa potrzebuje najmniej iteracji, aby wyznaczyć rozwiązanie najbliższe optimum. Tuż za nią plasowała się selekcja rangowa. Najgorzej radzi sobie selekcja proporcjonalna (kilku/kilkunasto-krotnie więcej iteracji do uzyskania zbliżonego wyniku).

Pomimo, iż wydaje się, że selekcja proporcjonalna jest najgorszą metodą, zdarzają się sytuację, kiedy okazuje się to najlepszy sposób selekcji. Selekcja ta jest odporna na ugrzęźnięcia w lokalnych optimach (w odróżnieniu od pozostałych sposobów selekcji), co w niektórych przypadkach rozwiązywanych problemów może okazać się zbawienną opcją.

**Wpływ zmian operatorów krzyżowania:**

W programie zaimplementowano trzy operatory krzyżowania (jednopunktowe, wielopunktowe i jednostajne). W wyniku przeprowadzanych badań, zaobserwowano skrajnie zróżnicowane wyniki. Wybrany operator krzyżowania był odpowiedni dla jednego typu funkcji, dla innego zaś okazywał się najgorszym (np. dla funkcji Rastrigina najlepsze było krzyżowanie wielopunktowe, które w przypadku doliny Rosenbrocka się nie sprawdziło). Operator należy dobierać, więc zgodnie ze specyfiką rozwiązywanego problemu.

Na bazie wyników można wysnuć wniosek, iż jednopunktowy operator krzyżowania jest najbezpieczniejszy i najbardziej uniwersalny.

**Algorytm wykorzystujący chromosomy z reprezentacją zmiennopozycyjną (Michalewicz):**

Algorytm genetyczny wykorzystujący chromosomy z reprezentacją zmiennopozycyjną jest modyfikacją standardowego algorytmu genetycznego. W tym przypadku reprezentacja binarna została zamieniona na reprezentację składającą się z liczb rzeczywistych. Dla tego algorytmu wykorzystuje się również inne operatory krzyżowania oraz mutacji. Zasada działania pozostaje jednak taka sama.

Wcześniej zaimplementowany standardowy algorytm genetyczny został wykorzystany do wykonania kolejnej części zadania. Mianowicie część kodu została wykorzystana, a stworzenie algorytmu się o reprezentacji zmiennopozycyjnej sprowadzało się do napisania kilku dodatkowych funkcji.

Zaimplementowano kolejne operatory krzyżowania:

* pełne krzyżowanie arytmetyczne,
* pojedyncze krzyżowanie arytmetyczne,
* proste krzyżowanie arytmetyczne.

Oraz dodano operator mutacji nierównomiernej.

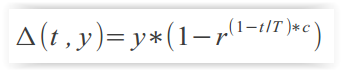
Pozostałe funkcjonalności tego algorytmu nie różnią się od funkcjonalności standardowego algorytmu genetycznego, wobec czego nie będą one powtórnie badane i analizowane.

W tej części przeprowadzono testy jakości osiąganych wyników pod kątem:

* zmiany wielkości parametru C dla mutacji nierównomiernej,
* wyboru różnych operatorów krzyżowania.

**Wpływ parametru C w mutacji nierównomiernej:**

Mutacja nierównomierna bazuje na założeniu, iż chromosomy są modyfikowane w coraz mniejszym stopniu wraz z zaawansowaniem algorytmu. Innymi słowy na początku obliczeń algorytmu mutacja osobników następuje z większą częstotliwością niż wtedy, gdy algorytm zbliża się ku końcowi.



Wzór 1. Mutacja nierównomierna.

We wzorze służącym do obliczenia wartości liczby, o którą modyfikowany jest chromosom, wykorzystuje się parametr C. W trakcie badań zaobserwowano pogorszenie wyników oraz zwiększenie zróżnicowania rezultatów w powtarzanych próbach.

Zgodnie z zaleceniami w literaturze wartość parametru C nie powinna być wysoka, a najbardziej optymalną wartością jest c = 2.

**Wpływ zmian operatorów krzyżowania:**

Wyniki badań zdecydowanie pokazały, że dla każdej funkcji odpowiednie były inne operatory krzyżowania. Nie można wskazać najbardziej uniwersalnego z nich. Przykładowo dla doliny Rosenbrocka najlepszy okazał się operator pojedynczego krzyżowania arytmetycznego, drugi był operator prostego krzyżowania arytmetycznego, a najgorszy operator pełnego krzyżowania arytmetycznego. W przypadku funkcji Rastrigina, operatory pod kątem najlepszej jakości, wyniku ułożyły się dokładnie odwrotnie.

Operator krzyżowania dla algorytmu genetycznego o reprezentacji zmiennopozycyjnej powinien być dobierany po analizie specyfiki rozwiązywanego problemu.

**Porównanie obu algorytmów genetycznych:**

Aby móc porównać działanie obu algorytmów genetycznych przeprowadzono dwa badania:

* niewielka populacja, mała ilość iteracji, funkcje testowe 2 wymiaru
* niewielka populacja, duża ilość iteracji, funkcje testowe 30 wymiaru

**Badanie nr 1:**

Pierwsze badanie wykonano dla wszystkich zaimplementowanych funkcji testowych drugiego wymiaru. Można się spodziewać, że najlepiej w tym przypadku poradzi sobie standardowy algorytm genetyczny. Przyjęto, że algorytm standardowy potrzebuje 50 iteracji, aby osiągnąć rozwiązanie stabilne, to znaczy takie, które w czasie trwania kolejnych iteracji nie poprawi dalej swojego rozwiązania. Badanie dla algorytmu standardowego przeprowadzono w oparciu o następujące parametry:

|  |  |
| --- | --- |
| **Parametr** | **Wartość** |
| Wielkość populacji | 800 |
| Precyzja | 6 |
| Prawdopodobieństwo mutacji | 0,001 |
| Krzyżowanie | Jednopunktowe |
| Selekcja | Turniejowa |
| Rozmiar turnieju | 4 |
| Wielkość elity | 2 |
| Wymiar funkcji | 2 |
| Warunek stopu | Liczba iteracji |
| Wartość warunku stopu | 50 |
| Długość przedziału | 10 [-5, 5] |

Tabela 1. Parametry standardowego algorytmu genetycznego zastosowane do porównania jakości osiągniętych rezultatów optymalizacji poszczególnych funkcji testowych o niskim wymiarze.

W wyniku eksperymentów osiągnięto następujące wyniki:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Funkcja** | **Schaffera F6** | **Rastrigina** | **Easoma** | **De Jonga 1** | **Dolina Rosenbrocka** | **Griewanka** |
| Przeciętne dopasowanie | 0,009716 | -1.987662 | -0,999587 | 0 | 0,024415 | 0 |

Tabela 2. Wyniki badania przy zastosowaniu standardowego algorytmu genetycznego dla funkcji o niskim wymiarze.

Dla każdej z testowanych funkcji algorytm osiągnął wynik optymalny lub bardzo zbliżony do optymalnego. Największe problemy miał on z optymalizacją funkcji Schaffera F6. Nie zawsze udawało się osiągnąć optimum globalne dla funkcji Rastrigina, Easoma i Doliny Rosenbrocka. Pozostałe funkcje w każdym wywołaniu zostały w pełni zoptymalizowane. Odchylenie2standardowe poszczególnych rezultatów jest bliskie zeru. Należy podkreślić, że nawet gdy osiągnięty wynik nie był optymalny, bardzo był do niego zbliżony. Pozwala to na wysoką ocenę stworzonego rozwiązania.

Najdłużej trwała optymalizacja funkcji problematycznych. W pozostałych przypadkach ilość iteracji potrzebnych do osiągnięcia optymalnego wyniku była zbliżona i relatywnie niewielka. To także świadczy o poprawności i wysokiej jakości zaimplementowanego algorytmu.

Po sprawdzeniu, jak ukształtowały się wyniki dla standardowego algorytmu genetycznego, przeprowadzono badanie dla algorytmu zmiennopozycyjnego. Dla każdej funkcji algorytm przebył tyle iteracji, ile potrzebował algorytm standardowy do osiągnięcia stabilnego wyniku. Parametry badań kształtowały się tak, jak zaprezentowano w poniższej tabeli.

|  |  |
| --- | --- |
| Wielkość populacji | 800 |
| Precyzja | 6 |
| Prawdopodobieństwo mutacji | 0,001 |
| Krzyżowanie | Proste krzyżowanie arytmetyczne |
| Selekcja | Turniejowa |
| Rozmiar turnieju | 4 |
| Wielkość elity | 2 |
| Wymiar funkcji | 2 |
| Warunek stopu | Liczba iteracji: 50 |
| Długość przedziału | 10 [-5, 5] |
| Parametr C | 2 |

Tabela 3. Parametry algorytmu genetycznego o reprezentacji zmiennopozycyjnej zastosowane do porównania jakości osiągniętych rezultatów optymalizacji poszczególnych funkcji testowych o niskim wymiarze.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Funkcja** | **Schaffera F6** | **Rastrigina** | **Easoma** | **De Jonga 1** | **Dolina Rosenbrocka** | **Griewanka** |
| Przeciętne dopasowanie | 0,001047 | -1,68864 | -0,962531 | 0,000962 | 0,481815 | 0 |

Tabela 4. Wyniki badania przy zastosowaniu algorytmu genetycznego o reprezentacji zmiennopozycyjnej genetycznego dla funkcji o niskim wymiarze.

Jak widać, w przypadku algorytmu o reprezentacji zmiennopozycyjnej z reguły osiągano rozwiązanie bliskie optymalnemu, z wyjątkiem funkcji Rosenbrocka, której nie udało się zoptymalizować. Rozwiązania w większości przypadków były jednak gorszej jakości, niż w przypadku algorytmu standardowego. Jedyną funkcją, dla której osiągnięto nie gorsze rezultaty przy użyciu algorytmu o reprezentacji zmiennopozycyjnej, jest funkcja Michalewicza. Porównanie rezultatów działania obu algorytmów przedstawiono poniżej.

Zgodnie z wcześniejszym omówieniem rezultatów, dla każdej funkcji wyniki działania algorytmu standardowego są nie gorsze niż w przypadku algorytmu o reprezentacji zmiennopozycyjnej.

**Badanie nr 2:**

Drugie badanie przeprowadzono dla funkcji testowych o znacznie większym wymiarze. Problem ten jest dużo bardziej zbliżony do rzeczywistego, niż w przypadku poprzedniego badania. Przyjęto, że badana będzie funkcja o wymiarze równym 30. Zgodnie z teorią można spodziewać się, że znacznie lepiej w takich warunkach poradzi sobie algorytm genetyczny o reprezentacji zmiennopozycyjnej. W przypadku tego badania przyjęto klasyczne, bardziej miarodajne założenia, niż w przypadku poprzedniego badania. Jego celem nie było bowiem sprawdzenie hipotezy o wyższości jednego algorytmu nad drugim, lecz sprawdzenie, który z nich sprawdzi się lepiej. Założono stałą liczbę iteracji i sprawdzono, przy użyciu którego z algorytmów otrzymane zostanie rozwiązanie lepsze.

|  |  |
| --- | --- |
| **Parametr** | **Wartość** |
| Wielkość populacji | 600 |
| Precyzja | 6 |
| Prawdopodobieństwo mutacji | 0,001 |
| Krzyżowanie | Wielopunktowe |
| Selekcja | Rangowa |
| Presja selekcyjna | 1.2 |
| Wielkość elity | 0 |
| Wymiar funkcji | 30 |
| Warunek stopu | Liczba iteracji |
| Wartość warunku stopu | 20000 |
| Długość przedziału | 100 [-50, 50] |

Tabela 5. Parametry standardowego algorytmu genetycznego zastosowane do porównania jakości osiągniętych rezultatów optymalizacji poszczególnych funkcji testowych o dużym wymiarze.

Po przeprowadzeniu znacznie dłużej trwających niż w przypadku poprzedniego badania obliczeń otrzymano następujące rezultaty:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Funkcja** | **Schaffera F6** | **Rastrigina** | **De Jonga 1** | **Dolina Rosenbrocka** | **Griewanka** |
| Przeciętne dopasowanie | 0,47985321 | 35146,86548 | 37214,29846 | 6124589467 | 11,00468406 |

Tabela 6. Wyniki badania przy zastosowaniu standardowego algorytmu genetycznego dla funkcji o dużym wymiarze.

Otrzymane wyniki znacznie różnią się od optymalnych. W przypadku funkcji Rastrigina, De Jonga oraz Doliny Rosenbrocka, algorytm standardowy kompletnie nie poradził sobie z zadaniem. Obliczone rozwiązania w przypadku tych funkcji nie zbliżały się nawet do optimum, były nieracjonalnie od niego odlegle. Nieco lepsze wyniki otrzymano w przypadku pozostałych funkcji. Także nie były one optymalne, jednak ich odchylenia standardowe świadczą o tym, że były stabilne. Rozwiązanie było niskiej jakości, jednak było rozwiązaniem dążącym do poprawności. Nie można tego powiedzieć o trzech wymienionych wcześniej funkcjach, dla których odchylenia standardowe poszczególnych wyników były bardzo duże. Zdecydowanie najlepiej udało się algorytmowi zoptymalizować funkcję Schaffera F6. Otrzymane rozwiązanie było bardzo bliskie optymalnemu.

To samo badanie przeprowadzono dla algorytmu o reprezentacji zmiennopozycyjnej. Parametry badania przedstawiają się następująco:

|  |  |
| --- | --- |
| **Atrybut** | **Wartość** |
| Wielkość populacji | 600 |
| Precyzja | 6 |
| Prawdopodobieństwo mutacji | 0,001 |
| Krzyżowanie | Proste krzyżowanie arytmetyczne |
| Selekcja | Rangowa |
| Presja selekcyjna | 1,2 |
| Wielkość elity | 0 |
| Wymiar funkcji | 30 |
| Warunek stopu | Liczba iteracji: 20000 |
| Długość przedziału | 100 [-50, 50] |
| Parametr C | 2 |

Tabela 7. Parametry algorytmu genetycznego o reprezentacji zmiennopozycyjnej zastosowane do porównania jakości osiągniętych rezultatów optymalizacji poszczególnych funkcji testowych o dużym wymiarze.

Przeprowadzone badanie dało następujące rezultaty:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Funkcja** | **Schaffera F6** | **Rastrigina** | **De Jonga 1** | **Dolina Rosenbrocka** | **Griewanka** |
| Przeciętne dopasowanie | 0,01502459 | -8,94812478 | 0,70245987 | 1578,41342 | 0,00294298 |

Tabela 4. Wyniki badania przy zastosowaniu algorytmu genetycznego o reprezentacji zmiennopozycyjnej genetycznego dla funkcji o dużym wymiarze.

Widać wyraźnie, że wyniki otrzymane w wyniku działania algorytmu o reprezentacji zmiennopozycyjnej są nieporównywalnie lepsze niż w przypadku algorytmu standardowego. Wyjątek stanowi tuta, o ironio, funkcja Michalewicza, która została lepiej zoptymalizowana przez standardowy algorytm genetyczny. Za wyjątkiem Doliny Rosenbrocka, wszystkie funkcje zostały dobrze zoptymalizowane. Zadany problem był na tyle trudny, że algorytm nie osiągnął rozwiązań optymalnych, wyraźnie jednak do nich się zbliżał. Za każdym razem, za wyjątkiem Doliny Rosenbrocka, rozwiązania były stabilne – odchylenia standardowe poszczególnych uruchomień są bardzo niskie.

Na podstawie przeprowadzonego eksperymentu widać wyraźnie, że z optymalizacją funkcji wielowymiarowej algorytm genetyczny o reprezentacji zmiennopozycyjnej radzi sobie znacznie lepiej niż standardowy algorytm genetyczny. Wynik ten potwierdza teorię, która mówi o tym, że z rozwiązywaniem rzeczywistych problemów optymalizacyjnych algorytm o reprezentacji zmiennopozycyjnej jest bardziej przydatny niż algorytm o reprezentacji binarnej.

**Wnioski:**

W ramach wykonania zadania zaimplementowano dwa rodzaje algorytmów genetycznych: standardowy algorytm genetyczny o reprezentacji binarnej oraz algorytm o reprezentacji zmiennopozycyjnej. Oprócz standardowej wersji algorytmów, dokonano implementacji szeregu dodatkowych funkcjonalności i alternatyw możliwych do wykorzystania przy użyciu każdego z algorytmów, ulepszających je i umożliwiających owocne zastosowanie ich w różnego rodzaju problemach optymalizacyjnych. Na podstawie zaimplementowanego rozwiązania przeprowadzono szereg badań dotyczących optymalnej wielkości poszczególnych parametrów algorytmu oraz jego dodatkowych funkcjonalności. W przypadku każdego badania optymalne okazywały się wielkości parametrów przedstawiane zwykle w literaturze. Świadczy to o poprawnej implementacji algorytmu i wszystkich zaimplementowanych dodatków. Przeprowadzenie tego rodzaju badań pozwoliło na empiryczne sprawdzenie powszechnie znanej teorii oraz dobór optymalnych parametrów pod kątem rozwiązania zadanego, konkretnego problemu – optymalizacji funkcji testowych.

Po dobraniu optymalnych parametrów zbadano, jak zaimplementowane algorytmy radzą sobie z zadanym w ramach projektu problemem. Obie wersje programu wykorzystano do optymalizacji przykładowych funkcji testowych. Eksperyment podzielono na dwie części. Sprawdzono, który z algorytmów genetycznych poradzi sobie lepiej z optymalizacją prostych, dwuwymiarowych funkcji, a który okaże się skuteczniejszy dla funkcji wielowymiarowych. Okazało się, że z prostymi funkcjami lepiej uporał się standardowy algorytm genetyczny, chociaż algorytm o reprezentacji zmiennopozycyjnej także dał dobre wyniki. Przy problemie optymalizacji funkcji wielowymiarowych znacząco lepiej sprawdził się algorytm o reprezentacji zmiennopozycyjnej. W tym przypadku różnica była wyraźna – standardowy algorytm genetyczny nie sprostał zadaniu i dla większości przetestowanych funkcji dał wyniki odległe od optymalnych. Dobór algorytmu oraz jego parametrów powinien być oparty o analizę problemu, do którego rozwiązania ma on posłużyć. W przypadku niektórych problemów lepiej sprawdzi się algorytm binarny, w przypadku innych – zmiennopozycyjny. Podobnie jest z parametrami algorytmu. Przykładowo, dla zadanego problemu optymalizacji funkcji, w zależności od tego, jaka jest to funkcja, jak szeroka jest przestrzeń jej rozwiązań, jak wysoki jest jej wymiar, należy rozważnie dobrać wielkość populacji, warunek stopu, czas trwania algorytmu, rodzaj selekcji, operator krzyżowania, mutacji, wielkość elity itd. Wybór powinien odbyć się w oparciu o analizę problemu. Należy rozważyć między innymi, jak szybko algorytm powinien dążyć do rozwiązania optymalnego, wybrać optymalny kompromis między szybkością osiągniętego rozwiązania a ryzykiem utknięcia w minimum lokalnym, zdecydować o wpływie czynnika losowego (mutacji) na postać rozwiązania. W podobny sposób należy rozwiązać wpływ wszystkich istotnych aspektów algorytmu, omawianych we wcześniejszej części raportu podczas opisu poszczególnych badań, na otrzymane w wyniku jego działania rozwiązanie.

Celem projektu było porównanie rezultatów otrzymanych w wyniku działania standardowego algorytmu genetycznego z rozwiązaniami, które dał algorytm genetyczny o reprezentacji zmiennopozycyjnej. Jak widać, oba algorytmy mogą być przydatne i dawać dobre rozwiązania w zależności od natury problemu. Należy tutaj jednak podkreślić, że standardowy algorytm genetyczny lepiej sprawdził się w przypadku akademickim, prostym, nie stanowiącym rzeczywistego problemu, natomiast algorytm genetyczny o reprezentacji zmiennopozycyjnej okazał się być użyteczny w przypadku skomplikowanego problemu optymalizacyjnego. Do rozwiązywania takich właśnie, niebanalnych problemów, służyć powinien algorytm genetyczny. Pod tym względem należy wyżej ocenić algorytm genetyczny o reprezentacji zmiennopozycyjnej jako bliższy rzeczywistym, realnym problemom. Wszystkie założenia zadania postawione przed jego wykonaniem zostały spełnione, a rezultaty empirycznie potwierdziły oczekiwania. Wobec tego można stwierdzić, że stworzone rozwiązanie zostało zrealizowane poprawnie i cechuje się wysoką jakością.