ZASTOSOWANIE ALGORYTMÓW EWOLUCYJNYCH

Wyważanie dźwigni

Matematyczne Metody Wspomagania Decyzji

Autorzy: Mariusz Grabowski, Michał Grenda, Rafał Gembalczyk

Wydział: Elektrotechniki, Automatyki, Informatyki i Inżynierii Biomedycznej

Kierunek: Automatyka i Robotyka

Rok: III

Spis Treści

Spis Treści 2

1 Model zagadnienia 4

1.1 Opis 4

Przyjęte uproszczenia 4

1.2 Model matematyczny 4

Ilustracja problemu 4

Oznaczenia 4

Postać funkcji celu 4

Rozwiązanie 4

Warunki ograniczające 4

2 Zastosowany algorytm 5

2.1 Uproszczony schemat implementacji 5

2.2 Opis Implementacji 5

Selekcja 5

Operatory krzyżowania 6

Operatory mutacji 6

2.3 Parametry 7

3 Aplikacja 7

3.1 Wykorzystane oprogramowanie 7

Python 3.7 7

Biblioteki 7

Inne 7

3.2 Formaty danych 8

Dane wejściowe 8

Postać rozwiązania 8

3.3 Funkcjonalność 8

Czas trwania algorytmu 9

4 Testy 10

4.1 Instancje testowe 10

Mały rozmiar problemu: 10

Średni rozmiar problemu: 10

Duży rozmiar problemu: 11

4.2 Metodyka badań 11

4.3 Charakterystyka zadań testowych 12

4.4 Opis testów 14

Oznaczenia 14

Test z wartościami domyślnymi (do porównywania) 15

Test wpływu parametru w – ilości wybieranych osobników w trakcie selekcji 16

Test wpływu parametru m – wielkość grup w turniejach 18

Test wpływu parametru fCA – częstotliwość alternatywnego krzyżowania 21

Test wpływu parametru amountOfAttemptsSick – ilość prób utworzenia nowego rozwiązania potomnego w krzyżowaniu alternatywnym 24
Test wpływu parametru fM – częstotliwość mutowania 24
Test wpływu parametru fMA – częstotliwość mutowania alternatywnego 27
Test wpływu parametru offset – wielkość przesunięcia w rotateMutation 30
Test wpływu parametru I – ilość iteracji 32
Testy przypadków złośliwych 32

5 Podsumowanie 34
5.1 Wnioski 34
5.2 Stwierdzone problemy 34
5.3 Kierunki dalszego rozwoju 35

6 Podział Pracy 35

1 Model zagadnienia

1.1 Opis

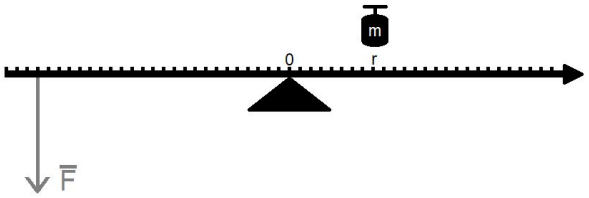
Analizowany problem jest fragmentem dyscypliny optymalizacji matematycznej. W wybranym przykładzie przedstawiono zadanie znalezienia w przybliżeniu optymalnego rozkładu ciężarków w celu wyważenia dźwigni.
Mając określoną liczbę danych odważników, założono potrzebę minimalizacji ustalonego momentu siły z użyciem algorytmu ewolucyjnego.

Przyjęte uproszczenia

- pomija się tarcie oraz wpływy otoczenia na dźwignię,
- tornado, powódź, pożary, depresję, zły humor

1.2 Model matematyczny

Ilustracja problemu



Rysunek 1: Problem rozmieszczenia odważników na dźwigni

Oznaczenia

R – *maksymalna odległość od punktu podparcia dźwigni określana przez ilość możliwych pozycji na których można postawić ciężarek*

$M = r \times F$ – *zadany moment siły*

m_i – *masa punktowa*

M_S – *zbiór ciężarków*

r_i – *odległość ciężarka od punktu podparcia dźwigni*

Postać funkcji celu

$$f_c(x) = |M - g \sum_{i=1}^{n} m_i r_i| \rightarrow min$$

Rozwiązanie

π – *optymalne położenie ciężarków na dźwigni (forma podana w 3.2)*

Warunki ograniczające

– *punkt podparcia znajduje się w centrum*

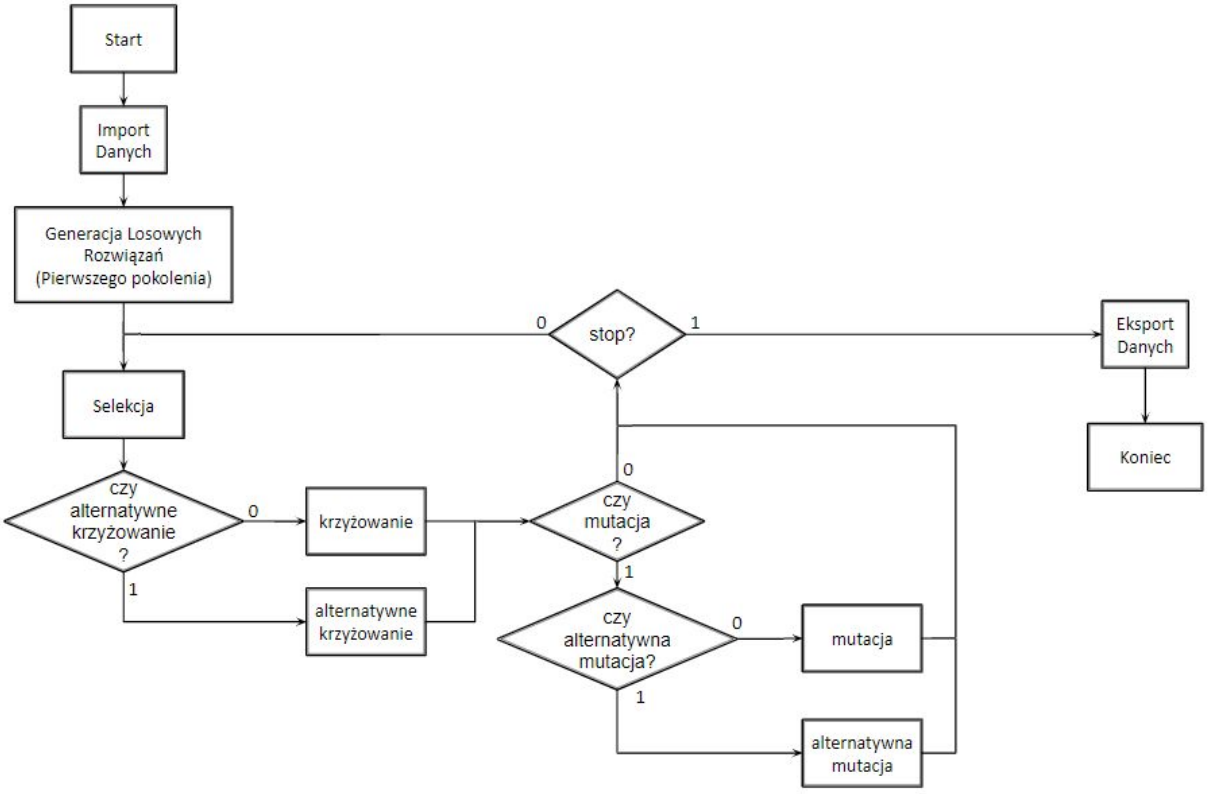
– $\forall_i : \; r_i \in Z \quad |r_i| \leq R$

– $\forall_{i,j} : \; m_i, \; m_j \in M_S \qquad r_i \neq r_j$

– $k \leq 2R + 1$

2 Zastosowany algorytm

2.1 Uproszczony schemat implementacji



Rysunek 2: Schemat postępowania algorytmu

2.2 Opis Implementacji

Selekcja

Zadanie selekcji polega na wyborze osobników, które zostaną wybrane do grupy rozrodczej (operandy krzyżowania) tworzącej nowe osobniki w populacji.
W pierwotnej wersji zaimplementowaliśmy selekcję która wybierała tylko najlepsze osobniki. Jednak ze względu na to że ta metoda była bardzo elitarna i traciliśmy przez to dużo informacji od usuwanych osobników, to zdecydowaliśmy się na metodę turniejową, w której poprzednie pokolenie dzielimy losowo na mniejsze grupy, z których są wybierane najlepsze osobniki.
Aby zapewnić niepowtarzalność osobników - jeśli zwycięzca turnieju został już wcześniej wybrany w innym turnieju, to wybierany jest kolejno osobnik na niższym miejscu w turnieju (2, 3, ... miejsce), a jeśli wszyscy byli już wcześniej wybrani, to tworzony nowy turniej.

Operatory krzyżowania

W algorytmie zastosowano dwa operatory krzyżowania:

- krzyżowanie standardowe
  Polega na tym że dla każdego nowego rozwiązania jest dwóch rodziców, w tym jeden główny (powiedzmy że tata) i drugiego dobrany (mama) (to bez znaczenia czy tata czy mama to główny, używamy tych zwrotów tylko jako skrótów myślowych).
  Dla każdego etapu na początku są losowane geny (miejsca na wadze z informacją jaki ciężarek się na nich znajduje) które będą przekazywane przez ojców, pozostałe będą przekazywane przez matki - zapewnia nam to że dzieci nie będą się powtarzać
  Później następuje iteracja po wszystkich osobnikach. Założeniem jest aby każdy osobnik był tatą przynajmniej raz, do którego jest dobierana losowo mama z pozostałych rozwiązań - rozwiązania mogą być mamą kilka razy, ale też mogą nie być ani razu.
  Ostatecznie nowych rozwiązań jest dokładnie tyle samo co rodziców i jest też zapewnienie że każdy osobnik z poprzedniego pokolenia przynajmniej raz jest rodzicem. (ograniczenie utraty danych od wyselekcjonowanych rozwiązań)

- krzyżowanie alternatywne
  Ta operacja zgodnie z intuicyjnym pojmowaniem teorii ewolucji wybiera połowę najlepszych rozwiązań i każde z nich dobiera w stałe pary z rozwiązaniami drugiej połowy. Im lepsza wartość funkcji celu rozwiązania, tym lepsze zostanie mu przydzielone do pary. Następnie z powstałych par w sposób iteracyjny od jednego do drugiego końca belki dźwigni generowane są losowo nowe rozwiązania. Nowo powstałe rozwiązanie, które jest zgodne ze zbiorem dostępnych ciężarków, jest umieszczane w zbiorze wynikowym, a niezgodne zostaje usuwane. Każda para ma szansę każdorazowo wygenerować jedno rozwiązanie. Iterując po parach w pętli nieskończonej zawartość zbioru wynikowego rośnie aż do momentu, w którym rozmiar wyrówna się z poprzednim pokoleniem. W wyniku czego rozwiązania mogą się powtarzać oraz rodzice nie przedostaną się do kolejnej generacji rozwiązań. Taka właściwość alternatywnego krzyżowania powoduje, że nie zaleca się z niej często korzystać, a jedynie wtedy, gdy pojawia się problem stałości funkcji celu.

Operatory mutacji

Mutacja zapewnia różnorodność osobników i jest aplikowana z pewnym prawdopodobieństwem na losowego osobnika populacji. Zaletą takiego działania jest dużo większa różnorodność rozwiązań, jednak z możliwością zepsucia dobrego rozwiązania. W związku z charakterem algorytmu zaproponowano 2 rodzaje mutacji, które zapewniają przeszukanie względnie dużej części przestrzeni rozwiązań:

- mutowanie standardowe – zmiana pozycji odważnika:
  $[3,\ 0,\ 0,\ 2,\ 5] \rightarrow [0,\ 3,\ 0,\ 2,\ 5]$

  W przypadku kiedy nie ma miejsca wolnego
- mutowanie alternatywne – przesunięcie odważników o dowolną wartość (offset):
  $[1,\ 5,\ 4,\ 6,\ 8,\ 9] \rightarrow [9,\ 1,\ 5,\ 4,\ 6,\ 8]$ (dla wartości 1)

2.3 Parametry

a. wejściowe:
   - M – *moment siły*
   - M_S – *zbiór odważników*
   - S – *zbiór rozwiązań początkowych*
b. generatora zadań:
   - R – *promień belki dźwigni*
   - n – *liczba rozwiązań początkowych*
   - $high$ – *najcięższy możliwy odważnik*
   - k – *liczba odważników*
   - g – *przyspieszenie ziemskie*
c. algorytmu:
   - w – *ilość wybieranych osobników w trakcie selekcji*
   - m – *wielkość grup w turniejach*
   - f_{CA} – *częstotliwość alternatywnego krzyżowania*
   - $amoutntOfAttemptsSick$ – *ilość prób utworzenia nowego rozwiązania potomnego w krzyżowaniu alternatywnym*
   - f_M – *częstotliwość mutacji*
   - f_{MA} – *częstotliwość alternatywnych mutacji*
   - I – *liczba iteracji*

Uwaga! Parametry częstotliwościowe algorytmu, powinny być rozumiane w sensie fizycznym jako okres, to znaczy odwrotność częstotliwości. Im wyższa wartość, tym operacja pojawia się rzadziej.

3 Aplikacja

3.1 Wykorzystane oprogramowanie

Python 3.7

W projekcie jako język został wykorzystany język programowania wysokiego poziomu Python w wersji 3.7. Został wybrany ze względu na bardzo szeroki pakiet bibliotek standardowych oraz funkcjonalnych. Jest absolutnie darmowy (licencja PSFL).
Jedynym minusem jest jego niska relatywna szybkość względem innych języków.

Biblioteki

- NumPy – do obliczeń (zwłaszcza na dużych tablicach)
- Matplotlib – do tworzenia wykresów

Inne

- Git & GitHub
- PyCharm Professional IDE

- Spyder
- JSON – format wymiany danych komputerowych
- Matlab

3.2 Formaty danych

Dane wejściowe

$M - przyłożony\ moment\ siły$
$M_S - zbiór\ ciężarków$
$R - promień\ dźwigni\ (liczba\ miejsc\ po\ jednej\ stronie\ dźwigni)$
$S - Rozwiązania\ początkowe$

Postać rozwiązania

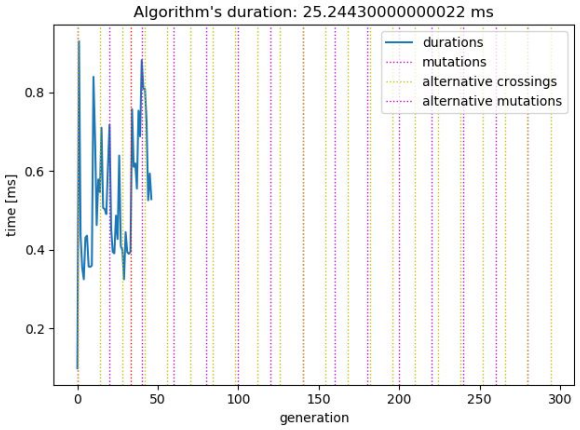
Rozwiązanie przedstawiamy w formie wektora π o długości równej ilości możliwych pozycji na których możemy postawić ciężarek na wadze $(2R+1)$. Indeksy wektora oznaczają odpowiednio kolejne pozycje, a wartości – jaki ciężarek ma zostać postawiony na danej pozycji.

3.3 Funkcjonalność

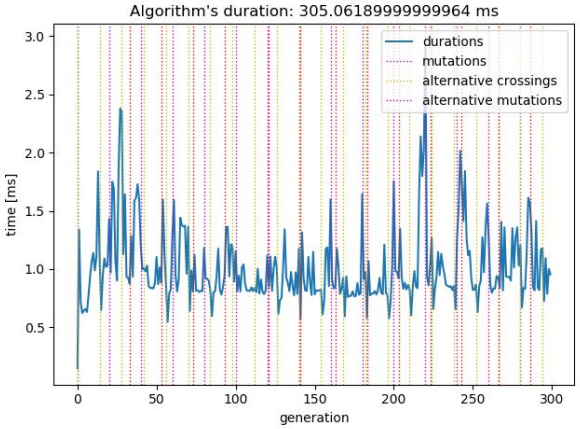
Zaimplementowany algorytm został wyposażony w mechanizmy sprawdzające poprawność sformułowanego problemu oraz możliwość utworzenia nowych rozwiązań:

- *cannotSolve()* – funkcja, która powoduje zwrócenie komunikatu wyjątku „odważniki są zbyt lekkie, aby zredukować wypadkowy moment siły do zera”, jeśli najsilniej wpływające na moment obrotowy ustawienie dostępnych odważników nie jest w stanie sprawić, że funkcja celu przyjmie zerową wartość,
- *sick()* – funkcja, która zwraca negatywną wartość logiczną, jeśli utworzone rozwiązanie nie może zostać utworzone z danego zbioru odważników,
- jeśli ilość dostępnych odważników jest większa od liczby dostępnych na belce dźwigni miejsc, to zwracany jest komunikat błędu „odważniki nie mieszczą się na takiej dźwigni”,
- jeśli parametr k opisujący ilość odważników jest różny od właściwego rozmiaru wektora odważników, to zwracany jest komunikat błędu „parametr k nie zgadza się z wektorem odważników MS”,
- jeśli alternatywne krzyżowanie będzie zwracało przez pewien czas niezadowalającego potomka, zostanie użyte domyślne krzyżowanie

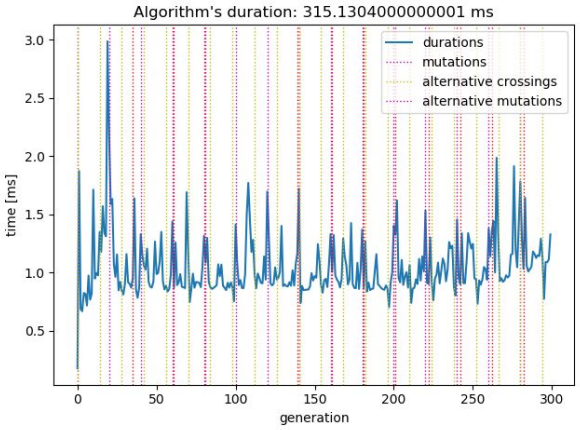
Czas trwania algorytmu



Rysunek: Wykres trwania algorytmu – przypadek 1



Rysunek: Wykres trwania algorytmu – przypadek 2



Rysunek: Wykres trwania algorytmu – przypadek 3

Można zauważyć, że algorytm dla 300 iteracji działa sprawnie. Piki pojawiają się najczęściej na operacjach alternatywnego krzyżowania oraz mutacjach. Jeśli złożoności problemów są do siebie podobne, algorytm wykonuje się w podobnych czasach.

4 Testy

4.1 Instancje testowe

Instancje testowe zostały wygenerowane losowo, aby móc dobrać dane wejściowe obiektywnie.
Zostały wygenerowane trzy instancje testowe (o różnych), dla których przeprowadzono testy:

1. Mały rozmiar problemu:

$(R = 8)$
$M = 750$
$M_S = 1 * [10] + 2 * [5] + 2 * [2] + 5 * [1]$
$S =$
[0, 1, 1, 5, 0, 2, 0, 1, 1, 0, 0, 1, 0, 5, 0, 2, 0]
[0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0]
[1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1]
[0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0]
[1, 1, 0, 1, 0, 0, 0, 2, 0, 0, 1, 0, 2, 0, 0, 1, 5]
[0, 0, 1, 5, 0, 0, 1, 1, 2, 0, 0, 1, 0, 5, 1, 2, 0]
[1, 1, 0, 1, 0, 0, 5, 0, 1, 0, 2, 0, 0, 2, 0, 0, 1]

2. Średni rozmiar problemu:

$(R = 12)$
$M = 5620$
$M_S = 5 * [8] + 4 * [6] + 3 * [5] + 4 * [4] + 5 * [3] + 3 * [2]$
$S =$
[0, 3, 3, 2, 0, 5, 0, 4, 4, 2, 5, 0, 5, 2, 4, 6, 0, 6, 3, 4, 3, 0, 3, 6, 6],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 2],
[5, 3, 3, 4, 0, 4, 4, 0, 3, 2, 0, 0, 0, 2, 0, 6, 3, 5, 5, 0, 0, 6, 4, 3, 2],
[2, 0, 2, 3, 5, 5, 0, 6, 4, 0, 0, 3, 5, 0, 3, 3, 3, 0, 6, 6, 0, 4, 4, 2, 4],
[3, 0, 6, 4, 5, 2, 3, 8, 2, 6, 4, 5, 4, 3, 2, 0, 8, 5, 0, 0, 3, 6, 6, 3, 4],
[2, 0, 0, 0, 3, 0, 0, 3, 0, 0, 0, 0, 0, 2, 0, 0, 0, 3, 3, 0, 0, 2, 0, 0, 0],
[0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 4, 0, 2, 3, 3, 2, 3, 2, 0, 4],
[0, 0, 0, 4, 0, 0, 0, 0, 0, 2, 0, 3, 0, 3, 3, 0, 3, 0, 0, 0, 0, 2, 2, 3, 0],
[0, 3, 4, 3, 6, 6, 5, 2, 0, 4, 3, 2, 3, 0, 2, 5, 3, 5, 4, 4, 0, 0, 0, 0, 0],
[3, 4, 4, 5, 2, 4, 0, 3, 4, 0, 6, 6, 3, 6, 2, 8, 3, 0, 6, 2, 8, 5, 3, 0, 5],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 3, 0, 0, 0, 0, 0, 3, 2, 0, 3, 0, 2, 0, 0],
[0, 3, 6, 4, 3, 4, 4, 2, 3, 8, 5, 8, 6, 8, 0, 2, 6, 2, 4, 5, 3, 6, 8, 3, 5]

3. Duży rozmiar problemu:

$(R = 20)$
$M = 49000$
$M_S = 2 * [50] + 10 * [25] + 5 * [10] + 5 * [5] + 5 * [3] + 5 * [2] + 5 * [1]$
$S =$
[0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 2, 2, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 2, 0, 0, 1, 0, 0, 0, 2, 0, 2, 0, 0, 3, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0]
[1, 25, 5, 2, 0, 0, 25, 0, 1, 0, 0, 10, 3, 25, 5, 25, 2, 0, 10, 1, 10, 3, 0, 3, 0, 3, 1, 25, 5, 0, 1, 10, 0, 5, 2, 0, 10, 5, 3, 2, 2]
[0, 2, 0, 3, 2, 10, 0, 1, 3, 0, 1, 5, 0, 0, 2, 5, 3, 0, 0, 0, 0, 0, 0, 0, 1, 2, 2, 0, 5, 1, 0, 3, 1, 0, 3, 10, 0, 5, 0, 5, 0]
[2, 0, 2, 0, 0, 0, 1, 0, 0, 0, 1, 0, 3, 1, 3, 0, 2, 5, 0, 0, 0, 0, 1, 0, 2, 0, 3, 0, 0, 0, 0, 2, 1, 3, 0, 0, 3, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[1, 2, 0, 0, 0, 0, 0, 0, 2, 1, 1, 0, 0, 0, 0, 3, 3, 2, 0, 1, 0, 1, 0, 0, 0, 3, 0, 0, 0, 0, 2, 0, 0, 3, 0, 0, 0, 0, 3, 2, 0]
[5, 3, 0, 0, 0, 1, 0, 0, 0, 0, 3, 5, 0, 0, 1, 0, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 3, 2, 2, 2, 2, 3, 2, 3, 0, 1]
[5, 1, 25, 0, 0, 10, 5, 2, 3, 0, 0, 1, 25, 3, 10, 25, 25, 5, 10, 10, 2, 0, 5, 0, 25, 2, 3, 3, 0, 2, 2, 5, 1, 0, 25, 1, 25, 1, 3, 0, 10]
[0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 1, 3, 5, 3, 2, 0, 0, 0, 0, 0, 5, 10, 0, 2, 1, 1, 0, 5, 3, 3, 2, 2, 3, 0, 10, 0, 1, 0, 5, 0, 2, 0, 10, 5, 10, 0, 1, 0, 0]

4. Domyślne wartości parametrów dla testów

w = 8
m = 2
f_{CA} = 14
amountOfAttemptsSick = 50
f_M = 20
f_{MA} = 20
l = 300

4.2 Metodyka badań

1. Testowanie parametrów zostało zaplanowane dla pięciu prób wywołań algorytmu dla każdej z trzech różnych wielkości. Wyniki będą notowane w tabelach wzorowanych na Tabeli 1.

Tabela 1

lp.	testowany parametr	$f_{c_{pocz}}(x)$	$\overline{f_c(x)}_5$	σ_{f_c}	$min_i(f_{c_i}(x))$	$max_i(f_{c_i}(x))$	$I_{śr}$
1.	mały	…	…	…	…	…	…
2.	średni	…	…	…	…	…	…
3.	duży	…	…	…	…	…	…

2. Testowanie zachowania algorytmu dla złośliwych przypadków odbędzie się z jednorazowym wywołaniem funkcji i zorientowaniu się w jaki sposób napisany program jest w stanie sobie poradzić z zadanym problemem. Wyniki zostaną przedstawione w formie porównania wartości funkcji sprzed oraz po zastosowaniu algorytmu. Dodatkowo zostaną przedstawione wykresy przebiegów funkcji celu dla każdego ze skonstruowanych przypadków.

4.3 Charakterystyka zadań testowych

Testy zostaną przeprowadzone na tzw. przypadkach „złośliwych” o znanych odgórnie rozwiązaniach optymalnych:

1. Wartości początkowe nie posiadają odważników na pożądanej stronie (wszystkie rozwiązania początkowe pogłębiają problem):

$(R = 12)$
$M = 3220$
$M_S = 4 * [2] + 4 * [3] + 4 * [6]$
$S =$ [[2,2,2,2,3,3,3,3,6,6,6,6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
[2,2,2,2,6,6,6,6,3,3,3,3,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
[3,3,3,3,6,6,6,6,2,2,2,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
[3,3,3,3,2,2,2,2,6,6,6,6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
[6,6,6,6,2,2,2,2,3,3,3,3,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
[6,6,6,6,3,3,3,3,2,2,2,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]]
$\pi =$ [0,0,0,0,0,0,0,0,0,0,0,0,0,0,3,3,2,2,3,3,6,6,2,2,6,6]

2. Odważniki są rozmieszczone symetrycznie względem punktu podparcia (żadne z rozwiązań początkowych nie wpływa na funkcję celu):

$(R = 13)$
$M = 2600$
$M_S = [1, 1, 2, 2, 3, 3, 4, 4, 5, 5, 6, 6, 7, 7, 8, 8, 9, 9, 10, 10, 11, 11, 12, 12]$

$S =$ [[1,2,0,0,0,0,3,4,5,6,7,8,9,10,9,8,7,6,5,4,3,0,0,0,0,2,1],
[9,8,7,6,5,4,0,0,0,0,3,2,1,10,1,2,3,0,0,0,0,4,5,6,7,8,9],
[1,3,5,7,9,0,0,0,0,2,4,6,8,12,8,6,4,2,0,0,0,0,9,7,5,3,1],
[0,0,0,0,8,6,4,2,9,7,5,3,1,11,1,3,5,7,9,2,4,6,8,0,0,0,0],
[3,2,0,0,1,0,0,6,5,4,9,8,7,12,7,8,9,4,5,6,0,0,1,0,0,2,3],
[7,8,9,4,5,6,1,2,3,0,0,0,0,0,0,0,0,0,3,2,1,6,5,4,9,8,7]]
$\pi =$ [0,0,0,0,1,1,2,2,3,3,4,4,5,0,5,6,6,7,7,8,8,9,9,0,0,0,0]

3. We wszystkich rozwiązaniach użyte odważniki są zbyt ciężkie:

$(R = 9)$
$M = 10$
$M_S = [1, 2, 3, 4, 5, 6]$
$S =$ [[0,3,0,0,0,0,2,0,0,0,0,0,0,5,0,0,0,0,0],
[0,0,0,0,0,0,0,4,3,0,0,6,0,0,0,0,0,0,0],
[0,0,0,0,0,0,5,2,0,0,0,0,6,0,0,0,0,0,0],
[5,0,0,0,0,4,0,0,0,6,0,2,0,0,0,0,0,0,0],
[0,0,0,0,0,0,0,4,0,0,0,0,0,0,0,2,0,0,6],
[0,0,0,5,2,0,0,0,0,0,0,0,0,4,3,0,0,0,6]]
$\pi =$ [0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0]

4. We wszystkich rozwiązaniach użyte odważniki są zbyt lekkie:

$(R = 10)$
$M = 3080$
$M_S = [1, 2, 3, 4, 5, 10, 20]$
$S =$ [[1,2,3,4,5,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
[0,0,0,0,0,0,1,0,4,2,3,0,0,0,0,0,0,0,0,0,0],
[0,0,0,0,3,2,1,0,0,0,4,5,0,0,0,0,0,0,0,0,0],
[0,1,5,0,0,0,0,0,0,0,0,2,3,0,0,0,0,0,0,0,0],
[0,0,0,0,0,0,0,0,0,0,0,0,1,2,3,4,5,0,0,0,0],
[1,0,0,0,0,0,0,0,0,0,2,3,4,0,0,0,0,0,0,0,0],
[0,0,0,0,0,0,0,0,2,3,0,0,0,0,0,0,0,0,0,0,0]]
$\pi =$ [0,0,0,0,0,0,0,0,0,0,0,3,4,0,0,0,0,0,10,20]

5. Wszystkie rozwiązania początkowe są identyczne:

$(R = 12)$
$M = 1000$
$M_S = [1, 2, 4, 6, 8]$

$S =$ [[0,4,0,0,0,0,0,0,6,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
[0,4,0,0,0,0,0,0,6,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
[0,4,0,0,0,0,0,0,6,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
[0,4,0,0,0,0,0,0,6,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
[0,4,0,0,0,0,0,0,6,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
[0,4,0,0,0,0,0,0,6,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
[0,4,0,0,0,0,0,0,6,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
[0,4,0,0,0,0,0,0,6,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
[0,4,0,0,0,0,0,0,6,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
[0,4,0,0,0,0,0,0,6,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1]]
$\pi =$ [0,0,0,0,0,0,0,0,6,0,0,0,0,0,0,0,0,4,0,2,0,0,0,0,0,8]

6. Wszystkie dostępne odważniki zostały wykorzystane (rozwiązania początkowe mają trudności w krzyżowaniu się):

$(R = 7)$
$M = 250$
$M_S = 10 * [1]$
$S =$ [[1,0,1,1,1,1,0,1,1,0,1,1,0,1],
[1,1,1,1,1,1,1,1,1,1,0,0,0,0],
[0,0,0,0,1,1,1,1,1,1,1,1,1,1],
[0,0,1,1,1,1,1,1,0,0,1,1,1,1],
[0,1,0,1,0,1,0,1,1,1,1,1,1,1],
[1,1,1,1,0,0,1,0,0,1,1,1,1,1],
[0,1,1,1,1,1,0,0,1,1,1,1,0,1],
[1,0,0,1,1,1,1,0,1,1,1,1,1,0]]
$\pi =$ [0,0,0,0,0,0,0,0,0,1,1,1,1,1]

4.4 Opis testów

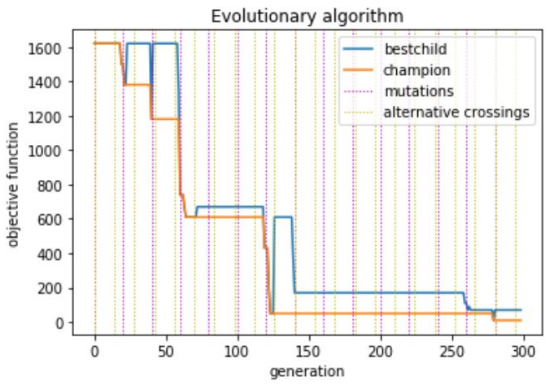
1. Oznaczenia

- $f_{c_{pocz}}(x)$ – *wartość funkcji celu dla najlepszego z rozwiązań początkowych*
- $\overline{f_c(x)}_5 = \frac{\sum_{i=1}^{5} f_{c_i}(x)}{5}$ – *średnia z wartości funkcji celu pięciu wywołań algorytmu*
- σ_{f_c} – *odchylenie standardowe*
- $min_i(f_{c_i}(x))$ – *minimum z wartości funkcji celu pięciu wywołań algorytmu*
- $max_i(f_{c_i}(x))$ – *maksimum z wartości funkcji celu pięciu wywołań algorytmu*
- $I_{śr}$ – *średnia liczba iteracji, po których funkcja celu doszła do swojej najlepszej wartości w danym wywołaniu algorytmu*
- IT – *instancja testowa*

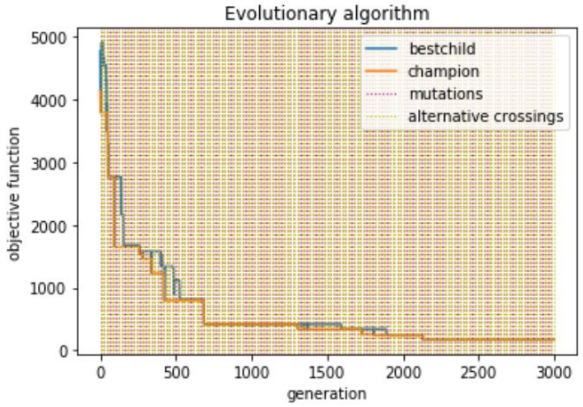
2. Test z wartościami domyślnymi (do porównywania)

Tabela 2

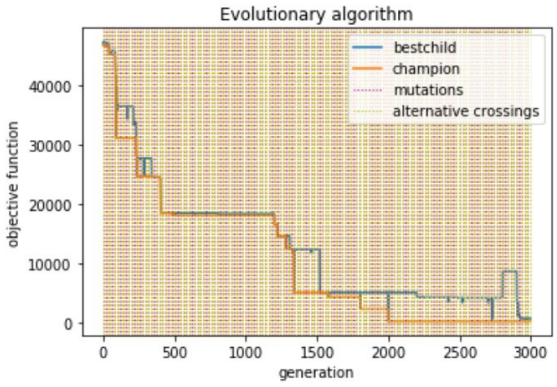
IT	$f_{c_{pocz}}(x)$	$\overline{f_c(x)}_5$	σ_{f_c}	$min_i(f_{c_i}(x))$	$max_i(f_{c_i}(x))$	$I_{śr}$	I_{stop}
1.	400	2	4.47	0	10	204	300
2.	4140	288	131.80	130	430	1938	3000
3.	46940	402	641.54	10	1500	2053	3000



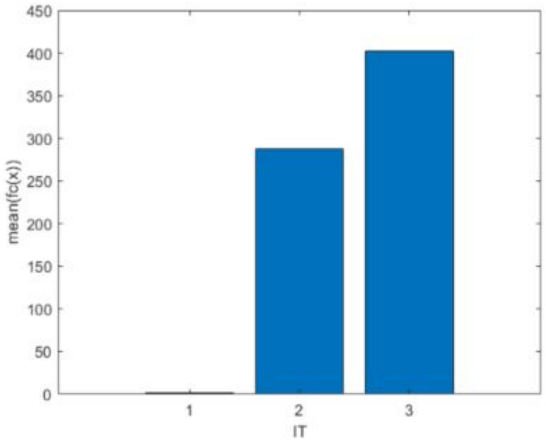
Rysunek 3



Rysunek 4



Rysunek 5

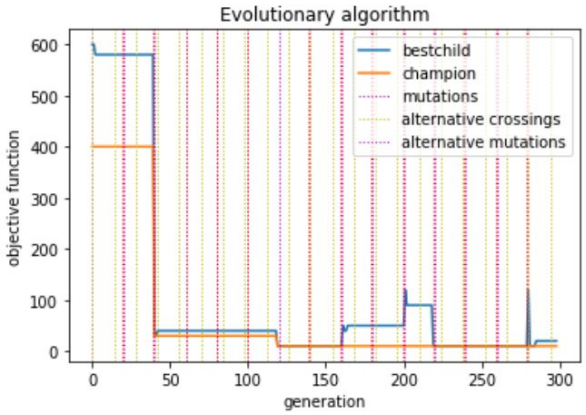


Rysunek 6

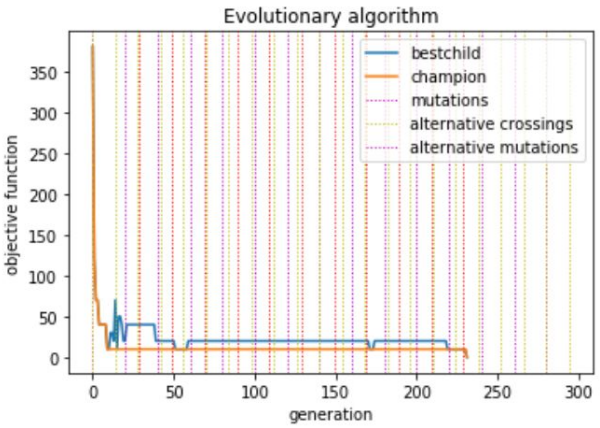
3. Test wpływu parametru w – ilości wybieranych osobników w trakcie selekcji

Tabela 3: IT1

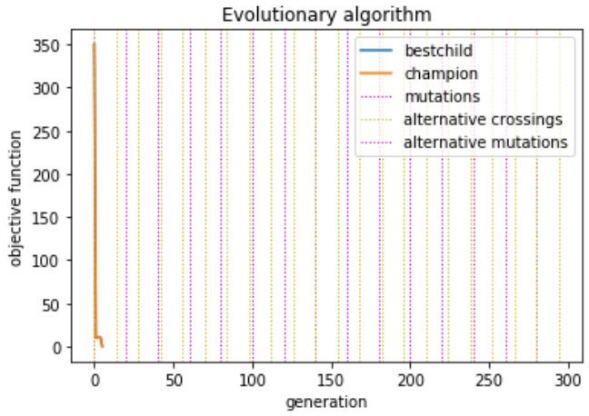
lp.	w	$f_{c_{pocz}}(x)$	$\overline{f_c(x)}_5$	σ_{f_c}	$min_i(f_{c_i}(x))$	$max_i(f_{c_i}(x))$	$I_{śr}$	I_{stop}
1.	2	400	88	174.56	0	400	110	300
2.	10	400	4	5.48	0	10	147	300
3.	50	400	0	0	0	0	5	300



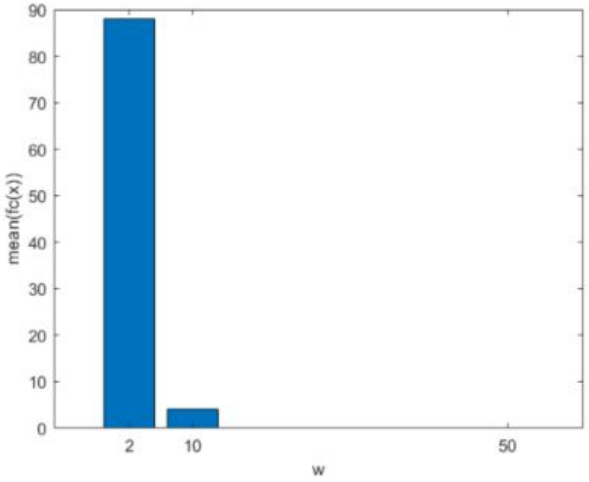
Rysunek 7



Rysunek 8



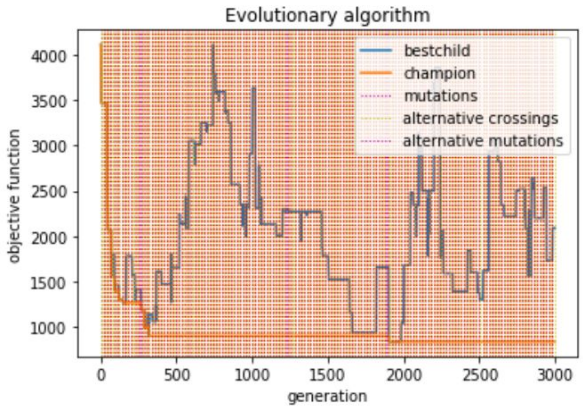
Rysunek 9



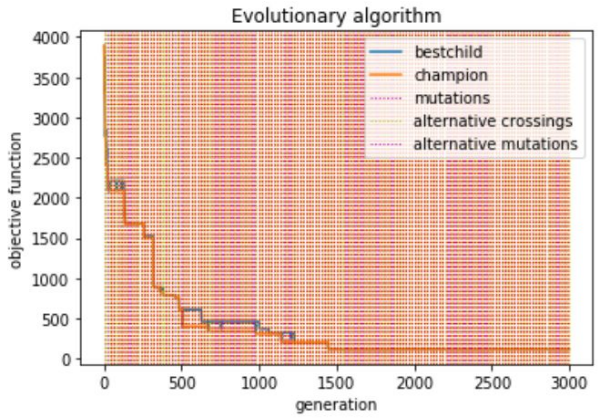
Rysunek 10

Tabela 4: IT2

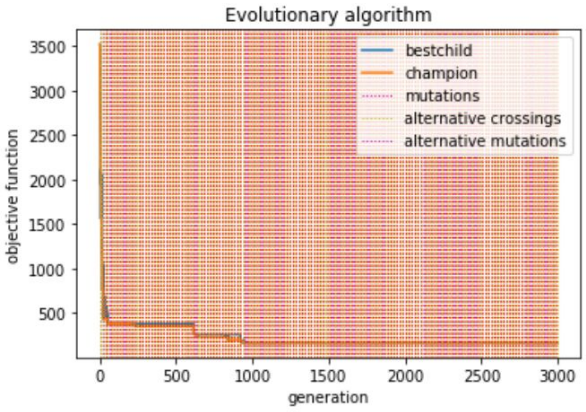
lp.	w	$f_{c_{pocz}}(x)$	$\overline{f_c(x)}_5$	σ_{f_c}	$min_i(f_{c_i}(x))$	$max_i(f_{c_i}(x))$	$I_{śr}$	I_{stop}
1.	2	4140	617	279.31	200	887	2053	3000
2.	10	4140	120	36.74	60	150	1323	3000
3.	50	4140	76	62.69	20	170	1298	3000



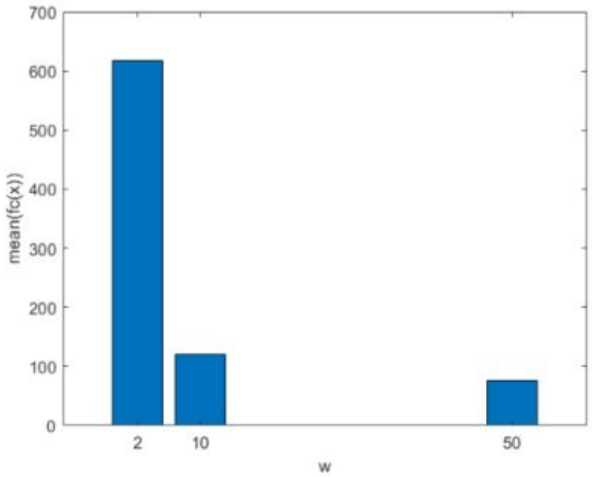
Rysunek 11



Rysunek 12



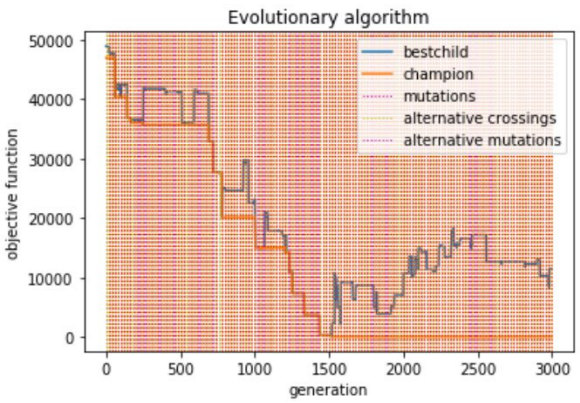
Rysunek 13



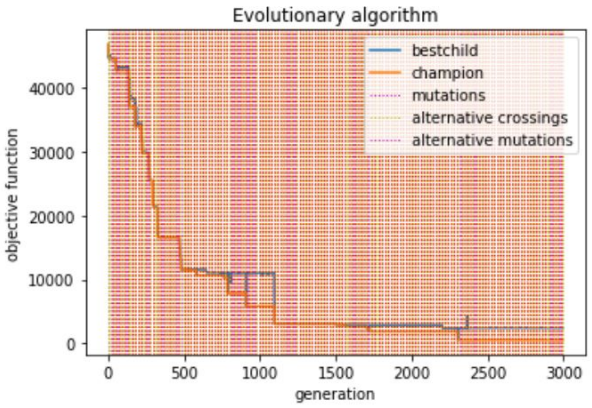
Rysunek 14

Tabela 5: IT3

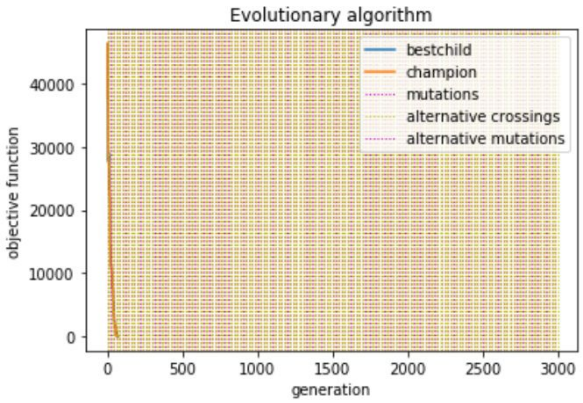
lp.	w	$f_{c_{pocz}}(x)$	$\overline{f_c(x)}_5$	σ_{f_c}	$min_i(f_{c_i}(x))$	$max_i(f_{c_i}(x))$	$I_{śr}$	I_{stop}
1.	2	46940	6340	7794.34	100	19200	1802	3000
2.	10	46940	144	254.42	0	590	3258	3000
3.	50	46940	12	21.68	0	50	470	3000



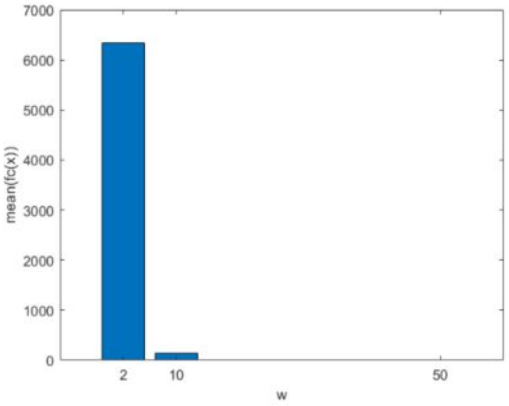
Rysunek 15



Rysunek 16



Rysunek 17



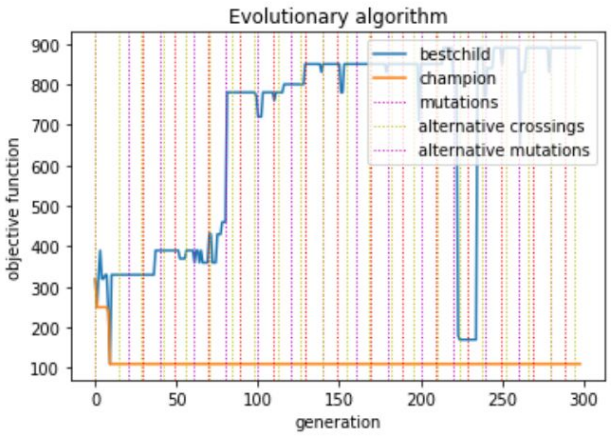
Rysunek 18

Wnioski:
Jak można było się spodziewać im większa ilość wybieranych osobników w trakcie selekcji (co przekłada w naszym algorytmie również na wielkości pokoleń), tym lepiej znajdowane jest rozwiązanie. Jedynym minusem dużych wartości ilości wybieranych osobników, jest konieczność wykorzystywania większych zasobów obliczeniowych.

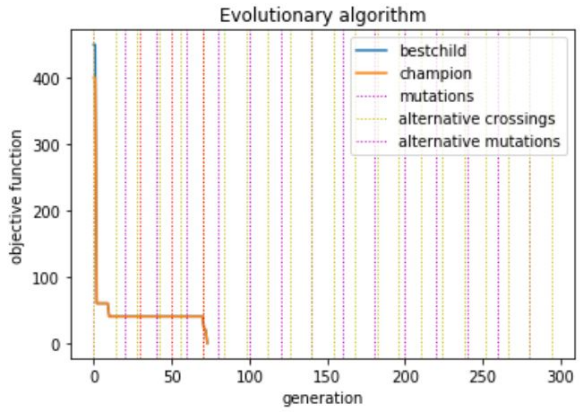
4. Test wpływu parametru m – wielkość grup w turniejach

Tabela 6: IT1

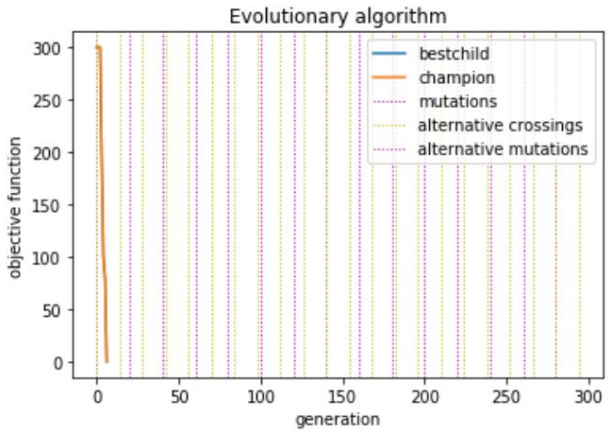
lp.	m	$f_{c_{pocz}}(x)$	$\overline{f_c(x)}_5$	σ_{f_c}	$min_i(f_{c_i}(x))$	$max_i(f_{c_i}(x))$	$I_{śr}$	I_{stop}
1.	1	400	152	172.34	0	400	53	300
2.	4	400	2	4.47	0	10	52	300
3.	8	400	2	4.47	0	10	23	300



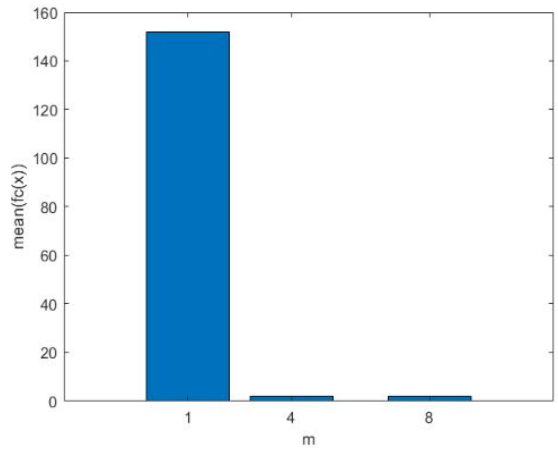
Rysunek 19



Rysunek 20



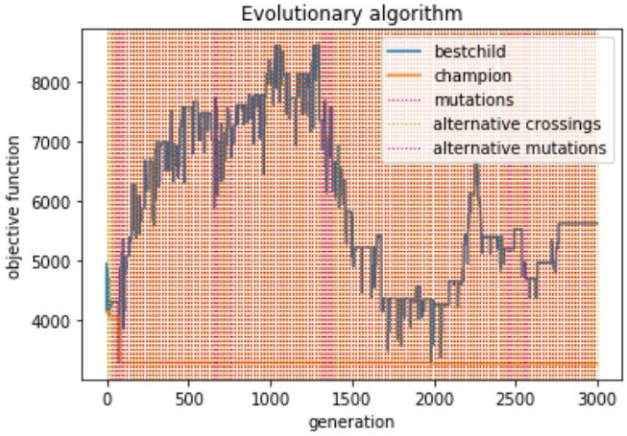
Rysunek 21



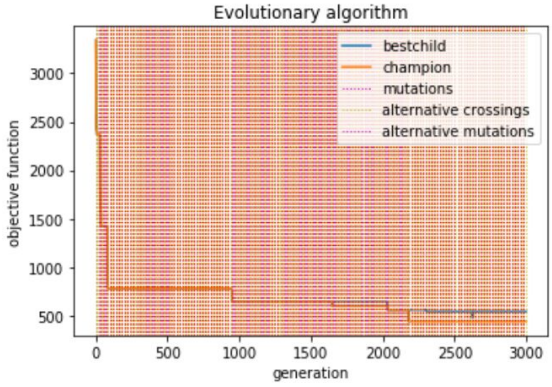
Rysunek 22

Tabela 7: IT2

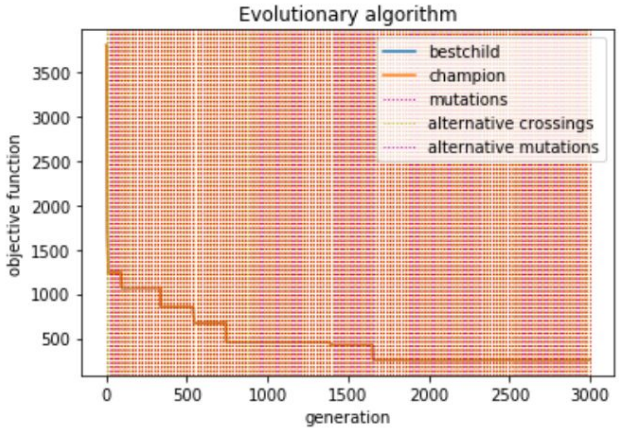
lp.	m	$f_{c_{pocz}}(x)$	$\overline{f_c(x)}_5$	σ_{f_c}	$min_i(f_{c_i}(x))$	$max_i(f_{c_i}(x))$	$I_{śr}$	I_{stop}
1.	1	4140	2830	880.17	1960	4140	1028	3000
2.	4	4140	246	160.09	70	440	1929	3000
3.	8	4140	208	197.66	0	410	1502	3000



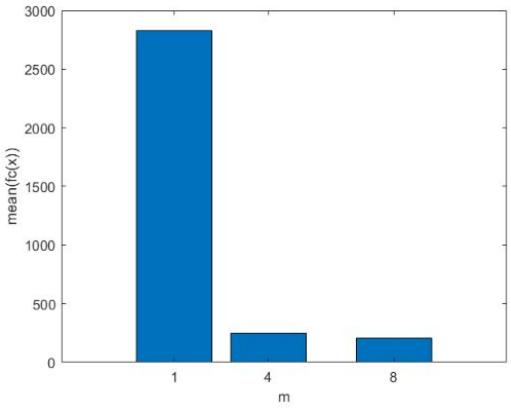
Rysunek 23



Rysunek 24



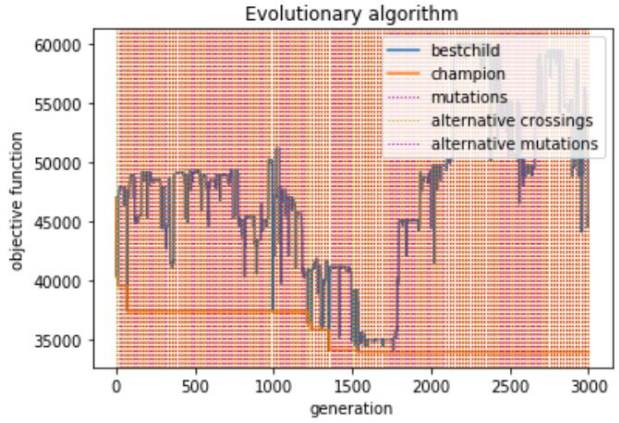
Rysunek 25



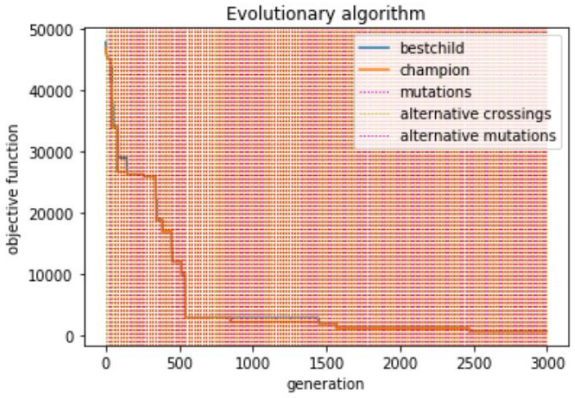
Rysunek 26

Tabela 8: IT3

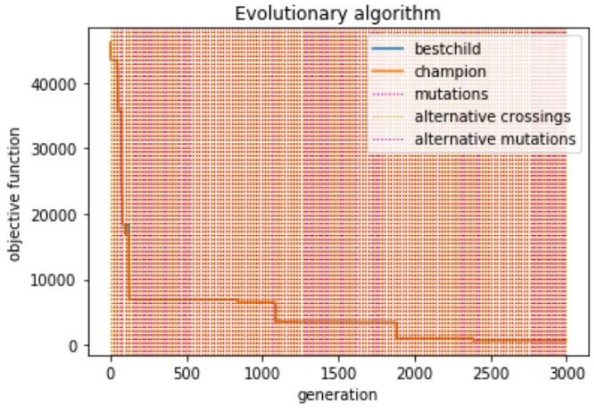
lp.	m	$f_{c_{pocz}}(x)$	$\overline{f_c(x)}_5$	σ_{f_c}	$min_i(f_{c_i}(x))$	$max_i(f_{c_i}(x))$	$I_{śr}$	I_{stop}
1.	1	46940	31152	3732.55	26510	33980	2131	3000
2.	4	46940	807	792.37	20	1780	1897	3000
3.	8	46940	156	332.16	0	750	1636	3000



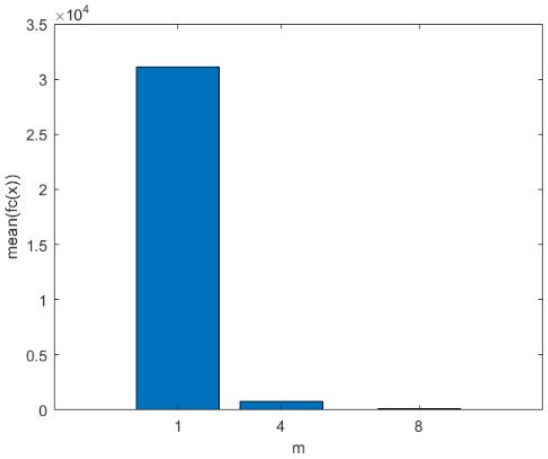
Rysunek 27



Rysunek 28



Rysunek 29



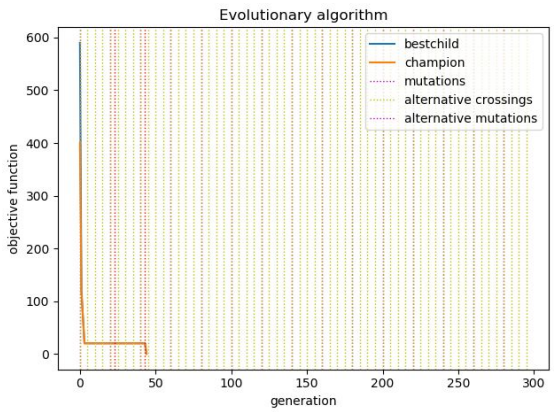
Rysunek 30

Wnioski:
Określenie wielkości grup w turniejach wprowadza określenie elitarności algorytmu.
Dla minimalnej wartości równej 1, kolejne osobniki wybierane są całkowicie losowo, przez co najlepsze geny są szybko tracone i mało wykorzystywane. Tę losowość wyraźnie widać na wykresach. Natomiast dla maksymalnej wartości równej ilości wyselekcjonowanych osobników, algorytm wybiera tylko najlepsze osobniki (na wykresach widać, że w żadnym momencie nie pozbywamy się najlepszych genów). W naszych przypadkach takie działanie sprawdziło się, natomiast dla trudniejszych przypadków, algorytm zatrzymałby się na minimach lokalnych.

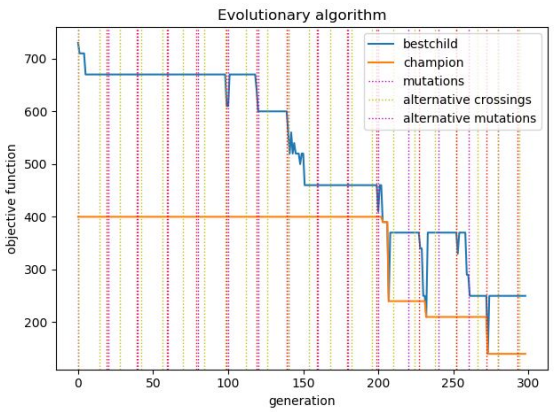
5. Test wpływu parametru f_{CA} – częstotliwość alternatywnego krzyżowania

Tabela 9: IT1

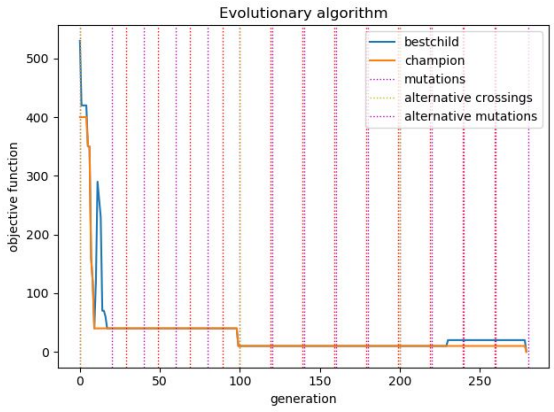
lp.	f_{CA}	$f_{c_{pocz}}(x)$	$\overline{f_c(x)}_5$	σ_{f_c}	$min_i(f_{c_i}(x))$	$max_i(f_{c_i}(x))$	$I_{śr}$
1.	10	400	68	62.21	0	120	147
2.	14	400	60	73.48	0	140	83
3.	100	400	6	13.42	0	30	191



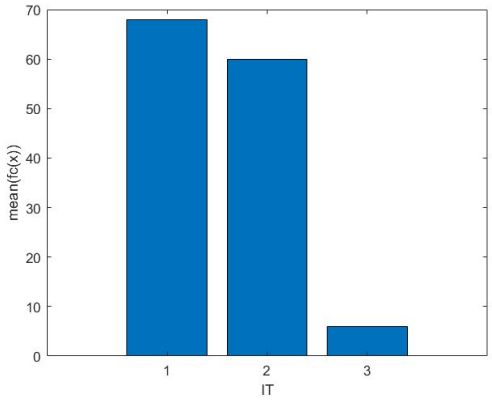
Rysunek 31



Rysunek 32



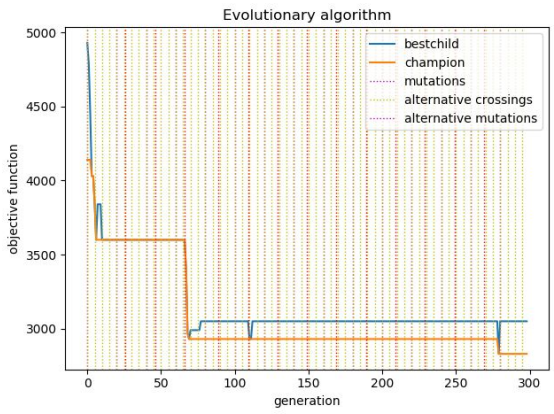
Rysunek 33



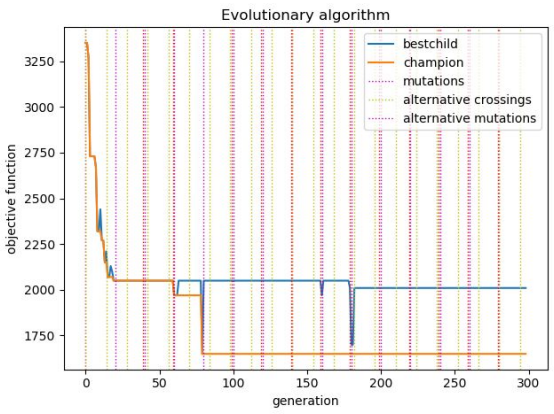
Rysunek 34

Tabela 10: IT2

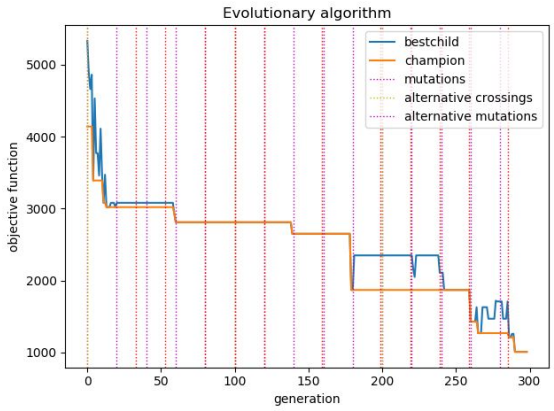
lp.	f_{CA}	$f_{c_{pocz}}(x)$	$\overline{f_c(x)}_5$	σ_{f_c}	$min_i(f_{c_i}(x))$	$max_i(f_{c_i}(x))$	$I_{śr}$
1.	10	4140	2032	777.80	1240	2830	193
2.	14	4140	900	498.70	450	1650	184
3.	100	4140	1194	641.86	280	1840	221



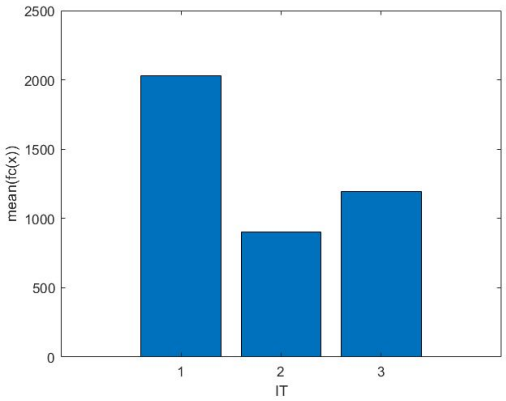
Rysunek 35



Rysunek 36



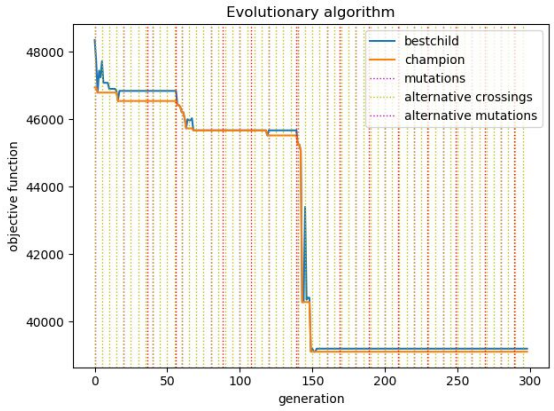
Rysunek 37



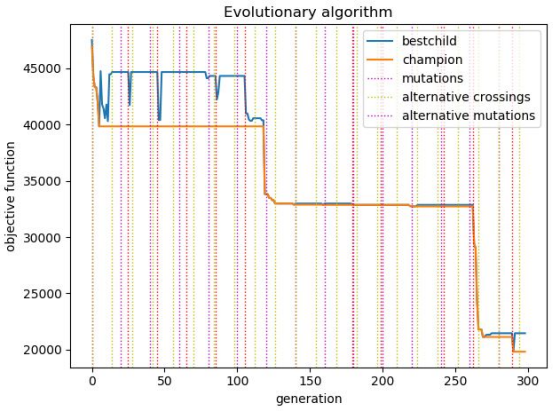
Rysunek 38

Tabela 11: IT3

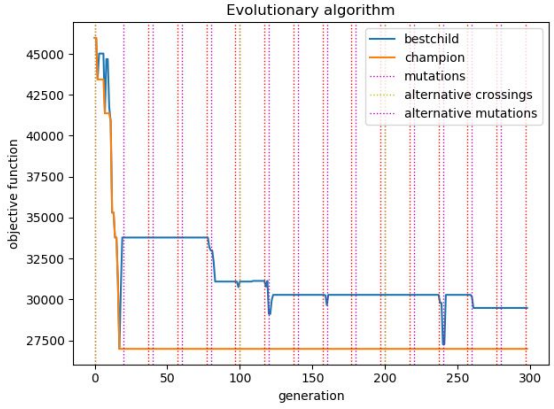
lp.	f_{CA}	$f_{c_{pocz}}(x)$	$\overline{f_c(x)}_5$	σ_{f_c}	$min_i(f_{c_i}(x))$	$max_i(f_{c_i}(x))$	$I_{śr}$
1.	10	46940	29512	11144.20	11750	39110	233
2.	20	46940	20176	8289.95	10140	30570	232
3.	100	46940	18746	10814.74	5220	28240	219



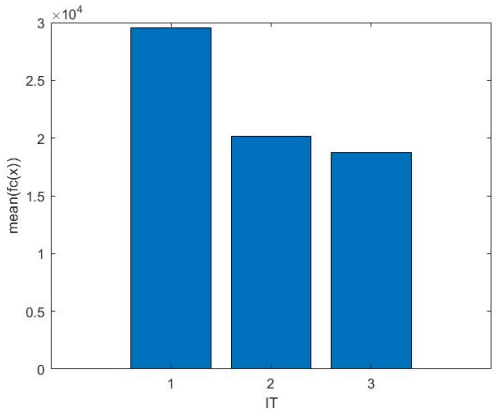
Rysunek 39



Rysunek 40



Rysunek 41



Rysunek 42

Wnioski:
Testy zgodnie pokazują, że poza wyjątkami krzyżowanie w wersji alternatywnej nie jest operacją pożądaną. Niższa wartość parametru, a zatem częstsze wykonywanie, powoduje, że wartości funkcji celu wolniej zdążają do zera. Prawdopodobnie jest to spowodowane powtarzalnością nowych rozwiązań w zbiorze wynikowym i w efekcie zaniżonych możliwościach ich ewolucji.

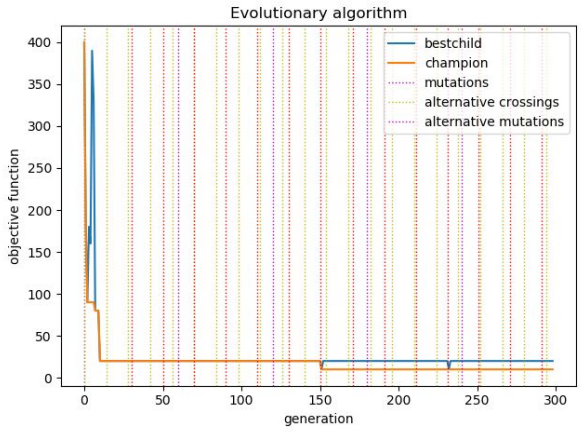
6. Test wpływu parametru amountOfAttemptsSick – ilość prób utworzenia nowego rozwiązania potomnego w krzyżowaniu alternatywnym

   Badanie parametru nie przynosi użytecznych informacji, ponieważ alternatywne krzyżowanie napotyka na problem nieskończonej pętli w sposób nieregularny i nieprzewidywalny. Dopasowanie jego wartości należy dobrać empirycznie zgodnie z intuicyjnym ryzykiem. Zmniejszenie wartości parametru powoduje zmniejszone użycie krzyżowania alternatywnego, zatem w konsekwencji również częstotliwości pojawiania się bezproduktywnych iteracji oraz prawdopodobieństwa generowanych przez ten sposób krzyżowania powtórzeń rozwiązań w zbiorze wynikowym.

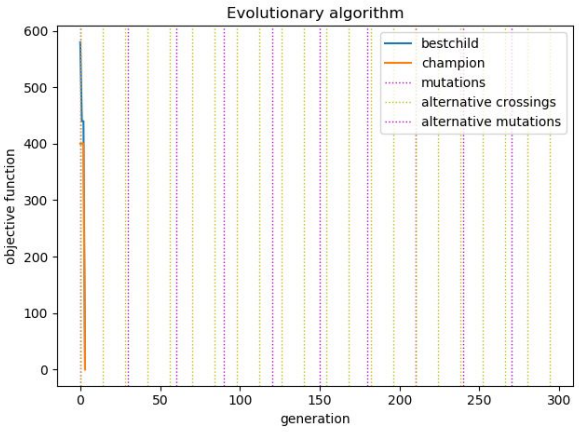
7. Test wpływu parametru f_M – częstotliwość mutowania

Tabela 12: IT1

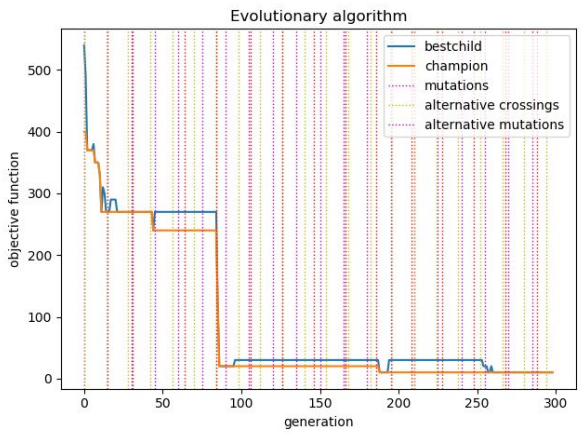
lp.	f_M	$f_{c_{pocz}}(x)$	$\overline{f_c(x)}_5$	σ_{f_c}	$min_i(f_{c_i}(x))$	$max_i(f_{c_i}(x))$	$I_{śr}$	I_{stop}	$Ilość \frac{I_{stop}}{f_M}$
1.	15	400	8	8.37	0	20	172	300	20
2.	30	400	4	5.48	0	10	106	300	10
3.	60	400	30	46.37	0	110	150	300	5



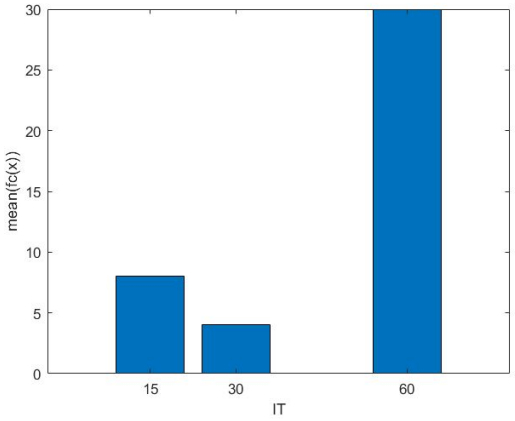
Rysunek 43: $f_M = 60$ *przykładowy wykres*



Rysunek 44: $f_M = 30$ *przykładowy wykres*



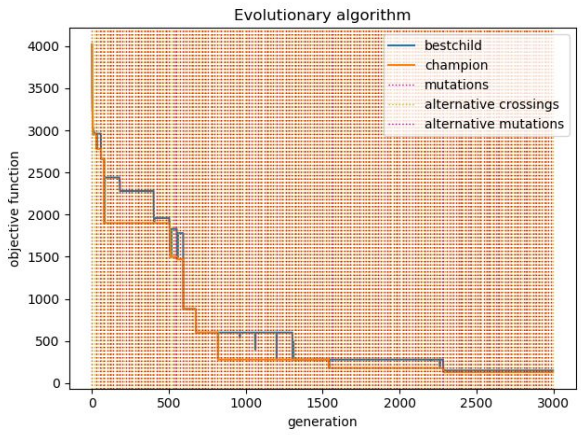
Rysunek 45: $f_M = 15$ *przykładowy wykres*



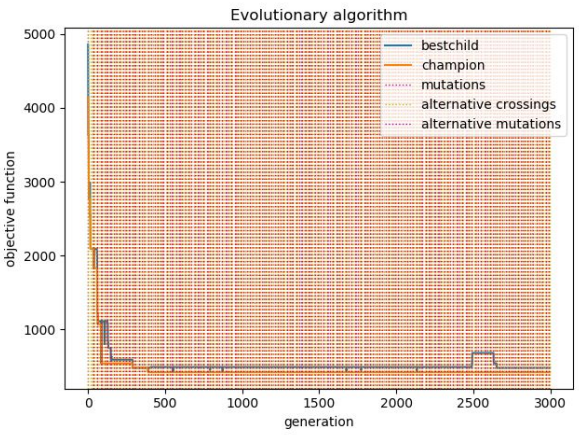
Rysunek 46: Średnie wartości MATLAB

Tabela 13: IT2

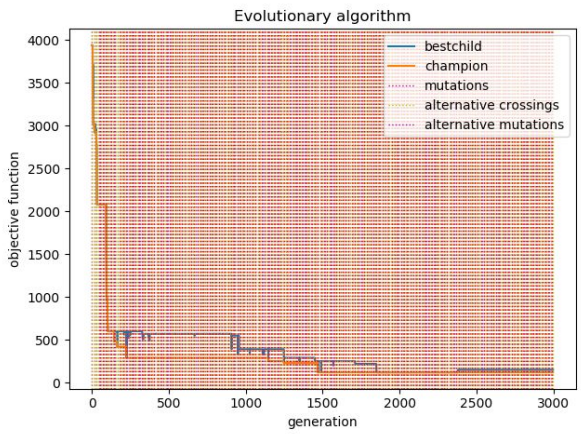
lp.	f_M	$f_{c_{pocz}}(x)$	$\overline{f_c(x)}_5$	σ_{f_c}	$min_i(f_{c_i}(x))$	$max_i(f_{c_i}(x))$	$I_{śr}$	I_{stop}	$Ilość \frac{I_{stop}}{f_M}$
1.	15	4140	96	51.28	30	160	1790	3000	200
2.	30	4140	264	255.79	0	630	1859	3000	100
3.	60	4140	298	110.32	130	400	1372	3000	50



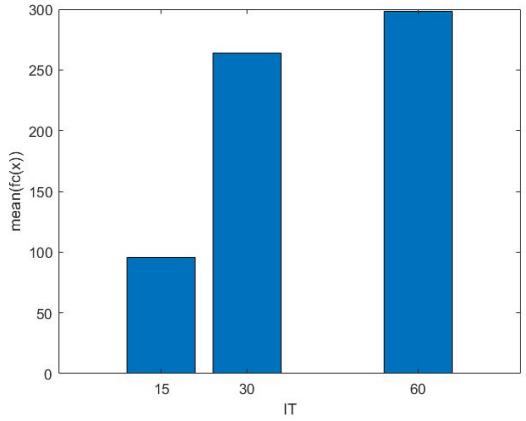
Rysunek 47: $f_M = 60$ *przykładowy wykres*



Rysunek 48: $f_M = 30$ *przykładowy wykres*



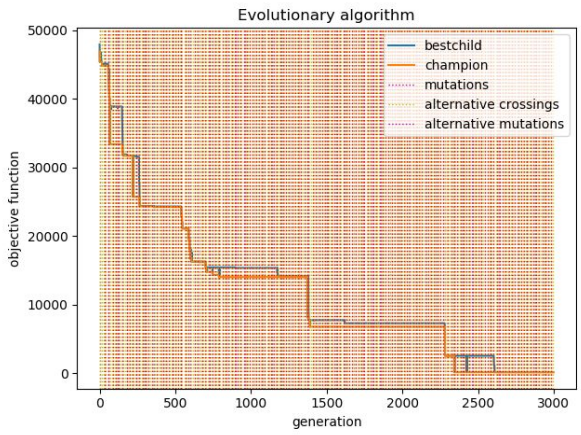
Rysunek 49: $f_M = 15$ *przykładowy wykres*



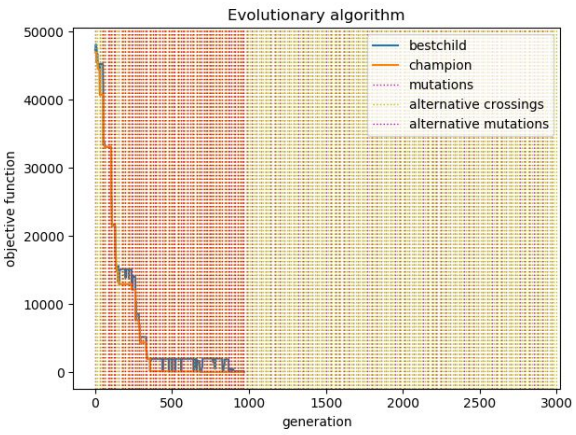
Rysunek 50: Średnie wartości MATLAB

Tabela 14: IT3

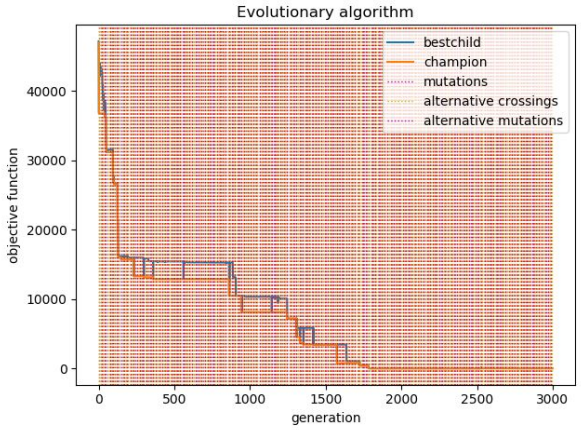
lp.	f_M	$f_{c_{pocz}}(x)$	$\overline{f_c(x)_5}$	σ_{f_c}	$min_i(f_{c_i}(x))$	$max_i(f_{c_i}(x))$	$I_{śr}$	I_{stop}	Ilość $\frac{I_{stop}}{f_M}$
1.	15	46940	262	365.40	10	890	2116	3000	200
2.	30	46940	978	1506.94	0	3470	1835	3000	100
3.	60	46940	570	1071.07	0	2480	2361	3000	50



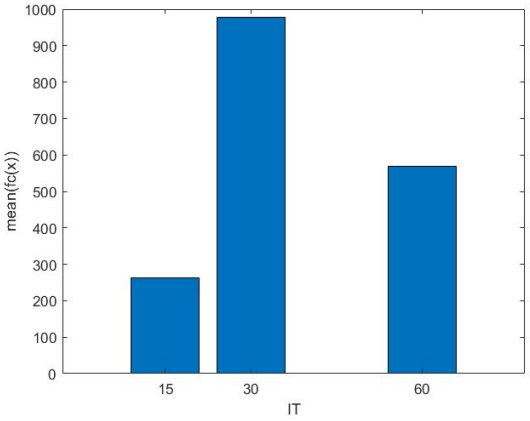
Rysunek 51: $f_M = 60$ *przykładowy wykres*



Rysunek 52: $f_M = 30$ *przykładowy wykres*



Rysunek 53: $f_M = 15$ *przykładowy wykres*



Rysunek 54: Średnie wartości MATLAB

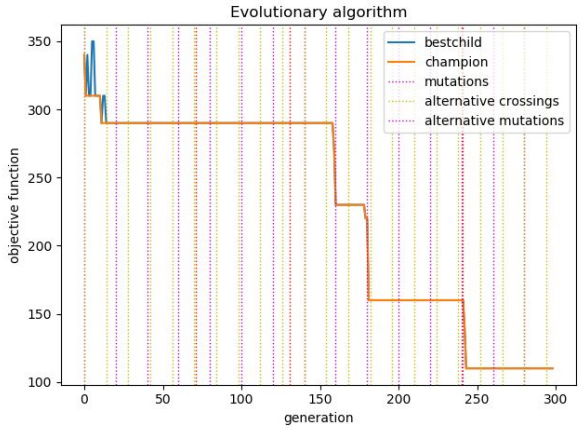
Wnioski:
Wyciągając wnioski z zamieszczonych zdjęć powyżej, zauważalne jest, że z reguły im więcej mutacji tym ostateczna minimalizacja funkcji kryterialnej jest lepsza. Dodatkowo istotną rzeczą jest, że szybkość znajdowania najlepszego rozwiązania znajduje się na bardzo podobnym poziomie. Oznacza to, że bez względu na ilość przeprowadzonych mutacji optymalny rezultat jest odnajdywany w podobnym czasie.

8. Test wpływu parametru f_{MA} – częstotliwość mutowania alternatywnego

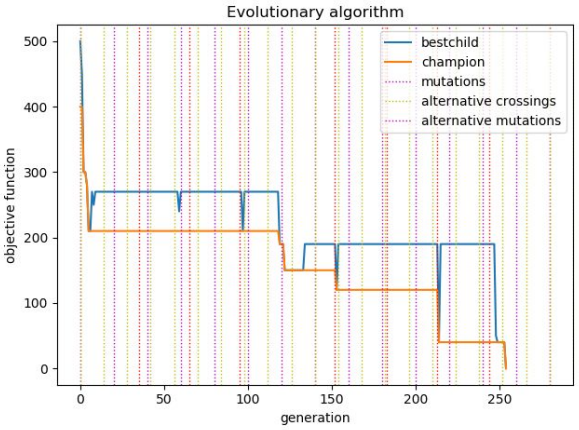
Tabela 15: IT1

lp.	f_{MA}	$f_{c_{pocz}}(x)$	$\overline{f_c(x)}_5$	σ_{f_c}	$min_i(f_{c_i}(x))$	$max_i(f_{c_i}(x))$	$I_{śr}$	I_{stop}	Ilość $\frac{I_{stop}}{f_M}$
1.	15	400	2	4.47	0	10	82	300	20?
2.	30	400	12	21.68	0	50	208	300	10?
3.	60	400	24	48.27	0	110	178	300	5?

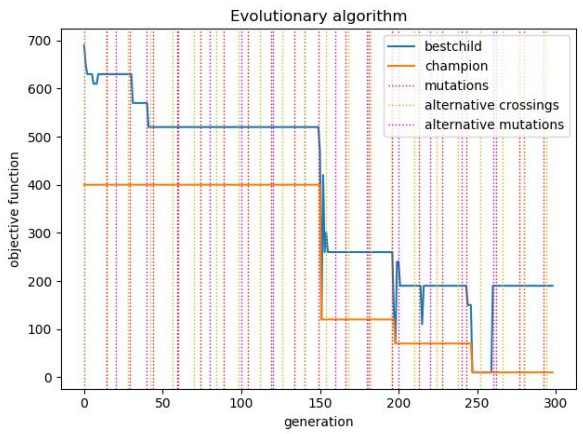
? – największa możliwa liczba wykonań



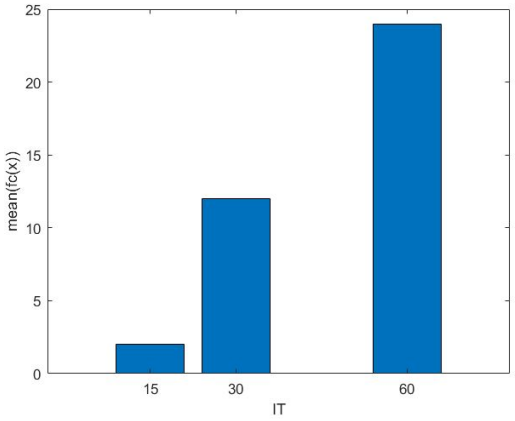
Rysunek 55: $f_{MA} = 60$ *przykładowy wykres*



Rysunek 56: $f_{MA} = 30$ *przykładowy wykres*



Rysunek 57: $f_{MA} = 15$ *przykładowy wykres*

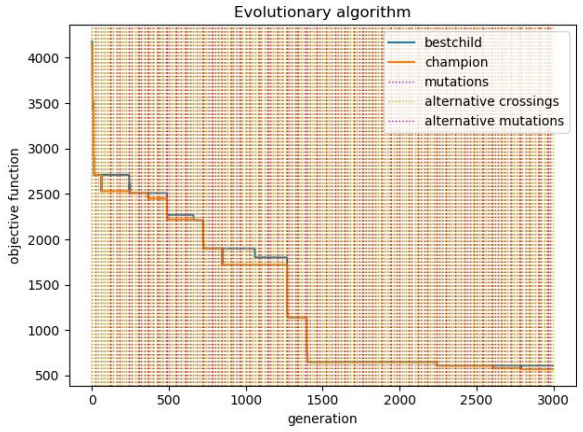


Rysunek 58: Średnie wartości MATLAB

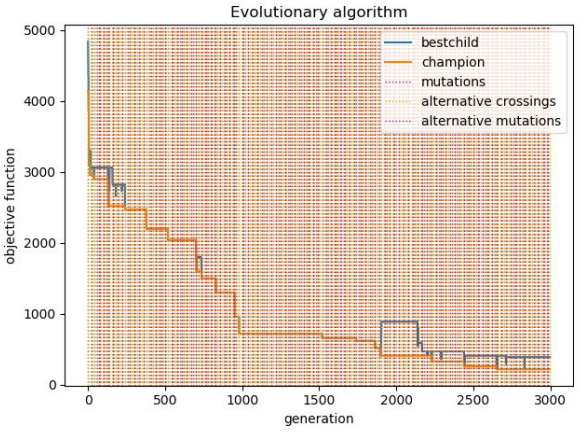
Tabela 16: IT2

lp.	f_{MA}	$f_{c_{pocz}}(x)$	$\overline{f_c(x)}_5$	σ_{f_c}	$min_i(f_{c_i}(x))$	$max_i(f_{c_i}(x))$	$I_{śr}$	I_{stop}	Ilość $\frac{I_{stop}}{f_M}$
1.	15	4140	170	107.94	30	280	902	3000	200?
2.	30	4140	390	114.89	220	540	2110	3000	100?
3.	60	4140	560	516.09	50	1420	2345	3000	50?

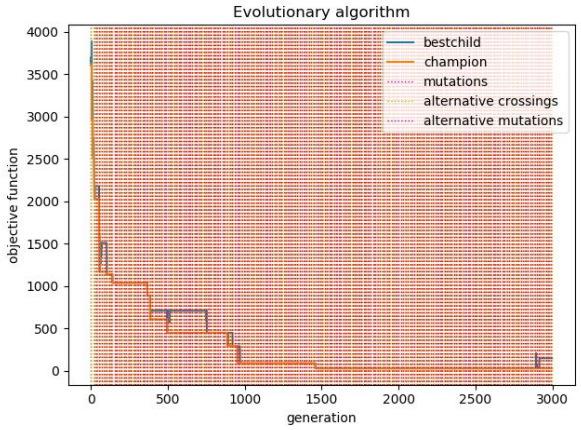
? – największa możliwa liczba wykonań



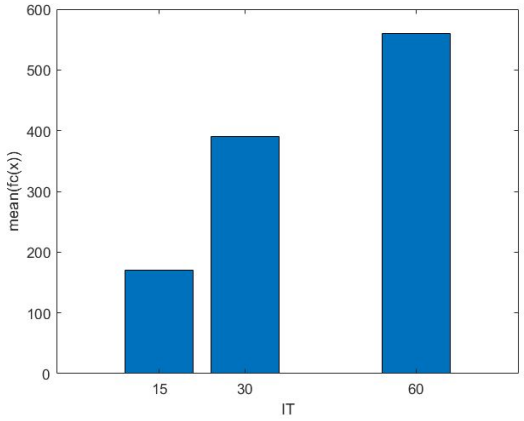
Rysunek 59: $f_{MA} = 60$ *przykładowy wykres*



Rysunek 60: $f_{MA} = 30$ *przykładowy wykres*



Rysunek 61: $f_{MA} = 15$ *przykładowy wykres*

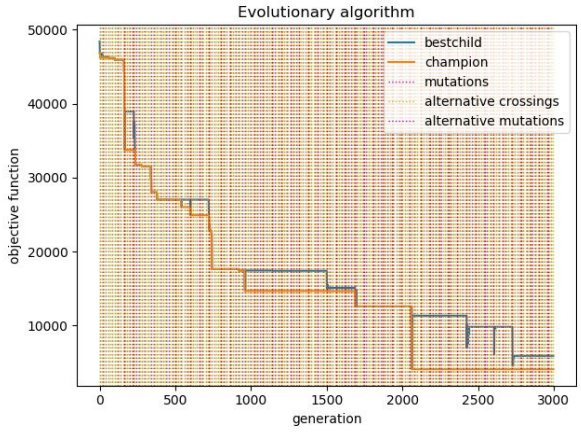


Rysunek 62: Średnie wartości MATLAB

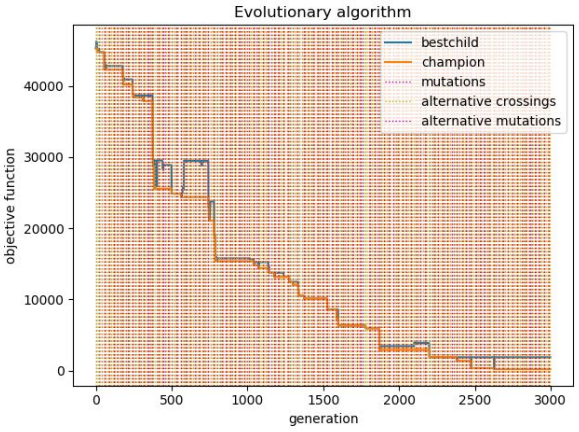
Tabela 17: IT3

lp.	f_{MA}	$f_{c_{pocz}}(x)$	$\overline{f_c(x)}_5$	σ_{f_c}	$min_i(f_{c_i}(x))$	$max_i(f_{c_i}(x))$	$I_{śr}$	I_{stop}	Ilość $\frac{I_{stop}}{f_M}$
1.	15	46940	406	548.84	0	1210	1768	3000	200?
2.	30	46940	76	95.29	0	180	2157	3000	100?
3.	60	46940	6746	6103.58	10	16400	2626	3000	50?

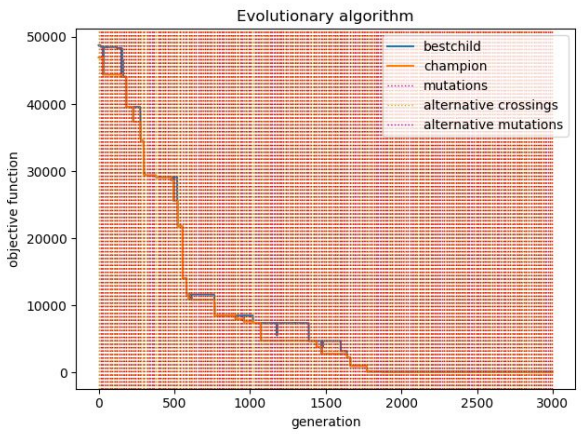
? – największa możliwa liczba wykonań



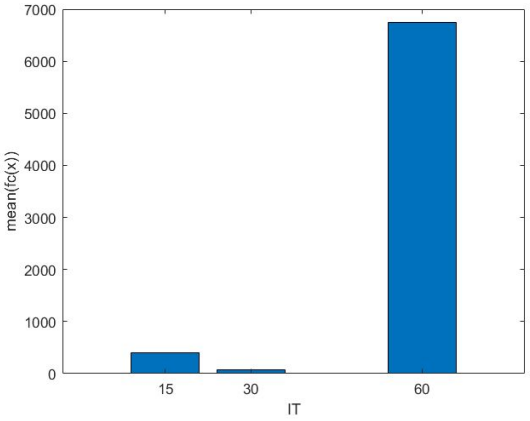
Rysunek 63: $f_{MA} = 60$ *przykładowy wykres*



Rysunek 64: $f_{MA} = 30$ *przykładowy wykres*



Rysunek 65: $f_{MA} = 15$ *przykładowy wykres*



Rysunek 66: Średnie wartości MATLAB

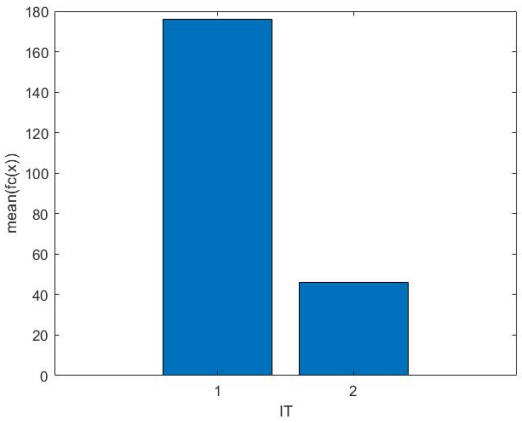
Wnioski:
Tak jak w standardowej mutacji, widoczne jest, że teoretycznie im alternatywne mutowanie występuje częściej, tym zachodzi lepsza minimalizacja funkcji kryterialnej. Różnicą jest to, że częstość występowania alternatywnej mutacji wpływa na szybkość znajdowania optymalnego rozwiązania, to znaczy im jest jej więcej tym najlepszy wynik osiągany jest sprawniej.

9. Test wpływu parametru offset – wielkość przesunięcia w rotateMutation

Tabela 18

lp.	$offset$	$f_{c_{pocz}}(x)$	$\overline{f_c(x)}_5$	σ_{f_c}	$min_i(f_{c_i}(x))$	$max_i(f_{c_i}(x))$	$I_{śr}$	I_{stop}
1.	losowe	400	176	185.55	0	400	54	300
2.	stała wartość	400	46	97.37	0	220	146	300

stała wartość – cecha połowy długości pojedynczego rozwiązania

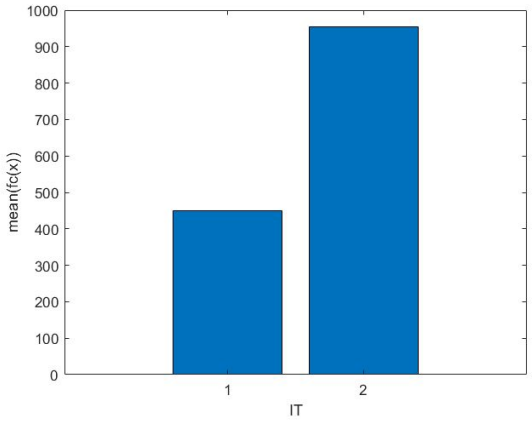


Rysunek 67: Średnie wartości MATLAB

Tabela 19

lp.	$offset$	$f_{c_{pocz}}(x)$	$\overline{f_c(x)}_5$	σ_{f_c}	$min_i(f_{c_i}(x))$	$max_i(f_{c_i}(x))$	$I_{śr}$	I_{stop}
1.	losowe	4140	450	219.43	230	770	1992	3000
2.	stała wartość	4140	954	331.93	500	1380	1999	3000

stała wartość – cecha połowy długości pojedynczego rozwiązania

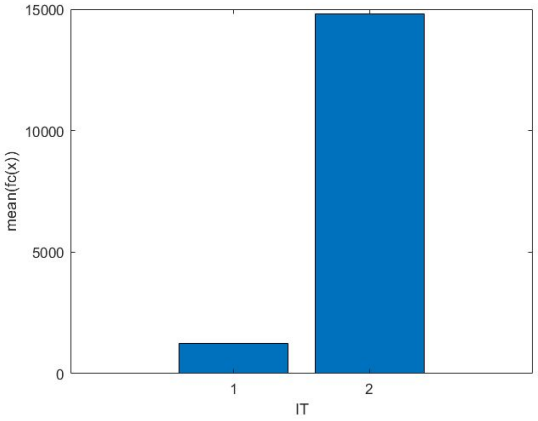


Rysunek 68: Średnie wartości MATLAB

Tabela 20

lp.	$offset$	$f_{c_{pocz}}(x)$	$\overline{f_c(x)}_5$	σ_{f_c}	$min_i(f_{c_i}(x))$	$max_i(f_{c_i}(x))$	$I_{śr}$	I_{stop}
1.	losowe	46940	1260	1559.25	60	3570	2273	3000
2.	stała wartość	46940	14796	7504.50	4330	23390	2230	3000

stała wartość – cecha połowy długości pojedynczego rozwiązania



Rysunek 69: Średnie wartości MATLAB

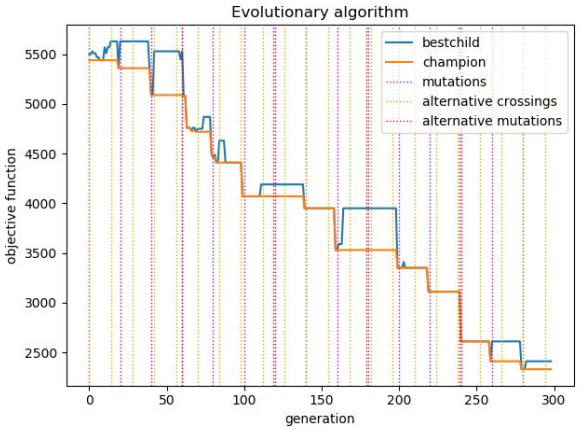
Wnioski:
Z powyższych obrazów, jasnym faktem jest, że ogólnie rzecz biorąc losowanie przesunięcia daje lepszy rezultat niż wartość stała. Co więcej, wartość tego parametru nie ma wpływu na szybkość znajdowania najlepszego rezultatu.

10. Test wpływu parametru I – ilość iteracji

Parametr był testowany współbieżnie z pozostałymi, zatem rozsądnym wydaje się wysnucie wniosku o proporcjonalności poprawy funkcji celu względem parametru I. Ma to uzasadnienie we właściwości algorytmu, który ma jedynie możliwość poprawy wyniku, a nie pogorszenia go.

11. Testy przypadków złośliwych

1. Wartości początkowe nie posiadają odważników na pożądanej stronie (wszystkie rozwiązania początkowe pogłębiają problem):

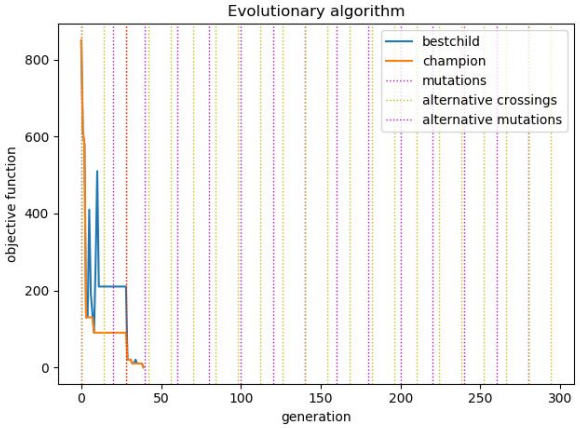


Rysunek 70:

$f_{c_{pocz}}(x) = 5440$ $\qquad$ $f_{c_{końc}}(x) = 2330$

Algorytm zbiega liniowo do coraz mniejszych wartości, ale na znalezienie dobrego rozwiązania potrzebuje dużej ilości iteracji.

2. Odważniki są rozmieszczone symetrycznie względem punktu podparcia (żadne z rozwiązań początkowych nie wpływa na funkcję celu):

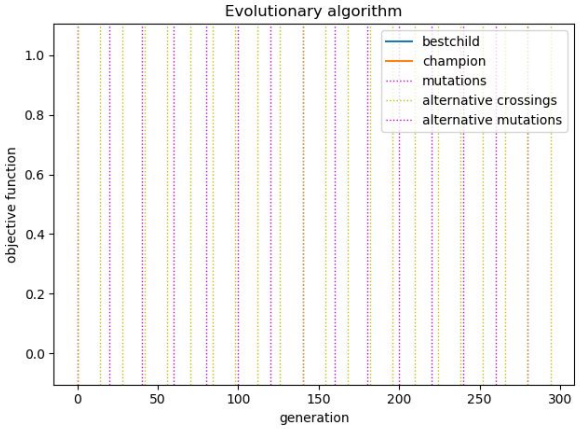


Rysunek 71:

$f_{c_{pocz}}(x) = 2600$ $\qquad$ $f_{c_{końc}}(x) = 0$

Algorytm już po 39 iteracjach jest w stanie znaleźć rozwiązanie optymalne.

3. We wszystkich rozwiązaniach użyte odważniki są zbyt ciężkie:



Rysunek 72:

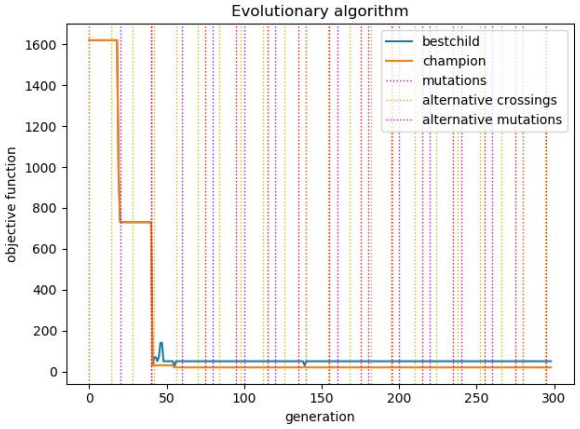
$f_{c_{pocz}}(x) = 0$ $\qquad$ $f_{c_{końc}}(x) = 0$

Algorytm uznaje, że rozwiązanie optymalne jest znajdowane natychmiast już przy pierwszej iteracji.

4. We wszystkich rozwiązaniach użyte odważniki są zbyt lekkie:

W tym przypadku algorytm zawodzi, wyrzucając błąd, ponieważ jedna z funkcji z niepowodzeniem próbuje przypisać element do struktury danych o innym typie.

5. Wszystkie rozwiązania początkowe są identyczne:



Rysunek 73:

$$f_{c_{pocz}}(x) = 1620 \qquad f_{c_{końc}}(x) = 20$$

Algorytm już po 55 iteracjach jest w stanie znaleźć dobre rozwiązanie.

6. Wszystkie dostępne odważniki zostały wykorzystane (rozwiązania początkowe mają trudności w krzyżowaniu się):

W tym przypadku algorytm zawodzi, wyrzucając błąd, ponieważ jedna z funkcji z niepowodzeniem próbuje przypisać element do struktury danych o innym typie.

5 Podsumowanie

5.1 Wnioski

- Algorytm dobrze sobie radzi ze znajdowaniem optymalnego ułożenia ciężarków na dźwigni. Przy dobrze dobranych parametrach potrafi bardzo dobrze wyważyć belkę.
- Algorytm jest uniwersalny – zmieniając funkcję celu można również znajdować inne ustawienia ciężarków dla postawionych problemów.

5.2 Stwierdzone problemy

- Alternatywne krzyżowanie może napotkać na problem nieskończonej pętli w momencie, gdy rozwiązania pierwotne są tak skonstruowane, że każde rozwiązanie potomne jest niedopuszczalne z perspektywy dostępnych ciężarków.
- Niektóre tzw. „złośliwe” przypadki testowe (1. i 6. przypadek) są w stanie spowodować, że algorytm napotka na błąd przypisania elementu do złej struktury danych, przez co program przestaje działać i nie zwraca żadnego wyniku.

5.3 Kierunki dalszego rozwoju

W dalszej kolejności można byłoby zainteresować się rozwinięciem problemu na drugi wymiar. W takiej sytuacji dźwignia staje się krążkiem (wałem obrotowym), którego wyważanie stanowi rzeczywisty problem inżynieryjny. Niewyrównoważone wały obrotowe są powodem przeciążeń, drgań i w konsekwencji szybszego tempa zużywania się elementów mechanicznych takich jak łożyska, prowadnice itp.

6 Podział Pracy

Tabela 21

Etap	Rafał Gembalczyk	Mariusz Grabowski	Michał Grenda
Model zagadnienia	100% sformułowanie problemu, postać funkcji celu, warunki ograniczające, postać rozwiązania	0%	0%
Algorytm opracowanie	20% alternatywne krzyżowanie	50% selekcja, standardowe krzyżowanie	30% mutacje
Implementacja aplikacji	genRandWeighs(), genRandSol(), objectiveFunc(), sortBestSol(), sick(), alternativeCrossing(), cannotSolve(), wyjątki, wykresy, testy jednostkowe	Select(), Crossing(), główna pętla	mutate, rotateMutate, markMutation, loadParameters (wczytywanie danych z plików), getTwoRandomNumbers, debuggowanie (naprawa bieżących błędów), główny program
Testy	33,(3)% przypadki „złośliwe” parametry algorytmu $f_{CA}\ I$, $amountOfAttemptsSick$	33,(3)% generacja instancji testowych, test z wartościami domyślnymi, parametry algorytmu w, m	33,(3)% parametry algorytmu $f_M, f_{MA}, offset$
Dokumentacja	2.2 – krzyżowanie alternatywne, sekcje 2.3, 3.3, 4.1,	okładka, 1.2, 2.1, 2.2 – selekcja, krzyżowanie	1 (opis), 2.2 – operatory mutowania,

	4.2, 4.3, 5.2, 5.3	standardowe, 3.1, 3.2, 4.1, 5.1, podział dokumentu na rozdziały	3.3