Sprawozdanie 1
Algorytmy genetyczne

1. Opis problemu TTP

Problem Mobilnego Złodzieja składa się z 2 problemów:

Problemu Plecakowego – KNP

KNP składa się ze zbioru n przedmiotów oraz plecaka. Przedmioty opisane są wagą, a plecak ma swoją pojemność. Problem polega na wybraniu jak zestawu przedmiotów do plecaka.

Problemu komiwojażera – TSP

TSP składa się ze zbioru m miast oraz macierzy odległości pomiędzy nimi (w przypadku naszego zadania macierz jest symetryczna). Problem polega na znalezieniu najkrótszej drogi pozwalającej odwiedzić wszystkie miasta, przy czym każde z miast możemy odwiedzić jedynie raz.

Celem rozwiązania TTP jest więc minimalizacja trasy przebytej przez złodzieja i maksymalizacja wartości zabranych z miast przedmiotów.

2. Rozwiązanie Problemu

Rozwiązanie składa się na podzielenie głównego problemu na subproblemy:

Subproblem KNP został rozwiązany dzięki użyciu algorytmu zachłannego. Wybrana przeze mnie strategia polega na zabraniu z miasta przedmiotu o najlepszym stosunku wartości do wagi, przy czym waga przedmiotu nie może być wyższa niż pozostała wytrzymałość plecaka.

Algorytm składa się z następujących kroków:

1. Sortowanie przedmiotów w mieście wg stosunku wartości do wagi
2. Pobranie z listy pierwszego przedmiotu, który zmieści się do plecaka

Subproblem TSP

Został rozwiązany przy użyciu algorytmu genetycznego.

3. Algorytm genetyczny

Parametry podstawowe:

POPULATION_SIZE – rozmiar populacji
POPULATIONS – ilość stworzonych populacji
PX – prawdopodobieństwo krzyżowania
PM – prawdopodobieństwo mutacji

Działanie algorytmu:

1. Utworzenie populacji początkowej złożonej z odpowiedniej liczby (*POPULATION_SIZE)* losowych osobników.
2. Ocenienie jakości osobników:
   - przejście miast zgodnie z kolejnością występującą w genotypie osobnika
   - uruchomienie algorytmu zachłannego na wszystkich etapach drogi dla każdego osobnika
   - obliczenie funkcji celu dla każdego osobnika ze wzoru:

   f(x) = wartość plecaka – czas przebycia całej drogi * rentng_ratio
3. Sprawdzenie, czy warunek zatrzymania algorytmu genetycznego został spełniony (czy stworzona została określona liczba populacji *POPULATIONS*)
4. Selekcja najlepszych osobników
5. Krzyżowanie osobników
6. Mutacja
7. Powrót do punktu 2.

Metody selekcji:

- Turniej

  Losowy wybór uczestników turnieju i wyłonienie zwycięzcy na podstawie najlepszej (największej) wartości funkcji celu.

  Dodatkowe parametry:
  TOURNAMENT_SIZE – wielkość turnieju
- Ruletka

  Przydzielenie każdemu osobnikowi prawdopodobieństwa z jakim zostanie wylosowany na podstawie wartości funkcji celu – im wyższa wartość funkcji celu, tym wyższe prawdopodobieństwo wybrania danego osobnika.
- Ranking

  Wybranie określonej liczby najlepszych osobników według ich wartości funkcji celu.

  Dodatkowe parametry:
  RANKING_SIZE – ilość osobników brana pod uwagę przy tworzeniu rankingu

*w każdym z poniższych wykresów:
serie 1 oznacza nr populacji (kolor niebieski)
serie 2 oznacza najlepszy wynik (kolor czerwony)
serie 3 oznacza średni wynik (kolor szary)
serie 4 oznacza najgorszy wynik (kolor żółty)

4. Badanie wpływu doboru parametrów i metod selekcji na działanie algorytmu

Początkowe parametry algorytmu:
POPULATION_SIZE: 100
POPULATIONS: 100
PX – 0.7
PM –0.1
TOURNAMENT_SIZE : 5
TOURNAMENT_WINNERS : 1

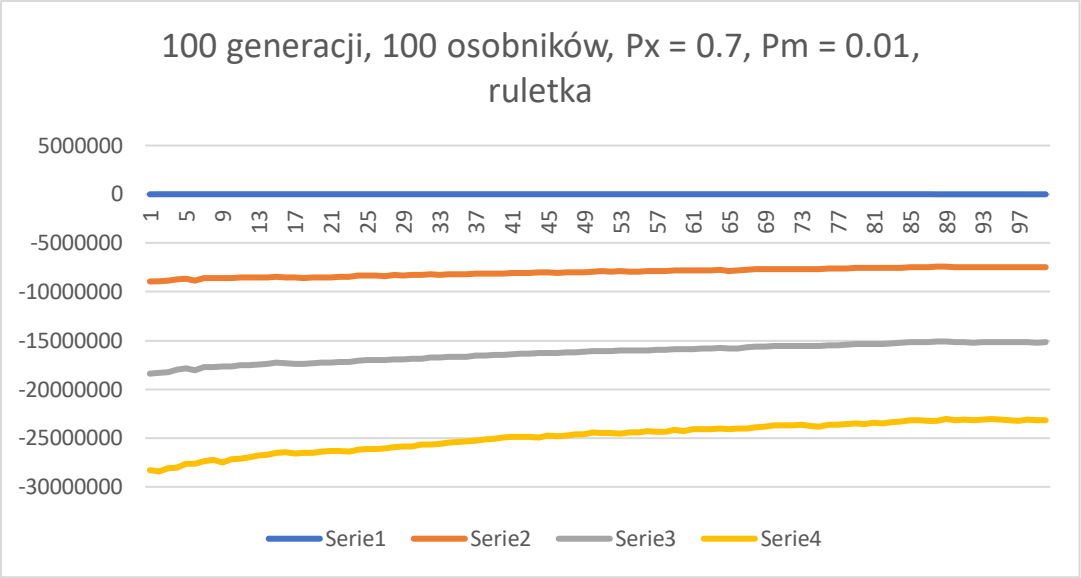
Wybór metody selekcji:

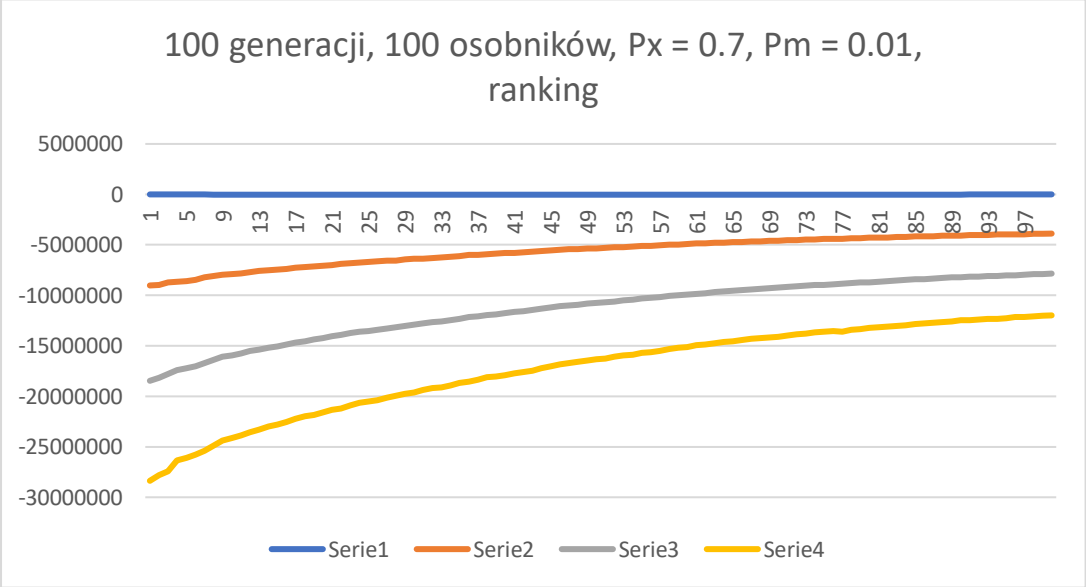
Pierwszym obiektem badania był wpływ doboru metody selekcji na działanie algorytmu. Zbadane zostały następujące metody:

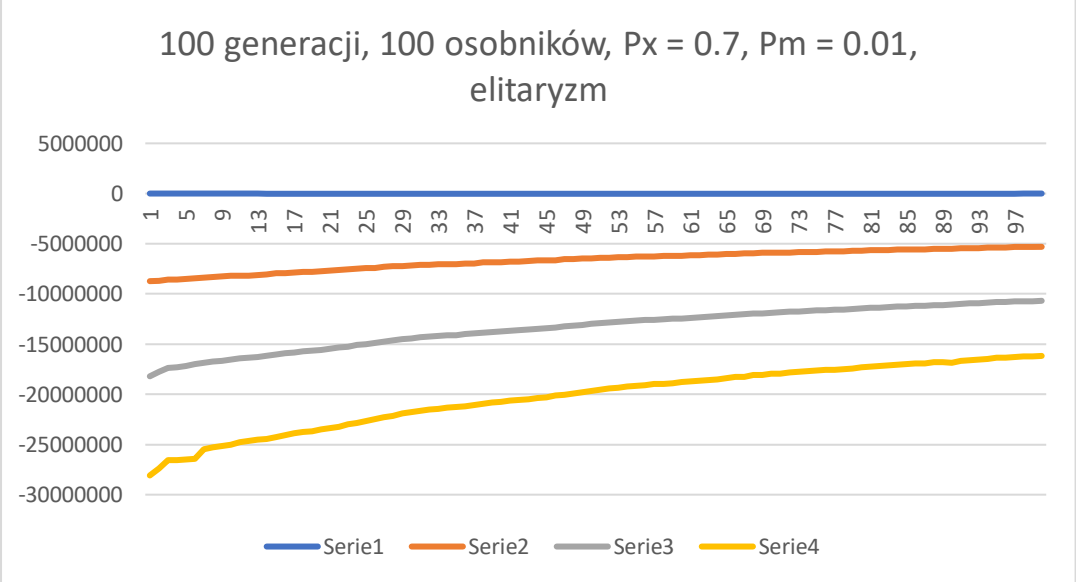
- Ruletka
- Ranking
- Elitaryzm
- Turniej dla parametru *TOURNAMENT_SIZE* przyjmującego wartości: 5,10, 25 oraz skrajne wartości 1 i 100;

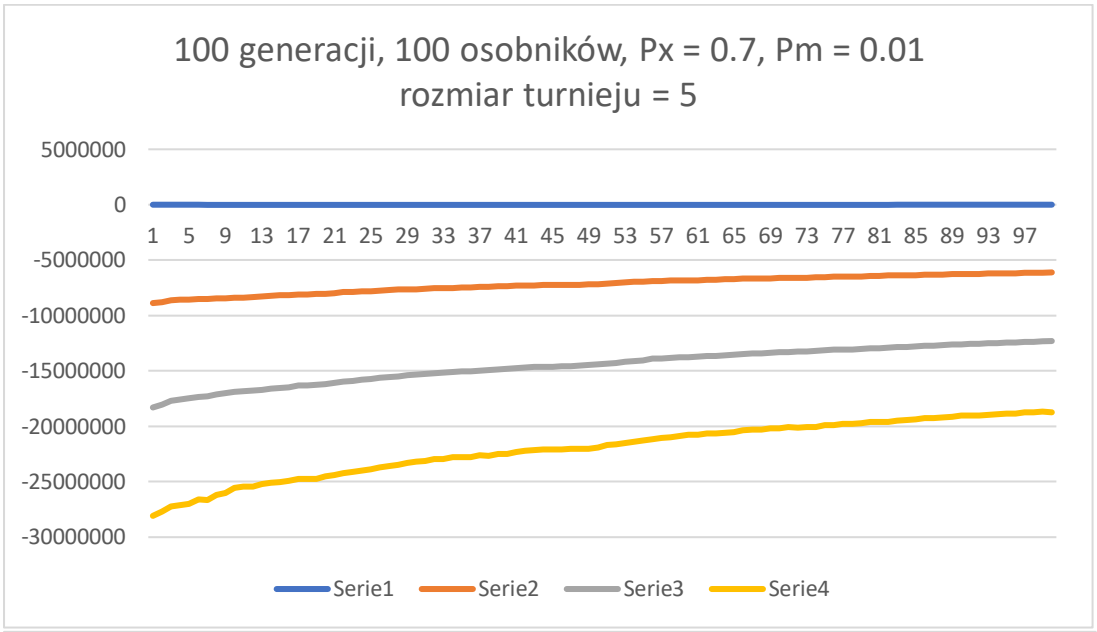
Jak widać to na poniższych wykresach, najwyższymi wynikami skutkowała metoda rankingowa, następnie elitaryzm. Te metody nie odzwierciedlają jednak rzeczywistej selekcji naturalnej, więc wybór podstawowej selekcji działającej w algorytmie genetycznym został wybrany turniej. Turniej o parametrze *TOURNAMENT_SIZE* = 25 skutkował wyraźnie wyższymi wynikami, niż turnieje o rozmiarach 5 i 10. Wynika to z tego, że zwycięzca turnieju wybierany był spośród większej puli uczestników, w związku z czym istnieje większe prawdopodobieństwo na to, że jego wartość będzie wyższa.

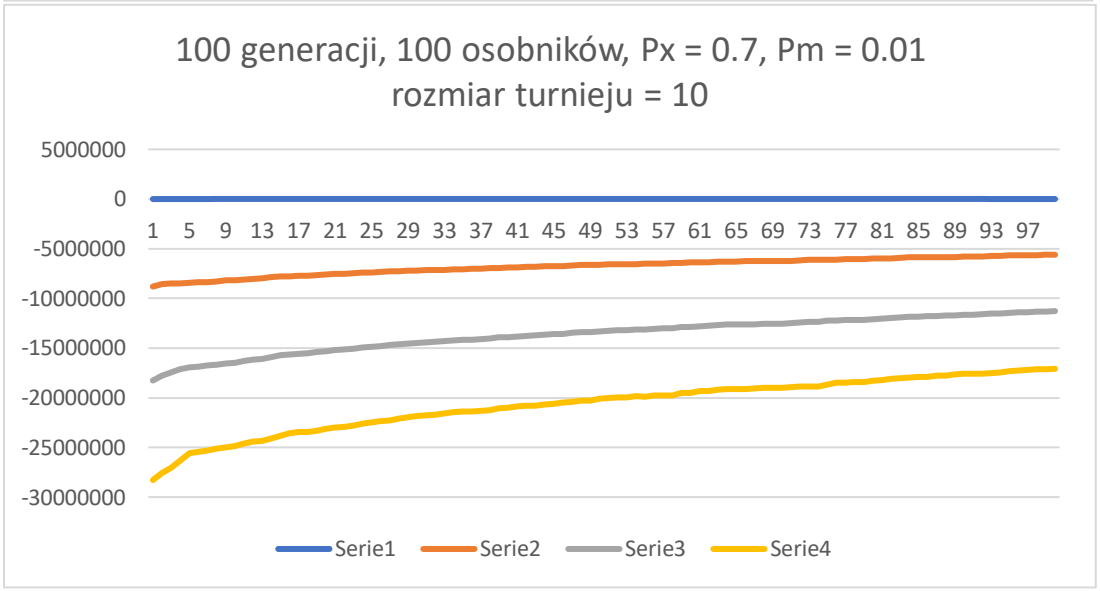
Turnieje o rozmiarach 100 i 1 nie były brane pod uwagę, ponieważ dla *TOURNAMENT_SIZE* = 100 zwycięzcą turnieju zawsze jest najsilniejszy osobnik populacji i nie ma tu miejsca na losowość, a populacja bardzo szybko stanie się niemal jednorodna. Dla *TOURNAMENT_SIZE* = 1 zawsze wygrywał losowy osobnik, w związku z czym kolejna populacja ponownie składała się z losowych osobników, co skutkowało bardzo niskimi wynikami.

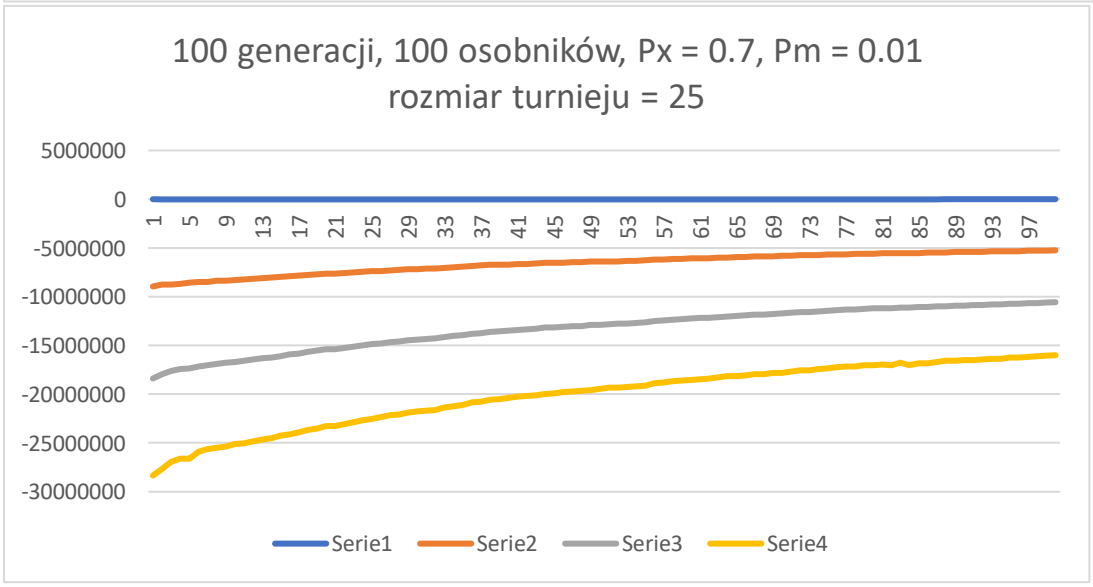

100 generacji, 100 osobników, Px = 0.7, Pm = 0.01, ruletka
5000000
0
-5000000
-10000000
-15000000
-20000000
-25000000
-30000000
1 5 9 13 17 21 25 29 33 37 41 45 49 53 57 61 65 69 73 77 81 85 89 93 97
Serie1
Serie2
Serie3
Serie4

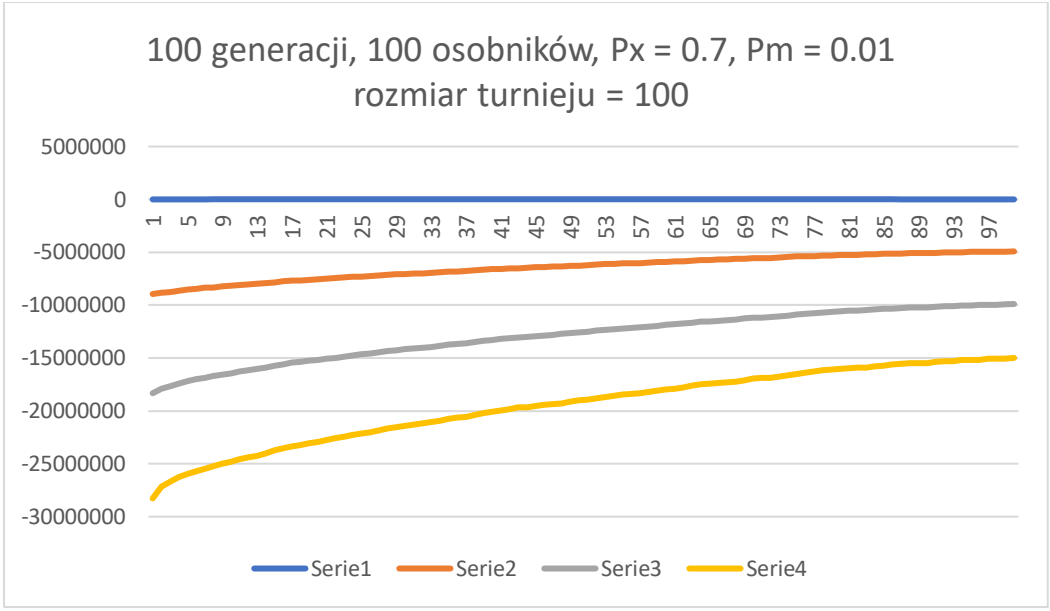

100 generacji, 100 osobników, Px = 0.7, Pm = 0.01, ranking
5000000
0
-5000000
-10000000
-15000000
-20000000
-25000000
-30000000
1 5 9 13 17 21 25 29 33 37 41 45 49 53 57 61 65 69 73 77 81 85 89 93 97
Serie1
Serie2
Serie3
Serie4

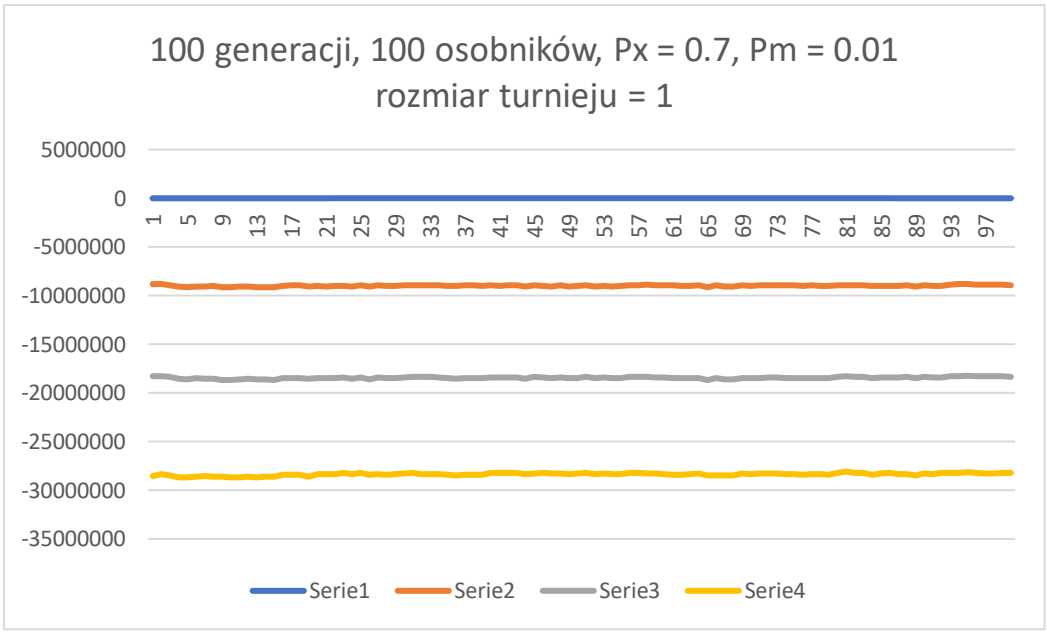

100 generacji, 100 osobników, Px = 0.7, Pm = 0.01, elitaryzm
5000000
0
-5000000
-10000000
-15000000
-20000000
-25000000
-30000000
1 5 9 13 17 21 25 29 33 37 41 45 49 53 57 61 65 69 73 77 81 85 89 93 97
Serie1
Serie2
Serie3
Serie4


100 generacji, 100 osobników, Px = 0.7, Pm = 0.01
rozmiar turnieju = 5
5000000
0
-5000000
-10000000
-15000000
-20000000
-25000000
-30000000
1 5 9 13 17 21 25 29 33 37 41 45 49 53 57 61 65 69 73 77 81 85 89 93 97
Serie1
Serie2
Serie3
Serie4


100 generacji, 100 osobników, Px = 0.7, Pm = 0.01
rozmiar turnieju = 10
5000000
0
-5000000
-10000000
-15000000
-20000000
-25000000
-30000000
1 5 9 13 17 21 25 29 33 37 41 45 49 53 57 61 65 69 73 77 81 85 89 93 97
Serie1
Serie2
Serie3
Serie4


100 generacji, 100 osobników, Px = 0.7, Pm = 0.01
rozmiar turnieju = 25
5000000
0
-5000000
-10000000
-15000000
-20000000
-25000000
-30000000
1 5 9 13 17 21 25 29 33 37 41 45 49 53 57 61 65 69 73 77 81 85 89 93 97
Serie1
Serie2
Serie3
Serie4


100 generacji, 100 osobników, Px = 0.7, Pm = 0.01
rozmiar turnieju = 100
5000000
0
-5000000
-10000000
-15000000
-20000000
-25000000
-30000000
1 5 9 13 17 21 25 29 33 37 41 45 49 53 57 61 65 69 73 77 81 85 89 93 97
Serie1
Serie2
Serie3
Serie4


100 generacji, 100 osobników, Px = 0.7, Pm = 0.01
rozmiar turnieju = 1
5000000
0
-5000000
-10000000
-15000000
-20000000
-25000000
-30000000
-35000000
1 5 9 13 17 21 25 29 33 37 41 45 49 53 57 61 65 69 73 77 81 85 89 93 97
Serie1
Serie2
Serie3
Serie4

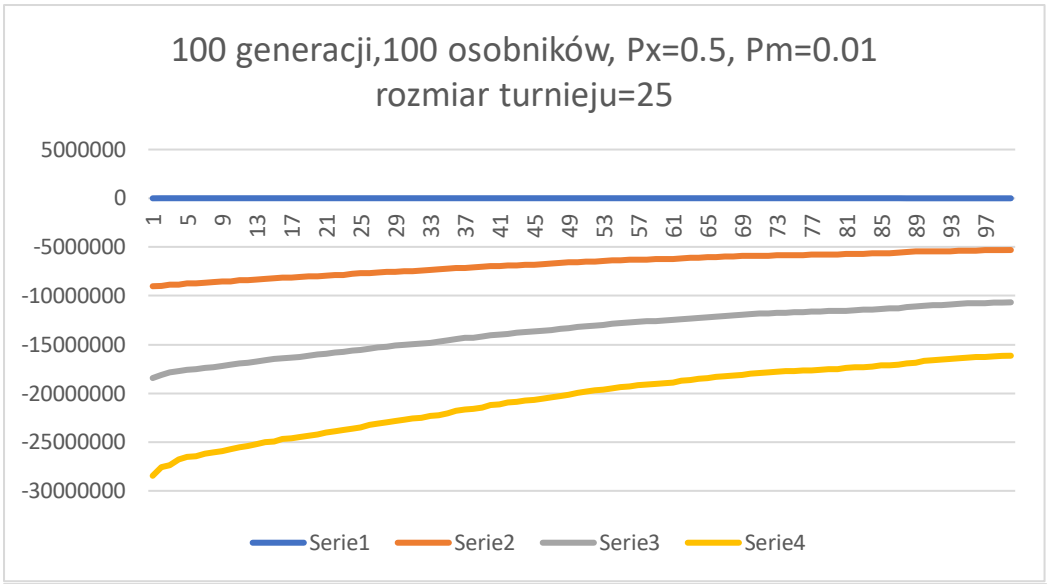
Wybór Px i Pm

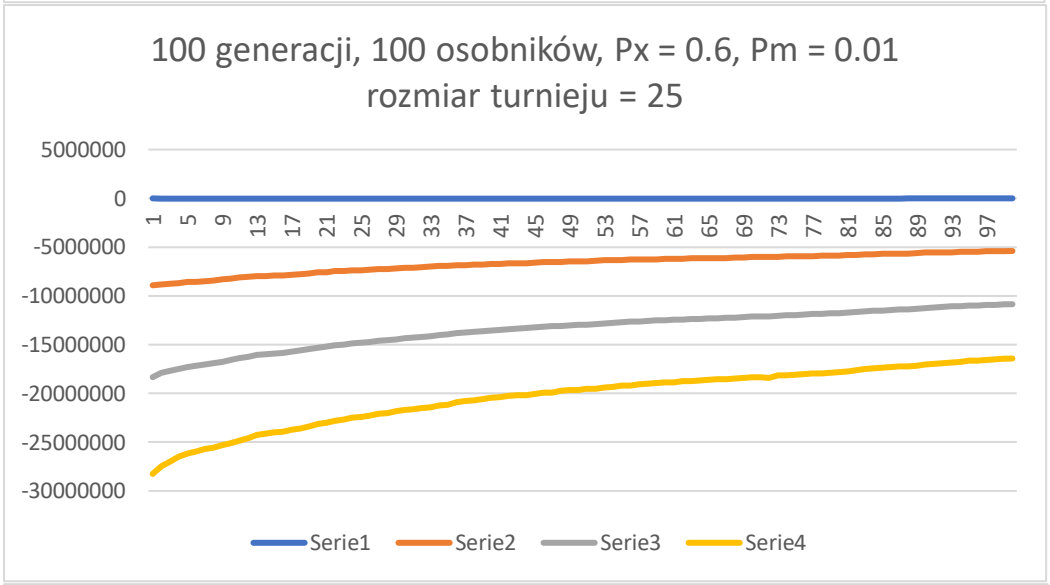
Aktualne parametry algorytmu:
POPULATION_SIZE: 100
POPULATIONS: 100
PX – 0.7
PM – 0.1
TOURNAMENT_SIZE : 25
TOURNAMENT_WINNERS : 1

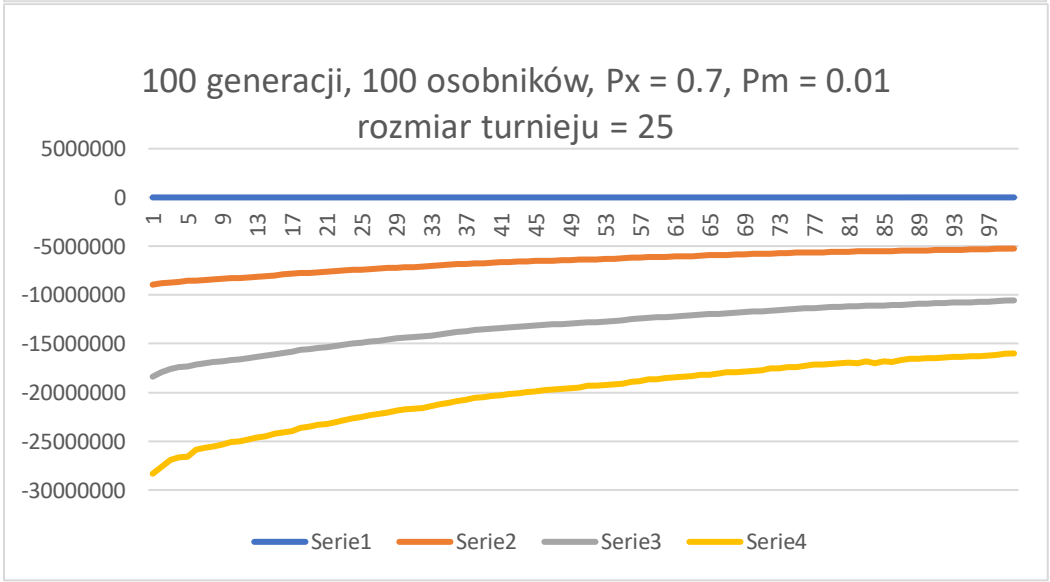
Zbadano skuteczność algorytmu dla wartości parametru PX: 0.5, 0.6, 0.7, 0.8, 0.9, 1. Dla parametru PX=0.9 algorytm wykazał się najwyższą skutecznością.

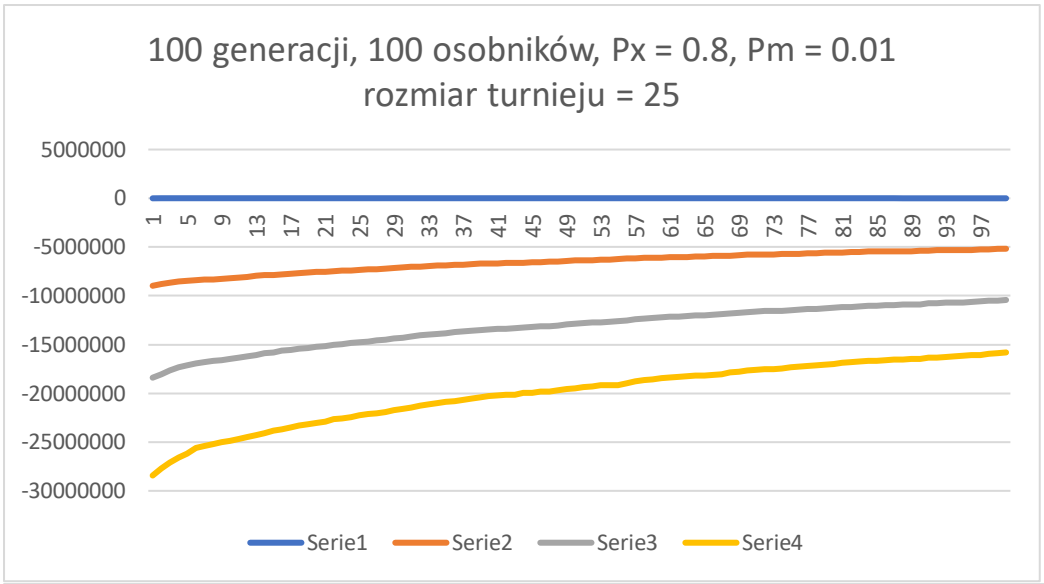
Następnie zbadano wartości parametru PM: 0.01, 0.05, 0.1 i 0.2. Algorytm osiągał najwyższą skuteczność dla parametru PM = 0.1. Dzięki mutacjom osobniki były nieco bardziej zróżnicowane, a ich wartości rosły w górę szybciej.

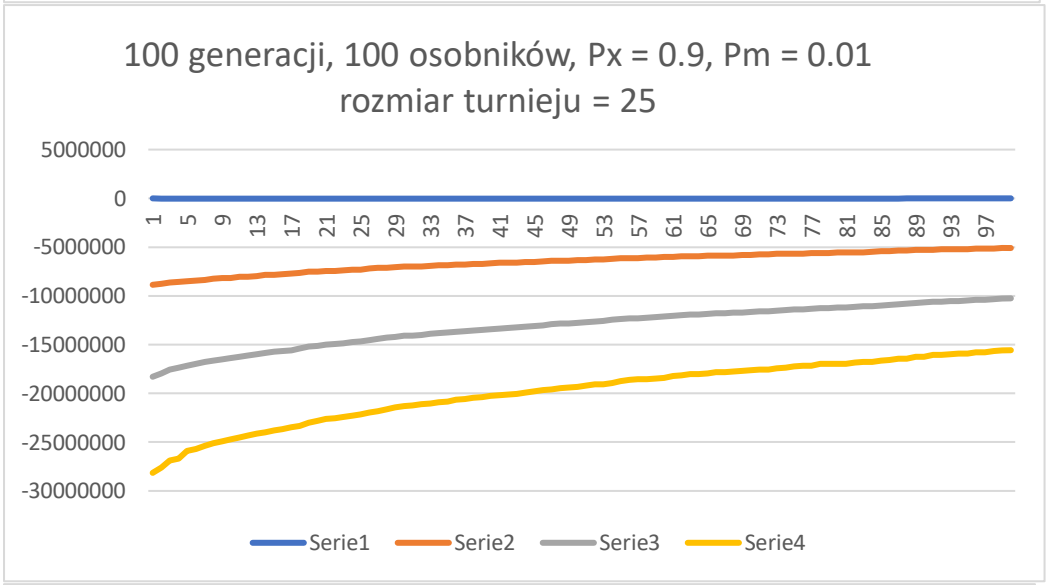
Zbadano także wpływ skrajnych wartości PX i PM na działanie algorytmu (PX=0 i PM=0.5) oraz (PX = 0.25 i PM = 0). Dla obu z tych przypadków algorytm osiągał podobną skuteczność (ponieważ dalej do nowej populacji przekazywane były wyselekcjonowane osobniki i nie były poddawane one dużym modyfikacjom).

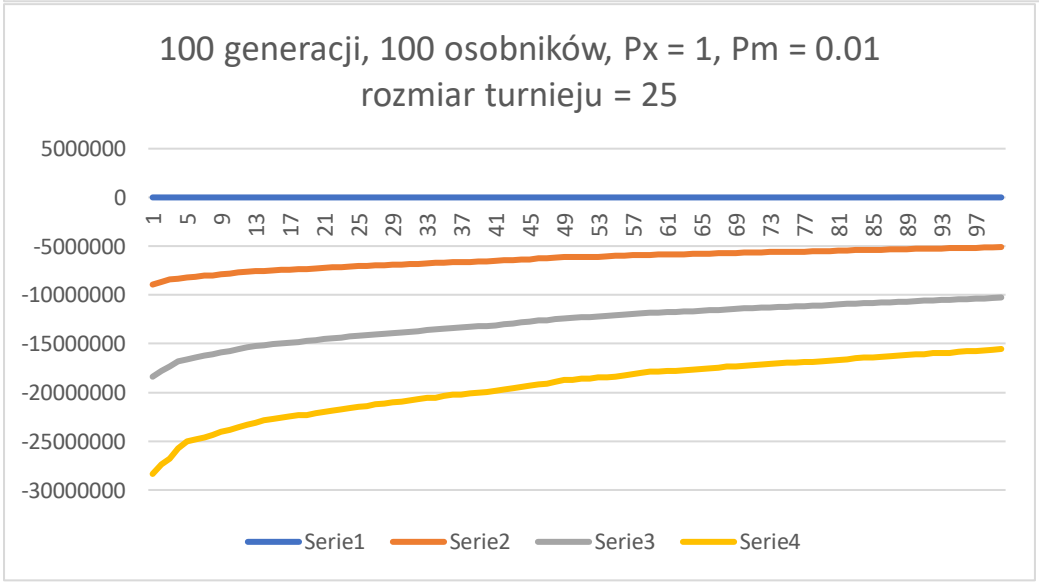

100 generacji,100 osobników, Px=0.5, Pm=0.01
rozmiar turnieju=25
5000000
0
-5000000
-10000000
-15000000
-20000000
-25000000
-30000000
1 5 9 13 17 21 25 29 33 37 41 45 49 53 57 61 65 69 73 77 81 85 89 93 97
Serie1
Serie2
Serie3
Serie4

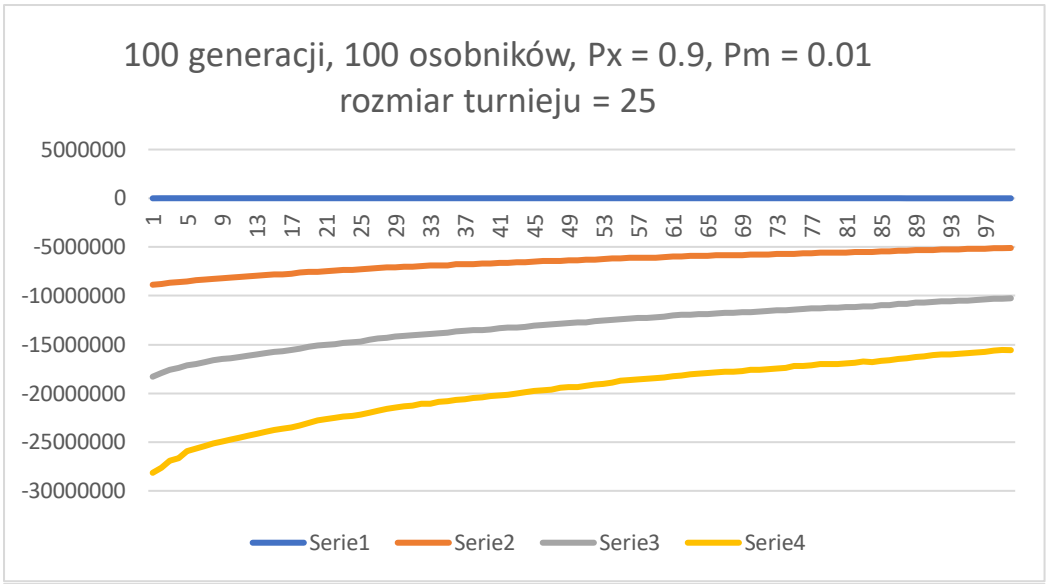

100 generacji, 100 osobników, Px = 0.6, Pm = 0.01
rozmiar turnieju = 25
5000000
0
-5000000
-10000000
-15000000
-20000000
-25000000
-30000000
1 5 9 13 17 21 25 29 33 37 41 45 49 53 57 61 65 69 73 77 81 85 89 93 97
Serie1
Serie2
Serie3
Serie4

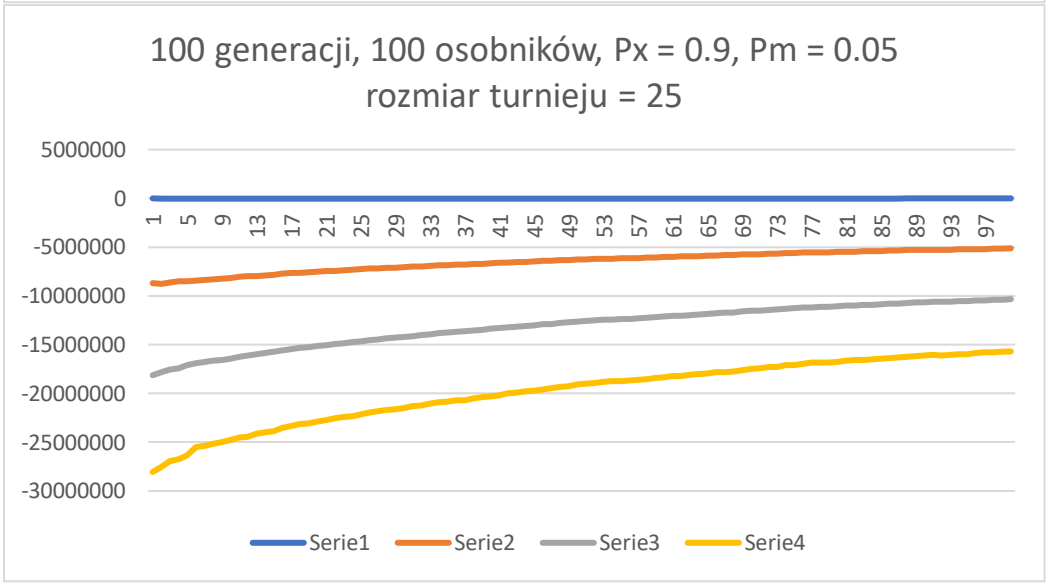

100 generacji, 100 osobników, Px = 0.7, Pm = 0.01
rozmiar turnieju = 25
5000000
0
-5000000
-10000000
-15000000
-20000000
-25000000
-30000000
1 5 9 13 17 21 25 29 33 37 41 45 49 53 57 61 65 69 73 77 81 85 89 93 97
Serie1
Serie2
Serie3
Serie4

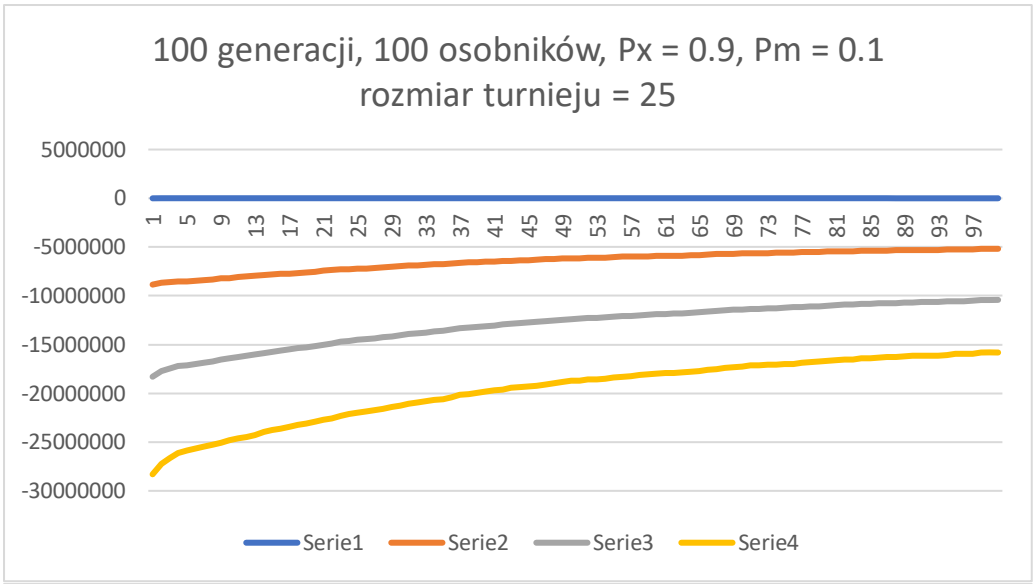

100 generacji, 100 osobników, Px = 0.8, Pm = 0.01
rozmiar turnieju = 25
5000000
0
-5000000
-10000000
-15000000
-20000000
-25000000
-30000000
1 5 9 13 17 21 25 29 33 37 41 45 49 53 57 61 65 69 73 77 81 85 89 93 97
Serie1
Serie2
Serie3
Serie4

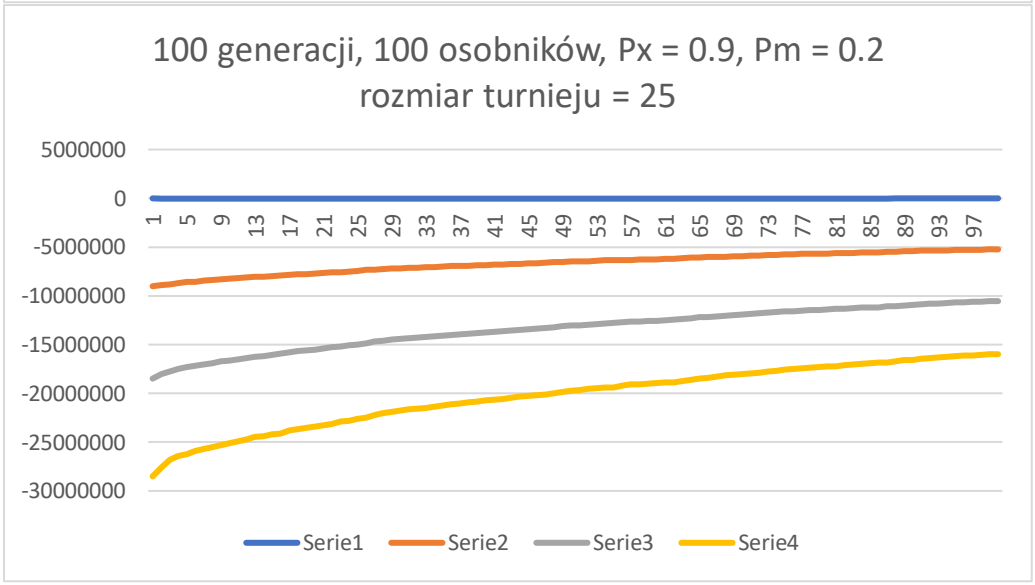

100 generacji, 100 osobników, Px = 0.9, Pm = 0.01
rozmiar turnieju = 25
5000000
0
-5000000
-10000000
-15000000
-20000000
-25000000
-30000000
1 5 9 13 17 21 25 29 33 37 41 45 49 53 57 61 65 69 73 77 81 85 89 93 97
Serie1
Serie2
Serie3
Serie4

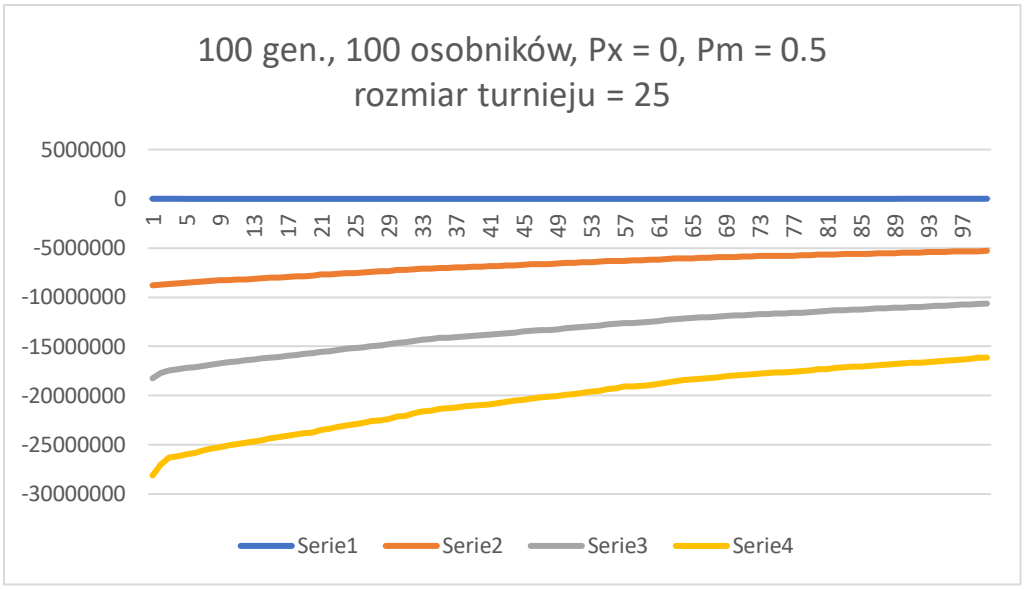

100 generacji, 100 osobników, Px = 1, Pm = 0.01
rozmiar turnieju = 25
5000000
0
-5000000
-10000000
-15000000
-20000000
-25000000
-30000000
1 5 9 13 17 21 25 29 33 37 41 45 49 53 57 61 65 69 73 77 81 85 89 93 97
Serie1
Serie2
Serie3
Serie4

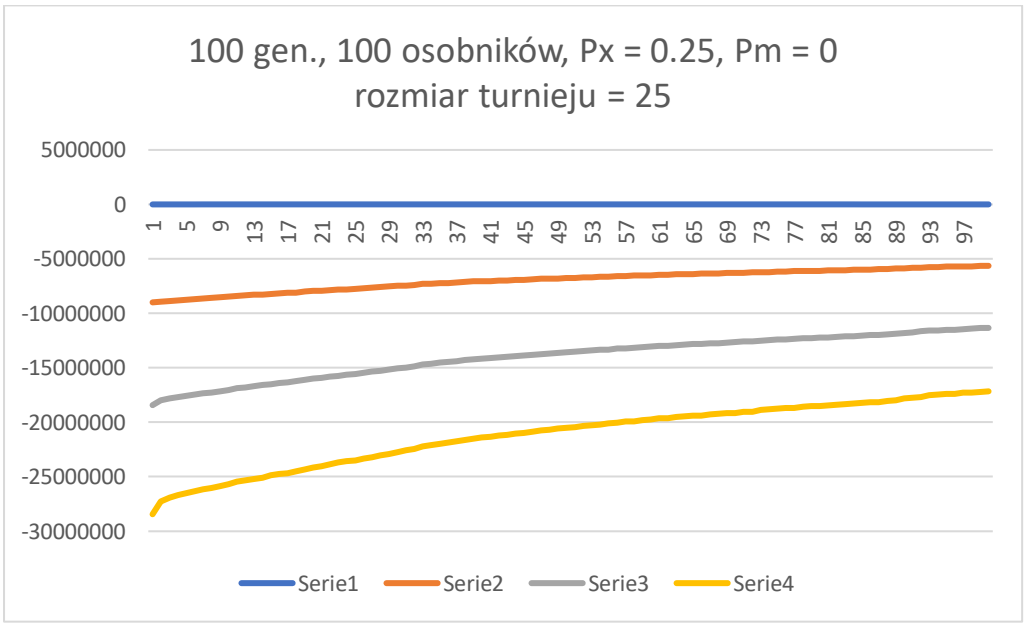

100 generacji, 100 osobników, Px = 0.9, Pm = 0.01
rozmiar turnieju = 25
5000000
0
-5000000
-10000000
-15000000
-20000000
-25000000
-30000000
1 5 9 13 17 21 25 29 33 37 41 45 49 53 57 61 65 69 73 77 81 85 89 93 97
Serie1
Serie2
Serie3
Serie4


100 generacji, 100 osobników, Px = 0.9, Pm = 0.05
rozmiar turnieju = 25
5000000
0
-5000000
-10000000
-15000000
-20000000
-25000000
-30000000
1 5 9 13 17 21 25 29 33 37 41 45 49 53 57 61 65 69 73 77 81 85 89 93 97
Serie1
Serie2
Serie3
Serie4


100 generacji, 100 osobników, Px = 0.9, Pm = 0.1
rozmiar turnieju = 25
5000000
0
-5000000
-10000000
-15000000
-20000000
-25000000
-30000000
1 5 9 13 17 21 25 29 33 37 41 45 49 53 57 61 65 69 73 77 81 85 89 93 97
Serie1
Serie2
Serie3
Serie4


100 generacji, 100 osobników, Px = 0.9, Pm = 0.2
rozmiar turnieju = 25
5000000
0
-5000000
-10000000
-15000000
-20000000
-25000000
-30000000
1 5 9 13 17 21 25 29 33 37 41 45 49 53 57 61 65 69 73 77 81 85 89 93 97
Serie1
Serie2
Serie3
Serie4


100 gen., 100 osobników, Px = 0, Pm = 0.5
rozmiar turnieju = 25
5000000
0
-5000000
-10000000
-15000000
-20000000
-25000000
-30000000
1
5
9
13
17
21
25
29
33
37
41
45
49
53
57
61
65
69
73
77
81
85
89
93
97
Serie1
Serie2
Serie3
Serie4


100 gen., 100 osobników, Px = 0.25, Pm = 0
rozmiar turnieju = 25
5000000
0
-5000000
-10000000
-15000000
-20000000
-25000000
-30000000
1
5
9
13
17
21
25
29
33
37
41
45
49
53
57
61
65
69
73
77
81
85
89
93
97
Serie1
Serie2
Serie3
Serie4

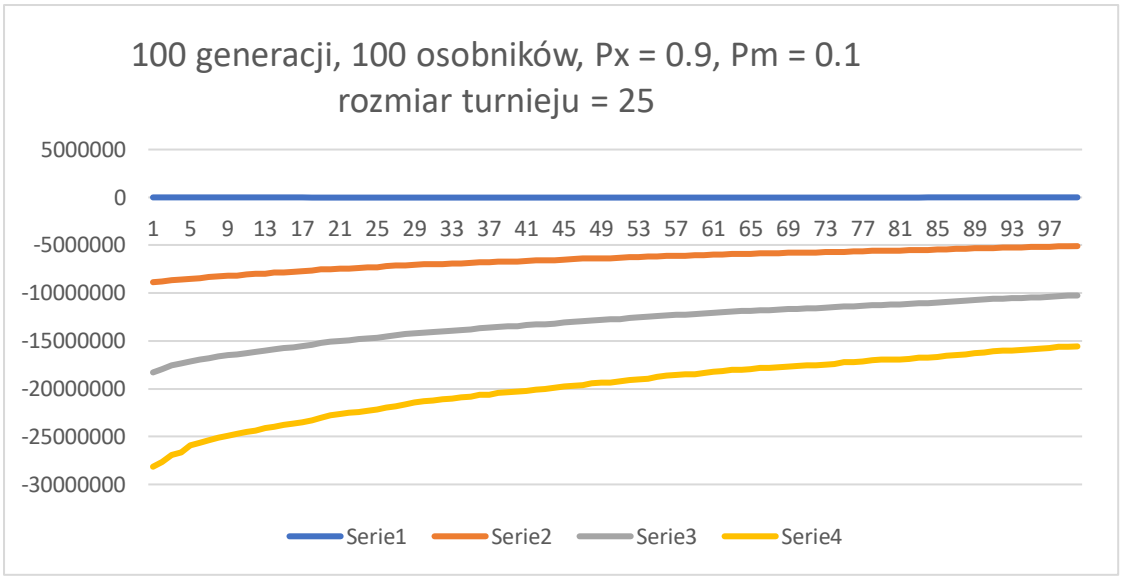
Badanie wpływu liczby generacji i rozmiaru populacji na wynik działania algorytmu

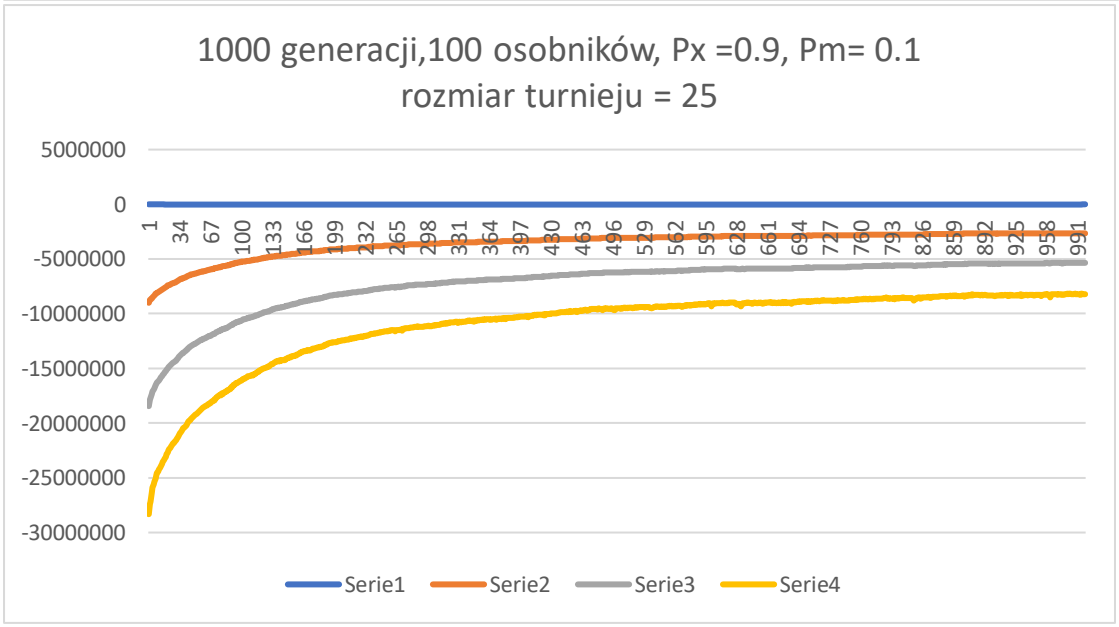
Aktualne parametry algorytmu:
POPULATION_SIZE: 100
POPULATIONS: 100
PX – 0.9
PM – 0.1
TOURNAMENT_SIZE : 25
TOURNAMENT_WINNERS : 1

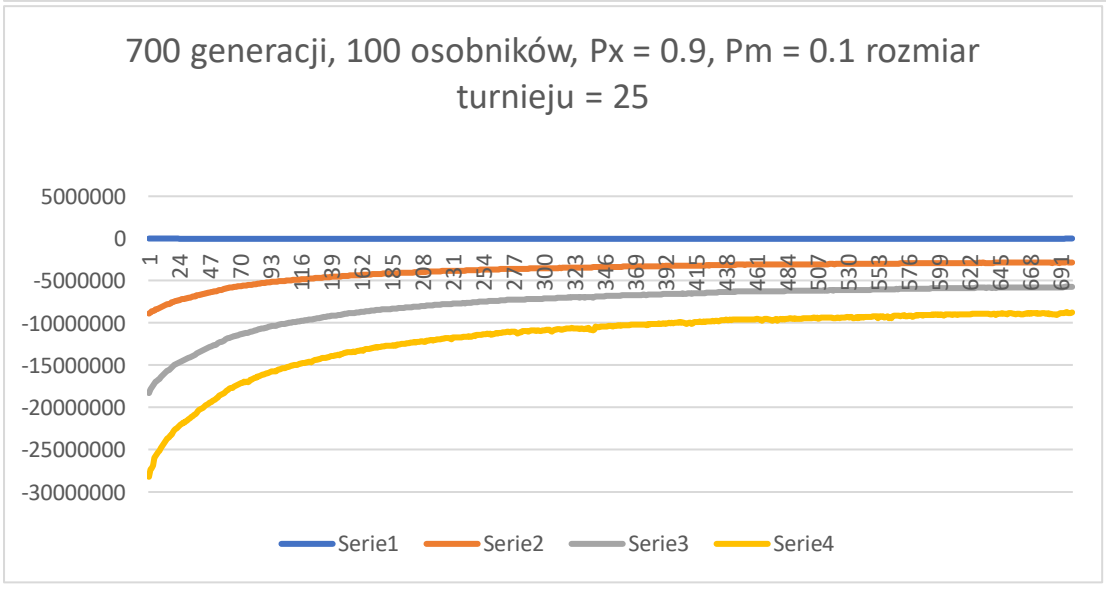
Logicznym jest, że im więcej stworzone zostanie generacji, tym lepszy będzie wynik działania algorytmu. Jednak powyżej 7000 generacji wzrost wartości funkcji celu osobników był coraz wolniejszy, dlatego na potrzebę zadania wartość parametru *POPULATIONS* = 7000 została uznana za optymalną.

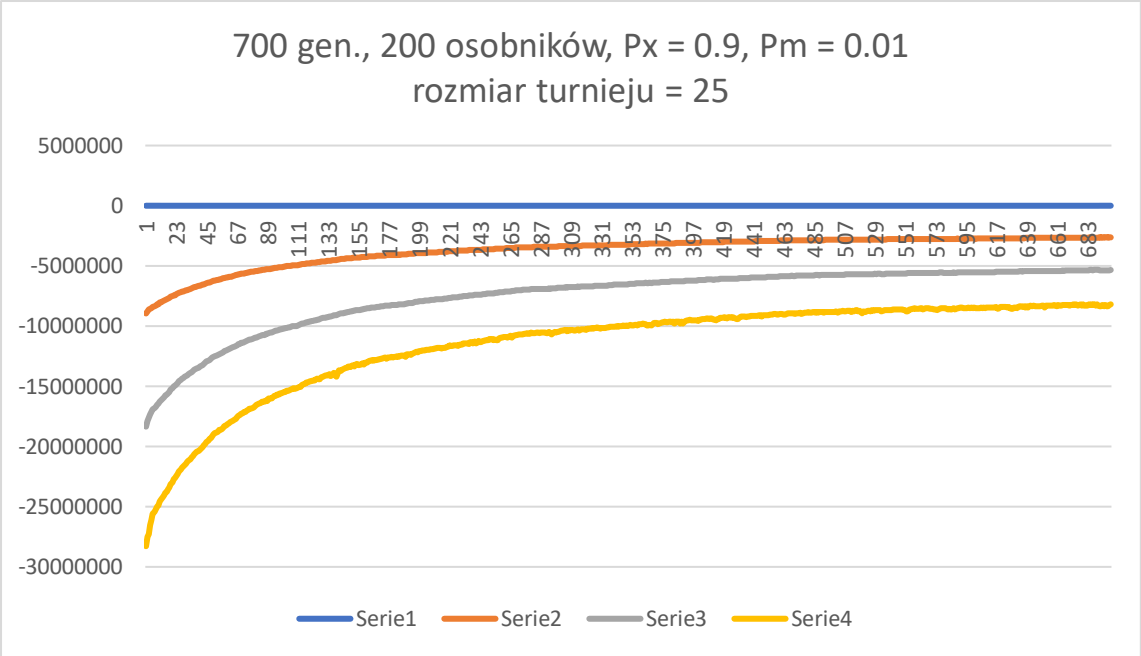
Rozmiar populacji *POPULATION_SIZE* niewiele wpływał na skuteczność algorytmu, dlatego w celu skrócenia jego działania została wybrana najniższa możliwa (nadal skuteczna) wartość wynosząca 100 osobników.

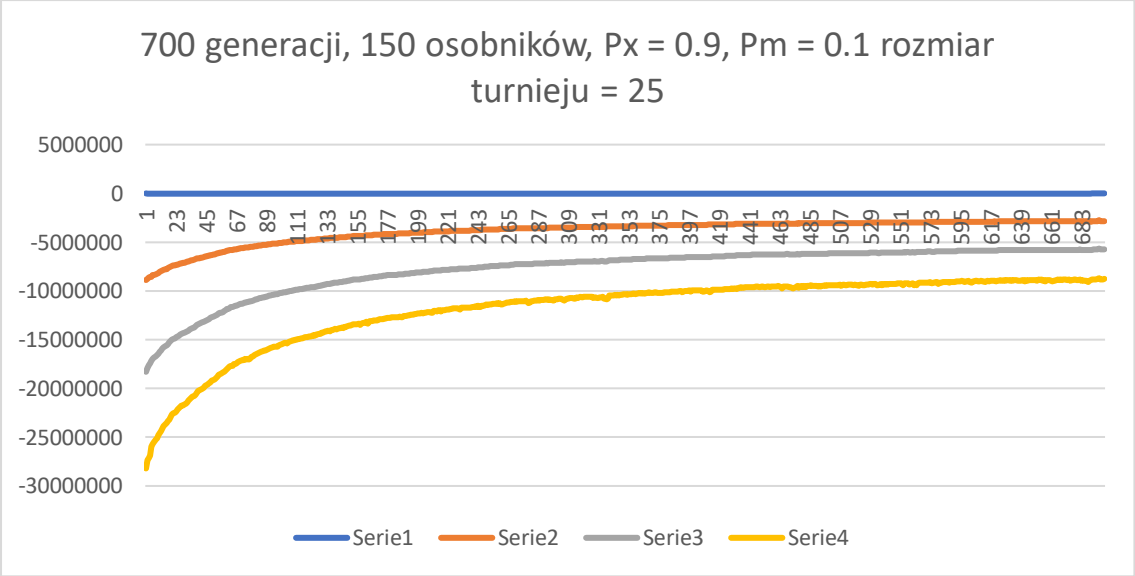
Badanie wpływu wartości parametrów *POPULATIONS i POPULATION_SIZE* zostało zobrazowane na poniższych wykresach.

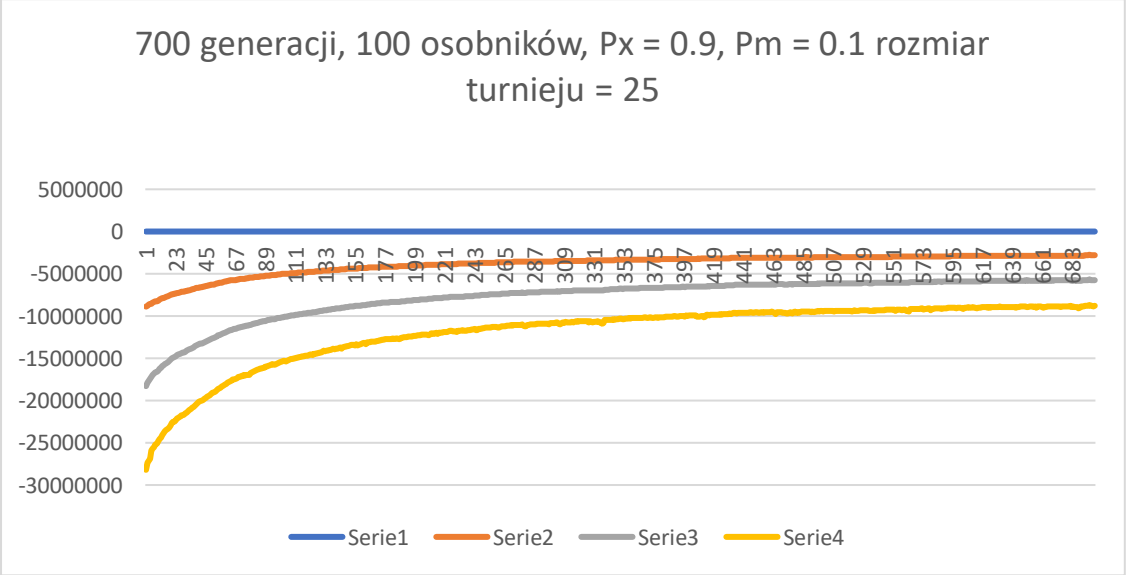

100 generacji, 100 osobników, Px = 0.9, Pm = 0.1
rozmiar turnieju = 25
5000000
0
-5000000
-10000000
-15000000
-20000000
-25000000
-30000000
1 5 9 13 17 21 25 29 33 37 41 45 49 53 57 61 65 69 73 77 81 85 89 93 97
Serie1
Serie2
Serie3
Serie4


1000 generacji,100 osobników, Px =0.9, Pm= 0.1
rozmiar turnieju = 25
5000000
0
-5000000
-10000000
-15000000
-20000000
-25000000
-30000000
1 34 67 100 133 166 199 232 265 298 331 364 397 430 463 496 529 562 595 628 661 694 727 760 793 826 859 892 925 958 991
Serie1
Serie2
Serie3
Serie4


700 generacji, 100 osobników, Px = 0.9, Pm = 0.1 rozmiar
turnieju = 25
5000000
0
-5000000
-10000000
-15000000
-20000000
-25000000
-30000000
1 24 47 70 93 116 139 162 185 208 231 254 277 300 323 346 369 392 415 438 461 484 507 530 553 576 599 622 645 668 691
Serie1
Serie2
Serie3
Serie4

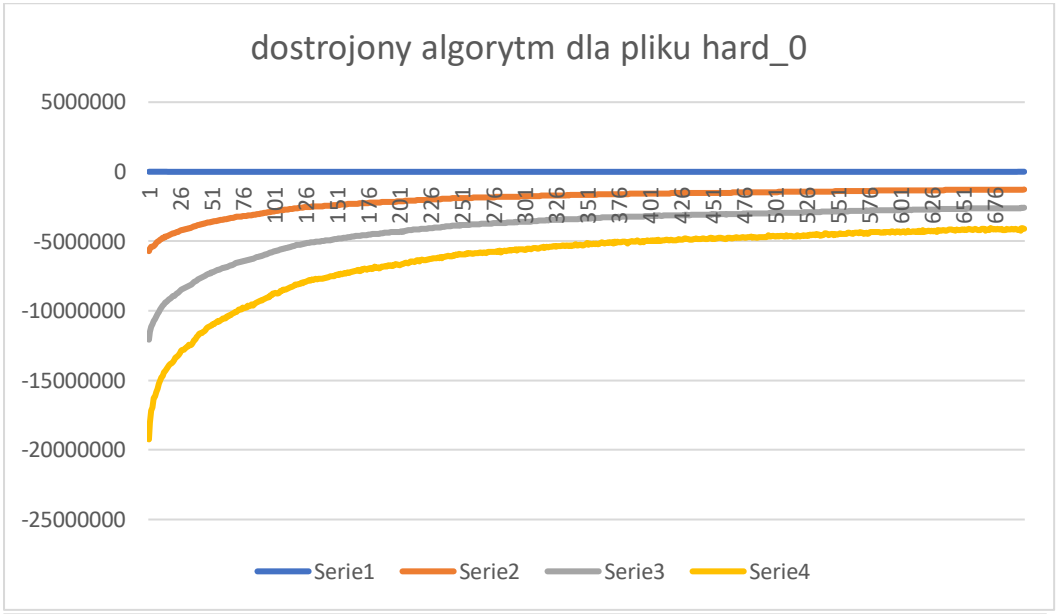

700 gen., 200 osobników, Px = 0.9, Pm = 0.01
rozmiar turnieju = 25
5000000
0
-5000000
-10000000
-15000000
-20000000
-25000000
-30000000
1 23 45 67 89 111 133 155 177 199 221 243 265 287 309 331 353 375 397 419 441 463 485 507 529 551 573 595 617 639 661 683
Serie1
Serie2
Serie3
Serie4

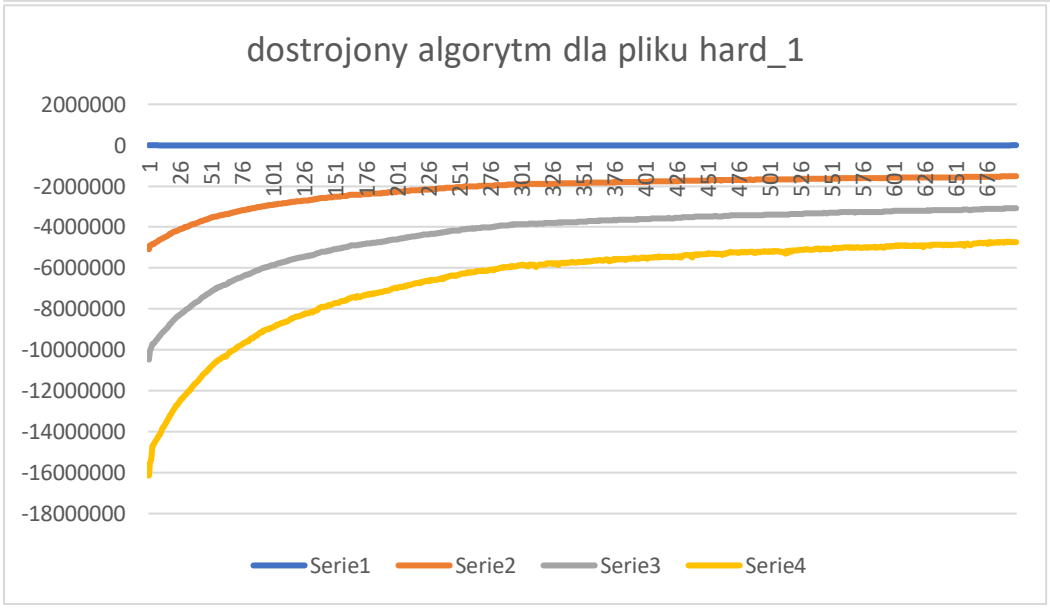

700 generacji, 150 osobników, Px = 0.9, Pm = 0.1 rozmiar turnieju = 25
5000000
0
-5000000
-10000000
-15000000
-20000000
-25000000
-30000000
1 23 45 67 89 111 133 155 177 199 221 243 265 287 309 331 353 375 397 419 441 463 485 507 529 551 573 595 617 639 661 683
Serie1
Serie2
Serie3
Serie4

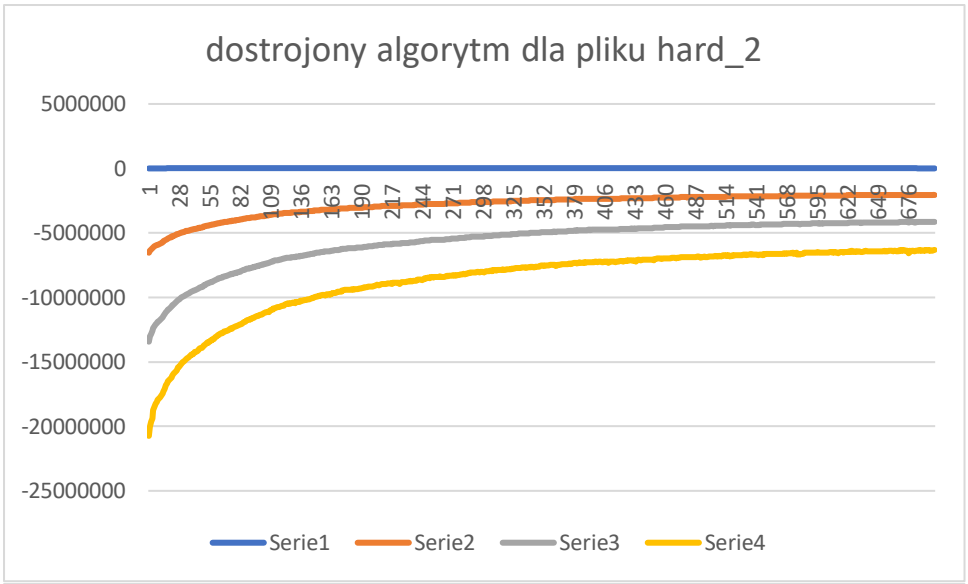

700 generacji, 100 osobników, Px = 0.9, Pm = 0.1 rozmiar turnieju = 25
5000000
0
-5000000
-10000000
-15000000
-20000000
-25000000
-30000000
1 23 45 67 89 111 133 155 177 199 221 243 265 287 309 331 353 375 397 419 441 463 485 507 529 551 573 595 617 639 661 683
Serie1
Serie2
Serie3
Serie4

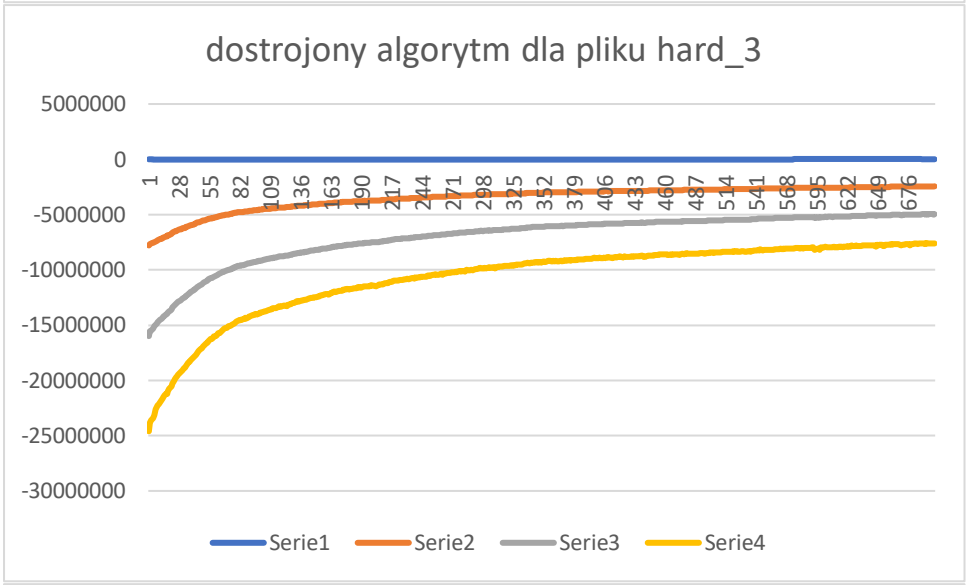
Działanie dostrojonego algorytmu

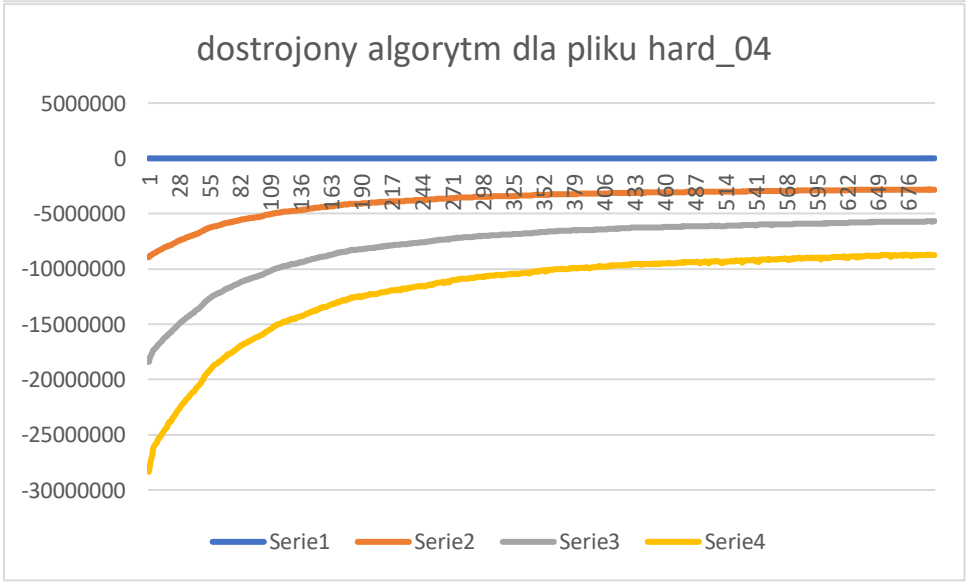
Poniższe wykresy przedstawiają uśrednione wyniki działania algorytmu na każdym z 5 plików „hard”.






dostrojony algorytm dla pliku hard_2
5000000
0
-5000000
-10000000
-15000000
-20000000
-25000000
1 28 55 82 109 136 163 190 217 244 271 298 325 352 379 406 433 460 487 514 541 568 595 622 649 676
Serie1
Serie2
Serie3
Serie4


dostrojony algorytm dla pliku hard_3
5000000
0
-5000000
-10000000
-15000000
-20000000
-25000000
-30000000
1 28 55 82 109 136 163 190 217 244 271 298 325 352 379 406 433 460 487 514 541 568 595 622 649 676
Serie1
Serie2
Serie3
Serie4


dostrojony algorytm dla pliku hard_04
5000000
0
-5000000
-10000000
-15000000
-20000000
-25000000
-30000000
1 28 55 82 109 136 163 190 217 244 271 298 325 352 379 406 433 460 487 514 541 568 595 622 649 676
Serie1
Serie2
Serie3
Serie4

5. Porównanie wyników algorytmu genetycznego z algorytmami niegenetycznymi i wnioski

Porównanie

Poniższa tabelka przedstawia uśrednione wyniki działania trzech algorytmów: genetycznego, zachłannego wybierającego zawsze najbliższe miasto i takiego, który wybiera najlepszego ze 100 losowych osobników.

algorytm	best	avg	worst
zachłanny	220000	98147	-9565
genetyczny	-2815003	-2884501	-3023889
losowy	-8880348	-9429901	-9872732

Łatwo zauważyć, że dla algorytmu zachłannego program osiągał wyraźnie lepsze wyniki, niż dostrojony wg wcześniejszych parametrów algorytm genetyczny. Dzieje się tak dlatego, że algorytm genetyczny uczy się bardzo powoli. Gdyby stworzyć znacznie więcej (niż 700 stworzonych na potrzeby zadania) populacji w pewnym momencie wyniki działania algorytmu genetycznego przerosłyby wyniki algorytmu zachłannego. Da się to zauważyć przez porównanie algorytmu wybierającego losowego osobnika - algorytm genetyczny już w 2 populacji osiągnął znacznie wyższe wyniki.

Wnioski

Końcowe parametry algorytmu:
POPULATION_SIZE: 100
POPULATIONS: 700
PX – 0.9
PM – 0.1
TOURNAMENT_SIZE : 25

Na podstawie badań wpływu wartości parametrów na działanie algorytmu dało się zauważyć następujące własności:

- im wyższy rozmiar turnieju *TOURNAMENT_SIZE* tym większa jest szansa na przeniesienie do kolejnej populacji osobników o możliwie najwyższej wartości funkcji celu;
- najlepszym prawdopodobieństwem krzyżowania *PX* okazało się 0.9, a prawdopodobieństwem mutacji *PM* 0.1. Te wartości pozwoliły na tworzenie zróżnicowanych populacji o wysokim wzroście wartości osobników.
- im więcej generacji *POPULATIONS*, tym algorytm uzyskuje wyższe wyniki. Przekłada się to jednak na czas wykonywania algorytmu, dlatego jest to parametr którego wysokości nie można podnosić w nieskończoność;
- ilość osobników w populacji *POPULATION_SIZE* nie wpływała znacznie na wyniki działania algorytmu, za to czas jego wykonywania podnosiła bardzo wyraźnie, dlatego wysokość tego parametru najlepiej było przybrać możliwie najniższą;