Wydział Elektroniki i Technik Informacyjnych
Politechnika Warszawska

Algorytmy Ewolucyjne

Projekt 2

Bartosz Goławski

Warszawa, 3.05.2020

Spis treści

1. Treść zadania i sposób implementacji

1.1. Treść zadania

Stosując algorytm genetyczny znajdź rozwiązanie problemu plecakowego:

$$\begin{aligned} \max_{\mathrm{x}} \sum_{i=1}^{n} p_i x_i \\ \sum_{i=1}^{n} w_i x_i \leqslant W \\ p_i > 0 \\ w_i > 0 \\ x_i \in \{0, 1\} \end{aligned}$$

Założenia:

— liczba przedmiotów: $n = 32$ i $n = 64$
— do generacji przedmiotów wykorzystać Skrypt 1, wagi w przedmiotów są losowane z rozkładem równomiernym z przedziału <0,1, 1> z dokładnością do 0,1, a wartości p przedmiotów są losowane z rozkładem równomiernym z przedziału: <1,100> z dokładnością do 1
— maksymalna waga plecaka: W = 30% wagi wszystkich przedmiotów
— dozwolone jest korzystanie ze środowiska MATLAB wraz z dodatkiem Global Optimization Toolbox (optimtool). Wykonanie projektu w Pythonie wymaga uprzedniej konsultacji z prowadzącym projekt.

Należy dobrać optymalne parametry algorytmu i metodę selekcji. W sprawozdaniu należy zawrzeć:

— Wektor binarny stanowiący rozwiązanie problemu
— Wartości liczności populacji i prawdopodobieństw mutacji i rekombinacji.
— Kryteria doboru optymalnych parametrów, np. warunku zatrzymania algorytmu
— Dla każdego uruchomienia wykres wartości funkcji celu (min., śr., max., wariancja) w funkcji numeru generacji.
— Porównanie działania GA dla 32 i 64 przedmiotów
— Sprawozdanie nie powinno zawierać niepotrzebnych informacji – takich jak np. teoria i opis metod optymalizacji.

1.2. Implementacja ograniczenia w funkcji celu

Ponieważ w przypadku wybrania populacji typu `bitstring` odrzucane są wszelkie ograniczenia wprowadzane jawnie do algorytmu GA, zdecydowałem się na sztuczne wprowadzenie ograniczenia wewnątrz funkcji celu. W przypadku zbyt ciężkich rozwiązań funkcja celu jest pogarszana o $\alpha(W_{\text{akt}} - W_{\text{max}})$, gdzie α oznacza współczynnik proporcjonalności, a W_{akt} oznacza wagę badanego zestawu przedmiotów.

1.3. Rozwiązanie za pomocą programowania liniowego

Rozwiązywany problem jest problemem programowania liniowego całkowitoliczbowego, więc można wykorzystać do jego rozwiązania solver (na przykład wykorzystany przeze mnie `GLPK`). Wyniki optymalizacji:

— $n = 32$
  — wartość funkcji celu = 982
  — X = [0 1 1 0 1 1 0 0 1 0 0 1 0 1 0 0 0 0 0 1 0 1 0 0 1 0 1 1 0 1 1]
— $n = 64$
  — wartość funkcji celu = 1980
  — X = [0 1 0 1 0 0 0 0 1 1 1 1 0 1 1 0 0 0 1 0 0 1 1 1 0 1 1 0 1 1 0 0 1 0 0 0 0 1 0 1 1 0 0 0 0 0 1 0 1 1 1 0 1 0 0 1 0 0 1 0 0 1 0 0]

1.4. Warunki zatrzymania algorytmu genetycznego

Przy wyborze warunków zatrzymania GA zostałem przy standardowych ustawieniach, to znaczy:

— maksymalna liczba pokoleń = $100N$
— brak ograniczeń limitu czasu wykonywania
— brak ograniczenia na maksymalną wartość funkcji celu
— co najwyżej 50 pokoleń, które się zatrzymały (stall generations)
— kryterium utknięcia pokolenia - średnia zmienność funkcji celu

2. Optymalizacja dla $n = 32$

W każdym z przypadków liczność generacji to 200. Na rysunkach zawarto wykresy wskaźników maksimum, minimum, średniej oraz wariancji wartości funkcji celu zarówna dla wszystkich zestawów przedmiotów, jak i dla przedmiotów o dopuszczalnej wadze, jak i tych, które przekroczyły dopuszczalną wagę w danym pokoleniu.

2.1. Strojenie parametru α

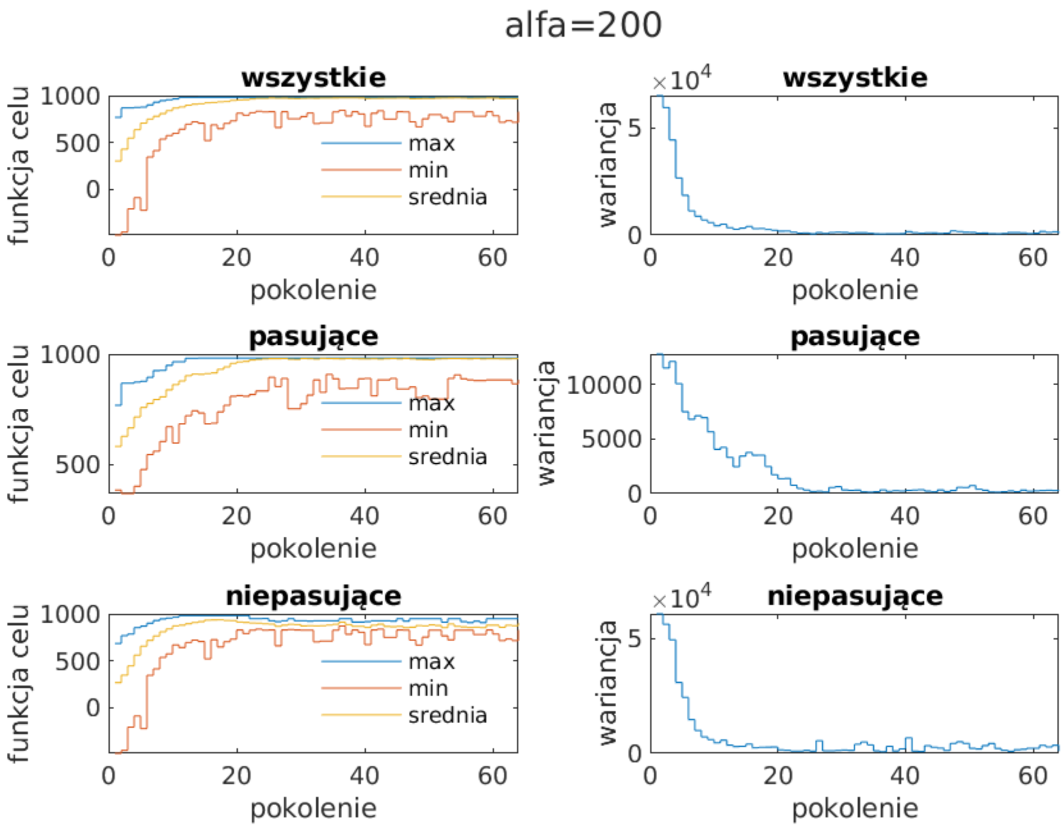
W przypadku strojenia parametru α jedyne sensowne wartości są dla $\alpha \geqslant 200$, ponieważ dla mniejszych wartości α kara za przekroczenie dopuszczalnej wagi jest zbyt mała. Wyniki optymalizacji:

α	y	p mutacji	p krzyżowania
200	982	0,01	0,8
300	982	0,01	0,8
500	967	0,01	0,8
700	982	0,01	0,8

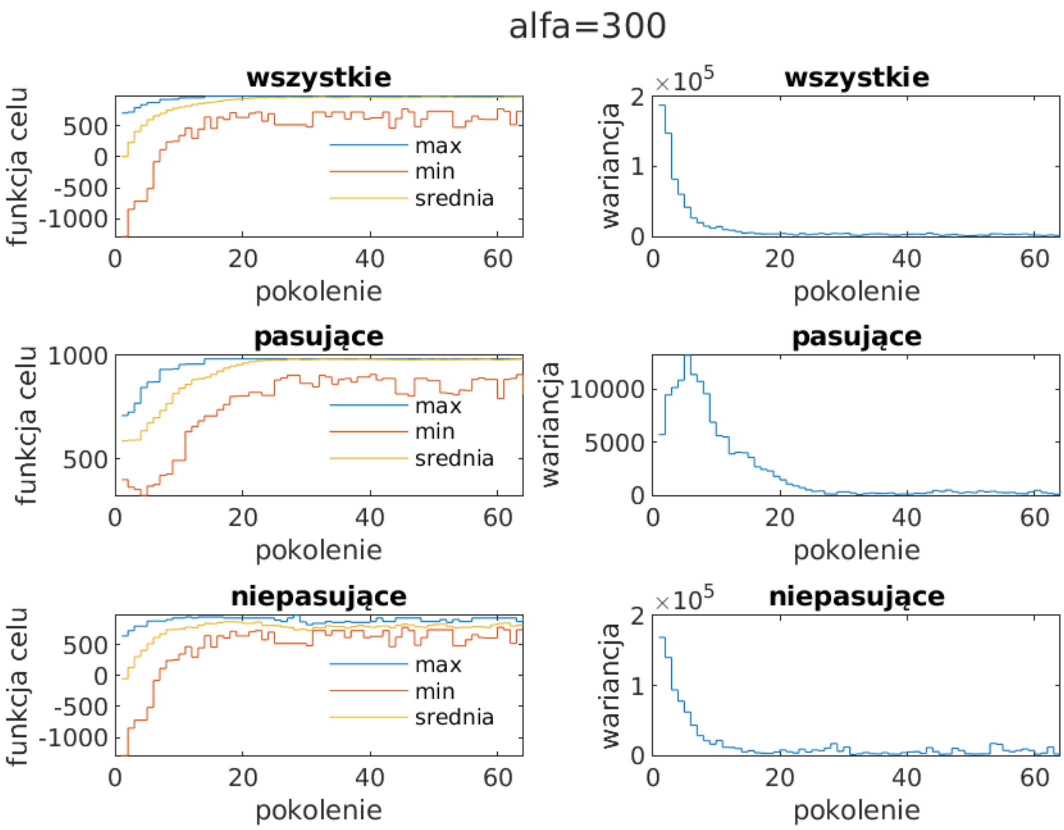
Wektory rozwiązań odpowiadające kolejnym wartościom α:

— X = [0 1 1 0 1 1 0 0 1 0 0 1 0 1 0 0 0 0 0 0 1 0 1 0 0 1 0 1 1 0 1 1]
— X = [0 1 1 0 1 1 0 0 1 0 0 1 0 1 0 0 0 0 0 0 1 0 1 0 0 1 0 1 1 0 1 1]
— X = [0 1 1 0 1 0 0 0 1 0 0 1 0 1 1 0 0 0 1 0 1 0 1 0 0 1 0 0 1 0 1 1]
— X = [0 1 1 0 1 1 0 0 1 0 0 1 0 1 0 0 0 0 0 0 1 0 1 0 0 1 0 1 1 0 1 1]

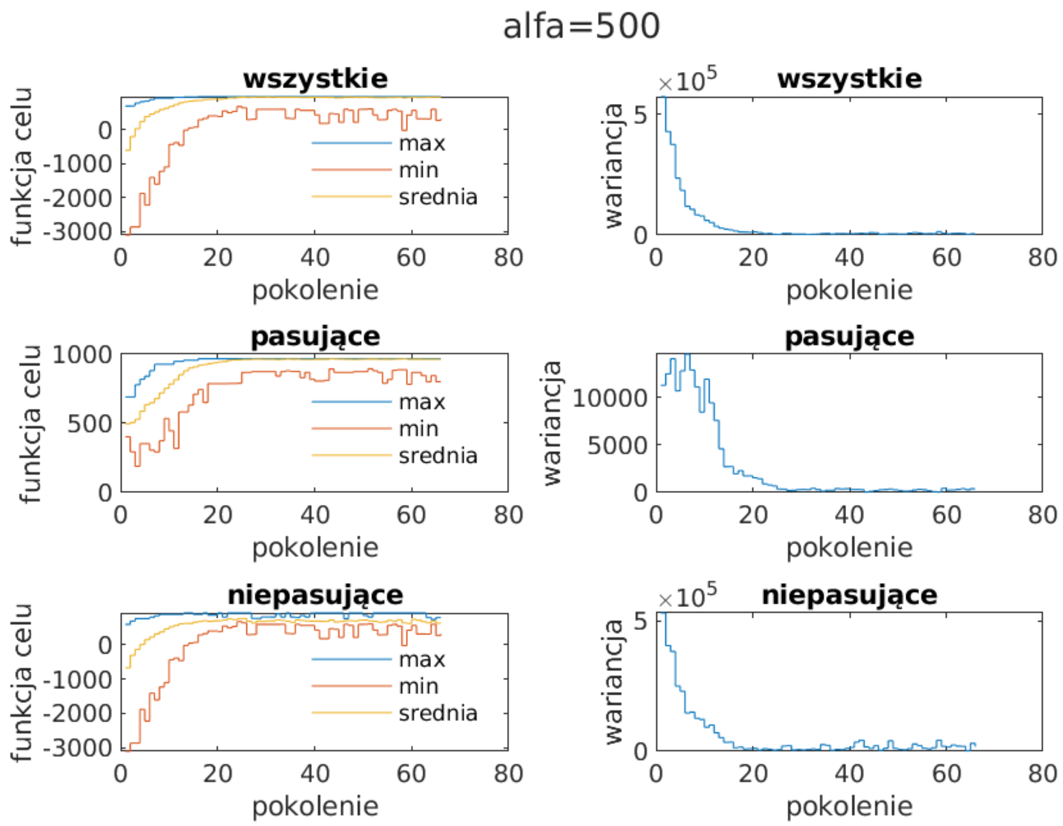
Zdecydowałem się na wybranie $\alpha = 200$. Warto utrzymywać możliwie jak najniższy współczynnik kary, by nie zatracić szybko różnorodności kolejnych pokoleń. Wykresy pokazujące przebieg symulacji znajdują się poniżej.



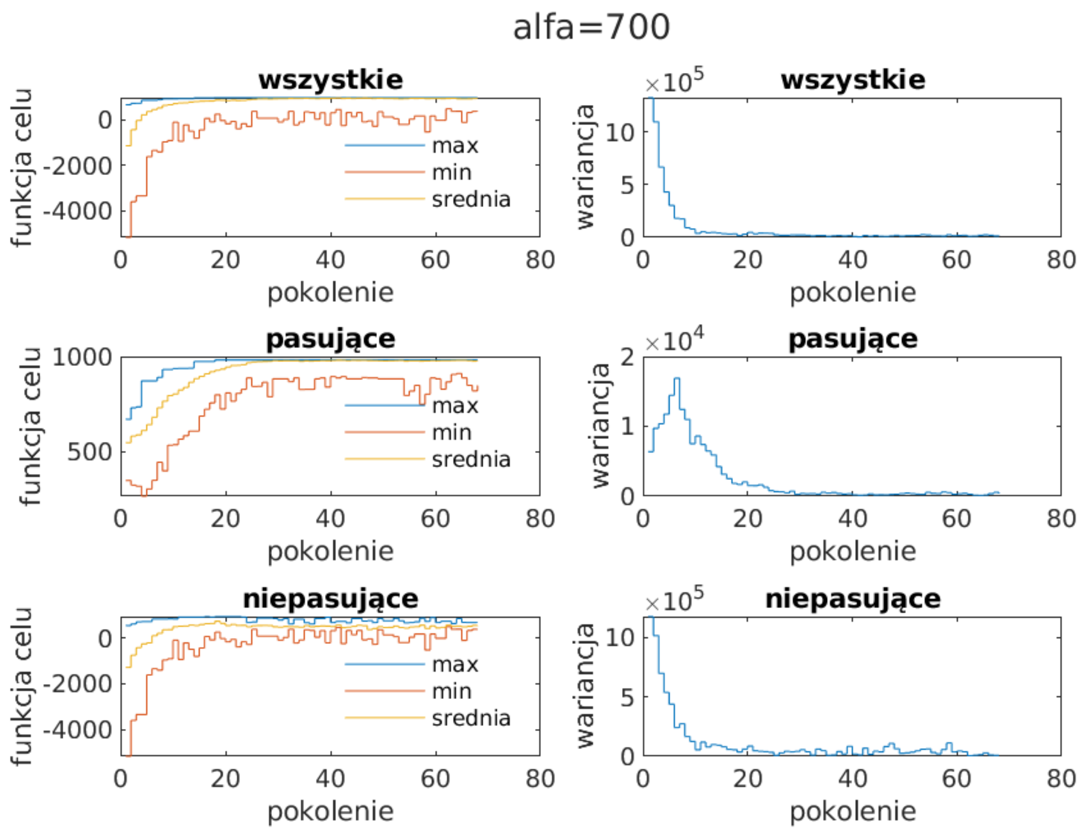
Rys. 2.1. Wyniki symulacji dla $\alpha = 200$



Rys. 2.2. Wyniki symulacji dla $\alpha = 300$



Rys. 2.3. Wyniki symulacji dla $\alpha = 500$



Rys. 2.4. Wyniki symulacji dla $\alpha = 700$

2.2. Dobór prawdopodobieństwa mutacji

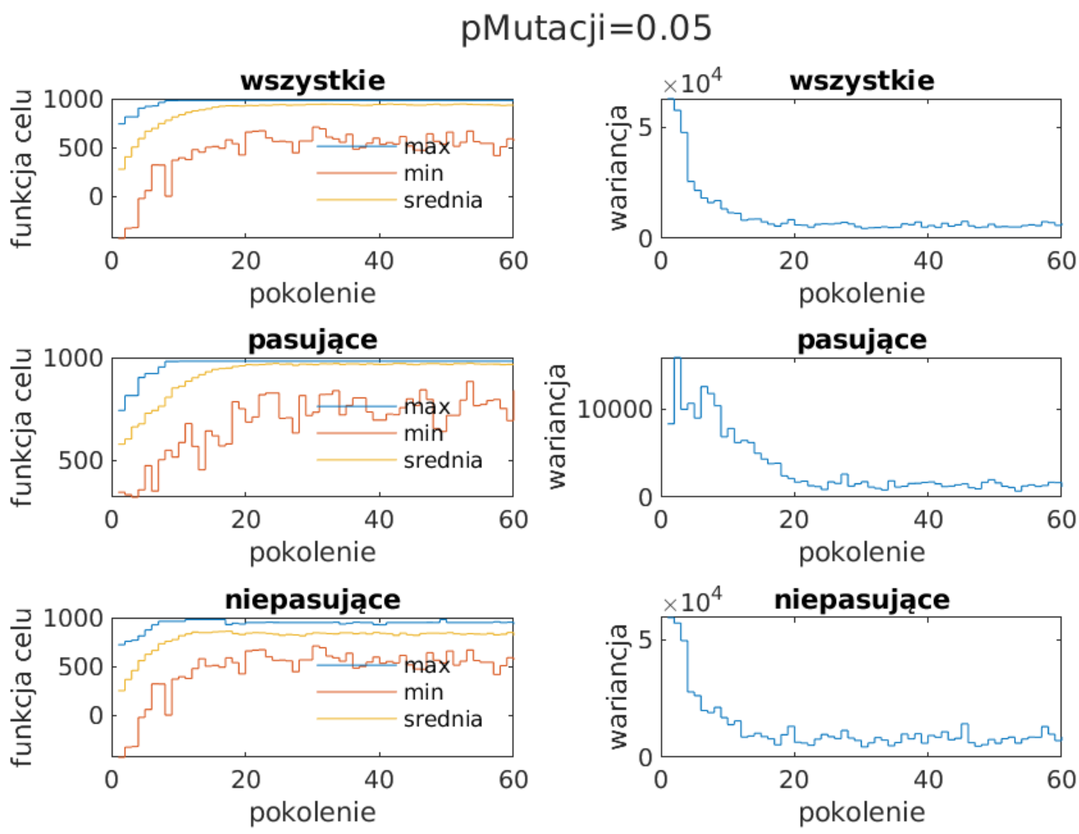
Wyniki optymalizacji:

p mutacji	y	p krzyżowania
0,05	982	0,8
0,1	982	0,8
0,2	982	0,8
0,3	982	0,8
0,5	982	0,8
0,8	965	0,8

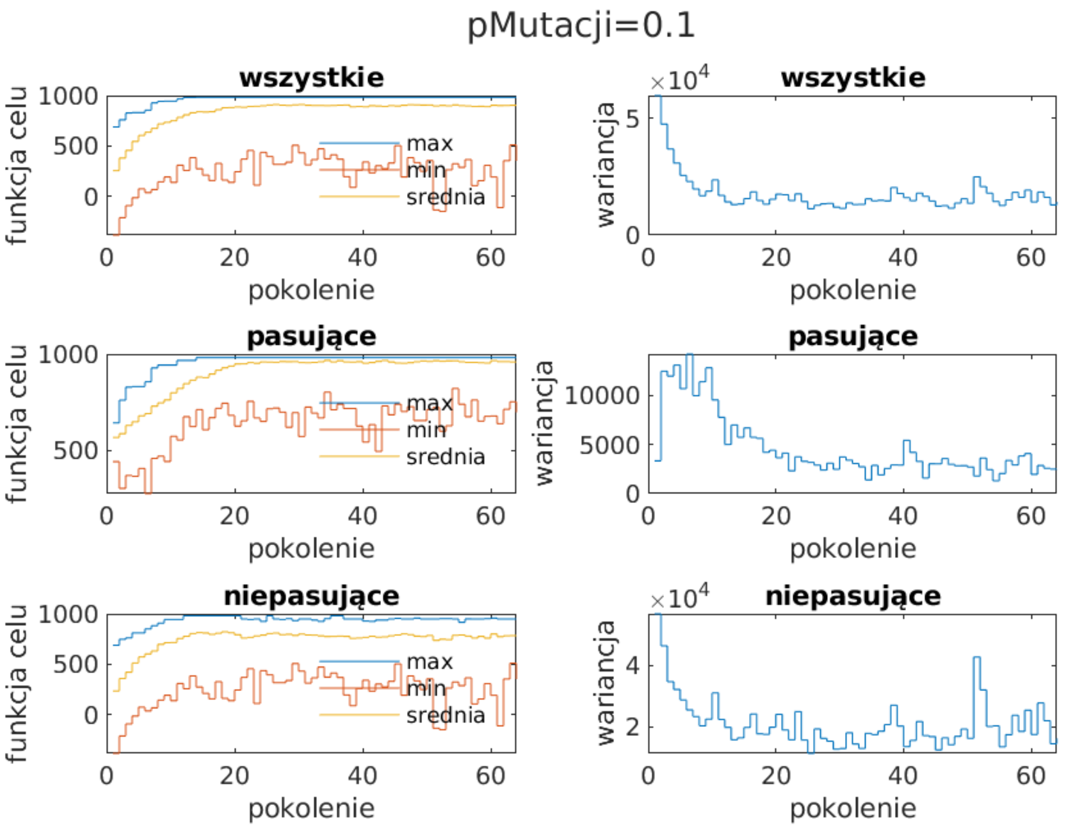
Wektory rozwiązań odpowiadające kolejnym wartościom prawdopodobieństwa mutacji:

— X = [0 1 1 0 1 1 0 0 1 0 0 1 0 1 0 0 0 0 0 0 1 0 1 0 0 1 0 1 1 0 1 1]
— X = [0 1 1 0 1 1 0 0 1 0 0 1 0 1 0 0 0 0 0 0 1 0 1 0 0 1 0 1 1 0 1 1]
— X = [0 1 1 0 1 1 0 0 1 0 0 1 0 1 0 0 0 0 0 0 1 0 1 0 0 1 0 1 1 0 1 1]
— X = [0 1 1 0 1 1 0 0 1 0 0 1 0 1 0 0 0 0 0 0 1 0 1 0 0 1 0 1 1 0 1 1]
— X = [0 1 1 0 1 1 0 0 1 0 0 1 0 1 0 0 0 0 0 0 1 0 1 0 0 1 0 1 1 0 1 1]
— X = [0 1 1 0 1 1 0 0 1 0 0 1 0 1 1 0 0 0 0 0 1 0 1 0 0 0 0 1 1 0 1 1]

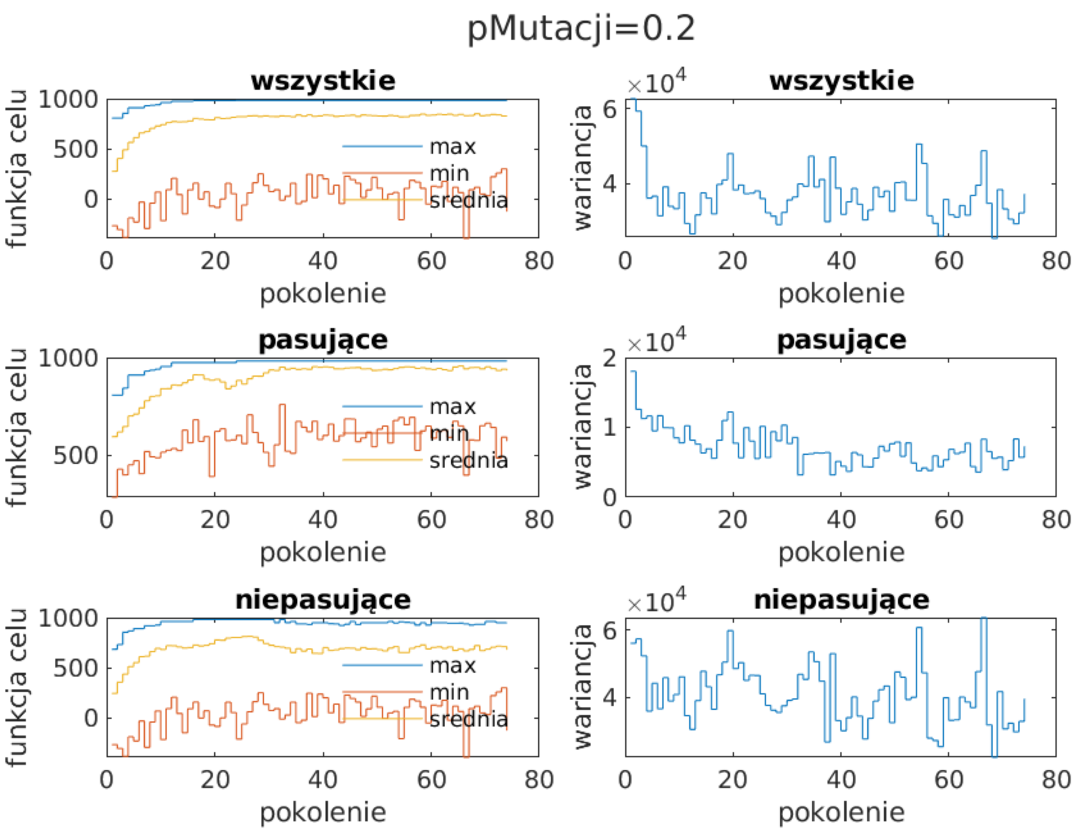
Zdecydowałem się na wybranie niskiego prawdopodobieństwa, równego 0,05. Dzięki temu uzyskujemy odpowiednio zmniejszającą się wariancję wraz z kolejnymi pokoleniami, zmienność wariancji nie jest chaotyczna. Wykresy pokazujące przebieg symulacji znajdują się poniżej.



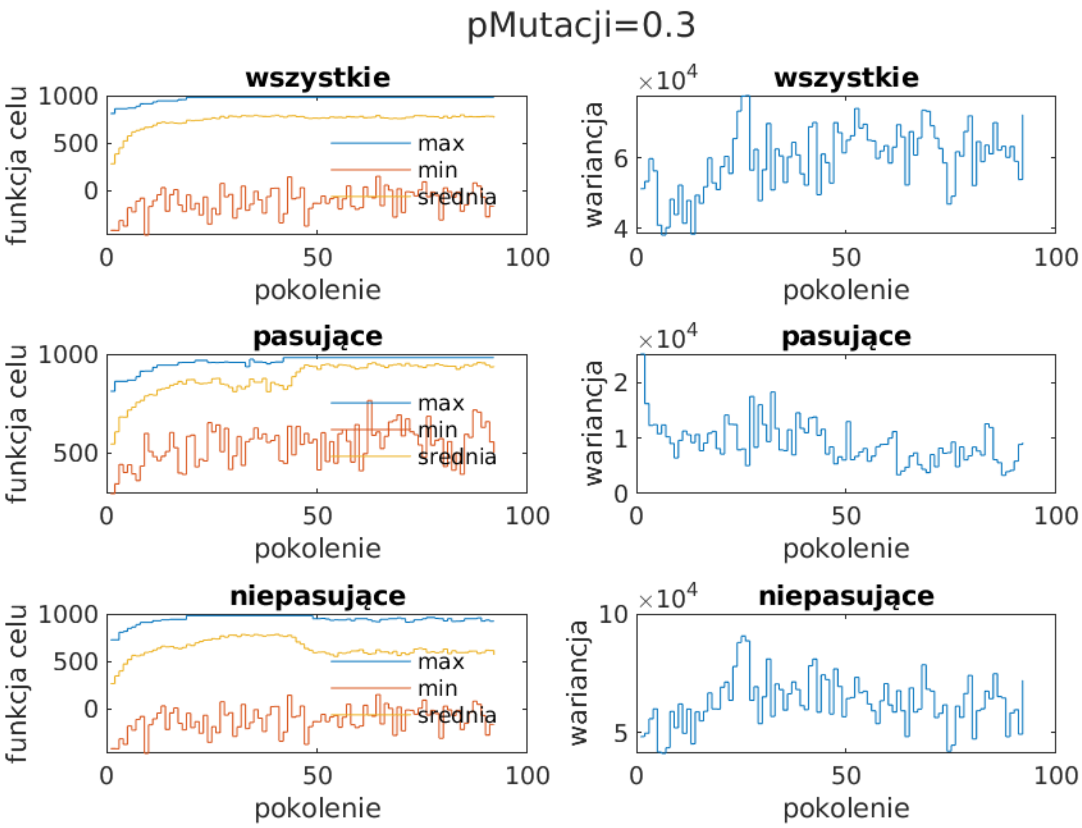
Rys. 2.5. Wyniki symulacji dla p mutacji = 0,05



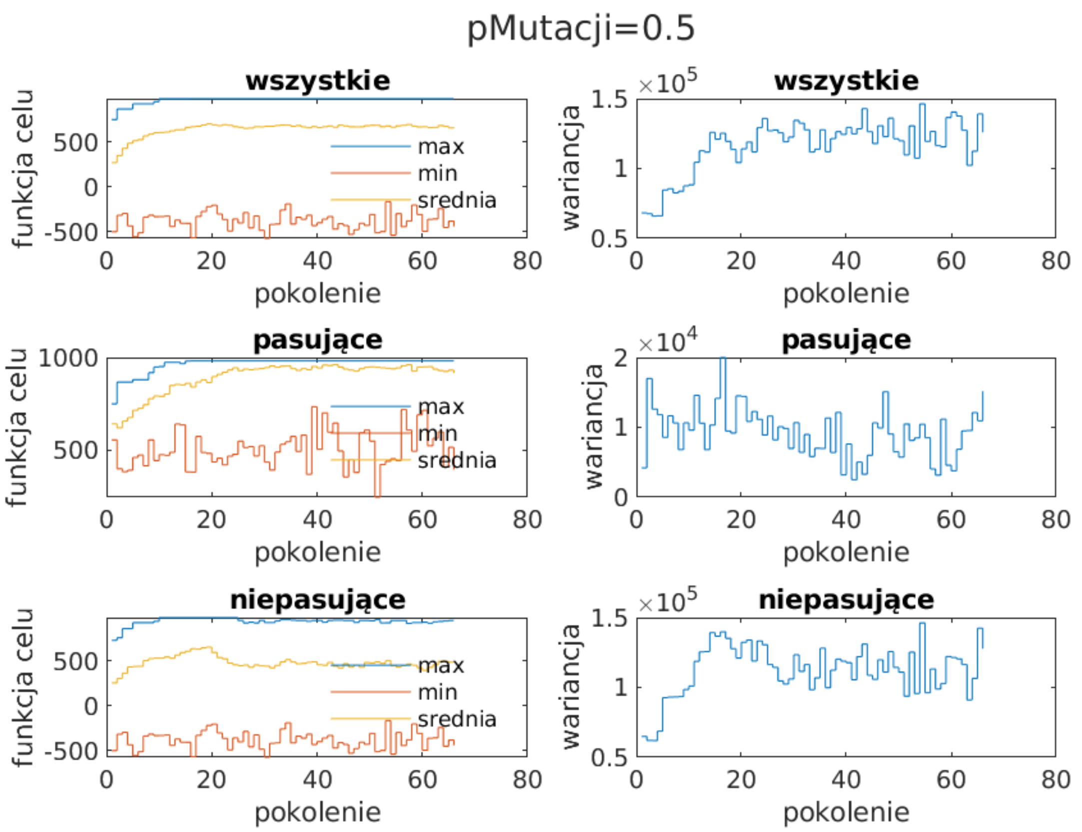
Rys. 2.6. Wyniki symulacji dla p mutacji $= 0{,}1$



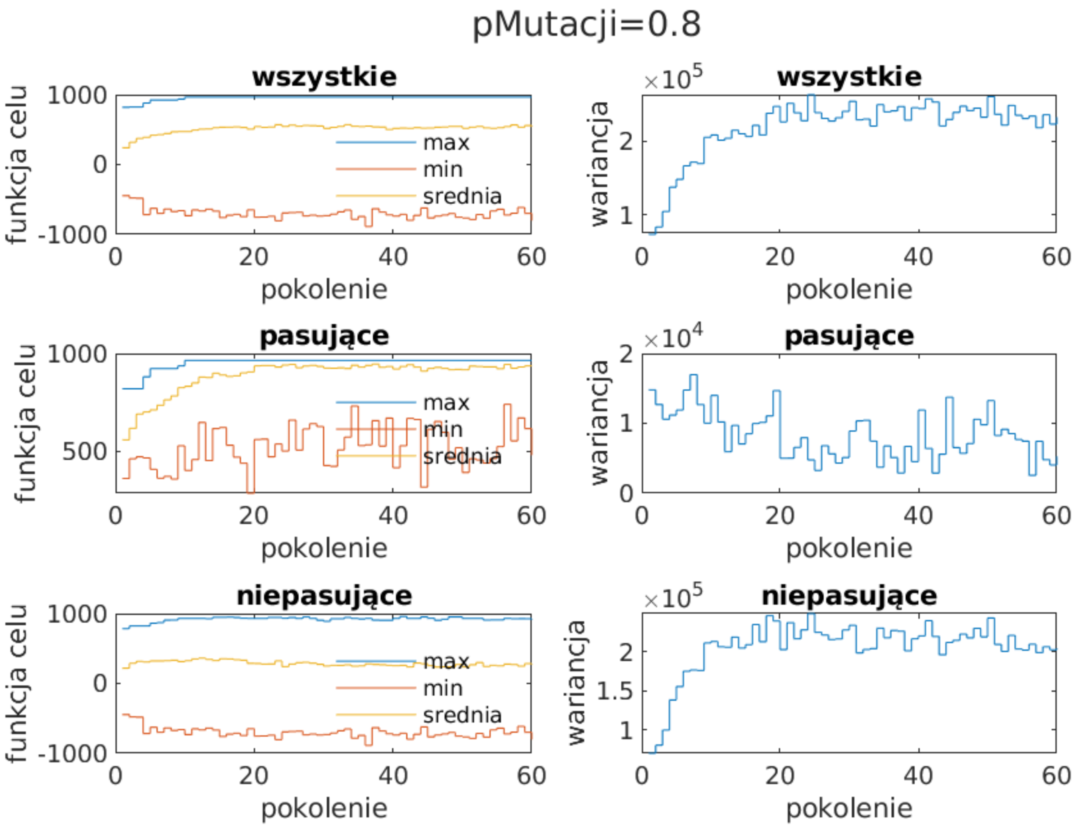
Rys. 2.7. Wyniki symulacji dla p mutacji $= 0{,}2$



Rys. 2.8. Wyniki symulacji dla p mutacji = 0,3



Rys. 2.9. Wyniki symulacji dla p mutacji = 0,5



Rys. 2.10. Wyniki symulacji dla p mutacji = 0,8

2.3. Dobór algorytmu krzyżowania

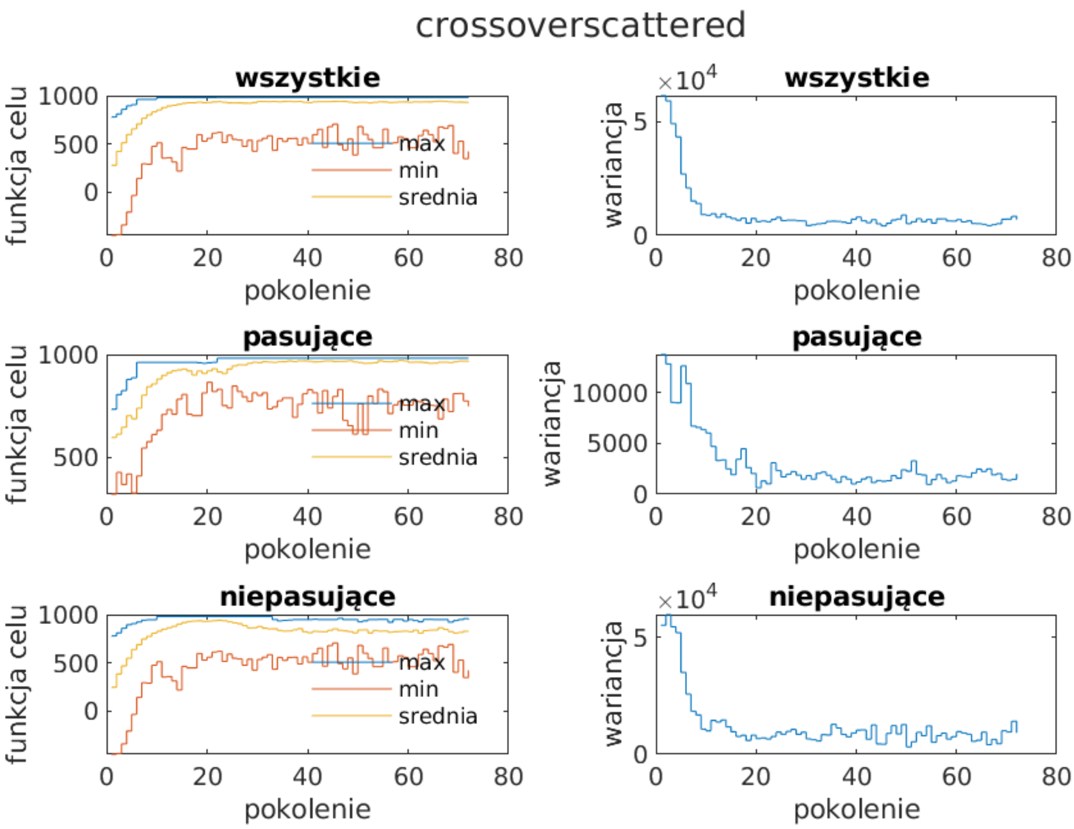
Wyniki optymalizacji:

metoda	y	p mutacji	p krzyżowania
crossoverscattered	982	0,05	0,8
crossoversinglepoint	955	0,05	0,8
crossovertwopoint	981	0,05	0,8

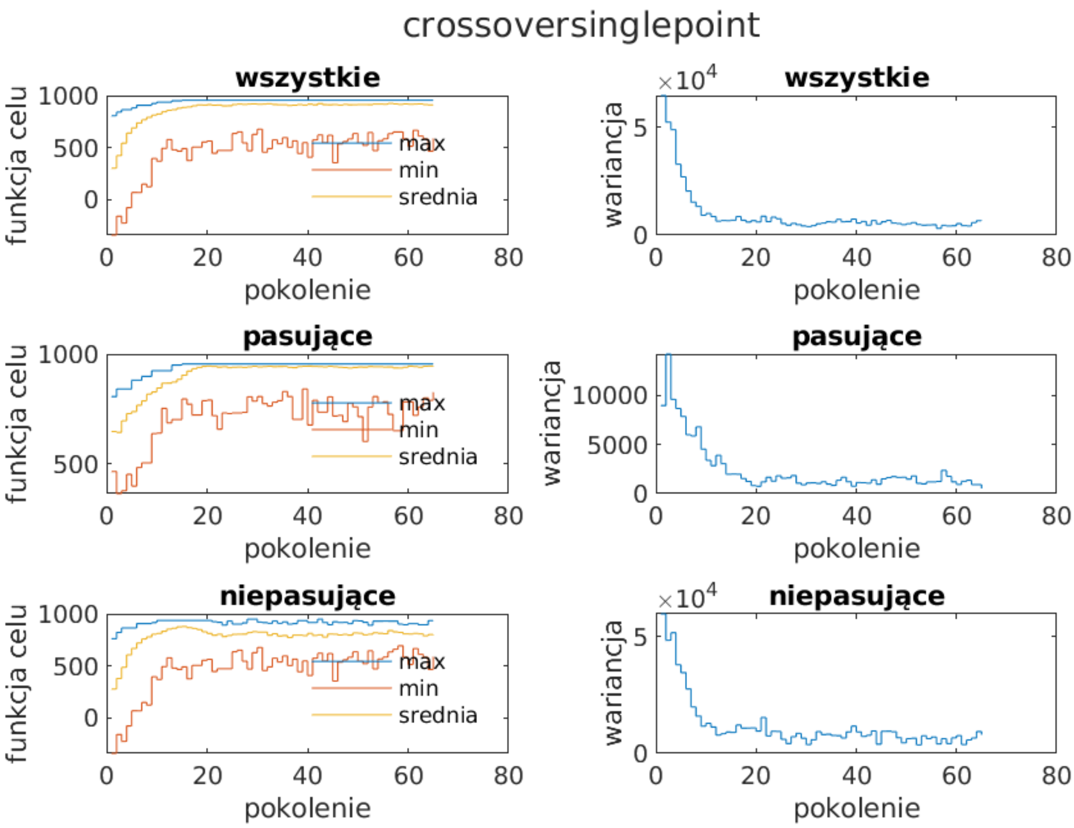
Wektory rozwiązań odpowiadające kolejnym algorytmom:

— X = [0 1 1 0 1 1 0 0 1 0 0 1 0 1 0 0 0 0 0 0 1 0 1 0 0 1 0 1 1 0 1 1]
— X = [0 1 1 0 1 1 0 0 1 0 0 1 0 1 1 0 0 0 0 0 1 1 1 0 0 1 0 0 1 0 1 1]
— X = [0 1 1 0 1 0 0 0 1 0 0 1 0 1 0 0 0 0 1 0 1 0 1 0 0 1 0 1 1 0 1 1]

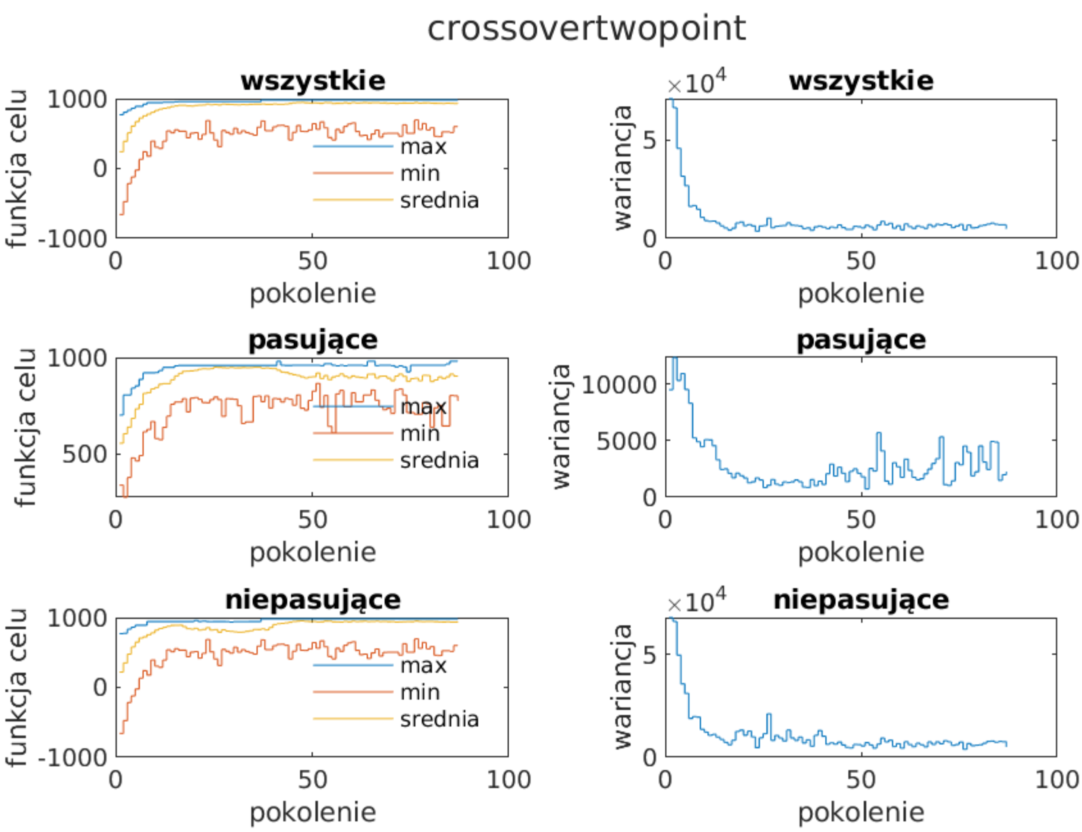
Zdecydowałem się na dalsze używanie crossoverscattered, pozostałe algorytmy zawiodły. Wykresy pokazujące przebieg symulacji znajdują się poniżej.



Rys. 2.11. Wyniki symulacji dla crossoverscattered



Rys. 2.12. Wyniki symulacji dla crossoversinglepoint



Rys. 2.13. Wyniki symulacji dla crossovertwopoint

2.4. Dobór metody selekcji

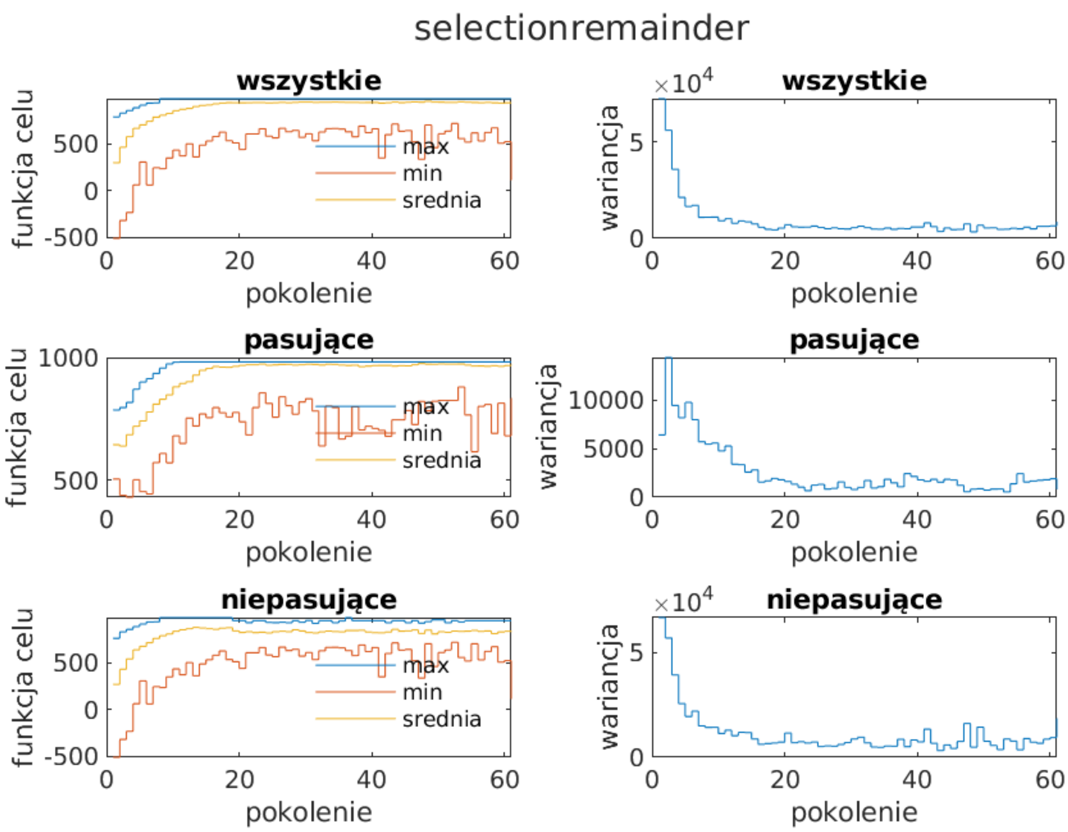
Wyniki optymalizacji:

metoda	y	p mutacji	p krzyżowania
selectionremainder	982	0,05	0,8
selectionroulette	981	0,05	0,8
selectionuniform	982	0,05	0,8
selectiontournament	982	0,05	0,8
selectionstochunif	982	0,05	0,8

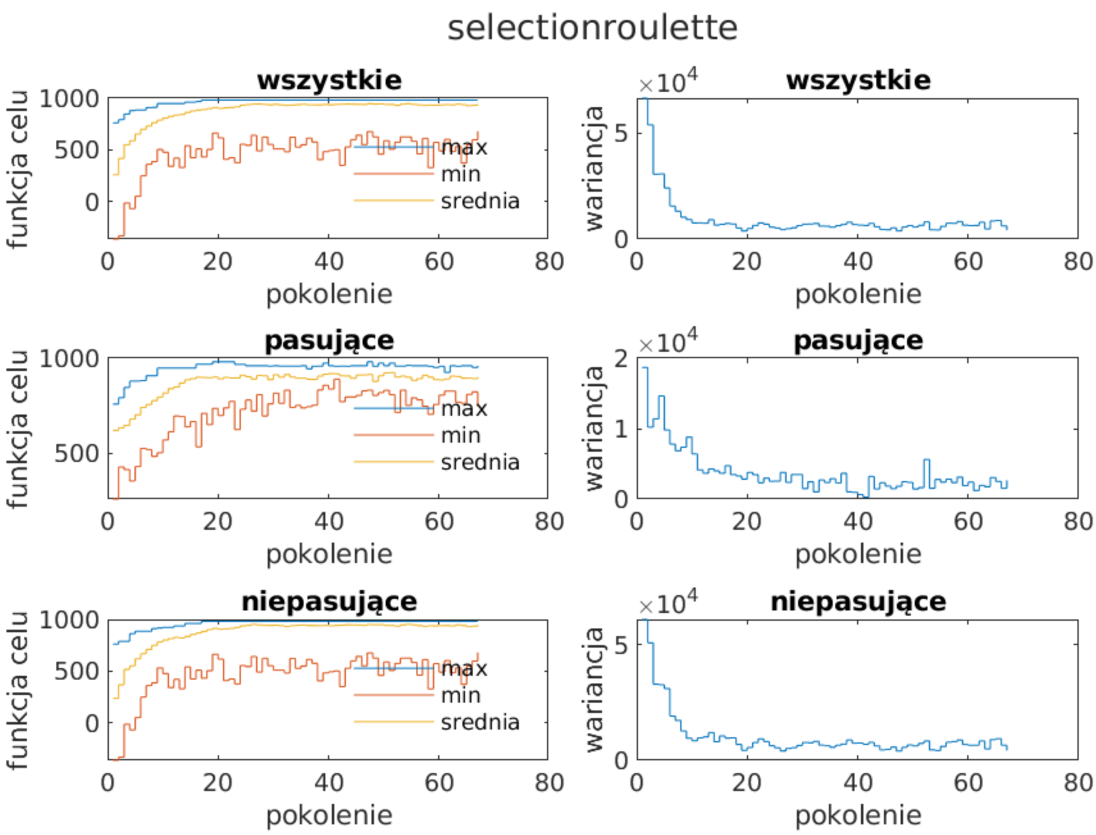
Wektory rozwiązań odpowiadające kolejnym metodom selekcji:

— X = [0 1 1 0 1 1 0 0 1 0 0 1 0 1 0 0 0 0 0 0 1 0 1 0 0 1 0 1 1 0 1 1]
— X = [0 1 1 0 1 0 0 0 1 0 0 1 0 1 0 0 0 0 1 0 1 0 1 0 0 1 0 1 1 0 1 1]
— X = [0 1 1 0 1 1 0 0 1 0 0 1 0 1 0 0 0 0 0 0 1 0 1 0 0 1 0 1 1 0 1 1]
— X = [0 1 1 0 1 1 0 0 1 0 0 1 0 1 0 0 0 0 0 0 1 0 1 0 0 1 0 1 1 0 1 1]
— X = [0 1 1 0 1 1 0 0 1 0 0 1 0 1 0 0 0 0 0 0 1 0 1 0 0 1 0 1 1 0 1 1]

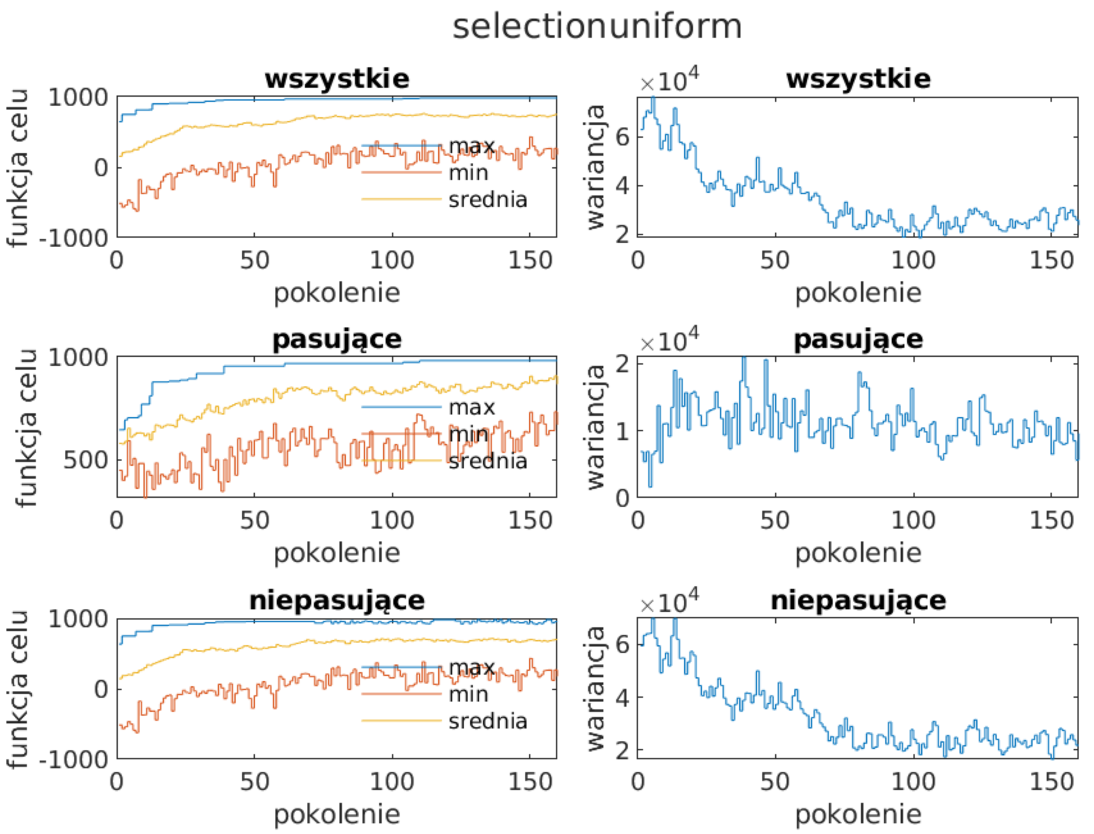
Zdecydowałem się na dalsze używanie selectiontournament, nie posiada on wad metody ruletkowej ani rankingowej (jest dużo lepszy w zachowywaniu różnorodności populacji), a poza tym jego działanie jest dla mnie najbardziej interesujące (na przykład ze względu na najbardziej monotoniczny rozkład wariancji elementów pasujących w danym pokoleniu).



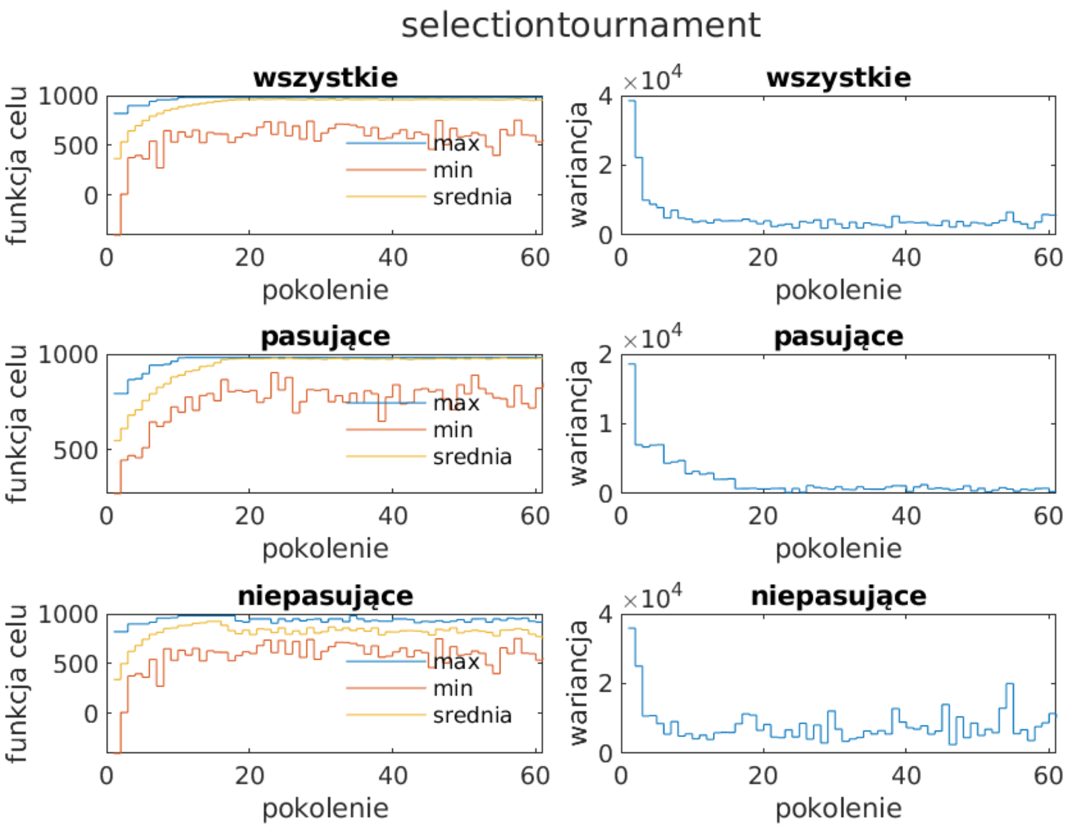
Rys. 2.14. Wyniki symulacji dla selectionremainder



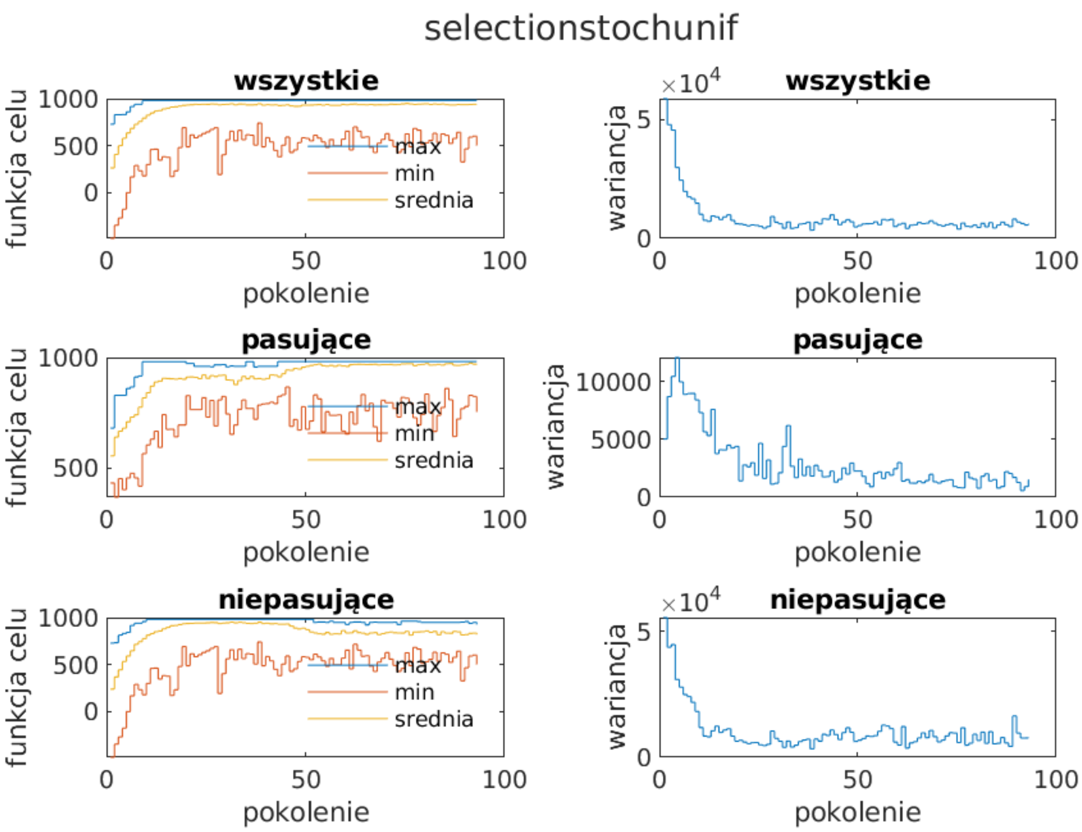
Rys. 2.15. Wyniki symulacji dla selectionroulette



Rys. 2.16. Wyniki symulacji dla selectionuniform



Rys. 2.17. Wyniki symulacji dla selectiontournament



Rys. 2.18. Wyniki symulacji dla selectionstochunif

2.5. Dobór prawdopodobieństwa krzyżowania

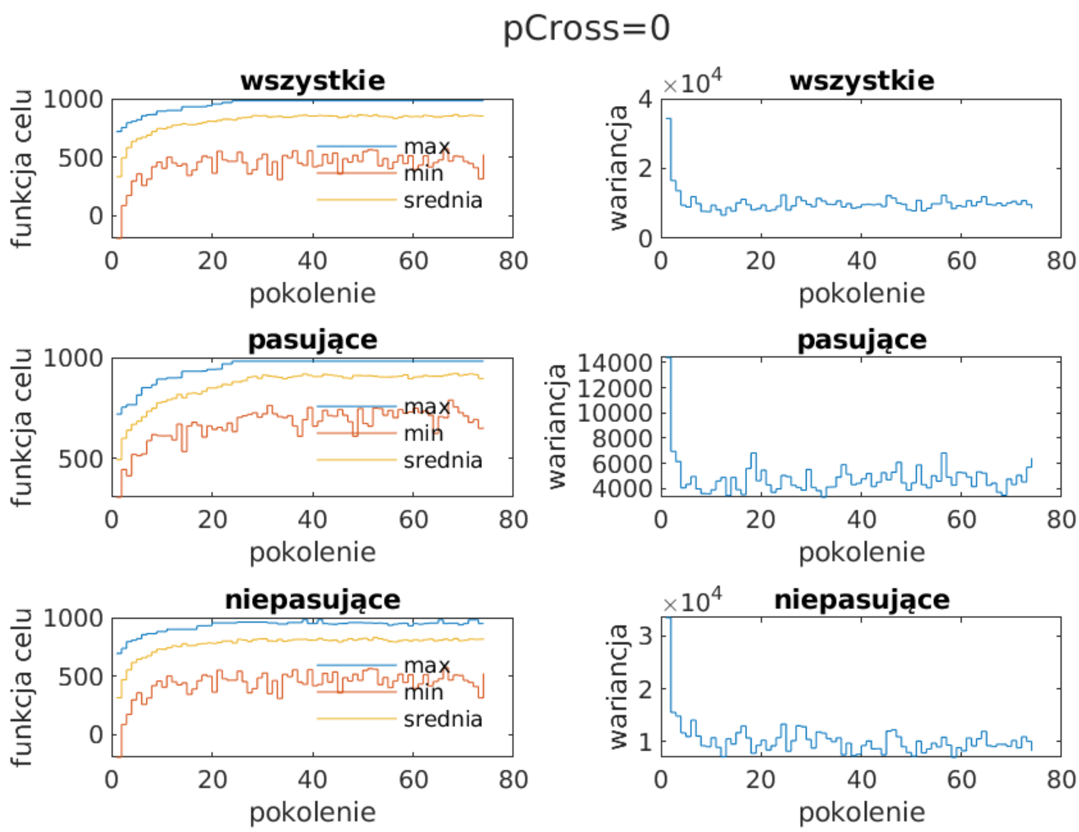
Wyniki optymalizacji:

p krzyżowania	y	p mutacji
0	982	0,05
0,1	982	0,05
0,3	982	0,05
0,5	982	0,05
0,8	981	0,05
0,9	982	0,05
0,95	982	0,05

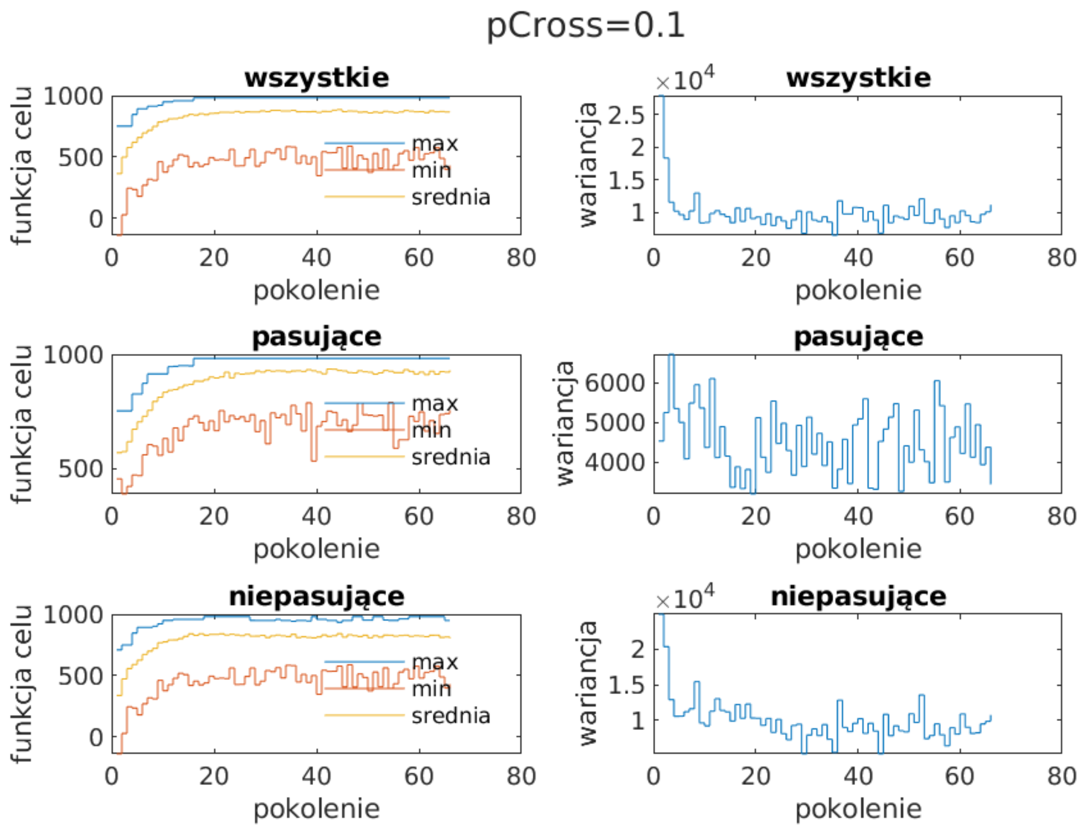
Wektory rozwiązań odpowiadające kolejnym wartościom prawdopodobieństwa krzyżowania:

— X = [0 1 1 0 1 1 0 0 1 0 0 1 0 1 0 0 0 0 0 0 1 0 1 0 0 1 0 1 1 0 1 1]
— X = [0 1 1 0 1 1 0 0 1 0 0 1 0 1 0 0 0 0 0 0 1 0 1 0 0 1 0 1 1 0 1 1]
— X = [0 1 1 0 1 1 0 0 1 0 0 1 0 1 0 0 0 0 0 0 1 0 1 0 0 1 0 1 1 0 1 1]
— X = [0 1 1 0 1 1 0 0 1 0 0 1 0 1 0 0 0 0 0 0 1 0 1 0 0 1 0 1 1 0 1 1]
— X = [0 1 1 0 1 0 0 0 1 0 0 1 0 1 0 0 0 0 1 0 1 0 1 0 0 1 0 1 1 0 1 1]
— X = [0 1 1 0 1 1 0 0 1 0 0 1 0 1 0 0 0 0 0 0 1 0 1 0 0 1 0 1 1 0 1 1]
— X = [0 1 1 0 1 1 0 0 1 0 0 1 0 1 0 0 0 0 0 0 1 0 1 0 0 1 0 1 1 0 1 1]

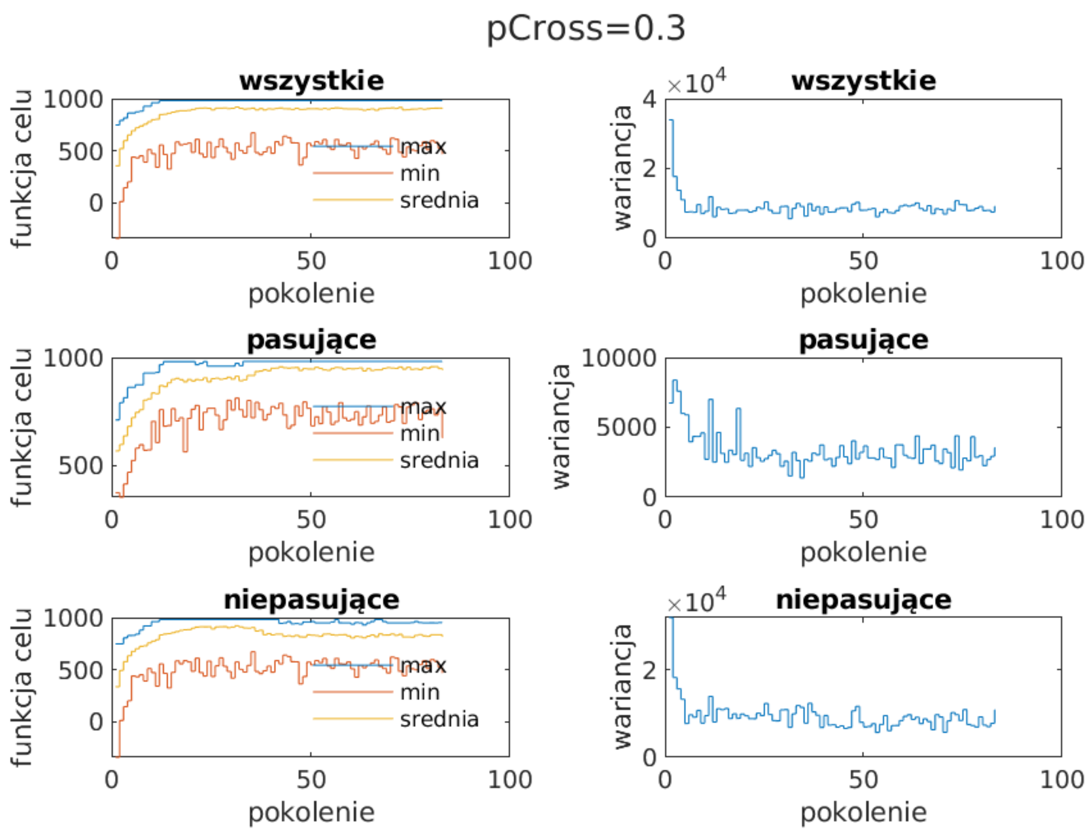
Zdecydowałem się na wybranie 0,9. Wysokie prawdopodobieństwo krzyżowania sprzyja szybszej zbieżności do optymalnego rozwiązania. Wykresy pokazujące przebieg symulacji znajdują się poniżej.



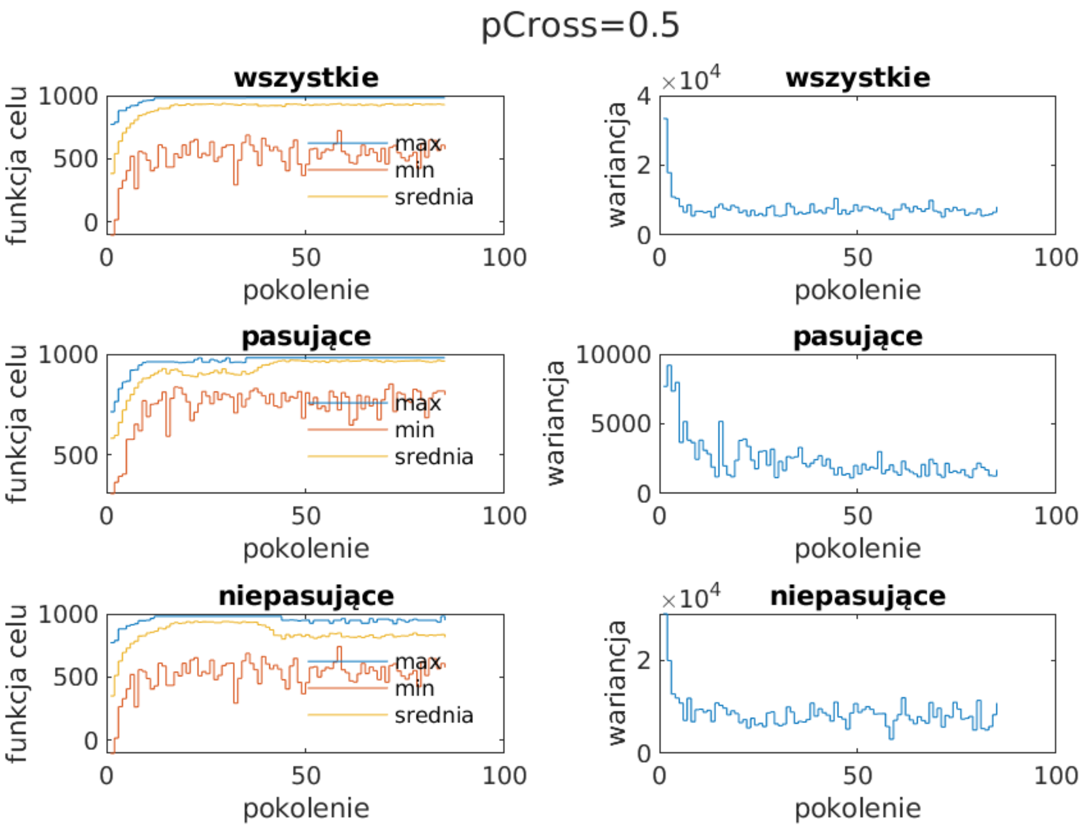
Rys. 2.19. Wyniki symulacji dla p krzyżowania = 0



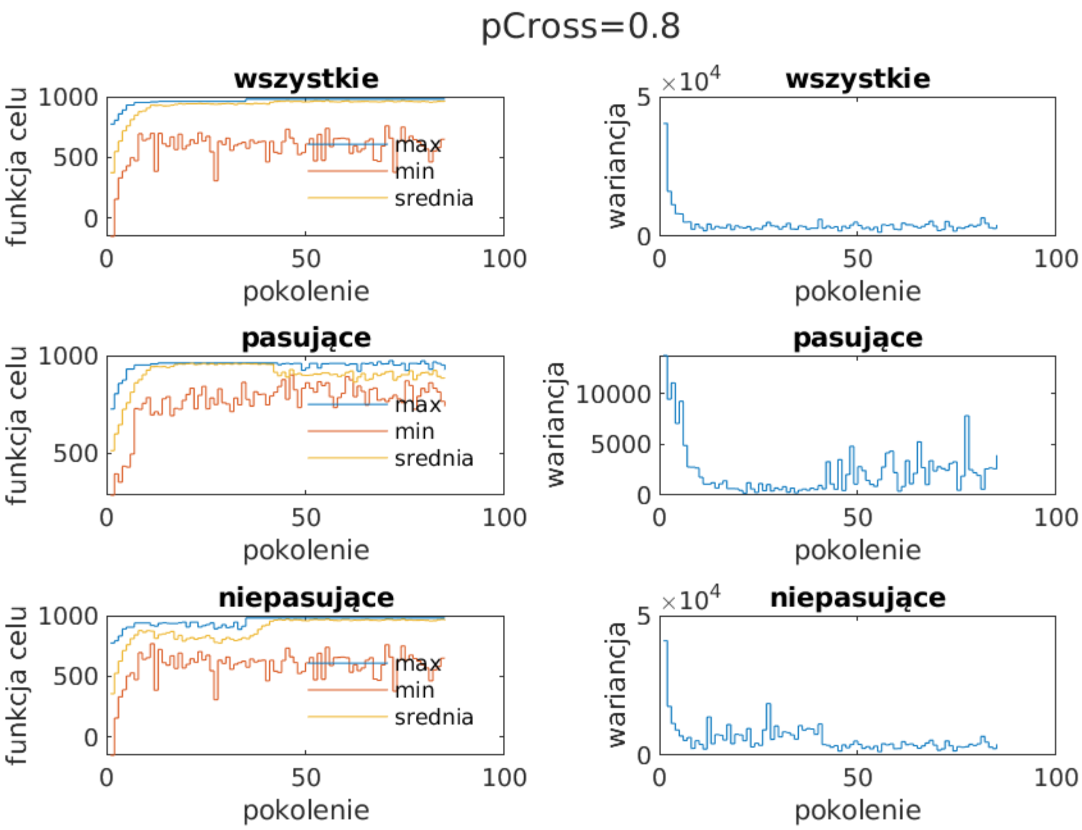
Rys. 2.20. Wyniki symulacji dla p krzyżowania = 0,1



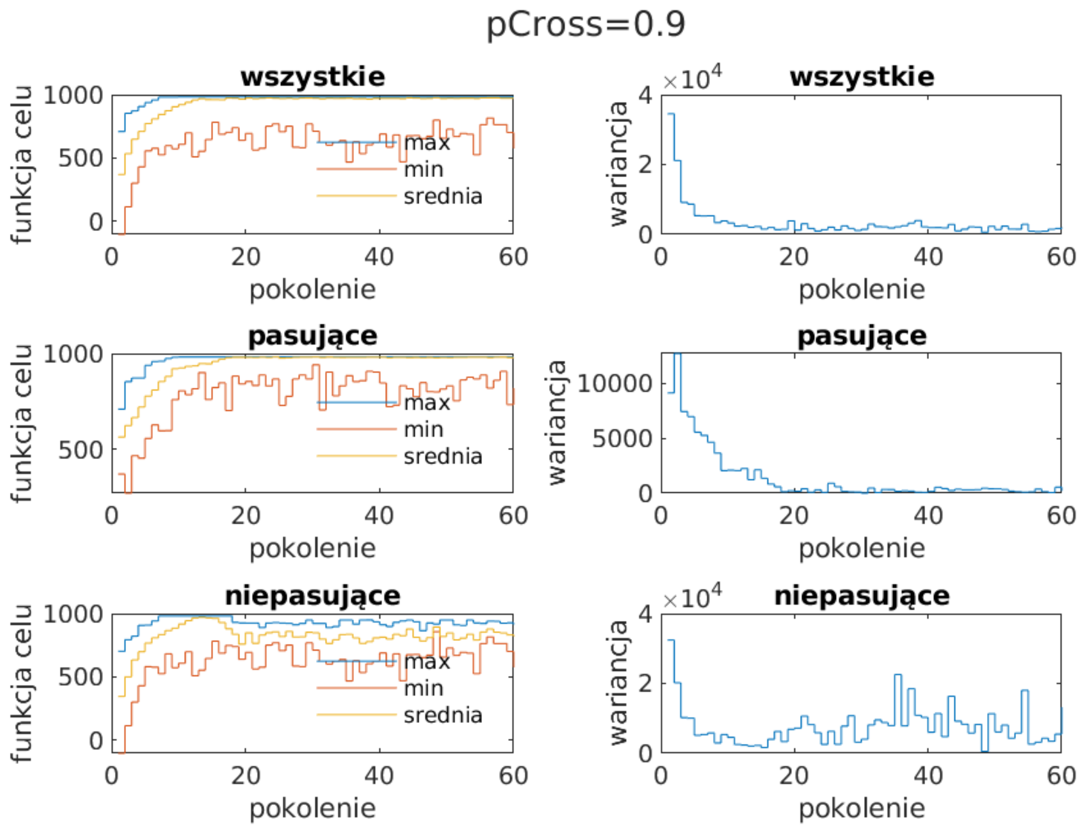
Rys. 2.21. Wyniki symulacji dla p krzyżowania = 0,3



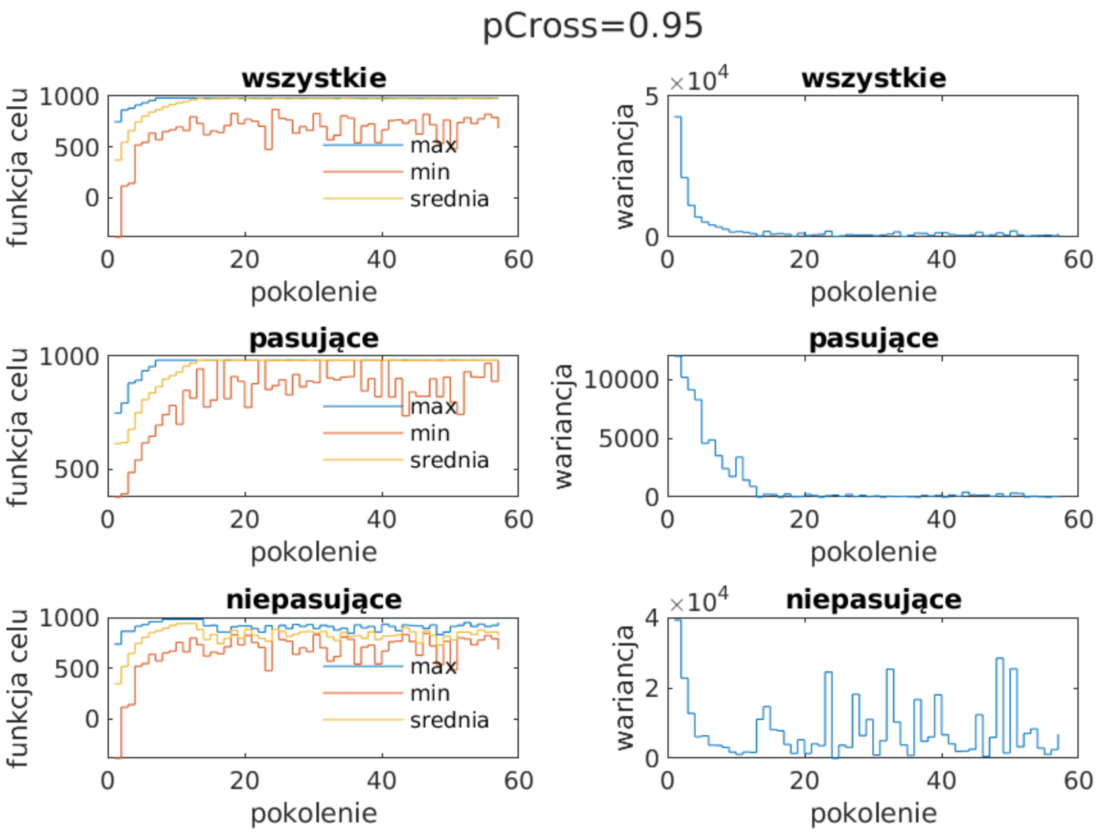
Rys. 2.22. Wyniki symulacji dla p krzyżowania $= 0{,}5$



Rys. 2.23. Wyniki symulacji dla p krzyżowania $= 0{,}8$



Rys. 2.24. Wyniki symulacji dla p krzyżowania = 0,9



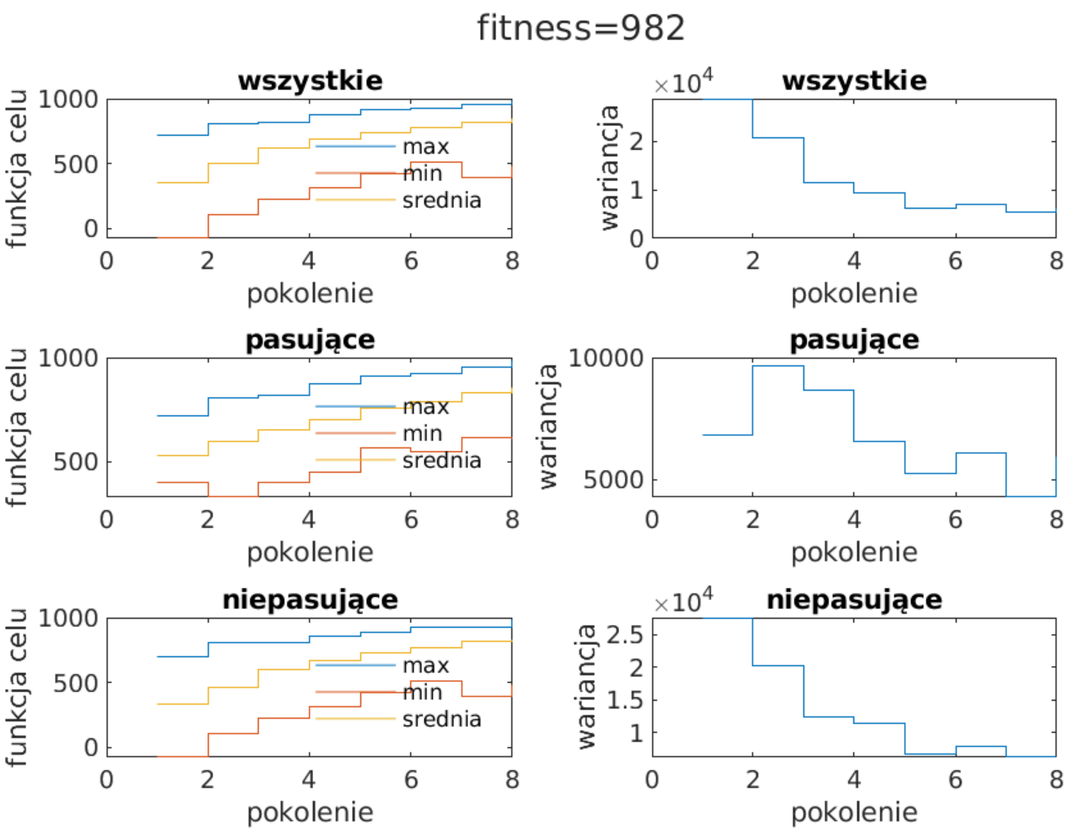
Rys. 2.25. Wyniki symulacji dla p krzyżowania = 0,95

2.6. Modyfikacja kryterium zatrzymania GA

Ponieważ znamy największą możliwą wartość funkcji, zdecydowałem się na wykorzystanie jej do wprowadzenia dodatkowego kryterium końca algorytmu genetycznego. Wyniki symulacji znajdują się poniżej.

y	p krzyżowania	p mutacji
982	0,9	0,05

Wektor rozwiązań X = [0 1 1 0 1 1 0 0 1 0 0 1 0 1 0 0 0 0 0 0 1 0 1 0 0 1 0 1 1 0 1 1].
Jak widać algorytm błyskawicznie, w ciągu ośmiu pokoleń, jest w stanie osiągnąć optimum globalne. To kryterium stopu nie było wcześniej wykorzystywane, ponieważ ograniczyłaby drastycznie liczbę pokoleń, co za tym idzie nie byłoby dobrze udokumentowanych wpływów różnych parametrów na działanie GA.



Rys. 2.26. Wyniki symulacji dla zmienionego kryterium stopu

2.7. Znalezione optimum globalne

W wyniku przeprowadzonego eksperymentu znaleziono następujące maksimum globalne:

X = [0 1 1 0 1 1 0 0 1 0 0 1 0 1 0 0 0 0 0 0 1 0 1 0 0 1 0 1 1 0 1 1]
wartość funkcji celu = 982

3. Optymalizacja dla $n = 64$

W każdym z przypadków liczność generacji to 200. Na rysunkach zawarto wykresy wskaźników maksimum, minimum, średniej oraz wariancji wartości funkcji celu zarówna dla wszystkich zestawów przedmiotów, jak i dla przedmiotów o dopuszczalnej wadze, jak i tych, które przekroczyły dopuszczalną wagę w danym pokoleniu.

3.1. Strojenie parametru α

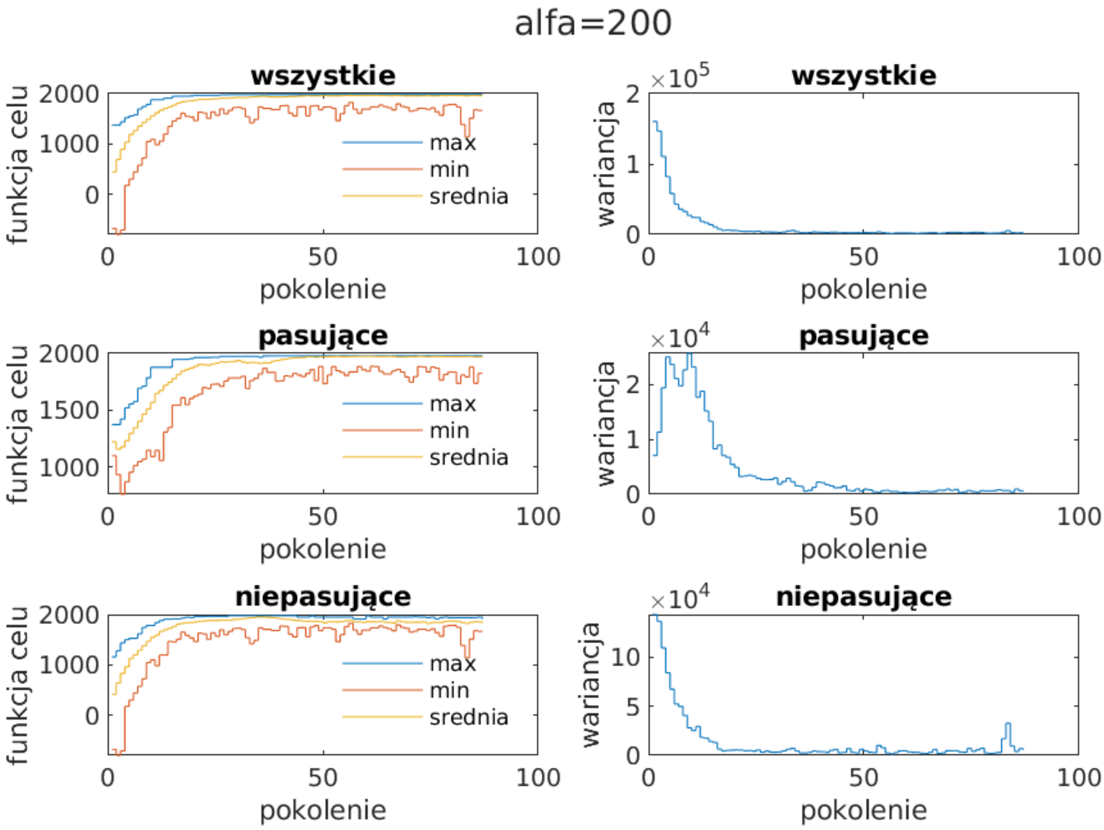
W przypadku strojenia parametru α jedyne sensowne wartości są dla $\alpha \geqslant 200$, ponieważ dla mniejszych wartości α kara za przekroczenie dopuszczalnej wagi jest zbyt mała. Wyniki optymalizacji:

α	y	p mutacji	p krzyżowania
200	1979	0,01	0,8
300	1956	0,01	0,8
500	1968	0,01	0,8
700	1962	0,01	0,8

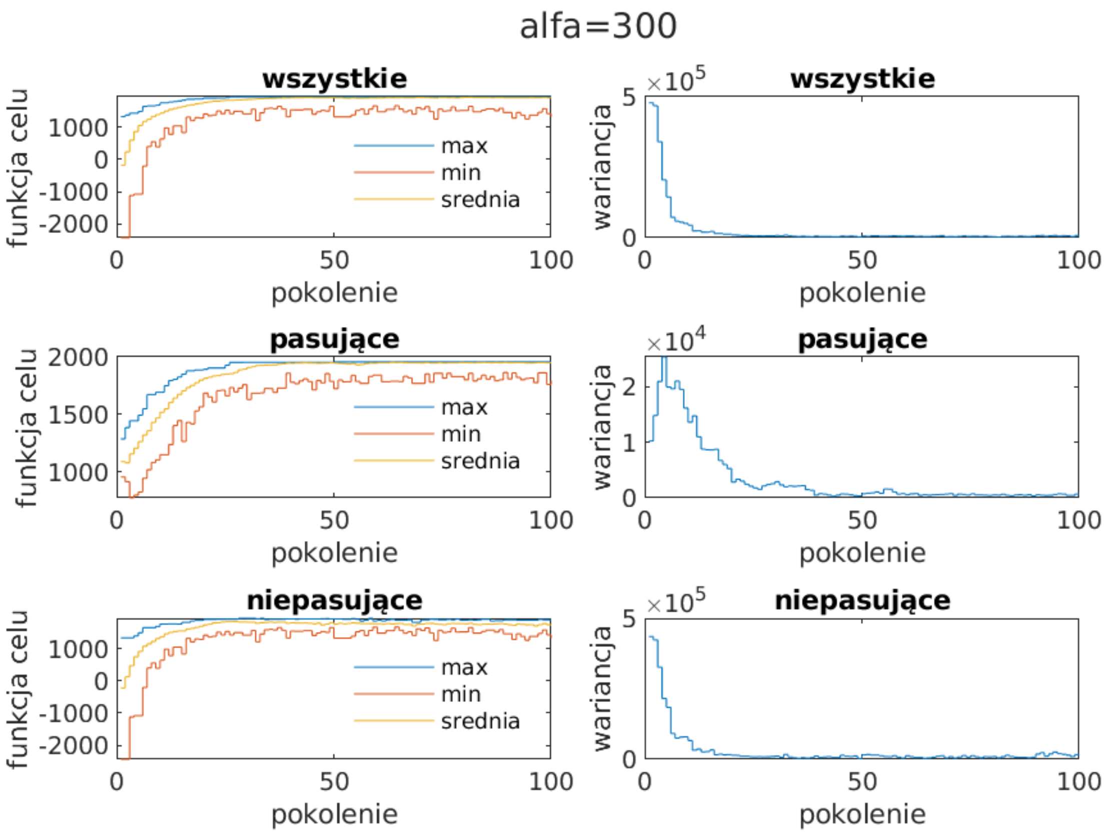
Wektory rozwiązań odpowiadające kolejnym wartościom α:

— X = [0 1 0 1 0 0 1 0 1 1 1 1 0 1 1 0 0 0 1 0 0 0 1 1 0 0 1 0 1 1 0 0 1 0 1 0 0 1 0 1 1 0 0 0 0 0 1 0 1 1 1 1 1 0 0 1 0 0 1 0 0 1 0 0]
— X = [0 1 0 1 0 0 0 0 1 0 1 1 0 1 1 0 0 0 1 0 1 1 1 1 0 0 1 0 1 0 0 0 1 0 0 0 0 1 0 1 1 0 0 1 0 0 1 0 1 1 1 1 1 0 0 1 0 0 1 0 0 1 0 0]
— X = [0 1 0 1 0 0 0 0 1 1 1 1 0 0 1 0 0 0 1 0 0 1 1 1 0 1 1 0 1 1 0 0 1 0 1 0 0 1 0 1 1 0 0 0 0 0 1 0 1 1 1 1 1 0 0 0 0 0 1 0 0 1 0 0]
— X = [0 1 0 1 0 0 0 0 1 1 1 1 0 1 1 0 0 0 1 0 1 1 1 1 0 1 1 0 1 0 0 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 1 0 1 1 1 1 1 0 0 1 0 0 1 0 0 1 0 0]

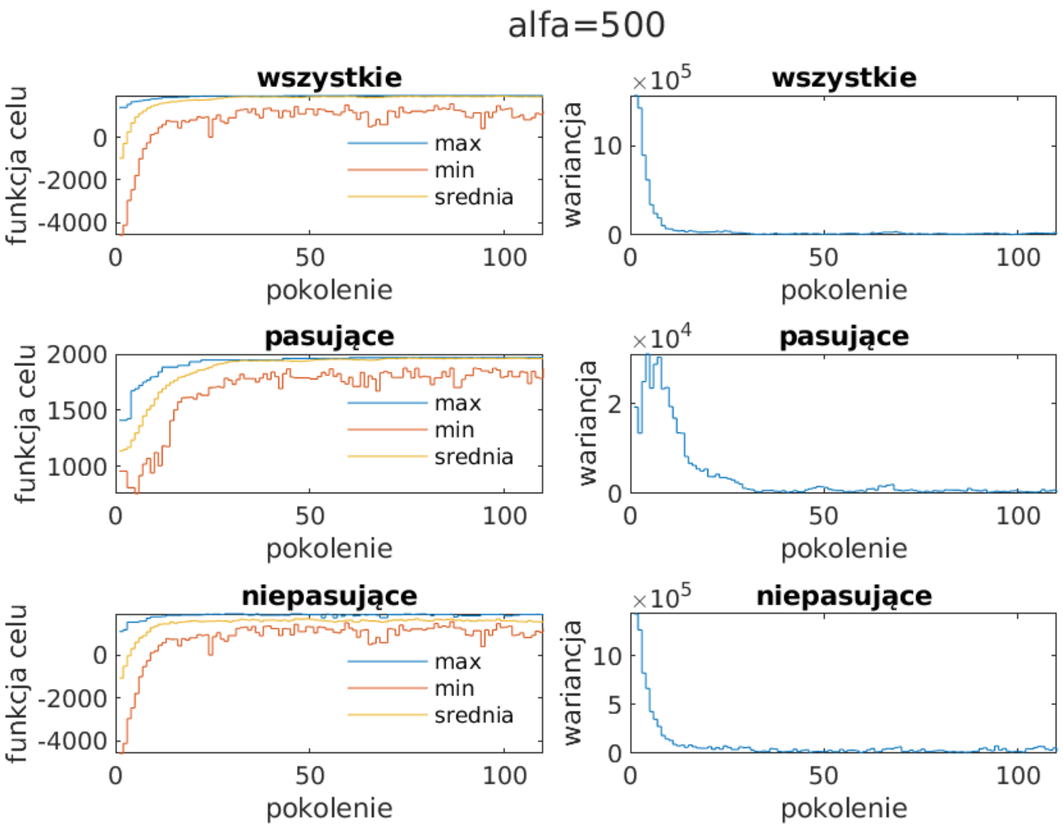
Zdecydowałem się na wybranie $\alpha = 200$. Dzięki jak najmniejszej karze różnorodność populacji nie jest mocno przetrzebiana, co sprzyja rozwiązaniu zwłaszcza problemu o 2 razy większym wymiarze, niż przy $n = 32$. Wykresy pokazujące przebieg symulacji znajdują się poniżej.



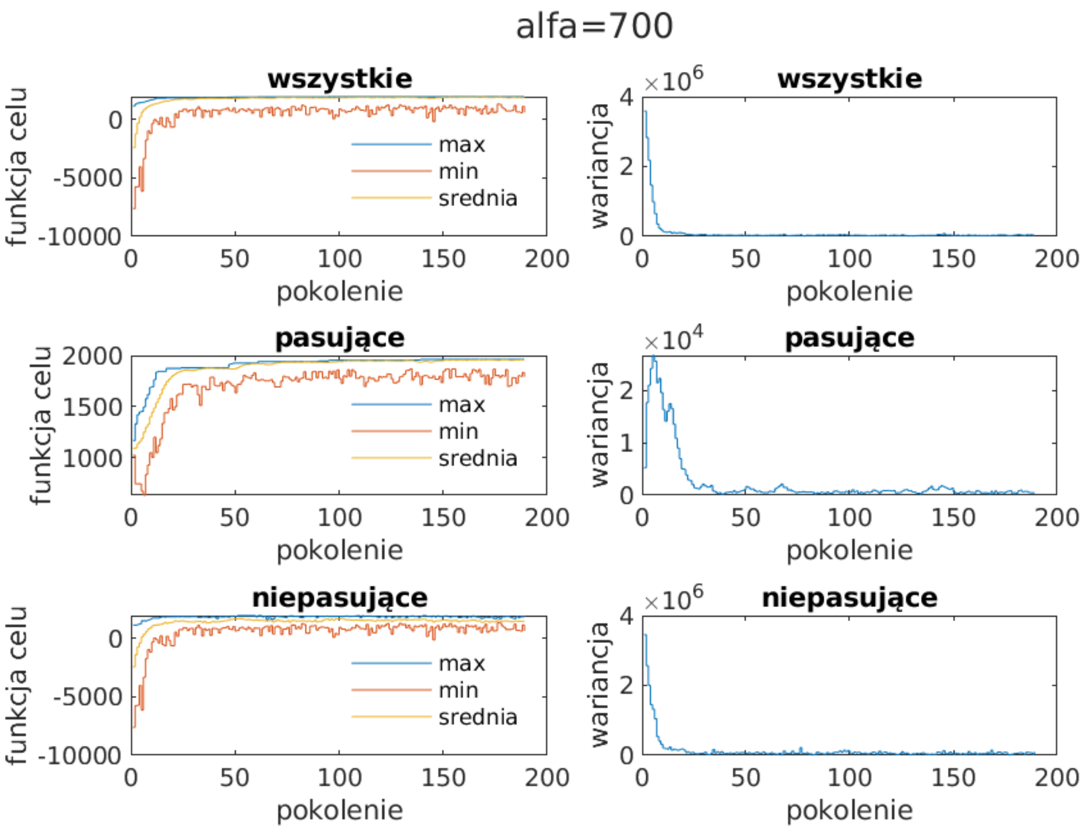
Rys. 3.1. Wyniki symulacji dla $\alpha = 200$



Rys. 3.2. Wyniki symulacji dla $\alpha = 300$



Rys. 3.3. Wyniki symulacji dla $\alpha = 500$



Rys. 3.4. Wyniki symulacji dla $\alpha = 700$

3.2. Dobór prawdopodobieństwa mutacji

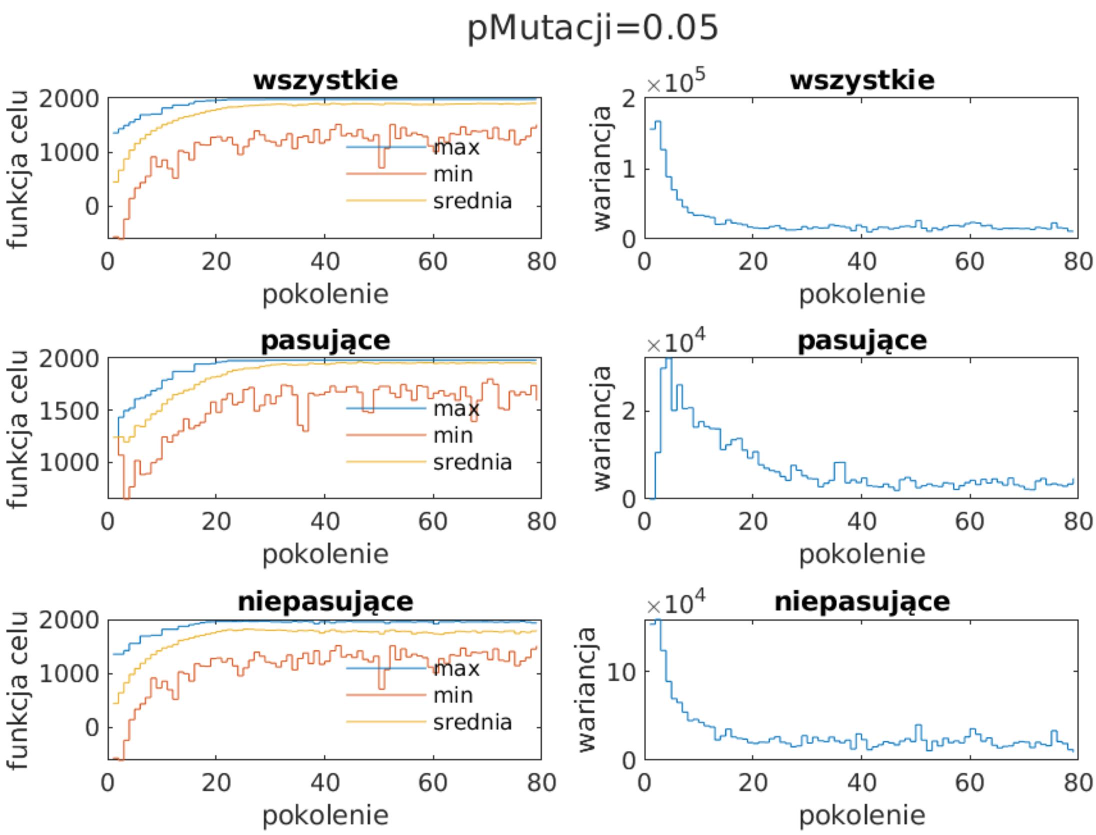
Wyniki optymalizacji:

p mutacji	y	p krzyżowania
0,05	1980	0,8
0,1	1980	0,8
0,2	1975	0,8
0,3	1952	0,8
0,5	1977	0,8
0,8	1969	0,8

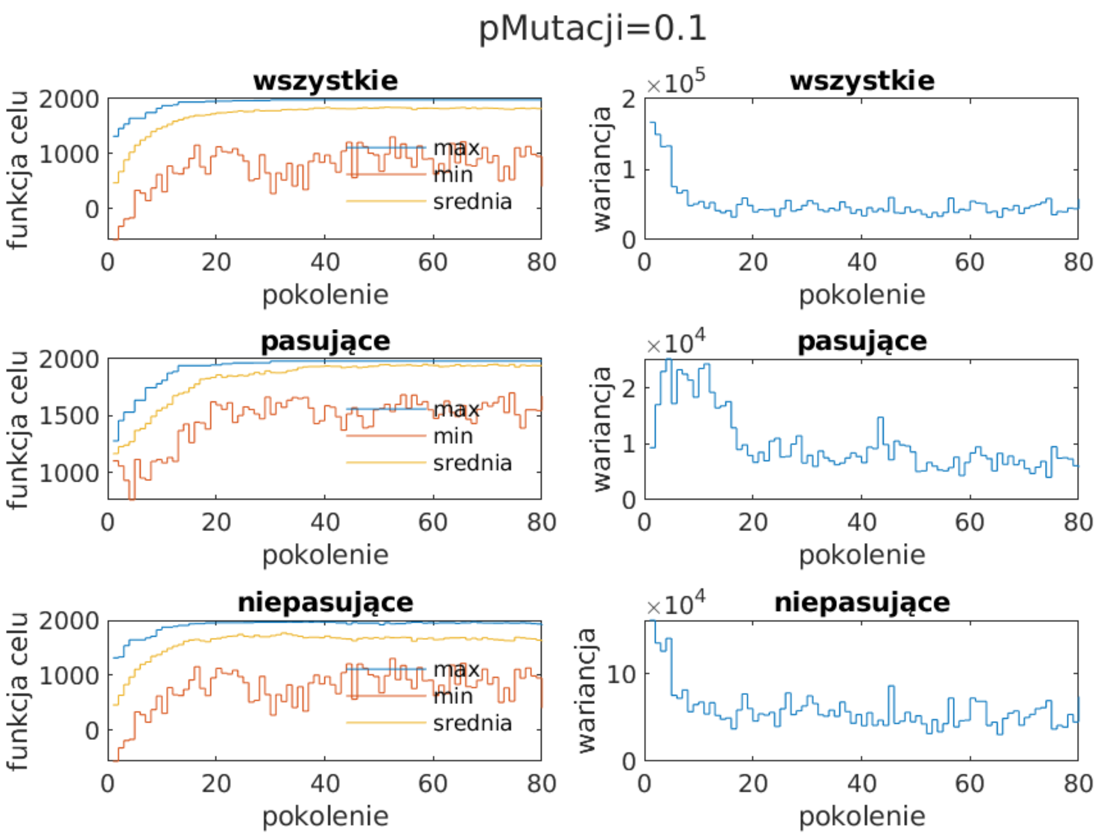
Wektory rozwiązań odpowiadające kolejnym wartościom prawdopodobieństwa mutacji:

— X = [0 1 0 1 0 0 0 0 1 1 1 1 0 1 1 0 0 0 1 0 0 1 1 1 0 1 1 0 1 1 0 0 1 0 0 0 0 1 0 1 1 0 0 0 0 0 1 0 1 1 1 0 1 0 0 1 0 0 1 0 0 1 0 0]
— X = [0 1 0 1 0 0 0 0 1 1 1 1 0 1 1 0 0 0 1 0 0 1 1 1 0 1 1 0 1 1 0 0 1 0 0 0 0 1 0 1 1 0 0 0 0 0 1 0 1 1 1 0 1 0 0 1 0 0 1 0 0 1 0 0]
— X = [0 1 0 1 0 0 0 0 1 1 1 1 0 1 1 0 0 0 1 0 0 1 1 1 0 1 1 0 1 1 0 0 1 0 0 0 0 1 0 1 1 0 0 0 0 0 1 1 1 1 1 1 1 0 0 0 0 0 1 0 0 1 0 0]
— X = [0 1 0 1 0 0 0 0 1 1 1 1 0 1 1 0 0 0 1 0 0 0 1 1 0 0 1 0 1 1 0 0 1 0 1 0 0 1 0 1 1 0 0 0 0 0 1 0 0 1 1 1 1 0 1 1 0 0 1 0 0 1 0 0]
— X = [0 1 0 1 0 0 0 0 1 0 1 1 0 1 1 0 0 0 1 0 0 1 1 1 0 0 1 1 1 1 0 0 1 0 1 0 0 1 0 1 1 0 0 0 0 0 1 0 1 1 1 1 1 0 0 1 0 0 1 0 0 1 0 0]
— X = [0 1 0 1 0 0 0 0 1 1 1 1 0 1 1 0 0 0 1 0 0 1 1 1 0 1 1 0 1 1 0 0 1 0 0 0 0 1 0 1 1 0 0 1 0 0 0 0 1 1 1 1 1 0 0 0 0 0 1 0 0 1 0 0]

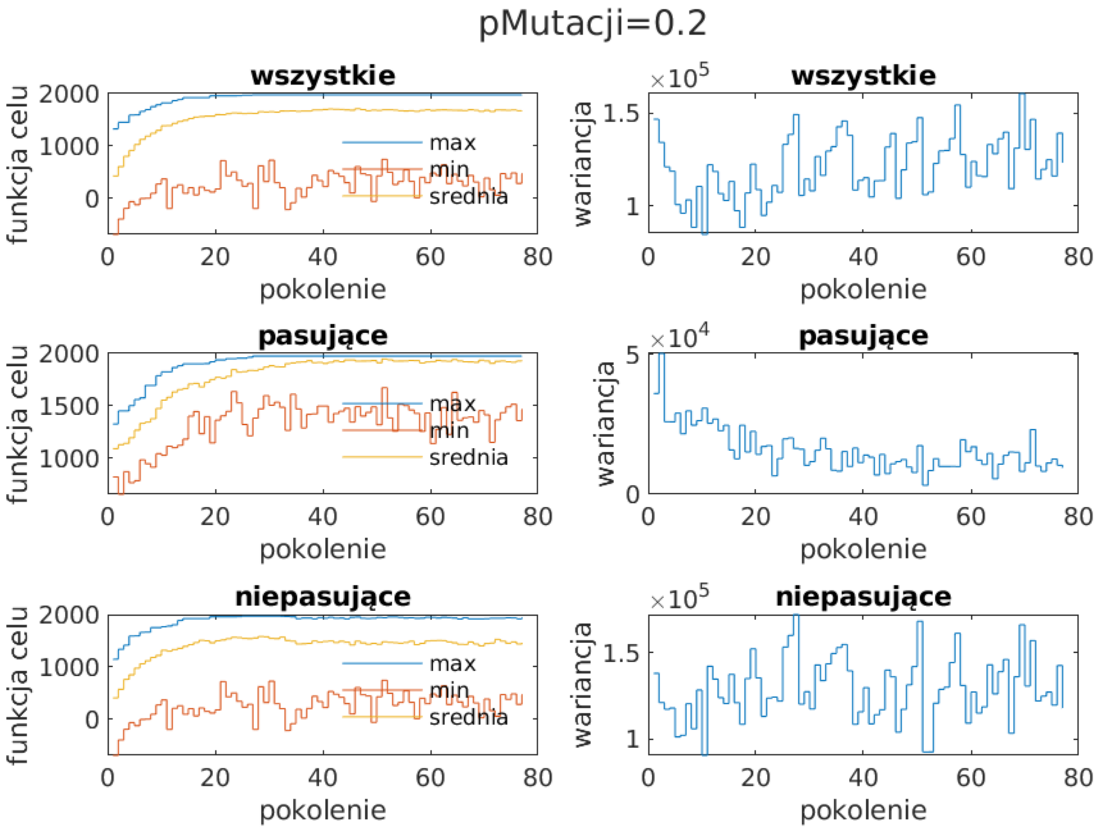
Zdecydowałem się na wybranie niskiego prawdopodobieństwa, równego 0,05. Dzięki temu uzyskujemy odpowiednio zmniejszającą się wariancję wraz z kolejnymi pokoleniami, zmienność wariancji nie jest chaotyczna. Sprzyja to więc znalezieniu optimum, algorytm nie ma takiej tendencji do ugrzęźnięcia w mimum lokalnym, stara się znaleźć minimum globalne. Zwiększanie prawdopodobieństwa nie jest dobrym rozwiązaniem. Wykresy pokazujące przebieg symulacji znajdują się poniżej.



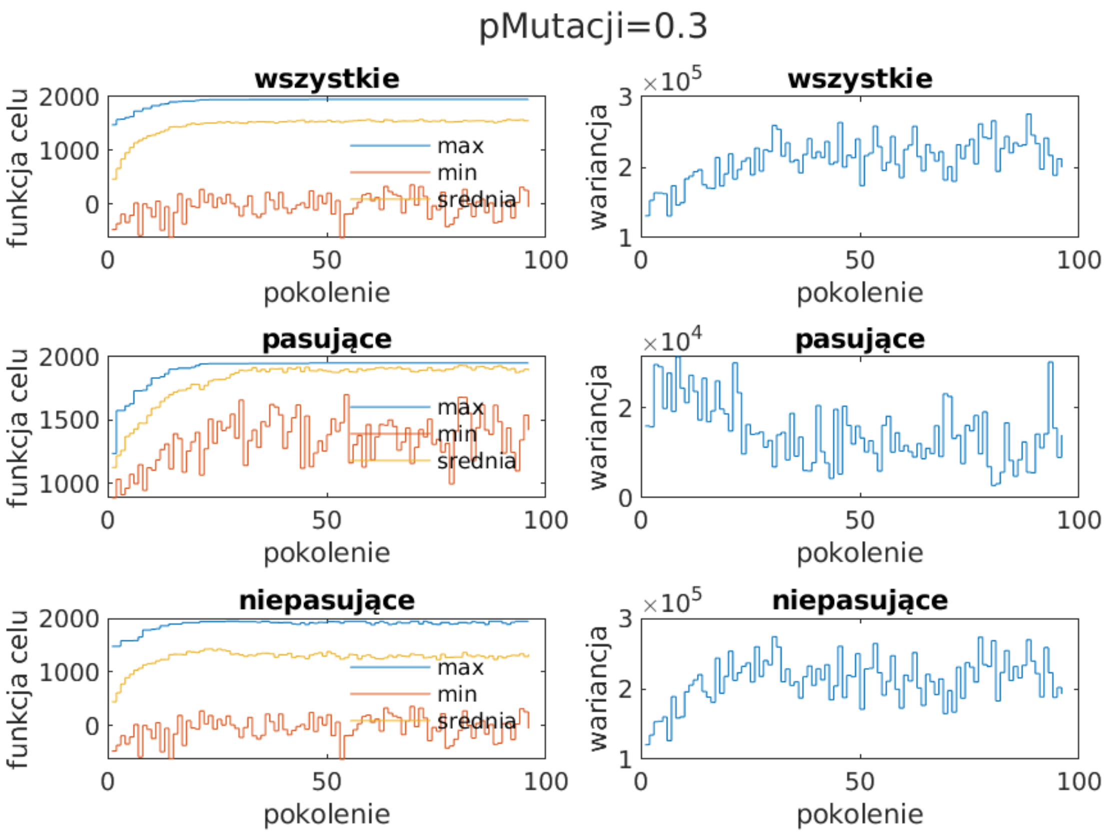
Rys. 3.5. Wyniki symulacji dla p mutacji = 0,05



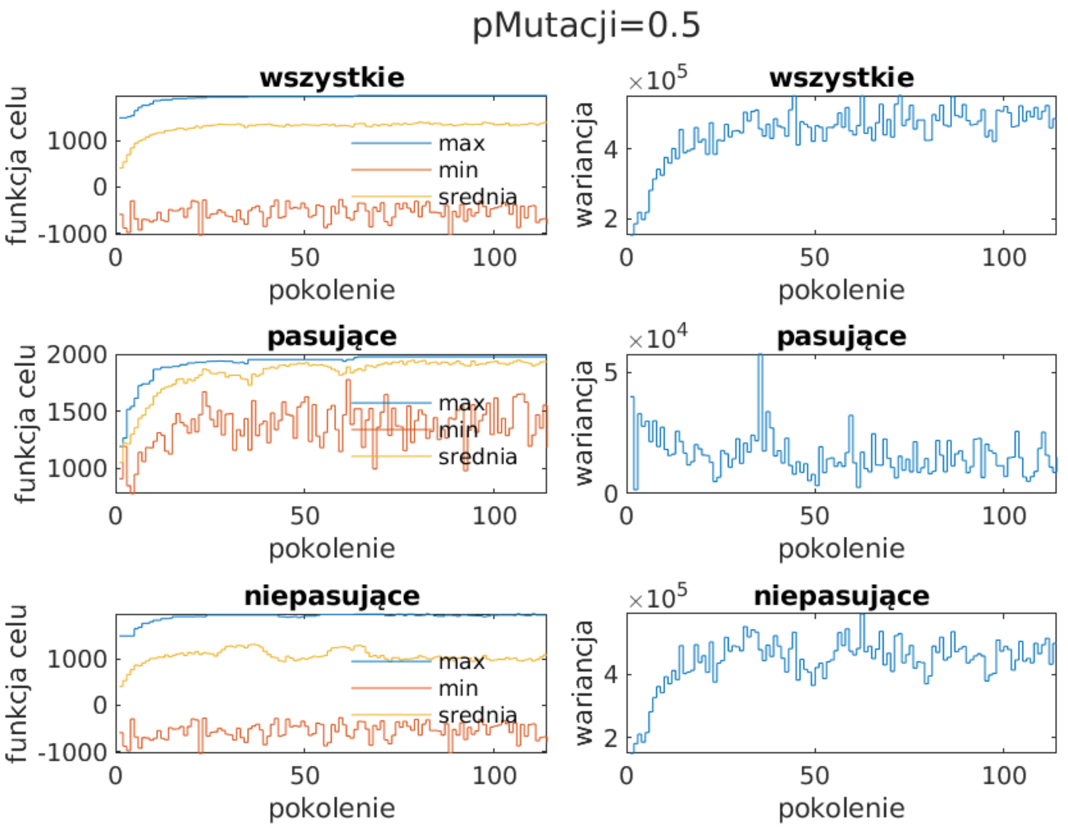
Rys. 3.6. Wyniki symulacji dla p mutacji = 0,1



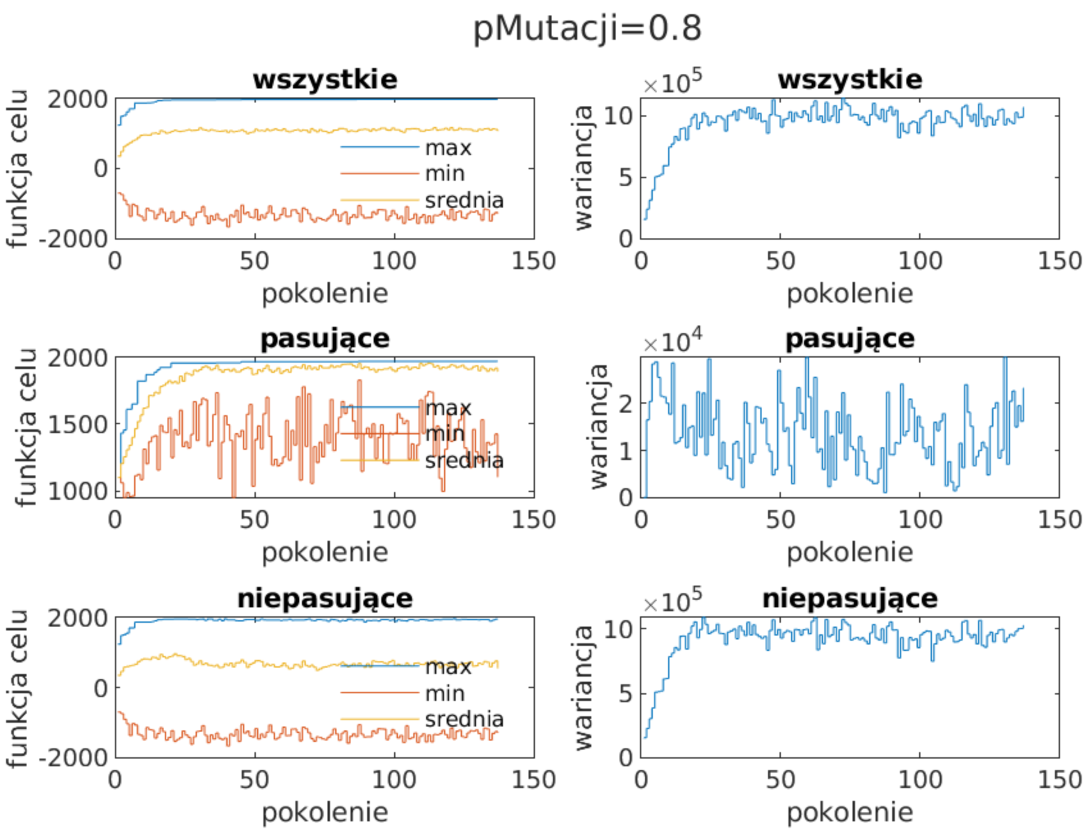
Rys. 3.7. Wyniki symulacji dla p mutacji = 0,2



Rys. 3.8. Wyniki symulacji dla p mutacji = 0,3



Rys. 3.9. Wyniki symulacji dla p mutacji $= 0{,}5$



Rys. 3.10. Wyniki symulacji dla p mutacji $= 0{,}8$

3.3. Dobór algorytmu krzyżowania

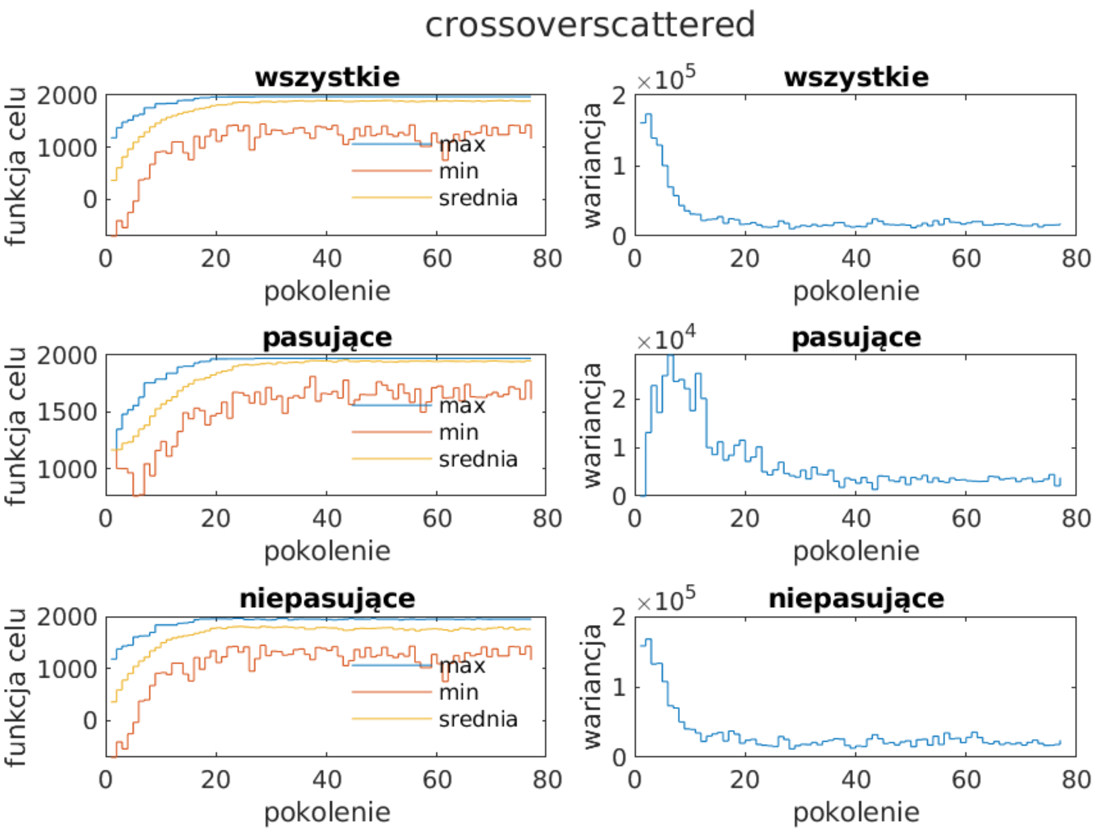
Wyniki optymalizacji:

metoda	y	p mutacji	p krzyżowania
crossoverscattered	1972	0,05	0,8
crossoversinglepoint	1971	0,05	0,8
crossovertwopoint	1966	0,05	0,8

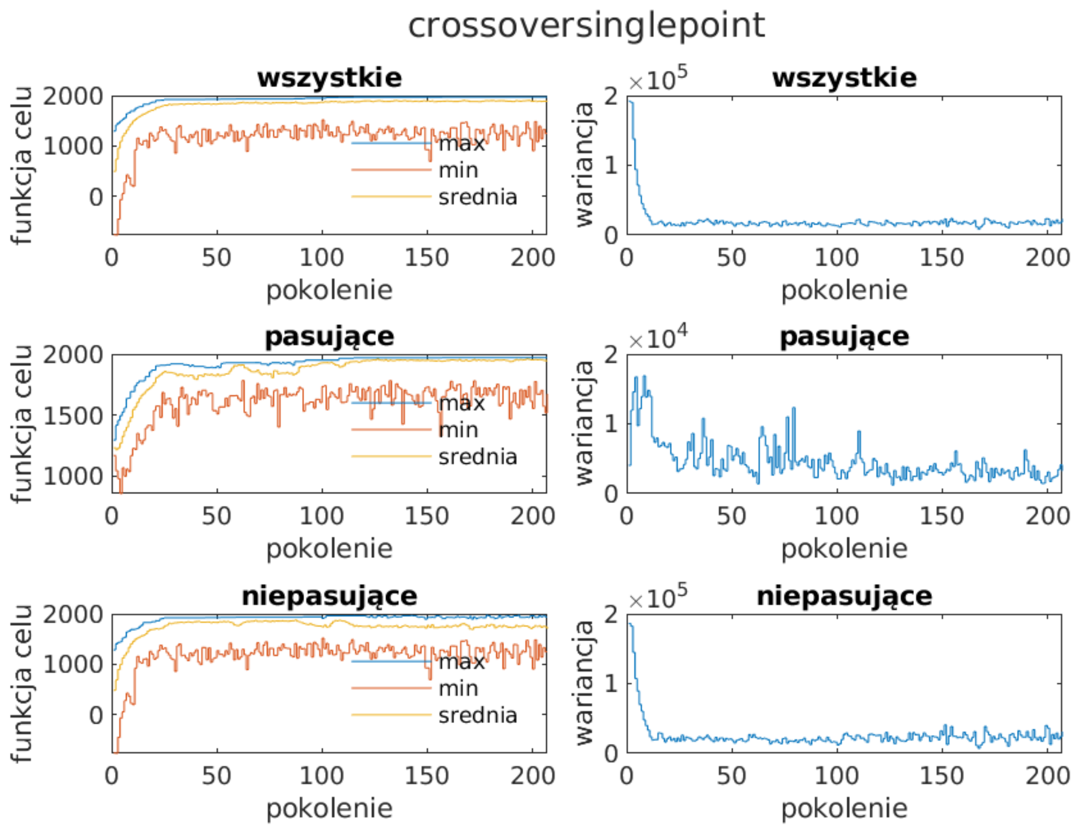
Wektory rozwiązań odpowiadające kolejnym algorytmom:

— X = [0 1 0 1 0 0 0 0 1 0 1 1 0 1 1 0 0 0 1 0 1 1 1 1 0 1 1 0 1 1 0 0 1 0 0 0 0 1 0 1 1 0 0 0 0 0 1 0 1 1 1 0 1 0 0 1 0 0 1 0 0 1 0 0]
— X = [0 1 0 1 0 0 1 0 1 1 1 1 0 1 1 0 0 0 1 0 0 0 1 1 0 1 1 0 1 1 0 0 1 0 0 0 0 1 0 1 1 0 0 0 0 0 1 0 0 1 1 1 1 0 0 1 0 0 1 0 0 1 0 0]
— X = [0 1 0 1 0 0 1 0 1 0 1 1 0 1 1 0 0 0 1 0 1 1 1 1 0 0 1 0 1 1 0 0 1 0 1 0 0 1 0 1 1 0 0 0 0 0 1 0 1 1 1 1 1 0 0 0 0 0 1 0 0 1 0 0]

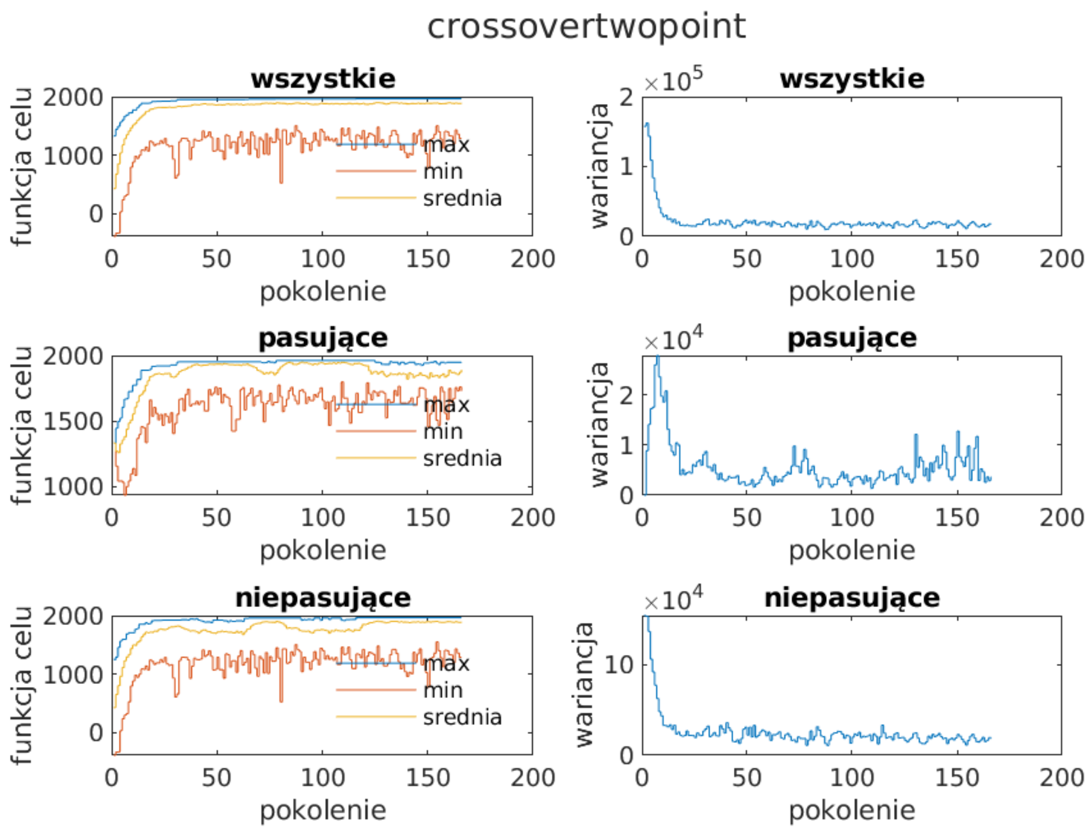
Zdecydowałem się na dalsze używanie crossoverscattered, pozostałe algorytmy działają gorzej zarówno pod względem zbieżności, jak i ilości potrzebnych obliczeń. Wykresy pokazujące przebieg symulacji znajdują się poniżej.



Rys. 3.11. Wyniki symulacji dla crossoverscattered



Rys. 3.12. Wyniki symulacji dla crossoversinglepoint



Rys. 3.13. Wyniki symulacji dla crossovertwopoint

3.4. Dobór metody selekcji

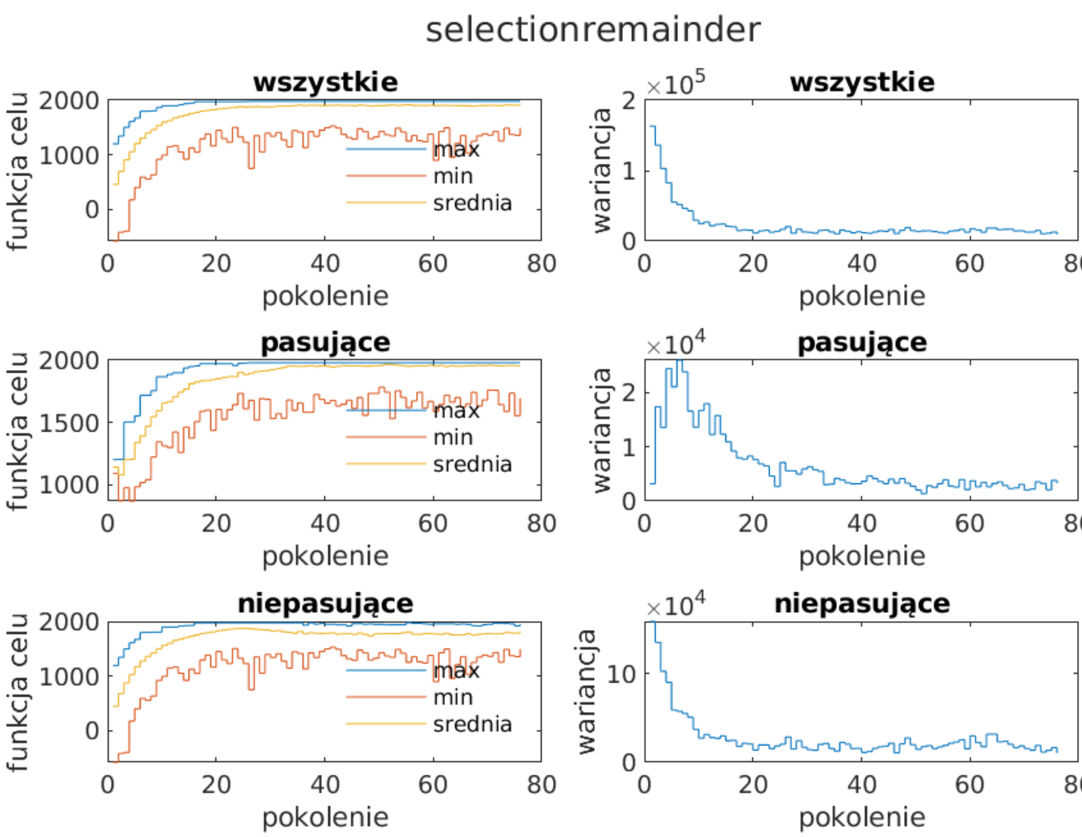
W przypadku ruletki oraz selectionuniform algorytmy nie dawały w generacji zróżnicowanych (część osobników zbyt ciężkich, pozostali o dopuszczalnej wadze). Zdecydowałem zatem, nie robić testów porównawczych dla tych algorytmów. Wyniki optymalizacji:

metoda	y	p mutacji	p krzyżowania
selectionremainder	1979	0,05	0,8
selectiontournament	1979	0,05	0,8
selectionstochunif	1980	0,05	0,8

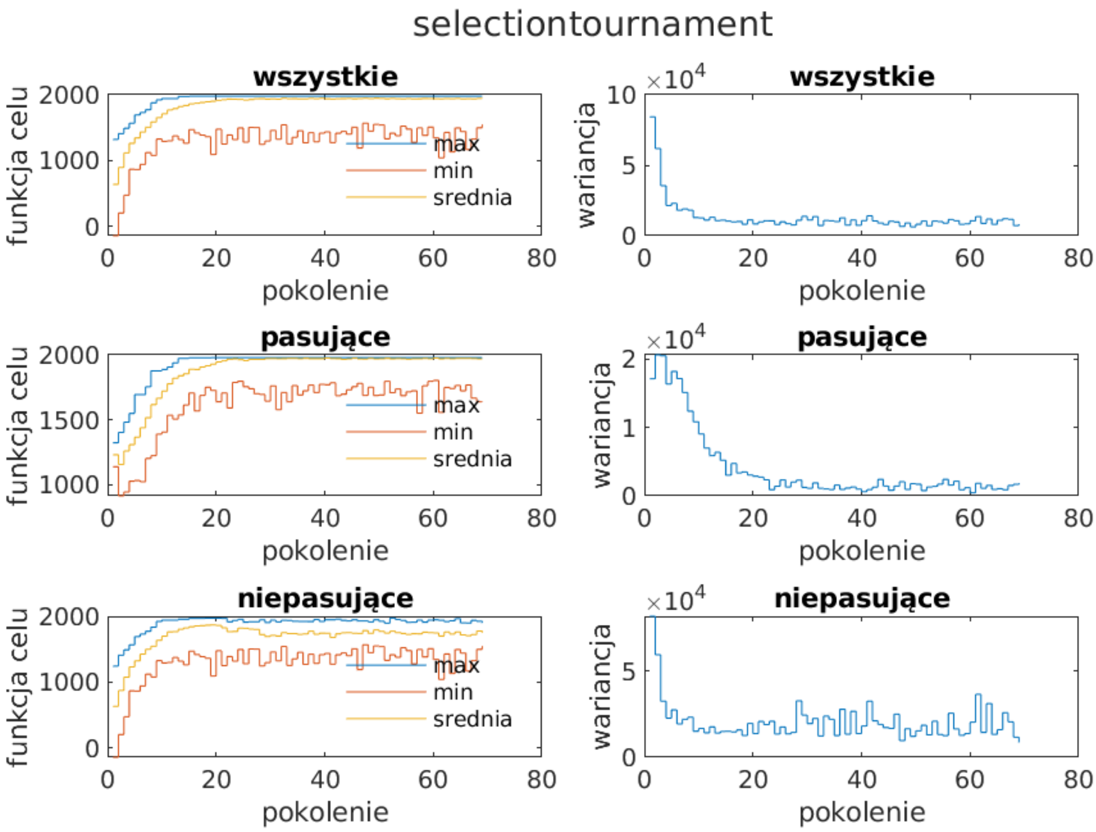
Wektory rozwiązań odpowiadające kolejnym metodom selekcji:

— X = [0 1 0 1 0 0 1 0 1 1 1 1 0 1 1 0 0 0 1 0 0 0 1 1 0 0 1 0 1 1 0 0 1 0 1 0 0 1 0 1 1 0 0 0 0 0 1 0 1 1 1 1 1 0 0 1 0 0 1 0 0 1 0 0]
— X = [0 1 0 1 0 0 1 0 1 1 1 1 0 1 1 0 0 0 1 0 0 0 1 1 0 0 1 0 1 1 0 0 1 0 1 0 0 1 0 1 1 0 0 0 0 0 1 0 1 1 1 1 1 0 0 1 0 0 1 0 0 1 0 0]
— X = [0 1 0 1 0 0 0 0 1 1 1 1 0 1 1 0 0 0 1 0 0 1 1 1 0 1 1 0 1 1 0 0 1 0 0 0 0 1 0 1 1 0 0 0 0 0 1 0 1 1 1 0 1 0 0 1 0 0 1 0 0 1 0 0]

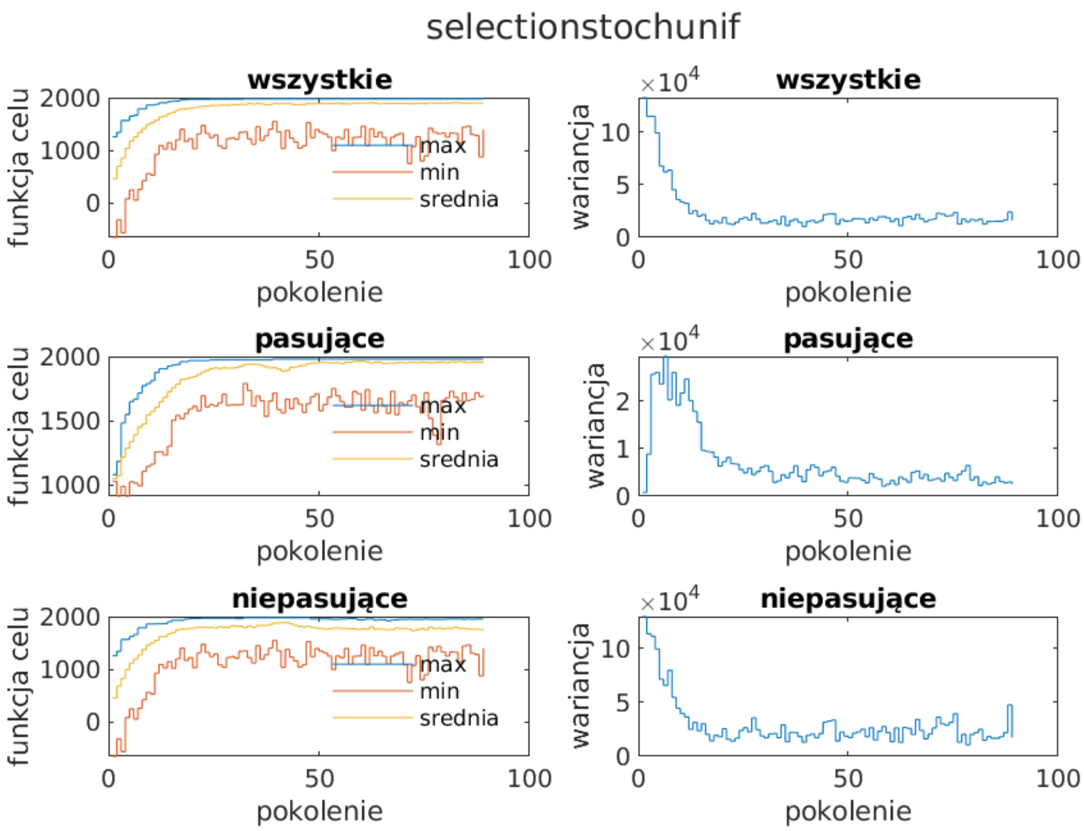
Zdecydowałem się na dalsze używanie selectiontournament, nie posiada on wad metody ruletkowej ani rankingowej (jest dużo lepszy w zachowywaniu różnorodności populacji), a poza tym jego działanie jest dla mnie najbardziej interesujące (na przykład ze względu na najbardziej monotoniczny rozkład wariancji elementów pasujących w danym pokoleniu). Zdecydowałem się na dalsze używanie tego algorytmu, mimo tego, że nie wygenerował optymalnego rozwiązania. Wykresy pokazujące przebieg symulacji znajdują się poniżej.



Rys. 3.14. Wyniki symulacji dla selectionremainder



Rys. 3.15. Wyniki symulacji dla selectiontournament



Rys. 3.16. Wyniki symulacji dla selectionstochunif

3.5. Dobór prawdopodobieństwa krzyżowania

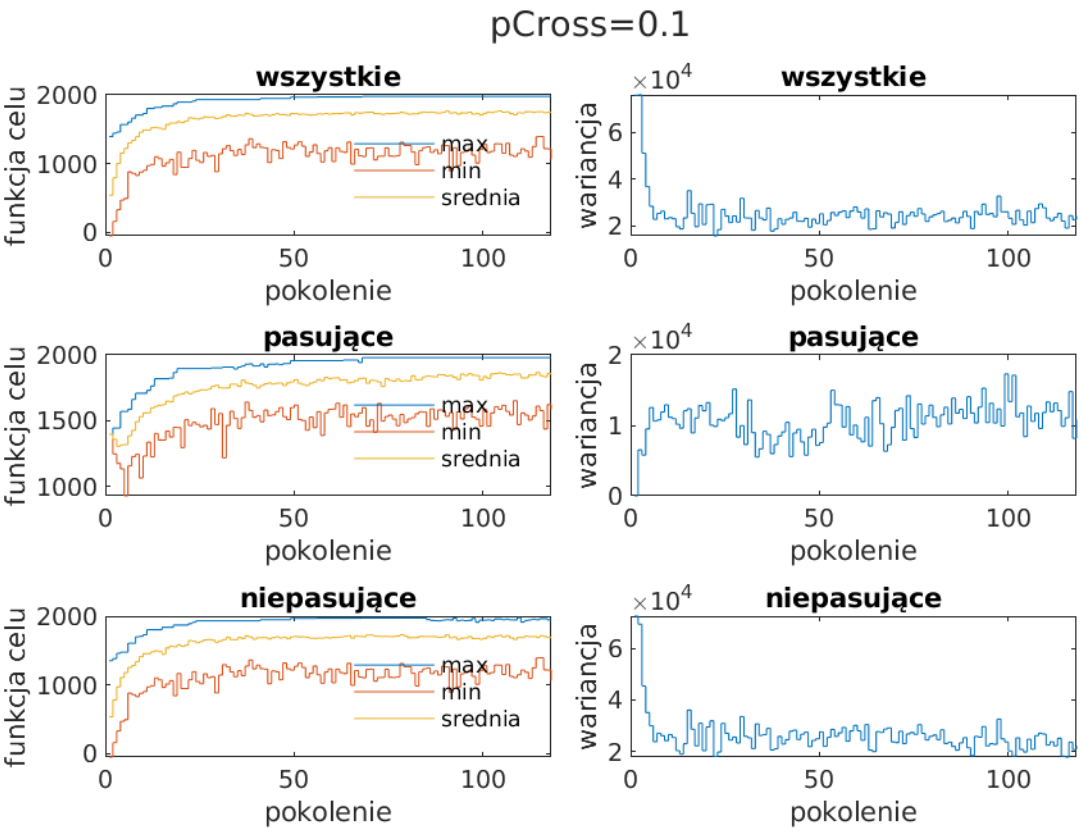
Wyniki optymalizacji:

p krzyżowania	y	p mutacji
0,1	1979	0,05
0,3	1979	0,05
0,5	1967	0,05
0,8	1980	0,05
0,9	1980	0,05
0,95	1980	0,05

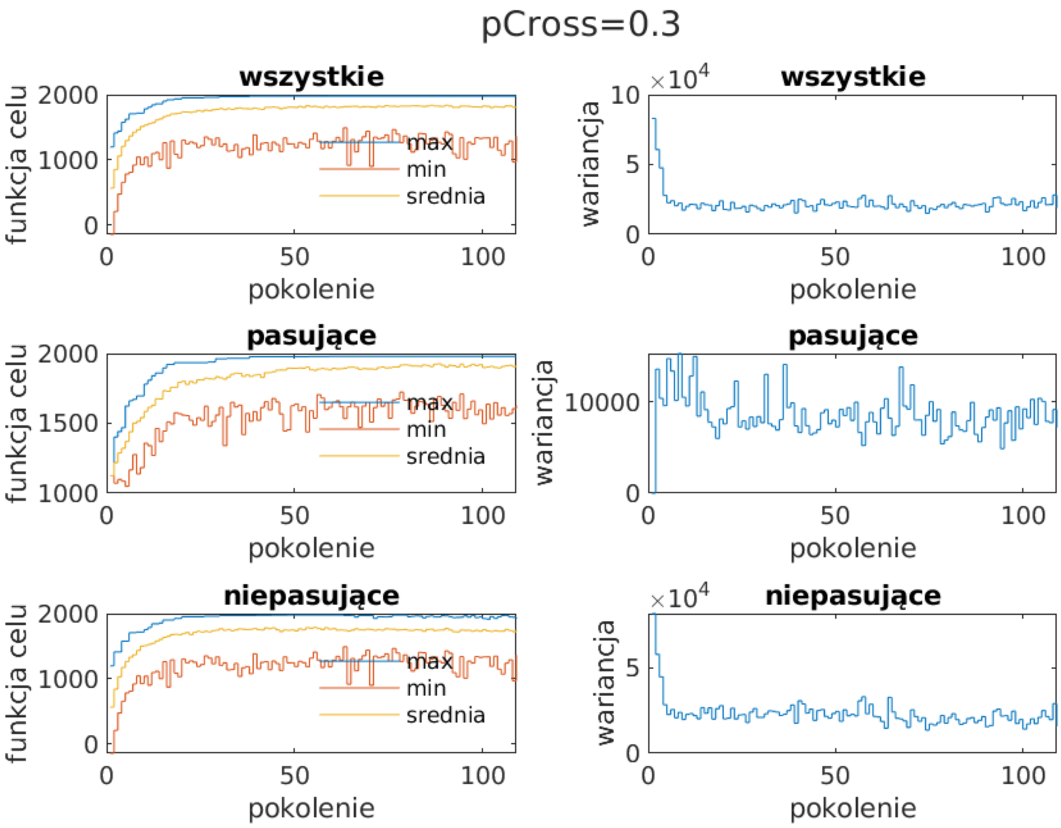
Wektory rozwiązań odpowiadające kolejnym wartościom prawdopodobieństwa krzyżowania:

— X = [0 1 0 1 0 0 1 0 1 1 1 1 0 1 1 0 0 0 1 0 0 0 1 1 0 0 1 0 1 1 0 0 1 0 1 0 0 1 0 1 1 0 0 0 0 0 1 0 1 1 1 1 1 0 0 1 0 0 1 0 0 1 0 0]
— X = [0 1 0 1 0 0 1 0 1 1 1 1 0 1 1 0 0 0 1 0 0 0 1 1 0 0 1 0 1 1 0 0 1 0 1 0 0 1 0 1 1 0 0 0 0 0 1 0 1 1 1 1 1 0 0 1 0 0 1 0 0 1 0 0]
— X = [0 1 0 1 0 0 0 0 1 1 1 1 0 1 1 0 0 0 1 0 0 1 1 1 0 0 1 1 1 1 0 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 1 1 1 1 1 1 1 0 0 1 0 0 1 0 0 1 0 0]
— X = [0 1 0 1 0 0 0 0 1 1 1 1 0 1 1 0 0 0 1 0 0 1 1 1 0 1 1 0 1 1 0 0 1 0 0 0 0 1 0 1 1 0 0 0 0 0 1 0 1 1 1 0 1 0 0 1 0 0 1 0 0 1 0 0]
— X = [0 1 0 1 0 0 0 0 1 1 1 1 0 1 1 0 0 0 1 0 0 1 1 1 0 1 1 0 1 1 0 0 1 0 0 0 0 1 0 1 1 0 0 0 0 0 1 0 1 1 1 0 1 0 0 1 0 0 1 0 0 1 0 0]
— X = [0 1 0 1 0 0 0 0 1 1 1 1 0 1 1 0 0 0 1 0 0 1 1 1 0 1 1 0 1 1 0 0 1 0 0 0 0 1 0 1 1 0 0 0 0 0 1 0 1 1 1 0 1 0 0 1 0 0 1 0 0 1 0 0]

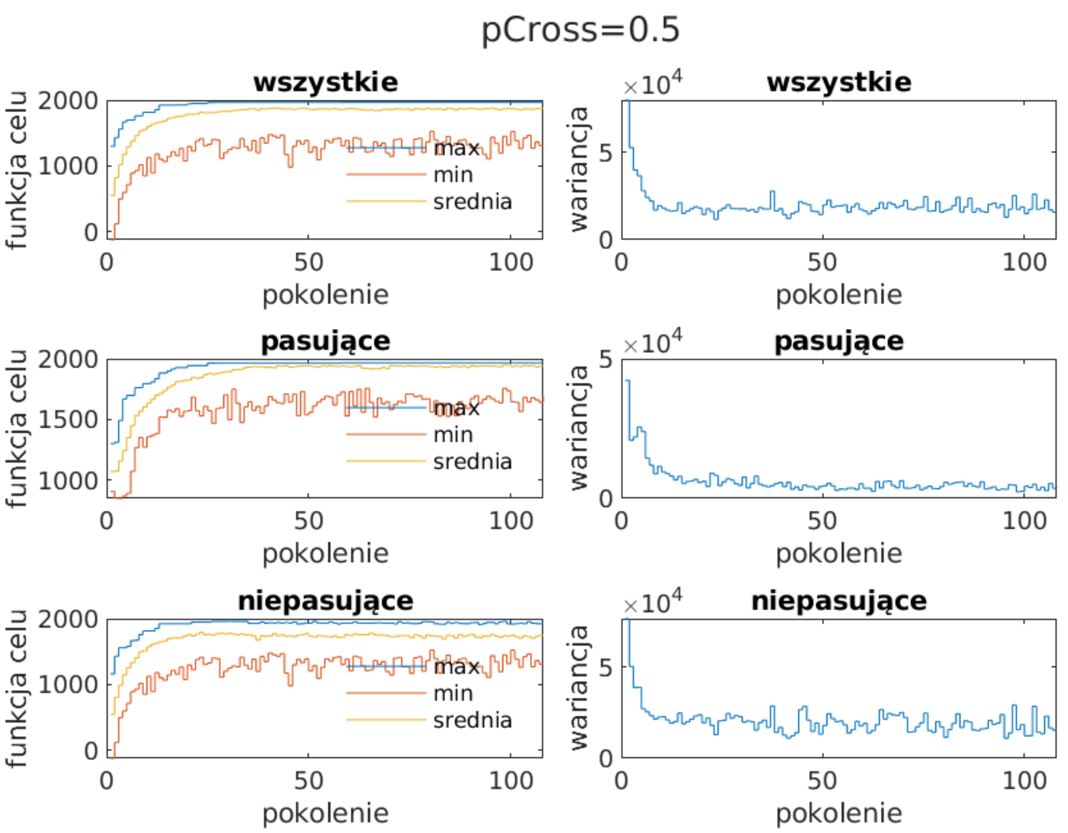
Zdecydowałem się na wybranie 0,9. Wysokie prawdopodobieństwo krzyżowania sprzyja szybszej zbieżności do optymalnego rozwiązania. Wykresy pokazujące przebieg symulacji znajdują się poniżej.



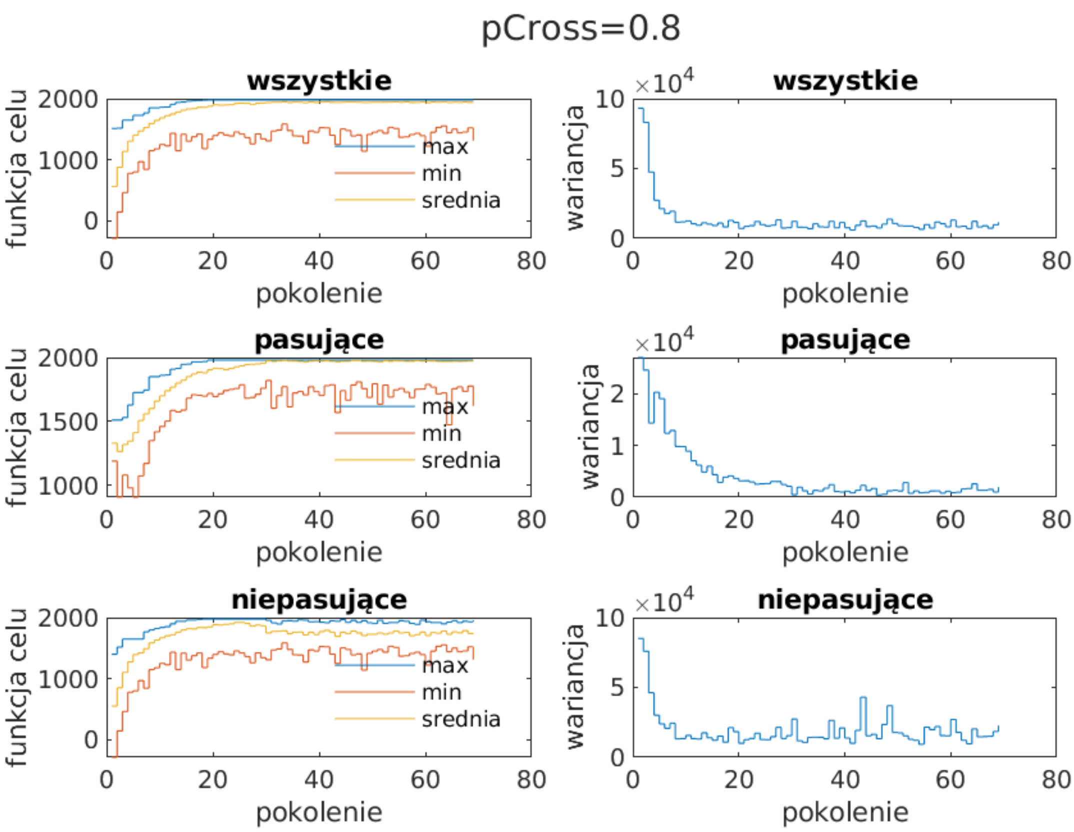
Rys. 3.17. Wyniki symulacji dla p krzyżowania = 0,1



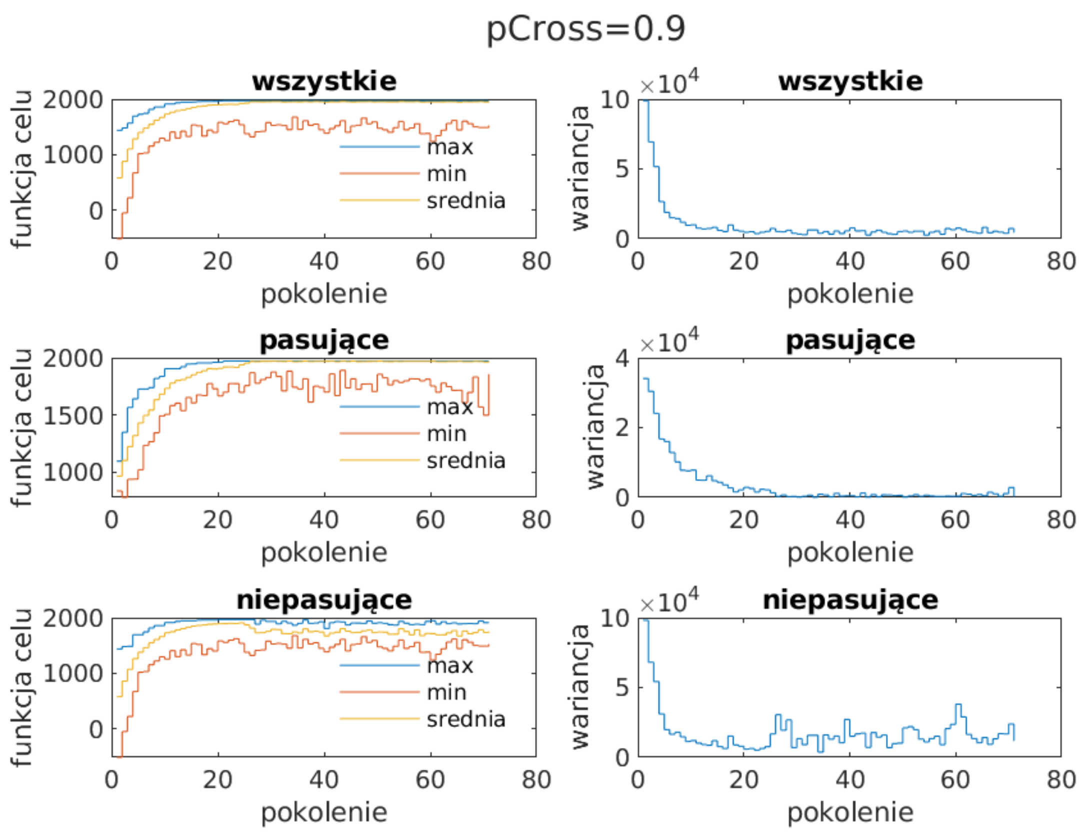
Rys. 3.18. Wyniki symulacji dla p krzyżowania = 0,3



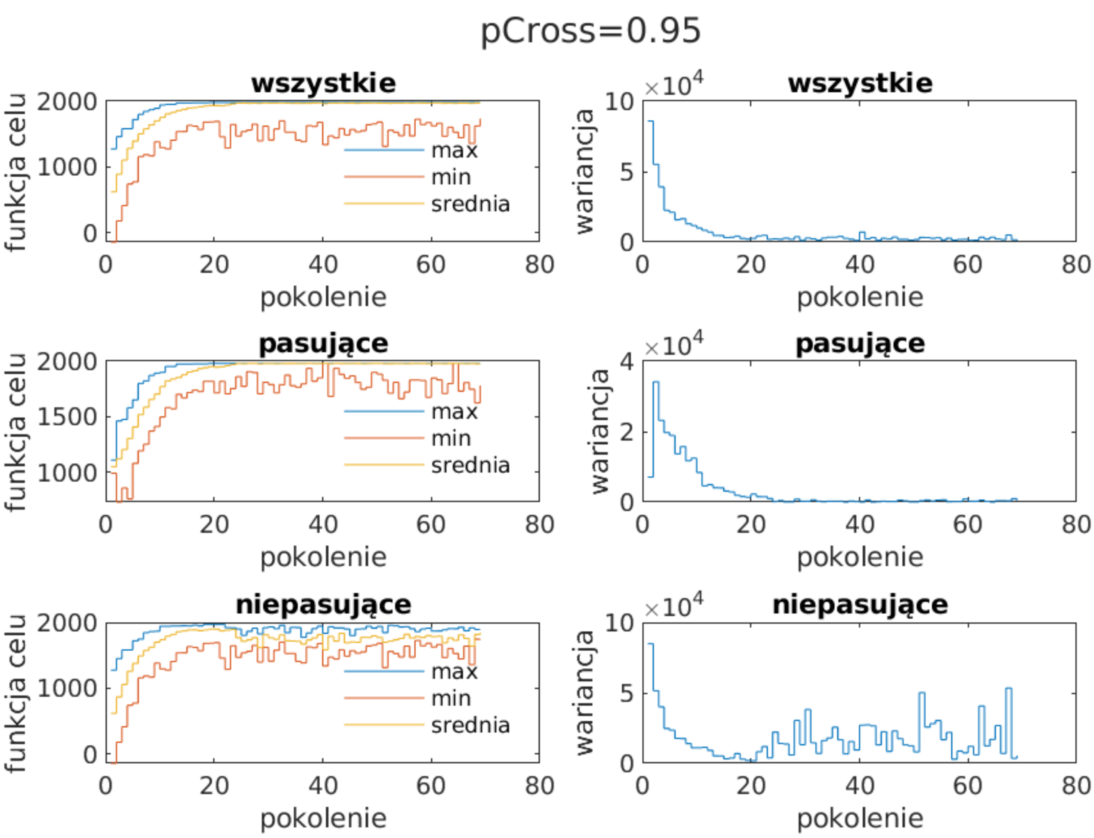
Rys. 3.19. Wyniki symulacji dla p krzyżowania $= 0{,}5$



Rys. 3.20. Wyniki symulacji dla p krzyżowania $= 0{,}8$



Rys. 3.21. Wyniki symulacji dla p krzyżowania = 0,9



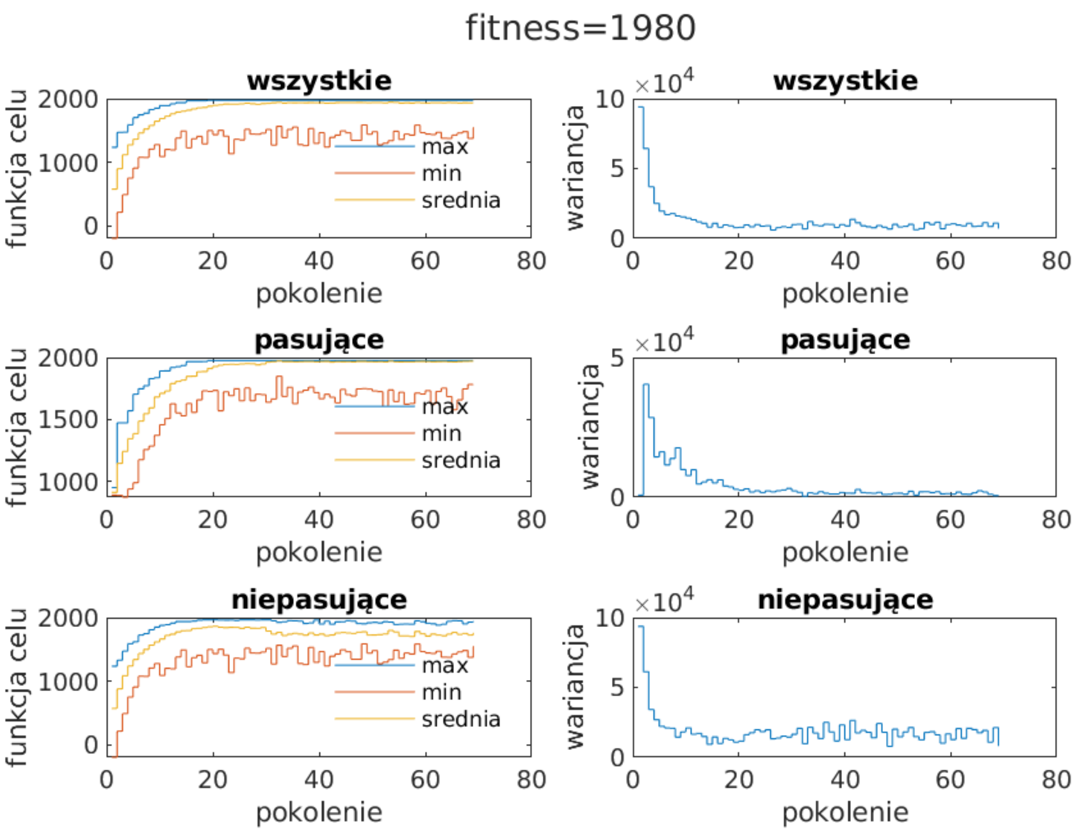
Rys. 3.22. Wyniki symulacji dla p krzyżowania = 0,95

3.6. Modyfikacja kryterium zatrzymania GA

Ponieważ znamy największą możliwą wartość funkcji, zdecydowałem się na wykorzystanie jej do wprowadzenia dodatkowego kryterium końca algorytmu genetycznego. Wyniki symulacji znajdują się poniżej.

y	p krzyżowania	p mutacji
1975	0,9	0,05

Wektor rozwiązań X = [0 1 0 1 0 0 0 0 1 1 1 1 0 1 1 0 0 0 1 0 0 1 1 1 0 1 1 1 1 1 0 0 1 0 0 0 0 1 0 1 1 0 0 0 0 0 1 0 1 1 1 0 1 0 0 0 0 0 1 0 0 1 0 0].
W tym przypadku, mimo użycia wiedzy o optimum globalnym, algorytm nie zdołał go osiągnąć. Nie jest w tym nic dziwnego - mamy do czynienia z dużą złożonością problemu, algorytm prawdopodobnie utknął w minimum lokalnym. Należałoby zrobić tą samą symulację ponownie, przy kilkukrotnym uruchomieniu GA prawdopodobieństwa szybkiej zbieżności do optimum globalnego znacznie wzrasta. Ten konkretny przykład został wybrany intencjonalnie - ma on pokazać, że mimo posiadanej wiedzy o optimum algorytm genetyczny nie zawsze jest w stanie go osiągnąć.



Rys. 3.23. Wyniki symulacji dla zmienionego kryterium stopu

3.7. Znalezione optimum globalne

W wyniku przeprowadzonego eksperymentu znaleziono następujące maksimum globalne:

X = [0 1 0 1 0 0 0 0 1 1 1 1 0 1 1 0 0 0 1 0 0 1 1 1 0 1 1 0 1 1 0 0 1 0 0 0 0 1 0 1 1 0 0 0 0 0 1 0 1 1 1 0 1 0 0 1 0 0 1 0 0 1 0 0]
wartość funkcji celu = 1980

4. Wnioski

4.1. Dobór parametru α

Widać, że współczynnik kary za zbyt dużą wagę zestawu pełni rolę filtra - zwiększając α coraz bardziej odrzucamy nie tylko o wiele za ciężkie zestawy, lecz także te, które są odrobinę za ciężkie. Widać to na przykład po wykresach wariancji w zależności od numeru pokolenia - różnorodność generacji znacząco maleje w przypadku dużych α. Należy zatem dobierać możliwie jak najmniejszy współczynnik α.

4.2. Dobór prawdopodobieństwa mutacji

Prawdopodobieństwo mutacji jest parametrem bardzo zależnym od rodzaju problemu, z jakim mamy do czynienia. Widać, że dla problemu plecakowego nie mamy sytuacji z bardzo dużą liczbą minimów lokalnych, przez co wystarczy to prawdopodobieństwo zachować na odpowiednio niskim poziomie - w ten sposób unikamy chaotycznych mutacji przy wyższych numerach pokoleń. Dobór zerowego prawdopodobieństwa mutacji też byłby błędem - ograniczamy w ten sposób dodatkowy czynnik wprowadzający różnice między pokoleniami. Gdyby to był problem trudniejszy do rozwiązania, należy modyfikować prawdopodobieństwo mutacji dynamicznie (w trakcie działania algorytmu) - na przykład poprzez chwilowe zwiększenie prawdopodobieństwa mutacji w sytuacji, gdy na danym horyzoncie pokoleń średnia zmienność funkcji celu jest zbyt mała.

4.3. Dobór algorytmu krzyżowania

Tutaj wybór jest prosty - należy ponad wszelką wątpliwość wybrać crossoverscattered - ten algorytm gwarantuje dużo lepszą jakościowo wariancję na przestrzeni pokoleń. Pozostałe dwa algorytmy były w stanie zaburzać mocno wariancję w późniejszych pokoleniach, gdzie raczej powinniśmy zaobserwować zjawisko tunelowania wokół poprawnego rozwiązania.

4.4. Dobór metody selekcji

Selekcja turniejowa działała bardzo poprawnie, ponieważ nie występują w niej główne bolączki selekcji rankingowej, ruletkowej, bądź innej - używając turnieju nie mamy problemu z chaotyczną wariancją lub samo napędzającymi się gwałtownymi zmianami w późniejszych populacjach. Oczywiście ważnym jest, by dobierać metodę pod konkretny problem - przy trudniejszych problemach to, co jest niekorzystne w algorytmach innych niż turniej, może się okazać zbawienne.

4.5. Dobór prawdopodobieństwa krzyżowania

Ponownie zależy nam na zachowaniu początkowo dużej różnorodności populacji, dlatego należy dobierać dość duże prawdopodobieństwo krzyżowania przy jednoczesnym unikaniu nadmiernego tworzenia kolejnych populacji z samych elit (ograniczamy sobie w ten sposób różnorodność kolejnych generacji).

4.6. Modyfikacja kryterium zatrzymania

W przypadku dodania maksymalnej możliwej wartości funkcji celu widać, że wcale nie gwarantuje to szybszego osiągnięcia optymalnych rozwiązań. Owszem, jest to pomocne - takie działanie przy wielu uruchomieniach algorytmu genetycznego jest w stanie bardzo ograniczyć potrzebną liczbę pokoleń, lecz nie zawsze. Wynika to z tego, że w wielu przypadkach algorytm jest w stanie ugrzęznąć w optimum lokalnym. Widać to było zwłaszcza przy ilości przedmiotów $n = 64$. W przypadku, gdy zależy nam na jak najwcześniejszym osiągnięciu wartości optymalnej jest to oczywiście przydatne, lecz potencjalnie jest w stanie zamknąć nas na znalezienie pozostałych optimów globalnych lub lokalnych.

4.7. Porównanie działania GA dla różnych liczb przedmiotów n

Widać, że w przypadku zwiększenia liczby przedmiotów z 32 do 64 problem staje się dużo trudniejszy do rozwiązania przez algorytm ewolucyjny. Nie jest w tym nic dziwnego - wraz ze wzrostem wymiaru problemu rośnie liczba zmiennych decyzyjnych oraz zwiększa się możliwość utknięcia algorytmu w różnych optimach lokalnych. Objawiło się to na przykład dużo rzadszym zwracaniem najbardziej optymalnego rozwiązania w przypadku 64 przedmiotów. Z drugiej strony, zwiększenie wymiarowości problemu nie zwiększyło zbytnio liczby generacji potrzebnej do znalezienia rozwiązania. Wspólne dla obu problemów ($n = 32$ oraz $n = 64$) jest zjawisko tunelowania się kolejnych generacji wokół potencjalnych optimów lokalnych bądź, co ważniejsze, globalnych. Co ciekawe, zbiory osobników o zbyt dużej wadze charakteryzowały się często dużo większą wariancją niż w przypadku zbiorów osobników o dopuszczalnej wadze. Wynika to z zaimplementowania istniejącego ograniczenia w funkcji celu.