1 Sposób implementacji algorytmu

Do rozwiązania problemu szeregowania zadań został zaimplementowany algorytm genetyczny. Poszczególne elementy algorytmu zostały zrealizowane w następujący sposób:

1.1 Wybieranie populacji początkowej

Populacja początkowa generowana jest przy pomocy algorytmu losowego. Generuje on wszystkie numery zadań, po czym przy miesza wygenerowaną listę. Później dla każdego zadania próbuje dodać po nim przerwę techniczną z prawdopodobieństwem 1%. Tak przygotowane rozwiązania są generowane aż do osiągnięcia limitu populacji.

1.2 Selekcja

Z każdego pokolenia wybierany jest pewien procent populacji, opisany przez parametr *selectionSize*, który następnie uczestniczy w procesie krzyżowania. Aby to osiągnąć przeprowadzany jest turniej z udziałem dwóch osobników i ten z gorszym wynikiem jest usuwany. Proces powtarza się aż do osiągnięcia pożądanej ilości rozwiązań.

1.3 Krzyżowanie

W aglorytmie zostało zastosowane krzyżowanie jednopunktowe. Punkt podziału wybierany jest losowo. Powstają dwa nowe rozwiązania zbudowane w taki sposób, że pierwsza część jest identyczna jak jednego z rodziców, do punktu podziału, a druga jest generowana na podstawie kolejności rozwiązań u drugiego z rodziców.

1.4 Mutacja

Dla każdego rozwiązania mutacja może przyjąć z następujących form:

1. Zamiana miejscami numerów dwóch zadań
2. Dodanie przerwy technicznej
3. Usunięcie przerwy technicznej
4. Zmiana długości przerwy technicznej
5. Zmiana zadania po którym wystąpi istniejąca już przerwa techniczna

1.5 Kolejne pokolenia

Liczba rozwiązań w ciągu następnych pokoleń utrzymywana jest na stałym z góry ustalonym poziomie. Ze wszystkich rozwiązań z obecnego pokolenia i tych powstałych po krzyżowaniu wybierane są najlepsze.

1.6 Zakończenie działania

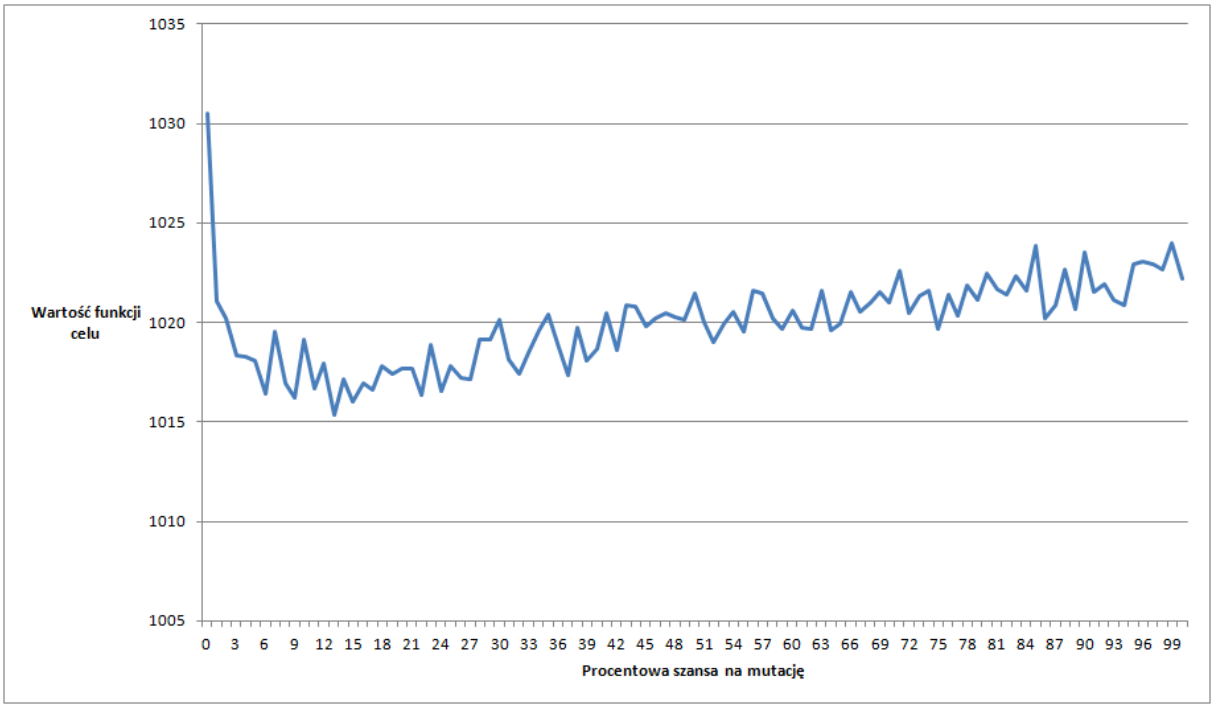
Algorytm kończy swoje działanie po upływie ustalonego czasu, ustalonego na 30s, lub gdy najlepsze rozwiązanie nie zostanie poprawione przez 1000 pokoleń.

2 Strojenie parametrów algorytmu

Początkowo wszystkie parametry algorytmu zostały ustawione w sposób przypadkowy. Następnie każdy parametr był oddzielnie zmieniany i testowany na instancjach zamieszczonych w plikach od 1 do 9. Dla każdej instancji proces został powtórzony pięciokrotnie, a następnie została policzona średnia wartość funkcji celu.

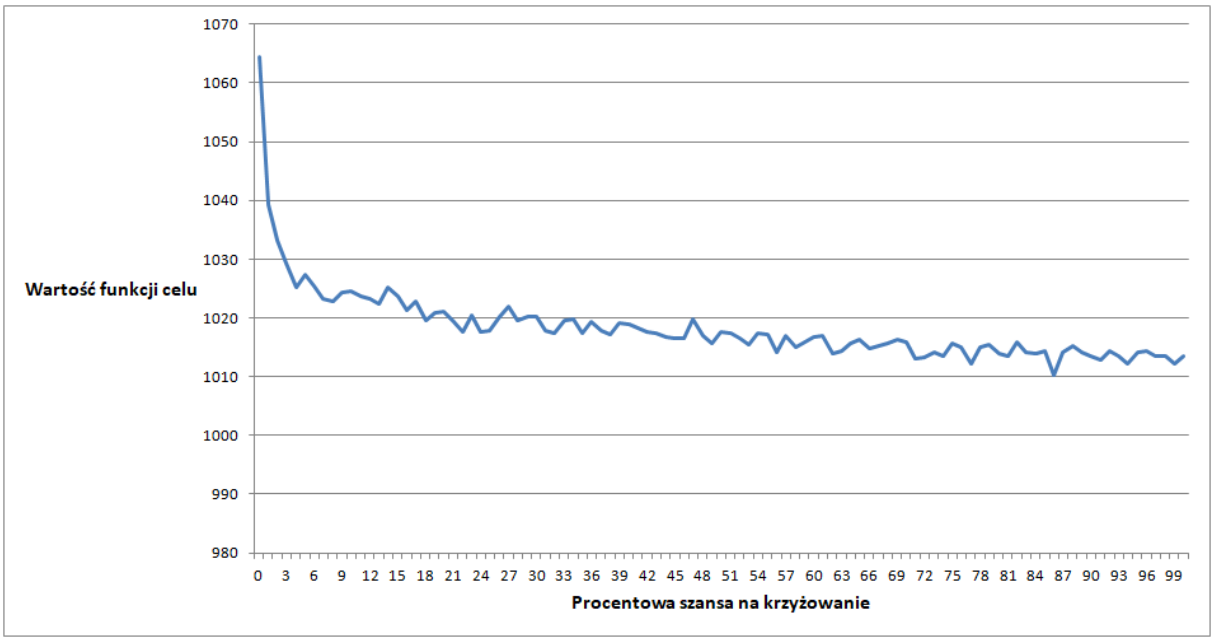
2.1 Mutacje

Wartość funkcji celu, gdy w algorytmie nie uwzględniono w ogóle mutacji wartości funkcji są najgorsze. Wraz ze wzrostem ilości mutacji wartość funkcji celu poprawia się i osiąga najlepszą wartość dla szansy na mutację wynoszącej 13%. Dalej im większa szansa na mutację tym osiągany wynik jest gorszy co spowodowane jest tym, że kolejne wygenerowane rozwiązania są psute przez losową zmianę.



Rysunek 1: Średnia wartość funkcji celu dla różnych wartości sznasy na mutację

2.2 Krzyżowanie

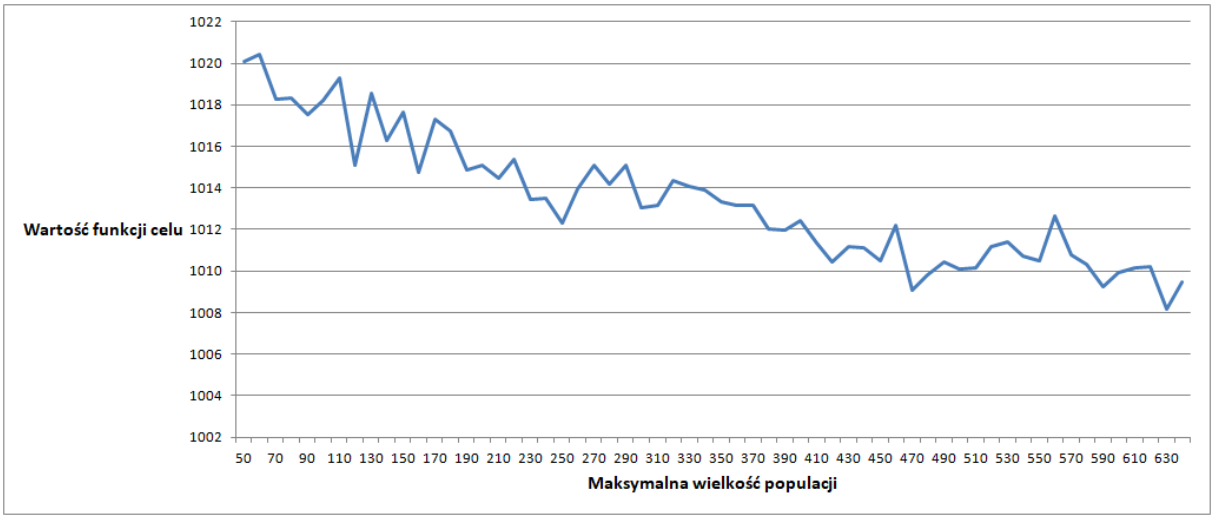


Rysunek 2: Średnia wartość funkcji celu dla różnych wartości szansy na krzyżowanie

Dla zerowej szansy na krzyżowanie algorytm jest w zasadzie algorytmem losowym. Nawet niewielka szansa na krzyżowanie powoduje znaczną poprawę rozwiązań algorytmu. Wraz ze wzrostem szansy na krzyżowanie rośnie jakość rozwiązań. Ostatecznie algorytm wypada najlepiej przy współczynniku wynoszącym 86%.

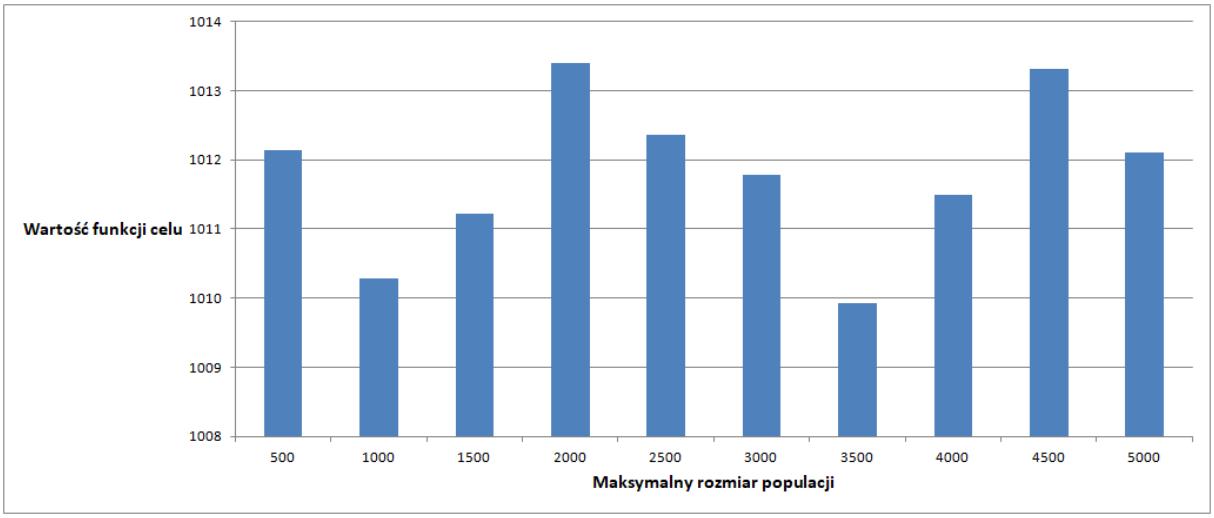
2.3 Rozmiar populacji

Na wykresie została przedstawiona jak zmieniała się średnia wartość funkcji celu w zależności od tego jak duża mogła być populacja. Wielkość ta była badana na przedziale od 50 do 640 ze skokiem co 10. Im większy mak-



Rysunek 3: Średnia wartość funkcji celu dla różnych wartości maksymalnego rozmiaru populacji

symalny rozmiar populacji tym generalnie osiągane wyniki są coraz lepsze. W celu sprawdzenia czy znaczne zwiększenie maksymalnego rozmiaru populacji nie poprawi wyniku zostały przeprowadzone dodatkowe testy do maksymalnej wartości równej 5000 ze skokiem co 500.



Znaczne zwiększenie ilości rozwiązań w jednej populacji nie wpłynęło pozytywnie na jakość wyniku, a wręcz go pogorszyło. Ostatecznie parametrem jakim przyjęto dla wielkości populacji było 630.

3 Testy

Ostatecznie testy były przeprowadzane dla następujących parametrów:

1. Prawdopodobieństwa mutacji - 13%
2. Prawdopodobieństwa krzyżowania - 86%
3. Maksymalnej wielkości populacji - 630

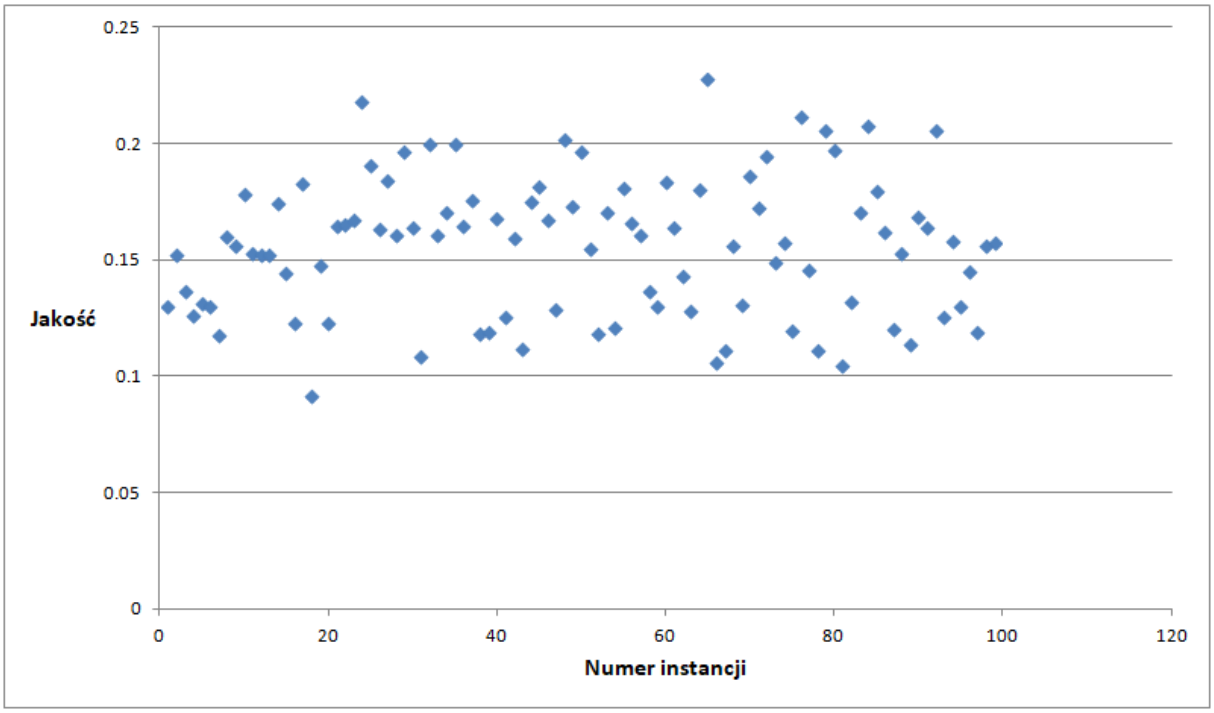
Testy zostały podzielone na 4 grupy:

1. Ilość zadań 50, czas operacji od 5 do 20 - pliki 1-99
2. Ilość zadań 50, czas operacji od 5 do 100 - pliki 101-199
3. Ilość zadań 50, czas operacji od 5 do 10 - pliki 201-299
4. Ilość zadań 500, czas operacji od 5 do 50 - pliki 301-399

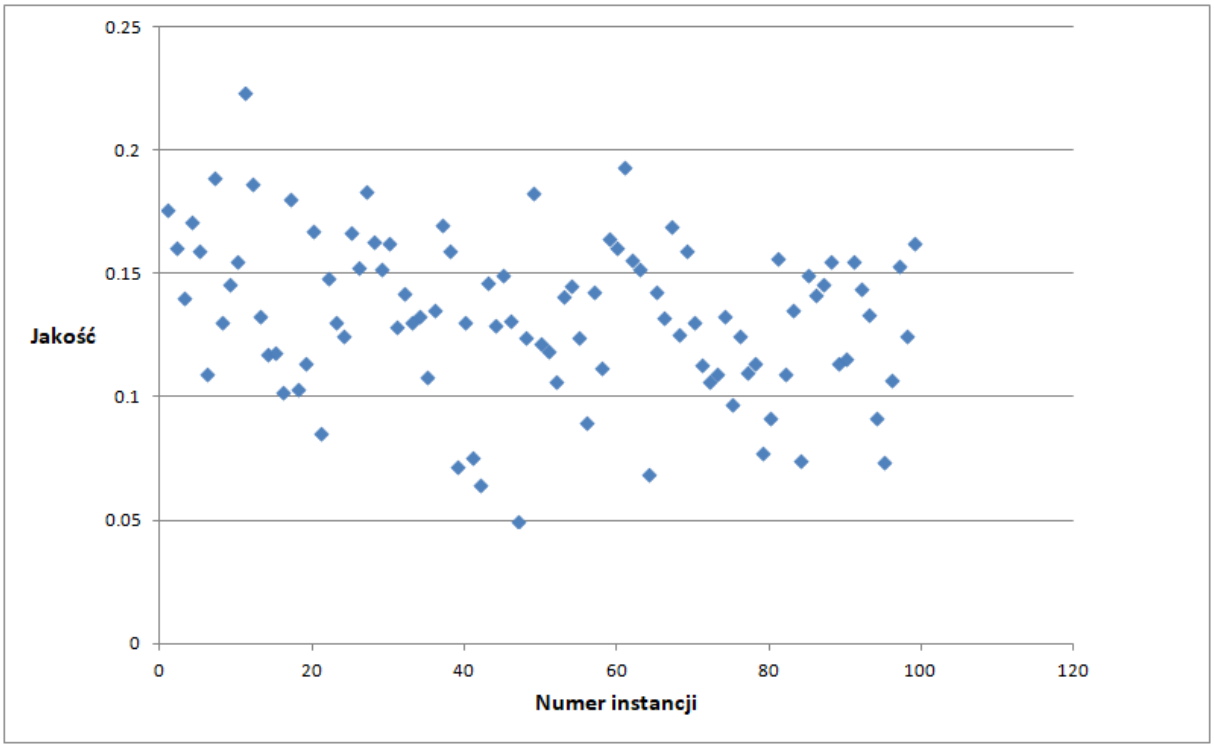
Dla każdej instancji została wyznaczona jakość rozwiązania wyrażonej przez wzór:

$$\frac{N_l}{N_{ga}} - 1$$

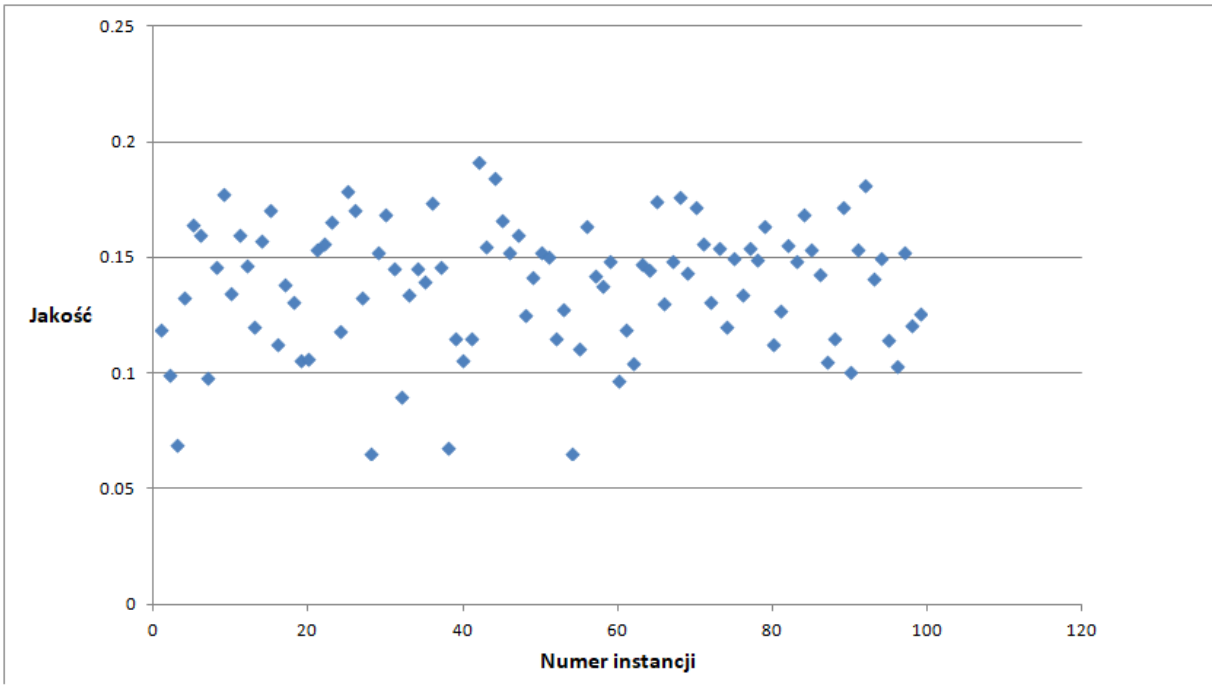
gdzie N_l jest najlepszym rozwiązaniem losowym z pierwszego pokolenia, a N_{ga} najlepszym rozwiązaniem wyznaczonym przez algorytm genetyczny.



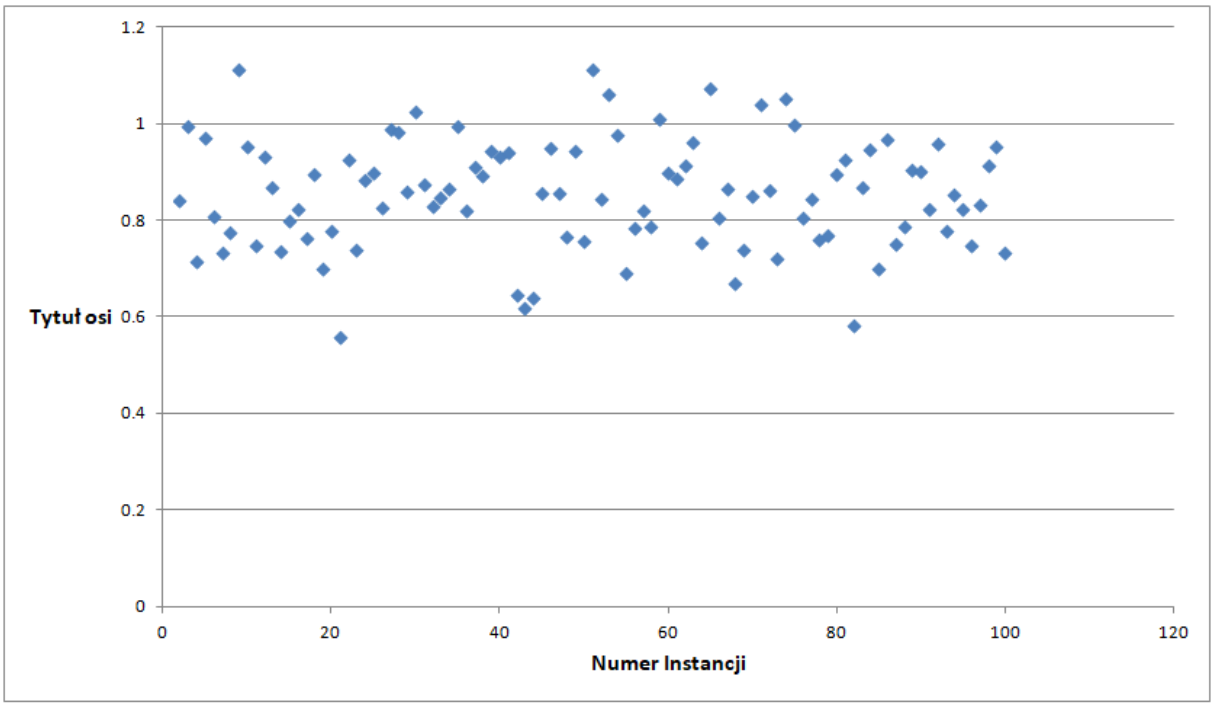
Na powyższym wykresie została pokazana jakość rozwiązań dla pierwszego zestawu testów. Dla wszystkich instancji algorytm był w stanie znaleźć rozwiązanie lepsze od zaproponowanego przez algorytm losowy. Średnio znalezione rozwiązanie było o 15,6% lepsze.



Przedstawione powyżej wyniki dotyczą testów z drugiej grupy, w których został zwiększony kilkukrotnie maksymalny czas dla danej operacji. Rozwiązania nadal są lepsze od algorytmu losowego, jednak ich jakość nieco spadła, bowiem średnia jakość rozwiązania wynosi 13,4%. Może to wynikać z faktu, że parametry algorytmu genetycznego były dostrojone do instancji z grupy pierwszej.



Dla grupy testów numer 3, czyli tej, gdzie czasy operacji są bardzo zbliżone do siebie, wyniki nie odbiegają zbytnio od poprzednich. Średnia poprawa jakości wynosi 13,9%. Mniejsza jakość rozwiązań wynika prawdopodobnie z tej samej przyczyny, co dla grupy numer 2.



Dla czwartej grupy testów można zaobserwować znaczącą poprawę jakości rozwiązania. Dziesięciokrotne zwiększenie ilości zadań uniemożliwiło algorytmowi losowemu znalezienie dobrej jakości rozwiązania, co z drugiej strony pozwoliło na wykazanie się algorytmu genetycznego. Średnia jakość rozwiązania dla tych testów wynosiła 85,5%.

4 Wnioski

Dla wszystkich testów algorytm genetyczny potrafił znaleźć zdecydowanie lepsze rozwiązanie niż algorytm losowy. Algorytm generował rozwiązanie o podobnej jakości niezależnie od długości zadań, czy różnic pomiędzy czasami poszczególnych operacji. Jedynym parametrem, który realnie wpłynął na wyniki była liczba zadań. Im ich więcej tym algorytm powinien radzić sobie lepiej.