Igor Urbanowicz

Kacper Majkowski

Sprawozdanie Etap 3

Implementacja Algorytmu Genetycznego dla problemu komiwojażera.

Wstęp teoretyczny

Celem wykonania trzeciego etapu była implementacja oraz analiza efektywności Algorytmu Genetycznego dla asymetrycznego problemu komiwojażera. Jest to algorytm heurystyczny a więc umożliwia on uzyskanie rozwiązania problemu bez gwarancji jego dokładności i optymalności. Jego zasada działania została zainspirowana zachodzącymi w biologii procesami ewolucji. Kluczowym elementem tego algorytmu jest osobnik, któremu dla problemu komiwojażera odpowiada ścieżka w przeszukiwanej przestrzeni rozwiązań, zwana również chromosomem lub genotypem. Zbiór osobników nazywany jest populacją, będącą podstawą dla procesu ewolucji odwzorowywanego przez Algorytm Genetyczny. Składa się on następujących elementów:

1. **Generowanie populacji początkowej –** najpierw wytwarzany jest zbiór osobników, będących losowymi rozwiązaniami znajdującymi się w przeszukiwanej przestrzeni.

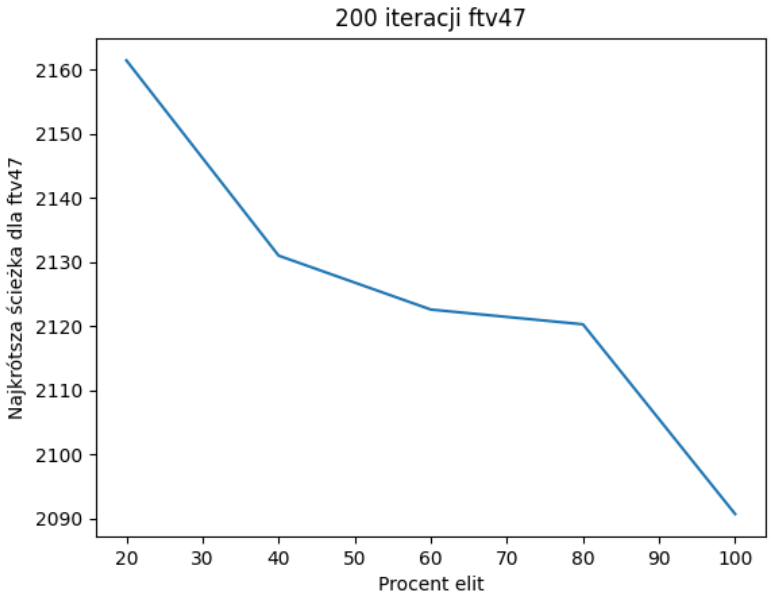
2. **Selekcja** – jest to najważniejszy element całego algorytmu. Polega ona na wybieraniu z populacji osobników najlepiej przystosowanych, które znajdą się w następnej populacji (pokoleniu).

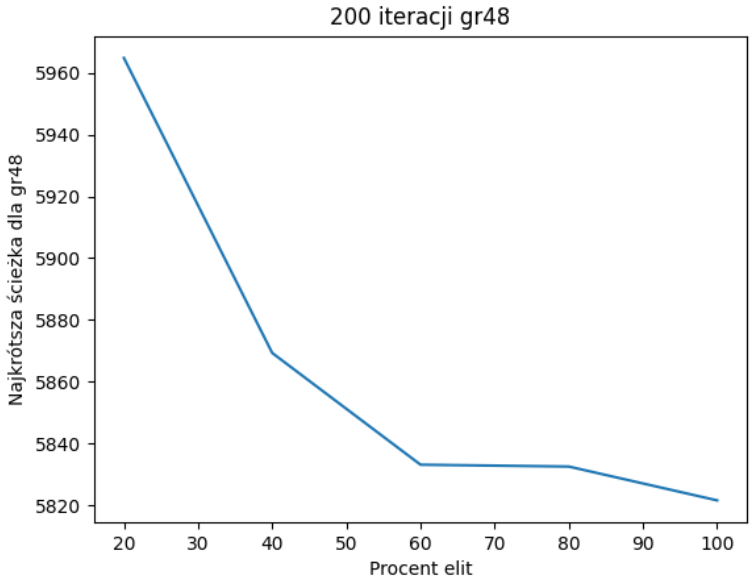
3. **Krzyżowanie** – proces ten polega na łączeniu ze sobą par osobników celem wytworzenia potomków zastępujących ich w populacji.

4. **Mutowanie** – z bieżącej populacji wybierana jest pewna ilość osobników poddawanych losowym modyfikacjom. Zadaniem tego procesu jest wspomaganie eksploracji przestrzeni rozwiązań.

Implementacja

1. **Generowanie populacji początkowej** – nasz jako populację początkową bierze określoną liczbę przypadkowych permutacji miast oraz sparametryzowany procent ‘elit’, które są reprezentowane przez ścieżki stworzone przez algorytm zachłanny.



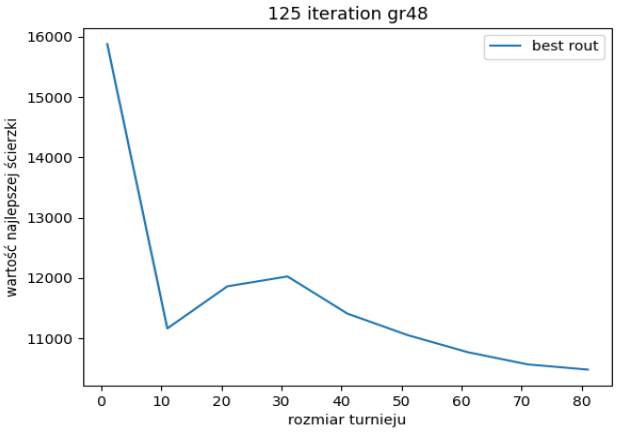


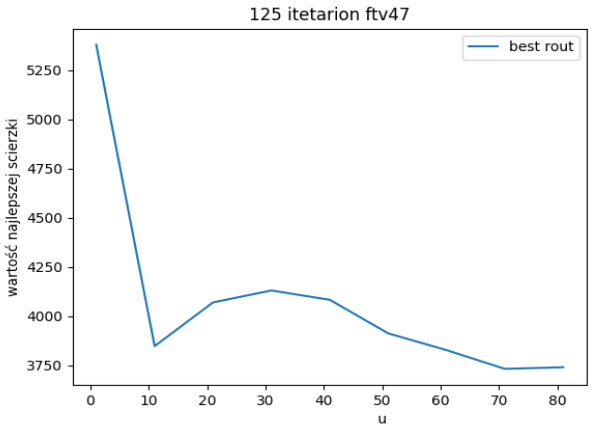
Jak widzimy algorytm początkowa ilość elit zwiększa efektywność naszego algorytmu, przynajmniej dla małych iteracji. Jednak doprowadza do małej początkowej puli genowej co przeszkadza w prowadzeniu rzetelnych badań, bo praktycznie nie możemy wyjść za tej „doliny”

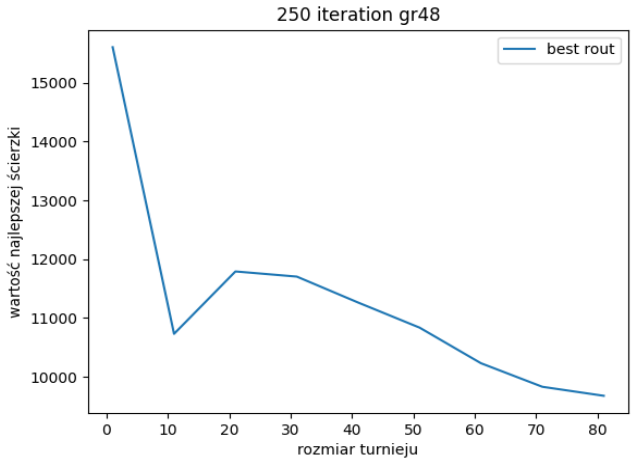
stworzonej przez algorytm zachłanny. Przez to w reszcie sprawozdania będziemy się posługiwać współczynnikiem elit równym zero.

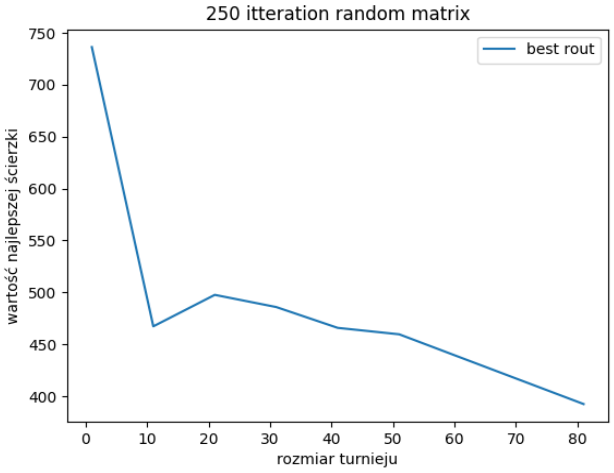
Selekcja

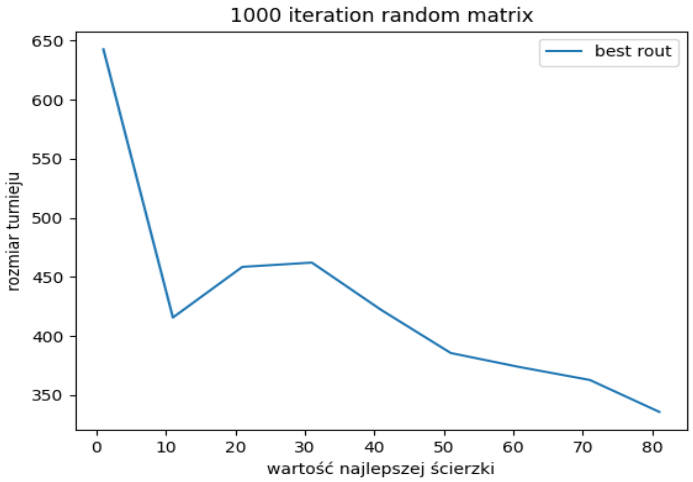
Posiadamy dwa sposoby selekcji populacji. Pierwszą z nich jest turniej. Z populacji losujemy określoną ilość osobników do turnieju i wygrywa ten który ma najkrótszą długość cyklu. Następnie wygrany przechodzi do następnego etapu i ma szansę skrzyżować się z innym osobnikiem. Powtarzamy to ilość razy zależną od wielkości populacji. Zwycięscy będą mieli szansę się krzyżować.





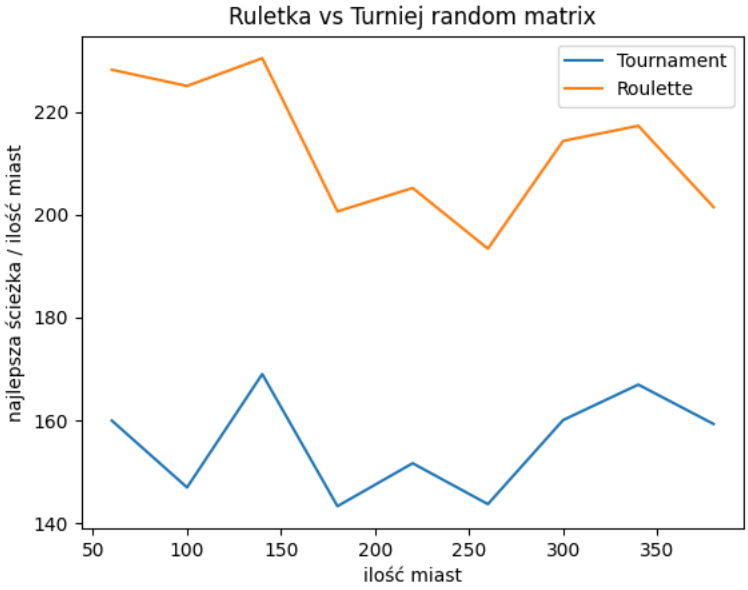






Wszystkie badania zostały wykonane dla populacji, która oscyluje między 100, a 200. Jak widzimy nie zależnie od ilości iteracji oraz rodzaju problemu większy rozmiar turnieju działa najlepiej.

Naszą drugą metodą jest ruletka. Podobnie jak w turnieju losujemy pewną część populacji. Następnie sumujemy wszystkie drogi po czym każdej z nich przypisujemy prawdopodobieństwo: (suma)/(długość ścieżki) + w prawdopodobieństwo poprzedniej ścieżki. Dzięki temu zyskujemy rosnący ciąg, który dla dobrych ścieżek ma dużą różnice między elementami. Po tym losujemy liczbę miedzy 0 a sumą i szukamy największy element z liczbą mniejszą od wylosowanej (przez to że ciąg jest rosnący możemy do przeszukiwania wykorzystać binary search, co trochę przyspieszy program).



Turniej został wykonany dla wielkości 80 i 1000 iteracji. Ruletka daje około 30% gorsze wyniki od turnieju. Może to być spowodowane małą selekcją ruletki. W ruletce ścieżka, która jest 2 razy

lepsza ma tylko 2 razy większą szanse na to ,że zostanie wybrana, gdzie w turnieju szczególnie przy tak dużej wielkości często będą wybierane najlepsze jednostki do krzyżowania.
Co do złożoności obliczeniowej selecta. Jeżeli do turnieju losujemy W osobników, a N to nasza populacja do selekcji. Dla jednego turnieju musimy znaleźć najkrótszą z W ścieżek oraz je wylosować i musimy to zrobić N razy, więc nasza złożoność obliczeniowa to O(W*N), przy założeniu, że długości ścieżek już są policzone. Co do ruletki na początku musimy przejść się po wszystkich osobnikach populacji sumując ich trasy po czym zrobić to jeszcze raz, aby nadać im wartość prawdopodobieństwa. Możem to zrobić liniowo, ale następnie trzeba wylosować N osobników do selekcji i każdego znaleźć w tablicy bin search'em więc finałowy czas to O(N*logN). To który jest szybszy zależy od tego czy W jest stałą czy nie.

Krzyżowanie

Osobnicy którzy wygrają turniej są następnie krzyżowani. Sprawdziliśmy efektywność dwóch sposobów krzyżowania:

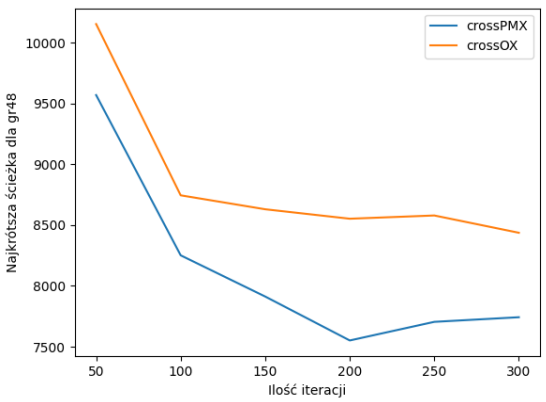
1: PMX

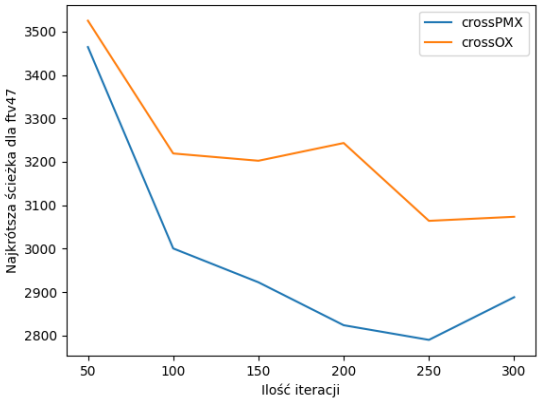
Losujemy 2 indeksy permutacji, a następnie tworzymy dzieci zamieniając środki (elementy pomiędzy wylosowanymi indeksami) rodziców. Jeżeli permutacja jest niedopuszczalna to „mapujemy" konfliktujące elementy, czyli wykorzystujemy wartości jako indeksy w drugim rodzicu. Robimy tak, aż rozwiążemy wszystkie konflikty.

2: OX

Jest to uproszczona wersja PMX-a. W tym wypadku zamiast od razu uzupełniać środki a następnie je naprawiać, od razu uzupełniamy środki brakującymi elementami, lecz w takiej kolejności w jakiej występują w drugim rodzicu.

Poniżej porównanie działania tych dwóch krzyżowań dla problemu symetrycznego oraz asymetrycznego:





Z tego badania wynika że PMX jest skuteczniejszym sposobem krzyżowania niż OX.

Złożoności:

1: PMX

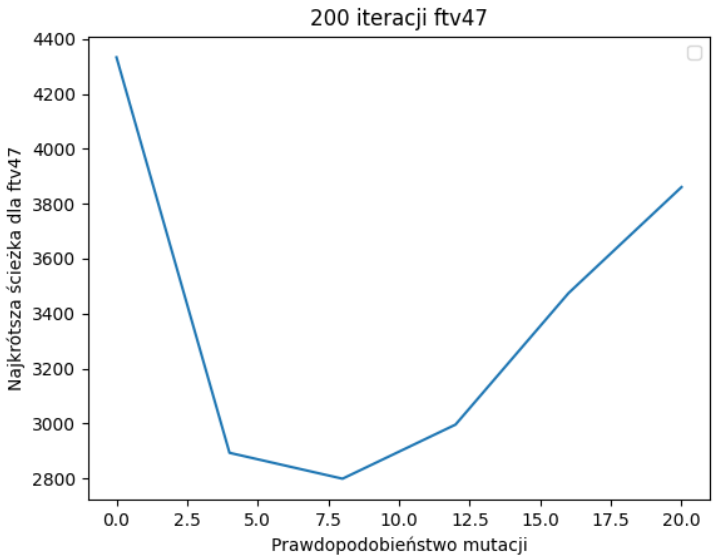
Podczas krzyżowania liczba elementów które wywołują konflikt zależy liniowo od długości permutacji (maksymalnie n/2 elementów). Mapowanie również będzie miało złożoność liniową, gdyż w najgorszym przypadku dla każdego elementu musimy wykonać tyle mapowań ile jest elementów w środkowym przedziale. Oznacza to że całkowita złożoność PMX wynosi O(n) * O(n) = O(n^2).

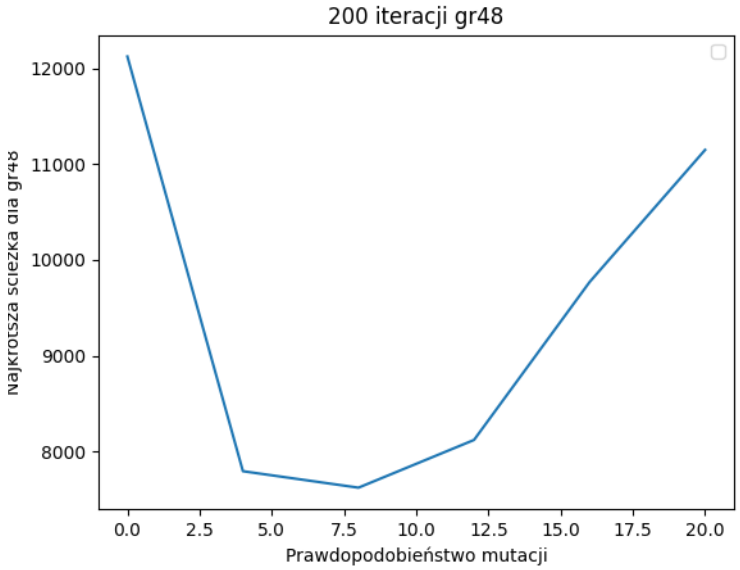
2: OX

Dla krzyżowania OX sprawa wygląda inaczej. Liczba elementów którymi musimy wypełnić środek zależy liniowo od n, lecz możemy sprawdzać czy należy konkretny element wstawić w czasie logarytmicznym. Dzieje się tak gdyż przed wstawieniem brzegi permutacji możemy posortować (O(n*log(n), oczywiście oryginały dalej mamy zapisane), a następnie używać algorytmu binary search aby sprawdzać, czy element w nich występuje. Całkowitą złożonością jest więc O(n*log(n)) + O(n)*O(log(n)) = O(n*log(n)).

Mutowanie

Nasza mutacja jest bardzo prosta. Każdy osobnik ma pewną szanse, na to aby dwa dowolne miasta zamieniły się miejscami. Mutacja może się powtórzyć dwukrotnie dla jednego osobnika, ale szansa jest spada geometrycznie.





Najbardziej efektywna jest mutacja między 5, a 10 procent. Większa sprawia że nasza populacja zbyt przypadkowo się porusza. Złożoność obliczeniowa to O(n), gdzie n to populacja, bo musimy przejść po wszystkich osobnikach i część z nich zmutować. Sama mutacja zajmuje O(1).

Starzenie

Musimy posiadać coś, aby nasza populacja nie wymknęła się z pod kontroli. Robimy to dzięki starzeniu. Każdy osobnik ma swój wiek który zwiększa się o jeden z każdym pokoleniem. Jeżeli przekroczymy pewną liczę osobników (Dla nas dwa razy większa startowa populacja). Sortujemy względem wieku po czym usuwamy najstarsze osobniki tak aby populacja nie przekraczała limitu. Złożoność czasowa to N*logN, gdzie N to liczebność populacji.

Podsumowanie

Analiza wykresów sporządzonych na podstawie zebranych w trakcie badań wyników pozwala zauważyć znaczącą korelację jakości uzyskanego rozwiązania z liczbą iteracji algorytmu oraz wielkością populacji osobników. Ponadto, porównana została również efektywność zaimplementowanych modeli selekcji. W każdym testowanym przypadku selekcja metodą turnieju okazywała się lepsza od ruletki. Należy jednak wspomnieć, że jeżeli wielkość turnieju jest liniowa to jest on wolniejszy od ruletki. Co do krzyżowania z naszego badania wynika, że jeżeli interesuje nas efektywność, należy wybrać PDX, lecz jeżeli cenimy szybkość algorytmu, OX będzie lepszym rozwiązaniem.