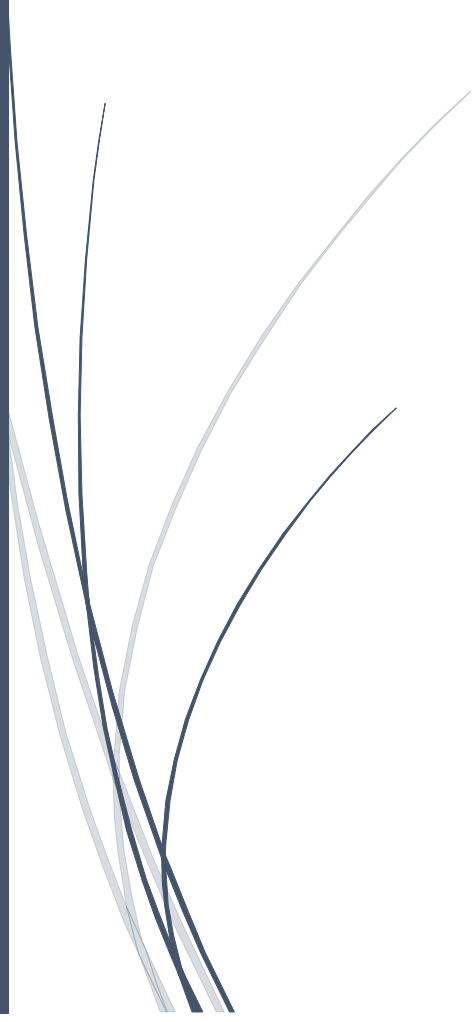
Algorytmy Ewolucyjne

Sprawozdanie z ćwiczenia 1.

Sztuczna Inteligencja i Inżynieria Wiedzy - laboratorium



Spis treści

Algorytm genetyczny – założenia implementacyjne

W ramach tego zadania zaimplementowałam algorytm genetyczny rozwiązujący problem komiwojażera. Algorytm opiera się na standardowym schemacie – tworzenia populacji, oceny osobników, tworzenia nowych pokoleń w oparciu o krzyżowanie, mutację i szukanie najlepszego rozwiązania. Pełny kod źródłowy implementacji będzie dołączony do sprawozdania.

Funkcja przystosowania

Jako funkcję przystosowania zaimplementowałam liczenie długości drogi – sumy odległości między miastami. W zależności od problemu, odległość obliczałam wg odległości Euklidesowej (2D) lub odległości geograficznej – z pominięciem promienia Ziemi (do porównywania wyników nie miałoby to wpływu).

Kodowanie rozwiązań

Każdy z osobników był reprezentowany przez ArrayListę Integerów. Ciąg liczb reprezentował kolejne miasta odwiedzane po drodze. Miasta numerowałam od 0 do N-1, gdzie N oznacza wielkość problemu (liczbę miast).

Implementacja metody losowej i zachłannej

Rozwiązania losowe były generowane przez wstawienie wszystkich miast do ArrayListy, a następnie wywołania metody *Collections.shuffle()*. Algorytm zachłanny został zaimplementowany standardowo, od punktu początkowego szukając dla każdego z odwiedzanych miast po kolei najbliższego punktu.

Początkowa populacja algorytmu genetycznego była tworzona metodą losową.

Implementacja operatorów

Operator selekcji

Zaimplementowałam dwa operatory selekcji – turniej i ruletkę. W przypadku metody turniejowej wybieram *n* razy losowych osobników, a następnie „wygrywa" najlepszy z nich. Do ruletki wykorzystałam funkcję nadającą wagę każdemu osobnikowi:

$$w_i = \max_j f(x_j) - f(x_i) + \varepsilon,$$

gdzie:

- w_i – waga i-tego osobnika,
- $f(\cdot)$ - funkcja przystosowania,
- x_j – j-ty osobnik populacji,
- ε – parametr odpowiadający za minimalny rozmiar wycinka koła ruletki, tzn. dla najgorszego osobnika w populacji (najgorszy osobnik bez ε miałby w_i równe 0).

Algorytm losował liczbę z przedziału od 0 do sumy wag osobników, a następnie sprawdzał, któremu osobnikowi ona odpowiada.

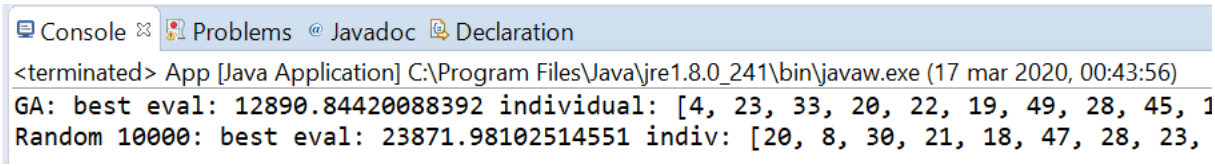
Selekcja turniejowa jest selekcją domyślną w przeprowadzonych badaniach.

Operatory krzyżowania i mutacji

Dla badanego algorytmu zaimplementowałam po 1 algorytmie krzyżowania i mutacji. Krzyżowanie wykonuję metodą PMX – *Partially-Matched Crossover*, a mutacja metodą *swap* – dla każdego osobnika jeden raz mogły zostać wylosowane dwie pozycje, na których wartości były zamieniane ze sobą.

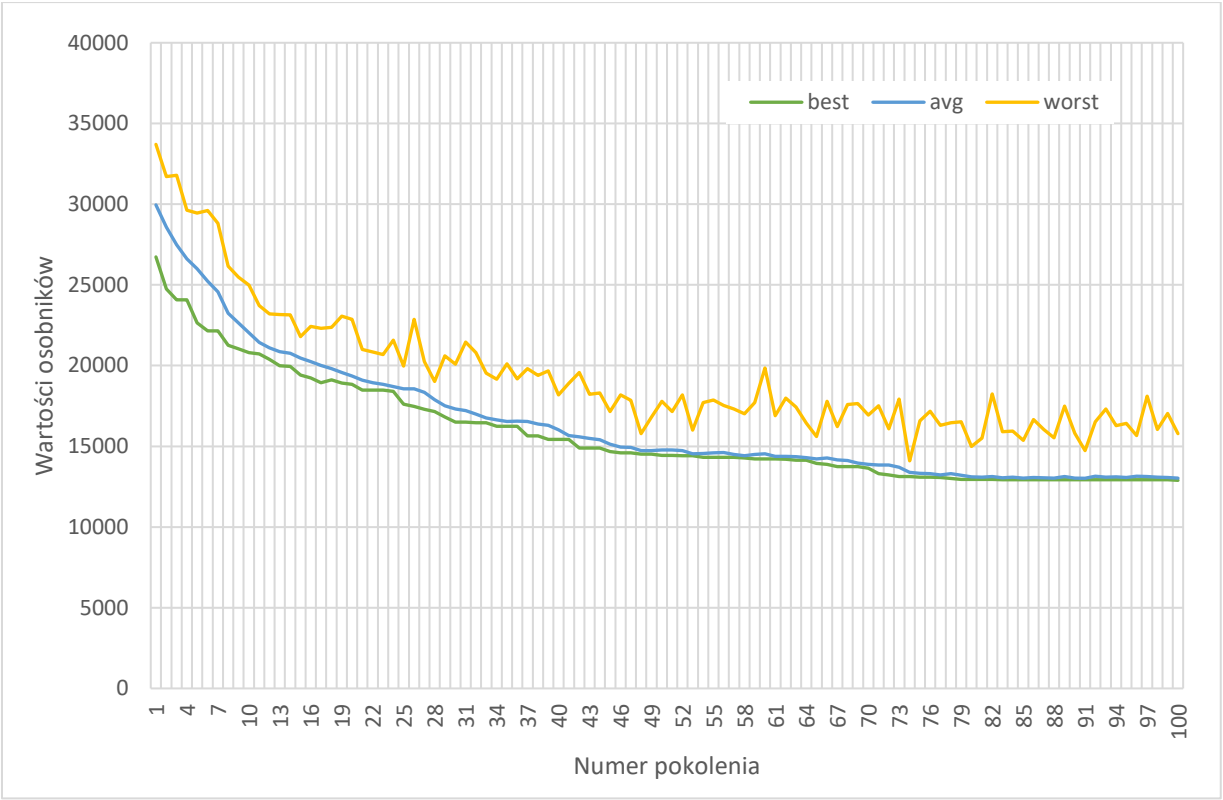
Sprawdzenie działania algorytmu

Dla sprawdzenia poprawności działania algorytmu genetycznego zaimplementowanego na poprzednich zajęciach uruchomiłam go dla problemu łatwego – berlin52.



Zrzut ekranu 1 Uruchomienie algorytmu genetycznego i losowego dla berlin52

Jak widać, algorytm (dla ustawień początkowych popSize=100, genNum=100, Px=0,7, Pm=0,1, Tour=5) daje dużo lepsze wyniki niż algorytm losowy przy 10 000 powtórzeń. Możemy także sprawdzić, jak rozkładają się rozwiązania w trakcie przebiegu metody:



***Rysunek 1** Wykres przebiegu algorytmu genetycznego dla problemu berlin 52 popSize=100, genNum=100, Px=0,7, Pm=0,1, Tour=5*

Widać na wykresie, jak rozwiązania zbiegają ku coraz lepszym rozwiązaniom, najlepszy osobnik w pokoleniu jest coraz mniejszy (nie jest gubiony), średnie wartości w populacji także są coraz lepsze, ale jednocześnie jest zachowany odstęp linii najgorszych osobników w pokoleniu, a więc zróżnicowanie populacji jest wystarczające, pojawiają się różne osobniki.

Jak wcześniej wspomniałam, dla wszystkich badań w tym sprawozdaniu metodą krzyżowania jest PMX (Partially-Matched Crossover), a metodą mutacji swap.

Zbadanie wpływu parametrów:

Wszystkie badania wpływu różnych parametrów na algorytm przeprowadziłam dla trzech instancji problemu o średnim poziomie trudności (przeznaczonych do strojenia metody) – kroA100, kroA150, kroA200. Dobór wartości optymalnych parametrów będę realizowała krokowo.

- ## rozmiar populacji

Badanie 1: wpływ rozmiaru populacji na działanie algorytmu genetycznego

Cel badania: znalezienie optymalnego rozmiaru populacji

Stałe w badaniu:
genNum=100
Px=0,7
Pm=0,1
tour=5

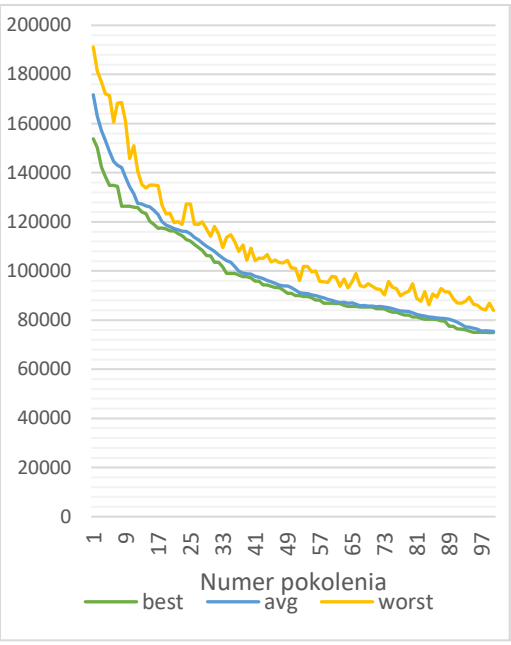
Zmienne w badaniu:
instancje problemu
rozmiar populacji (popSize)

Przebieg badania: Załadowanie danych, dziesięciokrotne uruchomienie algorytmu z zadanymi wartościami parametrów i zapis do pliku.

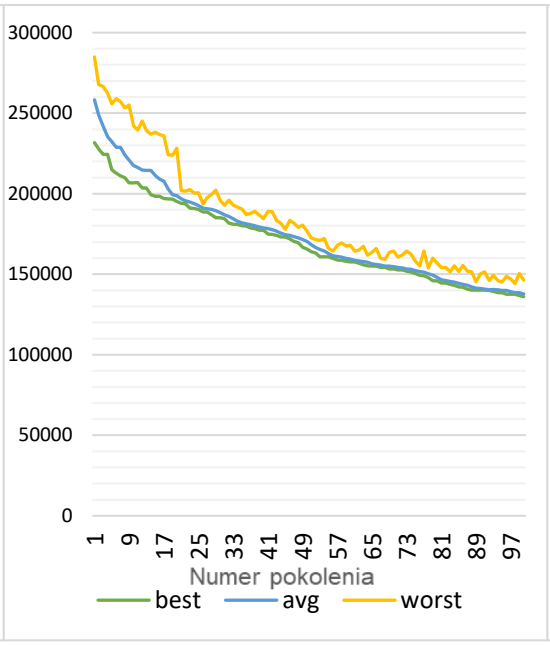
Wyniki:

Tabela 1 Wyniki 10 uruchomień algorytmu genetycznego dla kroA100, kroA150 i kroA200 przy rozmiarze populacji 100

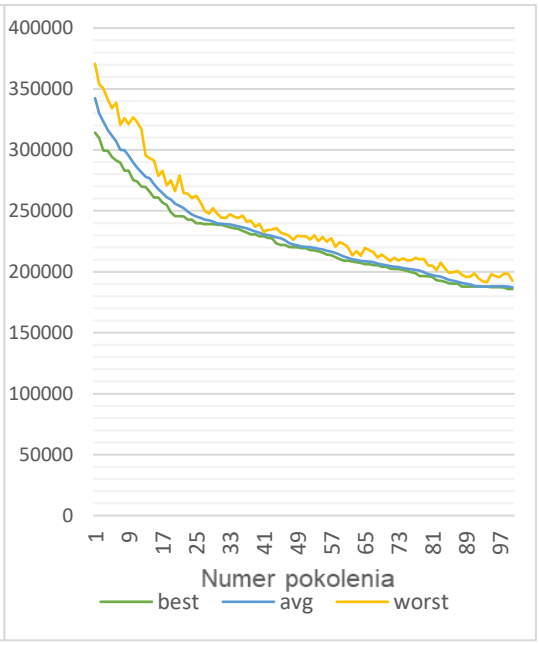
Problem	**kroA100**	**kroA150**	**kroA200**
Wyniki – najlepsze rozwiązanie popSize=100	74 923.955	136 058.605	185 835.856
	84 166.139	138 539.839	188 188.710
	82 015.760	136 057.258	196 839.293
	77 056.889	147 125.861	196 408.482
	77 665.018	127 191.658	181 704.790
	78 529.797	139 026.189	176 300.297
	80 646.365	128 222.884	193 002.506
	76 462.119	128 010.528	192 368.115
	77 353.713	135 041.645	192 405.088
	84 972.477	133 388.223	191 723.019
Średnia z 10	**79 379.223**	**134 866.269**	**189 477.615**



Rysunek 4 Wykres przebiegu algorytmu dla kroA100 przy popSize=100



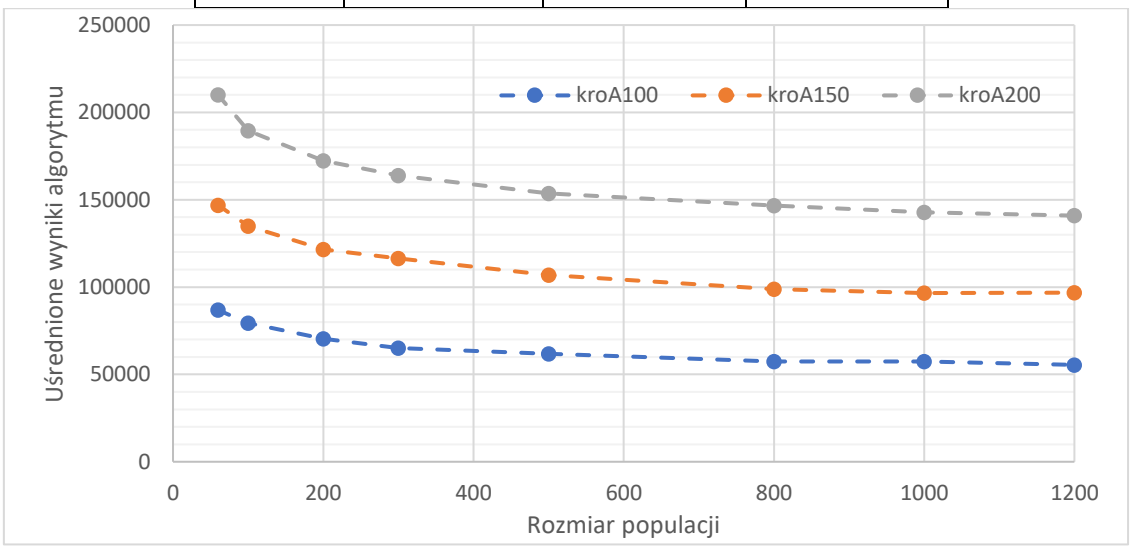
Rysunek 3 Wykres przebiegu algorytmu dla kroA150 przy popSize=100



Rysunek 2 Wykres przebiegu algorytmu dla kroA200 przy popSize=100

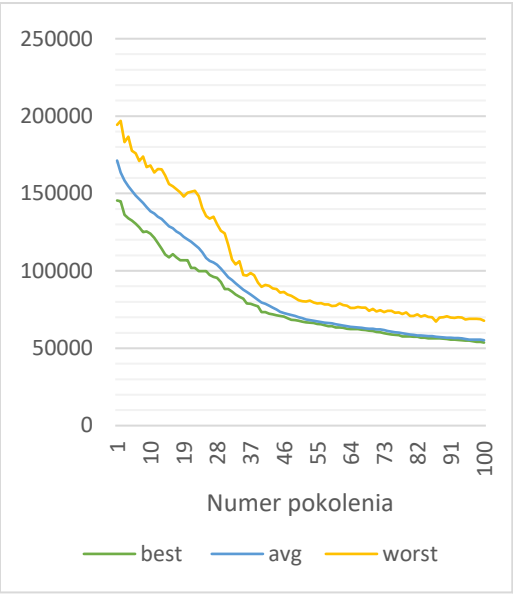
Tabela 2 Uśrednione z 10 wyniki algorytmu dla kroA100, kroA150, kroA200 w zależności od rozmiaru populacji

Rozmiar populacji	kroA100	kroA150	kroA200
60	86 854.861	146 915.525	210 006.522
100	79 379.223	134 866.269	189 477.615
200	70 413.360	121 476.521	172 394.106
300	65 087.584	116 410.067	163 905.387
500	61 800.933	106 839.086	153 630.161
800	57 434.508	98 753.228	146 572.809
1000	57 489.194	96 575.976	142 866.669
1200	55 459.927	96 874.736	140 899.953

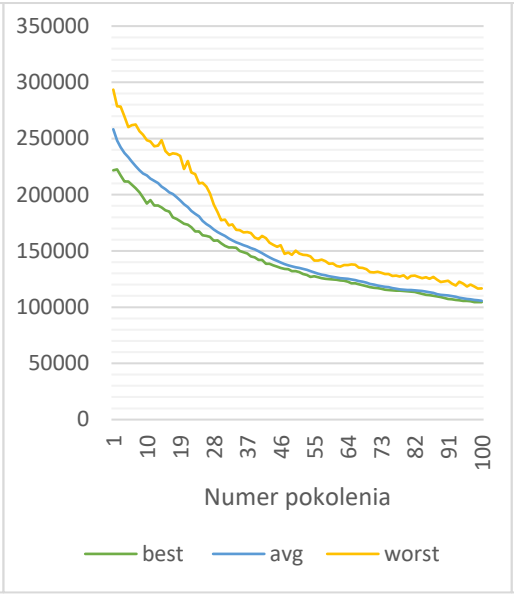


Rysunek 5 Wpływ rozmiaru populacji na wyniki algorytmu genetycznego

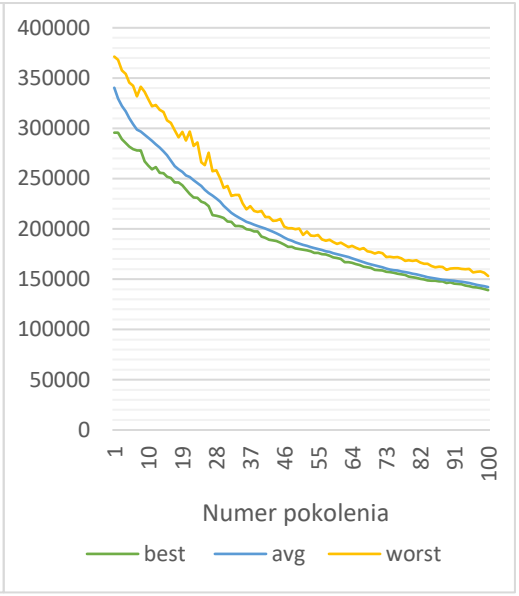
Wykres punktowy z linią przerywaną - celem linii jest ukazanie trendu, ale nie można interpretować pewnych wartości z wykresu.



Rysunek 8 Przebieg algorytmu dla kroA100 przy popSize=800



Rysunek 7 Przebieg algorytmu dla kroA150 przy popSize=800



Rysunek 6 Przebieg algorytmu dla kroA200 przy popSize=800

Wnioski:

Wyniki badań (Tabela 2) wskazują, że im większa populacja, tym wynik działania algorytmu lepszy. Widać jednak także na wykresie (Rysunek 5), że dla mniejszych wartości różnice są bardziej znaczące, później przebieg linii trendu „wypłaszcza" i różnice są coraz mniejsze. Jednak im większy rozmiar populacji, tym dłuższy czas kalkulacji, a także opieramy się mocniej na populacji początkowej, co zbliża algorytm do algorytmu losowego. Z tego powodu do dalszych badań wybiorę rozmiar populacji **800**.

Widzimy także na Rysunkach 2, 3 i 4, że dla wszystkich trzech problemów funkcja oceny w przebiegu algorytmu zbiega początkowo szybciej, później jednak wolniej. Porównując jednak dla rozmiaru populacji równego 100 (Rysunek 2, 3, 4) oraz 800 (Rysunek 6,7,8) widzimy, że przy większym rozmiarze populacji początkowy szybszy spadek jest gwałtowniejszy i trwa dłużej (o ok. 10 pokoleń).

- liczba pokoleń

Badanie 2: wpływ liczby pokoleń na działanie algorytmu genetycznego

Cel badania: znalezienie optymalnej liczby pokoleń

Stałe w badaniu:

popSize=800 (przyjęta wartość wyznaczona z Badania 1)
Px=0,7
Pm=0,1
tour=5

Zmienne w badaniu:

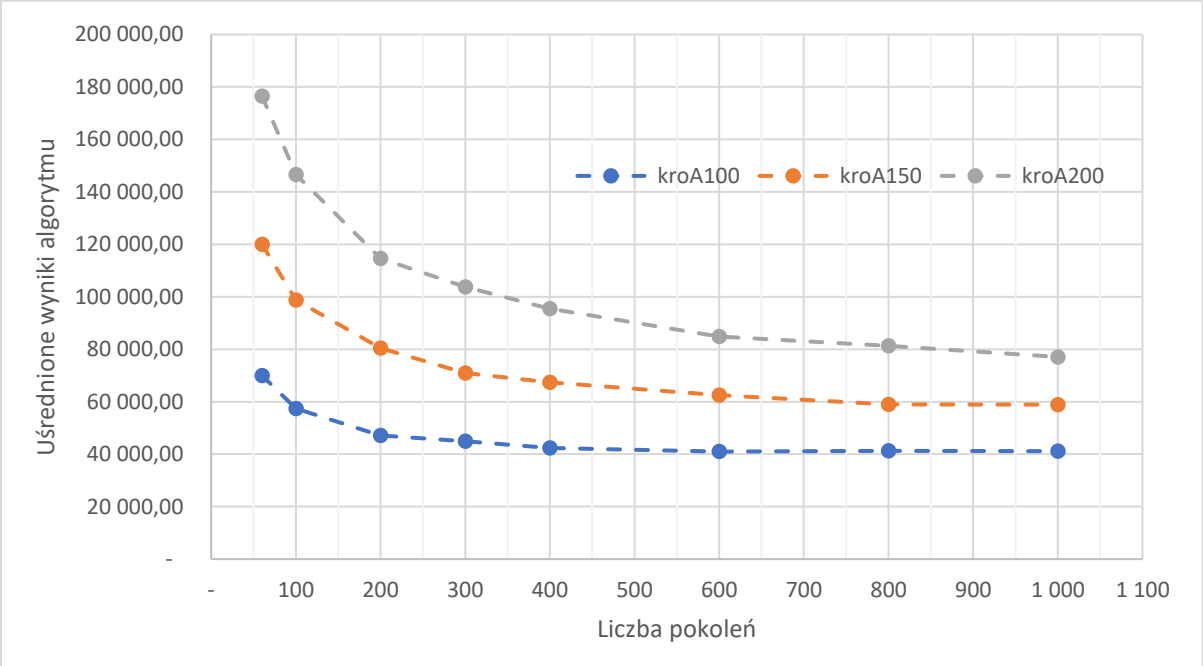
instancje problemu
liczba pokoleń (genNum)

Przebieg badania: Załadowanie danych, dziesięciokrotne uruchomienie algorytmu z zadanymi wartościami parametrów i zapis do pliku.

Wyniki:

Tabela 3 Uśrednione wyniki algorytmu dla kroA100, kroA150, kroA200 w zależności od liczby pokoleń

genNum	kroA100	kroA150	kroA200
60	69 995.07	120 003.60	176 501.94
100	57 434.51	98 753.23	146 572.81
200	47 093.54	80 456.68	114 642.35
300	44 950.41	70 947.95	103 764.38
400	42 402.49	67 356.40	95 421.76
600	40 984.61	62 488.58	84 807.58
800	41 284.17	58 994.81	81 281.57
1 000	41 152.69	58 902.13	77 089.74



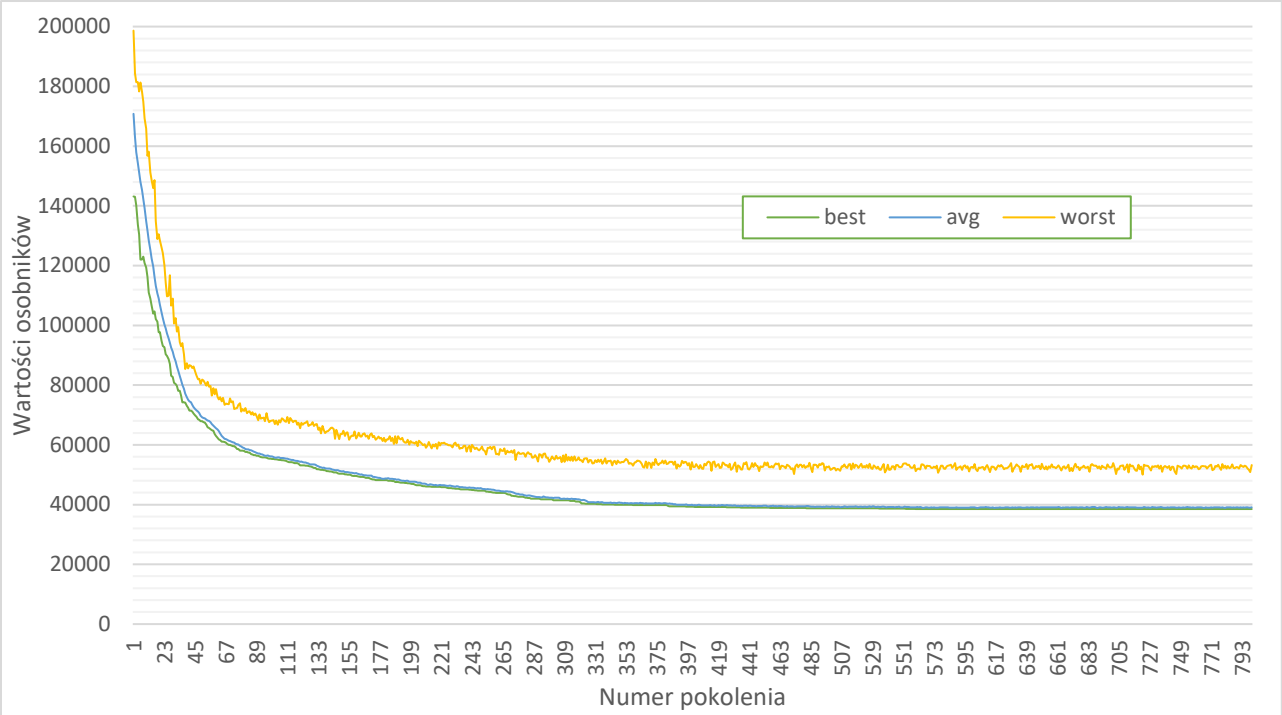
Rysunek 9 Wyniki algorytmu genetycznego w zależności od liczby generacji

Wykres punktowy z linią przerywaną - celem linii jest ukazanie trendu, ale nie można interpretować pewnych wartości z wykresu.

Wnioski:

Podobnie jak w przypadku wielkości populacji, im większa liczba pokoleń, tym lepsze wyniki daje algorytm, jednak dla mniejszych wartości różnice w wynikach są znaczące, dla większych niewielkie.

W przypadku kroA100 widać, że powyżej 600 iteracji wyniki są właściwie takie same, a różnice wynikają jedynie z losowości algorytmu (kroA200 jest bardziej złożony, więc różnice są jeszcze widoczne). Nie powinno to dziwić, jeżeli spojrzymy na wykres przebiegu algorytmu dla kroA100 (Rysunek 10) – mimo wystarczającej różnorodności populacji (worst jest stosunkowo oddalony od best), kilkaset ostatnich pokoleń algorytm nie jest już w stanie „ruszyć z miejsca” i poprawić wyniku.



Rysunek 10 Przebieg algorytmu dla kroA100 przy 800 generacjach

Z tego powodu (oraz znacznie dłuższego czasu wykonywania przy 1000 generacji) do dalszych badań parametrów wybiorę liczbę generacji równą **600**.

- rozmiar turnieju

Badanie 3: wpływ rozmiaru turnieju na działanie algorytmu genetycznego

Cel badania: znalezienie optymalnego rozmiaru turnieju

Stałe w badaniu:

popSize=800 (przyjęta wartość wyznaczona z Badania 1)
genNum=600 (przyjęta wartość wyznaczona z Badania 2)
Px=0,7
Pm=0,1

Zmienne w badaniu:

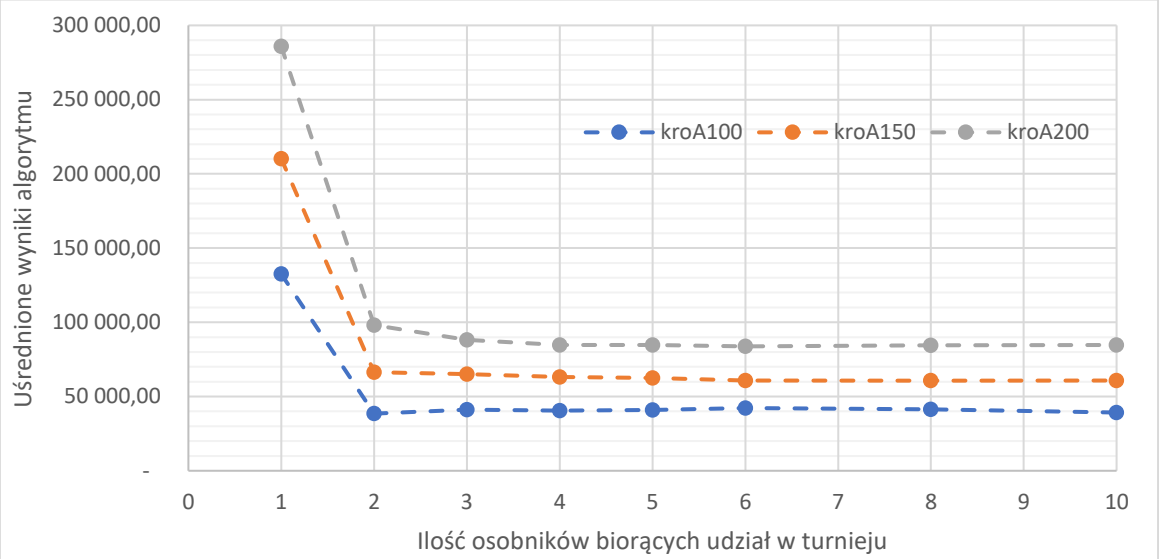
instancje problemu
rozmiar turnieju (tour)

Przebieg badania: Załadowanie danych, dziesięciokrotne uruchomienie algorytmu z zadanymi wartościami parametrów i zapis do pliku.

Wyniki:

Tabela 4 Uśrednione wyniki algorytmu dla kroA100, kroA150, kroA200 w zależności od rozmiaru turnieju

Rozmiar turnieju	kroA100	kroA150	kroA200
1	132 531.55	210 217.28	285 896.16
2	38 498.85	66 331.05	98 178.35
3	41 147.39	65 177.47	88 234.46
4	40 446.96	63 175.00	84 761.31
5	40 984.61	62 488.58	84 807.58
6	42 282.50	60 747.30	83 790.74
8	41 396.88	60 691.85	84 552.58
10	39 271.00	60 802.85	84 696.71



Rysunek 11 Wpływ rozmiaru turnieju na wyniki algorytmu

Wykres punktowy z linią przerywaną - celem linii jest ukazanie trendu, ale nie można interpretować pewnych wartości z wykresu.

Wnioski:

Wyraźnie widać, że dla rozmiaru turnieju równego 1 algorytm działa najsłabiej – nic dziwnego, za rodziców bierze wtedy losowych osobników, nie ma żadnego ciśnienia selekcyjnego. Zwiększanie jednak turnieju powyżej 2, do 6 poprawia wyniki, jednak bardzo nieznacznie. W ten sposób jest zwiększane ciśnienie selekcyjne, jednak wciąż każdy osobnik uczestniczący w turnieju jest wybierany na sposób losowy. Dalsze zwiększanie rozmiaru turnieju nie przynosi żadnych rezultatów w przypadku badanych instancji problemów, a różnice są nieznaczne i wynikają z losowości algorytmu. Do dalszych badań za optymalny uznam rozmiar turnieju równy **5**.

Porównanie operatorów selekcji – ruletka i turniej

W swoich poprzednich badaniach wykorzystywałam operator selekcji turniejowej. Dla popSize=800, genNum=600, Px=0,7, Pm=0,1, tour=4 uzyskałam wyniki:

Tabela 5 Wyniki algorytmu przy selekcji turniejowej (popSize=800, genNum=600, Px=0,7, Pm=0,1, tour=4)

kroA100	**kroA150**	**kroA200**
40 446.96	63 175.00	84 761.31

Dla popSize=800, genNum=1000, Px=0,7, Pm=0,1, Tour=5:

Tabela 6 Wyniki algorytmu przy selekcji turniejowej (popSize=800, genNum=1000, Px=0,7, Pm=0,1, Tour=5)

kroA100	**kroA150**	**kroA200**
41 152.69	58 902.13	77 089.74

Po uruchomieniu algorytmu z ruletką dla epsilon = 1 otrzymałam dla popSize=800, genNum=600, Px=0,7, Pm=0,1:

Tabela 7 Wyniki algorytmu z selekcją ruletką (popSize=800, genNum=600, Px=0,7, Pm=0,1, epsilon=1)

kroA100	**kroA150**	**kroA200**
76 243.07	151 761.18	224 385.65

Widać więc, że przy takich parametrach ruletka osiąga rażąco gorsze wyniki od metody turniejowej. Pomyślałam, że być może epsilon jest zbyt duże i słabsze osobniki mają zbyt duże szanse na zostanie wylosowanym, zbadałam więc wyniki w zależności od epsilon.

Badanie 4: wpływ epsilon na działanie algorytmu genetycznego z ruletką

Cel badania: znalezienie optymalnej wartości parametru epsilon

Stałe w badaniu:

popSize=800 (przyjęta wartość wyznaczona z Badania 1)
genNum=600 (przyjęta wartość wyznaczona z Badania 2)
Px=0,7
Pm=0,1

Zmienne w badaniu:

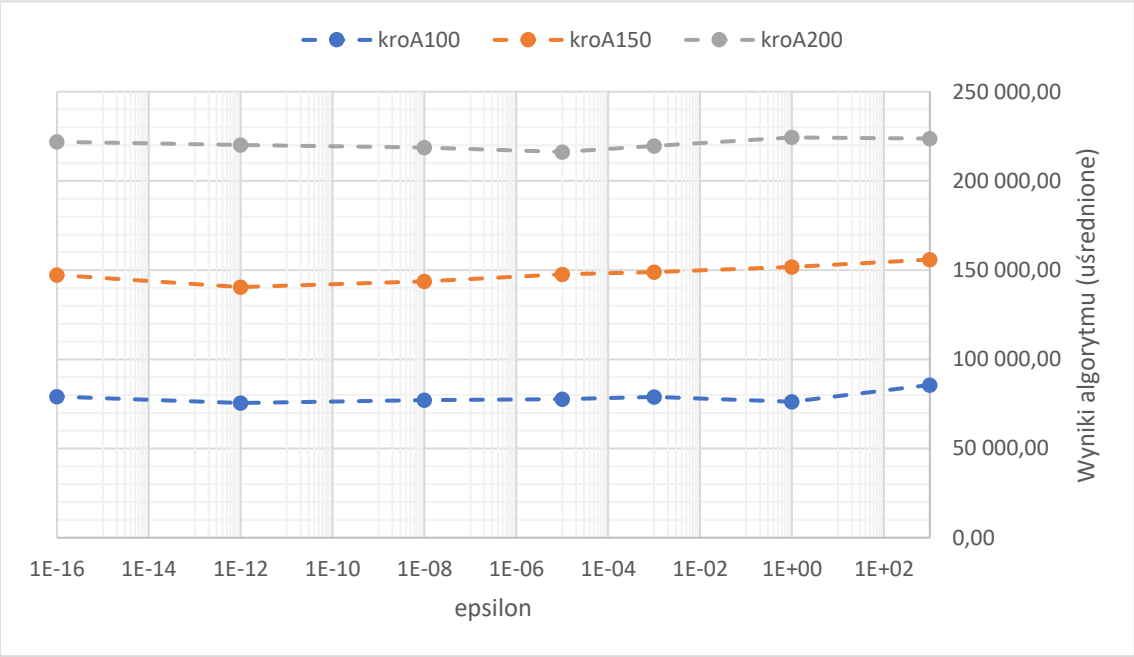
instancje problemu
wartość epsilon

Przebieg badania: Załadowanie danych, dziesięciokrotne uruchomienie algorytmu z zadanymi wartościami parametrów i zapis do pliku.

Wyniki:

Tabela 8 Wyniki algorytmu z ruletką dla kroA100, kroA150, kroA200 w zależności od epsilon

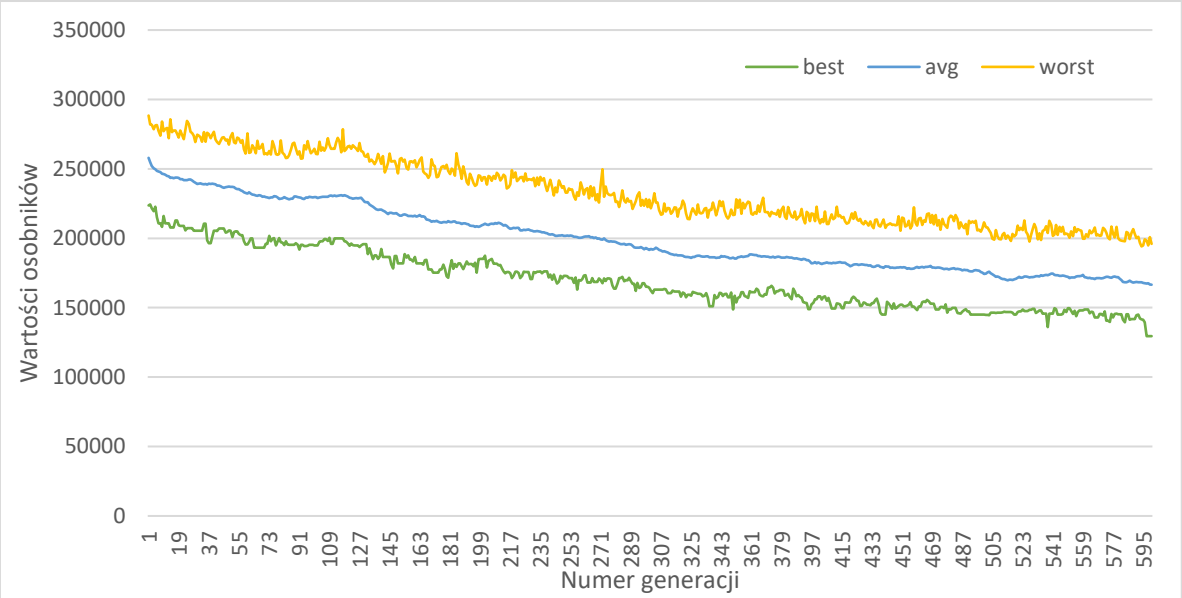
epsilon	kroA100	kroA150	kroA200
1 000	85 645.48	155 900.29	223 724.72
1	76 243.07	151 761.18	224 385.65
0.0010000000000000	78 928.13	148 945.70	219 572.12
0.0000100000000000	77 740.34	147 676.65	216 252.09
0.0000000100000000	77 067.15	143 794.14	218 774.12
0.0000000000010000	75 515.11	140 504.21	220 204.62
0.0000000000000001	79 090.97	147 272.08	221 912.31



Rysunek 12 Wpływ parametru epsilon na algorytm genetyczny z ruletką - skala logarytmiczna
Wykres punktowy z linią przerywaną - celem linii jest ukazanie trendu, ale nie można interpretować pewnych wartości z wykresu.

Wnioski:

Widać jednak, że epsilon nie ma aż tak dużego wpływu na wyniki. Wartości różnią się między sobą bardzo nieznacznie, co więcej, gdyby wybierać jedną wartość, dla której algorytm zwraca najlepsze wyniki – dla każdego z problemów byłaby to inna wartość. Do dalszych badań wybiorę więc epsilon równe 0.00000001 (1E-8) (próba kompromisu dla 3 problemów, aby wyniki były choć trochę lepsze).



Rysunek 13 Przebieg algorytmu z ruletką dla kroA150 z epsilon=0.00000001, popSize=800, genNum=600, Px=0,7, Pm=0,1

Jak widać na Rysunku 13, zupełnie inaczej wygląda także przebieg algorytmu dla ruletki (z epsilon=0.00000001, popSize=800, genNum=600, Px=0,7, Pm=0,1). Wygląda jak sam początek przebiegu, jednak choć funkcja oceny znalezionych rozwiązań jest wyraźnie malejąca, to maleje dużo wolniej, średnie wartości nie są zbliżone do tych najlepszych, a najlepsze rozwiązania często są gubione. Wynika to być może z małego ciśnienia selekcyjnego w przypadku ruletki.

Ponieważ zaniepokoiło i zastanowiło mnie działanie algorytmu z ruletką, postanowiłam zbadać także wielkość populacji.

Badanie 5: wpływ wielkości populacji na działanie algorytmu genetycznego z ruletką

Cel badania: znalezienie optymalnej wielkości populacji dla algorytmu z ruletką

Stałe w badaniu:

genNum=600 (przyjęta wartość wyznaczona z Badania 2)
epsilon=0.00000001 (przyjęta wartość wyznaczona po Badaniu 4)
Px=0,7
Pm=0,1

Zmienne w badaniu:

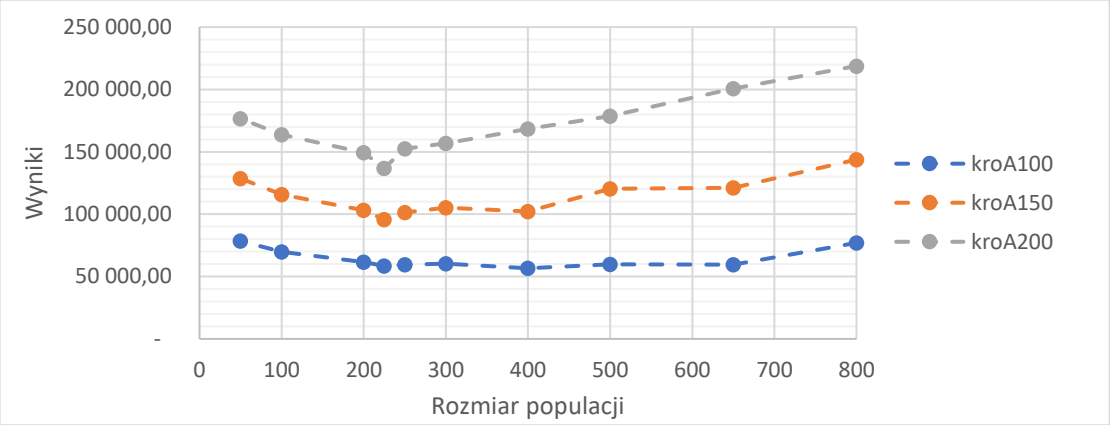
instancje problemu
wielkość populacji (popSize)

Przebieg badania: Załadowanie danych, dziesięciokrotne uruchomienie algorytmu z zadanymi wartościami parametrów i zapis do pliku.

Wyniki:

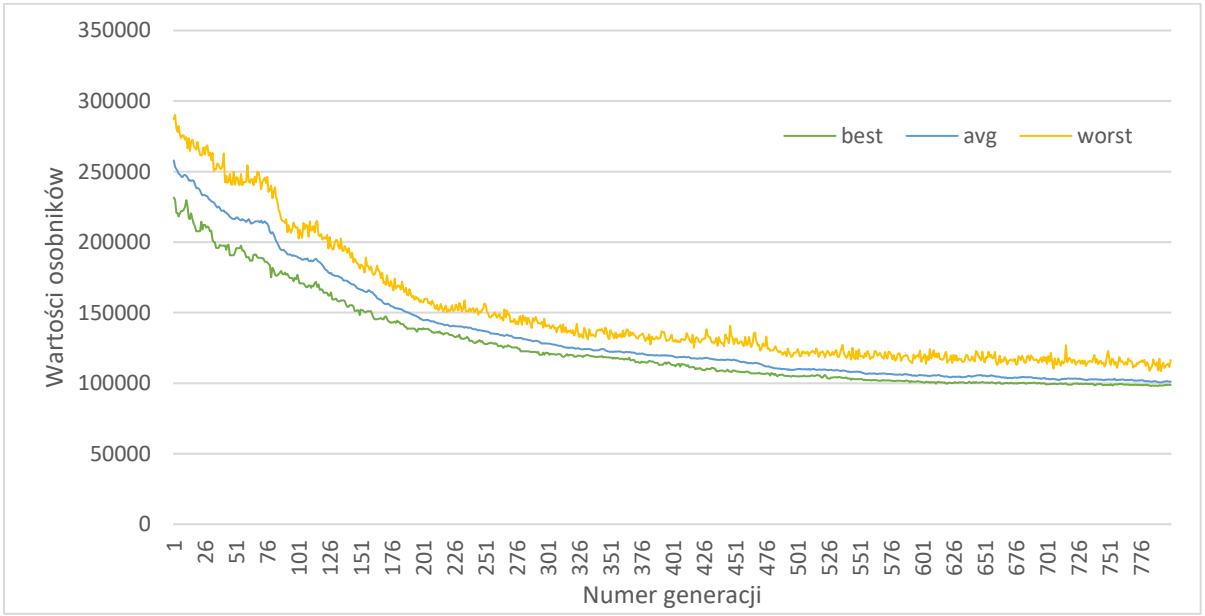
Tabela 9 Uśrednione wyniki algorytmu z ruletką dla kroA100, kroA150, kroA200 (epsilon=0.00000001, genNum=800, Px=0,7, Pm=0,1)

popSize	kroA100	kroA150	kroA200
800	77 067.15	143 794.14	218 774.12
300	60 291.67	105 244.83	156 881.52
250	59 583.11	101 505.73	152 516.73
200	61 495.56	103 093.59	149 318.18
100	69 909.07	115 846.98	163 739.62
50	78 497.07	128 552.22	176 578.53



Rysunek 14 Wpływ rozmiaru populacji na algorytm genetyczny z ruletką
Wykres punktowy z linią przerywaną - celem linii jest ukazanie trendu, ale nie można interpretować pewnych wartości z wykresu.

Wnioski: Wpływ rozmiaru populacji jest zupełnie inny dla algorytmu z ruletką niż algorytmu z selekcją turniejową. Wyniki są gorsze dla mniejszych populacji, ale także dla zbyt dużych populacji. Najlepsze wyniki algorytm osiąga w przedziale 200-250.



Rysunek 15 Przebieg algorytmu z ruletką dla kroA150 przy popSize=225 z epsilon=0.00000001, genNum=800, Px=0,7, Pm=0,1

Na Rysunku 15 widać także, że przy tak dobranych parametrach (popSize=225) przebieg algorytmu wygląda już dużo lepiej, „normalnie”- kształt ma zbliżony do algorytmu z turniejem. Widać więc, że algorytm i operator działa. Na podstawie wykresu (Rysunek 15) można wywnioskować także, że jeżeli chodzi o liczbę pokoleń, to sposób selekcji rodziców nie ma na to większego wpływu – powyżej 600 pokoleń poprawa rozwiązań pozostaje niewielka, podobnie jak w przypadku selekcji turniejowej.

Udało mi się więc poprawić uzyskać wyniki z ruletką (dla popSize=225 z epsilon=0.00000001, genNum=800, Px=0,7, Pm=0,1) :

Tabela 10 Wyniki algorytmu z ruletką (dla popSize=225 z epsilon=0.00000001, genNum=800, Px=0,7, Pm=0,1)

kroA100	kroA150	kroA200
58 482.01	95 739.50	136 872.35

co wciąż w porównaniu z wynikami algorytmu z turniejem (dla popSize=800, genNum=800, Px=0,7, Pm=0,1, Tour=5):

Tabela 11 Wyniki algorytmu z selekcją turniejową dla popSize=800, genNum=800, Px=0,7, Pm=0,1, Tour=5)

kroA100	kroA150	kroA200
41 284.17	58 994.81	81 281.57

jest dużo słabszym wynikiem.

Wniosek:

Operator selekcji turniejowej daje lepsze wyniki działania algorytmu genetycznego niż ruletka.

Zbadanie wpływu parametrów:

- ## prawdopodobieństwo krzyżowania

Badanie 6: wpływ prawdopodobieństwa krzyżowania na działanie algorytmu

Cel badania: znalezienie optymalnej wartości prawdopodobieństwa krzyżowania

Stałe w badaniu:

popSize=800 (przyjęta wartość wyznaczona z Badania 1)
genNum=600 (przyjęta wartość wyznaczona z Badania 2)
Pm=0,1
tour=5 (przyjęta wartość wyznaczona z Badania 3)

Zmienne w badaniu:

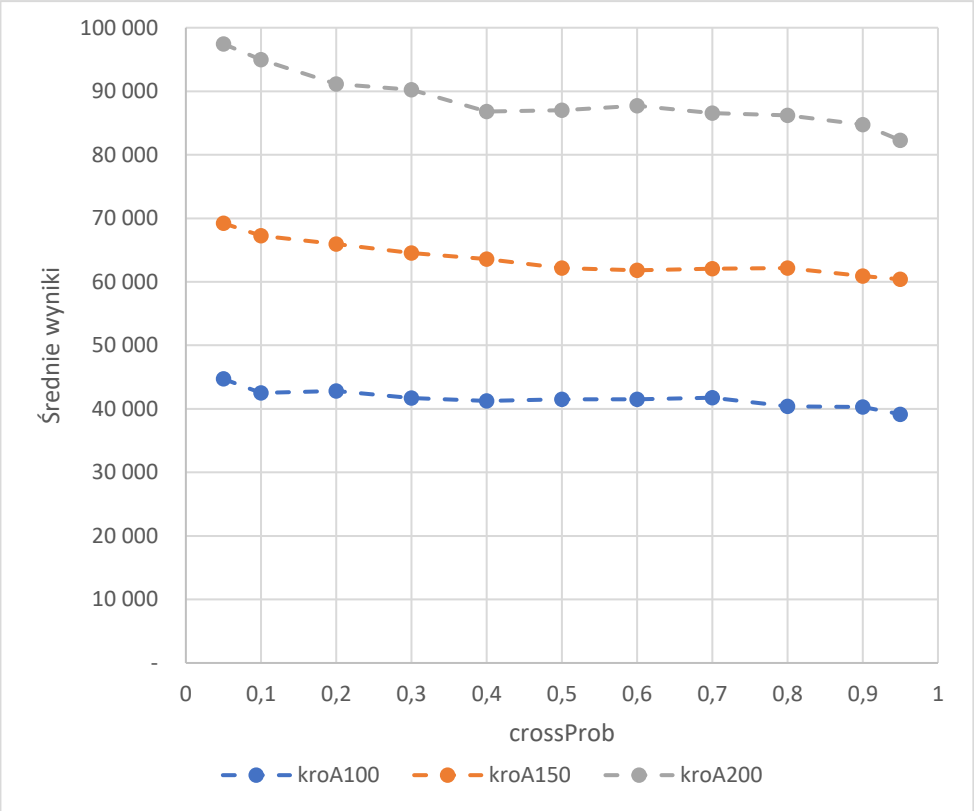
instancje problemu
prawdopodobieństwo krzyżowania (Px)

Przebieg badania: Załadowanie danych, **dwudziestokrotne**[1] uruchomienie algorytmu z zadanymi wartościami parametrów i zapis do pliku.

Wyniki:

Tabela 12 Uśrednione wyniki dla kroA100, kroA200, kroA150, kroA200 w zależności od prawdopodobieństwa krzyżowania (popSize=800, genNum=600, Pm=0,1, tour=5)

Px	kroA100	kroA150	kroA200
0.05	44 717.76	69 221.00	97 450.38
0.1	42 503.40	67 252.11	94 954.87
0.2	42 813.18	65 935.37	91 166.59
0.3	41 701.11	64 525.04	90 245.34
0.4	41 236.36	63 571.03	86 809.96
0.5	41 525.01	62 149.87	87 028.43
0.6	41 479.94	61 822.25	87 711.91
0.7	41 756.58	62 057.88	86 563.91
0.8	40 373.82	62 165.13	86 227.44
0.9	40 317.79	60 926.29	84 747.34
0.95	39 159.22	60 392.23	82 279.02



Rysunek 16 Wpływ prawdopodobieństwa krzyżowania na algorytm genetyczny

Wykres punktowy z linią przerywaną - celem linii jest ukazanie trendu, ale nie można interpretować pewnych wartości z wykresu.

[1] Przy wartościach uśrednionych z 10 uruchomień błąd wynikający z losowości algorytmu był większy niż różnice między poszczególnymi wartościami, dlatego postanowiłam powtórzyć doświadczenie dla kolejnych 10 i uśrednić wartości z 20 uruchomień.

Wnioski:

Badania pokazują, że prawdopodobieństwo krzyżowania nie ma aż tak znaczącego wpływu na wyniki algorytmu, jak wielkość czy ilość populacji (przynajmniej dla pozostałych parametrów wybranych we wcześniejszych badaniach). Generalnie widać jednak, że wykresy są malejące, a więc lepsze wyniki osiągane są przy wyższym prawdopodobieństwie krzyżowania. Wyższe prawdopodobieństwo krzyżowania także wydłuża czas kalkulacji. Mimo tego, do dalszych rozważań wybiorę **Px=0,95**

- prawdopodobieństwo mutacji

Badanie 7: wpływ prawdopodobieństwa mutacji na działanie algorytmu

Cel badania: znalezienie optymalnej wartości prawdopodobieństwa mutacji

Stałe w badaniu:

popSize=800 (przyjęta wartość wyznaczona z Badania 1)
genNum=600 (przyjęta wartość wyznaczona z Badania 2)
Px=0,95 (przyjęta wartość wyznaczona z Badania 6)
tour=5 (przyjęta wartość wyznaczona z Badania 3)

Zmienne w badaniu:

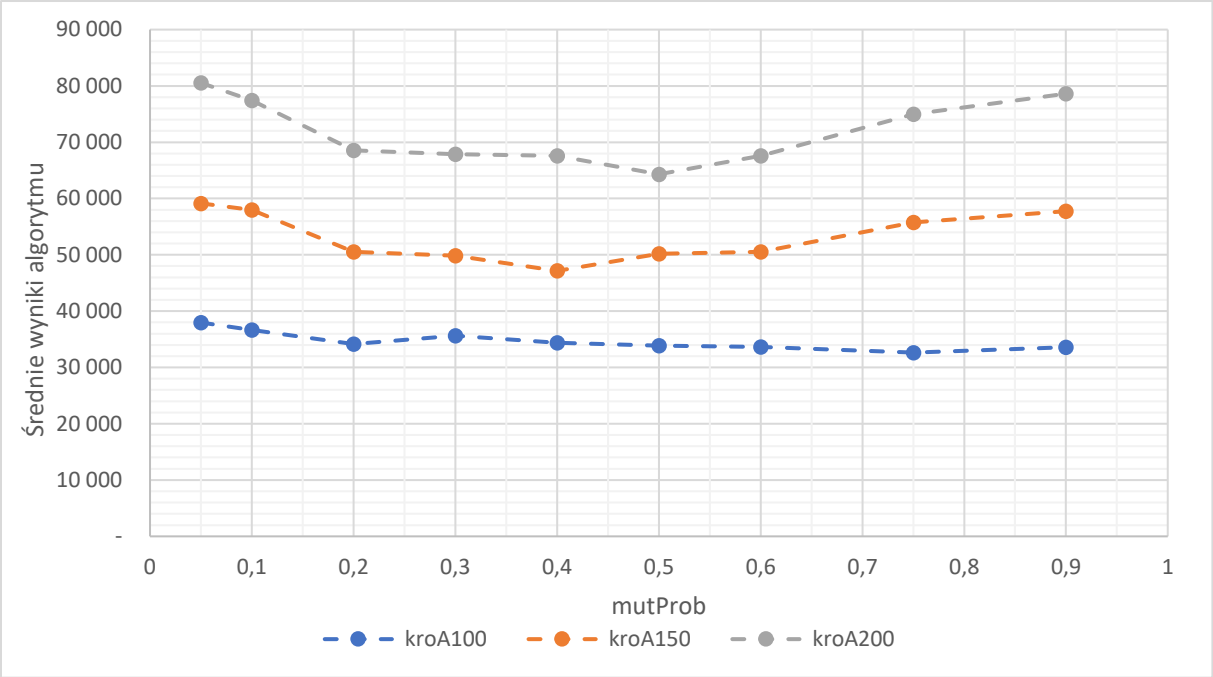
instancje problemu
prawdopodobieństwo mutacji (Pm)

Przebieg badania: Załadowanie danych, dziesięciokrotne uruchomienie algorytmu z zadanymi wartościami parametrów i zapis do pliku.

Wyniki:

Tabela 13 Uśrednione wyniki dla kroA100, kroA150, kroA200 w zależności od prawdopodobieństwa mutacji (popSize=800, genNum=600, Px=0,95, tour=5)

Pm	kroA100	kroA150	kroA200
0.05	37 965.87	59 140.42	80 538.89
0.10	36 644.66	57 948.72	77 431.31
0.20	34 153.54	50 512.80	68 536.41
0.30	35 642.79	49 839.38	67 860.68
0.40	34 355.55	47 148.91	67 596.34
0.50	33 876.95	50 192.92	64 312.02
0.60	33 639.02	50 547.39	67 564.45
0.75	32 639.81	55 770.59	74 965.88
0.90	33 561.94	57 747.33	78 620.08



Rysunek 17 Wpływ prawdopodobieństwa mutacji na algorytm genetyczny

Wykres punktowy z linią przerywaną - celem linii jest ukazanie trendu, ale nie można interpretować pewnych wartości z wykresu.

Wnioski:

Wpływ prawdopodobieństwa mutacji na wynik działania algorytmu także jest dużo mniejszy niż rozmiaru lub liczby populacji. Widać jednak, że prawdopodobieństwo nie może być ani za małe – element losowy jest pomocny przy szukaniu nowych, lepszych rozwiązań, ani za duże – wtedy losowość jest za duża i osobniki powstałe z krzyżowania dwóch osobników zostają zmutowane przed wejściem w populację. Najlepsze wyniki daje prawdopodobieństwo mutacji w przedziale <0,2-0,6>. Może być ono stosunkowo wysokie, ponieważ każdy osobnik może być zmutowany w mojej implementacji tylko raz i mutacja ma postać zamiany dwóch wartości miejscami (*swap*). Do ostatecznych badań wykorzystam **Pm=0,4.**

Porównanie algorytmu genetycznego z metodami „naiwnymi”

Badanie 8: porównanie wyników działania algorytmu genetycznego z algorytmem losowym i zachłannym

Cel badania: Porównanie algorytmu genetycznego z metodami „naiwnymi”

Stałe w badaniu:

Parametry dla algorytmu genetycznego:

popSize=800 (przyjęta wartość wyznaczona z Badania 1)

genNum=800 (przyjęta wartość wyznaczona z Badania 2, zwiększona jednak ze względu na sprawdzanie także trudniejszych problemów)

Px=0,95 (przyjęta wartość wyznaczona z Badania 6)

Pm=0,4 (przyjęta wartość wyznaczona z Badania 7)

tour=5 (przyjęta wartość wyznaczona z Badania 3)

Zmienne w badaniu:

instancje problemu

algorytmy

Przebieg badania: Załadowanie danych dla każdej z instancji, dziesięciokrotne uruchomienie algorytmu genetycznego z zadanymi wartościami parametrów, uruchomienie algorytmu losowego (z liczbą powtórzeń 640 tys.) oraz algorytmu zachłannego dla każdego z miast i zapis do pliku.

Wyniki:

Tabela 14 Porównanie działania algorytmów

Instancja	Optymalny wynik	Alg. Losowy [640k]				Alg. Zachłanny[N]				Alg. Ewolucyjny[10x]			
		best	worst	avg	std	best	worst	avg	std	best	worst	avg	std
berlin52	7 542	22 251.36	35 911.68	29 910.95	1 579.03	8 182.19	10 299.37	9 373.43	471.61	8 642.59	10 442.34	9 324.32	511.36
kroA100	21 282	132 485.38	209 217.12	171 080.51	8 231.59	24 698.50	28 694.68	27 045.21	812.11	32 996.55	45 074.52	37 654.34	3 799.65
kroA150	26 524	209 963.50	307 710.56	257 605.64	10 116.96	31 482.02	35 935.29	33 639.92	862.41	48 073.98	54 880.96	52 044.15	2 304.93
kroA200	29 368	282 639.66	396 992.73	340248.9486	11 781.39	34 547.69	42 045.19	37 468.91	1 405.40	62 178.58	75 463.65	68 121.44	4 554.81
fl417	11 861	431 127.53	558 991.31	496022.9349	13 571.91	13 802.38	16 722.67	15 724.26	493.47	99 916.45	118 566.87	106 794.15	5 920.26

Wnioski:

Algorytm losowy, nawet przy dużej liczbie powtórzeń i niewielkiej wielkości problemu nie daje zbyt satysfakcjonujących wyników. Algorytm genetyczny przy sprawdzeniu tej samej ilości osobników daje dużo lepsze wyniki, średnio ok. 4 razy lepsze (wartości best), ma także dużo niższe odchylenie standardowe (nie dziwi to, szczególnie, że porównujemy odchylenie 640 tys. z odchyleniem z 10 wyników).

Algorytm zachłanny daje dużo lepsze wyniki niż pozostałe 2 algorytmy (w krótszym czasie), są one stosunkowo zbliżone do wyników optymalnych. Być może algorytm genetyczny także dotarłby do podobnych wyników, jednak po duużo dłuższym czasie obliczeń, jednak ciężko to stwierdzić. Trzeba by sprawdzić także inne operatory krzyżowania i mutacji oraz przeprowadzić więcej badań nad parametrami i ich zależnościami między sobą.

Postanowiłam sprawdzić także, czy po zmianie metody inicjalizacji algorytm genetyczny byłby w stanie poprawić wyniki algorytmu genetycznego.

Badanie 9: porównanie inicjalizacji losowej i zachłannej

Cel badania: Porównanie działania algorytmu genetycznego z inicjalizacją losową i zachłanną

Stałe w badaniu:

genNum=800 (przyjęta wartość wyznaczona z Badania 2, zwiększona jednak ze względu na sprawdzanie także trudniejszych problemów)
Px=0,95 (przyjęta wartość wyznaczona z Badania 6)
Pm=0,4 (przyjęta wartość wyznaczona z Badania 7)
tour=5 (przyjęta wartość wyznaczona z Badania 3)

Zmienne w badaniu:

instancje problemu
popSize
metoda inicjalizacji

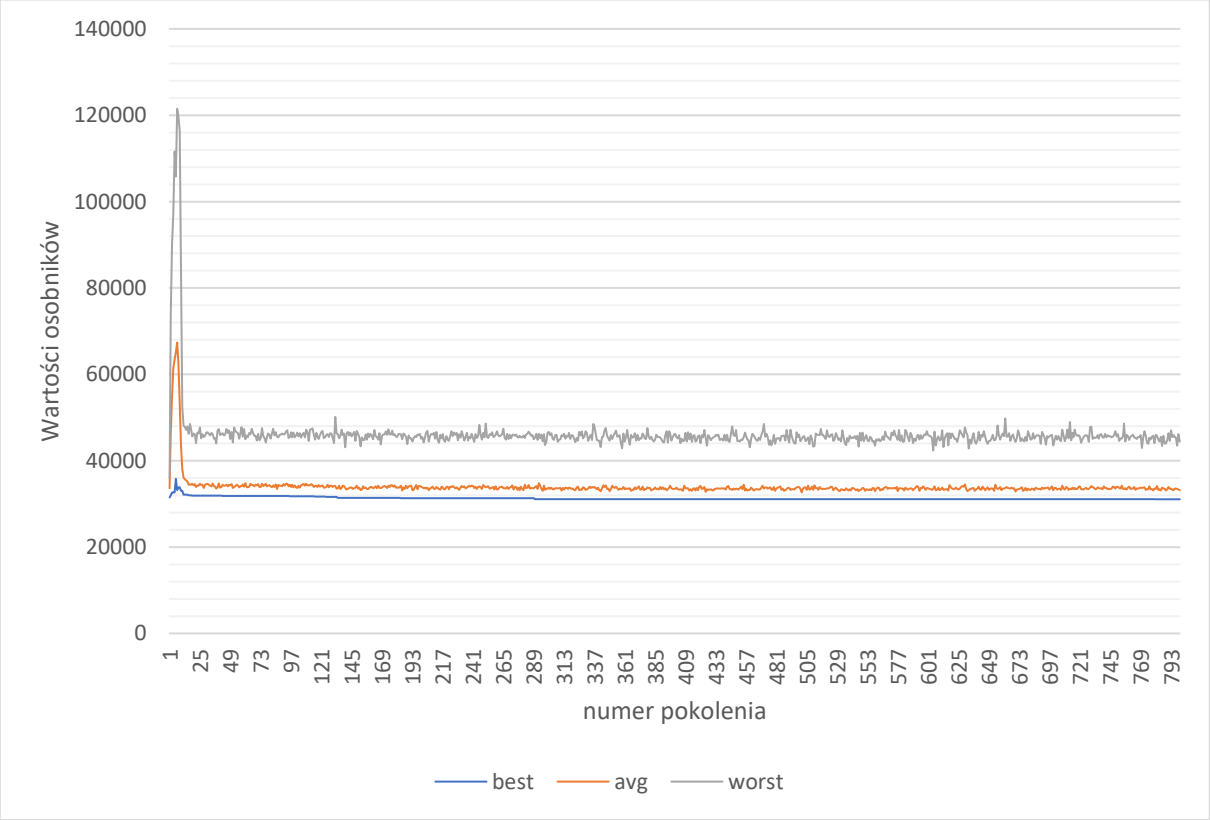
Przebieg badania: Załadowanie danych dla każdej z instancji, dziesięciokrotne uruchomienie algorytmu genetycznego z zadanymi wartościami parametrów z inicjalizacją zachłanną i zapis do pliku. Porównanie z pozostałymi wynikami z badania 8.

W przypadku inicjalizacji zachłannej wielkość populacji zależy od problemu i jest równa wielkości problemu – tworzę osobnika dla każdego miasta jako początkowego.

Wyniki:

Tabela 15 Porównanie wyników dla różnych algorytmów i metod inicjalizacji

Instancja	Optymalny wynik	Alg. Ewolucyjny z inicjalizacją zachłanną [10x]				Alg.Ewolucyjny z inicjalizacją losową[10x]			
		best	worst	avg	std	best	worst	avg	std
berlin52	7 542	8 134.90	8 182.19	8 153.82	18.92	8 642.59	10 442.34	9 324.32	511.36
kroA100	21 282	22 773.82	23 795.70	22 980.29	305.37	32 996.55	45 074.52	37 654.34	3 799.65
kroA150	26 524	30 532.10	31 482.02	30 870.51	364.64	48 073.98	54 880.96	52 044.15	2 304.93
kroA200	29 368	33 250.93	34 547.69	33 488.77	431.31	62 178.58	75 463.65	68 121.44	4 554.81
fl417	11 861	13 510.13	13 802.38	13 668.91	120.07	99 916.45	118 566.87	106 794.15	5 920.26
Instancja	Optymalny wynik	Alg. Losowy [640k]				Alg. Zachłanny[N]			
		best	worst	avg	std	best	worst	avg	std
berlin52	7 542	22 251.36	35 911.68	29 910.95	1 579.03	8 182.19	10 299.37	9 373.43	471.61
kroA100	21 282	132 485.38	209 217.12	171 080.51	8 231.59	24 698.50	28 694.68	27 045.21	812.11
kroA150	26 524	209 963.50	307 710.56	257 605.64	10 116.96	31 482.02	35 935.29	33 639.92	862.41
kroA200	29 368	282 639.66	396 992.73	340248.9486	11 781.39	34 547.69	42 045.19	37 468.91	1 405.40
fl417	11 861	431 127.53	558 991.31	496022.9349	13 571.91	13 802.38	16 722.67	15 724.26	493.4747



Rysunek 18 Przebieg algorytmu genetycznego z inicjalizacją zachłanną

Wnioski:

Algorytm ewolucyjny z inicjalizacją zachłanną daje dużo lepsze wyniki niż przy inicjalizacji losowej. Nie przy każdym uruchomieniu, ale przy 10-krotnym dla każdego wyniku był w stanie poprawić wynik uzyskany przez algorytm zachłanny. Przebieg algorytmu wygląda w tym przypadku jednak zupełnie inaczej, choć jest zróżnicowanie osobników to bardzo trudno jest znaleźć lepsze rozwiązania.

Podsumowanie

Algorytmy ewolucyjne są ciekawym sposobem szukania rozwiązań skomplikowanych problemów. Wymagają jednak długich badań, mają bardzo wiele możliwości m.in. sposobów implementacji operatorów i in. oraz bardzo wiele parametrów, których wartości w różny sposób wpływają na efektywność i skuteczność działania algorytmu. Znalezienie optymalnych wartości parametrów nie zawsze jest proste.

Algorytmy genetyczne nie zawsze są najlepszym rozwiązaniem, czasami algorytm zachłanny jest lepszy, jednak algorytm genetyczny potrafi poprawić wyniki znalezione przez algorytm zachłanny, mają więc w sobie potencjał i rozwiązują też dużo więcej innych problemów niż problem komiwojażera.