Adrian Frydmański 209865 Data oddania: **18 I 2016r.**Dawid Gracek 209929 Prowadzący: **dr inż. Tomasz Kapłon**

Zadanie projektowe 3

Projektowanie efektywnych algorytmów

**Temat: „Algorytm genetyczny dla problemu komiwojażera”**

# Zadanie

Należy zaimplementować algorytm genetyczny dla problemu komiwojażera oraz dokonać testów polegających na pomiarze czasu działania algorytmu w zależności od wielkości instancji oraz jakości dostarczanych rozwiązań (należy porównać rozwiązania dostarczone przez algorytm z najlepszymi znanymi rozwiązaniami dla przykładów testowych i z wynikami z poprzednich zadań).

# Informacje wstępne

Problem komiwojażera opisuje znajdywanie minimalnego cyklu Hamiltona w grafie pełnym. Innymi słowy: komiwojażer ma za zadanie obejść wszystkie miasta, mając do dyspozycji odległości między nimi. Musi to zrobić jak najkrótszą drogą.

Każdy wierzchołek grafu jest miastem, przez które musi przejechać komiwojażer. Grupę *n* miast reprezentuje zbiór *N = {1, 2, …, n}*. Drogi pomiędzy miastami reprezentowane są przez macierz   
*D = {dij, i є N, j є N, i ≠ j}*, gdzie *dij* jest odległością pomiędzy miastem *i* oraz *j*.

Są dwie odmiany problemu – symetryczna i asymetryczna. W symetrycznej droga z miasta a do b jest równa drodze z miasta b do a. W asymetrycznej drogi te mogą być różne. W macierzy odległości między miastami wygląda to następująco:

* dla problemu symetrycznego *dij = dji*,
* dla problemu asymetrycznego *dij ≠ dji*.

Algorytm genetyczny przeszukuje przestrzeń alternatywnych rozwiązań analogicznie do zjawiska ewolucji biologicznej – to właśnie z biologii jego twórca, John Henry Holland, czerpał inspirację. Ogólna grupa algorytmów, w której skład wchodzi genetyczny, nazywana jest grupą algorytmów ewolucyjnych.

Problem określa środowisko, w którym istnieje pewna populacja osobników, z których każdy ma przypisany pewien zbiór informacji – genotyp, będący podstawą do utworzenia fenotypu, który to jest zbiorem cech podlegających ocenie funkcji przystosowania modelującej środowisko. W skrócie: genotyp jest opisem proponowanego rozwiązania, a funkcja przystosowania ocenia, jak dobre jest owe rozwiązanie.

Działanie algorytmu przebiega następująco:

1. Losowanie populacji początkowej.
2. Selekcja populacji, reprodukcja najlepiej przystosowanych osobników.
3. Operacje ewolucyjne na genotypach: krzyżowania i mutacje.
4. „Narodziny” kolejnego pokolenia. W celu utrzymania stałej liczby osobników najlepsze są powielane, najsłabsze zaś usuwane. W przypadku nieznalezienia dostatecznie dobrego rozwiązania algorytm powraca do kroku 2. P przeciwnym razie rozwiązaniem jest genotyp najlepszego osobnika w populacji.

Aby algorytm genetyczny działał poprawnie, należy ustalić kilka zagadnień:

* Ustalenie genomu, jako reprezentanta wyniku,
* Ustalenie funkcji przystosowania,
* Ustalenie operatorów przeszukiwania.



Rysunek Schemat działania algorytmu

# Implementacja i Testowanie

Program został napisany w języku C++ za pomocą środowiska Visual Studio 2015. Program realizuje algorytm genetyczny dla problemu komiwojażera.

Algorytm dla każdego pliku wejściowego był wykonywany 10 razy, a następnie jego wyniki zostały uśrednione. Testy były przeprowadzane dla 4, 5, 10, 14, 17, 26, 29, 42, 48 i 76 miast (pliki załączone wraz z programem). Pliki wejściowe mogą być w dwóch postaciach:

* sposób dra Mierzwy: liczba miast, a następnie kolejne wartości z macierzy odległości między nimi;
* plik TSP ze współrzędnymi miast – liczba miast podana po „DIMENSION:” (lub „DIMENSION :”), współrzędne i numery wierzchołków podawane trójkami po „NODE\_COORD\_SECTION”.

Czas wykonywania się algorytmu był mierzony dzięki własnej klasie MyTimer. Zlicza ona takty zegara procesora i dzieli tę liczbę przez jego częstotliwość, a następnie mnoży przez 1000 aby wynik był podany w ms.

W zaimplementowanym algorytmie inicjacja populacji odbywa się przez losowanie osobników (permutacje miast).

Ocena przystosowania i selekcja chromosomów w populacji polega na wylosowaniu dwóch osobników poprzez imitację „koła fortuny”. Losowane osobniki mają odpowiednie wagi, które określa funkcja przystosowania – im osobnik silniejszy (rozwiązanie lepsze), tym większe szanse ma na wylosowanie, bo pole na „kole fortuny” jest większe. Dopuszcza się możliwość wylosowania dwóch tych samych osobników.



Krzyżowanie odbywa się poprzez połączenie dwóch chromosomów w miejscach, gdzie są takie same i wylosowaniu pozostałych genomów.

Nowopowstały osobnik umieszczany jest w populacji w miejscu osobnika najsłabszego, z najgorszą wartością funkcji przystosowania (najdłuższą drogą). Następuje powrót do oceny przystosowania.

# Wyniki

Wyniki pomiarów przedstawia poniższa tabela. Rozwiązania były liczone dla pięćdziesięciotysięcznej populacji z mutacjami na poziomie 2%.

Tabela 1. Porównanie wyników działania algorytmów

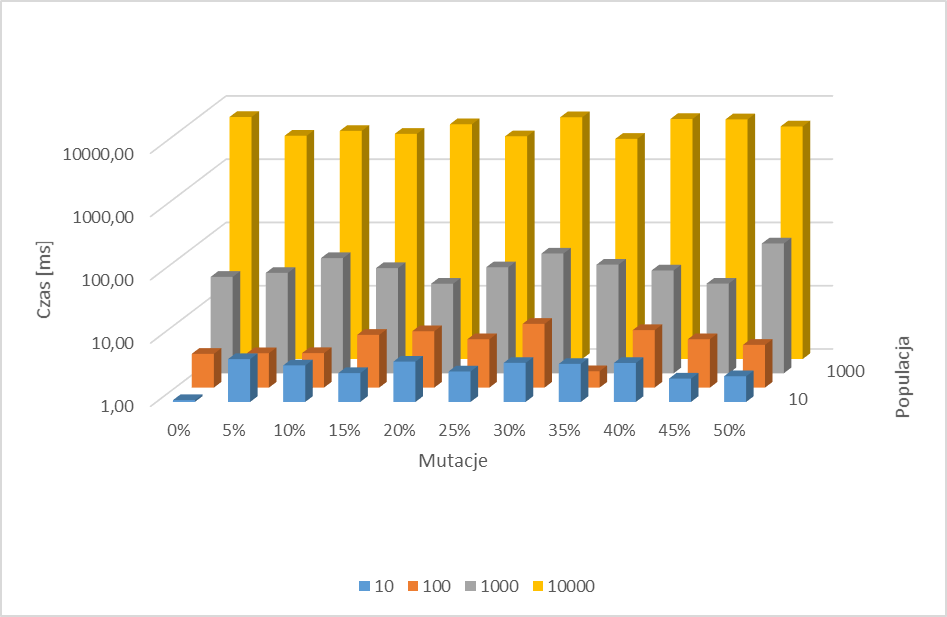
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Liczba miast | Przegląd zupełny (optimum) | Symulowane Wyżarzanie | | | | | | | | Tabu Search | | Algorytm Genetyczny | | Błąd AG/OPT | Polepszenie SW/AG | Polepszenie TS/AG |
| Drogi | | | | Czasy [ms] | | | | Drogi | Czasy [ms] | Drogi | Czasy [ms] |
| 0,9 | 0,95 | 0,995 | 0,999 | 0,9 | 0,95 | 0,995 | 0,999 |
| 4 | 200 | 200 | 200 | 200 | 200 | 0 | 1 | 17 | 90 | 300 | 2 | 200 | 39 | 0% | 0% | 50% |
| 5 | 16 | 22 | 21 | 17 | 16 | 1 | 2 | 28 | 144 | 16 | 5 | 16 | 41 | 0% | 0% | 0% |
| 10 | 212 | 331 | 340 | 281 | 270 | 4 | 9 | 97 | 486 | 212 | 19 | 257 | 74 | 21% | 5% | -18% |
| 14 | 33 | 49 | 46 | 45 | 40 | 7 | 17 | 179 | 825 | 39 | 41 | 36 | 126 | 10% | 10% | 7% |
| 17 | 2085 | 3824 | 3585 | 3303 | 3285 | 9 | 22 | 231 | 1055 | 2869 | 74 | 3103 | 56 | 49% | 6% | -8% |
| 26 | 937 | 2306 | 2195 | 1948 | 1904 | 19 | 40 | 408 | 2194 | 1140 | 270 | 1785 | 99 | 91% | 7% | -36% |
| 29 | 2020 | 5040 | 4977 | 4647 | 4354 | 24 | 50 | 512 | 2627 | 4236 | 320 | 4194 | 131 | 108% | 4% | 1% |
| 42 | 699 | 4159 | 4159 | 3869 | 3719 | 37 | 78 | 792 | 4007 | 699 | 1138 | 2231 | 109 | 219% | 67% | -69% |
| 48 | 10628 | 135279 | 127276 | 121034 | 116964 | 50 | 104 | 1063 | 5289 | 108310 | 1868 | 103633 | 187 | 875% | 13% | 5% |
| 76 | 108159 | 171510 | 179728 | 173609 | 179123 | 98 | 202 | 2070 | 10363 | 150780 | 7677 | 406662 | 680 | 276% | -56% | -63% |

Widać wydłużenie czasu względem poprzednich algorytmów dla małych instancji, ale skrócenie dla większych.

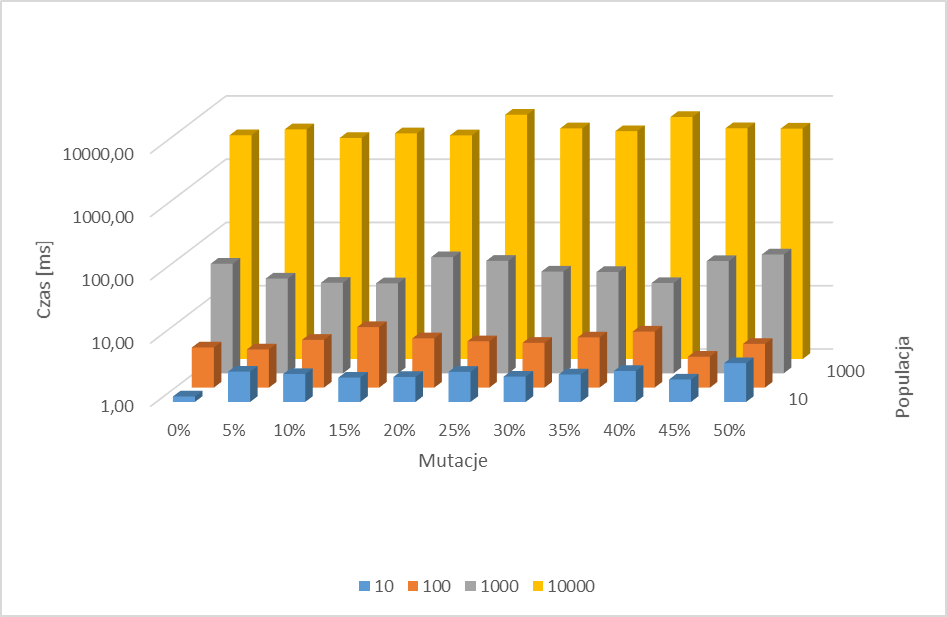
Poniższe tabele i wykresy przedstawiają czas wykonywania algorytmu dla różnych mutacji i krzyżowań dla przykładowego grafu.

Tabela 2. Porównanie czasów działania przy różnych wartościach krzyżowania, mutacji i populacji

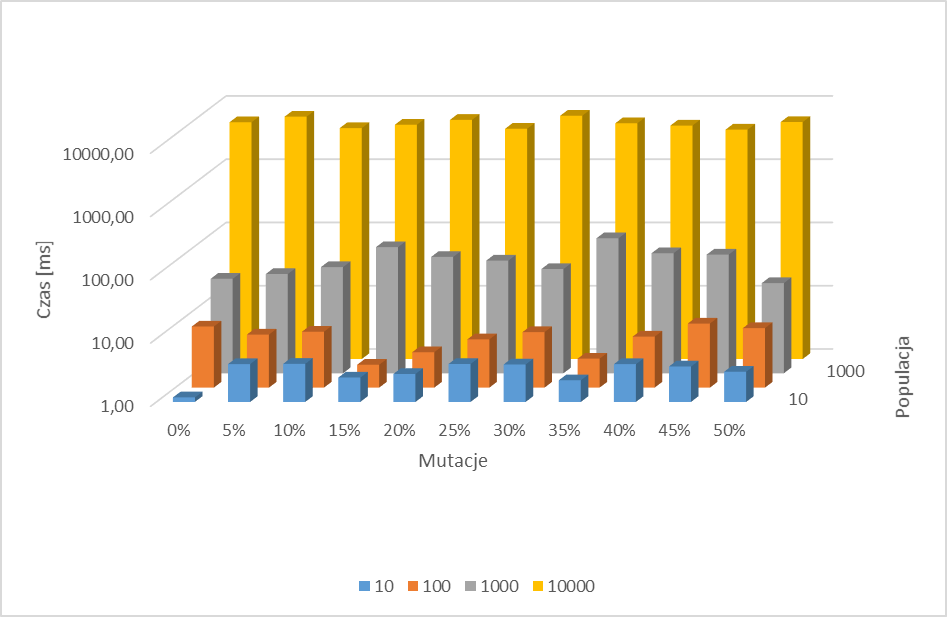
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Krzyżowanie | Mutacje | Populacja | | | |
| 10 | 100 | 1000 | 10000 |
| 20% | 0% | 1,07 | 3,42 | 33,50 | 6684,27 |
| 5% | 4,75 | 3,50 | 38,63 | 3373,91 |
| 10% | 3,78 | 3,51 | 66,12 | 4026,63 |
| 15% | 2,86 | 6,82 | 45,93 | 3621,38 |
| 20% | 4,33 | 7,78 | 26,10 | 5134,16 |
| 25% | 3,03 | 5,82 | 47,41 | 3310,41 |
| 30% | 4,14 | 10,17 | 78,20 | 6630,51 |
| 35% | 3,99 | 1,82 | 52,14 | 3015,34 |
| 40% | 4,13 | 8,12 | 42,23 | 6218,54 |
| 45% | 2,34 | 5,83 | 26,13 | 6129,94 |
| 50% | 2,55 | 4,74 | 112,58 | 4738,35 |
| 40% | 0% | 1,22 | 4,31 | 53,87 | 3437,24 |
| 5% | 3,01 | 4,01 | 31,43 | 4259,97 |
| 10% | 2,76 | 5,70 | 26,95 | 3118,13 |
| 15% | 2,42 | 9,04 | 26,42 | 3685,67 |
| 20% | 2,48 | 6,01 | 68,50 | 3426,62 |
| 25% | 2,99 | 5,40 | 59,91 | 7258,41 |
| 30% | 2,51 | 5,10 | 40,47 | 4416,42 |
| 35% | 2,72 | 6,22 | 39,94 | 3988,16 |
| 40% | 3,09 | 7,68 | 26,72 | 6675,26 |
| 45% | 2,26 | 3,08 | 59,51 | 4431,51 |
| 50% | 4,10 | 4,91 | 75,59 | 4377,92 |
| 60% | 0% | 1,18 | 9,24 | 31,21 | 5525,89 |
| 5% | 3,96 | 6,90 | 37,16 | 6759,41 |
| 10% | 4,00 | 7,66 | 47,70 | 4456,05 |
| 15% | 2,43 | 2,30 | 98,53 | 5034,60 |
| 20% | 2,77 | 3,63 | 69,10 | 6027,32 |
| 25% | 3,98 | 5,80 | 60,46 | 4356,08 |
| 30% | 3,91 | 7,54 | 44,47 | 7000,99 |
| 35% | 2,20 | 2,87 | 136,27 | 5317,29 |
| 40% | 3,96 | 6,41 | 79,04 | 4890,89 |
| 45% | 3,63 | 10,29 | 75,16 | 4215,53 |
| 50% | 3,00 | 8,74 | 26,66 | 5558,73 |
| 80 | 0% | 1,11 | 3,15 | 27,55 | 5295,01 |
| 5% | 4,18 | 8,25 | 28,81 | 5213,33 |
| 10% | 3,06 | 9,56 | 89,51 | 6999,53 |
| 15% | 4,26 | 7,22 | 49,38 | 4601,75 |
| 20% | 3,87 | 6,76 | 47,70 | 4861,29 |
| 25% | 2,31 | 8,44 | 51,69 | 7376,59 |
| 30% | 3,71 | 13,65 | 74,30 | 8543,01 |
| 35% | 2,08 | 7,86 | 84,54 | 5066,51 |
| 40% | 2,49 | 9,61 | 53,01 | 5921,86 |
| 45% | 3,00 | 5,56 | 84,66 | 6677,09 |
| 50% | 3,10 | 10,98 | 58,79 | 6790,47 |



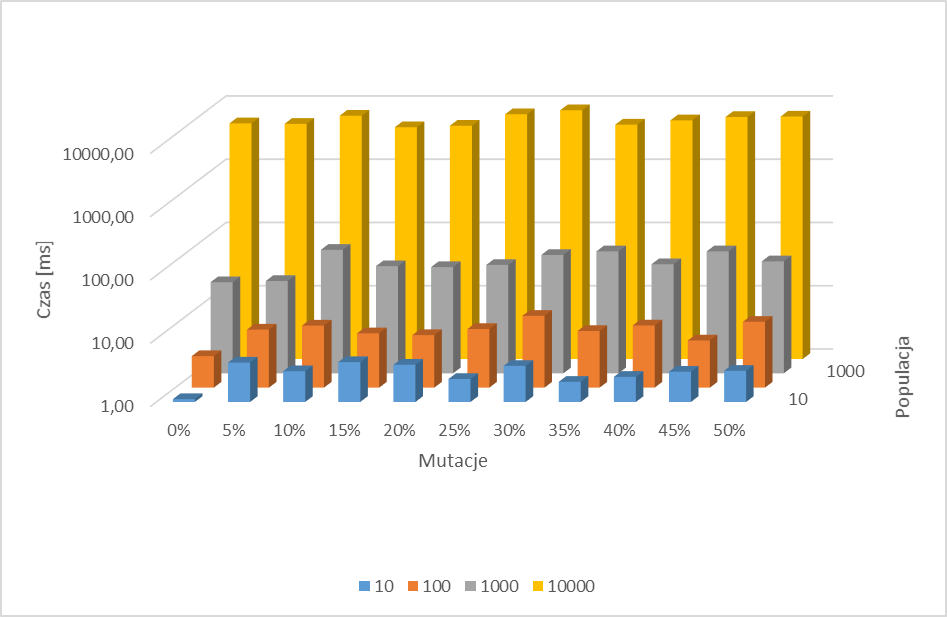
Rysunek Czas wykonywania w zależności od mutacji i populacji - dla 20% krzyżowań



Rysunek Czas wykonywania w zależności od mutacji i populacji - dla 40% krzyżowań

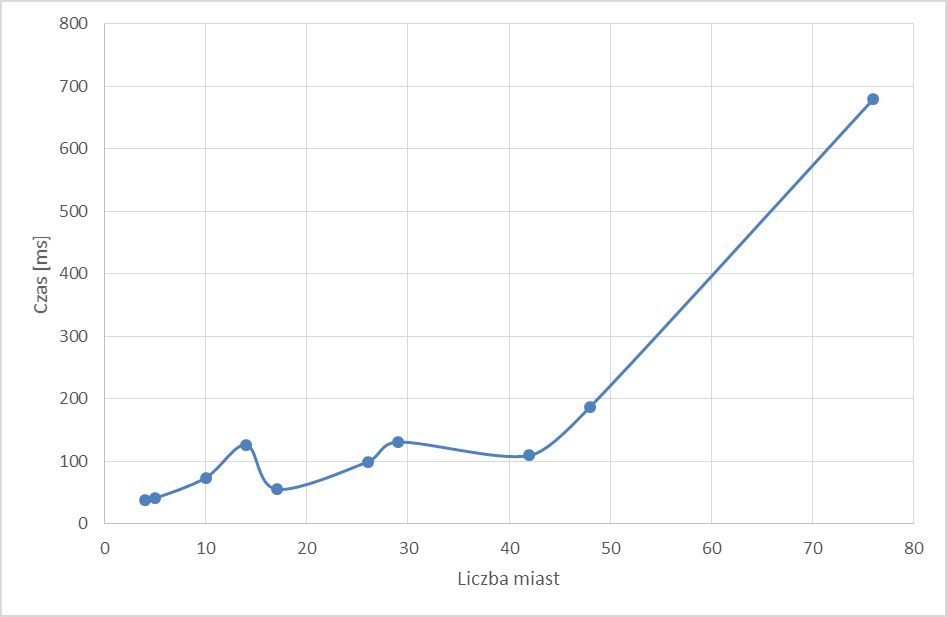


Rysunek Czas wykonywania w zależności od mutacji i populacji - dla 60% krzyżowań



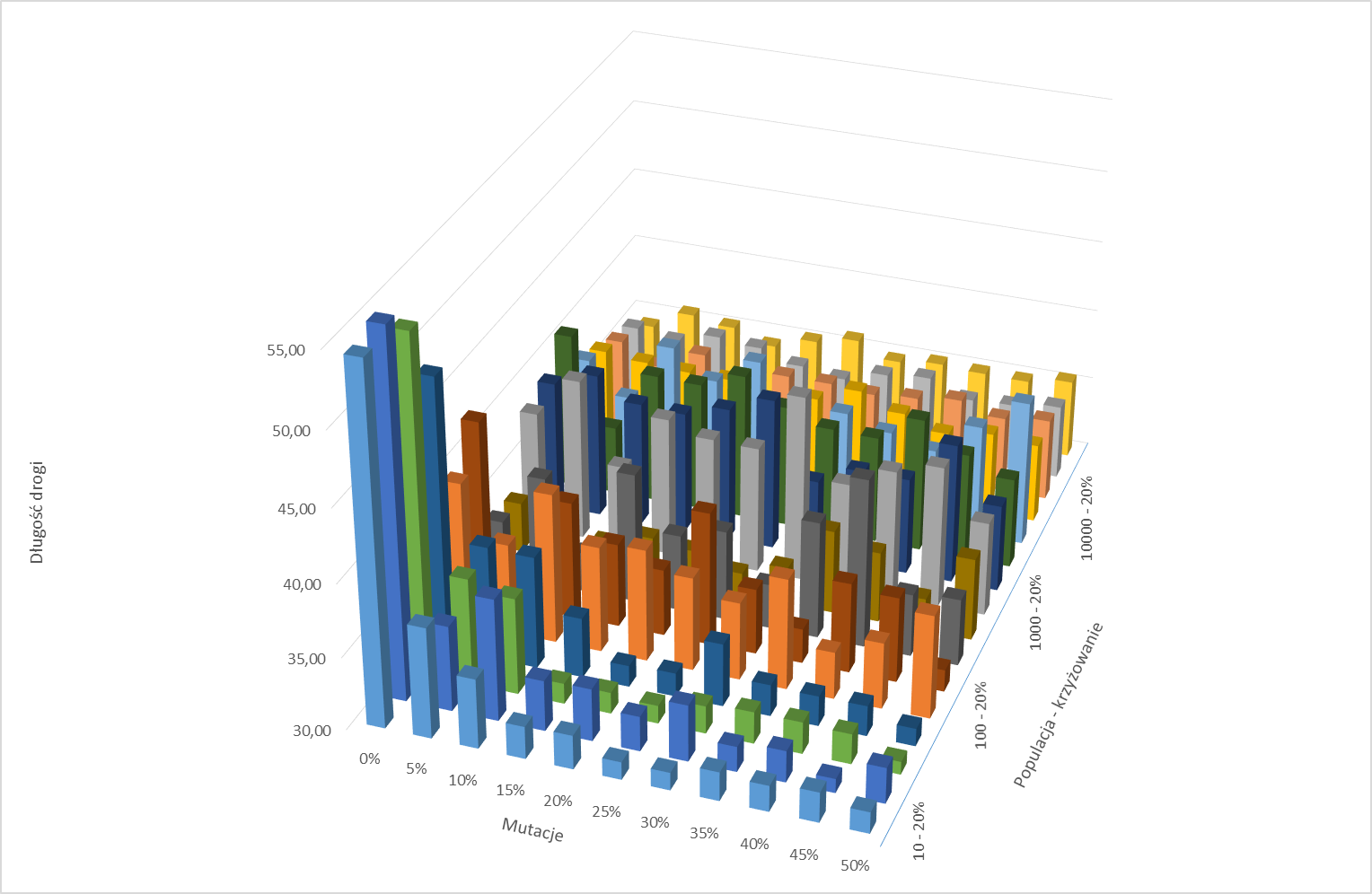
Rysunek Czas wykonywania w zależności od mutacji i populacji - dla 80% krzyżowań

Jak w poprzednich algorytmach widać ogólny wzrost czasu działania wraz ze wzrostem liczby miast.



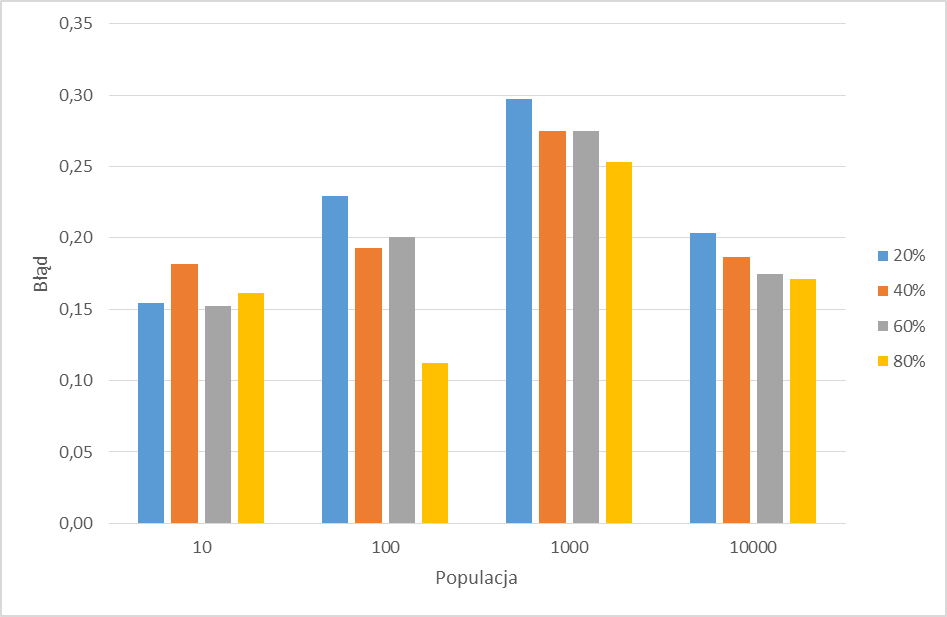
Rysunek Czas działania w zależności od liczby miast

Rozmiar populacji sprawia, że otrzymane wyniki są lepsze. Znów widać, że stopień mutacji nie ma znaczącego wpływu na wyznaczoną drogę.



Rysunek Wyznaczona droga w zależności od rozmiaru populacji, ilości krzyżowań i mutacji

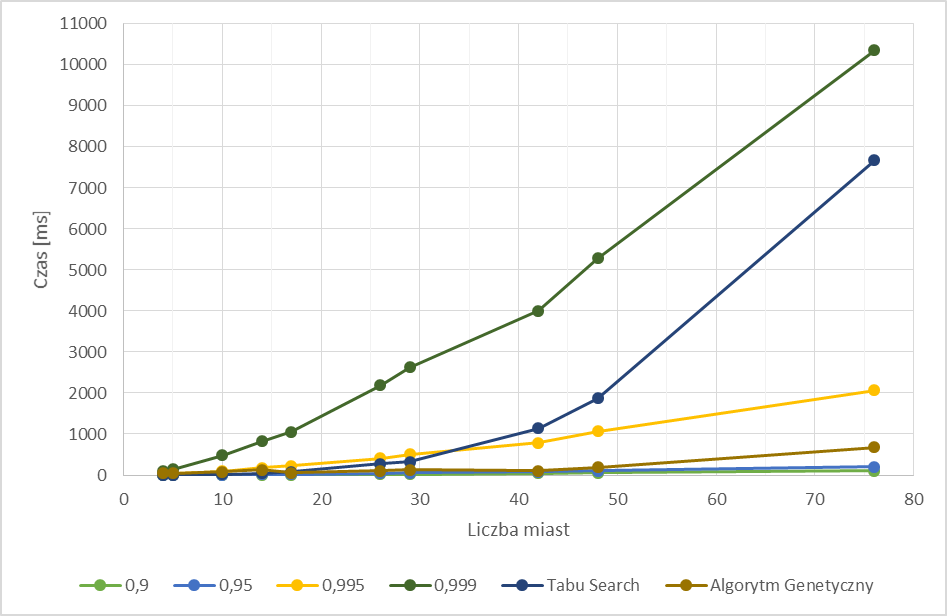
Mało mutacji i krzyżowania sprawia, że populacja rozwiązań przypomina populację misiów panda, których cechą charakterystyczną jest niewielka chęć do rozmnażania się.



Rysunek Średni błąd dla różnych populacji i ilości krzyżowań

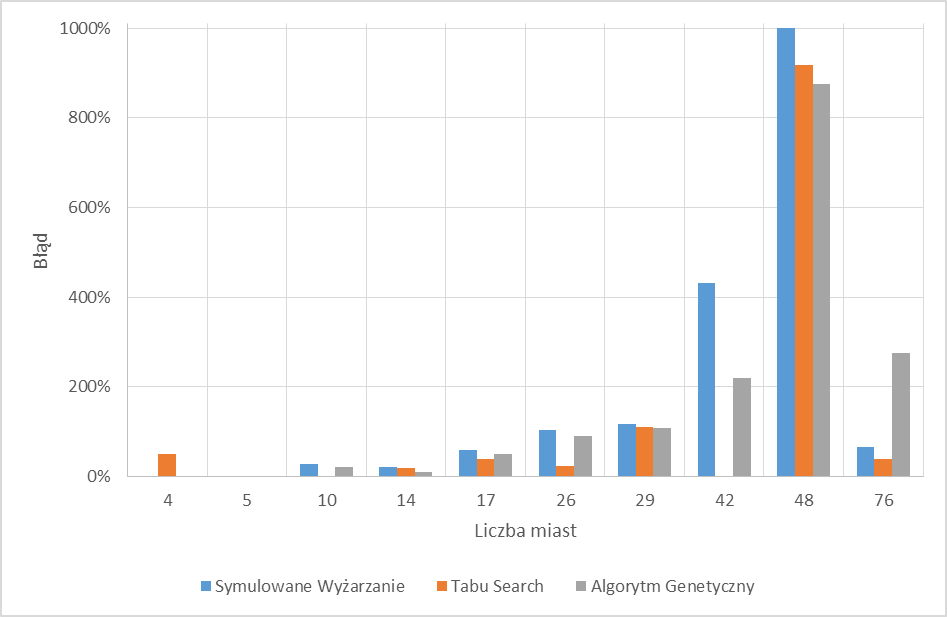
Jakość rozwiązania rośnie wraz ze wzrostem populacji i ilości krzyżowań.

Czas działania algorytmu genetycznego na tle innych rośnie bardzo powoli wraz z liczbą miast.



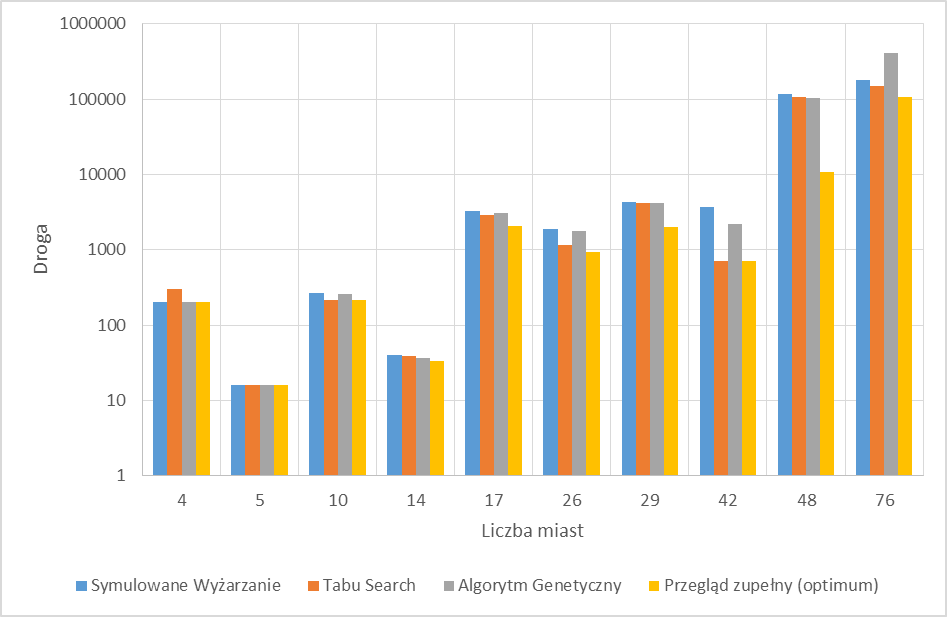
Rysunek Porównanie czasu działania algorytmów (liczby – współczynniki wyżarzania w algorytmie symulowanego wyżarzania)

Błąd względem optimum utrzymuje się na podobnym poziomie.



Rysunek Porównanie błędów dla każdego z algorytmów

W porównaniu wyznaczonych dróg widać, że przy dobranej do testów populacji 50000 wyniki są podobne do tych otrzymanych wcześniej.



Rysunek Porównanie dróg dla każdego z algorytmów

# Podsumowanie i wnioski

Przygotowanie projektu pozwoliło zapoznać się z tematyką algorytmiki metaheurystycznej w postaci innej niż w przypadku symulowanego wyżarzania, czy tabu search.

Z otrzymanych danych widać, że wszystkie trzy reprezentują podobną jakość wyznaczania optymalnej drogi w problemie komiwojażera. Algorytm genetyczny daje wyniki zbliżone do tabu search.

Dla dużych instancji algorytm pozwala uzyskać w miarę dobry wynik w krótszym czasie, niż pozostałe algorytmy.

# Bibliografia

* Problem komiwojażera, http://problem-komiwojazera.cba.pl
* Algorytm genetyczny, https://pl.wikipedia.org/wiki/Algorytm\_genetyczny
* Algorytmy genetyczne, http://www.zio.iiar.pwr.wroc.pl/pea/w9\_ga\_tsp.pdf
* Genetic Algorithm Solution of the TSP Avoiding Special Crossover and Mutation, https://www.ceng.metu.edu.tr/~ucoluk/research/publications/tspnew.pdf
* Crossover Technique: PMX, http://www.rubicite.com/Tutorials/GeneticAlgorithms/CrossoverOperators/PMXCrossoverOperator.aspx
* Genetic Algorithms and the Traveling Salesman Problem, http://www.codeproject.com/Articles/1403/Genetic-Algorithms-and-the-Traveling-Salesman-Prob
* A High-Performance Genetic Algorithm: Using Traveling Salesman Problem as a Case, http://www.hindawi.com/journals/tswj/2014/178621/
* A Fast TSP Solver Using GA on JAVA, http://www.gcd.org/sengoku/docs/arob98.pdf
* Analysis of the impact of parameters values on the Genetic Algorithm for TSP, http://ijcsi.org/papers/IJCSI-10-1-3-158-164.pdf
* The Traveling Salesman Problem – Genetic Algorithm in C++ and CUDA, http://techtorials.me/coding-projects/the-traveling-salesman-problem-genetic-algorithm-in-c-and-cuda/
* Partial-mapped Crossover - PMX - Genetic Algorithms, https://www.youtube.com/watch?v=c2ft8AG8JKE