Projektowanie Efektownych Algorytmów
Projekt
23/01/2024

259113 Hubert Bełkot

(6)Genetic algorithm

Spis treści	strona

1. Sformułowanie Zadania

Zadanie polega na opracowaniu i zaimplementowaniu algorytmu algorytmu genetycznego, rozwiązującego problem komiwojażera(eng. tsp- travelling salesman problem). Problem polega na znalezieniu w grafie minimalnego cyklu, w którym każdy wierzchołek jest odwiedzany dokładnie raz (cykl Hamiltona). Należy zbadać złożoność czasową oraz zależność zużycia pamięci w zależności od wielkości instancji.

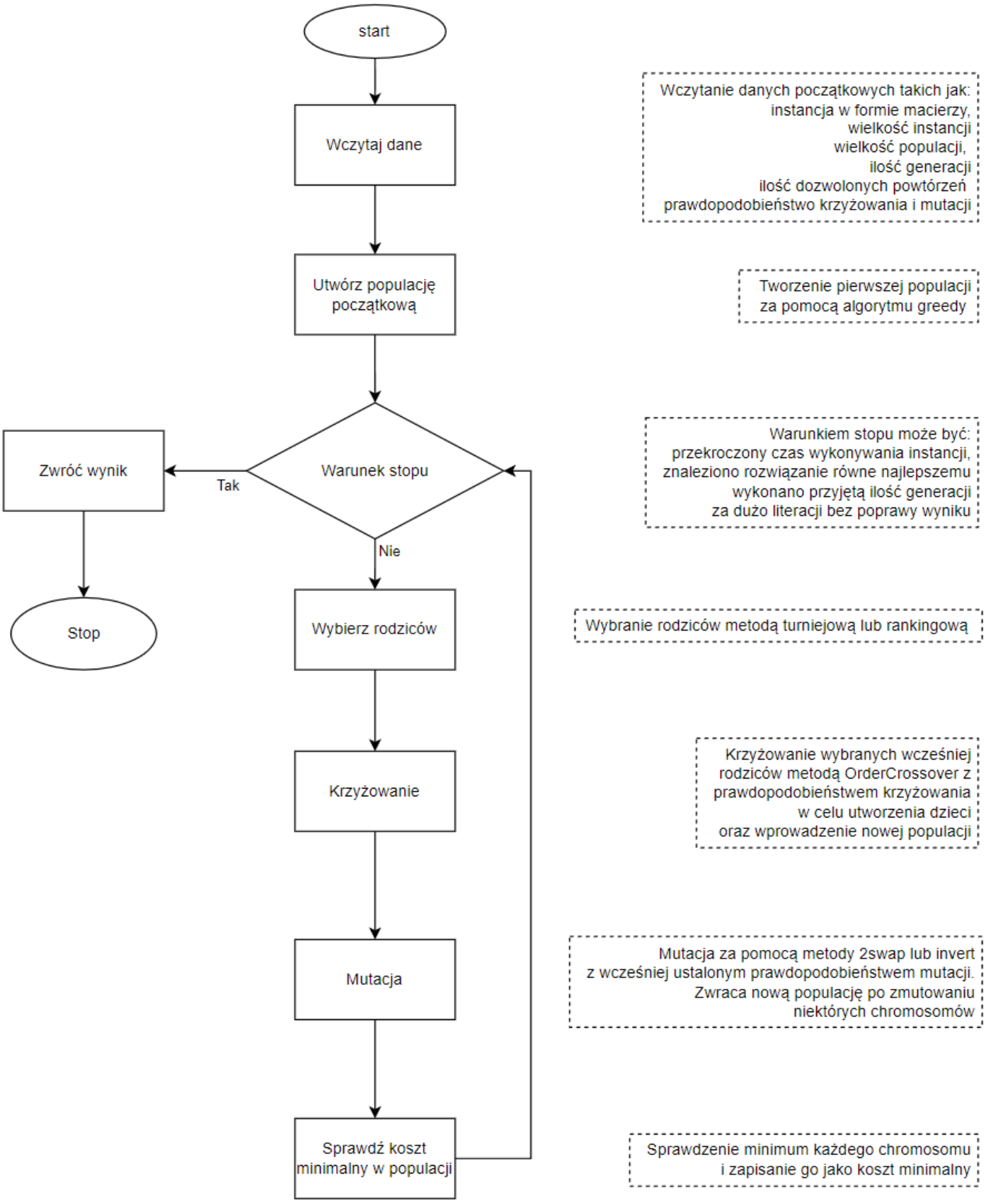
2. Metoda

Algorytm genetyczny jest jedną z metaheurestyk zainspirowana biologiczną ewolucją. Obecnie zaliczany jest do algorytmów ewolucyjnych. Algorytm genetycznych próbuje otrzymać jak najlepsze rozwiązanie za pomocą wybierania jak najlepszych cech rozwiązań z określonej puli. Zakładamy, że z każdym pokoleniem rozwiązania będą coraz lepsze. W algorytmie dla przypadku rozwiązania problemu komiwojażera, pokoleniem nazywamy populacje chromosomów w danej iteracji. Pojedynczy chromosom jest pojedynczym cyklem Hamiltona, natomiast populacja zbiorem chromosomów o zadanej liczebności.

Na algorytm genetyczny składają się operację:

- Tworzenie populacji początkowej – random
  polega na losowym wybraniu losowej pierwszej ścieżki.
- Selekcja – Metoda turniejowa / rankingowa. Metoda turniejowa polega na wylosowaniu pary osobników z której lepsza zostaje wybrana do krzyżowania. Metoda rankingowa polega na wybraniu najlepszych osobników.
- Krzyżowanie – budowanie potomstwa poprzez kopiowanie części rozwiązania z jednego rodzica, a następnie uzupełnianie brakujących liczb poprzez wstawiane w takiej kolejności, w jakiej występowały w drugim rodzicu.
- Mutacja – operacja 2-swap/invert. Mutacja ma za zadanie dokonać losowej zmiany na którymś z osobników. Do tego zostaje wykorzystana metoda 2-swap, czyli wylosowanie dwóch wierzchołków i zamiana ich, oraz metoda invert, która wybiera dwa wierzchołki i odwraca kolejnością dany odcinek ścieżki.
- Sukcesja (stworzenie nowej populacji) –metoda rankingowa. Wybranie najlepszych osobników z otrzymanych chromosomów i zastąpienie nimi nowej populacji
- Warunek stopu – liczba generacji / liczba iteracji bez poprawy wyniku / osiągnięcie pewnego błędu / przekroczenie czasu.

3. Algorytm



Rysunek 1: Schemat działania algorytmu genetycznego

4. Dane testowe

Do sprawdzenia poprawności algorytmu posłużyły następujące zestawy danych:

- tsp_6_2.txt
- tsp_10.txt
- tsp_15.txt
- gr21.tsp.txt

Do wykonania pomiarów wykorzystano:

- tsp_6_2.txt
- tsp_10.txt
- tsp_15.txt
- gr21.tsp.txt
- gr24.tsp.txt
- ftv38.atsp
- ftv64.atsp
- ftv170.atsp
- gr229.txt
- rbg323.atsp
- pcb442.tsp.txt
- rbg443.atsp
- tsp666.tsp.txt

Ze źródeł:
- http://jaroslaw.mierzwa.staff.iiar.pwr.wroc.pl/pea-stud/tsp/
- Teaching (uni-heidelberg.de)

5. Procedura badawcza

Należało zbadać zależność czasu rozwiązania problemu od wielkości instancji, czyli złożoność czasową oraz zużycie pamięci od wielkości instancji(złożoność pamięciową). Na początku procedury badawczej uruchamialiśmy plik ini (format_pliku: nazwa_instancji liczba_wykonań rozwiązanie optymalne [ścieżka_optymalna] nazwa_pliku.csv).

Treść pliku tsp.ini:
tsp_6_2.txt 5 80 [0 5 1 2 3 4 0]
tsp_10.txt 5 212 [0 3 4 2 8 7 6 9 1 5 0]
tsp_15.txt 5 291 [0 10 3 5 7 9 13 11 2 6 4 8 14 1 12 0]
gr21.tsp.txt 5 2707 [0 6 7 5 15 4 8 2 1 20 14 13 12 17 9 16 18 19 10 3 11 0]
gr24.tsp.txt 5 1272 [?]
ftv38.atsp 5 1530 [?]
ftv64.atsp 5 1839 [?]
ftv170.atsp 5 2755 [?]
gr229.txt 5 134602 [?]
rbg323.atsp 5 1326 [?]
pcb442.tsp.txt 5 50778 [?]
rbg443.atsp 5 2720 [?]tsp666.tsp.txt 5 194358 [?]
tsp666.tsp.txt 5 194358 [?]

output.csv

Przy próbie zbadania instancji pr1002, nie zdołałem otrzymać kosztu poniżej błędu procentowego 150% w czasie poniżej 10 minut, dlatego zrezygnowałem z badania tej instancji.

Każda instancja rozwiązywana była zgodnie z przyjętą liczbą jej wykonań, dla przykładu gr21.tsp.txt została wykonana 5 razy. Do pliku wyjściowego był zapisywany: jako nagłówek, dane odczytywane z pliku .init dla konkretnej instancji, pod spodem czas wykonania [ms], znalezione optymalne rozwiązanie oraz ścieżka. Poniżej fragment pliku wyjściowego zapisanego w formacie .csv dla gr21.tsp.tx:

gr21.tsp.txt	5	2707	[0 6 7 5 15 4 8 2 1 20 14 13 12 17 9 16 18 19 10 3 11 0]
144,2261	2707	[0 6 7 5 15 4 8 2 1 20 14 13 12 17 9 16 18 19 10 3 11 0]	
72,8875	2707	[0 11 3 10 19 18 16 9 17 12 13 14 20 1 2 8 4 15 5 7 6 0]	
155,9816	2707	[0 6 7 5 15 4 8 2 1 20 14 13 12 17 9 16 18 19 10 3 11 0]	
160,4757	2707	[0 6 7 5 15 4 8 2 1 20 14 13 12 17 9 16 18 19 10 3 11 0]	
137,0401	2707	[0 6 7 5 15 4 8 2 1 20 14 13 12 17 9 16 18 19 10 3 11 0]	

Tabela 1: wyniki dla instancji wielkości 21wykonane przy badaniu poprawności algorytmu

- Obliczanie błędu dla instancji:

$$\delta = \frac{|x - x_0|}{x} \cdot 100\%$$

Gdzie *x* jest optymalnym kosztem, a x_0 średnim kosztem zmierzonej instancji.

6. Wyniki

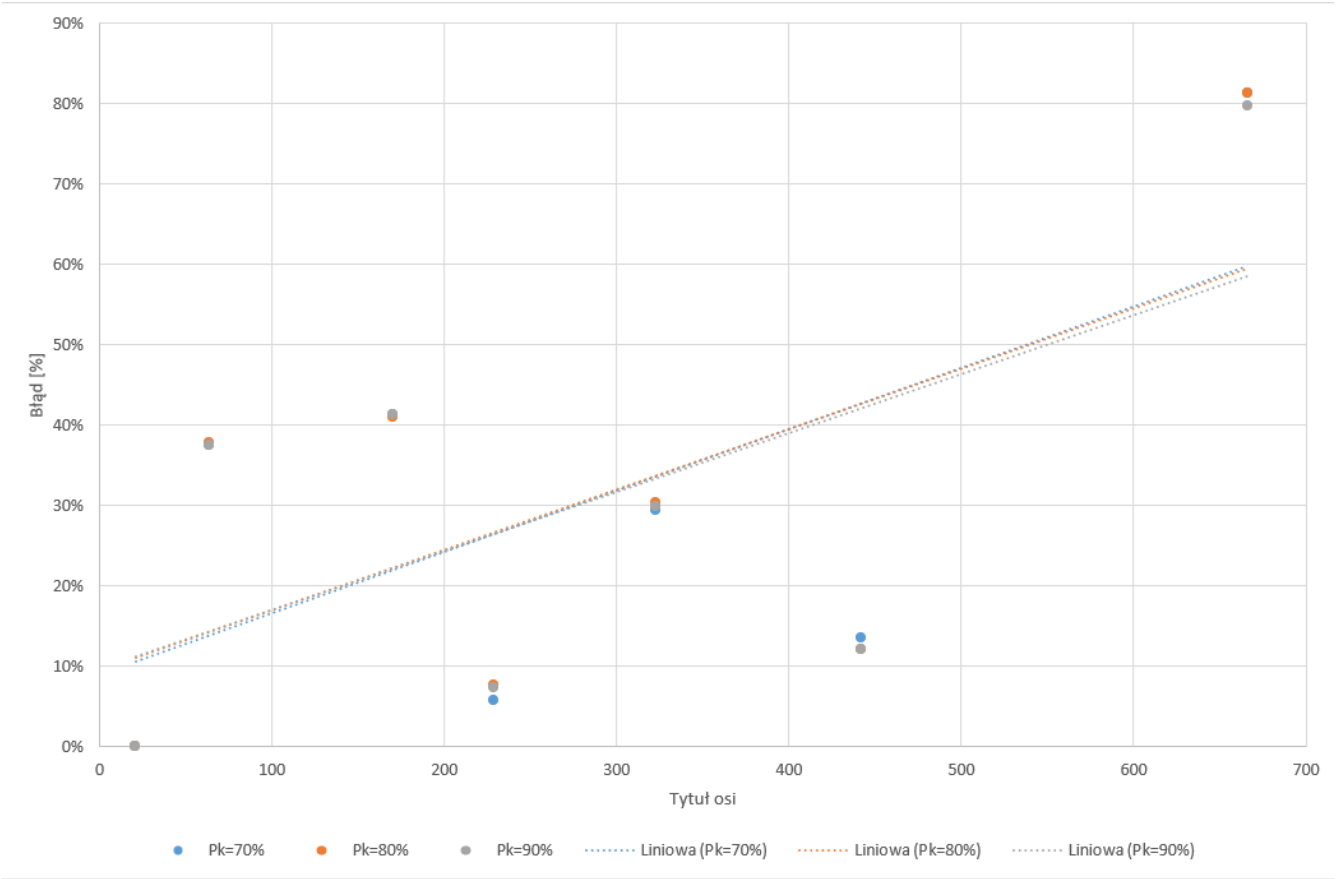
Parametry dla badań początkowych algorytmu:

- Prawdopodobieństwo krzyżowania = 90%
- Prawdopodobieństwo mutacji = 5%
- Wielkość populacji = 1000
- Liczba generacji = 1000
- Dozwolonych powtórzeń = 150
- Populacja początkowa – algorytm Greedy
- Metoda mutacji - Invert

- Badanie wpływu wielkości prawdopodobieństwa krzyżowania

		Pk=70%			Pk=80%			Pk=90%		
instancja	koszt optymalny	średni koszt	błąd[%]	czas [ms]	średni koszt	błąd[%]	czas [ms]	średnie	błąd[%]	czas [ms]
21	2707	2707	0%	140	2707	0%	135	2707	0%	265
64	1839	2530	38%	406	2534	38%	422	2531	38%	770
170	2755	3894	41%	2425	3887	41%	2148	3894	41%	2674
229	134602	142394	6%	6836	144928	8%	7611	144484	7%	8492
323	1326	1716	29%	10014	1729	30%	9849	1723	30%	11046
442	50778	57690	14%	22843	56920	12%	24560	56933	12%	26635
666	194358	352409	81%	57013	352543	81%	58440	349501	80%	61987

Tabela 2: Wyniki badania dla zmiany wielkości prawdopodobieństwa krzyżowania



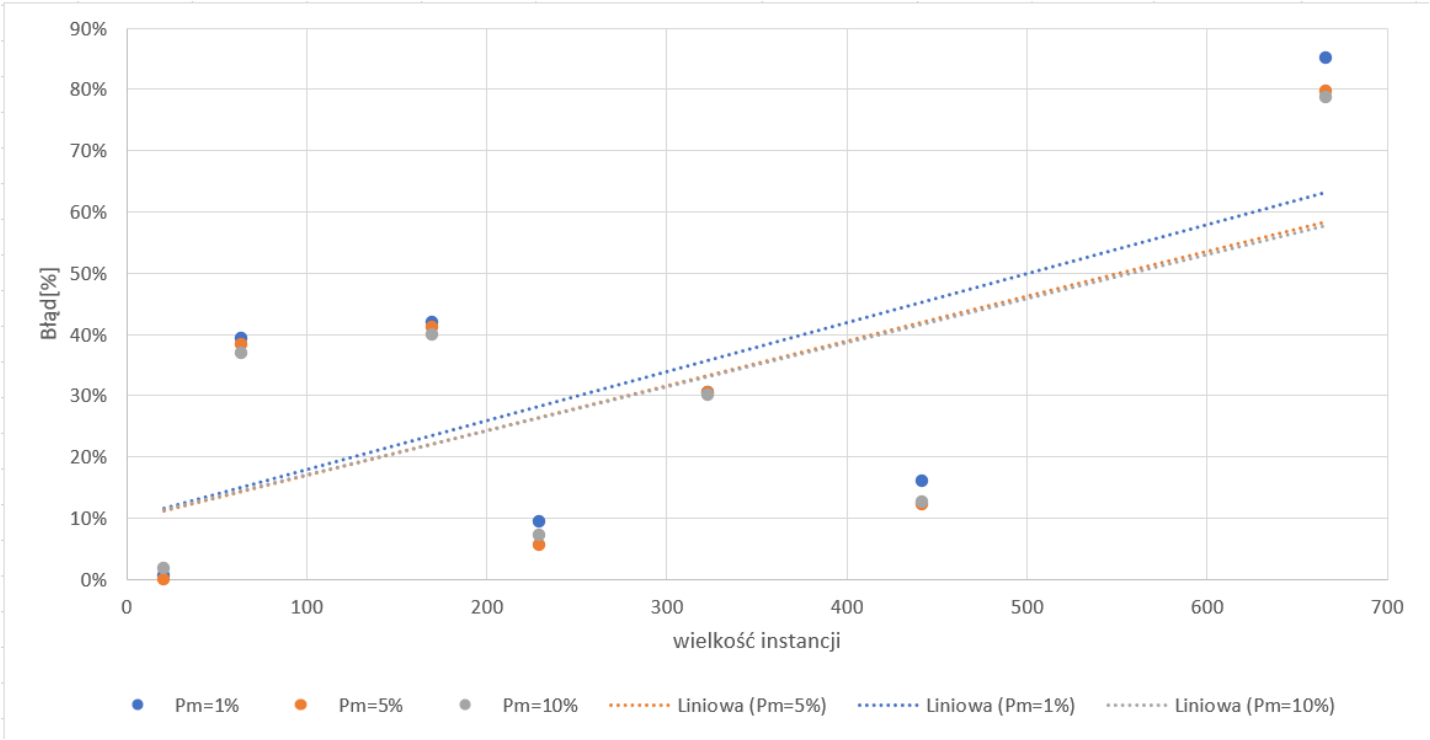
Rysunek 2: Wykres błędu procentowego w zależności od wielkości prawdopodobieństwa krzyżowania

Jak widać na powyższym rysunku oraz *tabeli 2,* średni błąd jest najmniejszy dla prawdopodobieństwa krzyżowania równego 90%, z tego względu je wybieramy do następnych testów.

- Badanie wpływu zmiany wielkości prawdopodobieństwa mutacji

		Pm=1%			Pm=5%			Pm=10%		
instancja	koszt optymalny	średni koszt	błąd[%]	czas [ms]	średni koszt	błąd[%]	czas [ms]	średni koszt	błąd[%]	czas [ms]
21	2707	2725	1%	193	2707	0%	125,42	2753	2%	669
64	1839	2562	39%	661	2545	38%	1015,75	2516	37%	1087
170	2755	3909	42%	1688	3887	41%	3616,63	3856	40%	4259
229	134602	147190	9%	7719	142218	6%	8573,55	144244	7%	8703
323	1326	1731	31%	9924	1731	31%	11780,03	1726	30%	13133
442	50778	58891	16%	25299	57021	12%	26877,90	57170	13%	26782
666	194358	359926	85%	62389	349343	80%	63491,35	347193	79%	63357

Tabela 3: Wyniki badania zmiany wielkości prawdopodobieństwa mutacji



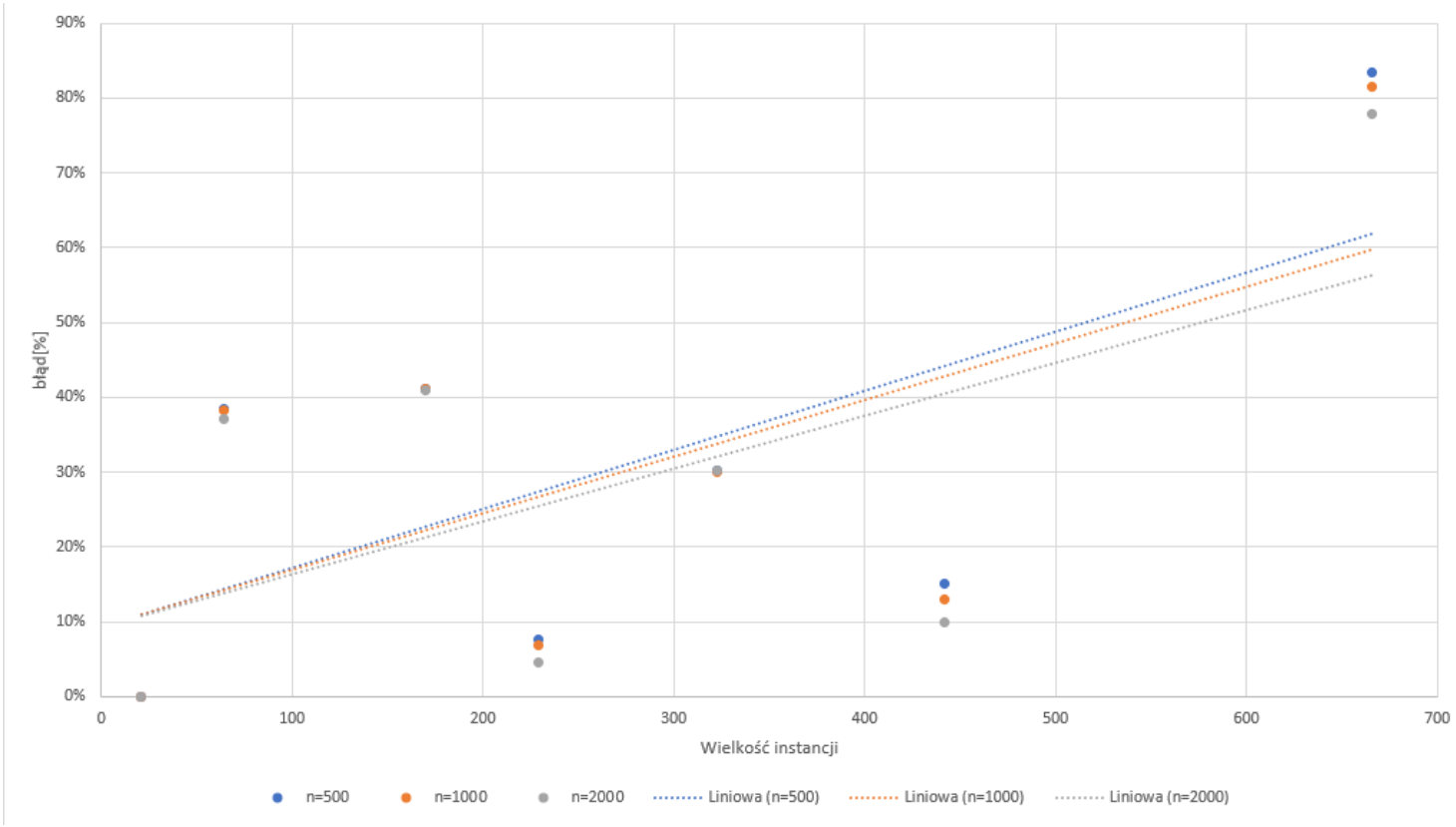
Rysunek 3: Wykres błędu procentowego w zależności od wielkości prawdopodobieństwa mutacji.

Patrząc na *tabelę 3* musimy wybrać prawdopodobieństwo mutacji równe 5% ze względu na warunek zadania, który mówi o zerowej wartości błędu dla instancji do wielkości 21.

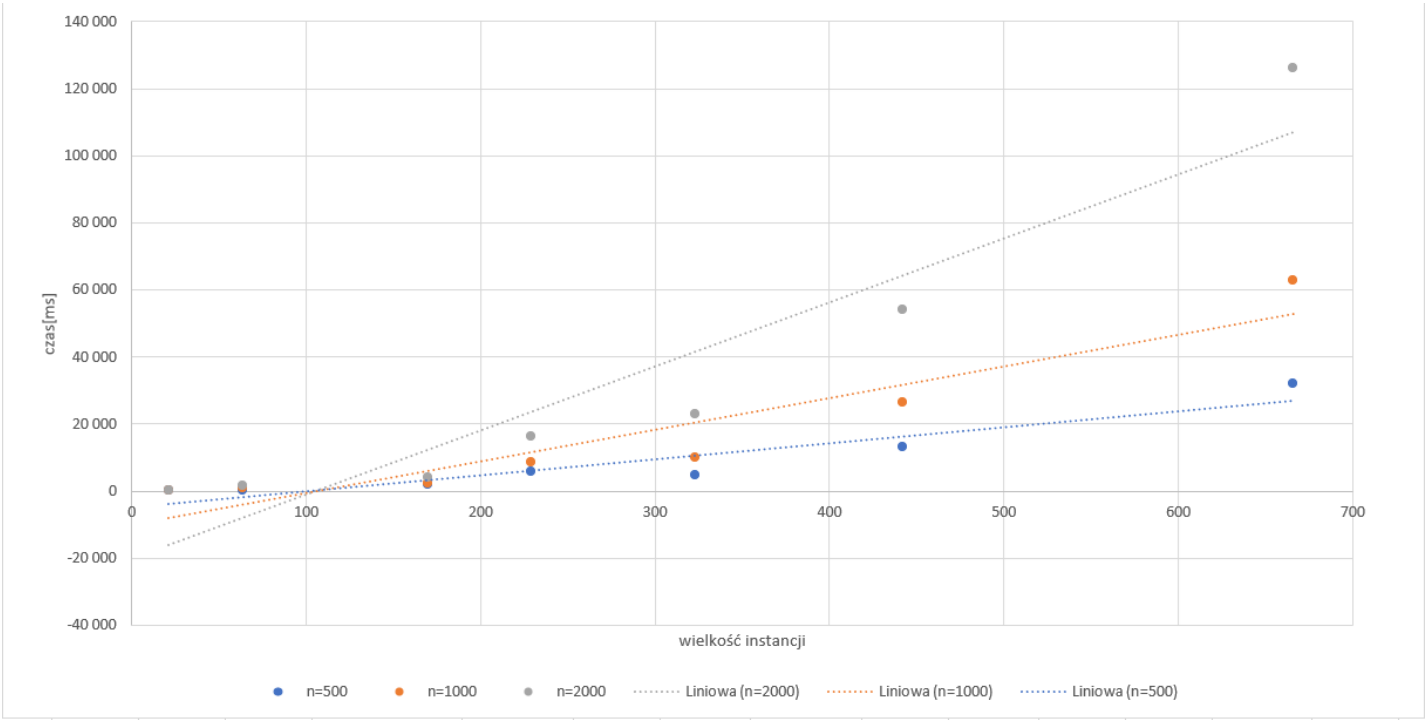
- Badanie wpływu wielkości populacji – n,

		n=500			n= 1000			n =2000		
instancja	koszt optymalny	średni koszt	błąd[%]	czas [ms]	średni koszt	błąd[%]	czas [ms]	średni kos	błąd[%]	czas [ms]
21	2707	2707	0%	114	2707	0%	176	2707	0%	292
64	1839	2547	39%	220	2541	38%	715	2522	37%	1632
170	2755	3887	41%	2011	3887	41%	2402	3883	41%	4102
229	134602	144957	8%	5616	143889	7%	8558	140711	5%	16334
323	1326	1727	30%	4608	1724	30%	9995	1727	30%	22800
442	50778	58418	15%	13214	57413	13%	26536	55850	10%	54215
666	194358	356488	83%	32020	352920	82%	62973	345863	78%	126196

Tabela 4: Wyniki badania zmiany wielkości populacji



Wykres 4: wykres zależności błędu procentowego od wielkości populacji



Wykres 5: Wykres zależności czasu wykonywania algorytmu od wielkości populacji.

Analizując wyniki pokazane na *wykres 4* oraz *tabela 4,* widzimy, że najmniejsza wartość błędu procentowego jest po stronie populacji równej 2000. Niestety ze względu bardzo dużą różnicę czasów wykonywania się algorytmu, pokazanych na *wykresie 5*, w następnych badaniach postanowiłem przyjąć wielkość populacji równą 500, której wyniki czasowe są bardzo korzystne a błąd procentowy nie ulega bardzo dużej zmianie.

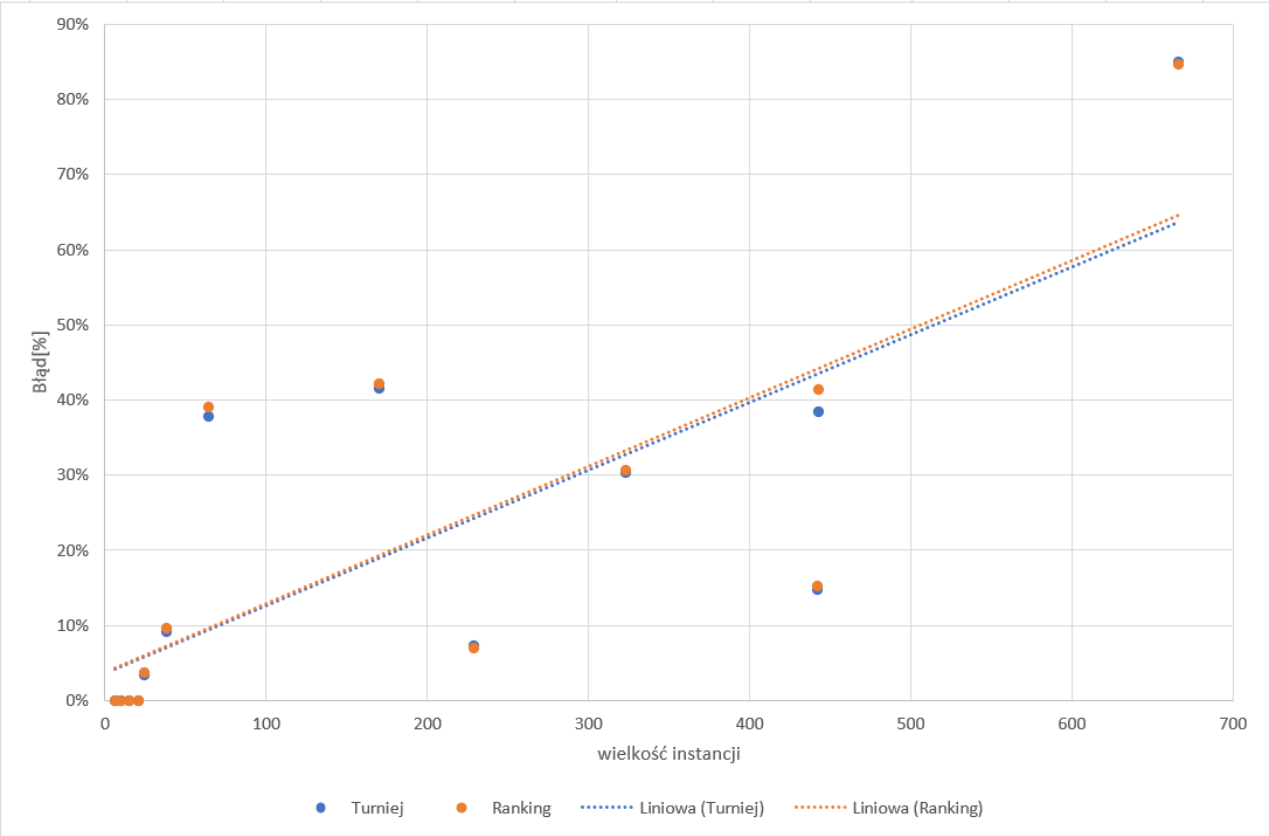
- Badanie wpływu metody selekcji na czas i jakość wykonywania algorytmu.

Parametry dla algorytmu zmieniono na :

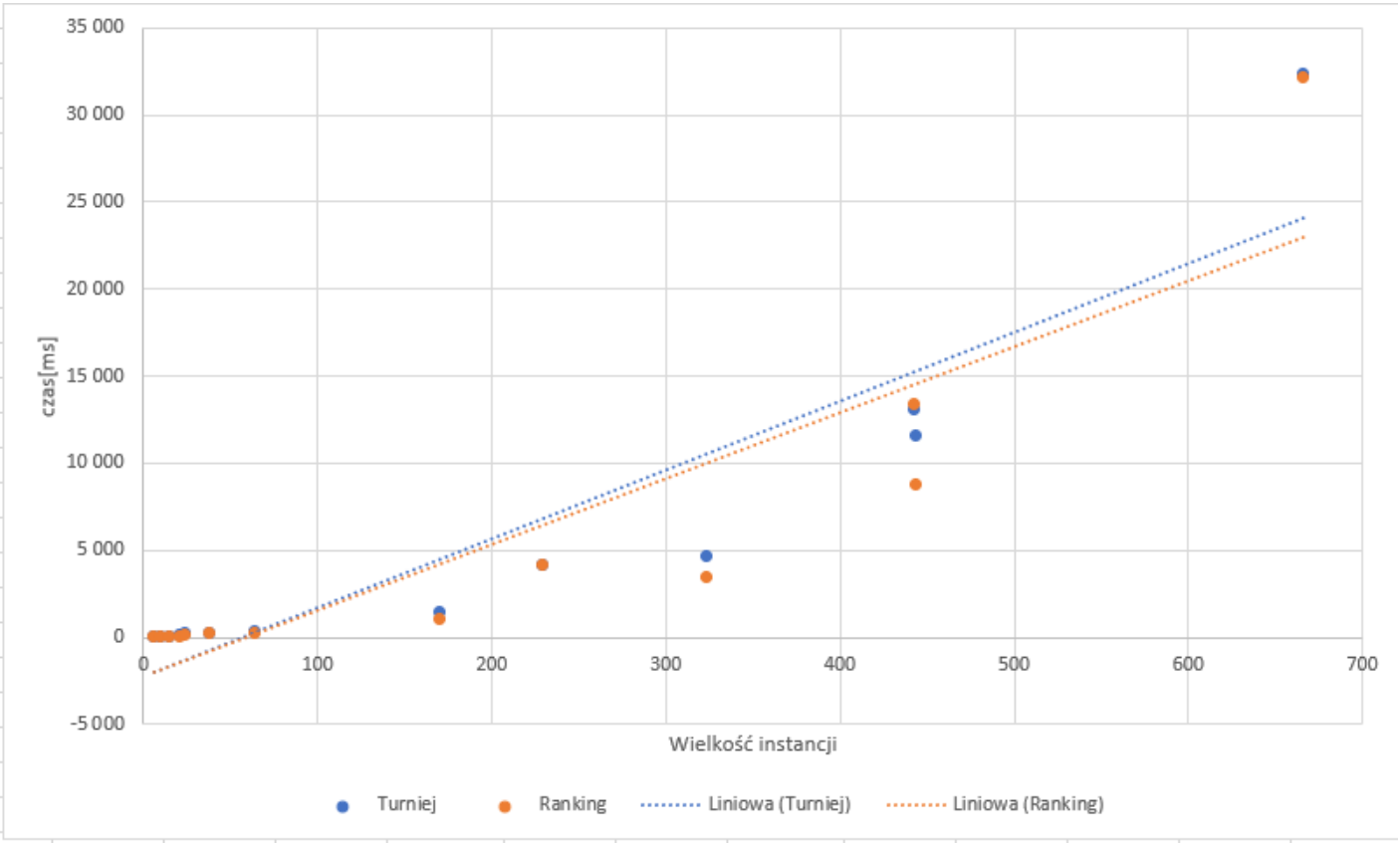
- Prawdopodobieństwo krzyżowania = 90%
- Prawdopodobieństwo mutacji = 5%
- Wielkość populacji = 500
- Liczba generacji = 1000
- Dozwolonych powtórzeń = 150
- Populacja początkowa – algorytm Greedy
- Metoda mutacji - Invert

		Turniej			Ranking		
instancja	koszt optymalny	średni koszt	błąd[%]	czas [ms]	średni koszt	błąd[%]	czas [ms]
6	80	80	0%	5	80	0%	9
10	212	212	0%	7	212	0%	25
15	291	291	0%	4	291	0%	16
21	2707	2707	0%	111	2707	0%	65
24	1272	1315	3%	236	1319	4%	142
38	1530	1670	9%	234	1678	10%	206
64	1839	2535	38%	379	2557	39%	264
170	2755	3901	42%	1484	3915	42%	1033
229	134602	144391	7%	4177	143961	7%	4123
323	1326	1728	30%	4641	1732	31%	3456
442	50778	58244	15%	13079	58537	15%	13447
443	2720	3765	38%	11607	3848	41%	8739
666	194358	359553	85%	32354	358925	85%	32146

Tabela 5: Wyniki badania wpływu metody selekcji na czas i jakość wykonania algorytmu



Wykres 6: Wykres zależności błędu rozwiązania od wielkości instancji porównująca metody selekcji



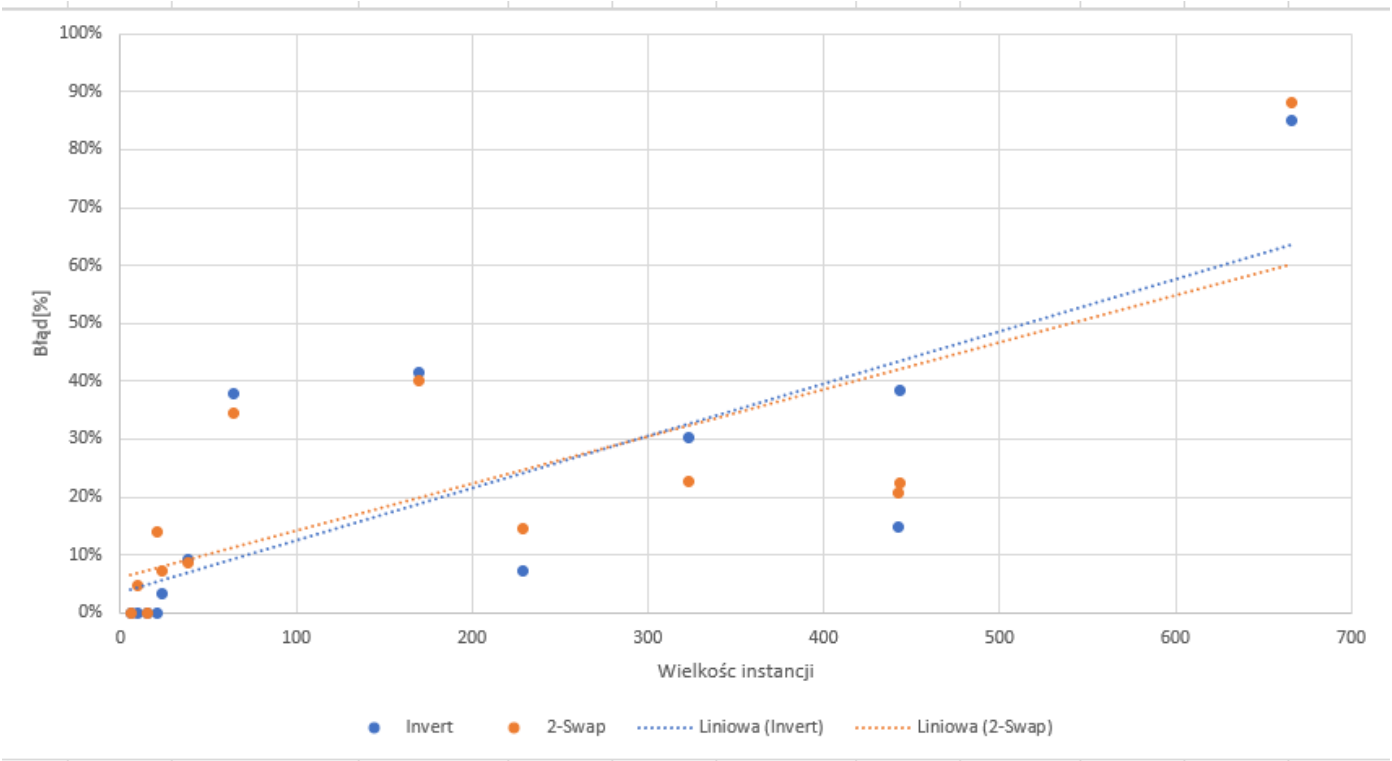
Wykres 7: Wykres zależności czasu wykonywania algorytmu od wielkości instancji porównująca metody selekcji

Zostały zbadane dwie metody selekcji: turniejowa oraz rankingowa. Na *wykresie 6* i *wykresie 7,* można zobaczyć kolejno, porównanie tych metod pod względem wielkości błędu oraz czasu wykonywania algorytmu. Analizując wcześniej wymienione wykresy oraz *tabelę 5,* można dojść do wniosku, że obie metody zwracają podobne wyniki, jednak linie trendu metody turniejowej pokazują tendencje do uzyskiwania wyniku z mniejszym błędem, kosztem czasu wykonywania algorytmu.

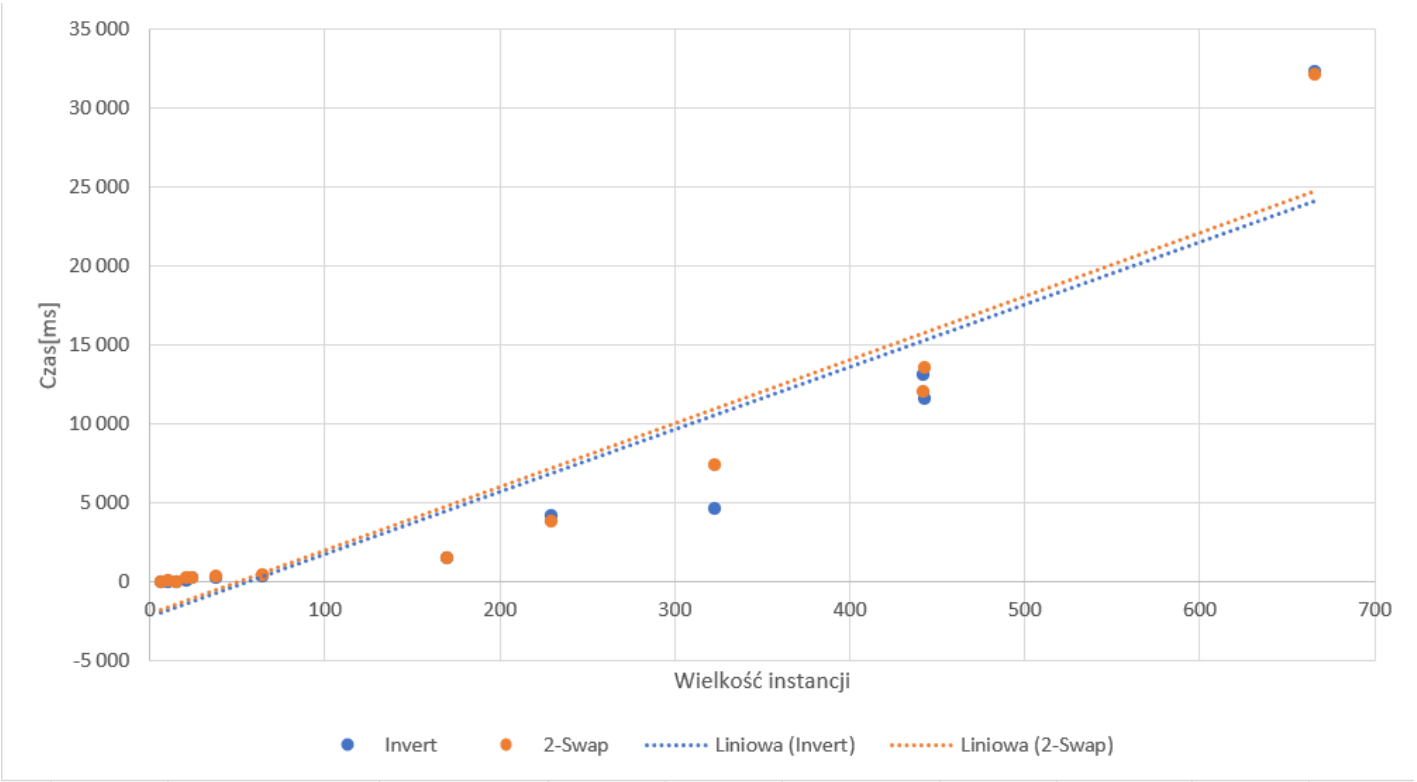
- Badanie wpływu metody mutacji na czas i jakość wykonywania algorytmu

		Invert			2-Swap		
instancja	koszt optymalny	średni koszt	błąd[%]	czas [ms]	średni koszt	błąd[%]	czas [ms]
6	80	80	0%	5	80	0%	5
10	212	212	0%	7	222	5%	97
15	291	291	0%	4	291	0%	7
21	2707	2707	0%	111	3086	14%	286
24	1272	1315	3%	236	1364	7%	216
38	1530	1670	9%	234	1662	9%	294
64	1839	2535	38%	379	2474	35%	403
170	2755	3901	42%	1484	3860	40%	1459
229	134602	144391	7%	4177	154001	14%	3823
323	1326	1728	30%	4641	1625	23%	7424
442	50778	58244	15%	13079	61299	21%	12001
443	2720	3765	38%	11607	3330	22%	13544
666	194358	359553	85%	32354	365794	88%	32152

Tabela 7: Wyniki badania dla wpływu metody mutacji



Wykres 8: Wykres zależności błędu od wielkości instancji dla różnych metod mutacji



Wykres 9: Wykres zależności czasu wykonywania algorytmu dla różnych metod mutacji

Jak można zauważyć analizując *wykres 8* oraz *wykres 9,* metoda mutacji Invert jest szybsza i lepsza jeśli chodzi o jakość rozwiązania. Linie trendu wskazują jej przewagę nad metodą 2-swap, która widocznie odstaje.

7. Analiza wyników

Finałowe badanie:

Przetestowano po 10 razy każdą instancję, aby porównać ją z algorytmem symulowanego wyżarzania. Plik .INI wygląda dla tego badania następująco:

gr17.tsp.txt 10 2085 [?]
gr21.tsp.txt 10 2707 [0 6 7 5 15 4 8 2 1 20 14 13 12 17 9 16 18 19 10 3 11 0]
gr24.tsp.txt 10 1272 [?]
ftv33.atsp 10 1286 [?]
ftv44.atsp 10 1613 [?]
ftv55.atsp 10 1608 [?]
ftv70.atsp 10 2755 [?]
gr96.txt 10 55209 [?]
ftv170.atsp 10 2755 [?]
gr229.txt 10 134602 [?]
rbg323.atsp 10 1326 [?]
pcb442.tsp.txt 10 50778 [?]
rbg443.atsp 10 2720 [?]
tsp666.tsp.txt 10 194358 [?]

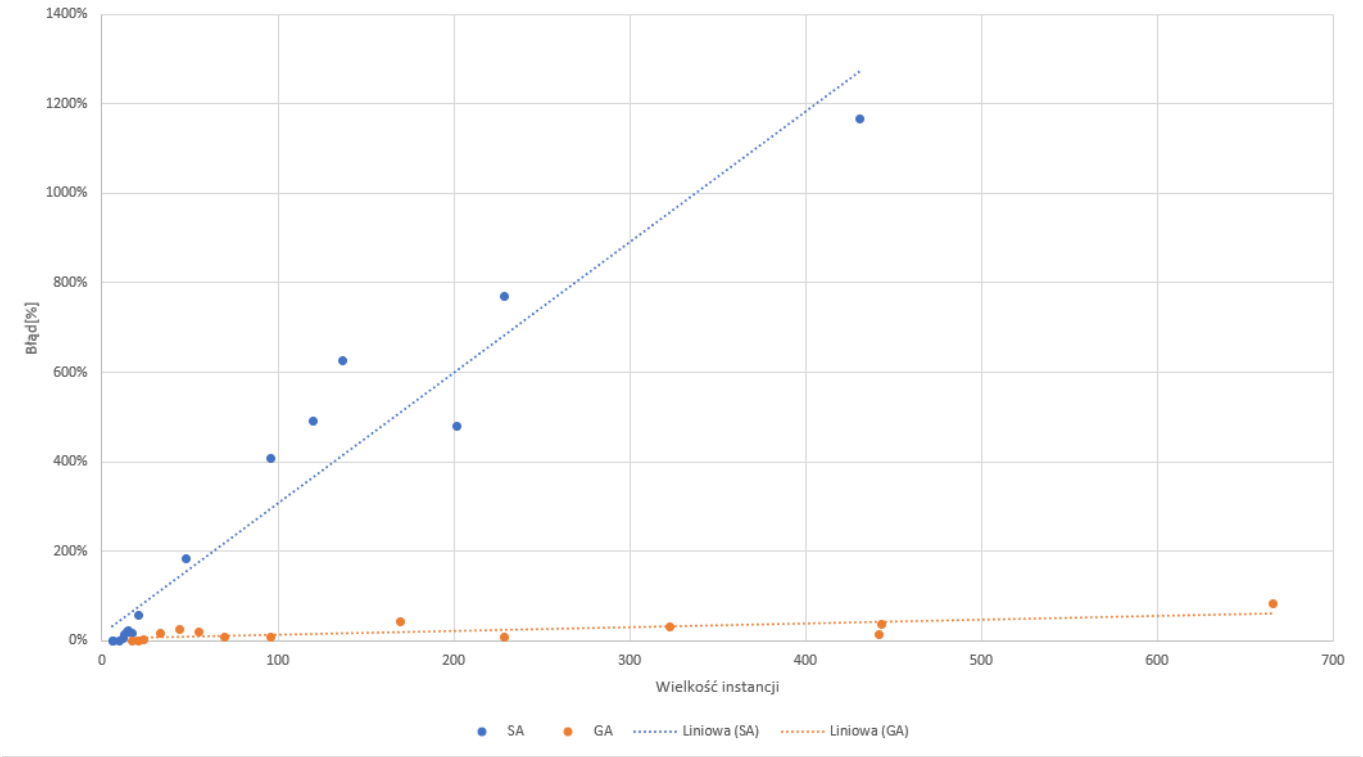
outputFinal.csv

Parametry dla algorytmu genetycznego zmienione dla :

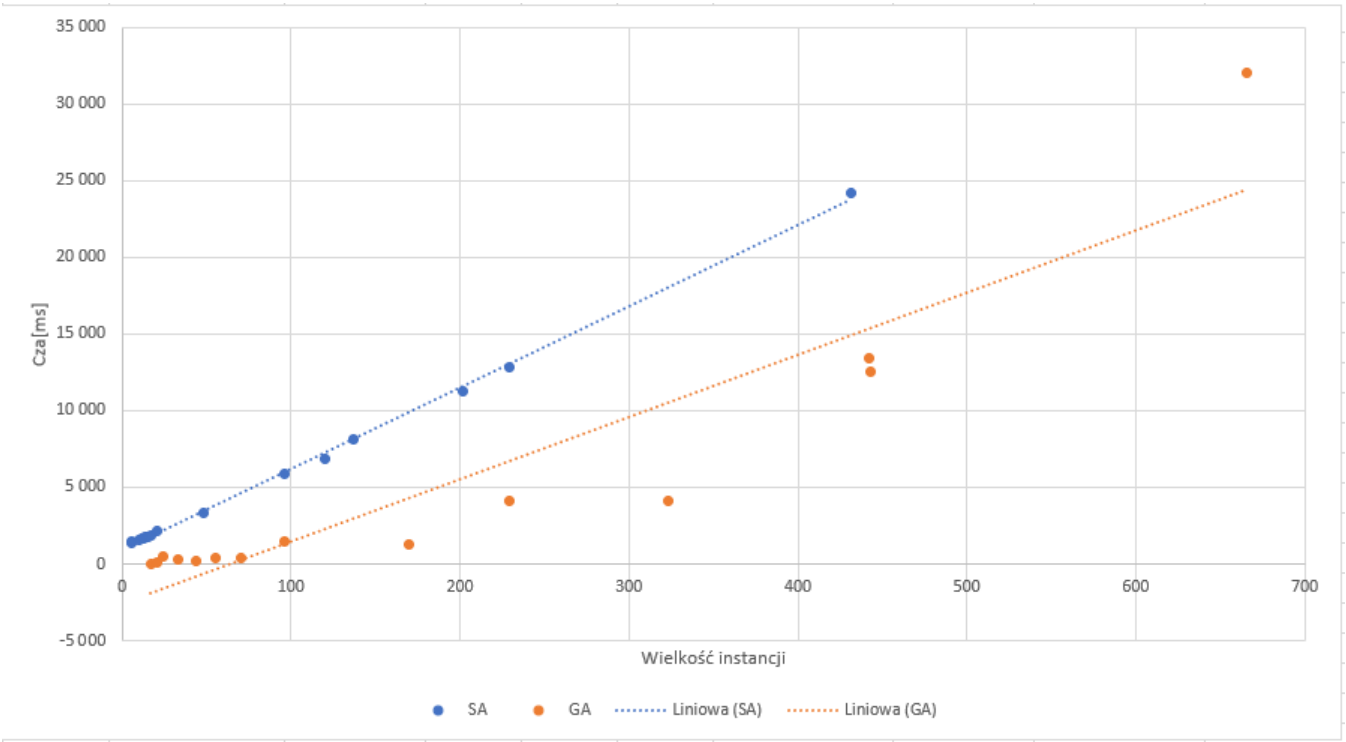
- Prawdopodobieństwo krzyżowania = 90%
- Prawdopodobieństwo mutacji = 5%
- Wielkość populacji = 500
- Liczba generacji = 1000
- Dozwolonych powtórzeń = 150
- Populacja początkowa – algorytm Greedy
- Metoda wyboru selekcji - Invert

Symulowane wyżarzanie					Algorytm Genetyczny				
instancja	koszt optymalny	średni koszt	błąd[%]	czas[ms]	instancja	koszt optymalny	średni koszt	błąd[%]	czas[ms]
6	132	132	0%	1352	17	2085	2085	0%	12
6	80	80	0%	1497	21	2707	2707	0%	100
10	212	212	0%	1541	24	1272	1309	3%	443
12	264	277	5%	1661	33	1286	1504	17%	273
13	269	307	14%	1684	44	1613	1999	24%	208
14	282	340	21%	1724	55	1608	1941	21%	345
15	291	360	24%	1735	70	2755	2505	9%	350
17	2085	2452	18%	1901	96	55209	59181	7%	1433
21	2707	4281	58%	2136	170	2755	3905	42%	1254
48	5046	14302	183%	3290	229	134602	145489	8%	4153
96	55209	280709	408%	5832	323	1326	1728	30%	4121
120	6942	40987	490%	6850	442	50778	58385	15%	13381
137	69853	506391	625%	8087	443	2720	3728	37%	12566
202	40160	232270	478%	11271	666	194358	354291	82%	32030
229	134602	1170554	770%	12803					
431	171414	2172139	1167%	24157					

Tabela 8: Porównanie czasu wykonania i jakości rozwiązania algorytmów symulowanego wyżarzania oraz algorytmu genetycznego



Wykres 10: porównanie błędów procentowych dla algorytmów: symulowanego wyżarzania i algorytmu genetycznego



Wykres 11: porównanie czasu wykonywania algorytmów: symulowanego wyżarzania i algorytmu genetycznego

Po wszystkich przeprowadzonych badaniach, możemy stwierdzić, że algorytm genetyczny pozwala osiągnąć wyniki trudno osiągalne dla innych algorytmów. Porównując go z algorytmem symulowanego wyżarzania, jasno widzimy, że jakość wykonania, a także czas wykonywania algorytmu, jest dużo lepszy. W przypadku pracy z dużymi instancjami, ta różnica jest wyjątkowo widoczna, pokazują nam to *wykres 11* oraz *wykres 12.* Z tego wynika, że można go wykorzystać właśnie przy pracy z takimi instancjami. Oczywiście, nie są to dokładne wyniki ale dzięki odpowiedniemu dostosowywaniu parametrów, możemy uzyskać znacząco lepsze rezultaty niż w przypadku innych algorytmów.