Prowadzący: dr inż. Tomasz Kapłon
termin projektu: wt TP, 15.15 – 16.55

Wrocław, 15.01.2017
Weronika Luźna, 209879

Projektowanie efektywnych algorytmów
Zadanie 3.

Algorytm genetyczny
dla problemu komiwojażera.

1. Wstęp.

Celem zadania było zaimplementowanie i przetestowanie algorytmu genetycznego dla problemu komiwojażera.

Problem komiwojażera polega na odnalezieniu minimalnego cyklu Hamiltona w pełnym grafie – przejściu przez n miast jak najkrótszą drogą, odwiedzając każde z nich dokładnie raz. Istnieją dwa rodzaje tego problemu: symetryczny (odległość z punktu A do B jest identyczna jak z B do A) oraz niesymetryczny (odległości są różne w każdą stronę).

Algorytm genetyczny to próba zasymulowania w pamięci komputera populacji jakiegoś gatunku. Każda populacja składa się z pojedynczych osobników. Osobniki te mogą się między sobą krzyżować, mogą również następować samoistne zmiany w strukturze pojedynczego osobnika (mutacja). W wyniku krzyżowania i mutacji powstają nowe osobniki. Ze względu na fakt, że populacja ma swój z góry narzucony maksymalny rozmiar, wymagana jest selekcja (część najmniej przystosowanych osobników zostaje usunięta). Cały proces (krzyżowanie – mutacja - selekcja) powtarzany jest w nieskończoność.

Rozwiązując problem komiwojażera jako gatunek przyjęto trasę (permutację n miast). Każdy osobnik pamięta jedną permutację. Liczbę takich osobników określono zadanym jako parametr rozmiarem populacji. Krzyżowanie wykonywano za pomocą operatora krzyżowania OX. Mutacja to zamiana miejscami dwóch miast w osobniku. Podczas selekcji wybierano do następnej populacji osobników najlepiej przystosowanych (to znaczy takich które reprezentowały trasy o najmniejszym koszcie).

2. Implementacja.

Program powstał przy użyciu języka C++ w środowisku Microsoft Visual Studio 2015. Oparty jest o programowanie zorientowane obiektowo; składa się z trzech klas. Pierwsza z nich: *Matrix* zawiera macierz, będącą reprezentacją grafu oraz metody potrzebne do wczytywania danych. Druga klasa: *Solution* służy do przechowywania i obsługi rozwiązań (losowanie x0, zamienianie miejscami losowych elementów danego rozwiązania, operator przypisania). Trzecia klasa: *Genetic* realizuje algorytm od początku do końca, czyli zawiera:

- inicjalizację parametrów (rozmiar populacji, współczynnik krzyżowania, współczynnik mutacji)

- obliczanie wartości funkcji (długości ścieżki dla danej permutacji)
- operator krzyżowania OX
- operator mutowania (zamiana miejscami dwóch miast)
- operację selekcjonowania nowej populacji
- algorytm

Przy czym zastosowano selekcję ruletkową (im większe dopasowanie ma dany osobnik, tym większe prawdopodobieństwo, że zostanie wybrany) oraz uwzględniono elitaryzm (dwóch najlepszych osobników przechodzi zawsze do następnej populacji).

Parametry zmieniane podczas testów:
- współczynnik krzyżowania (*c_r)* = {0.8, 0.85, 0.9, 0.95 , 0.99}
- współczynnik mutacji (*m_r)* = {0.01, 0.05, 0.1}
- rozmiar populacji (*S*) = {10, 20, 30, 50}

3. Testowanie.

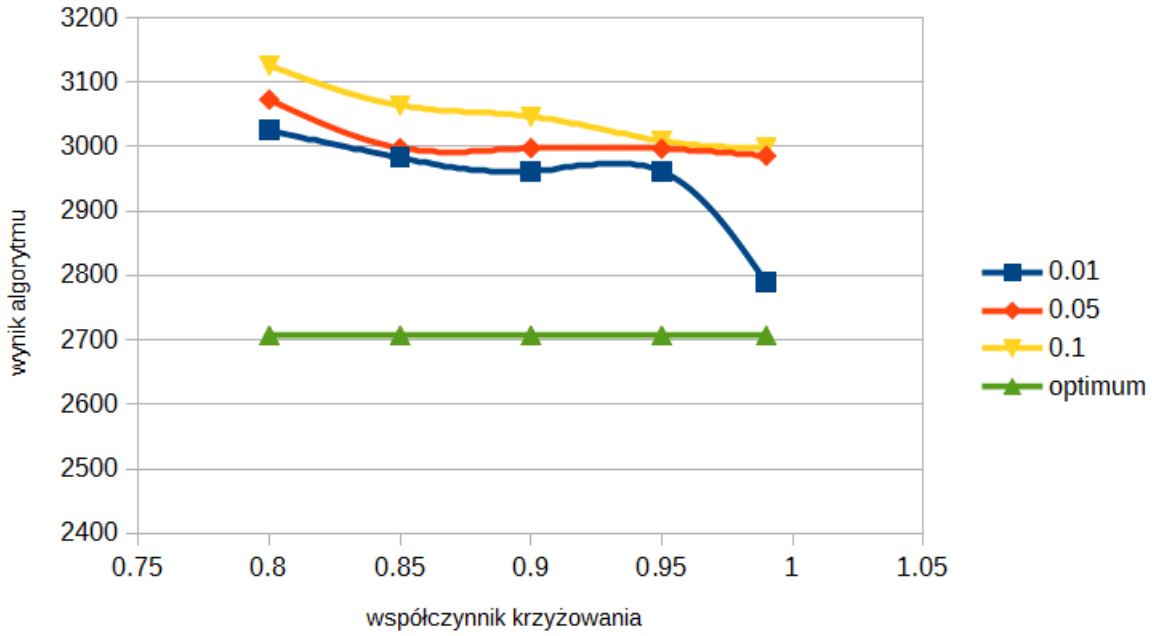
Do testów wybrano trzy instancje symetryczne i dwie asymetryczne o znanych rozwiązaniach optymalnych: (gr21, dantzig42, gr120, ftv33, ftv70). Dla każdej z nich zmierzono czas działania algorytmu przy zmiennych parametrach wejściowych. Pomiaru czasu dokonano przy pomocy gotowych funkcji z biblioteki <windows.h>. Dla wszystkich pomiarów ustalono limit iteracji przy braku poprawy rozwiązania na 100.

3.1. Instancja gr21.tsp, n=21, wynik optymalny: 2707.

Tabela 1. Zestawienie wyników algorytmu i pomiarów czasu dla gr21.

c_r / **m_r**	**0.8**	**0.85**	**0.9**	**0.95**	**0.99**
	WYNIK				
0.01	3025	2983	2962	2962	2789
0.05	3073	2998	2998	2997	2986
0.1	3126	3064	3047	3009	2999

	CZAS [ms]				
0.01	0.075	0.052	0.051	0.051	0.049
0.05	0.084	0.050	0.051	0.057	0.072
0.1	0.070	0.048	0.056	0.049	0.049

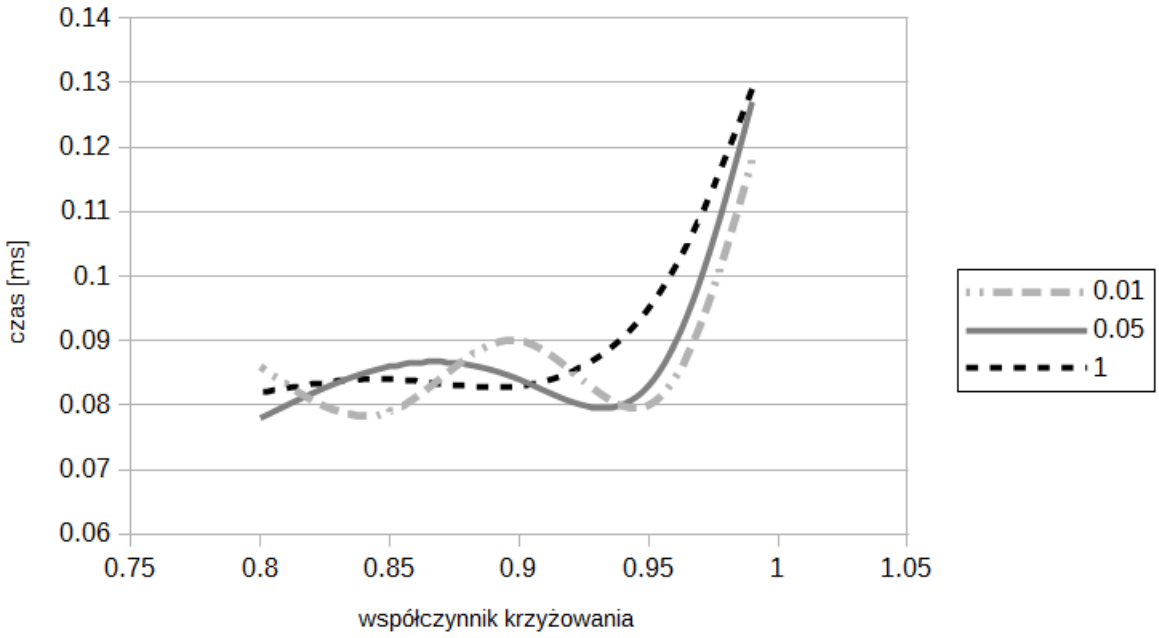


Wykres 1. Zależność wyniku algorytmu od współczynnika krzyżowania dla stałych współczynników mutacji.

3.2. Instancja dantzig42.tsp, n=42, wynik optymalny: 699.

Tabela 2. Zestawienie wyników algorytmu i pomiarów czasu dla dantzig42.

c_r / **m_r**	**0.8**	**0.85**	**0.9**	**0.95**	**0.99**
	WYNIK				
0.01	904	882	848	836	830
0.05	887	841	831	794	783
0.1	899	862	815	807	793
	CZAS [ms]				
0.01	0.168	0.080	0.090	0.079	0.086
0.05	0.187	0.083	0.084	0.086	0.078
0.1	0.179	0.095	0.083	0.084	0.082

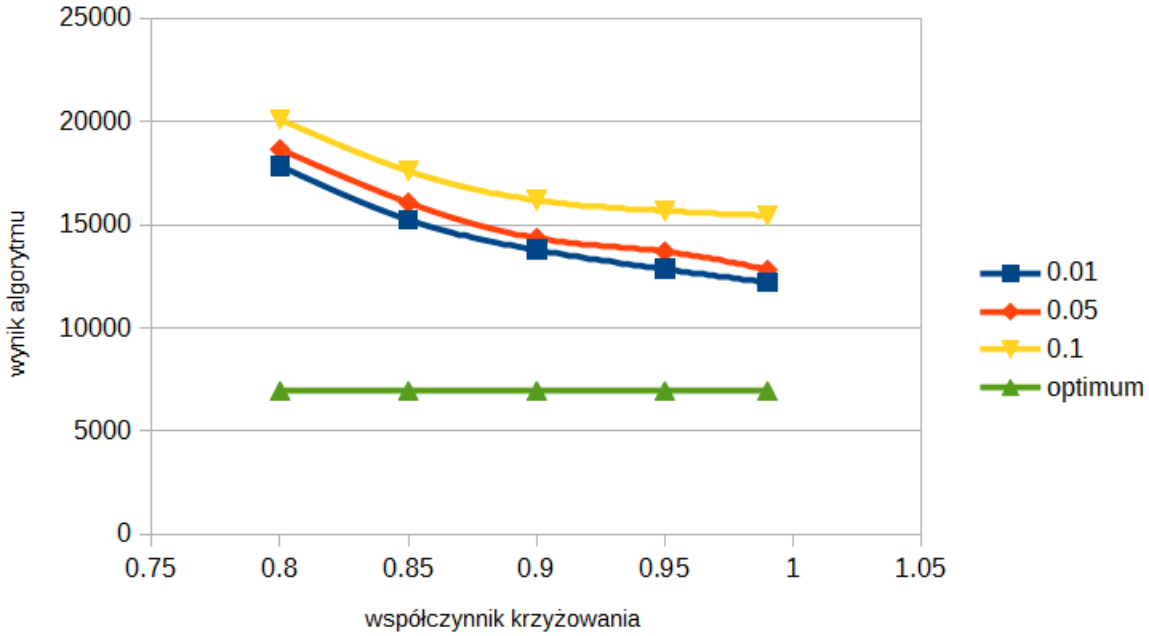


Wykres 2. Zależność czasu wykonywania algorytmu od współczynnika krzyżowania dla stałych współczynników mutacji.

3.3. Instancja gr120.tsp, n=120, wynik optymalny: 6942.

Tabela 3. Zestawienie wyników algorytmu i pomiarów czasu dla gr120.

c_r / **m_r**	**0.8**	**0.85**	**0.9**	**0.95**	**0.99**
	WYNIK				
0.01	17867	15254	13796	12866	12218
0.05	18677	16085	14378	13716	12829
0.1	20122	17619	16201	15688	15458
	CZAS [ms]				
0.01	0.668	0.285	0.341	0.267	0.316
0.05	0.670	0.333	0.343	0.245	0.350
0.1	0.699	0.401	0.325	0.245	0.208



Wykres 3. Zależność wyniku algorytmu od współczynnika krzyżowania dla stałych współczynników mutacji.

3.4. Instancja ftv33.atsp, n=34, wynik optymalny: 1286.

Tabela 4. Zestawienie wyników algorytmu i pomiarów czasu dla ftv33.

c_r / **m_r**	**0.8**	**0.85**	**0.9**	**0.95**	**0.99**
	WYNIK				
0.01	1693	1567	1471	1440	1432
0.05	1668	1668	1668	1659	1600
0.1	1804	1685	1659	1626	1626
	CZAS [ms]				
0.01	0.135	0.079	0.074	0.066	0.068
0.05	0.106	0.063	0.066	0.074	0.068
0.1	0.103	0.067	0.073	0.064	0.065

3.5. Instancja ftv70.atsp, n=71, wynik optymalny: 1950

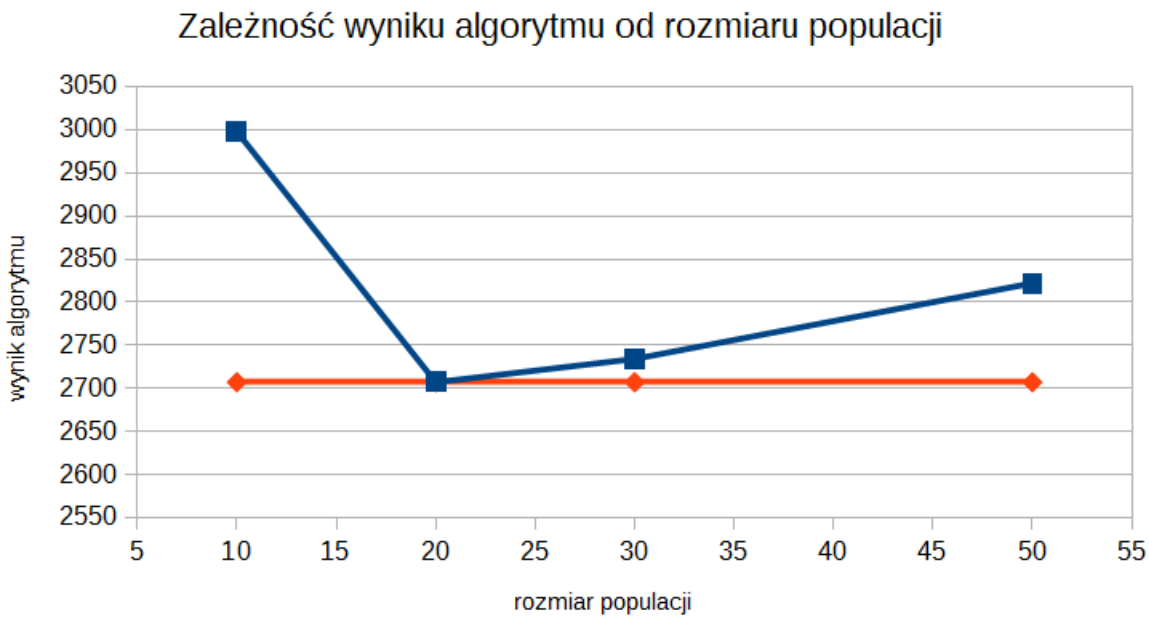
Tabela 5. Zestawienie wyników algorytmu i pomiarów czasu dla ftv70.

c_r / m_r	0.8	0.85	0.9	0.95	0.99
	WYNIK				
0.01	4709	3913	3553	3064	3022
0.05	4533	3985	3471	3404	3388
0.1	4066	3847	3801	3747	3654
	CZAS [ms]				
0.01	0.265	0.227	0.153	0.147	0.120
0.05	0.269	0.140	0.183	0.118	0.135
0.1	0.278	0.134	0.120	0.131	0.117

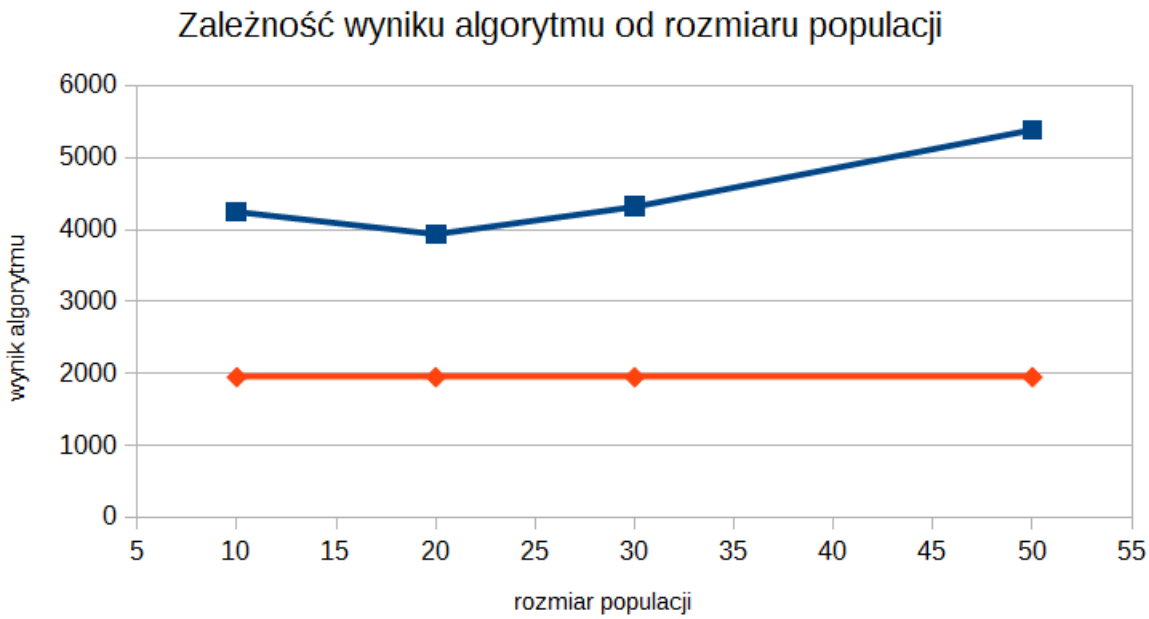
3.6. Wszystkie instancje dla zmiennego rozmiaru populacji (S).

Tabela 6. Zestawienie wyników algorytmu i pomiarów czasu dla 5 instancji.

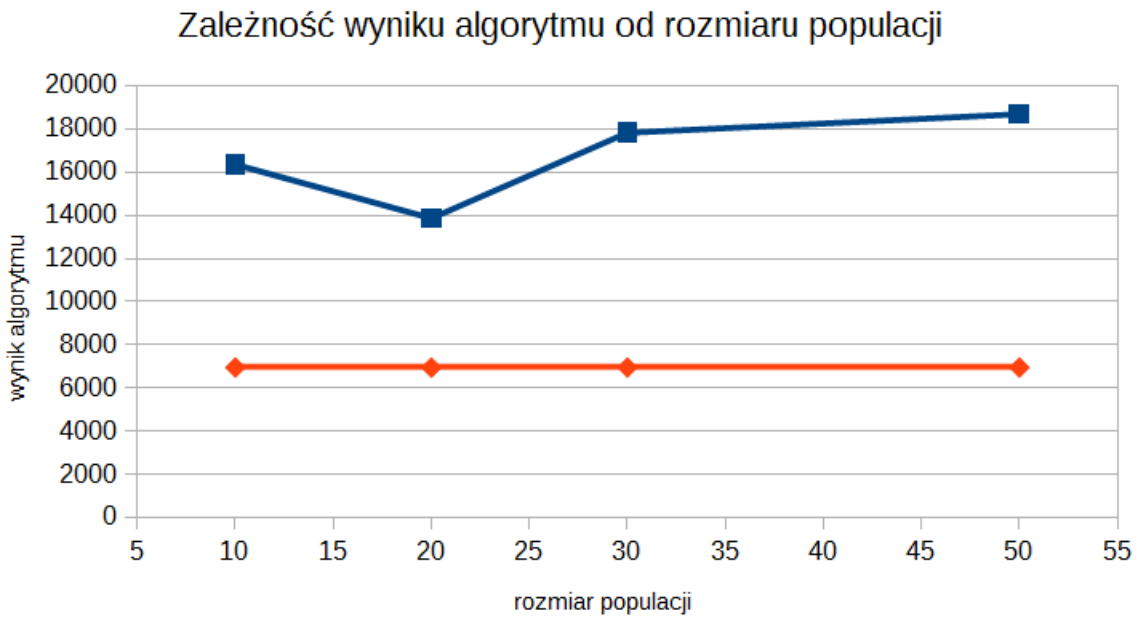
S / instancja	10	20	30	50
	WYNIK			
gr21	2998	2707	2734	2722
dantzig42	854	953	908	884
gr120	16352	13862	17822	18688
ftv33	1734	1658	1569	1682
ftv70	4248	3938	4318	5388
	CZAS [s]			
gr21	0.041	0.071	0.106	0.158
dantzig42	0.070	0.206	0.359	0.565
gr120	0.450	0.915	1.259	1.958
ftv33	0.049	0.130	0.184	0.358
ftv70	0.170	0.243	0.471	0.658



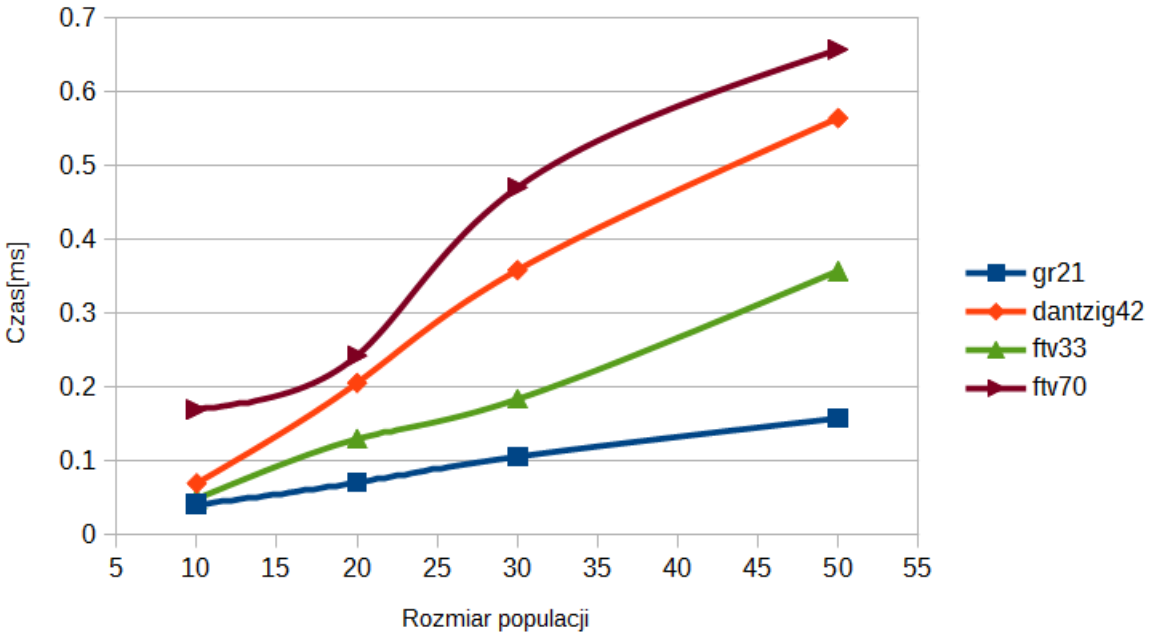
Wykres 4. Zależność wyniku algorytmu od rozmiaru populacji dla instancji gr21.



Wykres 5. Zależność wyniku algorytmu od rozmiaru populacji dla instancji ftv70.



Wykres 6. Zależność wyniku algorytmu od rozmiaru populacji dla instancji gr120.



Wykres 7. Zależność czasu wykonywania algorytmu od rozmiaru populacji oraz wielkości instancji.

4. Porównanie algorytmu gentycznego (GA) z przeszukiwaniem z zakazami (TS) i symulowanym wyżarzaniem (SA).

Tabela 7. Zestawienie najlepszych wyników.

instancja	TS	Błąd TS [%]	SA	Błąd SA [%]	GA	Błąd GA [%]	optimum
gr21	2707	0	2707	0	2707	0	2707
dantzig42	830	18.74	701	2.86	783	12.01	699
gr120	7014	1.03	7291	5.02	12218	76.00	6942
ftv33	1368	6.37	1548	20.37	1432	11.35	1286
ftv70	2016	3.38	2640	35.38	3022	54.97	1950
		5.90		**12.73**		**30.86**	

5. Wnioski.

W trakcie realizacji projektu zauważono, że dobór wartości współczynników krzyżowania i mutacji w algorytmie ma ogromny wpływ na jakość zwracanego rozwiązania. Na podstawie przeprowadzonych badań wywnioskowano, że wyniki najbliższe optymalnych udaje się uzyskać dla najmniejszego współczynnika mutacji (1 procentowego - bliskiego zeru) oraz największego współczynnika krzyżowania (bliskiego 100 procent), co widać na wykresach 1 i 3.

Dodatkowo zauważono, jak pokazuje wykres 2., że zwiększanie współczynnika krzyżowania osobników powoduje zwiększanie się czasu wykonywania algorytmu. Czas wydłuża się również wraz ze wzrostem rozmiaru populacji (wykres 7.) oraz jest ogółem porównywalnie większy, im większy jest rozmiar rozpatrywanej instancji.

Wykresy 4., 5. i 6. pokazują, że najlepsze wyniki algorytmu uzyskiwano dla populacji o rozmiarze wynoszącym 20.

Na zestawieniu wszystkich trzech algorytmów realizowanych w trakcie przedmiotu, algorytm genetyczny wypadł najsłabiej; zwrócił w najlepszych przypadkach najmniej optymalne rozwiązania. Co więcej im większą instancję rozwiązywano, tym większy wychodził błąd.