Marcin Całkowski
nr 218191
Śr. 15.15

Data oddania: 23.01.2018r.
Prowadzący: Dr inż. Tomasz Kapłon

Projektowanie efektywnych algorytmów

Sprawozdanie

Zadanie projektowe nr 3

Algorytm genetyczny dla problemu komiwojażera

1. Wstęp

Celem projektu jest zastosowanie 3 algorytmów dla wcześniej wybranego zagadnienia, w tym przypadku problemu komiwojażera a następnie porównanie wyników otrzymanych przy użyciu każdego z nich. W trzecim zadaniu projektowym zbadane zostało działanie algorytmu genetycznego na podstawie napisanego algorytmu, badając go w zależności od najbardziej istotnych parametrów dla wcześniej wybranych instancji testowych. Uzyskane wyniki zostaną potem porównane ze sobą pod względem dokładności otrzymanego wyniku, oraz porównane z metodami z poprzednich etapów projektu czyli Tabu Search oraz Branch and Bound.

2. Algorytm genetyczny

Algorytm genetyczny jest heurystyką przeszukującą przestrzeń alternatywnych rozwiązań bazującą na zjawisku ewolucji biologicznej. Zadaniem algorytmu jest symulowanie populacji danego rozwiązania dążącego do jak największego przystosowania się do otoczenia, czyli do uzyskania jak najlepszego rozwiązania badanego problemu. Wykorzystując naturalne mechanizmy takie jak rozmnażanie czy mutację osobników staramy się przystosować kolejne pokolenia rozwiązań by coraz bardziej zbliżać się do optymalnego rozwiązania (jednak nie mamy żadnej gwarancji na jego odnalezienie).

Podstawą algorytmu jest **populacja**, czyli grupa osobników danego problemu. To na jej podstawie przebiega dalsze dostosowywanie się osobników do otoczenia gdyż jest ona bazą dla wykonywanego rozmnażania. Początkowa populacja generowana jest zazwyczaj losowo. Kiedy istnieje już populacja, następuje właściwa część algorytmu zaczynając od **selekcji**. Polega ona na wybraniu jak najbardziej optymalnych osobników do późniejszego rozmnożenia. Zazwyczaj są to osobniki najlepiej przystosowane, jednak nie jest to regułą.

Po przeprowadzeniu selekcji wybrane osobniki przystępują do **rozmnażania**, podczas którego „krzyżują się” wyselekcjonowane wcześniej osobniki. Zazwyczaj brane do krzyżowania są dwa osobniki z których powstają dwa kolejne (dzieci). Dzięki temu otrzymujemy osobniki nowe, będące przemieszaniem się rozwiązań rodziców, co po wielokrotnym wykonaniu daje nam

nową pulę osobników z innym rozwiązaniem. Dodatkowym elementem mogącym zwiększyć różnorodność otrzymanych osobników jest **mutacja**. Pojawia się ona rzadko i zazwyczaj nie wprowadza wielkich zmian, jednak może pomóc w poprawieniu rozwiązania gdy krzyżowane osobniki nie mogą poprawić rozwiązania przez dłuższy czas. Jako że mutacja przebiega losowo, może ona zarówno poprawić, jak i pogorszyć przystosowanie osobnika. Ostatnim elementem algorytmu jest wybór **nowej populacji**, będącej następną generacją algorytmu. Polega ona na stworzeniu nowej grupy osobników, z do której należy część poprzedniej populacji oraz nowo wygenerowane dzieci skrzyżowanych osobników.

Po wybraniu nowej populacji powtarzana jest cała sekwencja dla nowych osobników, w celu utworzenia kolejnej generacji rozwiązań. Całość algorytmu powtarza się aż do spełnienia warunków zakończenia, wybierając jako wynik końcowy najlepiej przystosowanego osobnika, czyli rozwiązanie z najlepszym uzyskanym wynikiem.

3. Implementacja algorytmu

Napisany algorytm został oparty o standardowy schemat algorytmu genetycznego[1]:

1. *wybór populacji początkowej chromosomów (losowy)*
2. *ocena przystosowania chromosomów*
3. *sprawdzanie warunku zatrzymania*
  a. *selekcja chromosomów - wybór populacji macierzystej (ang. mating pool)*
  b. *krzyżowanie chromosomów z populacji rodzicielskiej*
  c. *mutacja - może być również wykonana przed krzyżowaniem*
  d. *ocena przystosowania chromosomów*
  e. *utworzenie nowej populacji*
4. *wyprowadzenie ,,najlepszego'' rozwiązania*

Algorytm po wczytaniu danych z pliku *config.txt* (w celu ustalenia parametrów działania algorytmu) generuje wektor **nowej populacji**. Populacja wygenerowana zostaje w ilości określonej w pliku konfiguracyjnym całkowicie losowo (tworząc drogę z losowo wybranych miast), po czym obliczana jest droga każdego wygenerowanego rozwiązania. Na końcu populacja jest sortowana pod kątem długości drogi, czyli najlepszego wyniku wylosowanej drogi.

Po utworzeniu pierwszej populacji, następuje pętla działając aż nie zostanie spełniony warunek końcowy działania algorytmu. W programie zastosowane zostało **ograniczenie czasowe** ustalane w pliku konfiguracyjnym. Wewnątrz tej pętli znajduje się główna część algorytmu. Na samym początku przepisywanych jest bezpośrednio **20% najlepszych osobników** do nowej populacji, z pominięciem powtarzających się wyników drogi. Ograniczenie to ma na celu przepisanie najlepszych osobników z poprzedniej puli do dalszego krzyżowania, bez powtarzania się tych samych rozwiązań. Chodź wadą selekcji po wyniku jest fakt, że może zostać odrzucone inne rozwiązanie z takim samym wynikiem, to jednak prawdopodobieństwo takiej sytuacji w badanym problemie jest na tyle małe, że rekompensuje to o wiele mniejszy koszt obliczeń wynikający z braku porównywania całego wektora drogi dwóch różnych osobników.

[1] Na podstawie wiadomości z wykładu. Źródło: http://www.zio.iiar.pwr.wroc.pl/pea/w9_ga_tsp.pdf

Po przepisaniu osobników następuje selekcja dwóch osobników do krzyżowania. Selekcja przebiega na podstawy **metody ruletki**, która losuje liczbę z określonego przedziału wybierając osobnika do krzyżowania, przy czym im lepiej dostosowany osobnik tym większe prawdopodobieństwo wyboru. Jako że TSP to problem minimalizacji, aby odpowiednio ustawić elementy w ruletce wynik danego rozwiązania odejmowany jest od najgorszego znalezionego wyniku do tej pory. Dzięki temu najlepsze drogi mają największą wartość w ruletce, co daje im większe prawdopodobieństwo na wylosowanie.

Po wylosowaniu dwóch osobników, losowane jest na podstawie podanego w konfiguracji prawdopodobieństwa, czy mają zostać skrzyżowane czy też nie. Jeśli odpowiednia wartość zostanie wylosowana, osobniki przechodzą do **krzyżowania metodą OX**. Metoda ta losuje dwa indexy, pomiędzy którymi nastąpi krzyżowanie się osobników zgodnie z tą metodą. Po wykonaniu krzyżowania zostaje obliczona droga dla dzieci i przekazywane są one do następnego etapu którym jest mutacja.

Mutacja podobnie jak samo krzyżowanie wykonywana jest z prawdopodobieństwem określonym w pliku konfiguracyjnym. Gdy zostanie wylosowana dla każdego skrzyżowanego dziecka osobno, następuje jego **mutacja metodą invert** pomiędzy dwoma wylosowanymi indexami. Po mutacji zostaje obliczona nowa wartość drogi dla osobnika.

Po krzyżowaniu oraz po mutacji, bez względu na to czy zaszła czy też nie, utworzona dwójka dzieci zostaje dodana do nowej populacji. W tym momencie algorytm cofa się do momentu selekcji, powtarzając całą sekwencję aż do momentu, gdy zostanie zapełniona cała nowa populacja. Gdy już to nastąpi, wektor starej populacji zostaje zastąpiony wektorem nowej, gdzie po posortowaniu nowej populacji algorytm wraca do punktu selekcji, tym razem z działając z nową populacją. Całość działa dopóki nie zostanie spełnione ograniczenie czasowe, które sprawdzane jest po utworzeniu nowej populacji.

Czas działania algorytmu jest przede wszystkim uzależniony od czasu ograniczającego podanego w konfiguracji. Z kolei jego główna część wewnątrz pętli ograniczonej czasowo ze względu na wykonywaną selekcję ruletkową wykonywaną dopóki nie zapełnimy nowej populacji daje nam **złożoność obliczeniową** $\boldsymbol{O(n^2)}$.

4. Procedura testowania

Testy przygotowanych zestawów zostały wykonane na prywatnym komputerze. Podczas przeprowadzania testów nie były uruchomione żadne dodatkowe aplikacje, a wszelkie procesy w tle zostały ograniczone do minimum poprzez wyłączenie aplikacji w tle oraz odłączeniu komputera od Internetu (zachowanie jedynie procesów wymaganych przez system operacyjny) w celu uzyskania jak najlepszych wyników. Konfiguracja sprzętowa maszyny testującej wygląda następująco:

Procesor: Intel Core i7-4700MQ 2.4 GHz
Pamięć RAM: 16 GB
System operacyjny: Windows 10 Pro N

Jako podstawy testowej użyłem zgodnie z poleceniem plików z podanego źródła[2]. Do testowania użyłem 6 różnych instancji problemu, 3 symetryczne oraz 3 asymetryczne:

gr17.tsp, pa561.tsp, si1032.tsp, br17.atsp, ftv170.atsp i rbg443.atsp

Każdy z plików posiada inny zestaw wierzchołków, w różnej liczbie oraz o różnej odległości względem siebie. Również do każdego z nich dołączona jest znaleziona optymalna ścieżka, co pozwoli nam porównać otrzymane wyniki podczas testów z najlepszymi znalezionymi do tej pory, dzięki czemu będziemy mogli określić czy dany algorytm działa prawidłowo. Wczytywanie plików **.tsp** i **.atsp** ograniczyłem jedynie do obsługi wybranych plików, jednak program powinien wczytać dowolne dane z wcześniej podanego źródła zapisane jako macierz zwykła bądź diagonalna.

Pomiary zostały przeprowadzenie poprzez wykonanie 10 prób dla każdej testowanej instancji w każdej konfiguracji oraz uśrednieniu ich wyników. Wszystkie testy przeprowadzone zostały z czasem działania algorytmu równemu **30 sekund**. Ze względu na ograniczony czas przeznaczony do testowania, testy zostały podzielone na dokładniejsze wyłącznie dla instancji pa561.tsp oraz ogólne testy dla pozostałych instancji z parametrami wybranymi na podstawie wcześniej badanego pliku oraz ich wstępnego testowania.

Podczas badań sprawdzona zostały sprawdzone zależności najistotniejszych parametrów, tj. **wielkość populacji, prawdopodobieństwo krzyżowania** oraz **prawdopodobieństwo mutacji**. Parametry testowe dla instancji pa561.tsp wyglądają następująco (n – wielkość instancji):

Czas działania: 30 sekund
Rozmiar populacji: n/4; n/2; n; n*2
Prawdopodobieństwo krzyżowania: 0,1; 0,2; 0,3; (…), 0,9; 1,0
Prawdopodobieństwo mutacji: 0,1; 0,2; 0,3; (…), 0,9; 1,0

Na podstawie testów instancji pea561.tsp zostały ustalone parametry testowe dla pozostałych instancji. Parametry te charakteryzują się stałym **prawdopodobieństwem mutacji równym 0,1,** oraz **prawdopodobieństwem krzyżowania** z zakresu **0,5 - 1,0**.

Wyniki pomiarów przedstawiają średnią wartość uzyskaną dla badanej kombinacji parametrów. Aby dowiedzieć się jak bardzo wygenerowane rozwiązania różnią się od rozwiązania optymalnego, dla pewnych danych został obliczony błąd względem rozwiązania optymalnego na podstawie wzoru: **B = (Droga(N) – Droga(O))*100 / Droga (O)**, gdzie **Droga(O)** to optymalna droga dla danej instancji, a **Droga(N)** to uzyskana wartość w testach**.** Błąd ten zapisany jest w procentach.

Wyniki optymalne dla poszczególnych instancji:

gr17.tsp: 2085
br17.atsp: 39
ftv170.atsp: 2755
rbg443.atsp: 2720
pa561.tsp: 2763
si1032.tsp: 92650

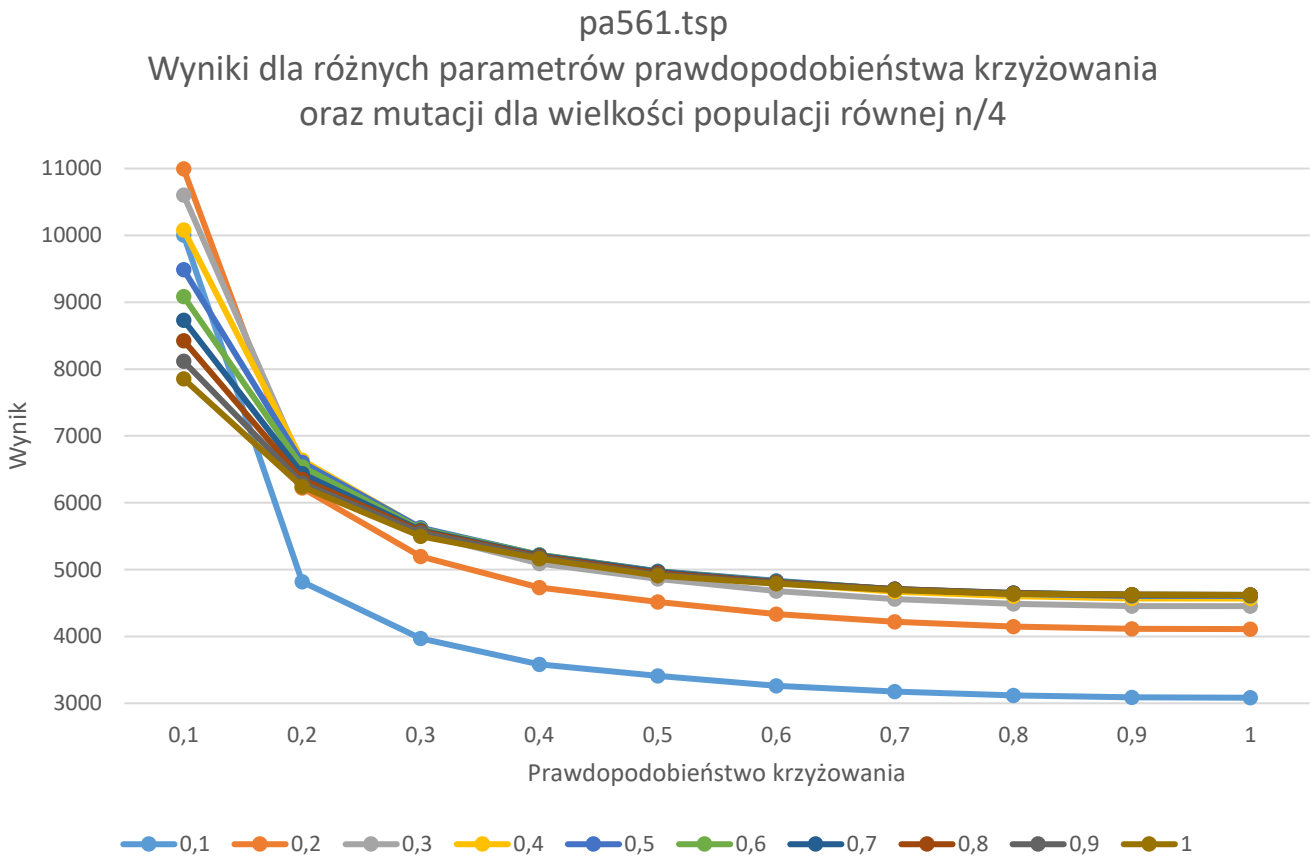
[2] http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/

5. Wyniki

5.1. Testy szczegółowe instancji pa561.tsp

Rozmiar populacji: n/4											
		Prawdopodobieństwo krzyżowania									
Prawdopodobieństwo mutacji		**0,1**	**0,2**	**0,3**	**0,4**	**0,5**	**0,6**	**0,7**	**0,8**	**0,9**	**1**
	0,1	10004	4818	3968	3581	3412	3259	3174	3118	3088	3083
	0,2	10996	6222	5195	4734	4517	4332	4217	4147	4115	4110
	0,3	10604	6571	5539	5092	4859	4681	4560	4487	4454	4453
	0,4	10082	6640	5623	5196	4950	4797	4671	4600	4567	4567
	0,5	9487	6602	5628	5221	4974	4832	4702	4638	4605	4605
	0,6	9084	6536	5610	5220	4973	4824	4709	4649	4618	4616
	0,7	8733	6435	5591	5211	4967	4816	4709	4651	4622	4619
	0,8	8426	6348	5573	5197	4945	4810	4706	4646	4623	4620
	0,9	8116	6292	5549	5181	4921	4797	4701	4641	4624	4621
	1	7853	6241	5497	5165	4909	4788	4693	4637	4624	4621

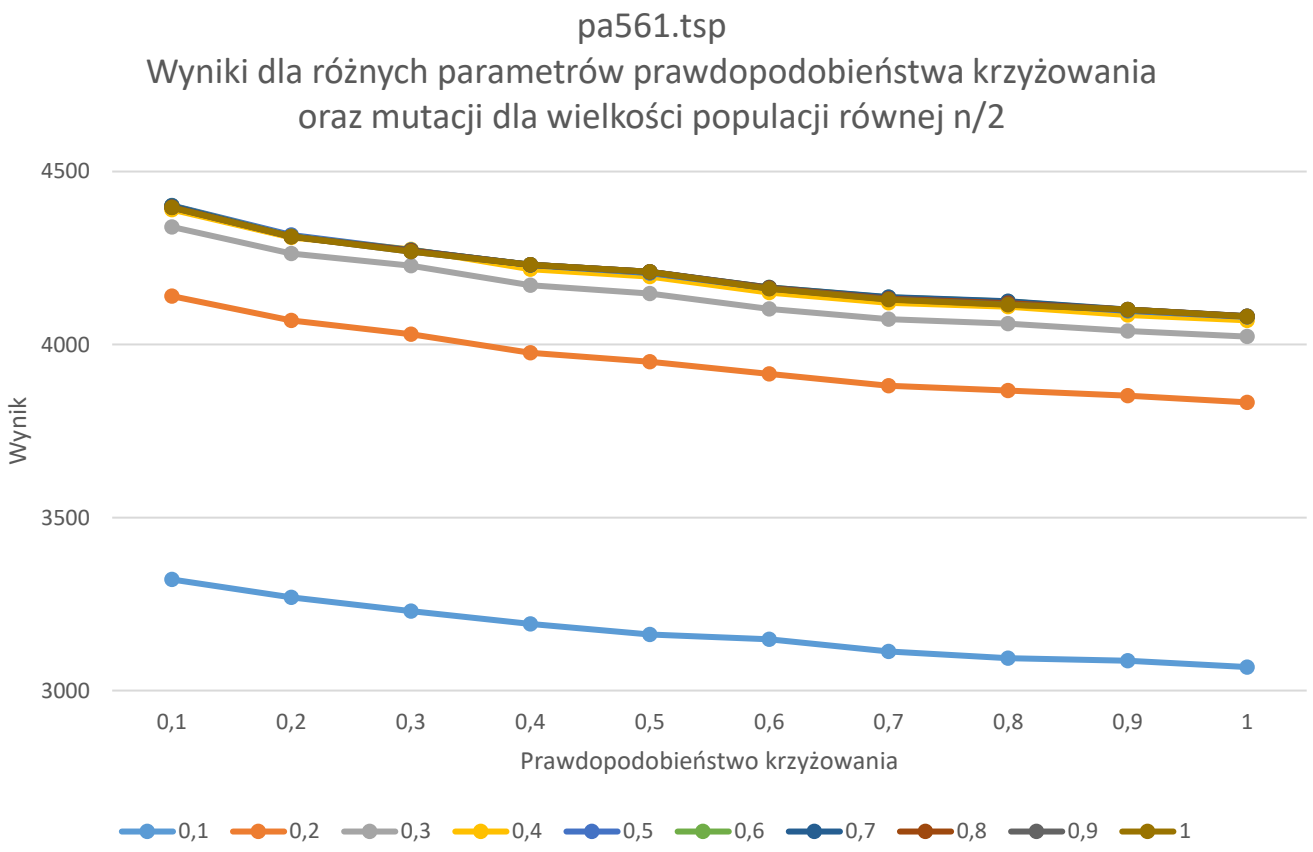
Tabela 1: Wyniki testów dla różnych wartości prawdopodobieństwa krzyżowania oraz mutacji dla rozmiaru populacji równemu n/4.



Wykres 1: Wykres uzyskanych wyników dla różnych wartości prawdopodobieństwa krzyżowania (oś OX) oraz mutacji (serie) dla rozmiaru populacji równemu n/4.

Rozmiar populacji: n/2											
		Prawdopodobieństwo krzyżowania									
Prawdopodobieństwo mutacji		**0,1**	**0,2**	**0,3**	**0,4**	**0,5**	**0,6**	**0,7**	**0,8**	**0,9**	**1**
	0,1	3321	3269	3230	3193	3162	3148	3113	3094	3086	3068
	0,2	4140	4070	4030	3976	3950	3915	3881	3867	3852	3833
	0,3	4340	4263	4228	4171	4147	4103	4073	4060	4039	4023
	0,4	4390	4309	4275	4218	4196	4150	4121	4109	4085	4070
	0,5	4402	4317	4273	4230	4207	4162	4133	4121	4097	4080
	0,6	4401	4313	4272	4231	4209	4165	4136	4124	4100	4082
	0,7	4401	4312	4272	4231	4210	4165	4137	4125	4101	4082
	0,8	4397	4312	4271	4231	4210	4164	4133	4120	4101	4082
	0,9	4396	4312	4269	4231	4210	4162	4131	4116	4101	4081
	1	4396	4312	4269	4231	4210	4161	4130	4115	4101	4081

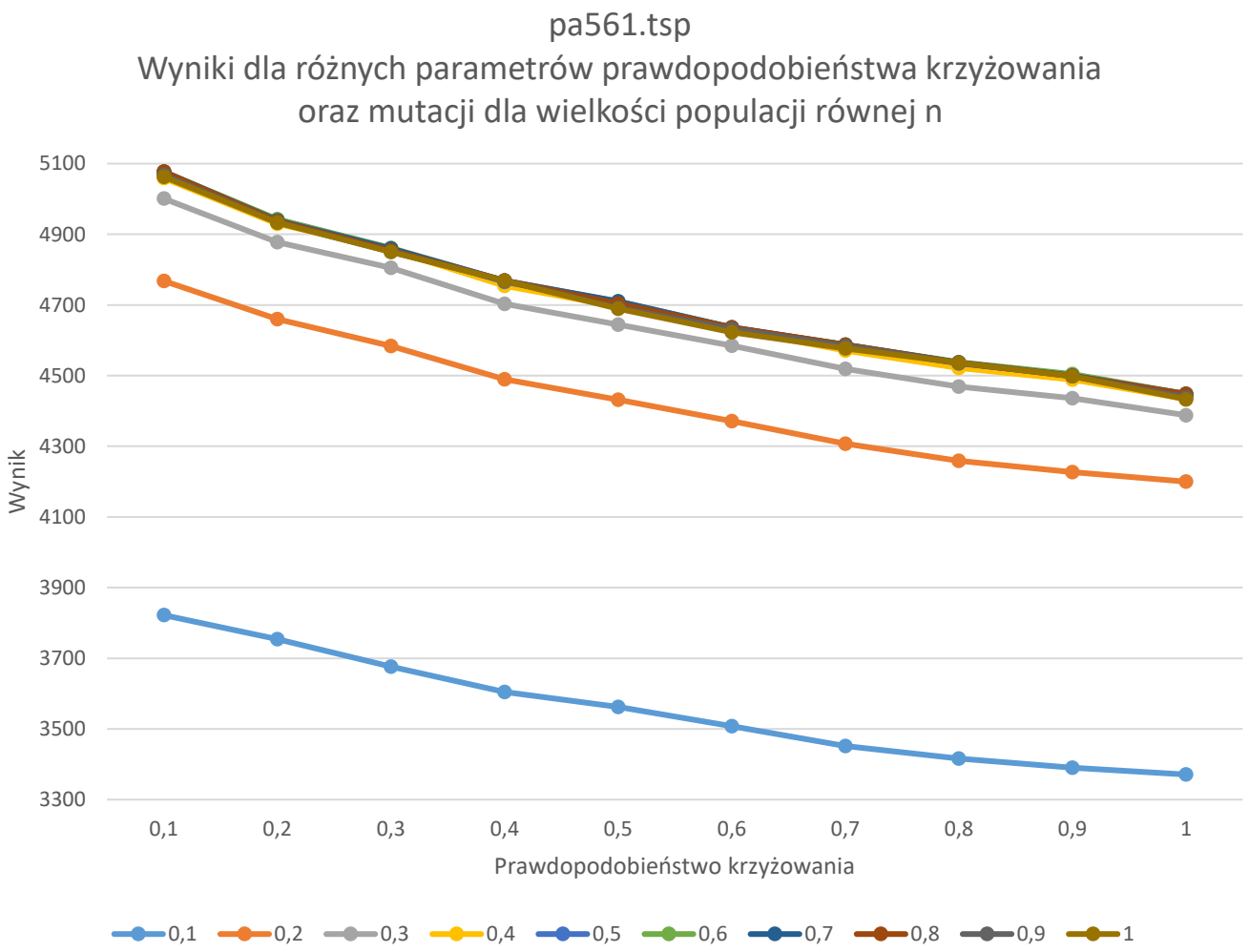
Tabela 2: Wyniki testów dla różnych wartości prawdopodobieństwa krzyżowania oraz mutacji dla rozmiaru populacji równemu n/2.



Wykres 2: Wykres uzyskanych wyników dla różnych wartości prawdopodobieństwa krzyżowania (oś OX) oraz mutacji (serie) dla rozmiaru populacji równemu n/2.

Rozmiar populacji: n											
	Prawdopodobieństwo krzyżowania										
Prawdopodobieństwo mutacji		**0,1**	**0,2**	**0,3**	**0,4**	**0,5**	**0,6**	**0,7**	**0,8**	**0,9**	**1**
	0,1	3822	3754	3676	3605	3562	3508	3452	3416	3390	3371
	0,2	4768	4660	4584	4490	4432	4371	4307	4259	4227	4200
	0,3	5001	4878	4805	4703	4644	4585	4519	4469	4436	4388
	0,4	5059	4930	4856	4754	4695	4631	4571	4522	4489	4434
	0,5	5073	4942	4860	4765	4707	4635	4584	4535	4501	4445
	0,6	5077	4943	4861	4768	4710	4636	4587	4538	4504	4448
	0,7	5077	4939	4860	4769	4710	4637	4587	4537	4500	4448
	0,8	5077	4937	4854	4769	4704	4637	4587	4536	4499	4448
	0,9	5068	4934	4851	4769	4692	4632	4584	4536	4498	4439
	1	5063	4933	4850	4769	4689	4623	4577	4536	4498	4433

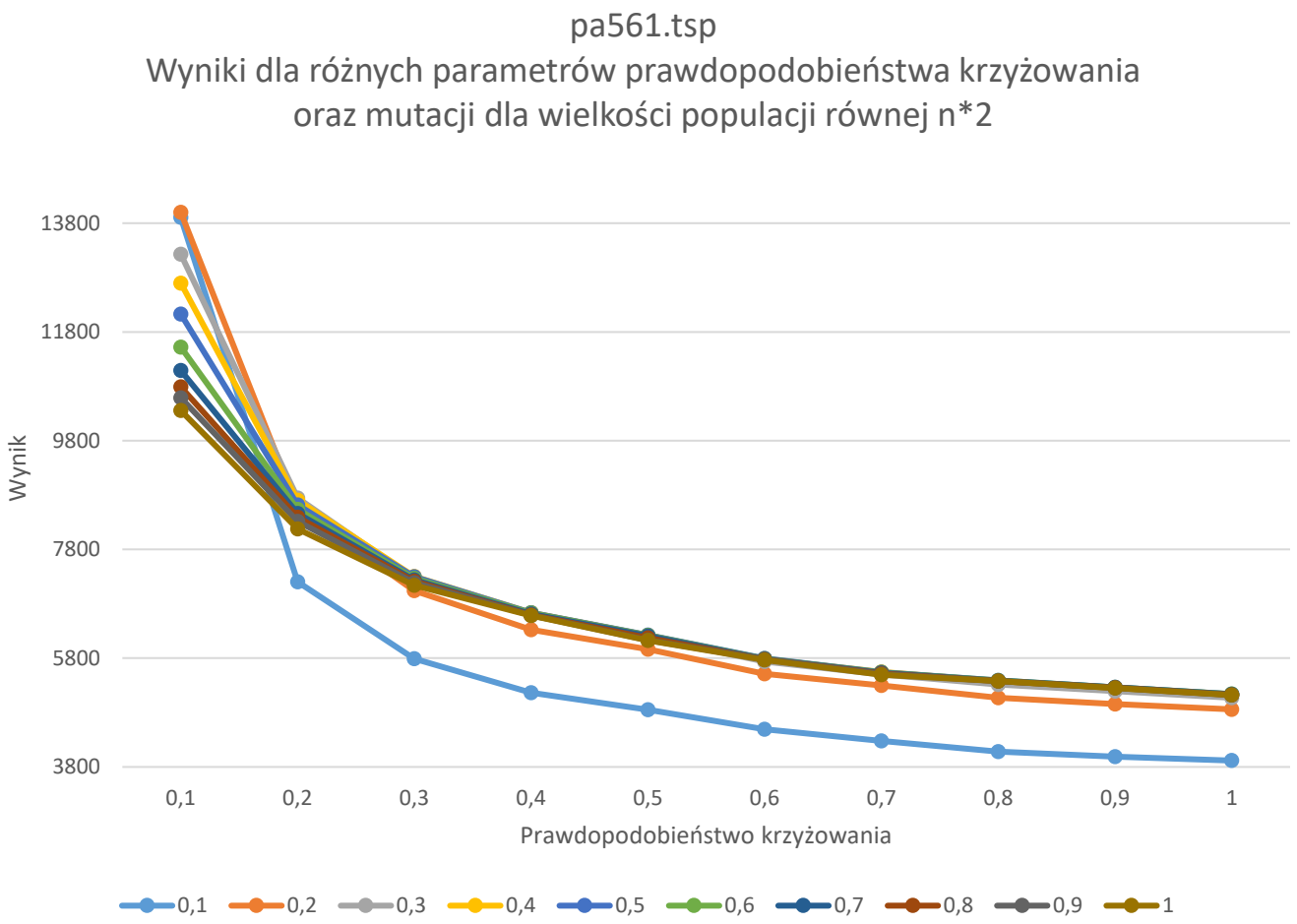
Tabela 3: Wyniki testów dla różnych wartości prawdopodobieństwa krzyżowania oraz mutacji dla rozmiaru populacji równemu n.



Wykres 3: Wykres uzyskanych wyników dla różnych wartości prawdopodobieństwa krzyżowania (oś OX) oraz mutacji (serie) dla rozmiaru populacji równemu n.

Rozmiar populacji: n*2											
		Prawdopodobieństwo krzyżowania									
Prawdopodobieństwo mutacji		**0,1**	**0,2**	**0,3**	**0,4**	**0,5**	**0,6**	**0,7**	**0,8**	**0,9**	**1**
	0,1	13907	7202	5793	5165	4853	4495	4281	4079	3991	3917
	0,2	13999	8564	7044	6322	5965	5512	5296	5075	4958	4859
	0,3	13230	8743	7263	6582	6190	5739	5504	5318	5192	5072
	0,4	12699	8707	7300	6637	6219	5787	5546	5377	5251	5126
	0,5	12129	8616	7295	6639	6223	5796	5543	5392	5262	5138
	0,6	11522	8536	7268	6629	6217	5793	5537	5393	5264	5140
	0,7	11091	8462	7241	6611	6210	5784	5528	5384	5264	5136
	0,8	10789	8391	7217	6597	6174	5782	5519	5381	5255	5126
	0,9	10589	8319	7185	6591	6134	5781	5509	5380	5251	5123
	1	10355	8180	7138	6583	6124	5770	5495	5380	5250	5122

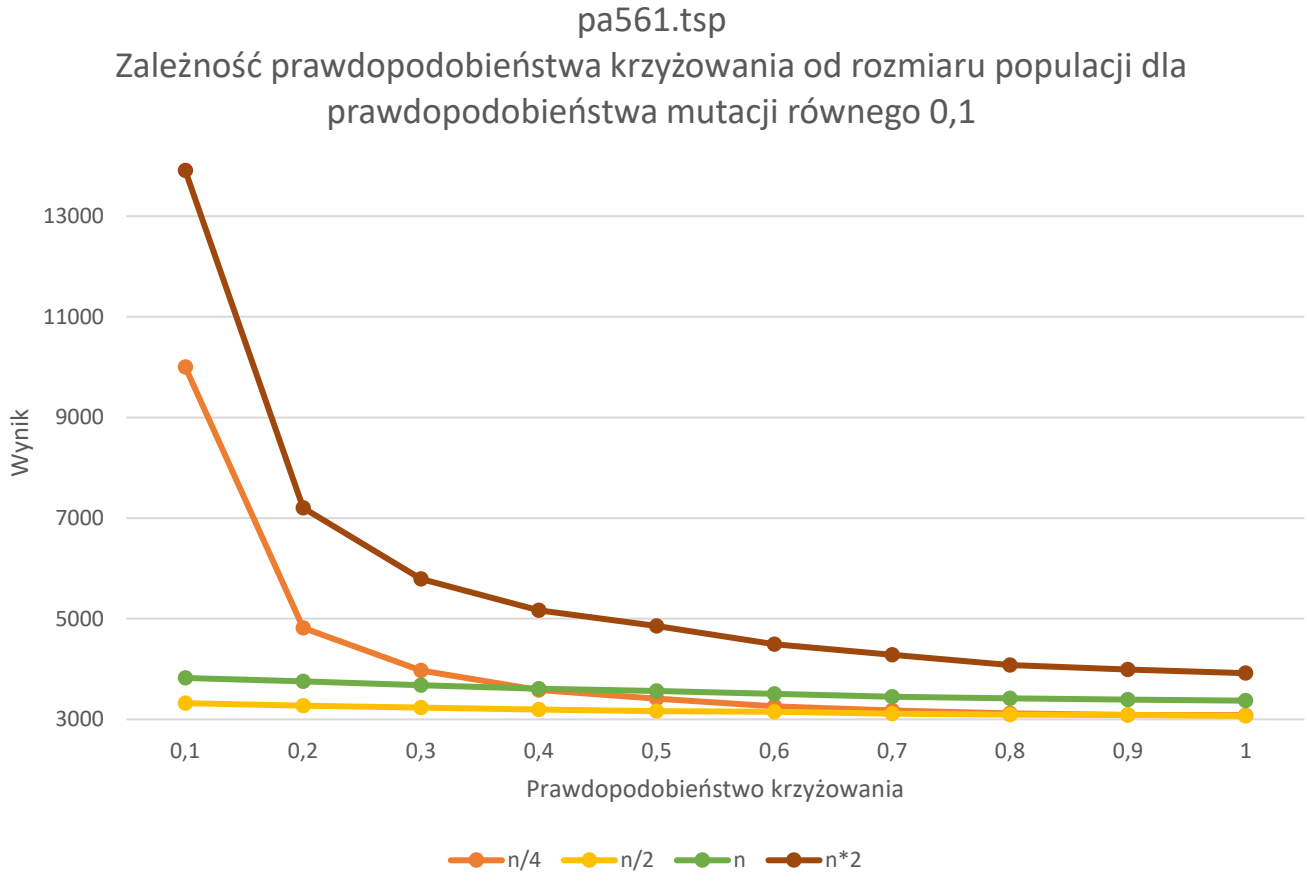
*Tabela 4: Wyniki testów dla różnych wartości prawdopodobieństwa krzyżowania oraz mutacji dla rozmiaru populacji równemu n*2.*



*Wykres 4: Wykres uzyskanych wyników dla różnych wartości prawdopodobieństwa krzyżowania (oś OX) oraz mutacji (serie) dla rozmiaru populacji równemu n*2.*

Prawdopodobieństwo mutacji: 0,1											
	Prawdopodobieństwo krzyżowania										
Wielkość populacji		**0,1**	**0,2**	**0,3**	**0,4**	**0,5**	**0,6**	**0,7**	**0,8**	**0,9**	**1**
	n/4	10004	4818	3968	3581	3412	3259	3174	3118	3088	3083
	Błąd dla n/4 (%)	262,07	74,38	43,61	29,61	23,49	17,95	14,88	12,85	11,76	11,58
	n/2	3321	3269	3230	3193	3162	3148	3113	3094	3086	3068
	Błąd dla n/2 (%)	20,20	18,31	16,90	15,56	14,44	13,93	12,67	11,98	11,69	11,04
	n	3822	3754	3676	3605	3562	3508	3452	3416	3390	3371
	Błąd dla n/2 (%)	38,33	35,87	33,04	30,47	28,92	26,96	24,94	23,63	22,69	22,01
	n*2	13907	7202	5793	5165	4853	4495	4281	4079	3991	3917
	Błąd dla n/2 (%)	403,33	160,66	109,66	86,93	75,64	62,69	54,94	47,63	44,44	41,77

Tabela 5: Porównanie wyników oraz ich błędów dla różnych wartości prawdopodobieństwa krzyżowania oraz rozmiaru populacji dla prawdopodobieństwa mutacji równego 0,1.



Wykres 5: Wykres uzyskanych wyników dla różnych wartości prawdopodobieństwa krzyżowania (oś OX) oraz rozmiaru populacji (serie) dla prawdopodobieństwa mutacji równego 0,1.

5.2. Testy pozostałych instancji

gr17.tsp

Prawdopodobieństwo mutacji: 0,1							
	Prawdopodobieństwo krzyżowania						
Wielkość populacji		**0,5**	**0,6**	**0,7**	**0,8**	**0,9**	**1**
	n/4	2085	2085	2085	2085	2085	2085
	Błąd dla n/4 (%)	0,00	0,00	0,00	0,00	0,00	0,00
	n/2	2085	2085	2085	2085	2085	2085
	Błąd dla n/2 (%)	0,00	0,00	0,00	0,00	0,00	0,00
	n	2085	2085	2085	2085	2085	2085
	Błąd dla n/2 (%)	0,00	0,00	0,00	0,00	0,00	0,00
	n*2	2085	2085	2085	2085	2085	2085
	Błąd dla n/2 (%)	0,00	0,00	0,00	0,00	0,00	0,00

Tabela 6: Porównanie wyników oraz ich błędów dla różnych wartości prawdopodobieństwa krzyżowania oraz rozmiaru populacji dla prawdopodobieństwa mutacji równego 0,1– instancja gr17.tsp.

br17.atsp

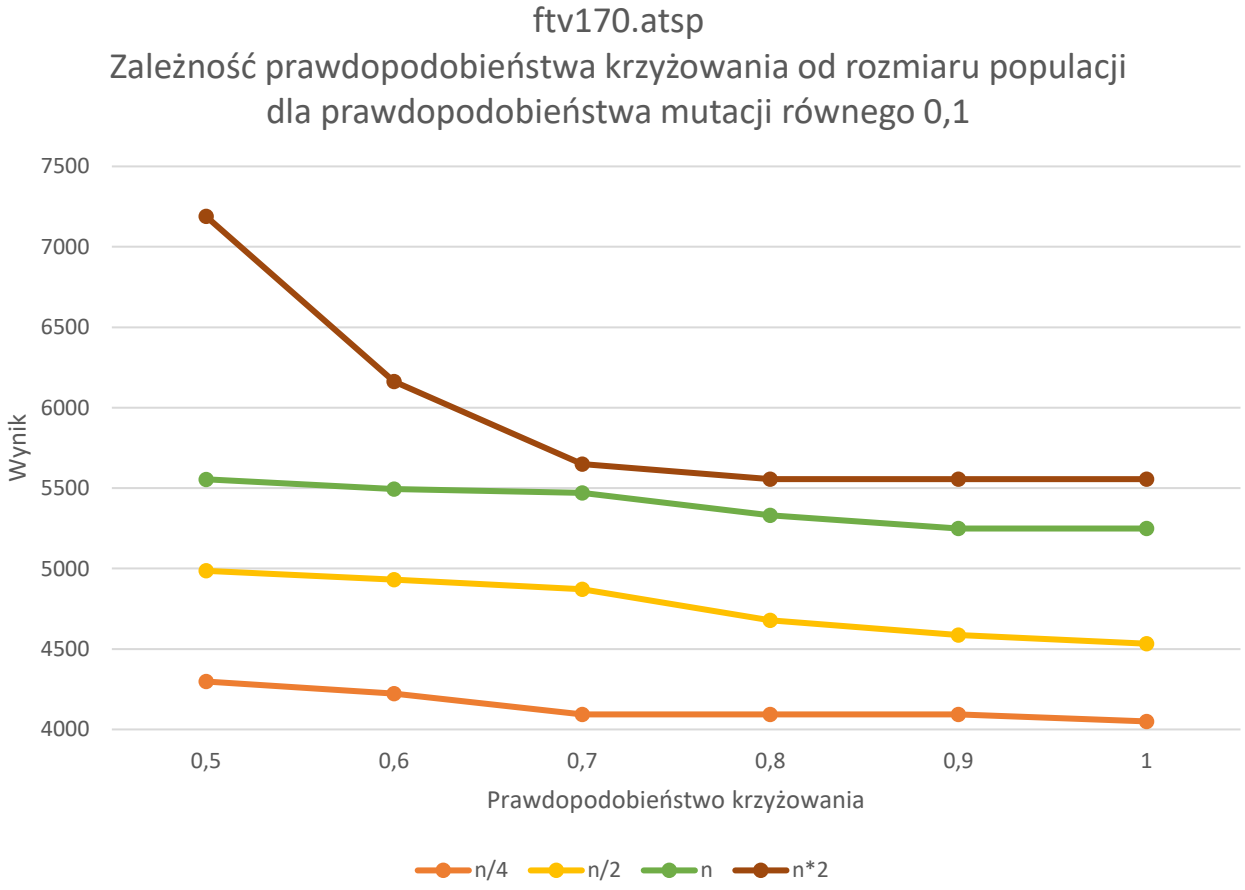
Prawdopodobieństwo mutacji: 0,1							
	Prawdopodobieństwo krzyżowania						
Wielkość populacji		**0,5**	**0,6**	**0,7**	**0,8**	**0,9**	**1**
	n/4	39	39	39	39	39	39
	Błąd dla n/4 (%)	0,00	0,00	0,00	0,00	0,00	0,00
	n/2	39	39	39	39	39	39
	Błąd dla n/2 (%)	0,00	0,00	0,00	0,00	0,00	0,00
	n	39	39	39	39	39	39
	Błąd dla n/2 (%)	0,00	0,00	0,00	0,00	0,00	0,00
	n*2	39	39	39	39	39	39
	Błąd dla n/2 (%)	0,00	0,00	0,00	0,00	0,00	0,00

Tabela 7: Porównanie wyników oraz ich błędów dla różnych wartości prawdopodobieństwa krzyżowania oraz rozmiaru populacji dla prawdopodobieństwa mutacji równego 0,1– instancja br17.atsp.

ftv170.atsp

Prawdopodobieństwo mutacji: 0,1							
	Prawdopodobieństwo krzyżowania						
Wielkość populacji		**0,5**	**0,6**	**0,7**	**0,8**	**0,9**	**1**
	n/4	4298	4223	4094	4094	4094	4050
	Błąd dla n/4 (%)	56,01	53,28	48,60	48,60	48,60	47,01
	n/2	4986	4931	4872	4678	4587	4533
	Błąd dla n/2 (%)	80,98	78,98	76,84	69,80	66,50	64,54
	n	5555	5495	5471	5332	5249	5249
	Błąd dla n/2 (%)	101,63	99,46	98,58	93,54	90,53	90,53
	n*2	7190	6163	5650	5557	5556	5556
	Błąd dla n/2 (%)	160,98	123,70	105,08	101,71	101,67	101,67

Tabela 8: Porównanie wyników oraz ich błędów dla różnych wartości prawdopodobieństwa krzyżowania oraz rozmiaru populacji dla prawdopodobieństwa mutacji równego 0,1– instancja ftv170.atsp.

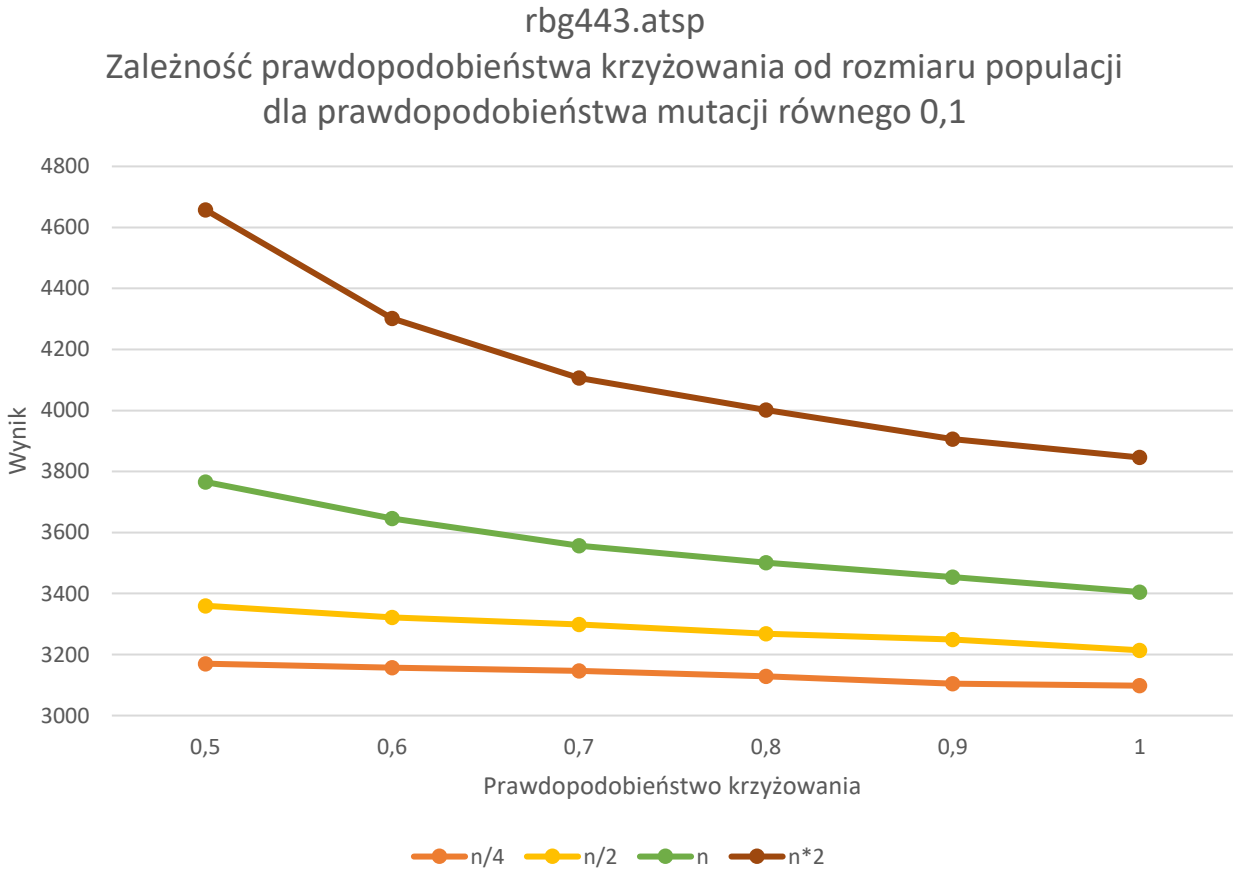


Wykres 6: Wykres uzyskanych wyników dla różnych wartości prawdopodobieństwa krzyżowania (oś OX) oraz rozmiaru populacji (serie) dla prawdopodobieństwa mutacji równego 0,1 – instancja ftv170.atsp

rbg443.atsp

Prawdopodobieństwo mutacji: 0,1							
	Prawdopodobieństwo krzyżowania						
Wielkość populacji		**0,5**	**0,6**	**0,7**	**0,8**	**0,9**	**1**
	n/4	3170	3157	3146	3129	3105	3098
	Błąd dla n/4 (%)	16,54	16,07	15,66	15,04	14,15	13,90
	n/2	3360	3322	3299	3268	3249	3214
	Błąd dla n/2 (%)	23,53	22,13	21,29	20,15	19,45	18,16
	n	3766	3646	3557	3501	3454	3405
	Błąd dla n/2 (%)	38,46	34,04	30,77	28,71	26,99	25,18
	n*2	4658	4302	4107	4002	3906	3846
	Błąd dla n/2 (%)	71,25	58,16	50,99	47,13	43,60	41,40

Tabela 9: Porównanie wyników oraz ich błędów dla różnych wartości prawdopodobieństwa krzyżowania oraz rozmiaru populacji dla prawdopodobieństwa mutacji równego 0,1– instancja rbg443.atsp.

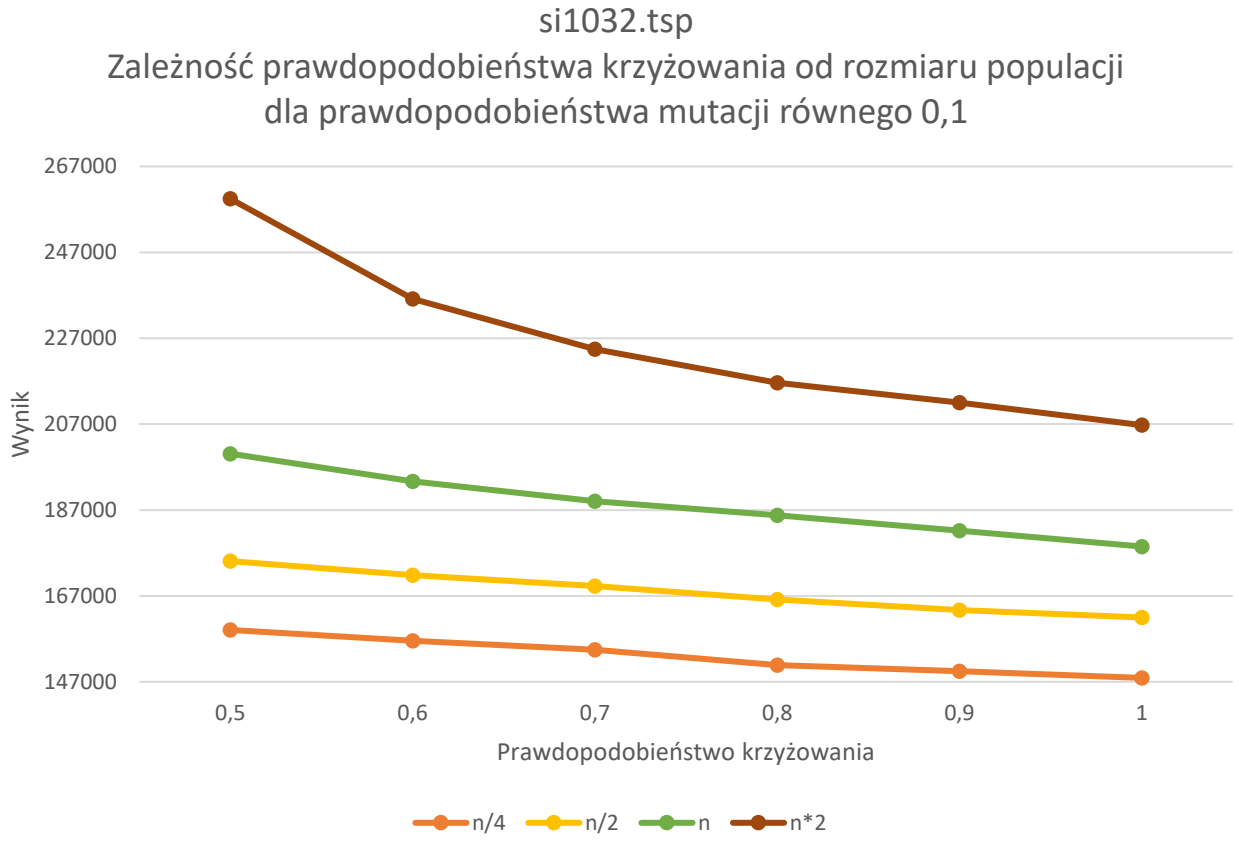


Wykres 7: Wykres uzyskanych wyników dla różnych wartości prawdopodobieństwa krzyżowania (oś OX) oraz rozmiaru populacji (serie) dla prawdopodobieństwa mutacji równego 0,1 – instancja rbg443.atsp

si1032.tsp

Prawdopodobieństwo mutacji: 0,1							
	Prawdopodobieństwo krzyżowania						
Wielkość populacji		**0,5**	**0,6**	**0,7**	**0,8**	**0,9**	**1**
	n/4	159078	156540	154435	150904	149491	147914
	Błąd dla n/4 (%)	71,70	68,96	66,69	62,88	61,35	59,65
	n/2	175098	171850	169291	166137	163727	161979
	Błąd dla n/2 (%)	88,99	85,48	82,72	79,32	76,72	74,83
	n	200115	193662	189087	185791	182183	178502
	Błąd dla n/2 (%)	115,99	109,03	104,09	100,53	96,64	92,66
	n*2	259459	236131	224478	216629	211976	206768
	Błąd dla n/2 (%)	180,04	154,86	142,29	133,81	128,79	123,17

Tabela 10: Porównanie wyników oraz ich błędów dla różnych wartości prawdopodobieństwa krzyżowania oraz rozmiaru populacji dla prawdopodobieństwa mutacji równego 0,1– instancja si1032.tsp.



Wykres 8: Wykres uzyskanych wyników dla różnych wartości prawdopodobieństwa krzyżowania (oś OX) oraz rozmiaru populacji (serie) dla prawdopodobieństwa mutacji równego 0,1 – instancja si1032.tsp

5.3. Porównanie najlepszych wyników pomiędzy algorytmem genetycznym a metodą Tabu Search.

	Tabu Search	Algorytm genetyczny
gr17.tsp	2135	2085
Błąd dla gr17 (%)	2,40	0,00
br17.tsp	39	39
Błąd dla br17 (%)	0,00	0,00
ftv170.atsp	10171	4050
Błąd dla ftv170 (%)	269,18	47,01
rbg443.atsp	4656	3098
Błąd dla rbg443 (%)	71,18	13,90
pa561.tsp	13091	3068
Błąd dla pa561 (%)	373,80	11,04
si1032.tsp	180766	147914
Błąd dla si1032 (%)	95,11	59,65

Tabela 11: Porównanie najlepszych wyników uzyskanych podczas badań w algorytmie genetycznym oraz Tabu Search dla wszystkich testowanych instancji oraz określenie ich błędów.

6. Wnioski

Na podstawie powyższych wyników widać że algorytm dla testowanych instancji daje dobre rezultaty, co wskazuje na jego prawidłową implementację. Wyniki uzyskane podczas badania instancji pa561.tsp zgodnie z teorią pokazują, że prawdopodobieństwo mutacji powinno znajdować się w okolicach 0,1, gdyż na niemal każdym wykresie widać ogromną różnicę pomiędzy Pm = 0,1, a jego większymi wartościami (wykresy 1 – 4). Można więc stwierdzić, że stosowanie większych wartości raczej nie jest opłacalne, szczególnie w przypadku tej instancji.

Również prawdopodobieństwo krzyżowania zachowuje stałą zależność. W praktycznie każdym przypadku niezależnie od testowanej instancji najlepsze wyniki wychodzą w testach, gdzie Pc jest równe 1. Mniejsze wartości dają zazwyczaj niewiele gorszy wynik, jednak te są również zależne od rozmiaru populacji oraz samej instancji. Najgorsze rezultaty widać w przypadku gdy Pc jest równe 0,1 oraz rozmiar populacji wynosił dwukrotność wielkości instancji (n*2) (wykres 4).

Co do samego rozmiaru populacji widać też, że nie może ona być wielka. Dla prawie wszystkich instancji najlepsze wyniki osiągane były dla rozmiaru populacji równemu n/4. Jedyną instancją gdzie lepsze wyniki osiągnięte zostały dla populacji o wielkości n/2 było pa561.tsp, jednak różnica ta była minimalna (tabela 5). Niestety do dokładnego określenia dolnego przedziału rozmiaru populacji, w jakim osiągane będą jeszcze lepsze wyniki wymagane by były dalsze badania, chodź na podstawie testów pa561 można stwierdzić, n/4 jest blisko tej granicy.

Z powyższych wniosków można stwierdzić, że dla wykonanej wersji algorytmu w większości przypadków sprawdzą się parametry gdzie prawdopodobieństwo krzyżowania jest równe 1, prawdopodobieństwo mutacji 0,1 a rozmiar instancji n/4. Do tych parametrów dochodzi również czas działania, który był ustawiony dla wszystkich testów na 30 sekund. Jak widać na podstawie wyników nawet w tak krótkim czasie dobrze skonstruowany algorytm może dawać dobre wyniki.

Ciekawym wynikiem eksperymentu jest fakt, że dla najmniejszych testowanych instancji, tj. gr17.tsp oraz br17.atsp dla wszystkich testowanych parametrów uzyskiwaliśmy wynik optymalny (tabela 6 i 7). Jako że czas działania ustawiony był na 30 sekund, pokazuje dużą przewagę metaheurystyki nad metodą podziału i ograniczeń, w której optymalne wyniki znajdywane były w 64 sekund dla gr17.tsp, oraz aż 16 minutami dla br17.atsp. Widać więc że algorytm generyczny równie dobrze nadaje się do dużych instancji, jak i małych i możliwe, że dla tych instancji udało by się osiągnąć optimum w jeszcze krótszym czasie. Warto jednak pamiętać, że nie mamy tutaj gwarancji optymalnego wyniku i w przypadku „złośliwych” danych nawet z najlepszym dobraniem parametrów możemy go nie osiągnąć.

W tabeli 11 przedstawione zostało porównanie najlepszych uzyskanych wyników dla poszczególnych instancji pomiędzy algorytmem genetycznym, a testowaną w poprzednim etapie projektu metodą Tabu Search. W każdym wypadku algorytm genetyczny daje lepsze wyniki (w br17 w obydwu przypadkach osiągnięto optimum) co może pokazywać przewagę algorytmu genetycznego nad TS. Przyczyn może być wiele jak chociażby sama implementacja algorytmu, jednak warto zwrócić uwagę na dwie rzeczy. Po pierwsze, w algorytmie generycznym bazujemy na populacji, czyli wykorzystujemy więcej niż jednego poszukiwacza a co za tym idzie, prowadzimy wiele przeszukań w jednym momencie, gdzie w przypadku Tabu Search używaliśmy tylko jednego osobnika. Po drugie, dobór parametrów w algorytmie generycznym wydaje się bardziej intuicyjny. Rezultaty dla wszystkich testowanych parametrów w każdej instancji są bardzo podobne, gdzie w Tabu Search ten sam zestaw w przypadku jednej instancji mógł dawać najlepsze wyniki, w przypadku innej dawał jedne ze słabszych. Ta niespójność powoduje że przeszukiwanie z zakazami wymaga dokładniejszego dostrajania parametrów algorytmu żeby otrzymać dobre wyniki dla każdej instancji osobno.