Józef Melańczuk i Artur Trześniewski

Algorytm Genetyczny

Spis treści

1. **Opis algorytmu i krótka notka o złożoności obliczeniowej**
   Napisany przez nas algorytm genetyczny, jest standardowy i podąża klasycznym schematem:

   1. Wybierz populację początkową
   2. Wybierz rodziców do stworzenia nowej generacji
   3. Przeprowadź krzyżowanie na rodzicach
   4. Z pewnym prawdopodobieństwem mutuj osobników
   5. Przeprowadź selekcję do następnej generacji
   6. Jeżeli nie zostało przekroczone ograniczenie wróć do punktu 2

   Oczywiście pomijam tutaj np. Mechanizmy przeciwdziałania stagnacji, gdyż nie zmieniają one podstawowej struktury algorytmu. Każdy z tych kroków może być wykonywany na różne sposoby(nie licząc kroku 6). Wykorzystuje się do tego tzw. Operatory, które dla danego algorytmu są ujednolicone strukturą i można je wymienić jak „klocki" bez konieczności zmian w kodzie. W naszym przypadku wykorzystaliśmy następujące operatory, o następującej złożoności obliczeniowej. Uwaga: w niektórych miejscach konieczne jest wylosowanie jakiejś liczby, nie będącej pewną konkretną wartością. Przykład, niekoniecznie z naszego kodu: nie chcielibyśmy wybierać najlepszego osobnika z jakichś dwóch losowo wybranych, jeżeli w tej dwójce byłby ten sam osobnik. Zakładamy, pomimo, że ktoś mógłby się do tego przyczepić, że taki proces jest <u>zawsze</u> O(1).

   Używana notacja:
   p_{size} – rozmiar populacji
   c_{size}– liczba rodziców
   m_{size}– liczba osobników które mutują
   n – rozmiar instancji
   Operatory generowania polucji początkowej:
   - Random population – O($p_{size} * n$)
   - K-means Clustering – O($\sqrt{n}$ *2-opt($\sqrt{n}$)) + O($p_{size} * \sqrt{n}$)
   Opeartory selekcji rodziców:
   - Random selection – O(c_{size})
   - Weighted selection – O($c_{size} * p_{size}$)
   - Rouletee wheel selection – O($c_{size} * p_{size}$)
   - Tournament selection – O(c_{size})
   Operatory krzyżowania:
   - Swap – O(c_{size})
   - Partial mapping – O($c_{size} * n$)
   - Order – O($c_{size} * n$)

- One point – O(c_size)

Operatory mutowania:

- Swap – O(m_{size})
- Reverse – O(m_{size})
- IRGIBNNM – O($m_{size} * n$)

Opearatory selekcji następnej generacji:

- Replace old gen – O(1)
- Selecting best – O($sort(p_{size} * 2 * c_{size})$)
- Tournament selection – O(p_{size})

Przeciwdziałanie stagnacji:

By przeciwdziałać stagnacji, przechowujemy listę elitarnych osobników na pewnej ustalonej przestrzeni generacji, a w razie potrzeby wybieramy z niej losowo osobników których wdrażamy do naszej populacji, a następnie również losowo pewnej liczbie osobników robimy sieczkę z trasy. Możliwym jest także wymienianie w takim przypadku operatorów na inne, za pomocą których być może znajdziemy rozwiązanie które pchnie nas na przód

2. **Środowisko testowe**

Maszyna: Procesor: Intel(R) Core(TM) i7-4700MQ CPU, liczba rdzeni fizycznych: 4, logicznych: 8 RAM: 8GB

System operacyjny: Windows 10, 64-bit

Język: Programy do testowania zostały napisane w języku Julia, używającym kompilatora LLVM.

Umówiliśmy się, że testy napiszemy osobno, ale użyjemy jednej maszyny by je wykonywać, by ujednolicić wyniki. Warto wspomnieć, że testy były pisane w „natywnej" Julii, tj. Nie wykorzystywaliśmy np. Funkcji z języka Python, co mogłoby wpłynąć na jakość rezultatów.

3. Algorytm odniesienia

Jako algorytm odniesienia użyliśmy algorytmu z czasopisma „Annals of Operations research"[1]. Wybraliśmy wersję opisaną tam jako „prostą", gdyż chcieliśmy zbadać wpływ poszczególnych elementów na wyniki. Nie chcieliśmy by sam algorytm był tak dobry(Albo co gorsza przeciętny), by zaburzało to klarowność wyników. Algorytm ten ma strukturą opisaną na początku punktu 1. Operatory których używa:

Generowanie populacji początkowej – Random population
Wybieranie rodziców – Roulette Wheel Selection
Krzyżowanie – One point
Mutacja – Swap

[1] https://www.inf.tu-dresden.de/content/institutes/ki/cl/study/summer14/pssai/slides/GA_for_TSP.pdf

Wybieranie następnej generacji – Replace old gen

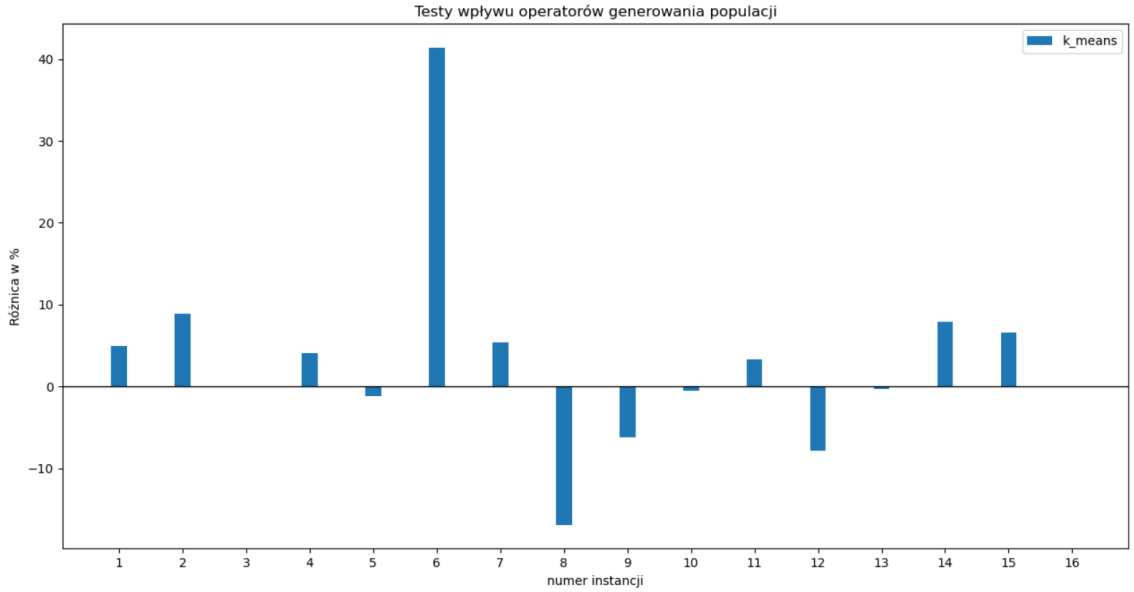
4. **Badania wpływu operatorów na jakość działania**

Metodologia:
Modyfikując operatory po kolei, sprawdzaliśmy ich wpływ na jakość działania algorytmu. Modyfikowaliśmy jeden parametr na raz, gdyż chcieliśmy sprawdzić wpływ danego etapu na wynik końcowy, a także wpływ innych operatorów danego etapu. Następnie badając kolejny parametr, używaliśmy z powrotem bazowego operatora, z alogrytmu odniesienia
Instancje:
Nasz algorytm badaliśmy na następujących instancjach:
Symetrycznych:
1. kroB100.tsp
2. kroA100.tsp
3. gr17.tsp
4. eil101.tsp
5. eil51.tsp
6. a280.tsp
7. lin105.tsp
8. berlin52.tsp

Asymetrycznych:
1. ry48p.atsp
2. p43.atsp
3. kro124p.atsp
4. ftv64.atsp
5. ftv44.atsp
6. ftv35.atsp
7. ftv33.atsp
8 .br17.atsp

We wszystkich badaniach używane były funkcje i metody działające prawidłowo dla danego typu instancji, czyli np. Nie używanie akceleracji przy funkcji reverse w przypadku instancji asymetrycznych.

Wpływ operatora generowania populacji początkowej:
Niestety mamy tylko jeden dodatkowy operator generowania populacji początkowej(teoretycznie dwa, ale drugi sam produkuje zbyt dobre wyniki dla małych instancji, co zaburzało badania. Operator ten 2-opta dla całych przypadkowych instancji, w przeciwieństwie do operatora k-means, który dzieli instancje na odcinki długości pierwiastka rozmiaru instancji, na nich wykonuje 2-opta, a następnie losowo rozpina i składa w jedno rozwiązanie całość). Badania dały następujące wyniki:
Gdzie wartości dodatnie znaczą poprawę, a ujemne wynik gorszy



(Odnośnik do listy miast i tego jakie numery im odpowiadają)

Tutaj nie widzimy jakiejś znaczącej przewagi tego operatora nad przypadkową populacją. Wynika to z tego, że ten operator działa lepiej dla większych instancji, gdzie grupy są większe. Dla tak małych instancji mamy grupy maksymalnie pod 10 miast. a280(nr 6) to wyjątek, w którym widać znacznie lepsze rezultaty. Po wykonaniu 2-opta na grupie, w momencie gdy rozpinamy przypadkowe wewnętrzne krawędzie, to dla grup małych rozmiarów wprowadzamy bardzo dużą zmianę, więc wynik nie jest aż tak znaczący. Jest to operator przeznaczony raczej dla instancji mających po co najmniej 100 miast miast wzwyż, co widać po znaczącym skoku jakości dla instancji mającej 280 miast.

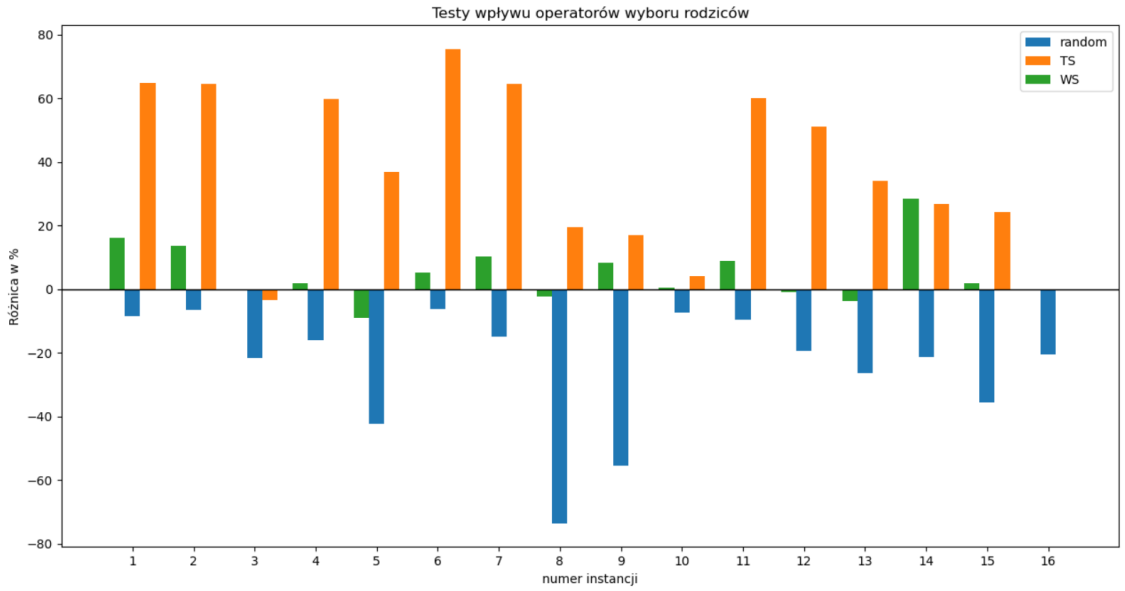
Sprawdźmy teraz statystyczną istotność tych różnic, testem wilcoxsona(test Friendmana tutaj nie ma zastosowania):

```
Exact Wilcoxon signed rank test
-------------------------------
Population details:
    parameter of interest:   Location parameter (pseudomedian)
    value under h_0:         0
    point estimate:          -48.0
    95% confidence interval: (-4955.0, 32.0)

Test summary:
    outcome with 95% confidence: fail to reject h_0
    two-sided p-value:           0.1726

Details:
    number of observations:      16
    Wilcoxon rank-sum statistic: 30.0
    rank sums:                   [30.0, 75.0]
    adjustment for ties:         0.0
```

Nie wykazał on by te statystyczne się różniły, ale jak wspomniałem, jestem pewien,

że to kwestia małych instancji

Wpływ operator wyboru rodziców:
Tutaj już mamy więcej operatorów, więc będzie można coś więcej pokombinować z wynikami. Sam wykres różnic prezentuje się następująco:



(Odnośnik do listy miast i tego jakie numery im odpowiadają)

Tutaj widzimy, że Najlepiej wypadł operator „Tournament selection”, najgorzej random selection, a porównywalnie dobrze „weighted selection”. Weighted selection bazuje na fitnessie danego osobnika. Tego samego parametru używa operator w algorytmie odniesienia, czyli „Roulette wheel selection”, stąd mniejsze różnice.
Sprawdźmy teraz jak wygląda test wilcoxsona dla tych operatorów. Jednakże jako, że jest ich całkiem sporo, to ograniczę się tylko do przedstawienia wyników, a dokładne rezultaty testu dołączę wraz ze sprawozdaniem. Mamy więc:

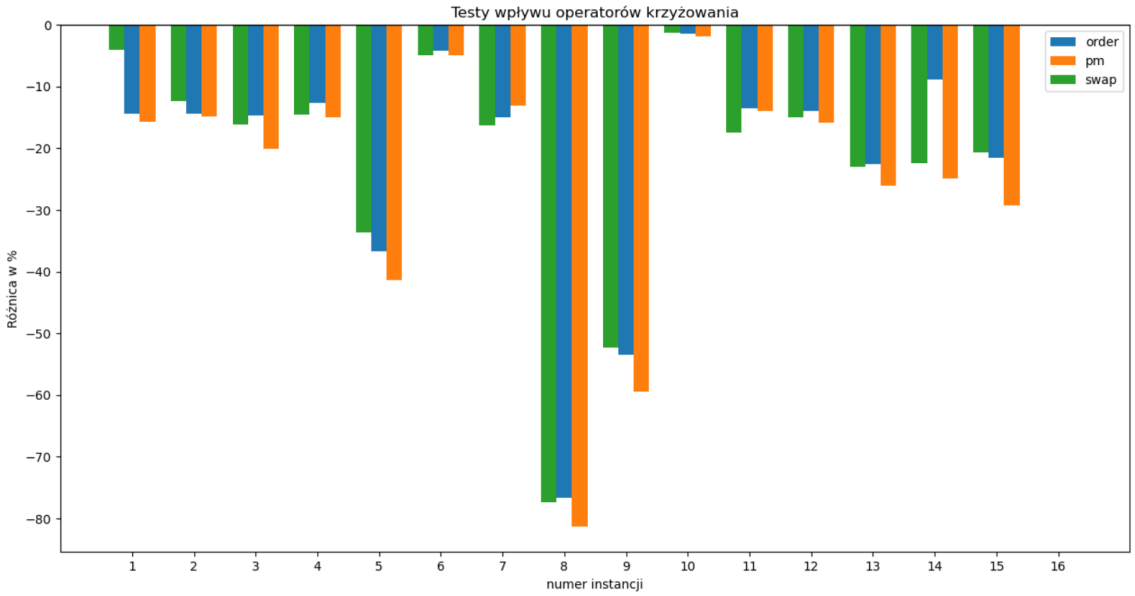
Operator	Odniesienie	Tournament	Weighted	Random
Odniesienie	X	Różnią się	Różnią się	Różnią się
Tournament	Różnią się	X	Różnią się	Różnią się
Weighted	Różnią się	Różnią się	X	Różnią się
Random	Różnią się	Różnią się	Różnią się	X

Jak widzimy, wszystkie operatory są od siebie różne. Ponadto przeprowadzimy test Friedmana dla tych operatorów, używając pythonowej bilbioteki „scipy”, który dał następujące wyniki:

statistic=39.03870967741932, pvalue=1.7031990368000346e-08

co oznacza, że operatory mają wpływ na jakość tego algorytmu, a te istotnie są od siebie różne

Wpływ operatorów Krzyżowania
Tutaj tak samo mamy kilka operatorów. Ich badania dały następujące wyniki:



(Odnośnik do listy miast i tego jakie numery im odpowiadają)

Okazuje się, że one point to całkiem niezły operator i pozostałe w żadnym przypadku nie dały lepszych wyników.
Sprawdźmy, czy test wilcoxona wykaże to samo(Ponownie, za dużo testów, by wstawiać wszystkie, więc dokładne są dołączone do sprawozdania, a tutaj tylko rezultaty):

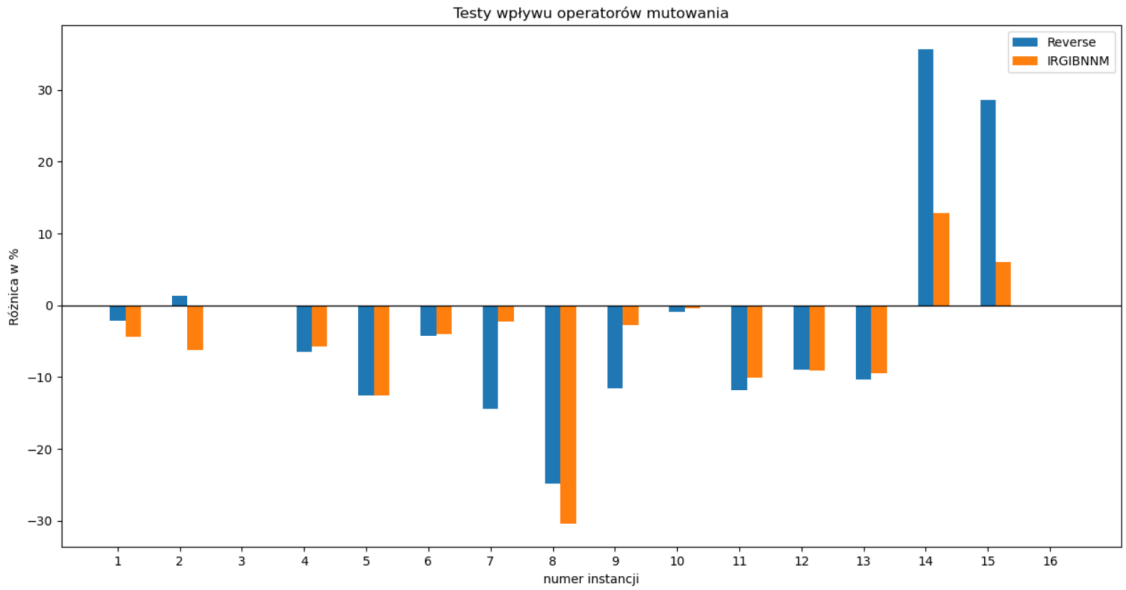
Operator	Odniesienia	Order	Partial Mapping	Swap
Odniesienia	X	Różnią się	Różnią się	Różnią się
Order	Różnią się	X	Różnią się	Nie różnią się
Partial Mapping	Różnią się	Różnią się	X	Nie różnią się
Swap	Różnią się	Nie różnią się	Nie różnią się	X

Tutaj operator Swap i Partial Mapping, oraz Swap i Order nie wykazały różnic. Wbrew pozorom nie zachodzi tutaj przechodniość i to że Swap nie wykazał istotnych różnic nie znaczy, że te nie wykazują takich między sobą
Test Friedmana dla tych Operatorów:

statistic=35.95999999999997, pvalue=7.635625061635886e-08

Ponownie wykazał on, że te operatory rzeczywiście się różnią.

Wpływ operatorów mutacji
Badania nad operatorami mutacji dały następujące rezultaty:



(Odnośnik do listy miast i tego jakie numery im odpowiadają)

Tutaj ponownie, standardowy operator mutacji tj. swap wydaje się być lepszy od innych. Sprawdźmy czy testy statystyczne wykażą to samo(Ponownie, dokładniejsze wydruki dołączam do sprawodzania):

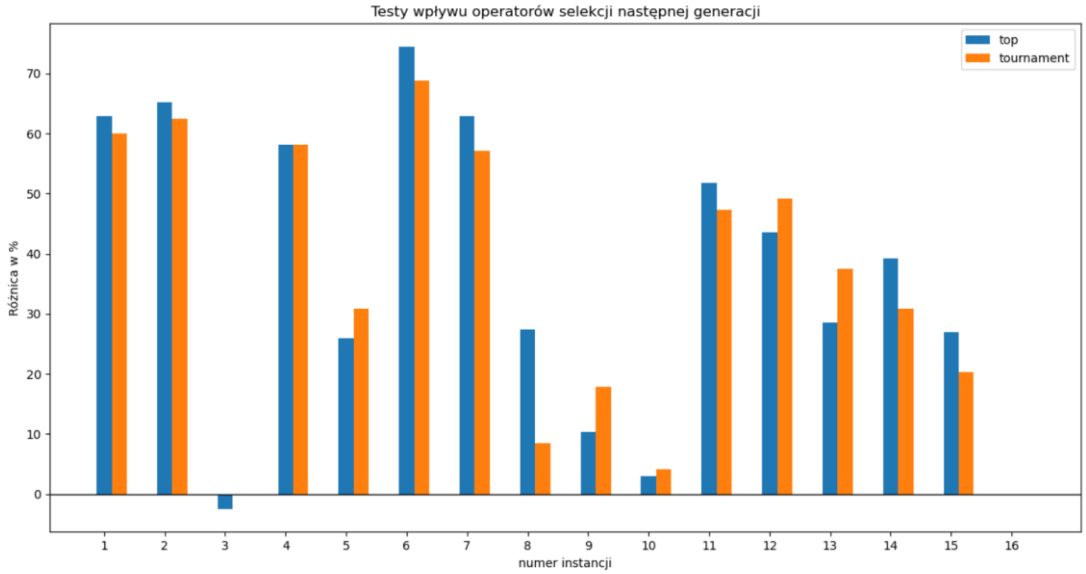
Operator	Odniesienia	Reverse	IRGIBNNM
Odniesienia	X	Nie różnią się	Różnią się
Reverse	Nie różnią się	X	Nie różnią się
IRGIBNNM	Różnią się	Nie różnią się	X

Tutaj ponownie należy uważać, żeby nie użyć przechodniości do porównania Operatorów odniesienia i IRGIBNNM, bo tak jak wcześniej wspomniałem, taka nie zachodzi.
Test Friedmana dla tych operatorów:

statistic=8.836363636363636, pvalue=0.012056132610413476

Tutaj p-value jest znacznie większe niż w poprzednich przypadkach, ale wciąż mniejsze niż 0.05 co oznacza, że odrzucamy hipotezę.

Wpływ operatorów wyboru następnej generacji:
Badania nad operatorami wyboru następnej generacji dały następujące rezultaty:



(Odnośnik do listy miast i tego jakie numery im odpowiadają)

Tym razem mamy znaczącą przewagę innych operatorów nad tym bazowym. One wybierają znacznie lepszych osobników do następnej generacji, podczas gdy bazowy operator po prostu zastępuję generację następną, nawet jeżeli jest gorsza.
Testy wilcoxona dla tych operatorów(Ponownie dokładniejsze dołączam do sprawozdania):

Operator	Odniesienia	Tournament	Select top
Odniesienia	X	Różnią się	Różnią się
Tournament	Różnią się	X	Nie różnią się
Select top	Różnią się	Nie różnią się	X

A test Friedmana:

statistic=18.93103448275862, pvalue=7.747794510626722e-05

A więc również odrzucił hipotezę zerową

Jak widzimy operatory mają istotny wpływ na jakość działania algorytmu, stąd w następnym punkcie spróbujemy znaleźć najelpsze kombinacje tych parametrów.

5. **Najlepsze kombinacje parametrów**
   Żeby znaleźć najlepsze kombinacje, po prostu testowaliśmy jakie dają wyniki, wykonując odpowiednio dużą liczbę próbek, tutaj jest to 80 losowych kombinacji dla każdej z instancji. Następnie przeanalizowaliśmy rezultaty i otrzymaliśmy następujące wyniki:

Operator -> Instancja	Populacji Początkowej	Wyboru Rodziców	Krzyżowania	Mutacji	Selekcji Następnej generacji	Wynik
a280	random	tournament	order	reverse	Top	2990.0
berlin52	K-means	random	order	swap	top	7764.0
br17	random	random	swap	reverse	top	39.0
eil101	random	roulette	order	IRGIBNNM	Top	651.0
eil51	random	roulette	order	swap	Top	437.0
Ftv33	K-means	weighted	order	swap	Top	1362.0
Ftv35	K-menas	roulette	order	IRGIBNNM	Top	1484.0
Ftv44	random	tournament	order	IRGIBNNM	tournament	1691.0
Ftv64	random	roulette	order	Swap	tournament	2046.0
Gr17	random	weighted	pm	swap	tournament	2085.0
Kro124p	Random	tournament	order	reverse	top	44664.0
kroA100	random	weighted	order	reverse	Top	22582.0
kroB100	k-Means	weighted	order	IRGIBNNM	Top	22529.0
Lin105	k-Means	weighted	pm	reverse	Top	15061.0
P43	random	roulette	swap	IRGIBNNM	tournament	5622.0
Ry48p	random	random	swap	IRGIBNNM	top	15031.0
Dominujący:	Random(11)	brak	Order(11)	IRGIBNNM(6)	Top(12)	

Więc tak prezentują się najlepsze parametry, oczywiście dla testowanych instancji.

Porównajmy teraz wyniki szukania najlepszych operatorów, metodą testowania losowych kombinacji, oraz testowaniem wpływu każdego z kolei, tj. sprawdźmy jak mają się badania z punktu 4, do tych powyżej.

Najlepsza kombinacja z punktu 4:
Generowanie populacji – k-means
Wybór rodziców – Tournament
Krzyżowanie – One point
Mutacja – Swap
Wybór do następnej generacji – Top

Najlepsza kombinacja z punktu 5:
Generowanie populacji – random
Wybór rodziców – weighted selection(tam był remis z ruletką)
Krzyżowanie – Order
Mutacja – IRGIBNNM
Wybór do następnej generacji – Top

Wykonaliśmy po 10 testów dla każdej instancji i oto wyniki:

Algorytm -> Instancja	Z punktu 4	Z punktu 5
Berlin52	10548	7960
P43	5695	5624
A280	6751	3121
Ftv64	3106	2083
Br17	39	39
Gr17	2092	2085
kroB100	40657	23367
Kro124p	59545	43794
Ry48p	19958	15010
Ftv44	2338	1728
Ftv33	1742	1388
Eil51	561	438
Ftv35	2070	1543
kroA100	42838	22409
Lin105	29691	15456
Eil101	1017	667

Ewidentnie widać przewagę tego drugiego, test Wilcoxona jest tutaj czystą formalnością:

```
Exact Wilcoxon signed rank test
-------------------------------
Population details:
    parameter of interest:   Location parameter (pseudomedian)
    value under h_0:         0
    point estimate:          816.5
    95% confidence interval: (440.0, 8950.0)

Test summary:
    outcome with 95% confidence: reject h_0
    two-sided p-value:           <1e-04

Details:
    number of observations:      16
    Wilcoxon rank-sum statistic: 120.0
    rank sums:                   [120.0, 0.0]
    adjustment for ties:         0.0
```

Widzimy więc, że nasze badania wykazały, że metoda badania wpływu pojedynczych parametrów, a następnie wybierania na ich podstawie najlepszych parametrów, jest mocno zawodna. Wynika to zapewne z faktu, że pewne operatory mogą współgrać z jednymi bardzo dobrze, a z innymi słabo, co powoduje zakłamanie wyników, gdy przyglądamy się tylko jednemu z nich. Niemniej jednak, mogą one stanowić solidną podstawę, do wykluczenia niektórych z operatorów, gdy te radzą sobie ewidentnie bardzo słabo.

6. **Najlepsze rozmiary populacji, liczba rodziców, prawdopodobieństwo mutacji**
   Przeprowadzane testy badające operatory, były prowadzone na wielkości populacji równej dwukrotności rozmiaru instancji i rodziców równej połowie tej liczby. Jako że parametrów jest znacznie mniej niż zakres liczb możliwych populacji, to postanowiliśmy najpierw doświadczalnie wyznaczyć, sensowne parametry populacji,
   następnie wyznaczyć najlepsze operatory i z ich pomocą wyznaczyć najlepsze parametry(trochę to pokrętne wiem, ale dało sensowne wyniki).
   Jako, że wykluczyłem ze spektrum wartości których użyłem w badaniach operatorów, to wyniki prezentują się następująco:

Wartości -> Instancja	p_size	c_size	$m(jaka\ część\ populacji\ mutuje)$	Wynik
A280	1081	29868	0.0124	9572.0
Berlin52	1049	539	0.011	7542.0
Br17	280	82	0.045	39.0
Eil101	652	167	0.012	654.0

Eil51	2588	517	0.0014	436.0
Ftv33	874	466	0.034	1286.0
Ftv35	812	343	0.026	1487.0
Ftv44	360	175	0.0076	1719.0
Ftv64	2757	1108	0.022	1947.0
Gr17	102	133	0.021	2085.0
Kro124p	3362	2206	0.042	38948.0
KroA100	1966	2915	0.03	21825.0
kroB100	4255	1404	0.017	22552.0
Lin105	831	4686	0.0077	14757.0
P43	496	336	0.04	5621.0
Ry48p	1830	850	0.029	15069.0

Co zabawne, w niektórych przypadkach liczba rodziców wybieranych do tworzenia nowych osobników przekracza rozmiar populacji, ale najwyraźniej nie przeszkadzało to mu w osiągnięciu lepszych wyników. Porównując tę tabelę z naszymi parametrami do badań, widzimy, że zazwyczaj radzimy sobie tak samo albo i lepiej niż te wyliczone tutaj będące, najlepszymi wynikami spośród dosyć sporej liczby próbek jakie wykonaliśmy, ale co widać przy każdej to to że najlepszy jest raczej mały, nieprzekraczający 0.05, współczynnik mutacji.

7. **Wielowątkowa implementacja wyspowa**
   Nasza implementacja wielowątkowa posiada w sobie trochę losowości. Na początku tworzymy elitarną wyspę, której operatory są z góry ustalone. Następnie uzupełniamy resztę wątków o wyspy, z losowymi operatorami. W założeniu, elitarna wyspa która jest najbardziej stabilna ma na początku produkować najlepsze rezultaty. Następnie po ustalonej ilości generacji(włączając w to pierwszą), każda wyspa kopiuje swoje najlepsze osobniki do wspólnej tablicy tablic najlepszych osobników. W przypadku stagnacji którejś z wysp, ta sięga do tej tablicy i wdraża elitarne osobniki z innej wyspy do swojej populacji. Robi to po procesie selekcji osobników do następnej generacji, by te na pewno przeszły do dalszej generacji. Następnie, jeżeli nie jest to elitarna wyspa, ten zmienia używane operatory na losowe. Następnie, już w każdym przypadku, losuje określoną liczbę osobników, której robi całkowity bajzel z ścieżki, ten zabieg ma na celu delikatne cofnięcie populacji w rozwoju. Algorytm w założeniu, ma tworzyć poprzez elitarną wyspę stabilny wynik domyślny, a poprzez zastosowanie modelu wymiany pomiędzy osobnikami, w przypadku stagnacji, rozważać ciekawe rozwiązania z innych wysp, podczas gdy inne wyspy robią to samo z jej rozwiązaniami. Wyniki są bardzo zadowalające, szczególnie, że algorytm przez nienajlepszą optymalizacji

tworzy znacznie mniej generacji niż normalny, a i tak daje z tą znacznie mniejszą ilością, porównywalne i lepsze wyniki od zwykłego.
Wyniki tego algorytmu dla poszczególnych instancji:

Instancja	Wynik
A280	3099
Berlin52	7542
Br17	39
Eil51	432
Eil101	658
Ftv33	1286
Ftv35	1492
Ftv44	1697
Ftv64	1965
Gr17	2085
Kro124p	40669
Kro100A	21709
Kro100B	22537
Lin105	14379
P43	5622
Ry48p	14733

Jak już wspominałem, jest to słabo zoptymalizowana wersja. Sam algorytm jest całkowicie osobnym bytem, wymagającym innych struktur i innych rozwiązań, niż zwykły algorytm genetyczny, ale wyniki które pokazał w tym samym czasie, pomimo znacznie mniejszej liczby generacji są bardzo obiecujące

8. **Wnioski i porównanie z Tabu searchem**

   Algorytm genetyczny jest trochę innym bytem niż przeszukiwanie tabu, w założeniu przeczesuje on znacznie większą przestrzeń rozwiązań, często poruszając się w różnych otoczeniach, jak to ma na przykład miejsce w przypadku implementacji wyspowej. Na tak małych instancjach nasz algorytm nie miał szans z algorytmem tabu search. Ten bardzo szybko znajdował optimum w instancjach w których nam zajmowało to kilkanaście-kilkadziesiąt sekund. Sam algorytm genetyczny jest według mnie bardzo ciekawy. Ale przewagą samego algorytmu genetycznego jest jego niebywała uniwersalność. Dobra implementacja pozwala wymieniać operatory i funkcję jak klocki, przez co tego samego algorytmu można by używać do wielu problemów, zmieniając jedynie podawane operatory i funkcję, co uważam za bardzo dużo zaletę i przewagę nad algorytmem tabu, w którym taka implementacja byłaby bardzo trudna i niosła by za sobą pewnie spadki w wydajności.