Algorytmy Metaheurystyczne
Komiwojażer Genetycznie

Gabriel Budziński(254609)
Franciszek Stepek (256310)

Przedmowa

Na samym początku omówimy po krótce naszą implementacje, oraz podamy kilka informacji ogólnych. Następnie bardziej szczegółowo opiszemy poszczególne parametry allgorytmu, a na koniec przedstawimy wyniki i opis wykonywanych eksperymentów.

1 Informacje ogólne

1.1 Implementacja

Algorytmy implementujemy w języku `C/C++`, odległości między wierzchołkami są przechowywane jako pełne tablice dwuwymiarowe typu `int`, a trasy (pojedyncze osobniki) są w kontenerach `vector`, co ułatwia operacje odwracania i mieszania. Korzystaliśmy z kompilatora g++ wraz z użyciem flag -lSDL2 (używanej przy wizualizacji, wraz z odpowiednim dla danego systemu operacyjnego podlinkowania do folderu zawierającego) oraz -lpthread (przy korzystaniu z wielowątkowości).

Dodajmy jeszcze tylko, że jako generatora pseudolosowego użyliśmy typu `std::mt19937` zdefiniowanego przez `C++`.

1.2 Sprzęt

Programy były testowane na dwóch maszynach, laptopie *Lenovo* i komputerze stacjonarnym. Obie jednostki są wyposażone w procesor architektury `x86` marki `intel` oraz 16GB pamięci RAM.

- PC - Komputer stacjonarny posiada procesor sześciordzeniowy i5-10600K 4,1 GHz (o obniżonym napięciu operacyjnym).
- Laptop - Laptop posiada procesor czterordzeniowy i7-6700HQ 2,6 GHz

1.3 Instancje

Używaliśmy instancji przygotowanych przez TSPLIB, które dzielą się na 2 kategorie:

- 8 instancji symetrycznych:
  - berlin52.tsp
  - st70.tsp
  - eil76.tsp
  - bier127.tsp
  - kroA150.tsp
  - lin318.tsp
  - linhp318.tsp
  - pr439.tsp
- 8 instancji asymetrycznych:
  - ftv55.atsp
  - ftv64.atsp

- ftv70.atsp
- kro124p.atsp
- ftv170.atsp
- rbg323.atsp
- rbg358.atsp
- rbg443.atsp

W dalszych częściach, instancje będziemy oznaczać przez liczbę mówiącą o rozmiarze problemu (czyli np. st70.tsp oznaczymy jako $n = 70$, albo po prostu 70) z drobną różnicą - aby rozróżnić lin318 od linhp318, instancję linhp318 będziemy onaczać liczbą 319.

1.4 Metodologia/cel

Testy przeprowadzono za pomocą zaimplementowanych w tym celu funkcji ku jak największej automatyzacji. Dane o przeprowadzonych testach zapisywano do plików tekstowych w formacie CSV, a następnie poddane analizie. Testy i eksperymenty miały na celu zbadanie wydajności naszej implementacji, oraz znalezienie jak najbardziej optymalnych trybów/hiperparametrów dla przypadku ogólnego.

2 Opis parametrów

W opisie przejdziemy najpierw przez kolejne 'tryby' działania, a następnie omówimy także każdy hiperparametr występujący w naszej implementacji, ale zanim, to wspomnijmy jeszcze tylko, że każda operacja krzyżowania daje nam 2 nowe osobniki.

2.1 Tryby działania

- StartMode - sposób generowania populacji początkowej:
  - 0 - Każdy osobnik jest wybierany z 10 całkowicie losowych (Chodzi o losowe ermutacje dróg)
  - 1 - Każdy osobnik jest tworzny jako puszczenie algorytmu NearestNeighbor (czyli zachłanne szukanie najbliższego sąsiada z tych co pozostali w każdej iteracji) z losowego punktu startowego
  - 2 - Hybrydowe połączenie 2 poprzednich, gdzie stosunek losowych do NN wynosi 4:1 (Czyli około 20% populacji to osobniki 'względnie dobre').
- SelectionMode - sposób w jaki jest wykonywana selekcja osobników:
  - 0 - Turniejowa, czyli wyieramy najlepszych, a najgorszych odrzucamy
  - 1 - Kwadratowo ruletkowa - najlepszy osobnik przechodzi dalej, a wszystkim pozostałym przyporządkowywane są wagi względem kwadratu pozycji (Czyli jeżeli mamy populację wielkości 15, to najlepszy przechodzi dalej, kolejny ma wagę $14 * 14$, później $13 * 13$ itd., a ostatni ma wagę 1), a następnie zgodnie z nimi jest robione losowanie.
- MutMode - sposób przeprowadzenia mutacji (o jej hiperparametrach będzie później):
  - 0 - Mutacja typu *Invert*
  - 1 - Mutacja typu *Insert*
  - 2 - Mutacja typu *Swap*
  - 3 - W każdej iteracji (co to oznacza będzie powiedziane później) losowe wybranie spośród 3 poprzednich
- crossMode - używany operator do krzyżowania osobników:
  - 0 - *Order Based Crossover*
  - 1 - *Modified Order Based Crossover*
  - 2 - *Partially Mapped Crossover*
- crossType - sposób przeprowadzenia i selekcji osobników do krzyżowania:
  - 0 - Wszystkie osobniki są ustawione losowo, a następnie krzyżujemy ze sobą 1 z 2, 3 z 4.. itd. Jeżeli osobników było nieparzyście, to ostatni osobnik jest dublowany.
  - 1 - W każdej iteracji losowana jest para osobników z całej populacji (Ustalona liczba na sztywno, w testach = 20)
  - 2 - Tak jak poprzednio, ale tym razem liczba losowań jest określona jako rozmiar problemu / 2 (czyli dla $n = 150$ mamy 75 losowań.

2.2 Hiperparametry

- time - czas działania algorytmu - w naszych eksperymentach każde 1 wywołanie trwa 30 sekund
- populationSize - rozmiar populacji początkowej (oraz co za tym idzie - rozmiar w każdej iteracji, ponieważ selekcja redukuje rozmiar do rozmiaru początkowego)
- mutationThreshold - określa z jakim prawdopodobieństwem zachodzi mutacja (mutacja może zajść podczas tworzenia nowych osobników, rozpatrywana dla każdego z osobna)
- mutationIntensification - górne ograniczenie na liczbę pojedynczych mutacji na jednym osobniku (jeżeli zajdzie mutacja, to następnie jest losowana jej intensyfikacja, co najmniej 1, definiuje liczbę iteracji przy mutacji - dlatego przy zastosowaniu MutMode3 może się okazać, że np. wykonają się 3 typu *Invert* oraz 1 typu *Swap*)
- crossSize - wielkość fragmentu podlegająca krzyżowaniu - przy naszych operatorach jest to wielkość 'wycinka', który definiuje operację krzyżowania
- crossCount - wykorzystywane tylko, gdy crossType 1, definiuje liczbę zachodzących krzyżowań

3 Opis eksperymentów

Jako wyniki eksperymentów będziemy pokazywali wartość funkcji celu dla podanych wykonań (Zazwyczaj jako minimum, oraz średnią z 4 wywołań), w formie stosunku wartości względem najlepszej znalezionej (informacja ze strony TSPLIB). Dodatkowo czasami będziemy mówili również o liczbie wykonanych iteracji, aby zastanowić się później nad (ewentualnym) jej wpływem na ostateczny wynik.

3.1 Poszukiwanie I - Tryb

Na samym początku przeprowadzony został eksperyment, który miał na celu znalezienie jak najlepszego zestawu trybów dla naszego algorytmu. Użyliśmy tutaj następującego zestawu hiperparametrów (wybranych empirycznie po kilkunastu przetestowaniach algorytmu):

- populationSize = 20
- mutationThreshold = 0.05
- mutationIntensification = 5
- crossSize = 7
- crossCount = 20

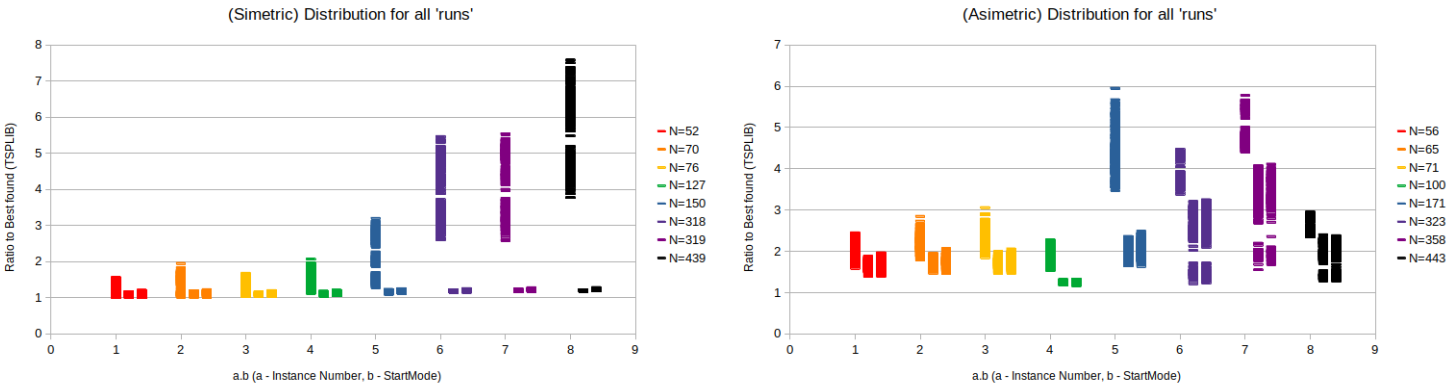
Testy wykonaliśmy dla każdej możliwej kombinacji trybów (3*3*2*4*3 = 216), dla każdej z 16 badanych instancji.
Przypomnijmy, że czas działania ograniczyliśmy do 30 sekund, a żeby odrobinę zredukować losowy wkład metody, każde wywołanie powtórzyliśmy 4 razy.
Pokazanie wszystkich wyników w tabeli byłoby dosyć kłoptliwe i nieczytelne, więc ograniczymy się do kilku mniejszych tabel oraz paru wykresów.

3.1.1 Ogólny rozrzut wyników

Na początek przedstawimy wykresy pokazujące rozrzut danych dla testów. Na osi Y wartością jest stosunek $\frac{a}{b}$, gdzie:

- a - wartość funkcji celu dla pojedynczego eksperymentu
- b - najlepsze znalezione rozwiązanie (wg. TSPLIB)

Dodatkowo na wykresie zawarta jest też informacja o trybie startującym, dlatego też dla instancji 'a' dane na osi X są na 3 'poziomach': a.0, a.1, a.2, gdzie kolejno oznacza to StartMode.

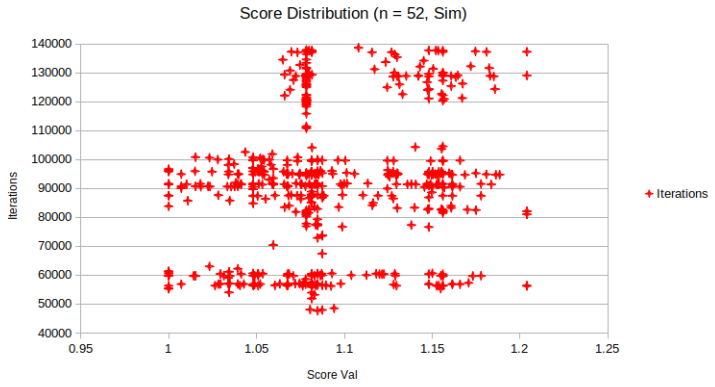


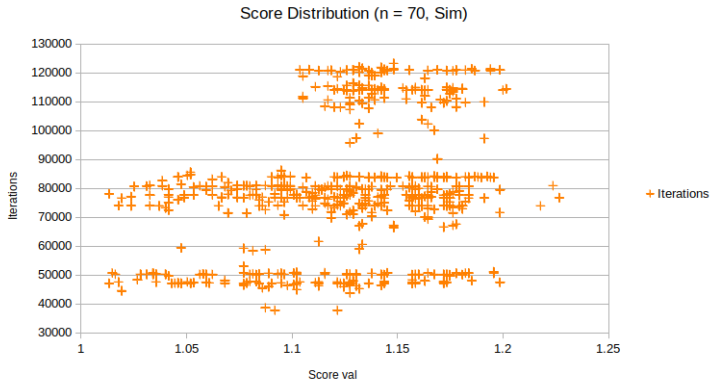
Z podanych wykresów można wyciągnąć dwa zasadnicze wnioski (które aplikują się zarówno do instancji symetrycznych, jak i asymetrycznych):

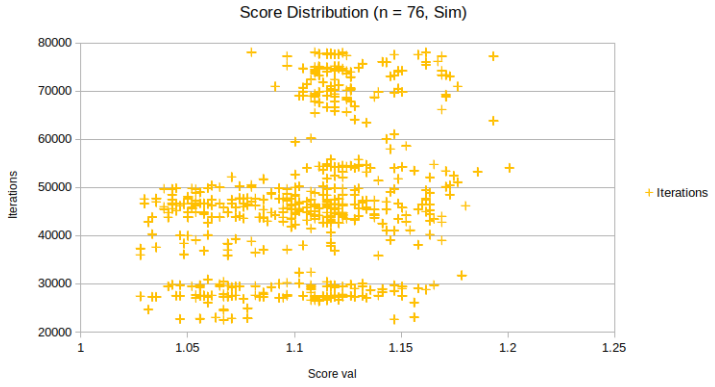
- Jak widać, dla większych instancji algorytm startujący z rozwiązań losowych nie miał wystarczającej ilości czasu, aby dotrzeć do rozwiązań (w sensie wydajności) zbliżonych do pozostałych trybów startowania. Jednak dla instancji mniejszego rozmiaru, z samego tylko wykresu trudno byłoby powiedzieć, czy jego rozwiązania są rzeczywiście gorsze, chociaż jasno widać, że mają one (losowy start) znacznie większy rozrzut.
- Dla większych instancji, pomimo dużego rozrzutu danych, dla samych rozwiązań startujących z całkowicie losowej populacji można wyznaczyć dwie 'grupy', daltego też zaryzykujemy tu swierdzenie, że dla pozostałych trybów startu także to zachodzi (chociaż na razie tego nie widać)

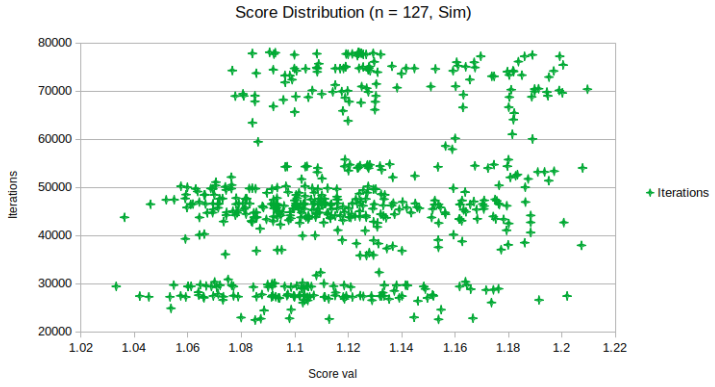
3.1.2 Rozrzut wyników dla poszczególnych instancji

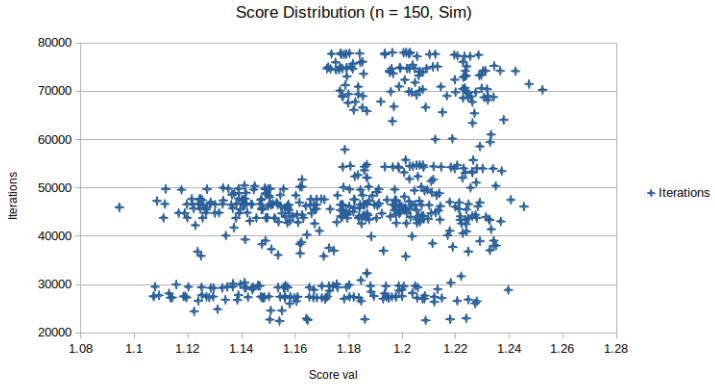
Jeszcze więcej wykresików!!!
Pokażemy teraz dystrybucję wyniku względem liczby iteracji osobno dla każdej instancji. Jednak, nie zawrzemy tutaj wyników, które zostały otrzymane przez użycie całkowicie losowej populacji początkowej, ponieważ tak jak zauważyliśmy na poprzednich wykresach, odbiegają znacznie od pozostałych 2 trybów startujących.

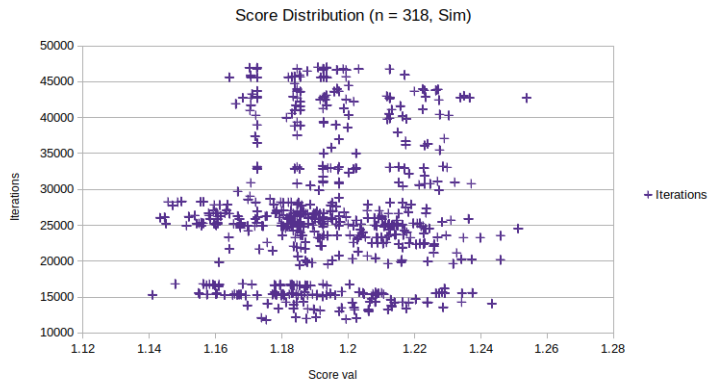


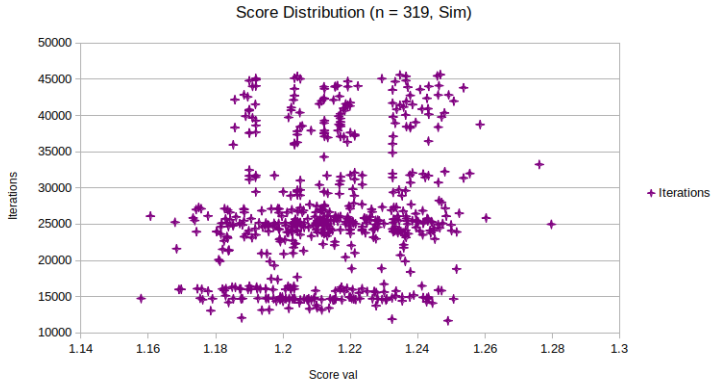


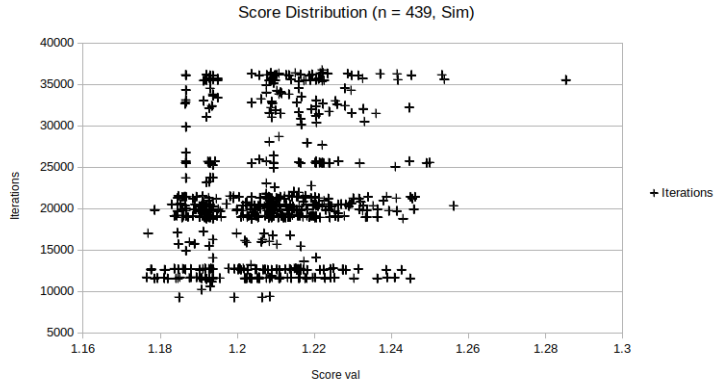


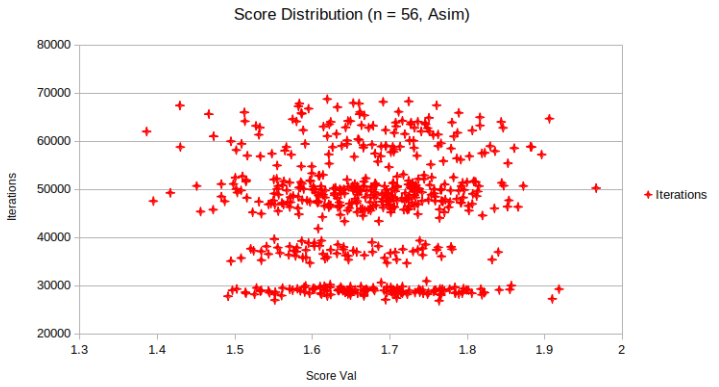


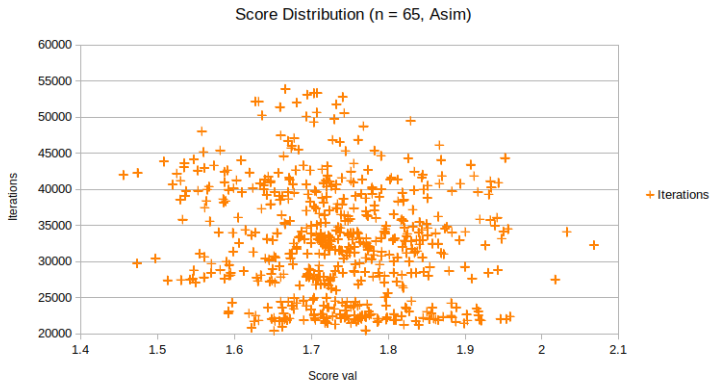


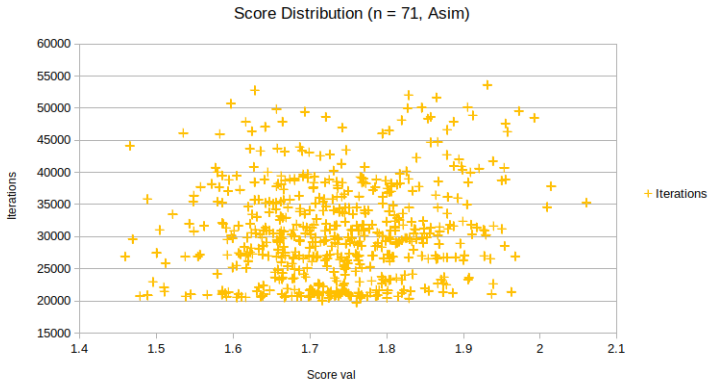


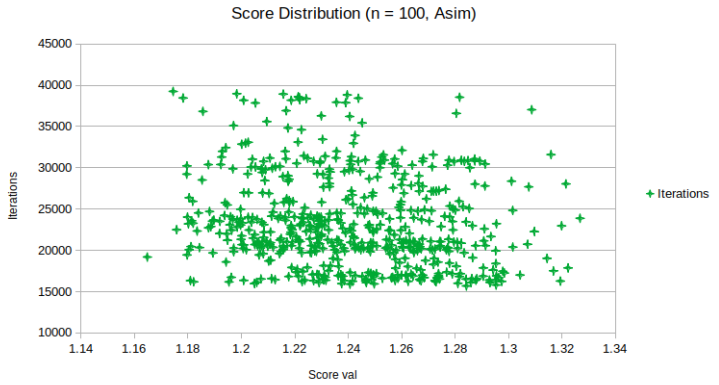


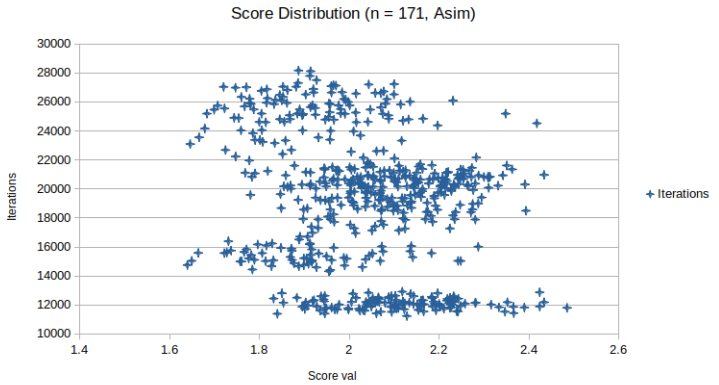


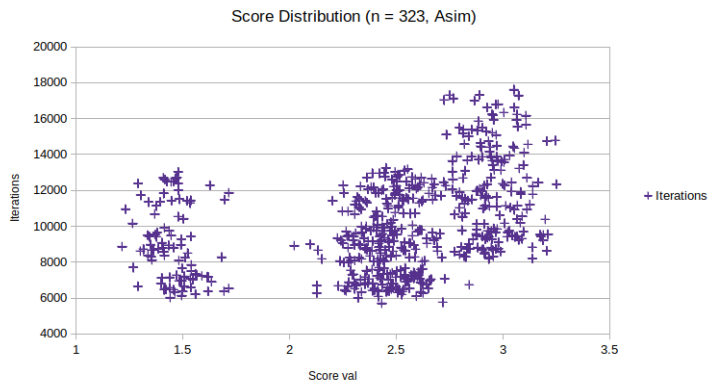


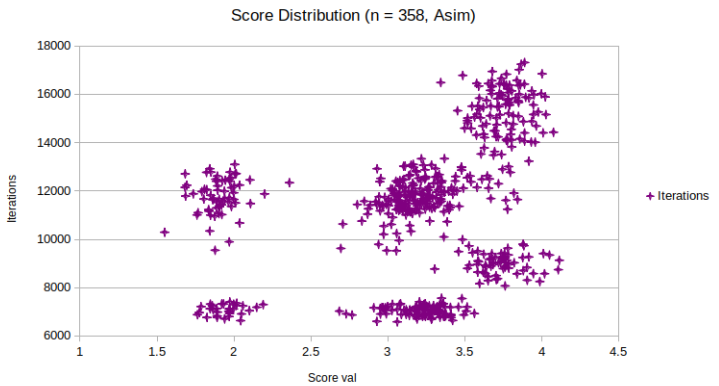


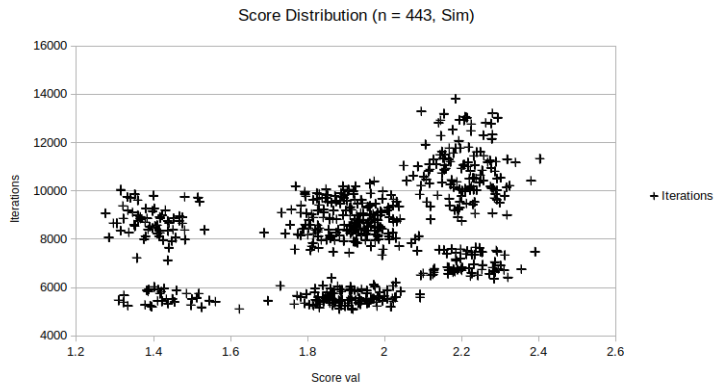












Z tych wykresów wyciągnijmy jeszcze kilka wniosków:

- Wniosek 1
- Wniosek 2
- Wniosek 3
- Wniosek 4

- Wniosek 5
- Wniosek 6

3.1.3 Tryby pod względem jednostkowym

Pokażemy teraz tabele podsumowującą wszystkie instancje symetryczne, oraz asymetryczne, w której to zawrzemy informacje o minimum i średniej rozwiązań w rozbiciu o każdy tryb (to znaczy, że będziemy patrzeć na całą naszą przestrzeń rozwiązań, a następnie będziemy ją dzielić na rozwiązania względem danego trybu, zatem najpierw (w pierwszych 3 wierszach) podzielimy całość na 3 metody startowe, następnie całość podzielimy na 2 metody selekcji, itd.), z czego wyciągniemy informacje, jakie wartośći danych trybów są ogółem najlepsze gdy się nie bierze pod uwagę kombinacji innych.

Tabela 1.

Tryb	Wartość trybu	Min (Sym)	Avg (Sym)	Min (Asym)	Avg (Asym)
StartMode	0	1	2.751	1.533	2.988
	1	1	1.144	1.175	1.981
	2	1	1.165	1.164	2.008
SelectionMode	0	1	1.682	1.174	2.321
	1	1	1.691	1.164	2.331
MutMode	0	1.020	1.562	1.164	2.275
	1	1	1.736	1.179	2.338
	2	1.067	1.851	1.174	2.336
	3	1	1.597	1.175	2.330
crossMode	0	1.065	1.819	1.174	2.419
	1	1	1.616	1.180	2.277
	2	1	1.625	1.164	2.281
crossType	0	1	1.697	1.178	2.326
	1	1	1.671	1.164	2.324
	2	1	1.692	1.174	2.328

Z obu powyższych tabel możemy wyciągnąć następujące wnioski:

- Wniosek 1
- Wniosek 2
- Wniosek 3

3.1.4 10 najlepszych kombinacji

Teraz spójrzmy jeszcze na ranking 10 najlepszych zestawów parametrów zebranych ze wszystkich instancji.
Ranking stworzony został w następujący sposób:

- W obrębie każdej instancji:
  - Najpierw policzyliśmy minimum oraz średnią z 4 wywołań każdej kombinacji
  - Później zsumowaliśmy ze sobą obie otrzymane wartości
  - Następnie posortowaliśmy kombinacje rosnąco względem otrzymanej sumy
  - Teraz przyporządkowaliśmy miejsca do otrzymanego porządku (przy dublowaniu któregoś z miejsc omijaliśmy kolejne: 32, 32, 34)
- W obrębie typu instancji (symetryczne / asymetryczne), a później globalnie (wszystkie 16 razem)
  - Zsumowaliśmy zajęte miejsca dla każdej kombinacji
  - posortowaliśmy rosnąco otrzymując ostateczny 'ranking' dla wszystkich kombinacji w obrębie danego typu.

Oto 'główki' dla otrzymanych wyników:
Z powyższych 3 tabel możemy wyciągnąć następujące wnioski:

- Wniosek 1
- Wniosek 2
- Wniosek 3

Tabela 2. Sym.

Id	StartMode	SelectionMode	MutMode	CrossMode	CrossType	Sum	Avg
107	1	0	3	2	2	98	12.25
102	1	0	3	1	0	124	15.5
106	1	0	3	2	1	127	15.875
103	1	0	3	1	1	141	17.625
142	1	1	3	2	1	161	20.125
139	1	1	3	1	1	163	20.375
140	1	1	3	1	2	188	23.5
143	1	1	3	2	2	197	24.625
138	1	1	3	1	0	202	25.25
104	1	0	3	1	2	206	25.75

Tabela 3. Asym.

Id	StartMode	SelectionMode	MutMode	CrossMode	CrossType	Sum	Avg
77	1	0	0	1	2	193	24.125
75	1	0	0	1	0	225	28.125
149	2	0	0	1	2	227	28.375
76	1	0	0	1	1	250	31.25
115	1	1	0	2	1	267	33.375
114	1	1	0	2	0	270	33.75
112	1	1	0	1	1	274	34.25
185	2	1	0	1	2	276	34.5
80	1	0	0	2	2	278	34.75
113	1	1	0	1	2	289	36.125

Tabela 4. Wspólne

Id	StartMode	SelectionMode	MutMode	CrossMode	CrossType	Sum	Avg
76	1	0	0	1	1	461	28.8125
75	1	0	0	1	0	471	29.4375
77	1	0	0	1	2	494	30.875
115	1	1	0	2	1	494	30.875
113	1	1	0	1	2	522	32.625
112	1	1	0	1	1	536	33.5
114	1	1	0	2	0	546	34.125
103	1	0	3	1	1	579	36.1875
116	1	1	0	2	2	590	36.875
111	1	1	0	1	0	596	37.25

3.1.5 Ostatecznie wybrane najlepsze

Udało się nam teraz wyłonić potencjalnie najlepsze kombinacje trybów (które będziemy stosować później):

- Dla wariantu symetrycznego:
- Dla wariantu asymetrycznego:
- Globalnie:

3.2 Poszukiwanie II - hiperparametry

Po wyznaczeniu rokujących zestawów trybów, przeszliśmy do wyznaczenia jak najlepszych hiperparametrów. W tym celu wyznaczyliśmy 1 zestaw dla wariantu symetrycznego (Spec), 1 dla asymetrycznego (Spec), oraz 1 wspólny (Gen).
Poprzez metodę losowego próbkowania dla każdego hiperparametru (losowy z zakresu podanego zaraz), wykonaliśmy 100 kombinacji, gdzie każdą testowaliśmy 4 razy.
Badane instancje:

- st70.tsp - sym.
- pr439.tsp - sym.
- ftv70.atsp - asym.
- rbg443.atsp - asym.

Badany zakres hiperparametrów:

- populationSize : $[10; 100]$
- mutationThreshold : $[0.0; 1.0]$
- mutationIntensification : $[1; 20]$
- crossSize : $[2; 20]$
- crossCount : $[10; 200]$

Z przeprowadzonych eksperymentów otrzymaliśmy następujące wyniki:

3.2.1 Rozrzut wyników dla poszczególnych instancji

Na samym początku zaprezentujemy na wykresach, jak wygląda rozkład otrzymanych wyników dla każdej z badanych instancji.

Z powyższych wykresów spróbujmy teraz wyciągnąć kilka wniosków:

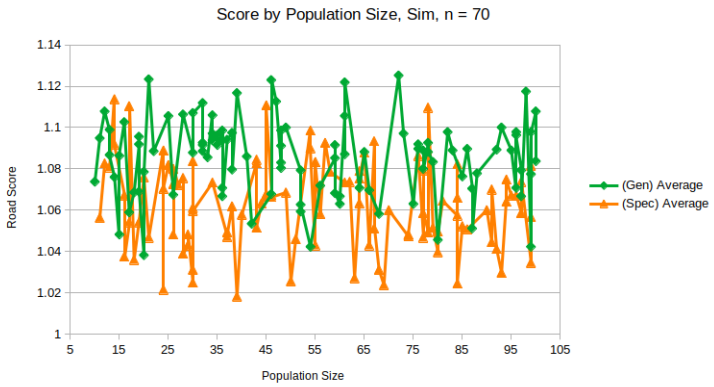
- Jak widać dla wariantu symetrycznego, dla rozmiaru wielkości 70 o wiele lepiej zadziałał zestaw trybów deykowany do tych instancji, jednakże dla rozmiaru 439, o wiele lepiej zachował się zestaw ogólny. Przez sformułowanie 'lepiej' rozumiemy tutaj wartości częściej osiągane bliższej wartości rozwiązania najlepszego.

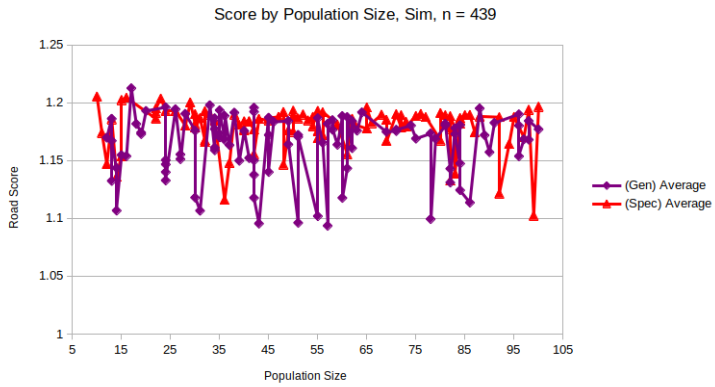
- Dla wariantu symetrycznego widać również, że dla każdej instancji zestaw danych ogólny wykonywał mniej iteracji (koncentracja w 'dolnej' części wykresu). Może być to powodowane tym, że jest on uwarunkowany po części parametrem *crossCount*, który to mówi o liczbie wykonywanych krzyżowań - widzimy tutaj zatem 'średnią' zależność tego parametru od wykonanej liczby iteracji.
- Wniosek 3
- Ogólny wniosek mamy zatem taki (i powtórzymy go zapewne przy okazji dalszej analizy): dla naszej implementacji różne zestawy trybów będą się inaczej zachowywały dla każdej instancji, zatem najlepszą metodą byłby dobór wszystkich trybów i parametrów do każdej instancji z osobna.

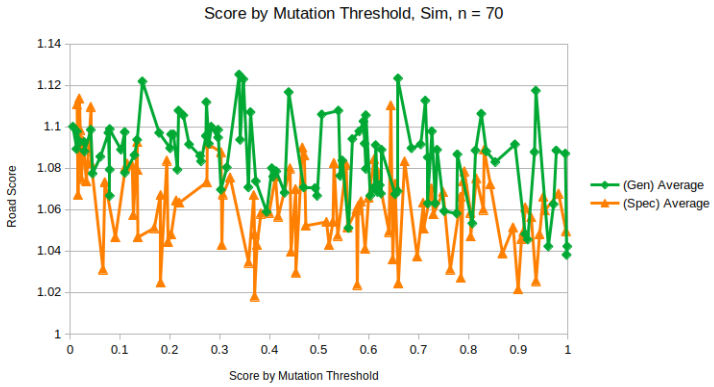
3.2.2 Wykresy wobec parametrów dla każdej instancji

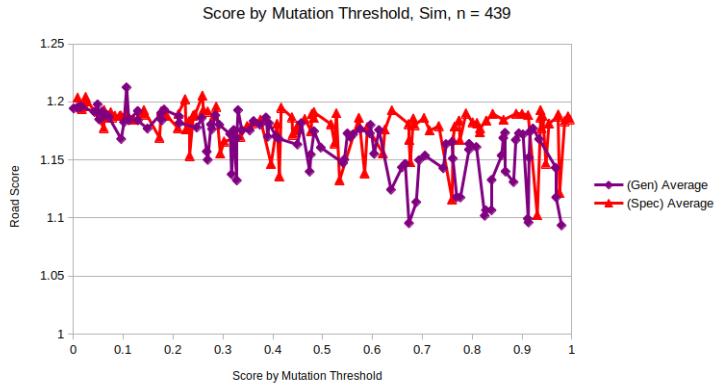
W tej części pokażemy wykresy w rozbiciu o każdy parametr z osobna (traktujemy teraz wszystkie 100 przeszukiwań jako 1 zbiór i dzielimy go względem danego parametru). Pominiemy jednak wykresy dla parametru crossCount, ponieważ dla zestawów specyficznych dla danego typu (a/symetryczne) nie bierze on udziału):

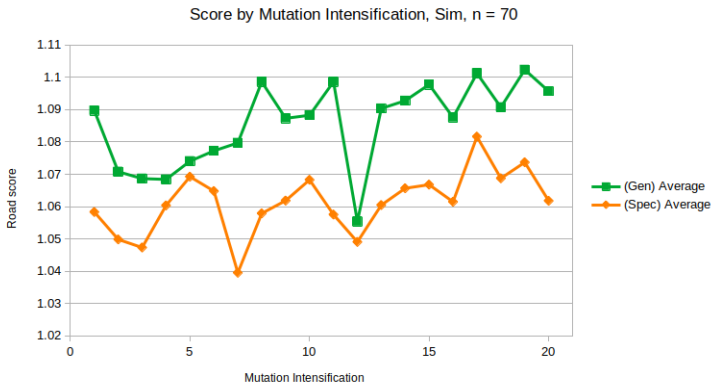
Symetyczne

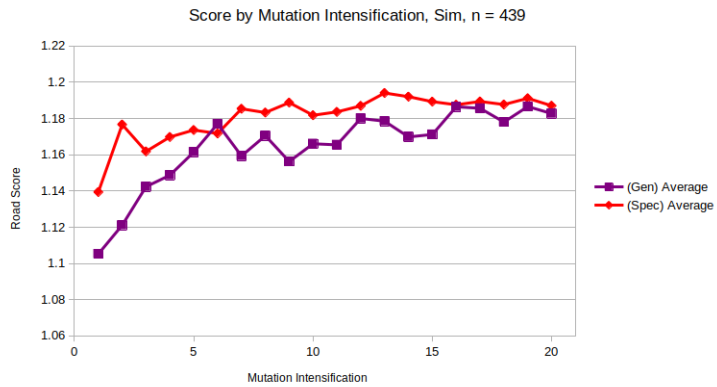


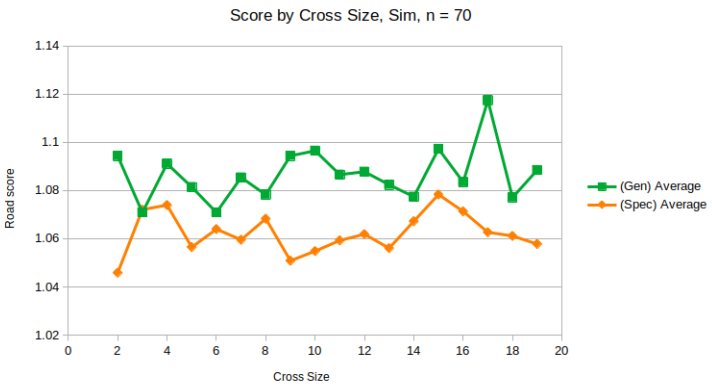


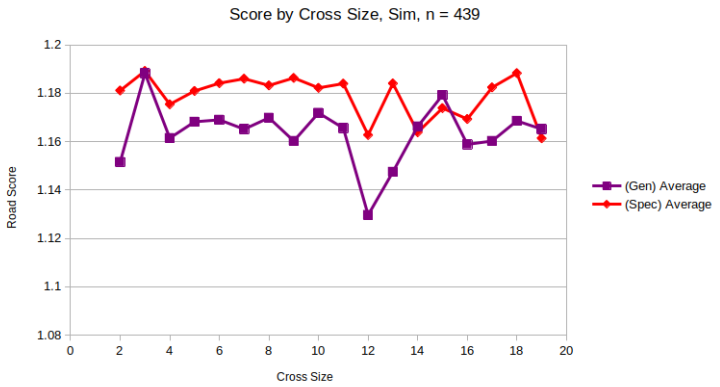




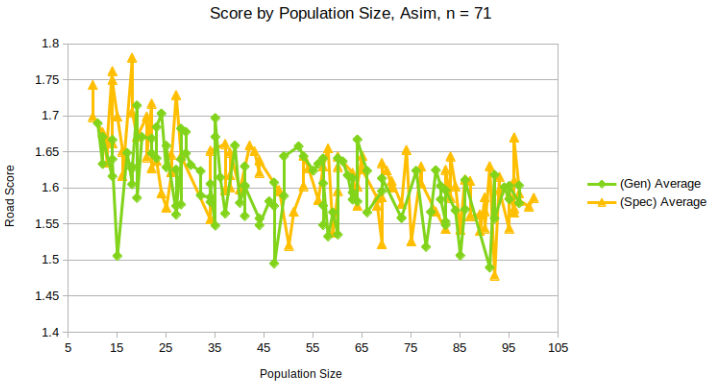


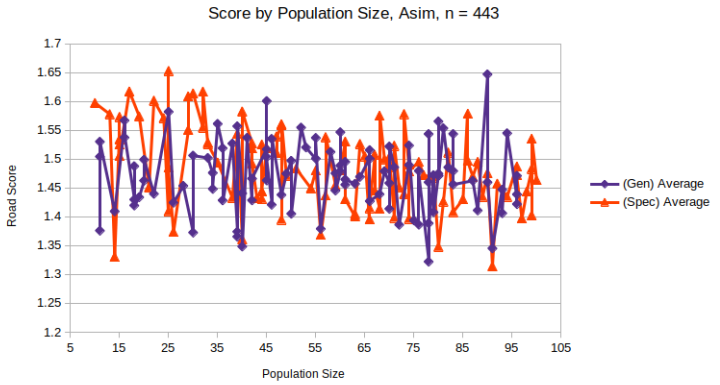


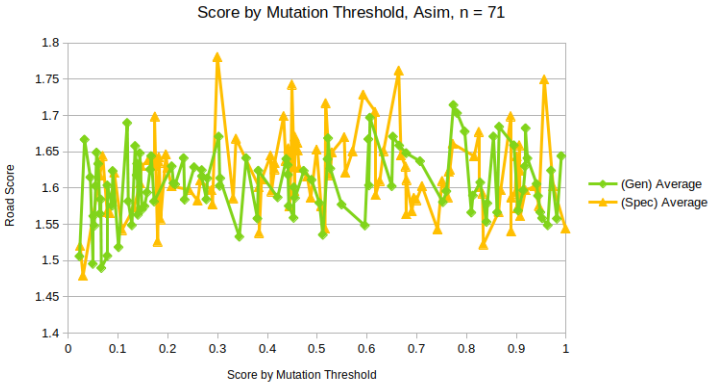


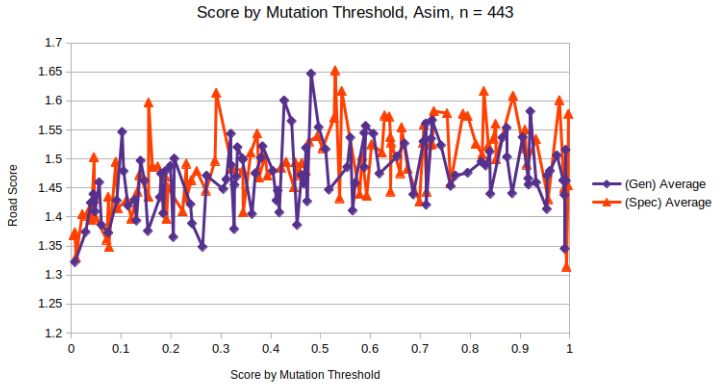


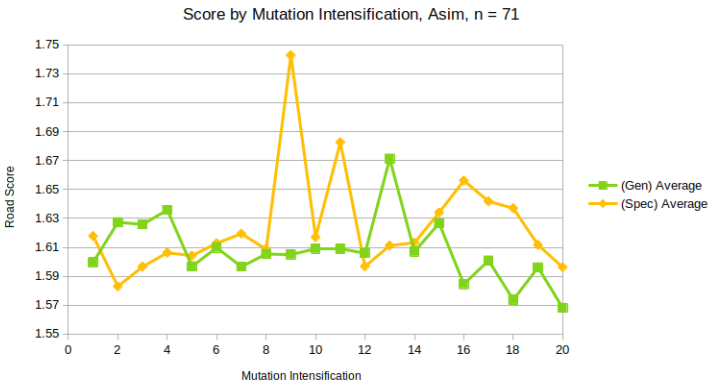
Asymetyczne

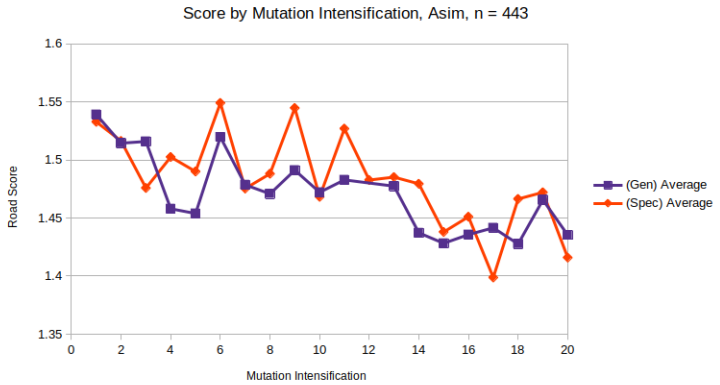


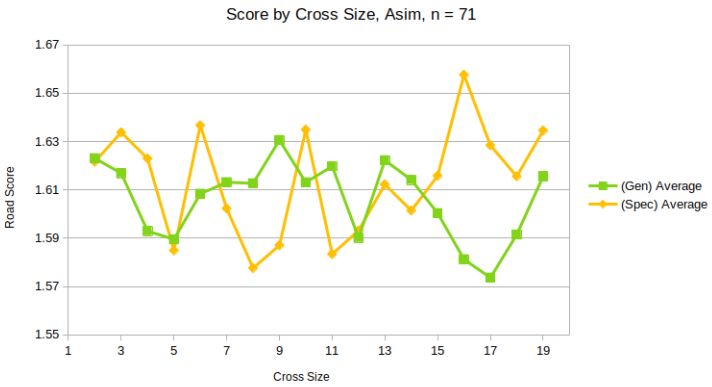


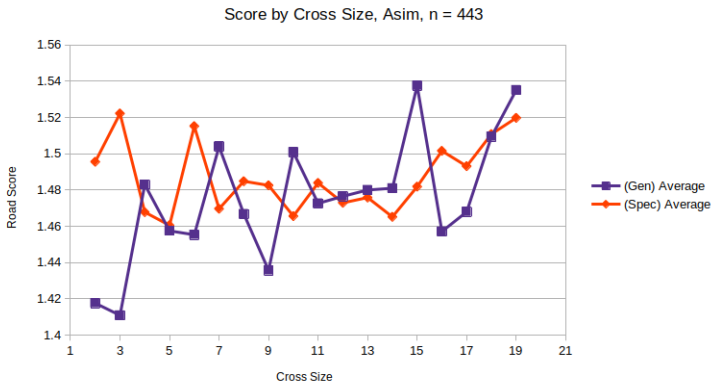












Z powyższych wykresów spróbujmy wyciągnąć kilka wniosków:

- Powtórzmy tutaj na początek wniosek z poprzedniej części, a konkretnie, że mimo wszystko zestawy trybów znacznie różnią się od siebie wydajnością (i wpływem na ogólny wynik hiperparametrów) w zależności od wybranej instancji.
- Wydawać by się mogło, że najmniejszy wpływ na wyniki ma wielkość populacji początkowej (na wszystkich wykresach wyniki są dosyć 'zbalansowane' niezależnie od tej wartości)
- Dla instancji mniejszego rozmiaru widzimy ogólnie mniejszą zależność od hiperparametrów, chociaż dla obu wariantów możemy zauważyć, że optimum dla intensyfikacji jest w okolicach *magicznej 7*, natomiast dla cross Size w okolicach 9.
- Dla instanji większego rozmmiaru możemy zauważyć, że zwiększanie prawdopodobieństwa mutacji znacznie poprawia jakość rozwiązania, jednakże zwiększanie intensyfikacji ją pogarsza. Można zatem powiedzieć, że najlepiej jest 'często mutować, ale delikatnie'. Natomiast wielkość parametru crossSize wacha się w okolicy 12.

3.2.3 5 najlepszych

Teraz spójrzmy na ranking 5 najlepszych zestawów parametrów dla każdej z badanych instancji:

Tabela 5. Symetryczne

Miejsce	n = 70 popSize	n = 70 mutThresh	n = 70 mutInt	n = 70 crosSize	n = 439 popSize	n = 439 mutThresh	n = 439 mutInt	n = 439 crosSize
1	39	0.370261	13	19	51	0.912935	1	11
2	69	0.576427	12	7	57	0.979261	2	2
3	84	0.658883	19	8	43	0.673103	1	12
4	24	0.899584	12	13	78	0.911664	1	12
5	50	0.935501	7	5	55	0.825131	1	14

Tabela 6. Asymetryczne

Miejsce	n = 71 popSize	n = 71 mutThresh	n = 71 mutInt	n = 71 crosSize	n = 443 popSize	n = 443 mutThresh	n = 443 mutInt	n = 443 crosSize
1	91	0.0661295	18	17	91	0.993212	19	12
2	15	0.0222585	14	4	78	0.00685312	17	3
3	92	0.0293365	20	8	91	0.989566	16	5
4	47	0.0490319	10	17	80	0.0753719	10	4
5	59	0.383063	5	7	93	0.1842	10	3

I wyciągnijmy z nich parę wniosków:

- Jak widzimy, dane otrzymane w tabeli nawet pokrywają się po części z tym, co zaobserwowaliśmy na wykresach, a konkretnie, a zwłasza to widać dla instancji 'dużych' i parametów związanych z mutacją (często, acz lekko).
- Dla instancji 'mniejszych' (przypomnijmy że zauważyliśmy na wykresach, że wyniki wydają się bardziej zbalansowane) dane w tabeli bardziej się różnią od siebie (większy rozrzut pomimo wzięcia tylko pierwszej 5), i powiedzmy, że ranking wydaje się nieco bardziej 'losowy' - ale dalej nam determiniuje 'zwycięzce'

3.2.4 Ostatecznie wybrane najlepsze

Tak jak wcześniej powiedzieliśmy, nie można jednoznacznie wyznaczyć najlepszyych hiperparametrów dla całej metody, jednakże możemy wyznaczyć najlepszy zestaw danych dla poszczególnych instancji. Dlatego tak właśnie teraz postąpimy, i w dalszych eksperymentach będziemy używać następujących zestawów:

Tabela 7.	Tryby					Hiperparametry			
Instancja	Start	Sel.	Mut.	CrossM	CrossT	popSize	mutThresh	mutInt	crosSize
Sym70	1	0	3	2	2	39	0.370	13	19
Sym439	1	0	0	1	1	51	0.913	1	11
Asym71	1	0	0	1	2	91	0.067	18	17
Asym443	1	0	0	1	2	91	0.993	19	12

3.3 Badanie wpływu zastosowania 'lokalnej poprawy'

Przy zastosowaniu wyników z 2 poprzednich eksperymentów postanowiliśmy zbadać wpływ działania mechanizmu 'lokalnej poprawy' na wyniki.
Lokalna poprawa - z pewnym prawdopodobieństwem (określanym parametrem *enhanceChance*) po skończonej mutacji (czyli po szystkich iteracjach) na osobniku wykonujemy algorytm lokalnej poprawy, czyli iteracyjnie przechodzimy przez piątki kolejnych miast i tak modyfikujemy trasę, aby w każdej z tych iteracji przejście było minimalne - zatem najpierw 'poprawiamy' miasta 1-5, potem 2-6 itd. Takich możliwych przejść jest 6 (ponieważ zaczynamy zawsze z 1 i kończymy na 5), zatem jest to w miarę szybkie (dzieje się w czasie liniowym względem liczby miast) i nie powinno znacząco wpływać na liczbę wykonywanych iteracji.
W testach wykonaliśmy 10 powtórzeń dla każdej wartości parametru z zakresu $[0.05; 1.0]$ ze skokiem o 0.05. Wyniki przedstawmy na wykresie:

3.4 Badanie wpływu 'wieku' osobników

Eksperyment analogiczny do poprzedniego (w sensie metodologii), jednak tym razem badaliśmy wpływ zastosowania wieku na rozwiązanie. Oznaczyliśmy go jako *AgeMax*, przy czym oznacza to, że osobnik będący w populacji dłużej niż *AgeMax* w procesie selekcji jest 'usuwany' z populacji. Zastosowaliśmy tutaj jednak pewną formę elitaryzmu, ponieważ podczas każdej selekcji 'zerujemy' wiek najlepszego osobnika (Najlepszy ma prawo *Picia ze źródła wiecznej młodości*), dzięki czemu go nie tracimy.
W testach wykonaliśmy 10 powtórzeń dla każdej wartości z zakresu $[1; 20]$, a wyniki przedstawiamy na wykresie:

3.5 Porównanie najlepszych wyników z algorytmem TabuSearch

Aby jakoś porównać działanie naszego algorytmu, zestawimy go tutaj z zaimplementoanym algorytmem *TabuSearch*, gdzie Tabu będzie miało następujące parametry:

- par1
- par2

Natomiast *Genetic* otrzymane z eksperymentów z poszukiwaniami. Dla przypomnienia:

- par1
- par2

Testowaliśmy każdą z 16 badanych instancji, oraz każdemu z algorytmów daliśmy budżet czasu równy 90 sekund. Aby jednak dać jakąś szansę (i odrobinę zredukować losowość) algorytmowi genetycznemu, wykonaliśmy dla niego 3 iteracje po 30 sekund. W ten sposób jego wyniki mają szansę być nieco bardziej miarodajne. Wyniki przedstawmy w tabeli:
Oraz jeszcze zobaczmy, co Wilcoxon nam o tym mówi:

3.6 Badania nad Modelem Wyspowym

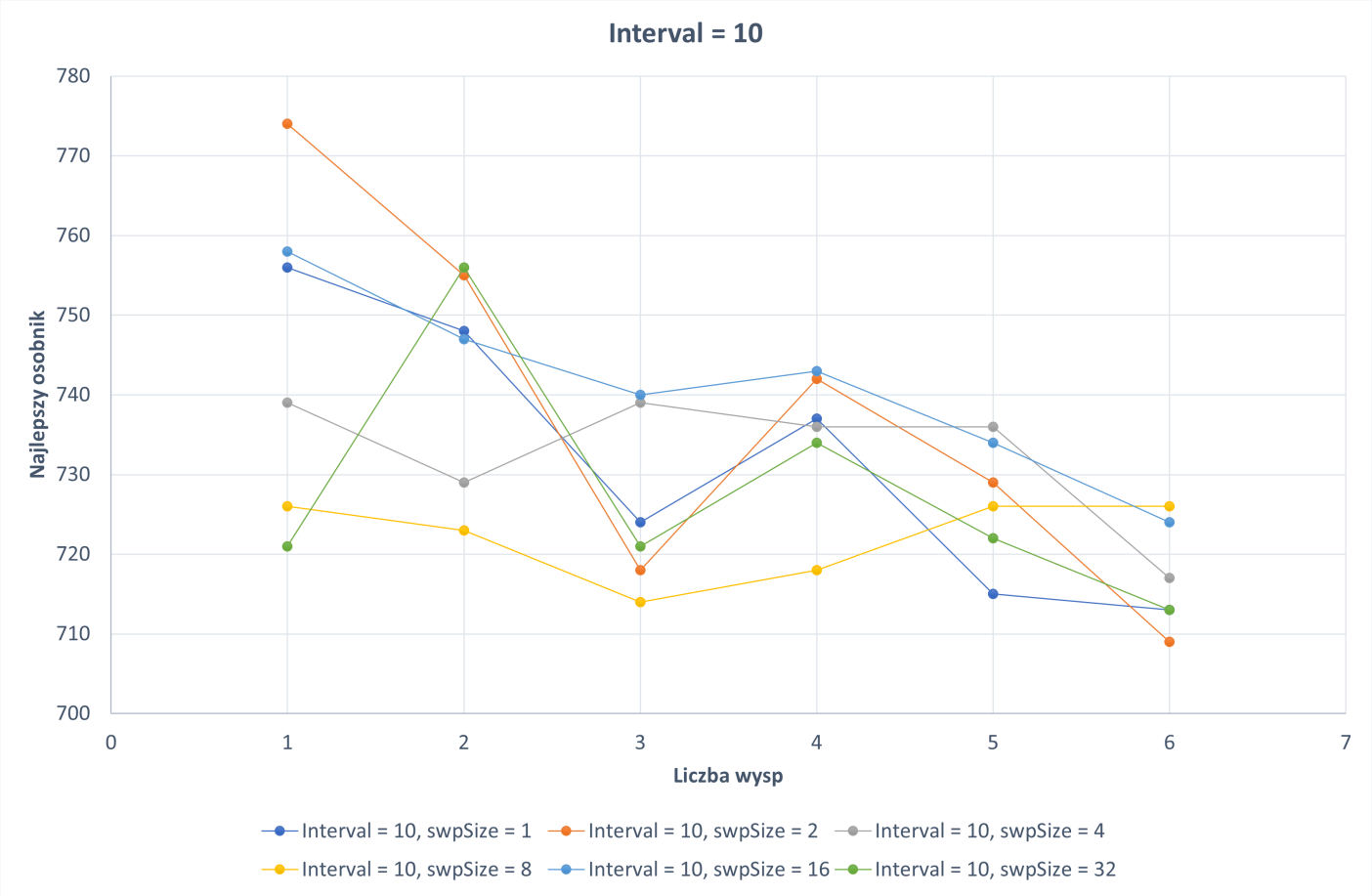
Model wyspowy zaimplementowano w następujący sposób:

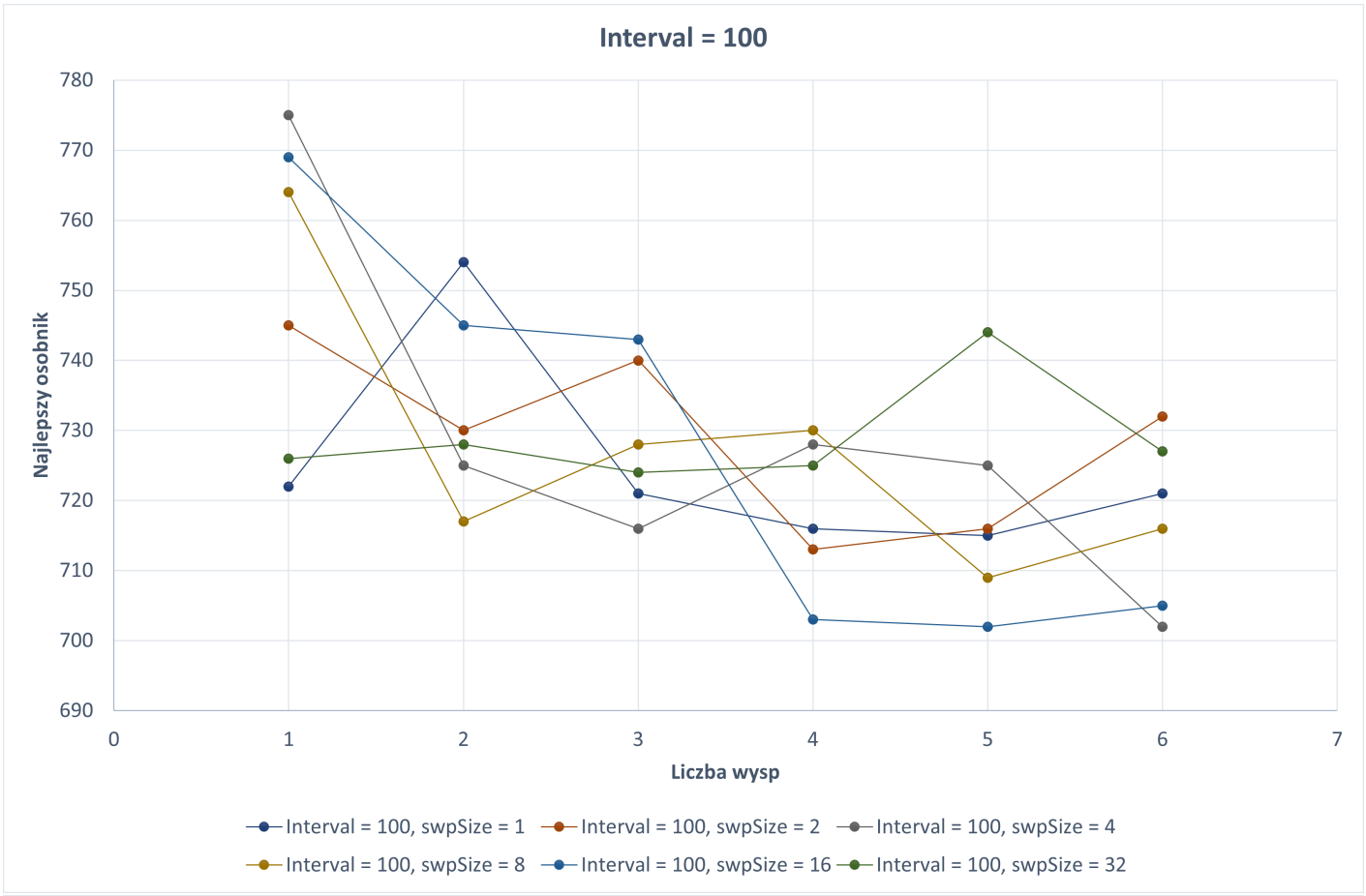
- Każda z wysp prowadzi własną instancję algorytmu genetyczne na pojedyńczym rdzeniu procesora, wyjściowa populacja losowa, parametry (wybrane najlepsze z uprzednich testów):

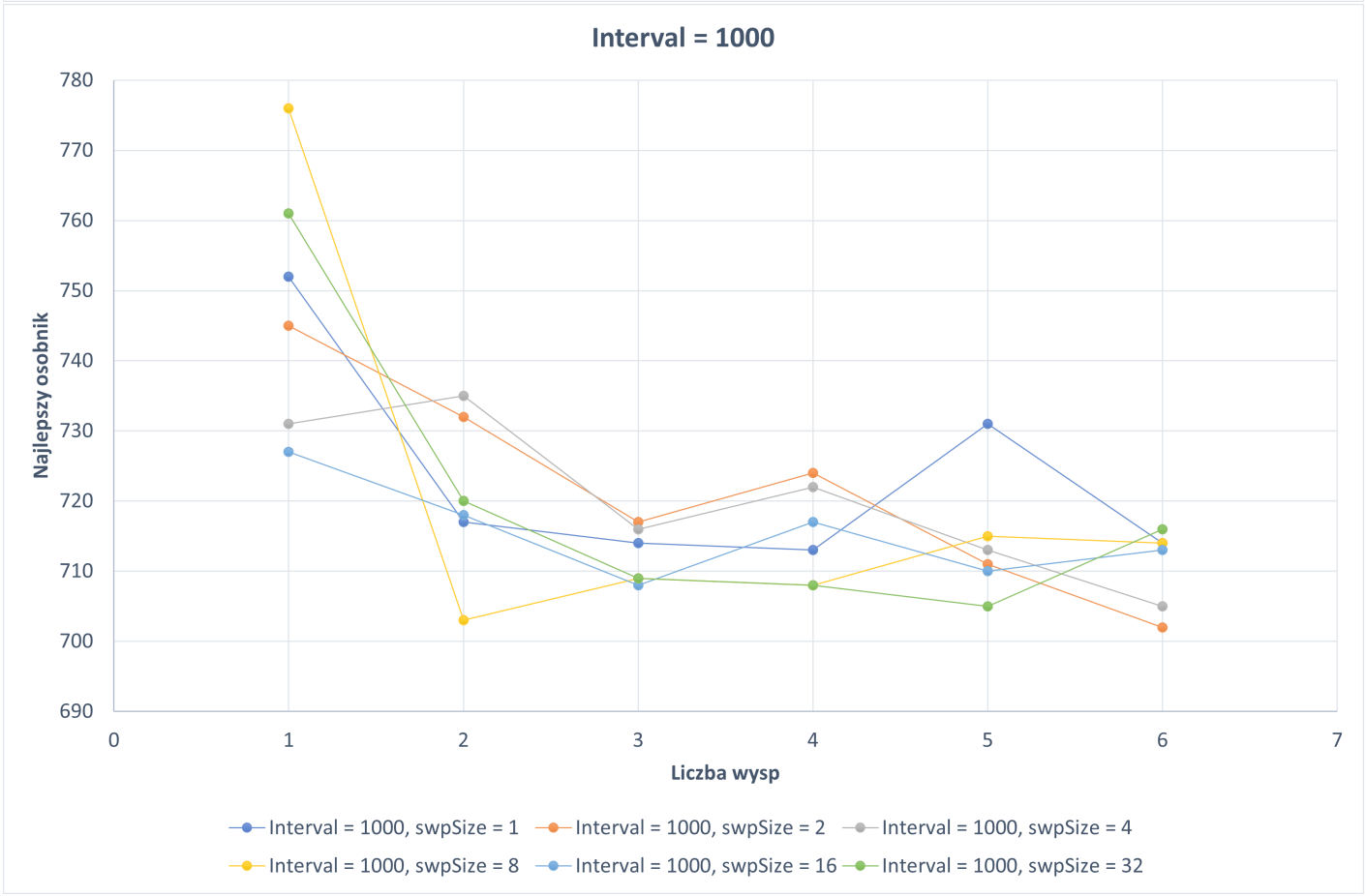
  - populationSize = 50
  - mutationIntensification = 13
  - crossSize = 1
  - crossCount = 177
  - crossMode = 1
  - mutMode = 0
  - crossType = 1
  - selectionMode = 0
  - double mutationThreshold = 0.6
  - time = 30.0
- po ustalonej liczbie iteracji, z każdej z wysp zabierano k osobników, po czym z powstałej puli w losowy sposób wysyłano ich z powrotem na wyspy

Badania przeprowadzono na instancji st70.tsp, każda kombinacja parametrów została uruchomiona trzykrotnie, a wynik został uśredniony.

Wpływ długości interwału pomiędzy mieszaniem




Interval = 100
780
770
760
750
740
730
720
710
700
690
Najlepszy osobnik
0
1
2
3
4
5
6
7
Liczba wysp
Interval = 100, swpSize = 1
Interval = 100, swpSize = 2
Interval = 100, swpSize = 4
Interval = 100, swpSize = 8
Interval = 100, swpSize = 16
Interval = 100, swpSize = 32


Interval = 1000
780
770
760
750
740
730
720
710
700
690
Najlepszy osobnik
0
1
2
3
4
5
6
7
Liczba wysp
Interval = 1000, swpSize = 1
Interval = 1000, swpSize = 2
Interval = 1000, swpSize = 4
Interval = 1000, swpSize = 8
Interval = 1000, swpSize = 16
Interval = 1000, swpSize = 32

Wpływ liczby osobników podlegających mieszaniu

Na każdym z wykresów mniej lub bardziej, ale zawsze widać tendencję spadkową, co pozwala zakładać, że model wyspowy został zaimplementowany prawidłowo. Ulepszenie nie jest liniowe, ale skuteczne. Najszybszy zysk widać przy długich interwałach pomiędzy mieszaniem, w przypadku `Interval=1000` wartość funkcji celu już przy dwóch wyspach była znacznie lepsza od pojedyńczej, potem jednak zyski kolejnych wysp były mniejsze. Dla parametru liczby osobników branych do mieszania nie widać szczególnie jakiś własności, poza nieprzewidywalnością wyniku, kiedy bierzemy tylko po jednym z osobników.

4 Drobne podsumowanie; Tabele dodatkowe

Zawrzemy tutaj niepokazane wcześniej tabele z wynikami, oraz pokusimy się o podsumowanie naszych eksperymentów.

Tabela 2 Pisemnie Id	StartMode	SelectionMode	MutMode	CrossMode	CrossType	Sum	Avg
107	Hybryda	Turniej	Losowa	Part.Map.	Size/2	98	12.25
102	Hybryda	Turniej	Losowa	Mod.Ord.B.	All Par	124	15.5
106	Hybryda	Turniej	Losowa	Part.Map.	Size/2	127	15.875
103	Hybryda	Turniej	Losowa	Mod.Ord.B.	Pop=20	141	17.625
142	Hybryda	Ruletka	Losowa	Part.Map.	Pop=20	161	20.125
139	Hybryda	Ruletka	Losowa	Mod.Ord.B.	Pop=20	163	20.375
140	Hybryda	Ruletka	Losowa	Mod.Ord.B.	Size/2	188	23.5
143	Hybryda	Ruletka	Losowa	Part.Map.	Size/2	197	24.625
138	Hybryda	Ruletka	Losowa	Mod.Ord.B.	All Par	202	25.25
104	Hybryda	Turniej	Losowa	Mod.Ord.B.	Size/2	206	25.75