Raport z optymalizacji przy pomocy Algorytmów Genetycznych

Jakub Belter, 18/04/2021

Wstęp

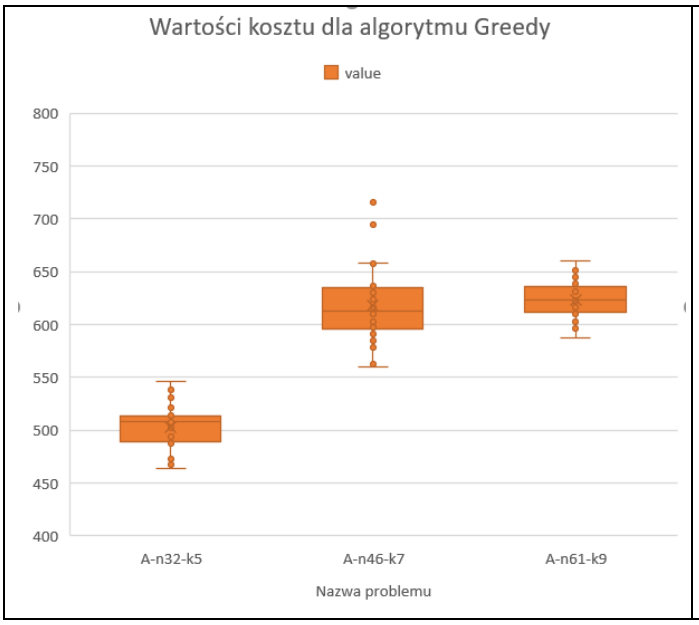
Celem tego raportu jest zaznajomienie się z Algorytmem Genetycznym i sprawdzenie jak jego parametry wpływają na uczenie i jakość rozwiązania. Problemem na którym ten algorytm będzie testowany to problem komiwojażera. Jest on zadaniem minimalizującym tj. im mniejsza wartość jakości (czyli w tym przypadku kosztu/przebytej drogi) tym lepiej. Również, w celach porównawczych zostanie użyty algorytm Greedy i RandomSearch, aby jak sobie AG radzi w porównaniu z nimi.

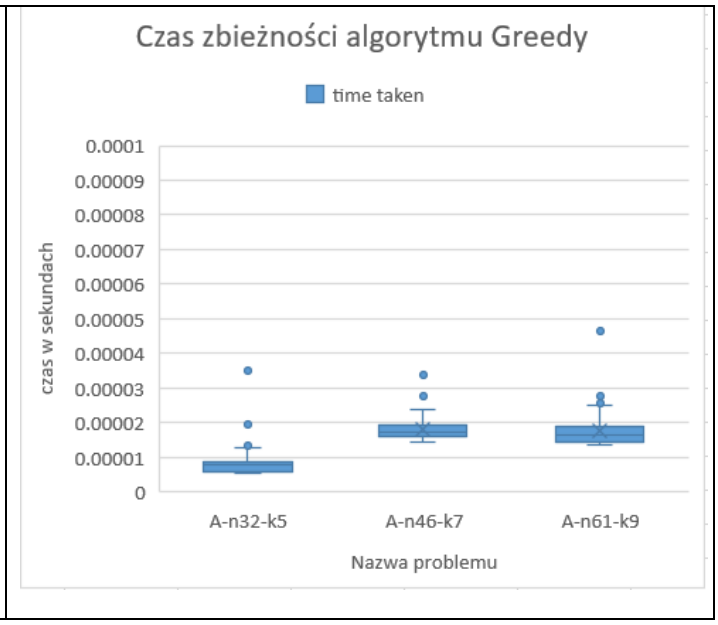
Wyniki

Greedy

Algorytm zachłanny został zaimplementowany w sposób zaproponowany na laboratorium.

1. Najpierw jest wybierany początkowy węzeł
2. Następnie wybieramy wierzchołek, który jest najbliżej do poprzedniego, ale bez cofania.
3. Poprzedni krok powtarzamy aż do odwiedzenia wszystkich węzłów.
4. Koniec

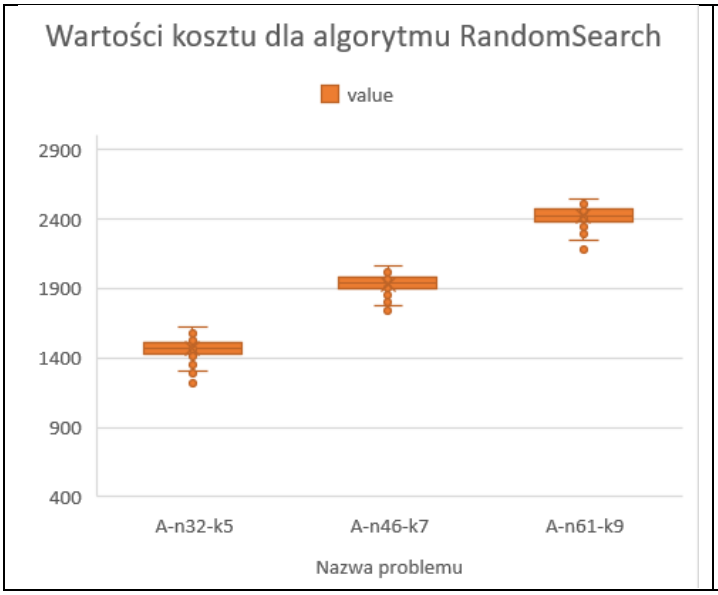


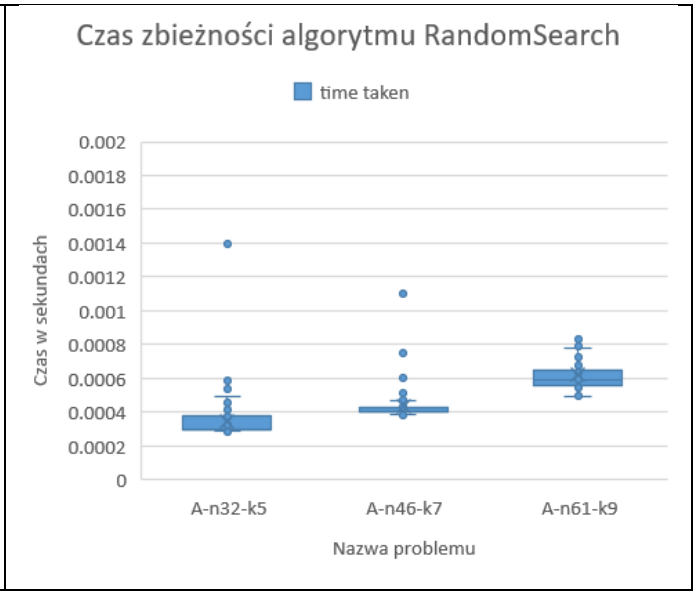


Powyżej został przedstawiony wynik jakościowy działania tego algorytmu. Widać, że koszt dla pierwszego problemu wychodził średnio *500*, dla drugiego około *620*, a trzeciego delikatnie więcej, bo *630*. Również można zwrócić uwagę na czas wykonania algorytmu zachłannego, wykonuje się poniżej ułamka milisekundy co sprawia, że jest to bardzo szybki algorytm.

RandomSearch

Algorytm losowego przeszukiwania nie wymaga większego wyjaśnienia – losujemy rozwiązanie i jeżeli jest ono lepsze od poprzedniego najlepszego, ustawiamy je jako najlepsze.





Powyżej zostały przedstawione wyniki dla losowego przeszukiwania przestrzeni rozwiązań. Jak można zauważyć rozwiązania podawane przy losowym wyborze rozwiązań posiadają **trzykrotnie** gorsze wyniki niż w przypadku podejścia zachłannego jak i również czas trwania algorytmu, chociaż jest nadal bardzo mały, to jest **dwudziestokrotnie** wolniejszy. Te średnie wartości miary dla tych problemów to około *1450*, *1950* oraz *2450* Można uznać, że jest to podstawa/linia przecięcia wyników. Gdyż jeżeli są gorsze od niego (w przypadku jakości rozwiązania), to nie są warte większej uwagi.

Algorytm Genetyczny

Algorytm, który został zaimplementowany delikatnie różni się kolejnością poszczególnych operacji, niż ten zaproponowany na zajęciach tj. zamiast wykonywać operacji na każdym osobniku w jednym wykonaniu pętli, w tej implementacji cała populacja jest modyfikowana. Uproszczenie działania można zaprezentować na poniższym urywku kodu.

```
private void AG()
        {
            Initialize();

            while (!ShouldStop())
            {
                Select();

                Crossover();
                Evaluate();

                bool foundNewBest = CheckNewBest();

                Mutate();
                Evaluate();

                AnalizePopulation();

                CheckNewBest();
            }
            var x = DateTime.UtcNow;
            timeTaken = (x - startTime).TotalSeconds;
        }
```

Operacja selekcji, która została użyta do badań do **Selekcja Turniejowa**. Wykorzystana operacja krzyżowania to operator **Order Crossover**. Samo działanie tej operacji zostało zaprezentowane w liście dotyczącej tego zadania. Natomiast mutacja, została zaimplementowana jako zamiana dwóch odwiedzanych miejsc; **Two-Point Swapping.**

Celem całego raportu jest prezentacja działania Algorytmu Genetycznego, ze zmiennymi hiper parametrami. Co też zostanie tutaj zaprezentowane. Badane będą:

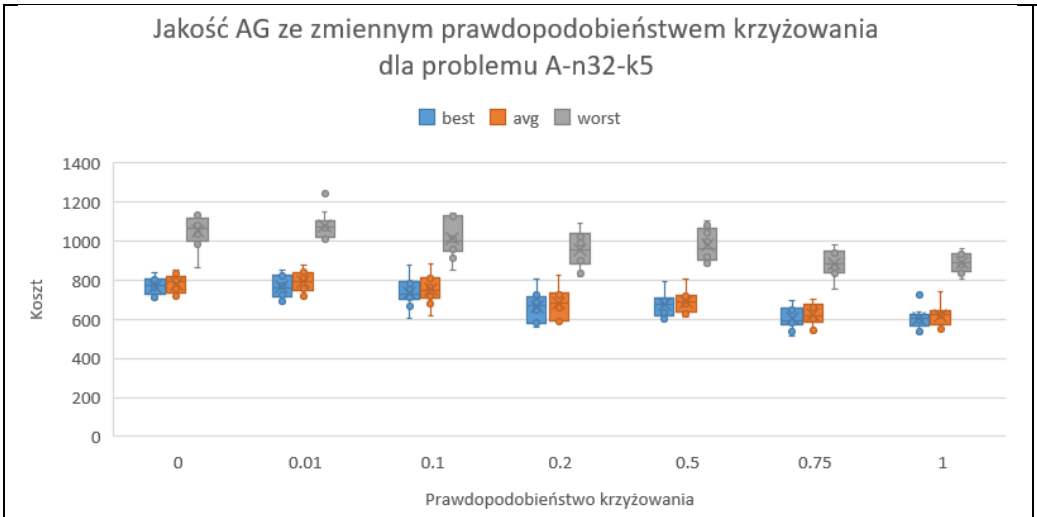
- Wpływ operatorów krzyżowania i mutacji na finalny model, a dokładniej ich prawdopodobieństwa
- Wpływ parametrów populacji czyli rozmiar populacji, liczba generacji jak i rozmiar turnieju na jakość finalnego modelu.

Co również jest istotne to to, że każdy z parametrów jest testowany osobno czyli reszta parametrów pozostaje na jednym poziomie; poziomie domyślnym. Poniżej zostały przedstawione domyślne wartości parametrów.

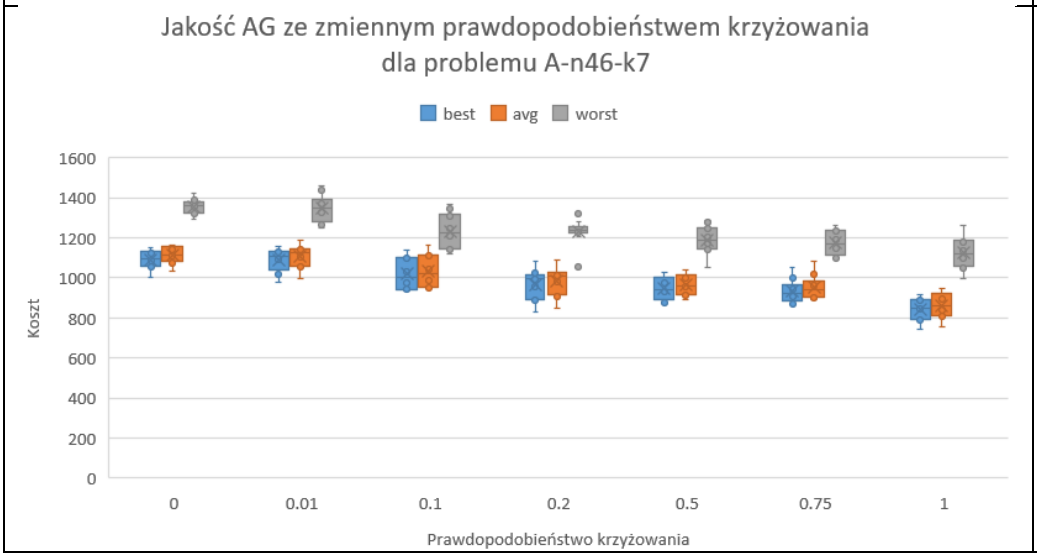
Nazwa parametru	*Wartość*
Prawdopodobieństwo krzyżowania	0.7
Prawdopodobieństwo mutacji	0.1
Rozmiar populacji	100
Ilość generacji	100
Rozmiar turnieju	5

Operatory krzyżowania

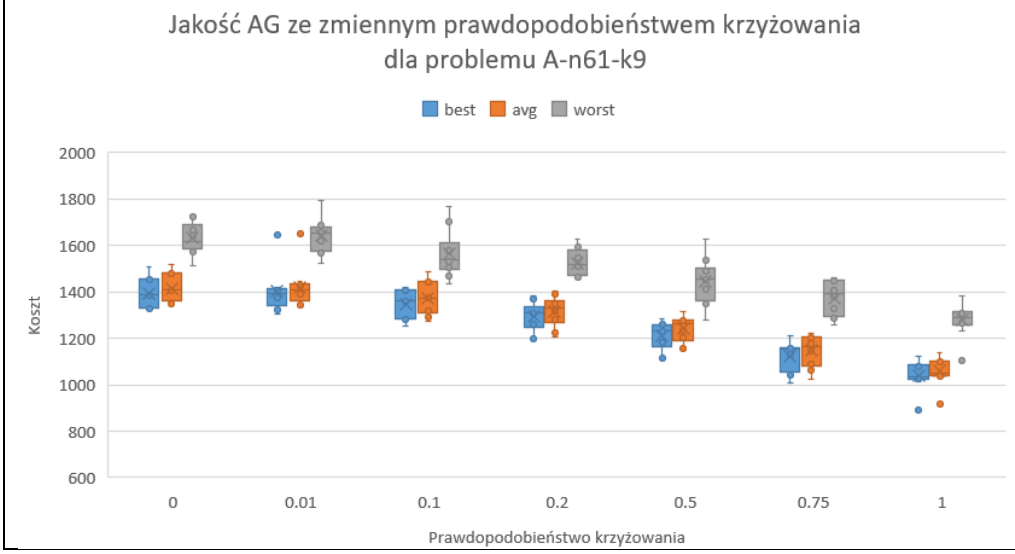
Poniżej zostały przedstawione wykresy z jakością algorytmu w porównaniu z innymi parametrami prawdopodobieństwa krzyżowania.



Na wykresie można zauważyć, że przy zwiększonej liczbie krzyżowań uzyskujemy coraz lepsze wyniki, aczkolwiek w tym przypadku nie jest do tak bardzo widoczne ze względu na małą złożoność problemu. Zakresy najlepszych, średnich i najgorszych ciągle obniżają swoją wartość kosztu.



W drugim problemie ta tendencja zauważona powyżej jest o wiele bardziej widoczna. W przypadku braku krzyżowań w najgorszym przypadku mamy jednak dalej lepsze wyniki niż Random Search, ale daleko to do wyników algorytmu zachłannego.

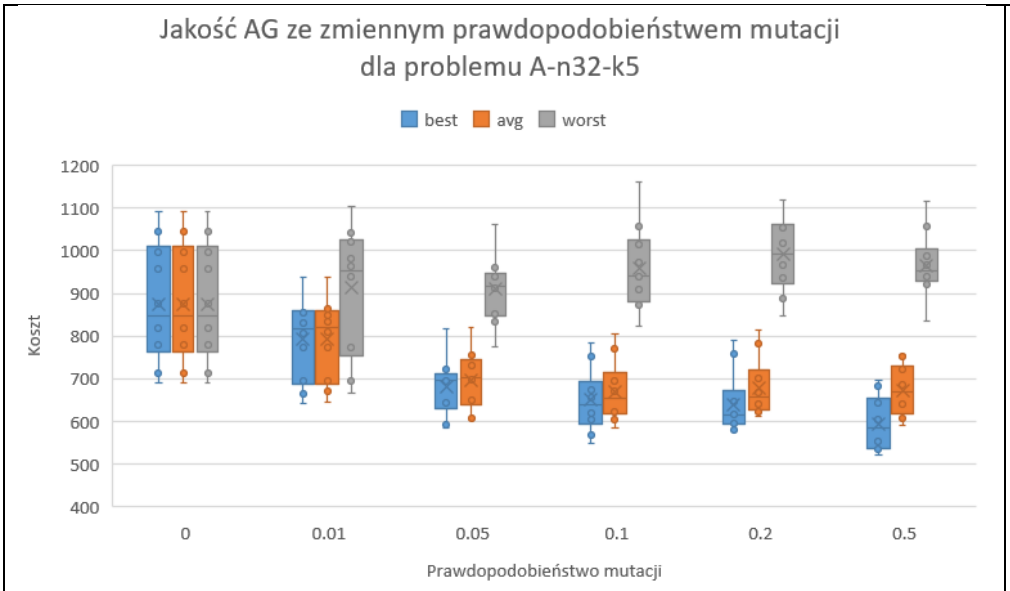


Jak można zauważyć trend jest coraz silniejszy dla dużych problemów – przy coraz większych problemów. Co ciekawe widać, że jest więcej punktów odstających od reszty wyników (tzw. outlierów), które się zbliżają do wyników algorytmu zachłannego.

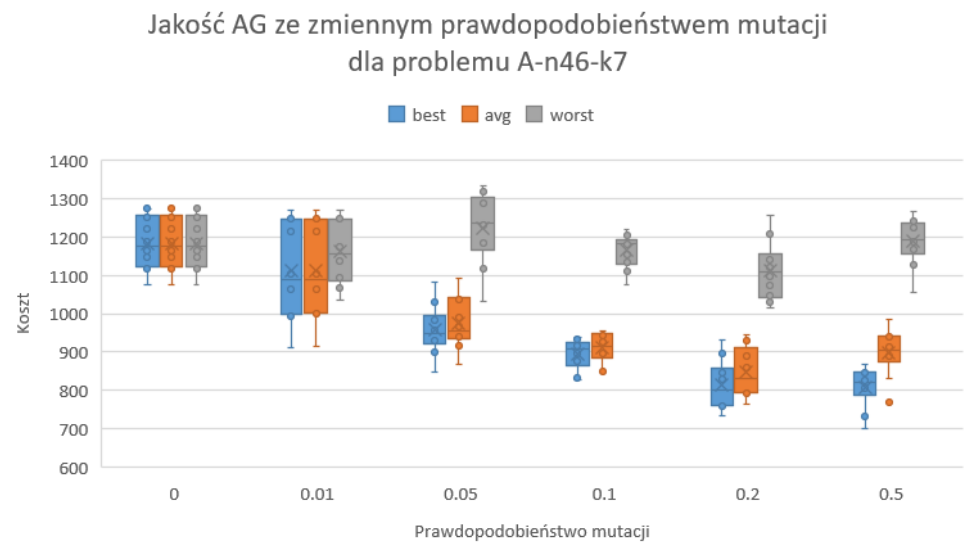
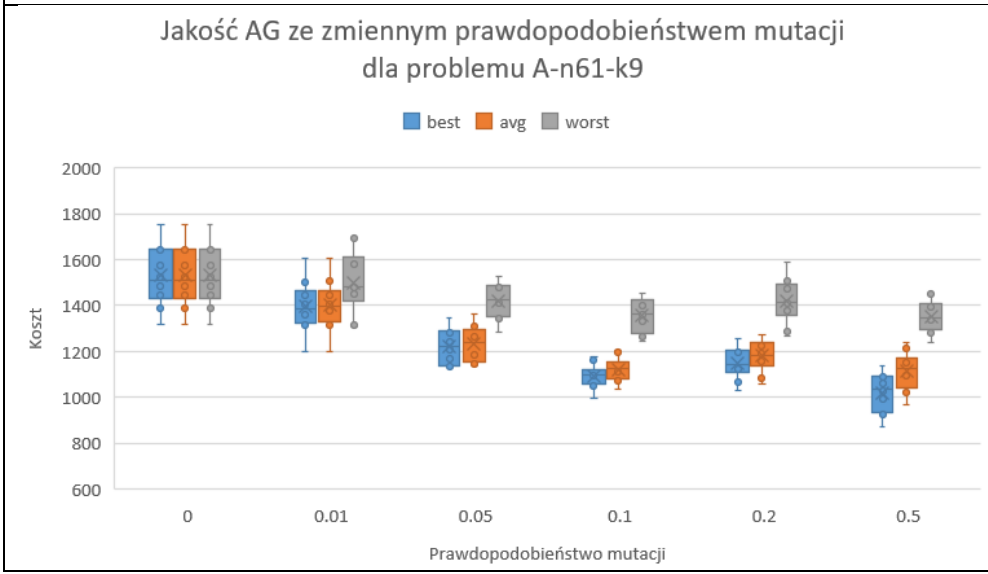
Można tutaj wywnioskować, że krzyżowanie dla małych problemów nie jest aż tak istotne ze względu na to, że nie musimy eksploatować danych rozwiązań gdyż są one mniej złożone, natomiast przy narastającej liczbie węzłów krzyżowanie jest jak najbardziej wskazane. Można powiedzieć, że wartość **prawdopodobieństwa krzyżowania** powinna być **jak największa**. Aczkolwiek nawet przy braku krzyżowania wynik jest **lepszy** od RandomSearcha.

Operatory mutacji

Poniżej zostały przedstawione wyniki GA dla zmiennego prawdopodobieństwa mutacji.



Można zauważyć, że przy braku mutacji, rozwiązania nie potrafią się rozwijać, „pula genów” jest zamknięta i nie rozwija się ona. Może to wskazywać na błędną implementacje lub tendencję wpadania w określone minimum lokalne. Aczkolwiek przy zwiększonym prawdopodobieństwie mutacji polepszają się znacznie wyniki najlepszych osobników, to średnie osobniki się pogarszają, a najgorsze osobniki mają jeszcze gorsze wyniki niż by mniejszych ilościach.

<table>
<tr>
<td> 
</td>
<td>Tendencje opisane wyżej powtarzają się. Aczkolwiek w tym przypadku przy bardzo dużym prawdopodobieństwie najlepsze osobniki się pogarszają. Wskazywać to może, że odpowiednia wartość tego parametry jest pomiędzy 0.2 a 0.5.</td>
</tr>
<tr>
<td> 
</td>
<td>Co ciekawe, poniżej problemy opisane wyżej to jest wyniki średnie i najgorsze nie zataczają łuku funkcji kwadratowej. Może to być jednak spowodowane małą ilością prób wykonania algorytmu genetycznego.</td>
</tr>
</table>

Jak można zauważyć wartość mutacji powinna być koniecznie **nie zerowa,** gdyż znacznie poprawia ona najlepszych osobników. Jednakże trzeba uważać z tym parametrem, aby umożliwić heurystyce eksploatacje rozwiązań. Co ciekawe dla wysokich wartości mutacji wyniki są **najbardziej zbliżone do wyników algorytmu zachłannego.**

Mutacja czy krzyżowanie?

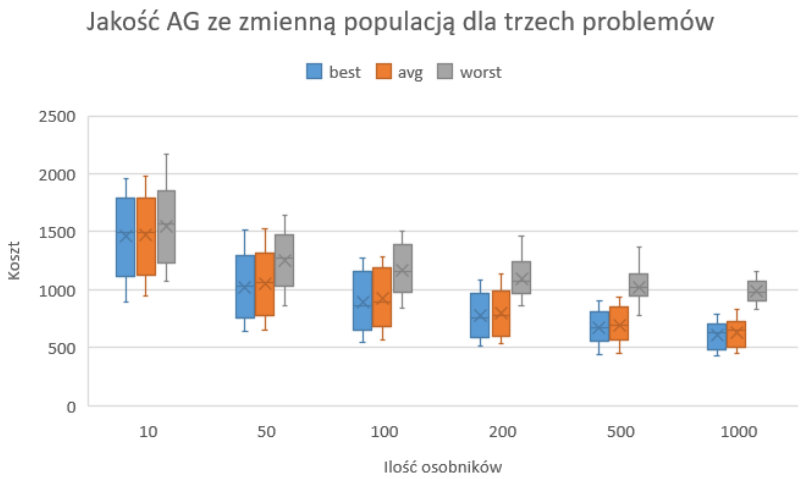
Aby odpowiedzieć na pytanie które ważniejsze, trzeba ustawić odpowiednie parametry do testów. Poniżej można znaleźć tabelę opisującą wartości modelu dla problemu „średniego” dla 20 powtórzeń, 200 iteracji, 100 osobników oraz 5 osobników w turnieju. Te parametry zostały dobrane na podstawie badań później opisanych.

Px	Pw	Avg(best)	Avg(avg)	Avg(worst)
0	0	1944.5	1944.5	1944.5
0.1	0	1851.85	1851.85	1851.85
0.8	0	1634.87	1634.87	1634.87
1	0	1509.03	1509.03	1509.03
0	.2	1375.86	1380.37	1433.2
0	.5	1278.04	1292.88	1380.89
0	1	1237.10	1284.86	1395.04

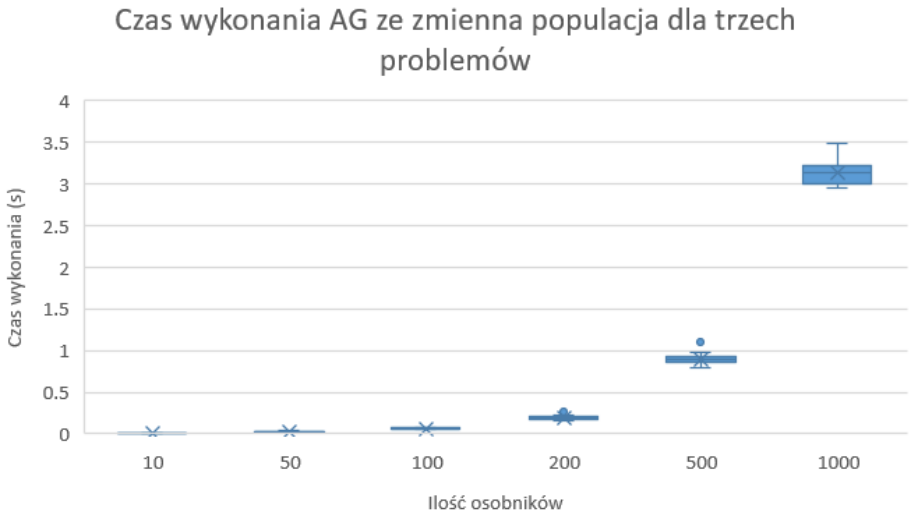
Po analizie powyższej tabeli można stwierdzić iż w tym przypadku mutacja jest znacznie ważniejsza od krzyżowania. Również z braku mutacji cała populacja składa się z podobnych wyników.

Rozmiar populacji

W przypadku tego parametru są badane dwie rzeczy – jak ilość osobników wpływa na jakość AG oraz na czas wykonania algorytmu. W tym przypadku nie ma podziału na problemy, gdyż były one zbyt podobne.



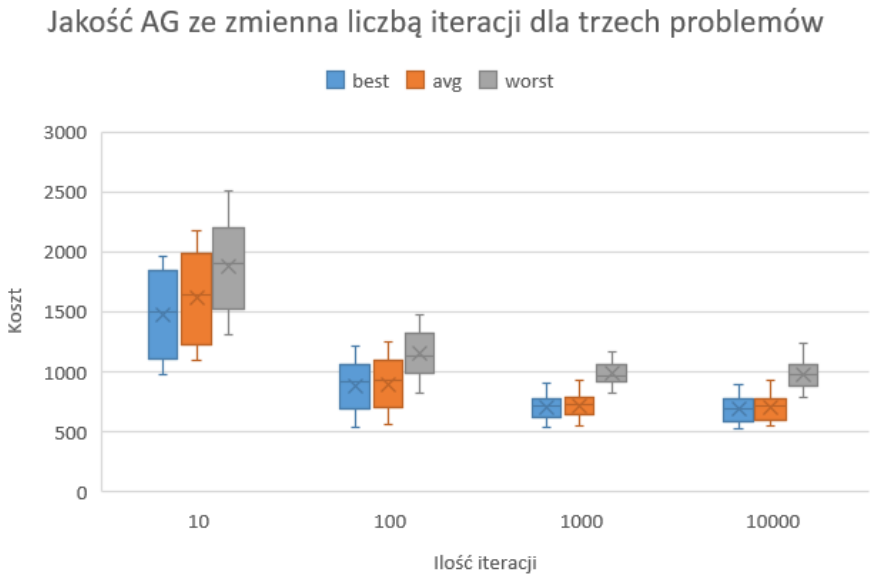
Można zauważyć iż przy większej ilości osobników AG poprawia swoje działanie, poprzez zmniejszenie wariancji przy kolejnych wykonaniach algorytmu. Jednakowoż największy skok jakości jest pomiędzy *10* osobników a *100*. Również wartość *200 i 500* wskazuje na pewną poprawę. W przypadku *1000* osobników nie widać szczególnej poprawy.



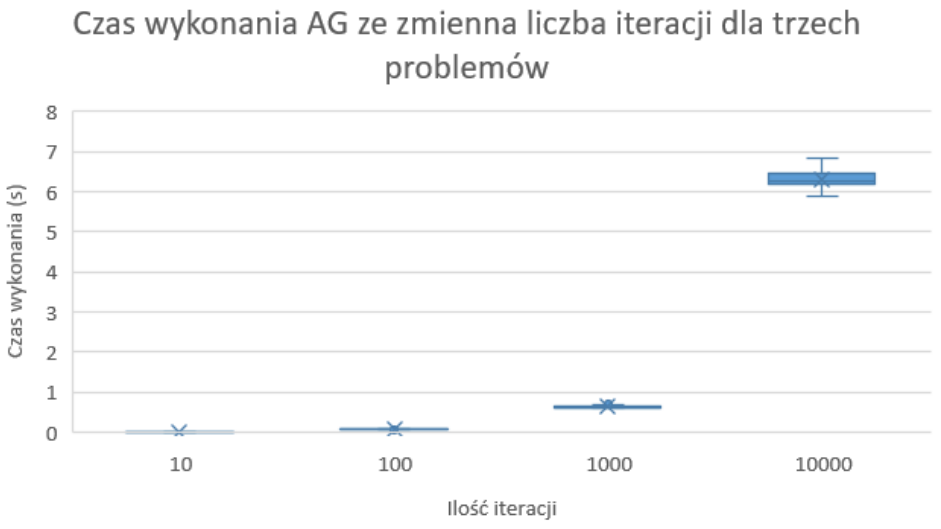
Niestety, pomimo popraw jakości wykonie pojedyncze algorytmu genetycznego dla *1000* osobników wzrosło do średnio **3 sekund co jest 300 000 razy dłuższy** niż w przypadku **algorytmu zachłannego.** Również jakość rozwiązań jest tylko zbliżona do algorytmu zachłannego co sprawia, że **zbyt duża ilość osobników** wpływa **negatywnie na czas rozwiązania.** Być może najlepszym rozwiązaniem byłoby ustawienie ilości osobników na około 100 i zwiększyć ilość iteracji algorytmu.

Ilość generacji

Przedstawienie wyników jest podobne jak dla parametru powyżej.



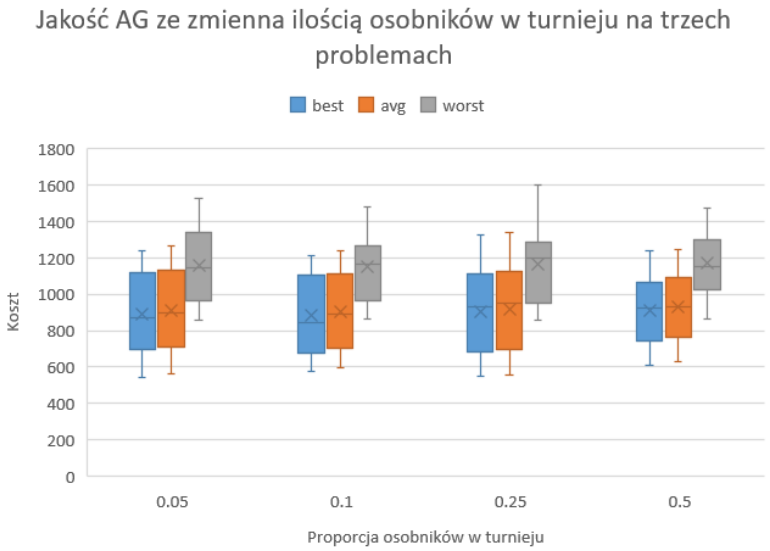
Pierwsze co się rzuca w oczy to fakt, iż 1000 iteracji w pełni wystarcza, żeby algorytm genetyczny zbiegał do minimum globalnego dla każdego z problemów. Można łatwo stwierdzić iż z zbyt małą liczbą iteracji algorytm nie jest w stanie efektywnie zbiec do minimum.



Również i w tym przypadku zbyt duża ilość sprawia, że algorytm wykonuje się zbyt długo, aby uznać go za akceptowalny. Dla tych problemów być może lepszym rozwiązaniem byłoby stworzenie końca uczenia z warunkiem poprawy rozwiązania o nowy hiperparametr epsilon i patience. Tj. jeżeli poprawił się wynik o epsilon zresetuj curPatience w innym przypadku zwiększ wartość. Jeżeli curPatience jest większy lub równy patience kończymy algorytm – uważamy, że nie da się poprawić tego co mamy – doszliśmy do minimum.

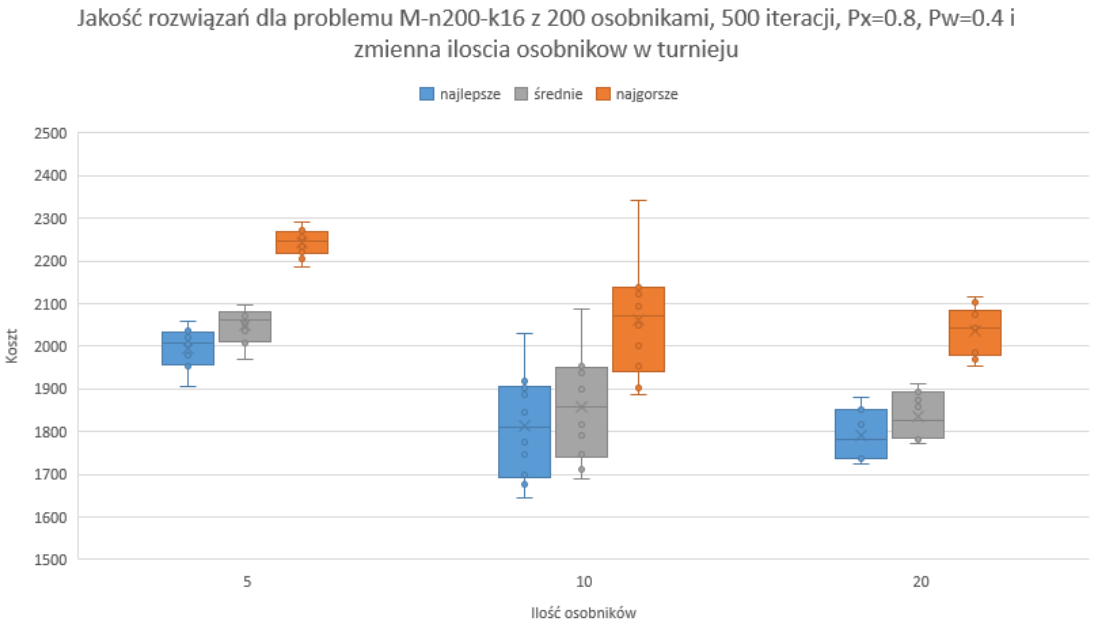
Rozmiar turnieju

Następnie zostaną przedstawione zmiany rozmiaru turnieju na wartości jakości.



Co dziwne przy coraz większej ilości osobników w turnieju (proporcja od 100 osobników), wyniki w cale się nie poprawiają. Jedynie co to się pogarszają przy zbyt dużej ilości osobników w turnieju. Można w takim razie dojść do wniosku, że zbyt duża ilość osobników w turnieju sprawia, że algorytm wpada w minimum lokalne i nie potrafi z niego odejść (bo zawsze wybieramy najlepszych osobników i nie dajemy szans tym gorszym.

Niemniej jednak jeżeli zwiększymy liczbę osobników powinniśmy również zwiększyć ilość osobników w turnieju. Ten wniosek wziął się z badań na ostatnim problemie dodatkowym (200 węzłów). Poniżej wykres przedstawiający tą zależność.



Czyli reasumując – turniej najlepiej sobie radzi, jeżeli liczba osobników w turnieju wynosi minimum 5% całej populacji.

Porównanie algorytmów

Algorytm	*Najlepszy wynik*	*Średni najlepszy wynik*	*Czas*
Greedy Search	560	618	< 0.1 ms
Random Search	1741	1930.5	~ 2 ms
Genetic Algorithm	581	643.8	10s > x > 3 ms

W tym przypadku algorytm zachłanny najlepiej sobie radzi z problemem komiwojażera (problem „średni” tj. 46 węzłów). Następnie mamy algorytm genetyczny, który z odpowiednimi parametrami dorównuje zachłannemu ale znacząco traci na szybkości. Najgorszym jest random search – najgorsze wyniki, ale czas posiada lepszy niż GA.

Podsumowanie

W podsumowaniu zostaną przytoczone dwie sprawy – kwestia doboru hiper parametrów oraz odpowiedzi na pytania pomocnicze. Jeżeli chodzi o pierwszą kwestię, po badaniu zachowania algorytmu można wysnuć poniższe wnioski.

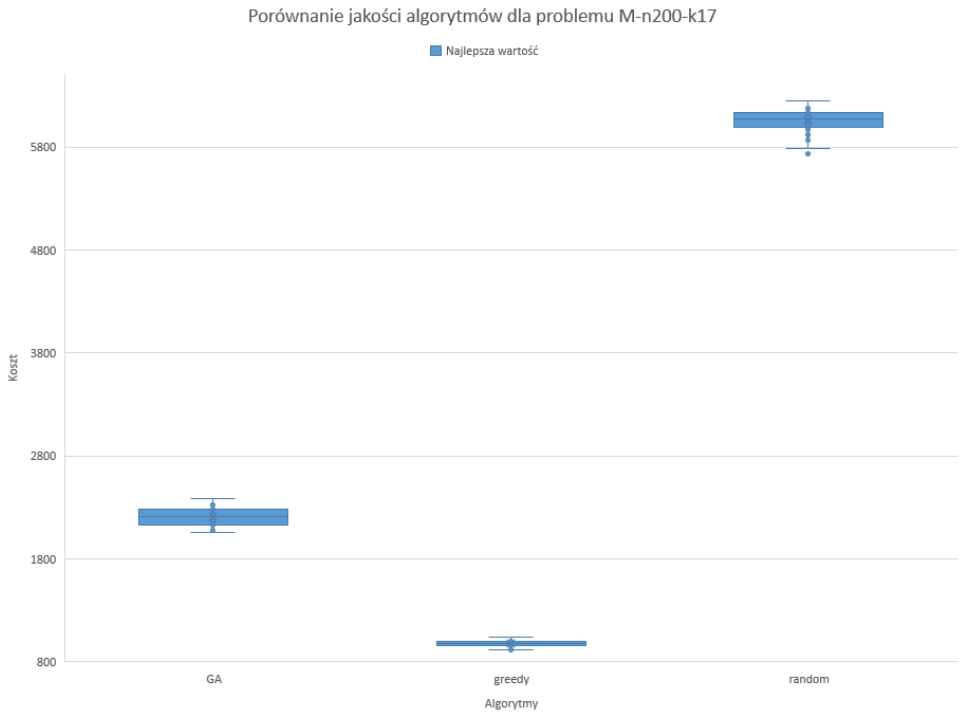
Nazwa hiper parametru	*Sugerowana wartość*
Prawdopodobieństwo krzyżowania	Najlepiej jak najwyższe, ale nie musi być równe 1 **(0.8)**
Prawdopodobieństwo mutacji	Nie może być zerowe, ani za duże **(0.2)**
Ilość osobników w populacji	W zależności od problemu, im bardziej złożony tym więcej. Jako podstawową wartość można podać **(100)**
Ilość generacji / iteracji	Wystarczająca, aby algorytm zdążył zbiec do minimum. Sugerowana modyfikacja warunku stopu – oczekiwanie na ciągłą poprawę z cierpliwością. **(500)**
Ilość osobników w turnieju	Nie powinna być zbyt duża, gdyż wykluczy ona mutację. Najlepiej wybierać do kilku osobników **(5)**

Odpowiedzi na pytania pomocnicze:

1. W przypadku turnieju na jednego osobnika, stanie się ona selekcją ruletkową z równym prawdopodobieństwem dla każdego osobnika. Może sprawić, że odrzucimy dobrego osobnika. Dobre dla eksploracji, złe dla eksploatacji.
2. W przypadku turnieju o rozmiarze równym populacji wybieramy tylko najlepszego osobnika, a gorsze nie mają możliwości bytu. Poprawia eksploatacje, znacznie pogarsza eksploatacje.
3. Co ciekawe ważniejsza jest mutacja. Poruszony został temat w analizie wcześniej.
4. Może być za mało mutacji, ale też maksymalna ilość mutacji negatywnie wpływa na całą populacje, oprócz tego jednego najlepszego osobnika.
5. Może być tylko za mało krzyżowania
6. Może być za dużo osobników w populacji co spowalnia cały algorytm. Jednakowoż mała ilość osobników wpływa na prędkość zbiegania algorytmu.
7. Coraz większa liczba pokoleń wpływa pozytywnie na algorytm GA, poza problemem zbyt długiego wykonywania.

Ostatni problem

Algorytmy zostały ostatni raz przetestowane na ogromnym zbiorze danych. GA zostało dostrojone wg testów tj. ilość iteracji została zwiększona do 500, liczba osobników pozostała taka sama, Px zostało zwiększone do 0.8. Natomiast Pw do poziomu 0.4. Turniej nie uległ zmianom.



Jak widać algorytm zachłanny dalej króluje, GA stara się go nadgonić, natomiast najgorzej radzi sobie losowe przeszukiwanie rozwiązań.