Sztuczna inteligencja

i

inżynieria wiedzy

1. **Wstęp**

**Cel ćwiczenia**

Zapoznanie się z metaheurystyką algorytmow genetycznych w praktyczny sposob poprzez

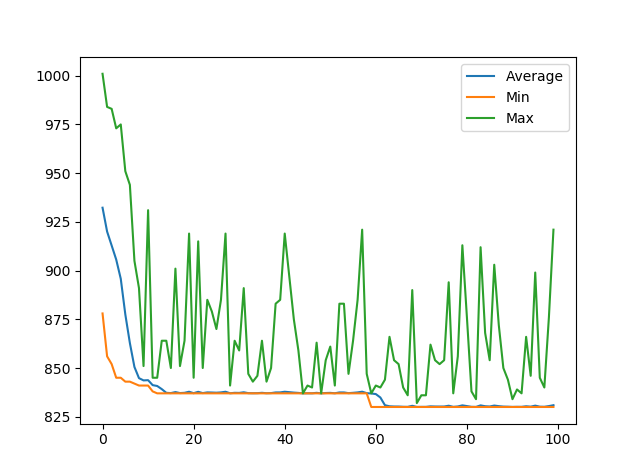
samodzielną implementację.

**Realizacja ćwiczenia**

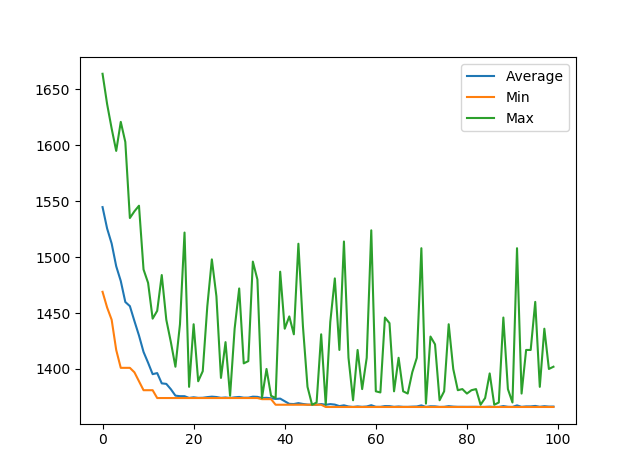
* Zapoznanie się z metaheurystyką algorytmow genetycznych
* Określenie problemu optymalizacyjnego do rozwiązania – minimalizacja W kwadratowy
* problemie przydziału QAP (*Quadratic Assignment Problem*)
* Zbudowanie algorytmu genetycznego: osobnik, funkcja oceny, krzyżowanie, mutacja,
* selekcja i inicjalizacja
* Implementacja modelu w dowolnym języku obiektowym (sugerowana Java, C/C++/C#, ew.
* python)
* Zbadanie wpływu rożnych parametrow (prawd. mutacji *Pm*, krzyżowania *Px*, selekcji,
* rozmiar populacji *pop\_size*, liczba pokoleń *gen*) na efektywność i skuteczność metody
* Sporządzenie sprawozdania z ćwiczenia
* Pokazanie na wykresach zmianę wartości przystosowania (wartość optymalizowanej funkcji
* celu) w poszczegolnych pokoleniach: najlepszy osobnik, średnia wartość w populacji i
* najgorszy osobnik
* Porownaj działanie algorytmu genetycznego z wybranymi przez siebie, nieewolucyjnymi
* metodami optymalizacji (np.: metoda losowego przeszukiwania, algorytm zachłanny).
* Proszę zwrocić uwagę na uzyskany wynik, czas działania oraz liczbę wartościowań
* optymalizowanej funkcji celu. Proszę pokazać i omowić najciekawsze wyniki
* Raport z ćwiczenia powinien zawierać wszystkie punkty wymagane w realizacji zadania

1. **Wyniki**
2. Przebieg zmiany funkcji kosztu dla kolejnych generacji

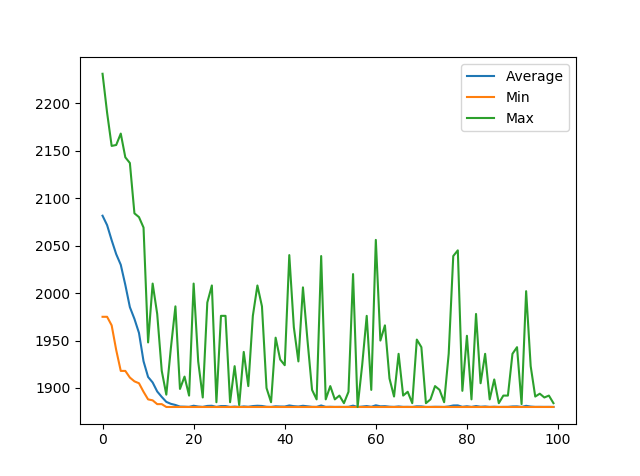
Rozwiązanie zadania przetestowano dla pięciu zestawów danych. Poniżej zamieszczono wyniki kolejno dla 12, 14, 16, 18, 20 fabryk w problemie QAP oraz dla px = 0,7, pm = 0,01, pop\_size = 100, tour = 5. Do selekcji zastosowano metodę turniejową.



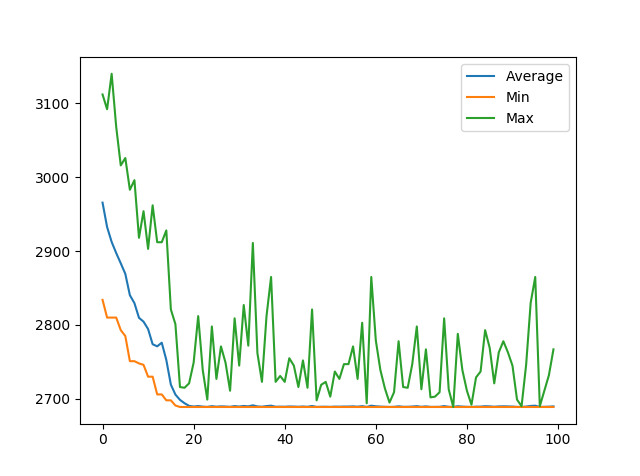
*Rys.1 Średnia, minimalna i maksymalna wartość funkcji kosztu dla kolejnego pokolenia dla zestawu danych nr 12*



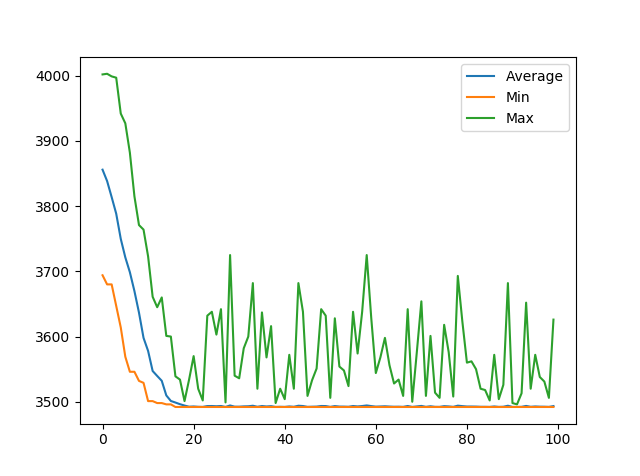
*Rys.2 Średnia, minimalna i maksymalna wartość funkcji kosztu dla kolejnego pokolenia dla zestawu danych nr 14*



*Rys.3 Średnia, minimalna i maksymalna wartość funkcji kosztu dla kolejnego pokolenia dla zestawu danych nr 16*



*Rys.4 Średnia, minimalna i maksymalna wartość funkcji kosztu dla kolejnego pokolenia dla zestawu danych nr 18*

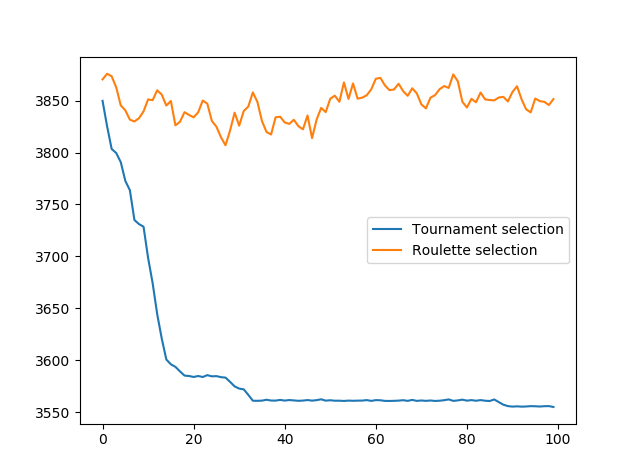


*Rys.5 Średnia, minimalna i maksymalna wartość funkcji kosztu dla kolejnego pokolenia dla zestawu danych nr 20*

1. Porównanie metody turniejowej i ruletki dla zestawu danych nr 20

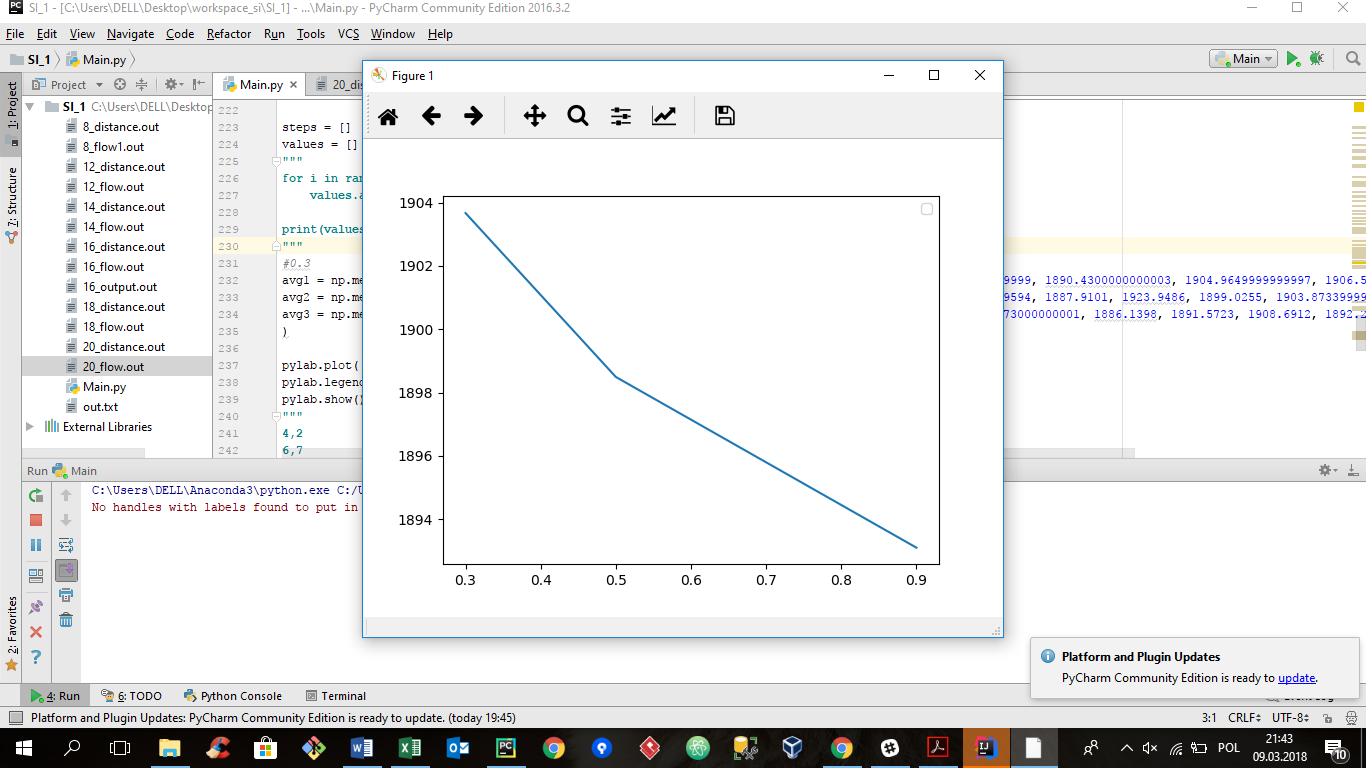
Na wykresie przedstawiono zmiany w przebiegu funkcji kosztu dla kolejnej generacji dla konkretnej metody selekcji. Ponadto średni czas liczenia dla tych metod jest różny: 4, 2s dla metody turniejowej, 6, 7s dla metody ruletki.

*Rys.6 Porównanie metody turniejowej i ruletki*



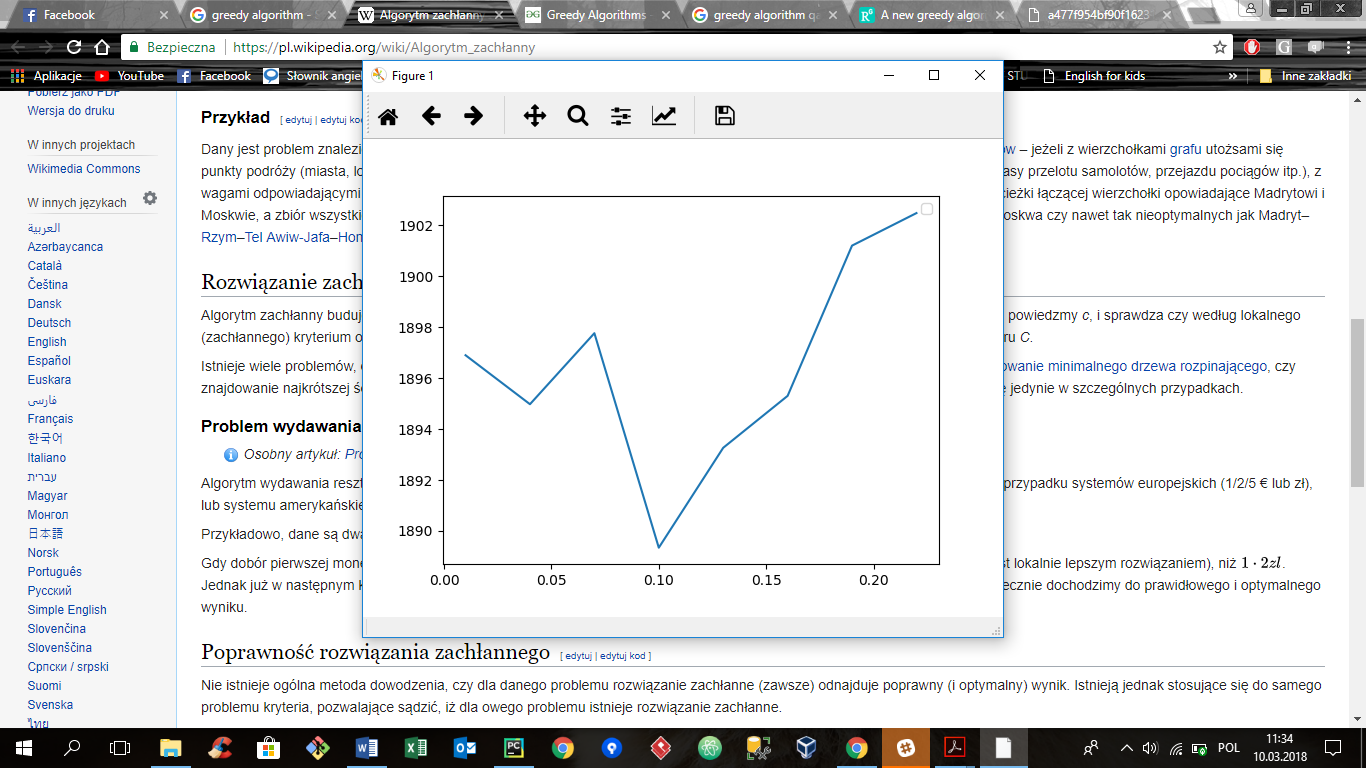
1. Porównanie średniej wartości funkcji kosztu dla różnych wartości parametru px

dla zestawu danych nr 16



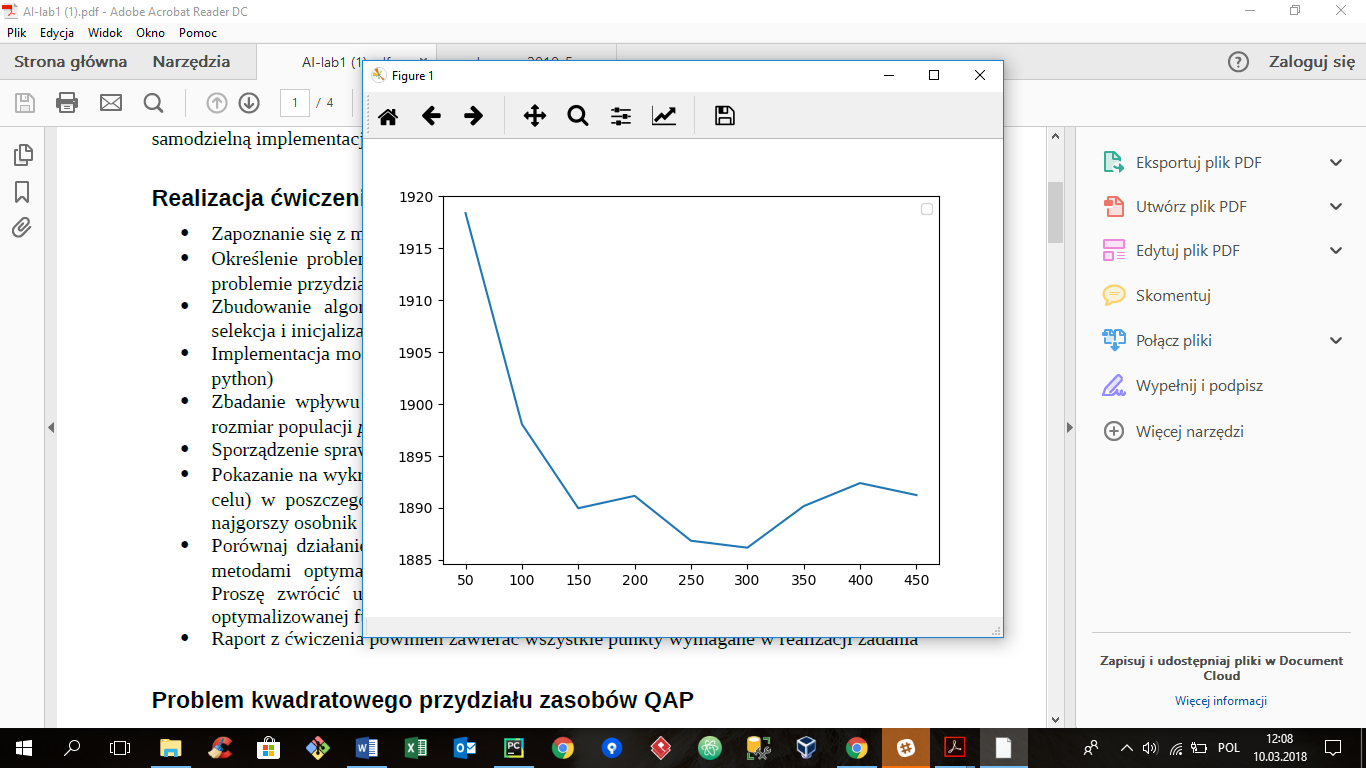
*Rys. 7 Badanie wpływu px na działanie GA*

1. Zbadanie wpływu mutacji na wynik działania GA dla zestawu danych nr 16



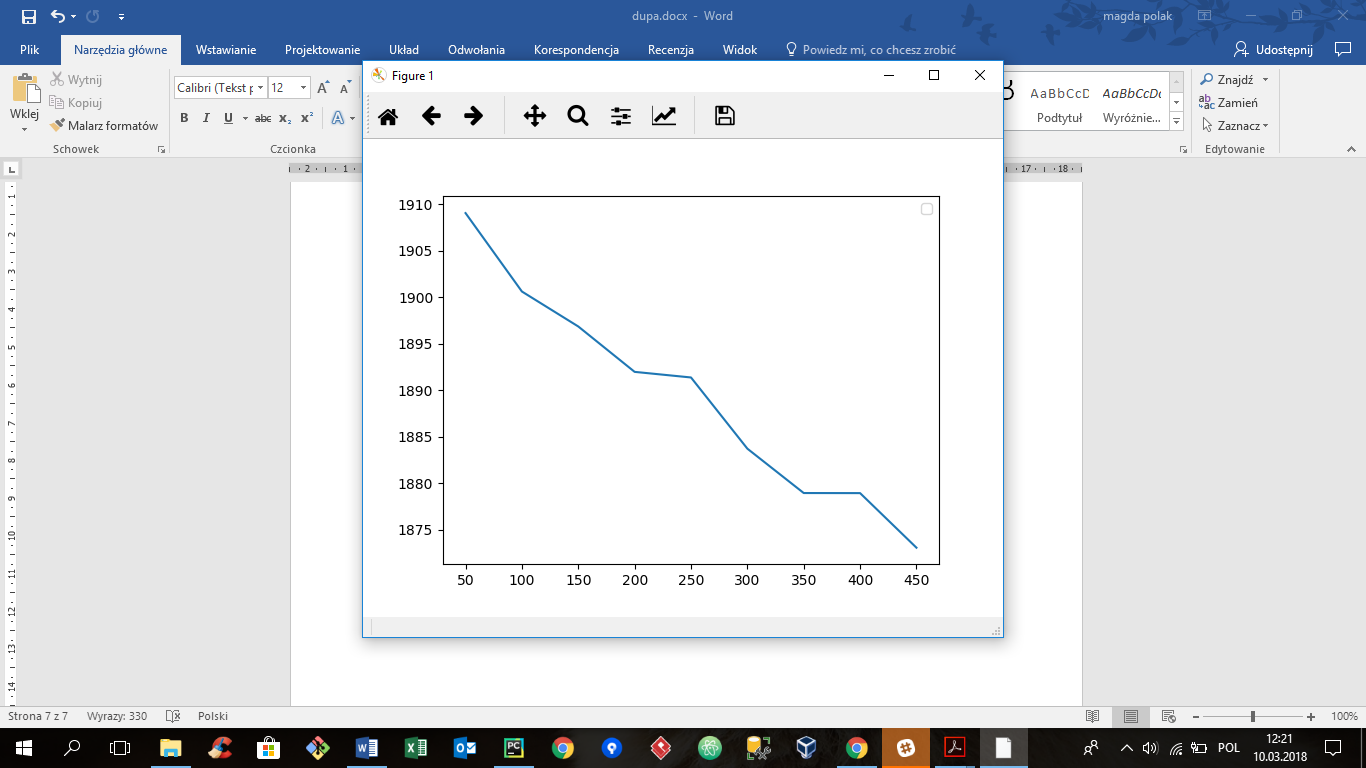
*Rys. 8 Wpływ parametru pm na działanie GA*

1. Wpływ rozmiaru populacji na działanie GA



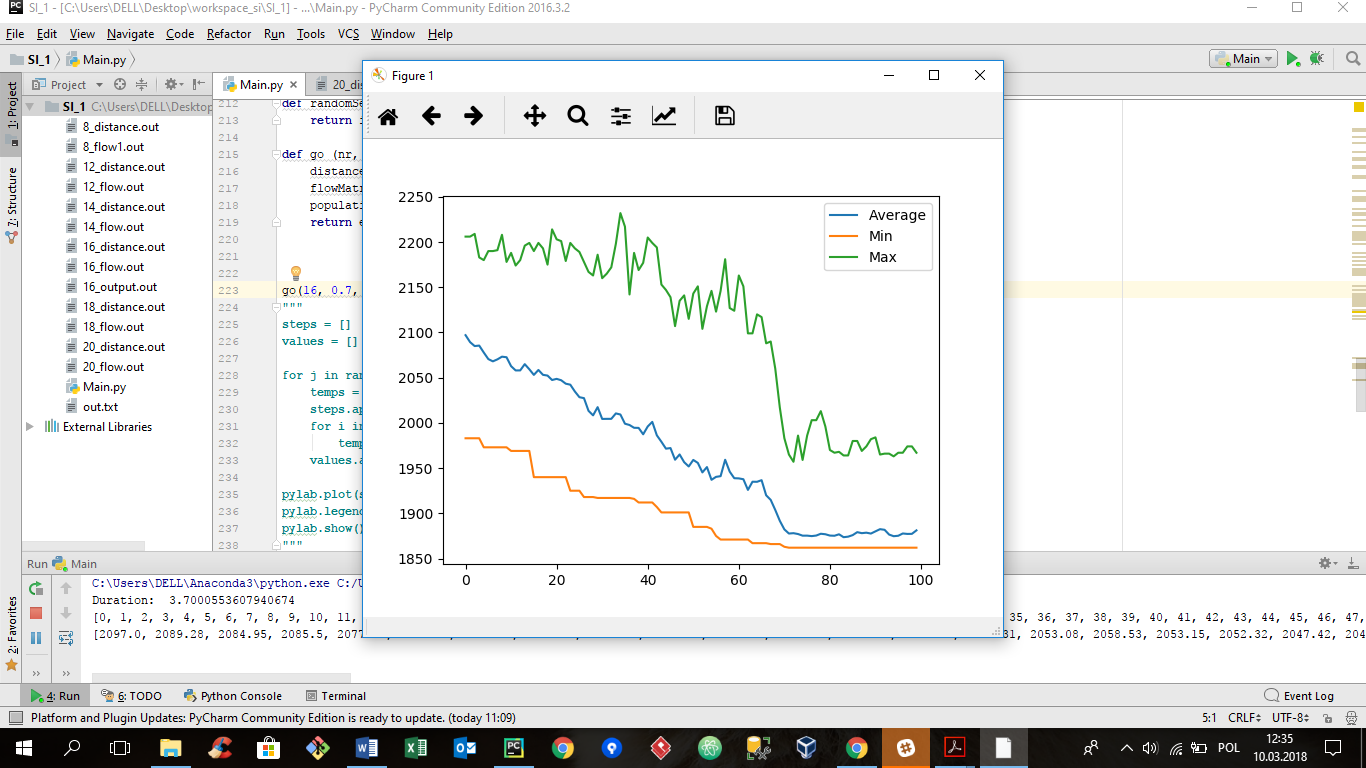
*Rys. 9 Wpływ parametru pop\_size na działanie GA*

1. Wpływ liczby kolejnych generacji na działanie GA



*Rys. 10 Wpływ liczby kolejnych pokoleń na działanie GA*

1. Przebieg działania GA dla rozmiaru turnieju równego 1, px=0,7, pm=0.0



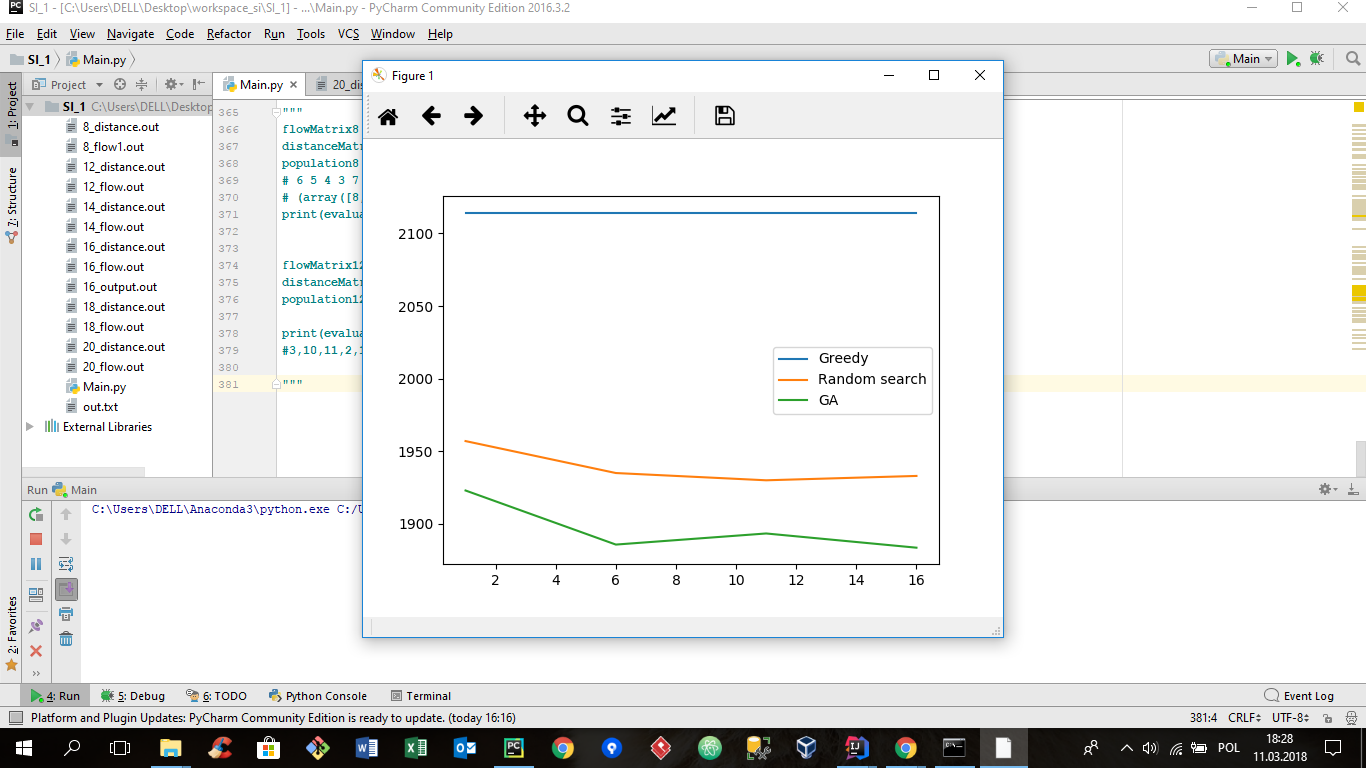
*Rys. 11 Przebieg działania GA dla rozmiaru turnieju równego 1, px=0,7, pm=0.0*

F) Przebieg działania GA dla rozmiaru turnieju równego 1, px=0.0,pm = 0.05



*Rys. 12 Przebieg działania GA dla rozmiaru turnieju równego 1, px=0,0, pm=0.05*

1. Porównanie działania algorytmu genetycznego z algorytmem zachłannym oraz algorytmem przeszukiwania losowego



*Rys. 13 Zależność funkcji kosztu od czasu działania dla trzech różnych algorytmów*

1. Wnioski

Algorytm genetyczny dobrze radzi sobie ze znajdowaniem rozwiązania bliskiego optymalnemu. Zależy to jednak od wielu parametrów. Im większy rozmiar populacji tym większe prawdopodobieństwo, że algorytm genetyczny odnajdzie rozwiązanie optymalne. Wiąże się to jednak z czasem działania tej heurystyki. Podobnie jest też z maksymalną liczbą kolejnych pokoleń. Im większa liczba generacji algorytmu generycznego, tym względnie lepsze jest jego rozwiązanie. Rozwiązanie zależy również od parametów px oraz pm. Badając wykresy zauważono, iż im większe jest prawdopodobieństwo krzyżowania, tym algorytm lepiej działa. Zaś w przypadku prawdopodobieństwa mutacji, wywnioskowano, że mutacja dobrze wpływa na rozwiązanie, jednak do pewnego momentu. W rozważanym przypadku stwierdzono, że prawdopodobieństwo wyższe niż 10%, wraz z jego wzrostem negatywnie wpływa na wynik.

Nie należy również zapomnieć o metodzie selekcji. W rozważanym problemie stwierdzono, że metoda turniejowa jest nie tylko lepsza od metody ruletki, ale również szybsza. Ponadto odpowiednia liczba osobników w metodzie turniejowej również znacząco wpływa na wynik.

Przy porównywaniu działania algorytmu genetycznego z dwoma algorytmami nieewolucyjnymi, wywnioskowano, że GA jest bez porównania lepszy od algorytmu zachłannego jak i od algorytmu przeszukiwania losowego. Algorytm zachłanny znajduje tylko lokalne rozwiązanie, zaś Random Search jest niestabilny.