Spis treści

Wstęp

Celem ćwiczenia było zaiplementowanie algorytmu genetycznego w języku python. Algorytm ten należy do grupy algorytmów ewolucyjnych, a jego zadaniem jest wyznaczenie najlepszego osobnika (rozwiązania) dla zadanego problemu.. Do analizy działania algorytmu wykorzystam problem związany z optymalizacją układu dróg i miejsc parkingowych. Przestrzeń pzreszukiwań dla tego problemu to wszystkie u-elementowe wektory zero-jedynkowe, gdzie u to ustalony na początku rozmiar osobnika.

Sposób działania algorytmu

Algorytm genetyczny jest przeznaczony dla zadań o charakterze dyskretnym i kombinatorycznym (wartości zmiennych xi przyjmują wartości 0 lub 1). Rozróżniamy w nim osobników (pojedyncze wektory cech złożone z 0 oraz 1), które tworzą populację. Algorytm przez t_max iteracji w odpowiedni sposób przekształca zadaną populację początkową, szukając w jej kolejnych wariacjach, które generuje poprzez mutacje i krzyżowanie, najlepszego osobnika.

Hiperparametry

Hiperparametry, których użyłam w swojej implementacji to:

- populacja początkowa P0,

- liczba osobników w populacji u

- prawdopodobieństwo mutacji osobnika pm

- prawdopodobieństwo krzyżowania osobnika pc

- liczba pokoleń, czyli maksymalna liczba iteracji t_max

Przykładowy dobry wynik

Oto przykładowy, całkiem dobry wynik dla poniższego doboru hiperparametrów.

Liczba osobników w populacji: 600

Prawdopodobieństwo mutacji osobnika: 0.9

Prawdopodobieństwo krzyżowania osobnika: 0.1

Liczba pokoleń: 500

Wynik: 69

```
{'mutation probability': 0.1, 'crossover probability': 0.9, 'max iterations': 500}
Individuals number: 600
Result:  69
```

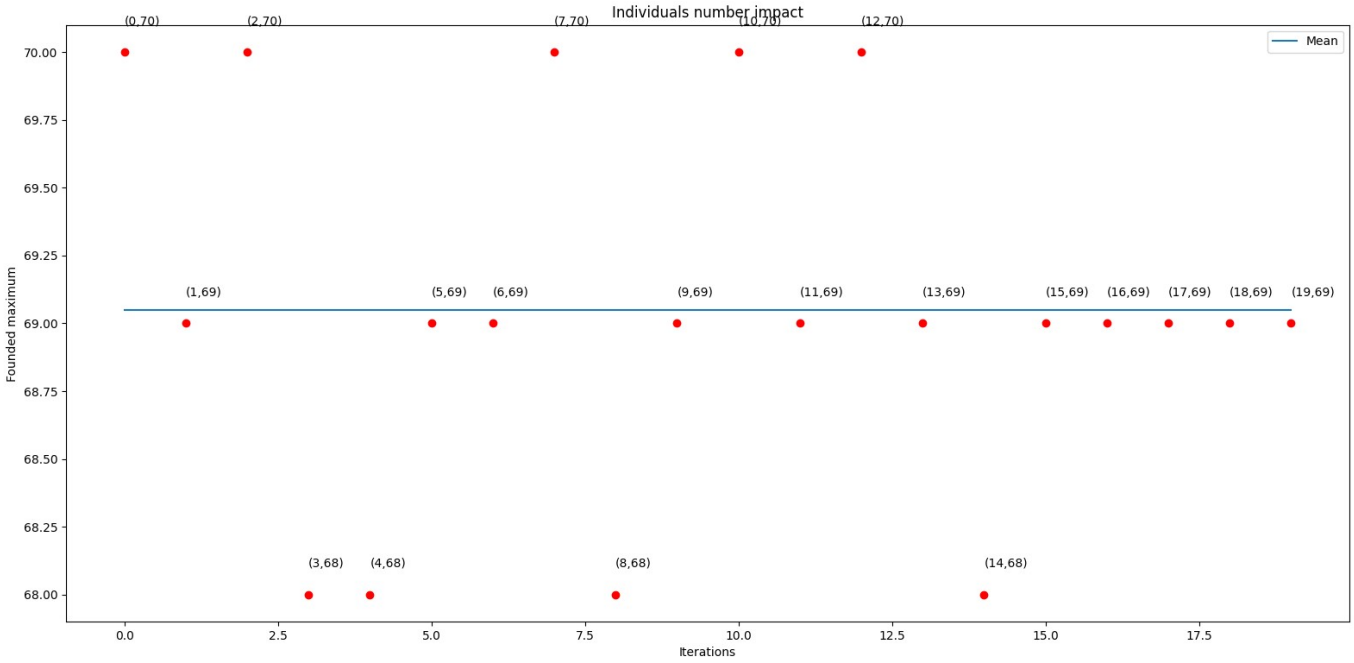
Wpływ liczby osobników

Stałe hiperparametry: pm = 0.1, pc = 0.8, t_max = 500

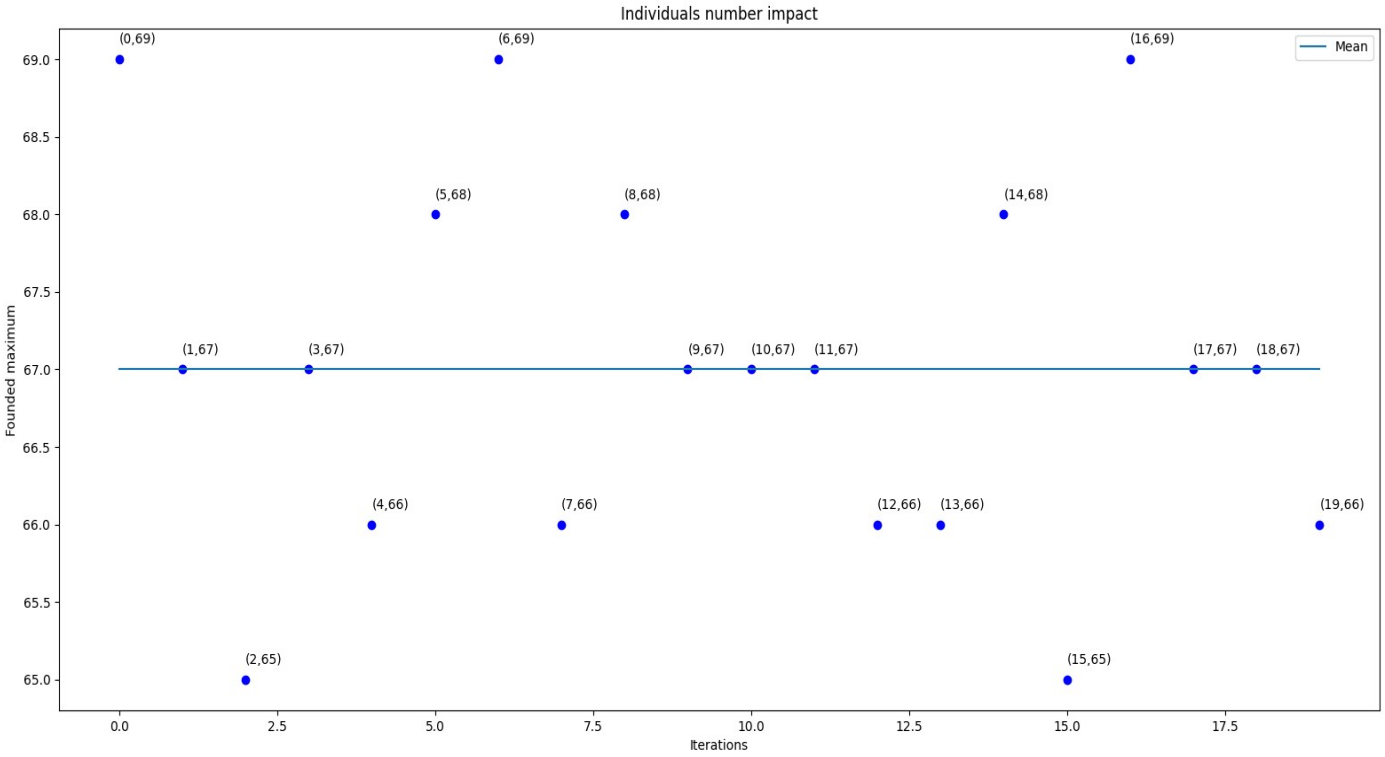
Populacja losowa

Eksperymenty

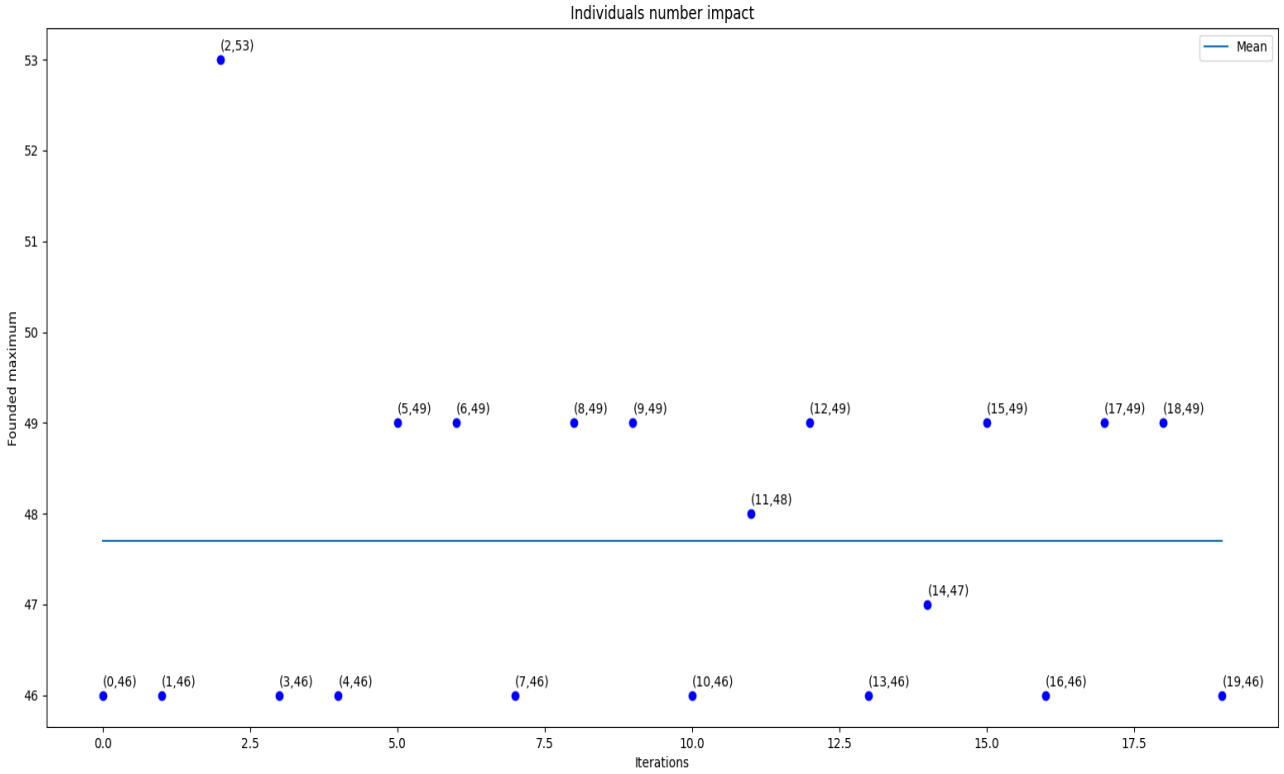
Duża liczba osobników u =500:



Średnia liczba osobników u = 250:



Mała liczba osobników u = 5:



Wnioski

Im populacja jest większa, tym prawdopodobieństwo znalezienia maksimum globalnego jest większe. Jak widać na powyższych eksperymentach, wynik dla 500 (średnio ~69) osobników jest lepszy, niż dla 5 (średnio ~48) oraz 250 (średnio ~67) osobników. Tak samo dla 250 osobników wyniki działania algorytmu są lepszy, niż dla 5 osobników. Dla 5 osobników, w 20 iteracjach

maksymalna znaleziona wartość osobnika to jedynie 53, podczas gdy dla 250 jest to 67, a dla 500 - 69.

Oczywiście wynik działania nie rośnie w nieskończoność, a do maksimum globalnego funkcji – w przypadku tego problemu, dla 1000 osobników wychodzi wynik 72, tak samo jak dla 2000 osobników (dalszy wzrost populacji najprawdopodobniej nie da lepszego wyniku dla zadanych pozostałych hiperparametrów, a jedynie bezsensownie wydłuży czas szukania). Widać także, że dla 500 osobników wynik nie wychodzi dużo lepszy, niż dla 250 (początki stagnacji wyniku).

Dokładność wyznaczania maksimum osiągana jest jednak kosztem czasu wykonania programu – dla 5 osobników czas wykonania jest bardzo krótki, podczas gdy dla 1000 znacząco dłuższy. Jest tak głównie ze względu na to, że algorytm wykorzystuje iterowanie po każdy osobniku przy selekcji, mutacji oraz krzyżowaniu.

Warto zwrócić uwagę na rozbieżności w wyniku działania algorytmu. Zmienna wartość wyniku dla różnych iteracji, widoczna na wykresach powyżej, wynika z losowości algorytmu – przy krzyżowaniu mutacji i selekcji. Oprócz tego, w może się zdarzyć, że dla pewnej kombinacji krzyżowania / populacji początkowej, algorytm “utknie” w pewnej populacji i nie będzie w stanie poprawić jej wyniku. Dlatego na wykresie zaznaczono jeszcze dodatkowo średnią, aby lepiej oddać wpływ rozpatrywanego hiperparametru.

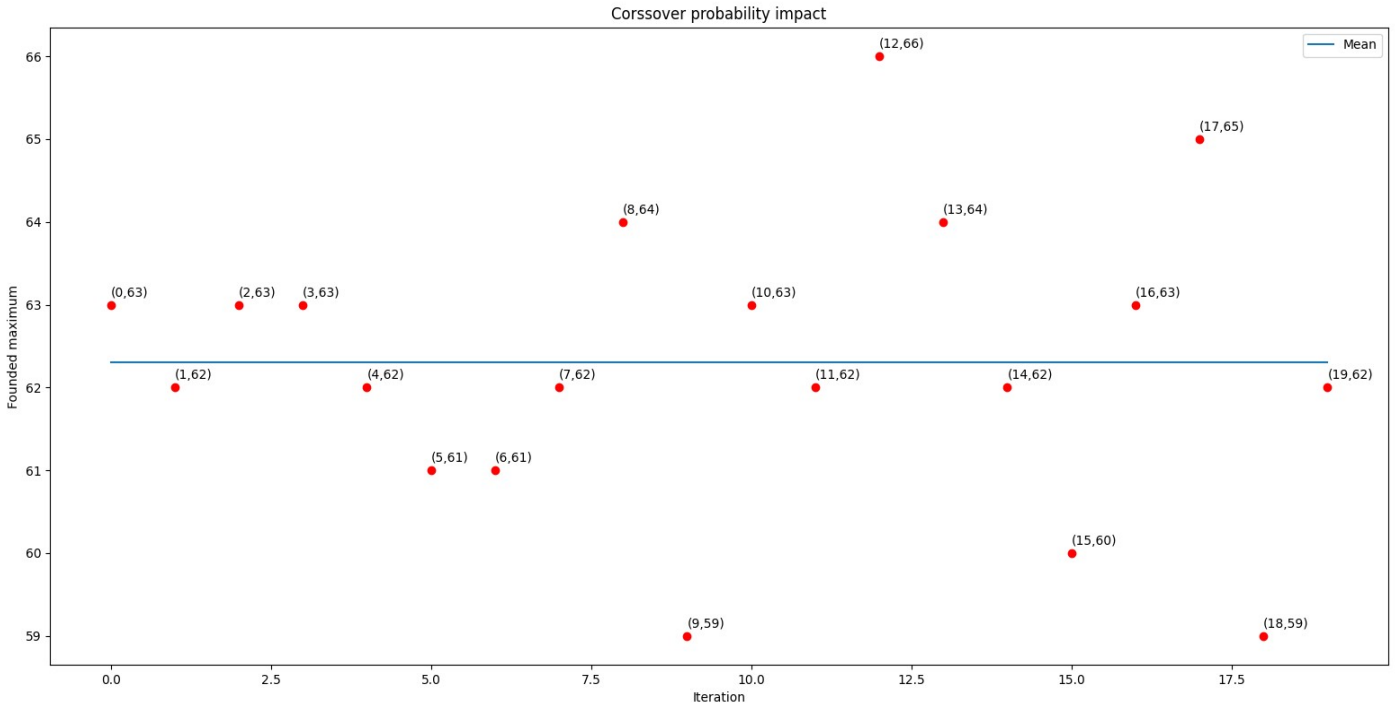
Wpływ prawdopodobieństw mutacji i krzyżowania

Stałe hiperparametry: u=50 (aby zmniejszyć czas oczekiwania na wynik), t_max = 500

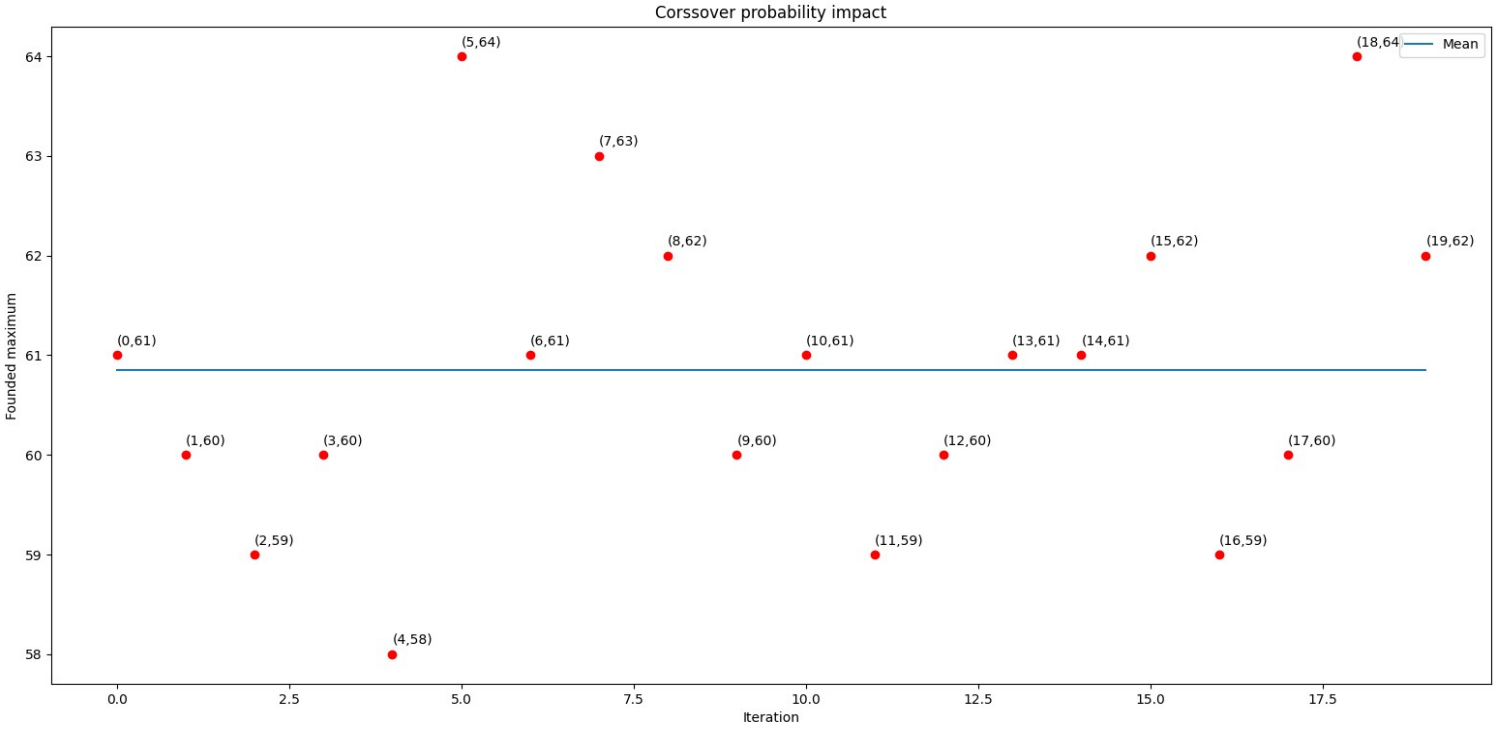
Populacja losowa

Eksperymenty

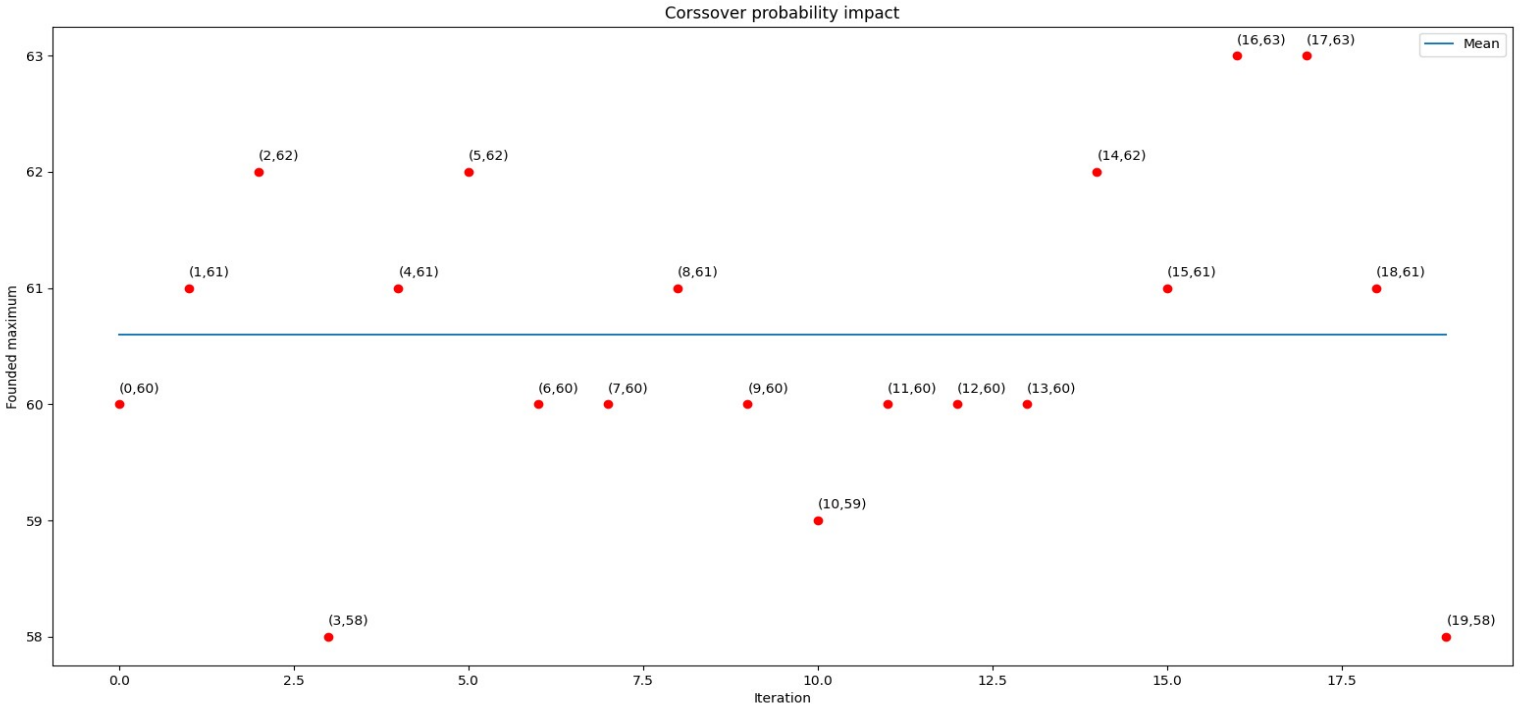
Bardzo małe prawdopodobieństwo mutacji pm=0.01 i bardzo duże prawdopodobieństwo krzyżowania pc=0.99:



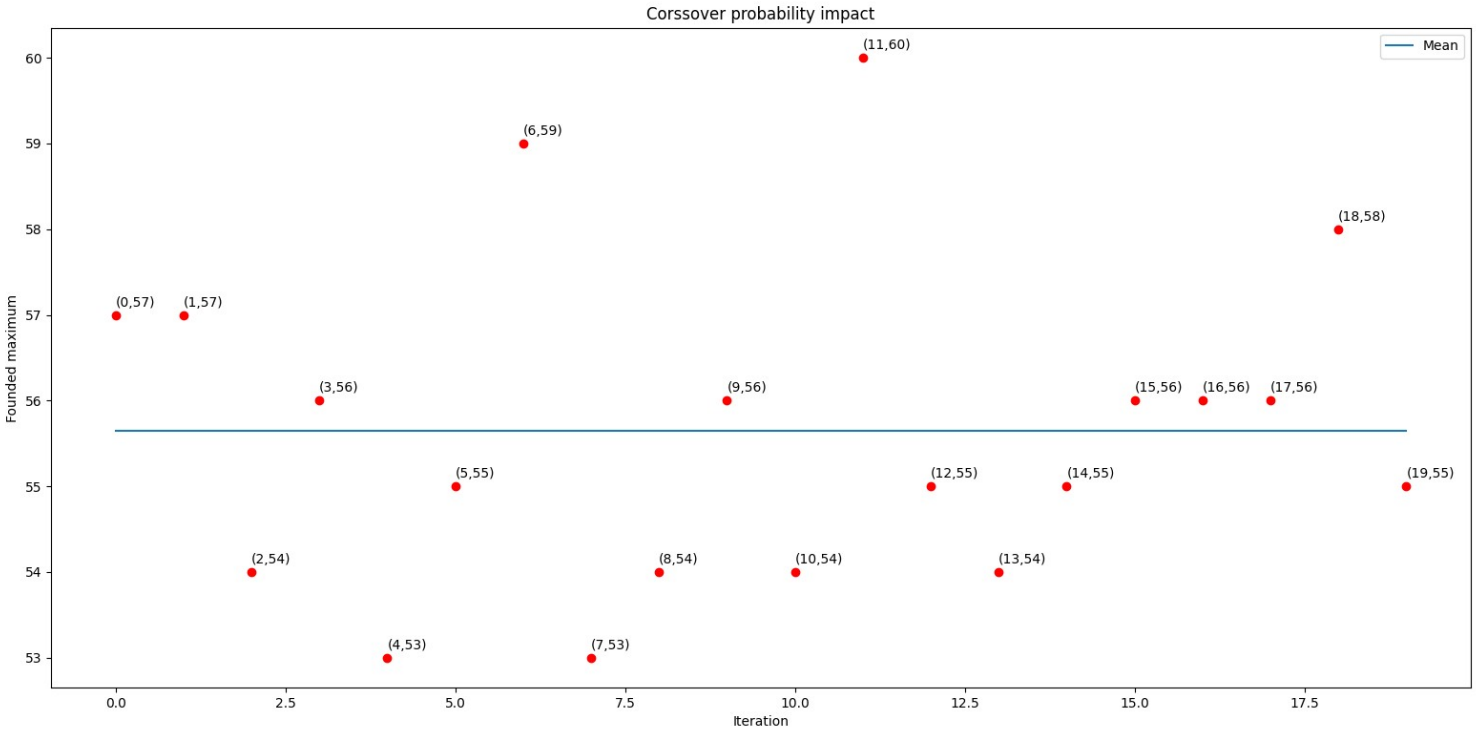
Małe prawdopodobieństwo mutacji pm=0.1 i duże prawdopodobieństwo krzyżowania pc=0.9:



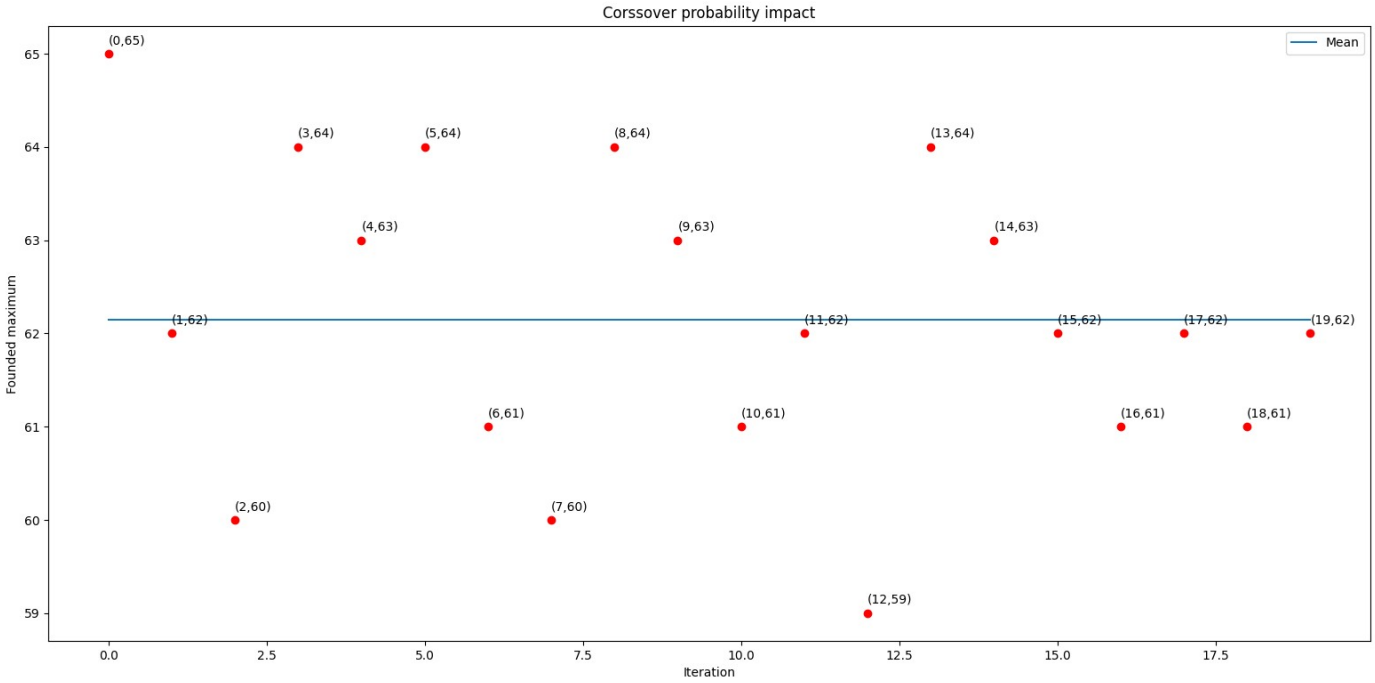
Średnie prawdopodobieństwo mutacji pm=0.3 i średnie prawdopodobieństwo krzyżowania pc=0.7:



Duże prawdopodobieństwo mutacji pm=0.8 i małe prawdopodobieństwo krzyżowania pc=0.2:



Zerowe prawdopodobieństwo mutacji i 100% pradopodobieństwa krzyżowania



Wnioski

Wynik działania algorytmu dla bardzo dużego prawdopodobieństwa krzyżowania i bardzo małego prawdopodobieństwa mutacji okazał się być najlepszy. Wraz ze zmianą proporcji między tymi dwoma wielkościami, średni znaleziony wynik malał (w proporcji pm=0.8, pc=0.2 wyniósł jedynie

niecałe 56). Pokazuje to, że algorytmy genetyczne powinny skupiać się na krzyżowaniu, nie mutacji, bowiem kładą większy nacisk na znalezieniu najlepszego rozwiązania z otoczenia punktów statowych. Zbyt duża mutacja prowadzi do nadmiernego tracenia dobrych rozwiązań, które znalezione zostały dzięki krzyżowaniu.

Nie powinno się jednak całkowicie wykluczyć mutacji w algorytmie (proporcja pm=0, pc=1), ponieważ jest ona pewnego rodzaju zabezpieczeniem przed utknięciem w nieoptymalnej populacji. Widać to na ostatnim wykresie - choć średni wynik jest bardzo dobry (prawie na takim poziomie, jak dla pc = 0.95 i pm = 0.01), to znalezione punkty są pocno “rozstrzelone” po wykresie (na wykresie dla pc = 0.95 i pm = 0.01 też to trochę widać). Jest tak, ponieważ dla niefortunnego losowania osobników do krzyżowania może się zdażyć, że algorytm straci jakąkolwiek różnorodność i nie będzie w stanie poprawić dotychczasowego wyniku (np. gdy wszystkie osobniki będą takie same, potrzebna jest mutacja, aby program zaczął tą populację jakkolwiek zmieniać).

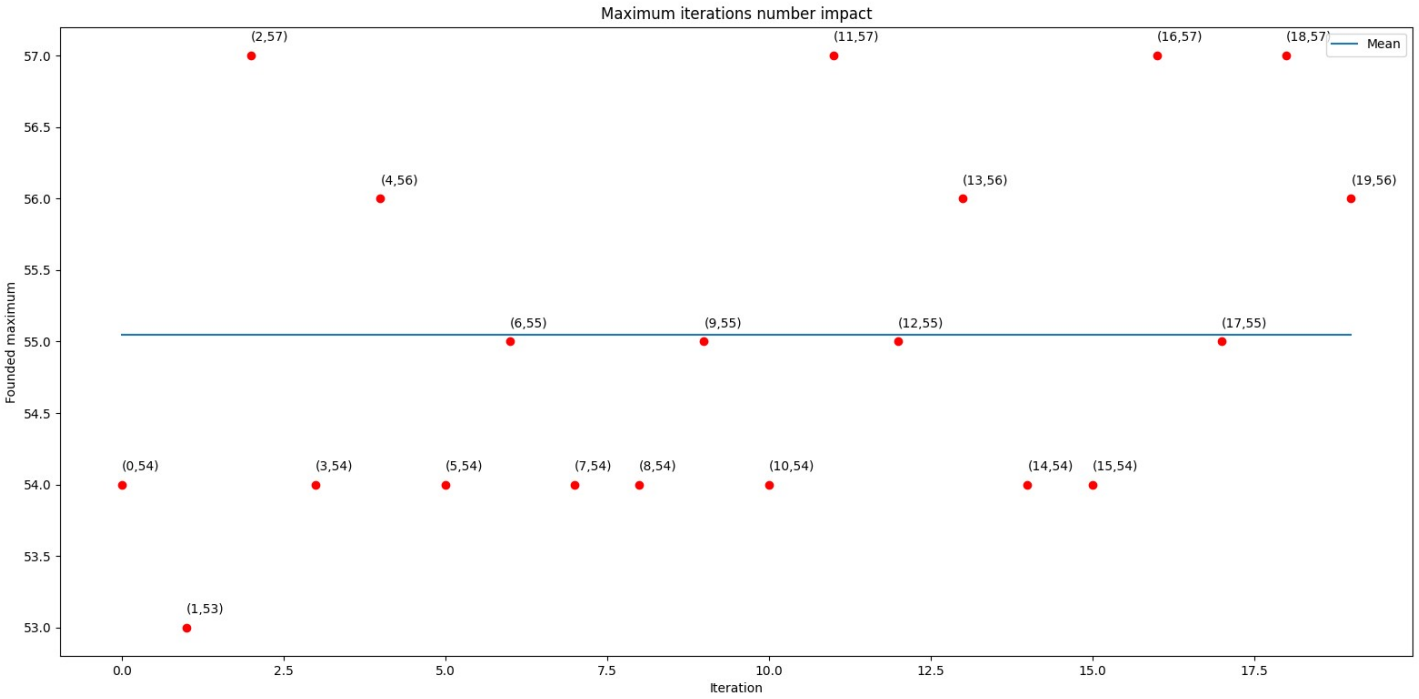
Wpływ liczby iteracji

Stałe hiperparametry: u=25 (aby zmniejszyć czas oczekiwania na wynik), pm=0.1, pc=0.9

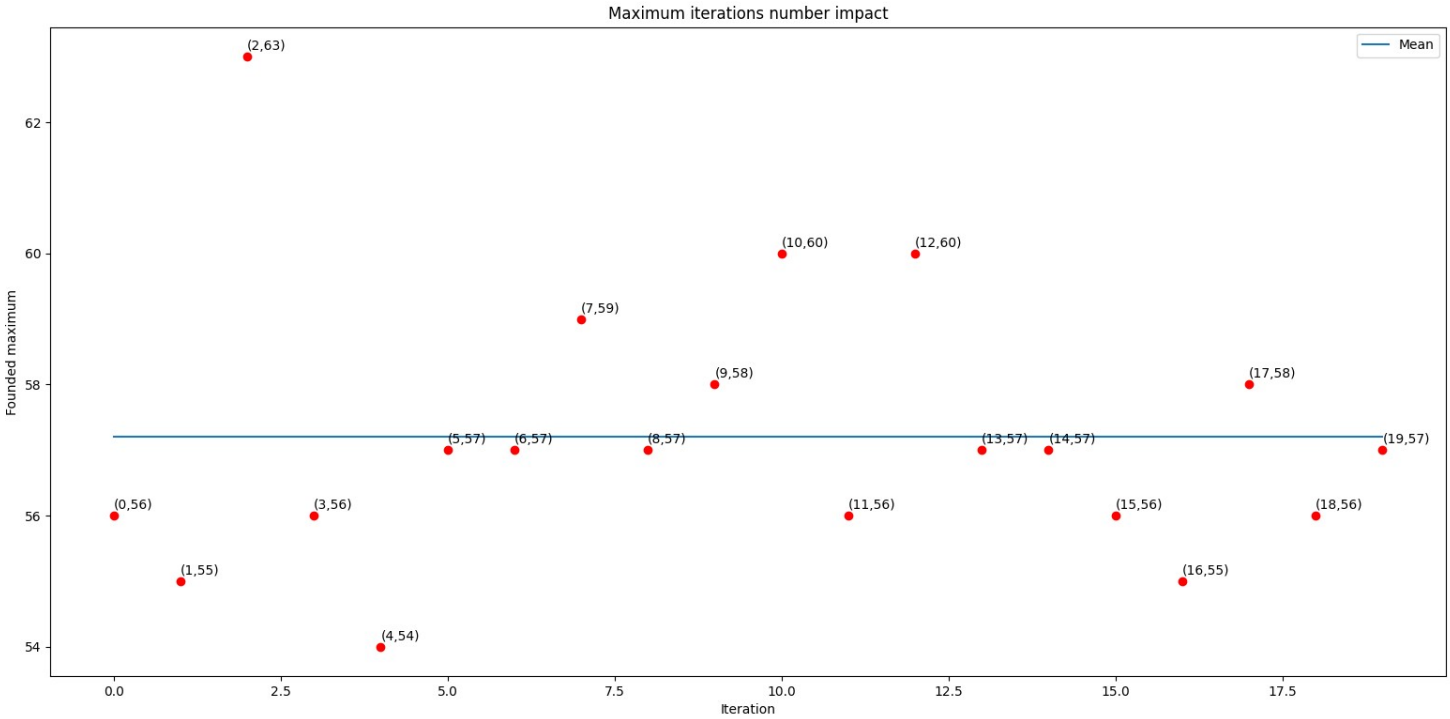
Populacja losowa

Eksperymenty

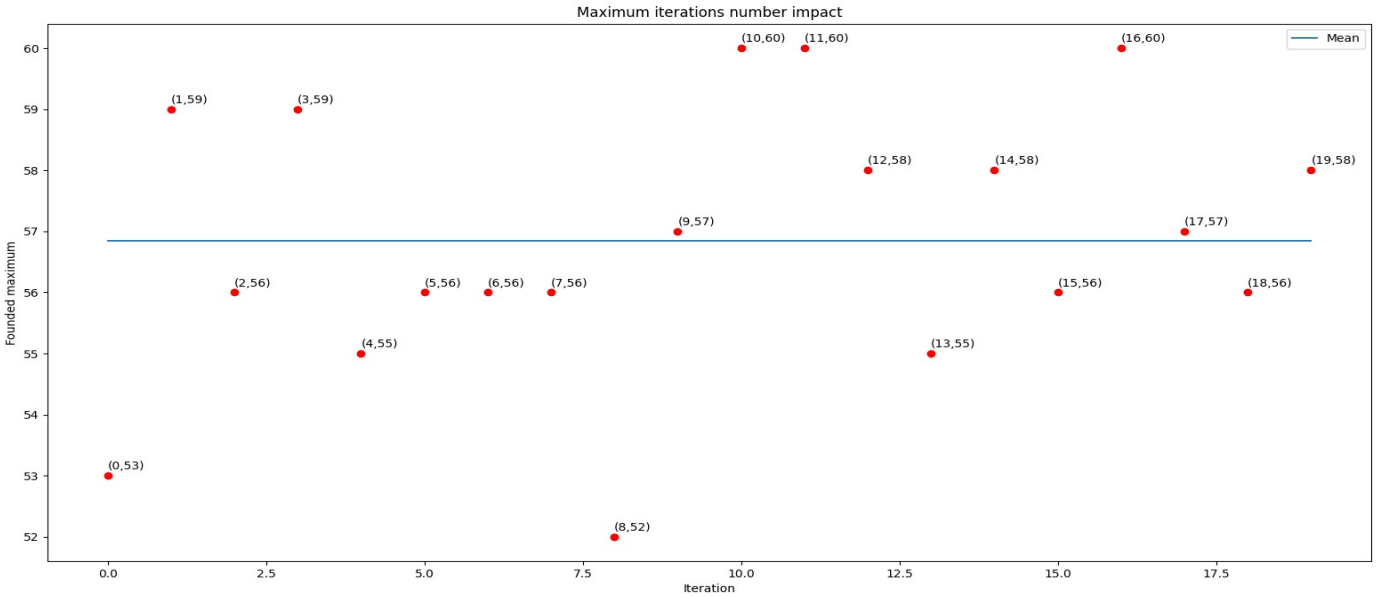
Mała liczba iteracji t_max = 25:



Średnia liczba iteracji t_max = 500:



Duża liczba iteracji t_max = 900:



Wnioski

Liczba iteracji odpowiada za liczbę rozpatrywanych pokoleń. Dla większej liczby iteracji algorytm znajduje lepsze maksimum (bardziej zbliżone dla maksimum globalnego). Jednak, podobnie jak w przypadku rozmiaru populacji, dzieje się to kosztem wydłużonego czasu wykonywania się programu.

Warto jednak zaznaczyć, że od pewnego poziomu maksymalnej liczby iteracji, wynik zaczyna się nie zmieniać, a czasami nawet pogarszać (tak jak to widać na wykresach powyżej - dla t_max=900, wynik okazał się być gorszy, niż dla t_max=500.). Jest to stagnacja algorytmu (nie jest on w stanie znaleźć lepszych rozwiązań, niezależnie od tego jak długo będzie szukać)