WSI Laboratorium 2 Raport

Algorytm genetyczny

Miłosz Andryszczuk 331355

1. Wykorzystane biblioteki

Do przeprowadzenia eksperymentu zostały wykorzystane następujące biblioteki:

- matplotlib
  do rysowania wykresów
  https://matplotlib.org/stable/index.html
- numpy
  do obsługi operacji matematycznych
  https://numpy.org/doc/

2. Opis eksperymentu

W ramach tego eksperymentu zbadano działanie algorytmu genetycznego. Algorytm znajduje przybliżone minimum funkcji, dzięki tworzeniu kolejnych generacji składających się z różnych rozwiązań danego problemu (argumentów funkcji celu). Dzięki odpowiedniej selekcji, każda generacja zawiera coraz lepsze rozwiązania.

W tym przypadku, funkcją celu jest funkcja generująca koszt energii zużywanej przez piec, którego stan (włączony/wyłączony) na przestrzeni czasu jest określony przez argument tej funkcji - listę składającą się z dwustu wartości binarnych.

Algorytm genetyczny znajduje taki ciąg binarny, który zapewnia jak najmniejsze zużycie energii przez piec.

Algorytm przyjmuje następujące hiperparametry:

- liczba generacji (max_iterations),
- liczba osobników w jednej generacji (population_size),
- prawdopodobieństwo mutacji (mutation_prob),
- prawdopodobieństwo krzyżowania (crossover_prob).

3. Przebieg Eksperymentu

W pierwszej kolejności przeprowadzono badanie wpływu poszczególnych hiperparametrów na wyniki optymalizacji.

3.1. Wpływ mutation_prob

Badania przeprowadzono dla jednej losowej populacji początkowej z następującymi hiperparametrami:

- max_iterations: 1000
- population_size: 100
- crossover_prob: 0.9

Zbadano 10 różnych wartości mutation_prob. Dla każdej wartości algorytm został uruchomiony 10 razy. Rezultat (średnia z tych 10 wyników) został przedstawiony na wykresie.

3.2. Wpływ crossover_prob

Podobnie jak w przypadku mutation_prob obliczono średni wynik optymalizacji dla 10 różnych wartości crossover_prob.

Użyte hiperparametry:

- max_iterations: 1000
- population_size: 100
- mutation_prob: 0.7

3.3. Wpływ population_size

Tym razem poza wartością średnią obliczono również odchylenie standardowe dla 5 różnych wartości population_size i wyniki przedstawiono na dwóch wykresach.

Użyte hiperparametry:

- max_iterations: 1000
- crossover_prob: 0.9
- mutation_prob: 0.7

3.4. Wybór najlepszych hiperparametrów

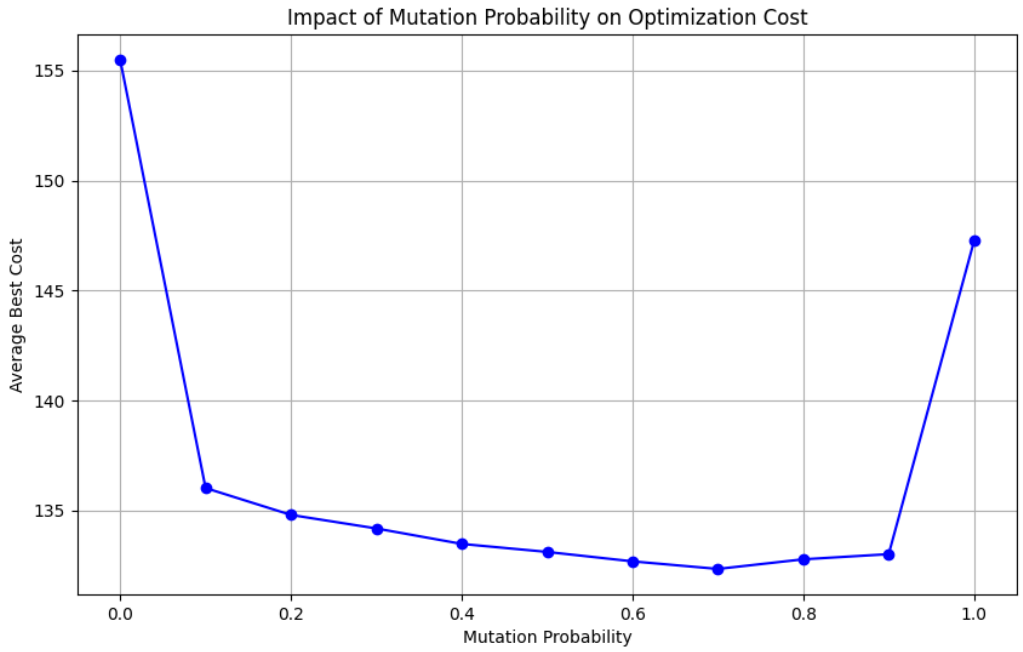
Na podstawie przeprowadzonych badań, wybrano zbiór hiperparametrów, który daje średnio najlepszy rezultat optymalizacji.

4. Wyniki

Na osi pionowej wykresów znajduje się średni koszt wyliczony przez algorytm z danymi hiperparametrami. Oznacza to, że niższa wartość na wykresie oznacza lepsze działanie algorytmu genetycznego.

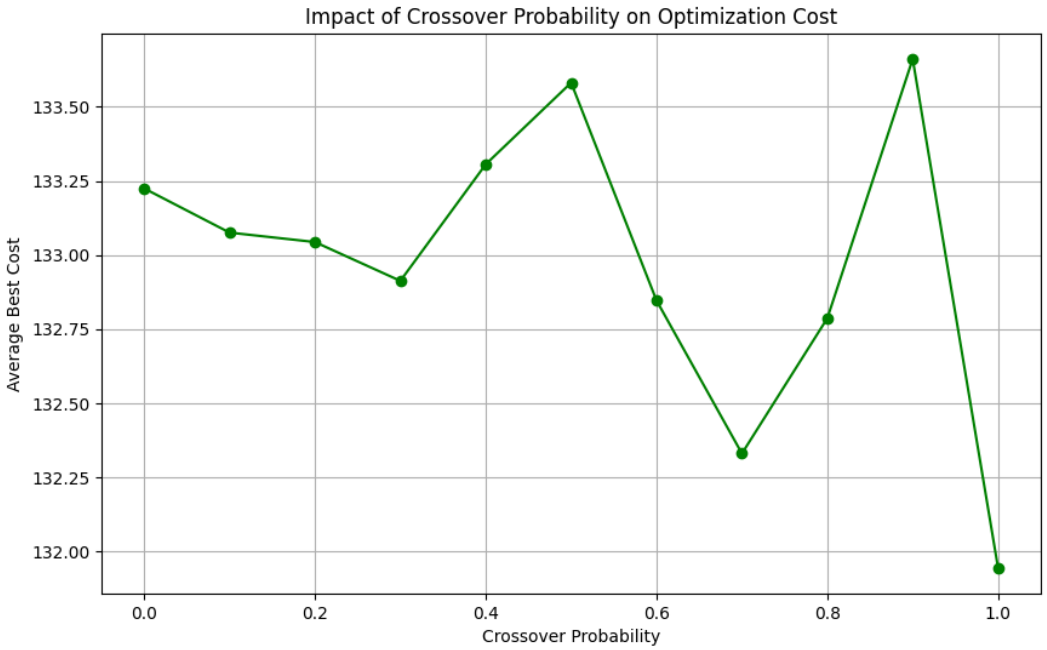
4.1. Wpływ mutation_prob

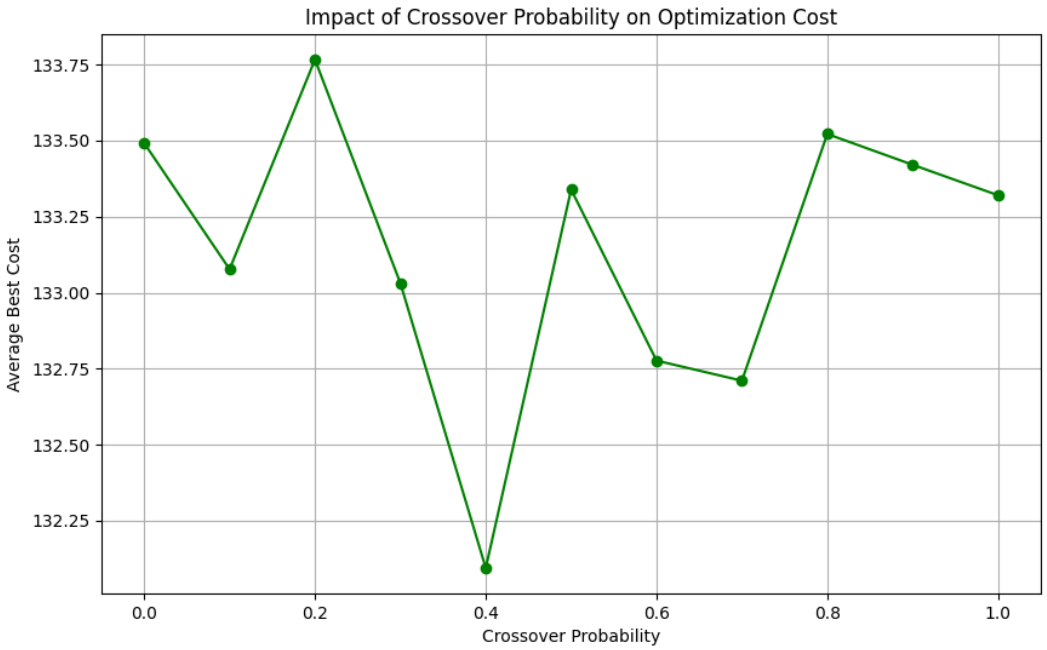
Badanie wykazało, że najlepsze prawdopodobieństwo mutacji wynosi 0.7, jednak również inne wartości mutation_prob w okolicach 0.5 dają dobre rezultaty. Zdecydowanie najgorzej wypada mutation_prob równe 0 i 1, czyli kolejno 0% i 100% szansy na mutację każdego z bitów danego osobnika.



4.2. Wpływ crossover_prob

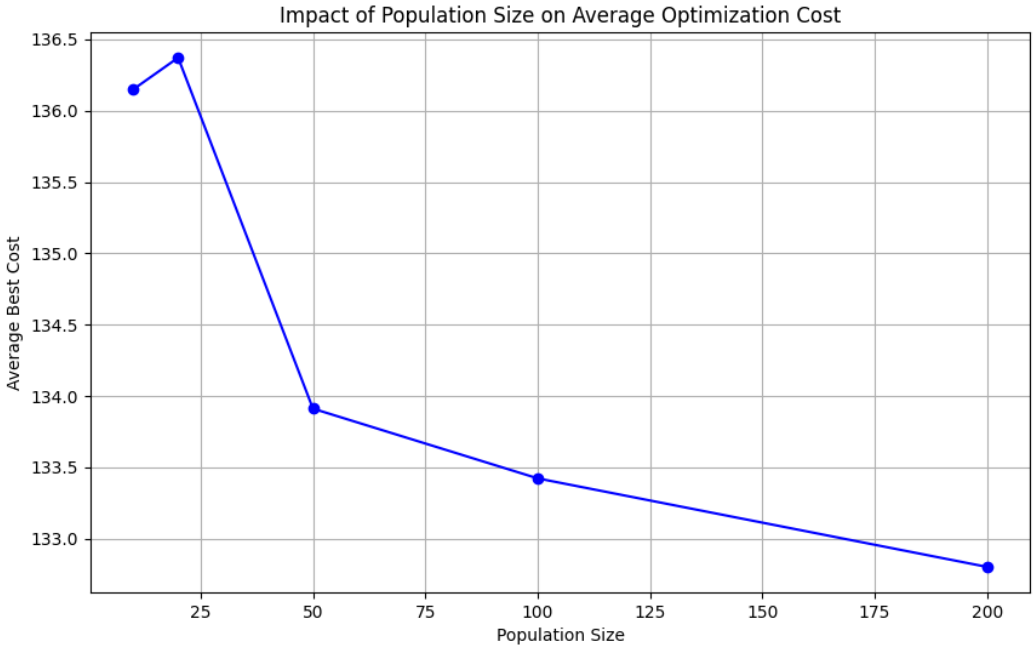
Eksperyment dla crossover_prob wykonano 2 razy. Wyniki te nie mają żadnych cech wspólnych. Jedyne oscylacje między wartościami kosztu dla kolejnych wartości crossover_prob wydają się być losowe, tym bardziej że są one nie większe niż 1.5. Wynika z tego, że crossover_prob nie ma większego wpływu na rezultat działania algorytmu genetycznego.



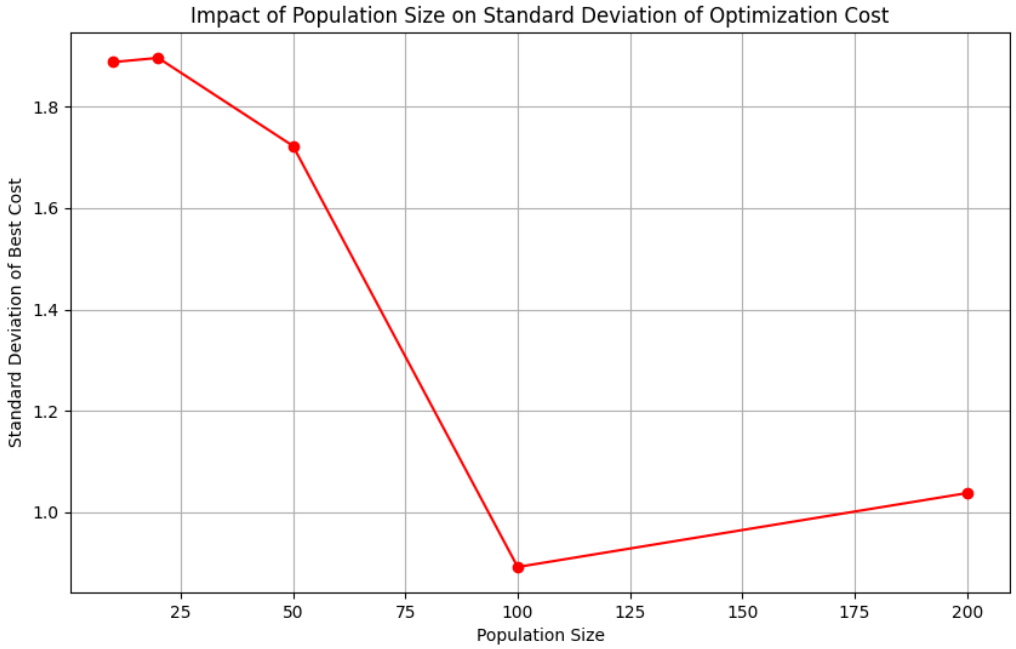


4.3. Wpływ population_size

Liczba osobników w generacji (population_size) wyraźnie wpływa na działanie algorytmu. Im wyższa ta wartość, tym więcej łącznie osobników jest tworzona i badana, dzięki czemu algorytm ma większe prawdopodobieństwo na odnalezienie lepszego rozwiązania. Ma to odzwierciedlenie na wykresie: jeśli algorytm tworzy małe populacje, w liczbie osobników rzędu 10, znajduje średnio gorsze rozwiązanie niż w przypadku liczniejszych generacji (100, 200 osobników).



Zwiększanie rozmiaru populacji zmniejsza również odchylenie standardowe. Oznacza to, że przy większych rozmiarach populacji algorytm uzyskuje bardziej stabilne (mniej zróżnicowane) wyniki o podobnej jakości, co zwiększa niezawodność algorytmu.



4.4. Najlepszy zestaw hiperparametrów

Najlepsza wartość mutation_prob to 0.7, natomiast crossover_prob zdaje się nie mieć większego wpływu na wyniki.

Wartości max_iterations = 1000 i population_size = 100 generują wystarczająco dobre rozwiązania (średni koszt poniżej 135) w rozsądnym czasie.

Dla podanych niżej parametrów:

- max_iterations: 1000
- population_size: 100
- crossover_prob: 0.9
- mutation_prob: 0.7

algorytm uruchomiony 20 razy zwracał średnio koszt równy: 132.7, natomiast najlepsze znalezione rozwiązanie miało koszt równy 129.7.

5. Wnioski

Z wyników eksperymentu wynika, że algorytm genetyczny jest bardzo wrażliwy na dobór hiperparametrów. Dla niewłaściwych parametrów algorytm znajdował rozwiązania o koszcie w okolicach 160, natomiast po dobraniu lepszych średni koszt zmalał do 130.

Najlepsza wartość prawdopodobieństwa mutacji wynosi około 0.5, a prawdopodobieństwo krzyżowania nie wpływa istotnie na wynik. Natomiast dobór liczby generacji i osobników w każdej z nich jest w dużej mierze uwarunkowane posiadanymi zasobami sprzętowymi. Im większa liczba osobników w populacji, tym algorytm znajduje lepsze minimum. To samo można powiedzieć o liczbie generacji . Niestety zwiększanie tych liczb znacząco zwiększa czas działania algorytmu.

Jeżeli więc zależy nam na znalezieniu jak najlepszego rozwiązania problemu, to wartości te powinny być jak największe. Jednak w przypadku, kiedy zależy nam na czasie, to trzeba się liczyć z faktem, że zmniejszanie tych wartości wiąże się z gorszymi wynikami algorytmu.