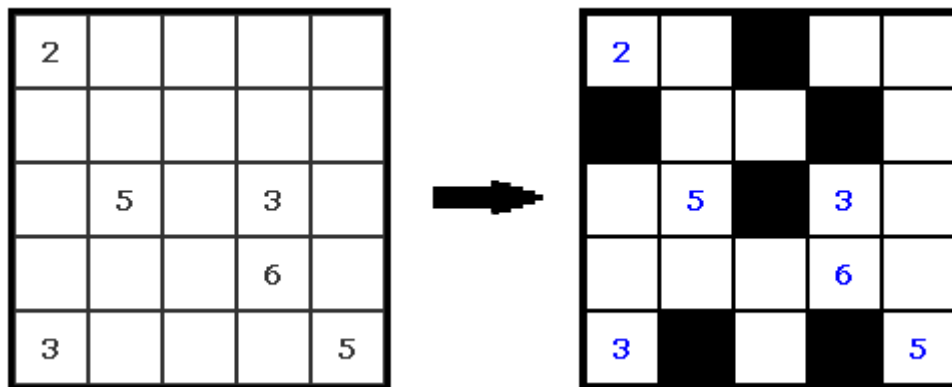


Kuromasu

Czym jest Kuromasu?

Kuromasu (kurodoko) to prostokątny diagram z liczbami w niektórych polach. Należy zaczernić niektóre pola diagramu tak, aby dla każdego pola liczbowego suma białych pól w tym samym wierszu i kolumnie ograniczona czarnymi polami lub brzegiem diagramu była równa wartości tej liczby. Trzeba przy tym przestrzegać zasady, że czarne pola nie mogą być połączone bokami, a wszystkie niezaczernione pola muszą tworzyć jedną spójną figurę.



Funkcja Fitness

Funkcję fitness stworzyłam wyliczając ile miejsc jest widocznych dla każdej liczby w łamigłówce kuromasu, następnie karząc algorytm za każdą pomyłkę. Im większa pomyłka tym większa kara jest przyznawana. To znaczy jeżeli dla liczby '5' jest widocznych '6' białych miejsc algorytm dostanie jeden minusowy (lub plusowy w przypadku algorytmu roju) punkt, jeżeli widocznych byłoby '10' miejsc zostanie przyznanych aż '5' punktów karnych.

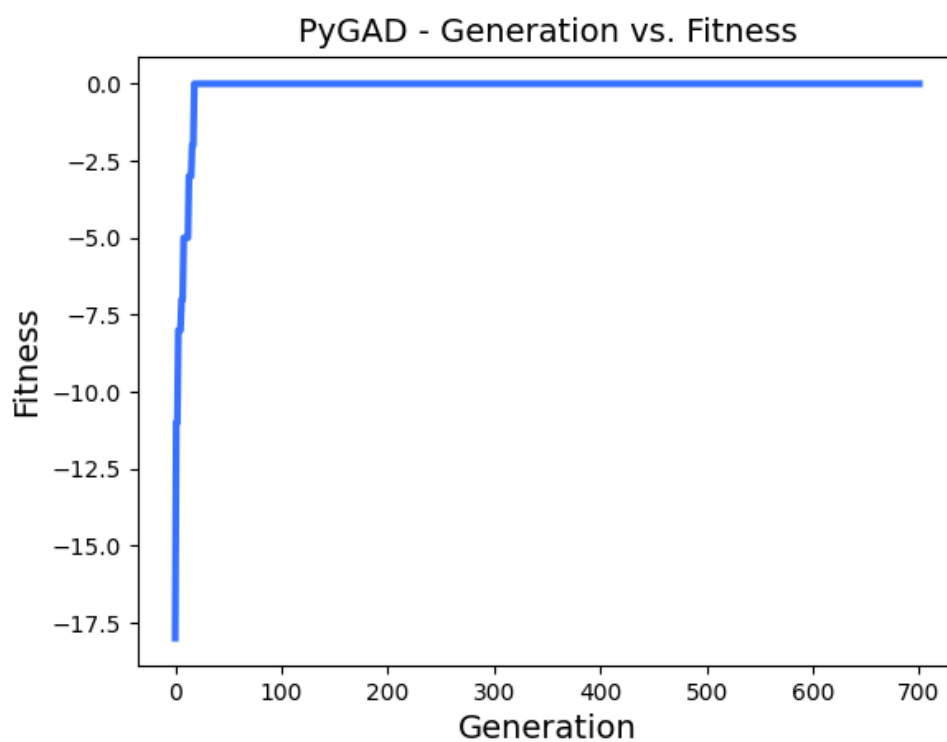
Poniżej znajduje się fragment kodu z funkcją fitness.

```
def fitness_func(solution, solution_idx):
    fitness = 0
    for row in range(len(kuromasu_five)):
        for column in range(len(kuromasu_five[row])):
            if isinstance(kuromasu_five[row][column], str):
                seen_in_row = []
                seen_in_column = []
                for i in range(len(solution)):
                    if row*5 <= i < (row+1)*5:
                        seen_in_row.append(solution[i])
                    if i-column in [0, 5, 10, 15, 20]:
                        seen_in_column.append(solution[i])
                if seen_in_row[column] == 1 or seen_in_column[row] == 1:
                    fitness -= int(kuromasu_five[row][column])
                else:
                    seen_in_row[column] = kuromasu_five[row][column]
                    seen_in_column[row] = kuromasu_five[row][column]
                    seen_count1 = 0
                    checked1 = False
                    seen_count2 = 0
                    checked2 = False
                    for i in seen_in_row:
                        if not checked1:
                            if i == 1:
                                seen_count1 = 0
                            if i == 0:
                                seen_count1 += 1
                            if isinstance(i, str):
                                checked1 = True
                                seen_count1 += 1
                        if checked1:
                            if i == 0:
                                seen_count1 += 1
                            if i == 1:
                                break
                    for i in seen_in_column:
                        if not checked2:
                            if i == 1:
                                seen_count2 = 0
                            if i == 0:
                                seen_count2 += 1
                            if isinstance(i, str):
                                checked2 = True
                        if checked2:
                            if i == 0:
                                seen_count2 += 1
                            if i == 1:
                                break
                    how_many_is_seen = seen_count1+seen_count2
                    diff = abs(how_many_is_seen -
int(kuromasu_five[row][column]))
                    fitness -= diff
    return fitness
```

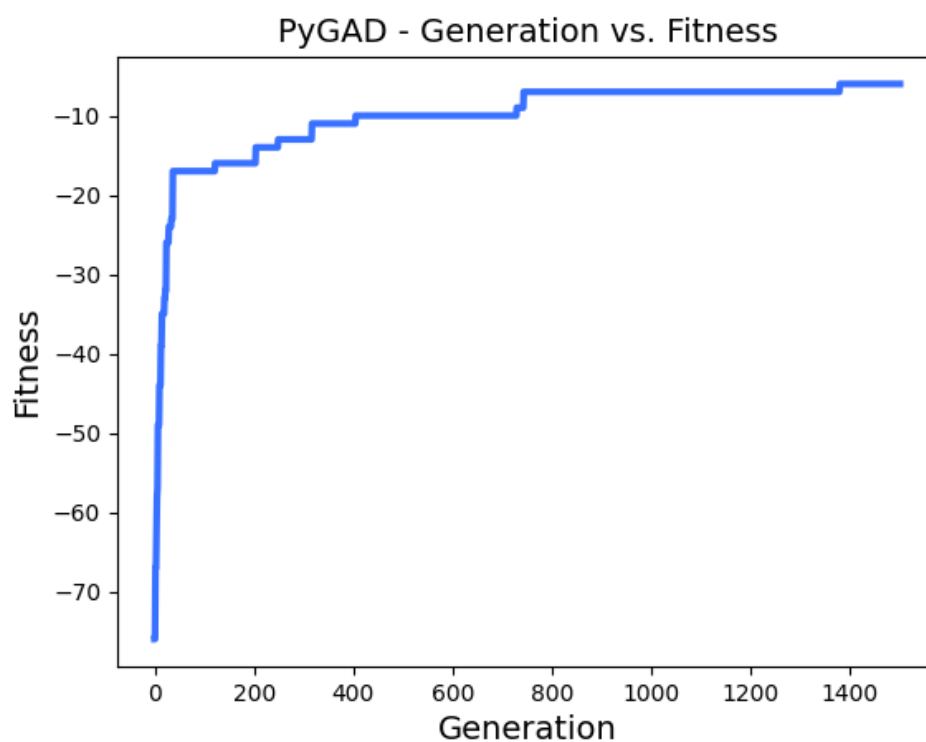
Algorytm Genetyczny

Algorytmy testowałam na łamigłówce dwóch różnych rozmiarów 5x5 oraz 11x11

Dla kuromasu 5x5 wystarczyło 700 generacji aby algorytm w 100% przypadków odnalazł prawidłowe rozwiązanie.



Dla kuromasu 11x11 algorytm radził sobie już dużo gorzej, przy zwiększeniu generacji do 1500, nadal nie odnajduje prawidłowego rozwiązania, funkcja fitness w najlepszych testach otrzymywała wyniki dochodzące do -2.



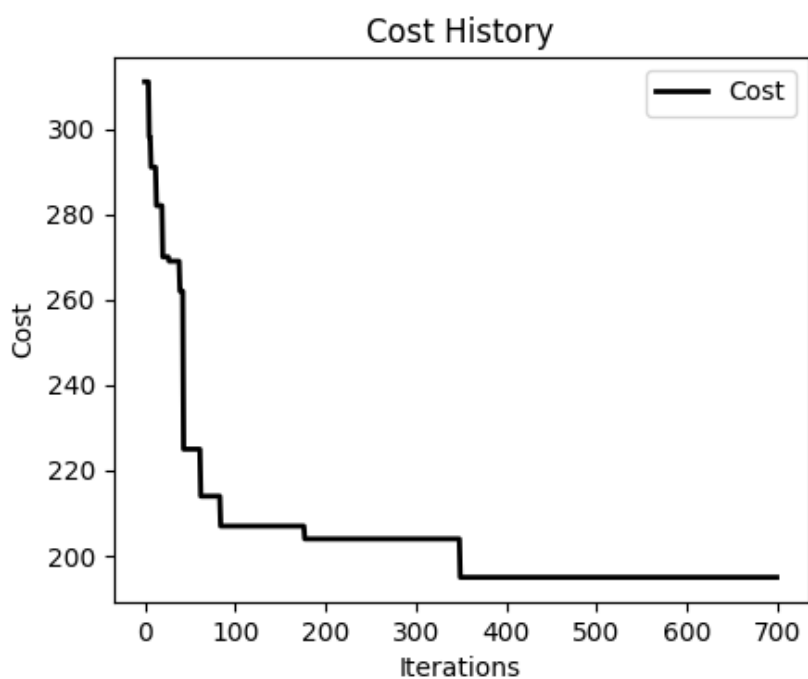
Jednocześnie wraz ze zwiększeniem wymiarów łamigłówki oraz generacji, zdecydowanie zwiększył się czas działania algorytmu

Czas

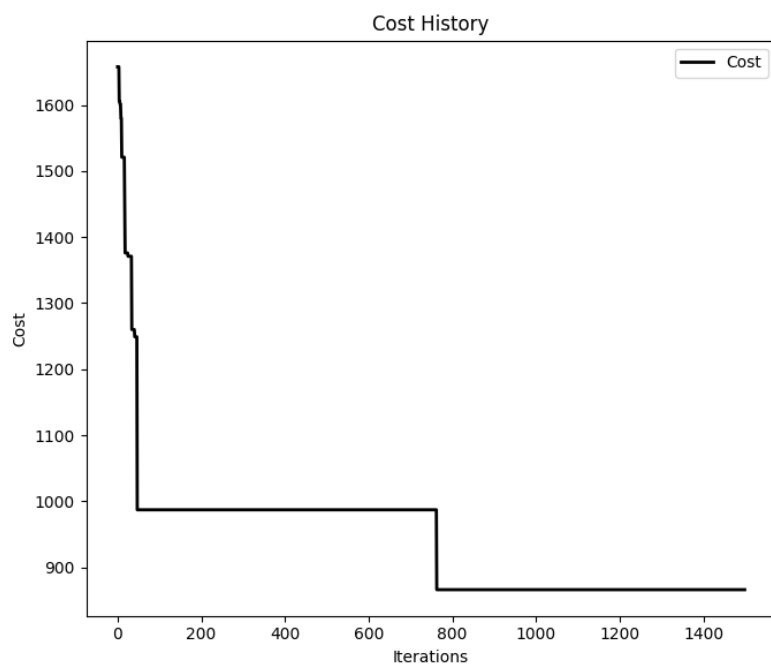
| 5x5 | 11x11 |
|------|-------|
| 1,5s | 62,8s |

Algorytm Roju

Dla algorytmu roju i łamigłówki 5x5 przy 700 iteracjach ani razu nie udało się uzyskać w 100% prawidłowego wyniku.



Podobnie w przypadku łamigłówki o wymiarach 11x11 i 1500 iteracjach.






Czas wykonania algorytmu roju podobny w przypadku kuromasu 5x5 i dużo szybszy w przypadku 11x11.

Czas

| 5x5 | 11x11 |
|------|-------|
| 1,2s | 17,3s |

Porównanie algorytmu Genetycznego oraz Algorytmu Roju

Dla zobrazowania różnic w osiągnięciach obu algorytmów przedstawiłam przykładowe wyniki na rysunkach.

-  Oznacza prawidłowo zamalowane pole
-  Oznacza pole które zostało zamalowanie nieprawidłowo.
-  Oznacza pole które powinno zostać zamalowane, ale niestety nie zostało

Kuromasu 5x5

| | | | | |
|---|---|---|---|---|
| | | | | 4 |
| | 3 | | 2 | |
| | | 2 | | 2 |
| 8 | | | | |
| | 3 | | 2 | |

Algorytm roju

| | | | | |
|---|---|---|---|---|
| | | | | 4 |
| | 3 | | 2 | |
| | | 2 | | 2 |
| 8 | | | | |
| | 3 | | 2 | |

Algorytm Genetyczny

Kuromasu 11x11

| | | | | | | | | | | |
|---|----|----|--|----|--|--|--|---|---|----|
| | | 9 | | | | | | 8 | | |
| | | | | | | | | 7 | | |
| | | | | 12 | | | | | | 16 |
| 9 | | | | | | | | | | |
| | 10 | | | | | | | | | |
| | | 12 | | 8 | | | | 3 | | |
| | | | | | | | | | 3 | |
| | | | | | | | | | | 3 |
| 7 | | | | | | | | | | |
| | | 7 | | | | | | | | |
| | | 2 | | | | | | 5 | | |

Algorytm Roju

| | | | | | | | | | | |
|---|----|----|--|----|--|--|--|---|---|----|
| | | 9 | | | | | | 8 | | |
| | | | | | | | | 7 | | |
| | | | | 12 | | | | | | 16 |
| 9 | | | | | | | | | | |
| | 10 | | | | | | | | | |
| | | 12 | | 8 | | | | 3 | | |
| | | | | | | | | | 3 | |
| | | | | | | | | | | 3 |
| 7 | | | | | | | | | | |
| | | 7 | | | | | | | | |
| | | 2 | | | | | | 5 | | |

Algorytm Genetyczny

Wnioski

Niestety żaden z algorytmów nie rozwiązuje problemu łamigłówki kuromasu w 100%. Jednak algorytm genetyczny radzi sobie z tym zdecydowanie lepiej zważając na to, że dla mniejszych wymiarów jest w stanie go rozwiązać, niestety jednak wraz ze skomplikowaniem problemu mocno rośnie czas jaki zajmuje wywołanie algorytmu, co sprawia że używanie go staje się nieopłacalne mimo możliwości otrzymania prawidłowych wyników.

W przypadku algorytmu roju czas potrzebny na wywołanie programu nie rośnie tak szybko, jednak szanse na otrzymanie rozwiązania przy rozsądnej liczbie iteracji są niewielkie co moim zdaniem sprawia, że użycie algorytm roju jest nie zasadne dla łamigłówki kuromasu niezależnie od wymiarów macierzy.

Autor: Natalia Turska