2 czerwca 2024

Raport Końcowy

Markov Chain Monte Carlo simulations of chromatin based on scHi-C

Autorzy:
Wiktor Woźniak
Sebastian Trojan
Aleksandra Kwiatkowska
Małgorzata Mokwa

Politechnika Warszawska

Spis treści

1 Wstep

Przedstawiony raport techniczny dotyczy projektu pt. "Markov Chain Monte Carlo simulations of chromatin based on scHi-C", który powstał w ramach przedmiotu Warsztaty Badawcze.

Celem projektu jest wykorzystanie techniki Markov Chain Monte Carlo (MCMC) do symulacji struktury chromatyny na podstawie danych pochodzacych z pojedynczych komórek (scHi-C). Chromatyna, składajaca sie głównie z DNA i białek histonowych, tworzy złożona strukture wewnatrz jadra komórkowego. Głównym celem projektu było zwizualizowanie sposobu, w jaki ta struktura jest zorganizowana.

2 Opis danych

Dane wykorzystane w tym projekcie pochodza z eksperymentu przeprowadzonego przez zespół badawczy z Instytutu Weizmanna w Izraelu. Eksperyment opierał sie na technice Single-cell Hi-C, która umożliwia badanie struktury chromosomów na poziomie pojedynczych komórek. Badacze wykorzystali próbki komórek jednojadrowych Mouse Th1 (myszy), które zostały poddane sekwencjonowaniu z obu końców. W sumie, do analizy właczono 10 próbek pojedynczych komórek oraz próbke zbiorcza wielokrotnego odczytu, razem z próbka Hi-C populacji.

W celu dokładniejszego zrozumienia danych, zespół wykonanał wizualizacje macierzy kontaktów poszczególnych chromosnomów. Poniżej przedstawna została maciecz wygenerowana z danych pochodzacych z chromosomu 1.

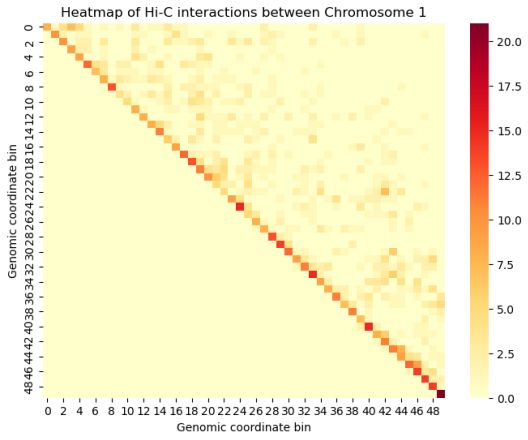

Figure 1: Mapa kontaktów dla chromosomu 1

3 Graf bazowy

W trakcie dalszego rozwoju projektu przygotowano graf bazowy, stanowiacy podstawe zwizualizowania modelu chromosomu. W tym celu napisana została funkcja generate_self_avoiding_walk, która pozwoliła nam stworzyć graf, unikajac nakładania sie na siebie fragmentów. Poniżej przedstawiamy wizualizacje grafu bazowego wykonana w Chimerze oraz kod generujacy random walk. UCSF Chimera to program do interaktywnej wizualizacji i analizy struktur molekularnych i powiazanych danych, w tym map gestości, zespołów supramolekularnych, dopasowań sekwencji, wyników dokowania, trajektorii i zespołów konformacyjnych.

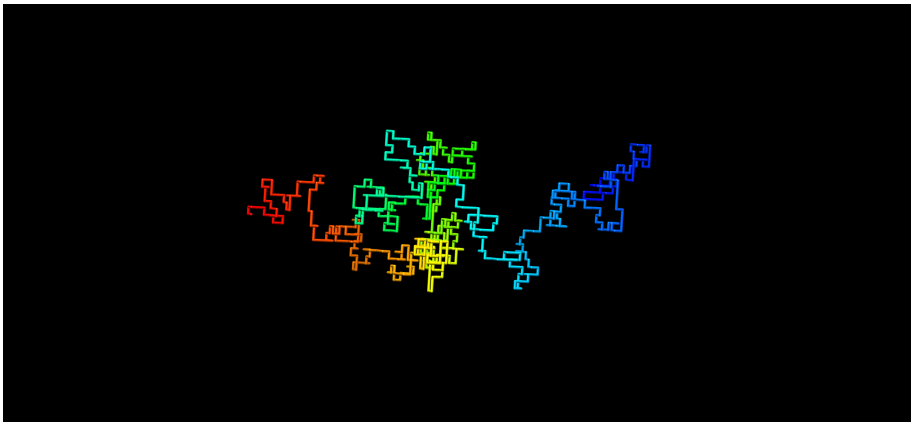

Figure 2: Model grafu bazowego

```python
def generate_self_avoiding_walk(max_steps: int, grid: int) -> list:
    """
    Generates a self-avoiding walk within a specified grid size.

    Parameters:
    max_steps (int): Maximum number of steps in the walk.
    grid (int): Grid size for the walk.

    Returns:
    list: List of coordinates representing the self-avoiding walk.
    """

    walk = [(0, 0, 0)]
    visited = set([(0, 0, 0)])

    moves = [(0, 0, 1), (0, 0, -1), (0, 1, 0), (0, -1, 0), (1, 0, 0), (-1, 0, 0)]

    for i in range(max_steps):
        dx, dy, dz = random.choice(moves)
        new_pos = (walk[-1][0] + dx, walk[-1][1] + dy, walk[-1][2] + dz)

        if any(abs(i) > grid for i in new_pos):
            continue

        if new_pos not in visited:
            walk.append(new_pos)
            visited.add(new_pos)

        else:
            continue
    return walk
```

4 Gestości funkcji proponujacej g i optymalizowanej f

Jednym z ważniejszych etapów projektu było zaimplementowanie dwóch funkcji: g i f.
Funkcja g:

dane wejściowe: graf, lista wezłów, trasa

dane wyjściowe: zmodyfikowana trasa oraz lista wezłów

Funkcja g służy do modyfikacju trasy, może wykonać kilka czynności: dodać nowy wezeł do trasy, wybieracjac pierwszego lub ostatniego sasiada. Może wstawić nowy wierchołek miedzy już dwa istniejace lub usunać losowo wybrany punkt. Funkcja wybiera czynność, która wykona, w sposób losowy.

- Prawdopodobieństwo dodania nowego wierzchołka na poczatku lub na końcu trasy: 0.4
- Prawdopodobieństwo dodania wierchołka pomiedzy dwa istniejace wierzchołki: 0.5
- Prawdopodobieństwo usuniecia losowego wierzchołka z trasy: 0.1

```
def g_function(graph: nx.graph, route: list, nodes: list)-> list:
"""
Modifies the route by either adding a new node, inserting a point, or removing a
                                            point.

Parameters:
graph (nx.Graph): Graph containing the nodes.
route (list): Current route.
nodes (list): List of nodes in the current route.

Returns:
list: Updated route and nodes list after modification.
"""

prob = random.uniform(0,1)
if prob < 0.4:
    route, nodes = __add_to_route(graph,route,nodes)
elif prob < 0.9:
    route = __change_edge_to_two(route)
else:
    if len(route) > 2:
        route = __change_edges_to_one(route)
return route, nodes
```

Funkcja f:

dane wejściowe: lista punktów (współrzednych), oryginalna macierz HI-C

dane wyjściowe: wartość korelacji Pearsona

Funkcja optymalizujaca f oblicza współczynnik korelacji Pearsona miedzy trasa (reprezentujaca trójwymiarowa strukture chromosomu) a oryginalna macierza Hi-C. Jest to kluczowy element naszej analizy, ponieważ ocenia, jak dobrze generowana trasa odzwierciedla rzeczywiste dane Hi-C. Wysoka korelacja oznacza, że nasz model skutecznie rekonstruuje przestrzenna strukture wybranego chromosomu. Dlatego dążymy do uzyskania jak najwyższego współczynnika korelacji.

```
def f_function(points: list, original_matrix: np.matrix) -> float:
"""
Computes the correlation between the Hi-C map of a given set of points and an
                                            original matrix.

Parameters:
points (list): List of coordinates.
original_matrix (np.matrix): Original Hi-C matrix.

Returns:
float: Pearson correlation coefficient between the Hi-C map of points and the
                                            original matrix.
"""
```

```
    matrix_of_distances = __get_hic_map(points)
    changed_dimension_matrix = __change_dimension(matrix_of_distances, original_matrix
                                                  .shape[0])
    corelation = __pearson_corelation(original_matrix, changed_dimension_matrix)
    return corelation
```

Wstepna analiza: Przeprowadzono 100 niezależnych inicjalizacji ścieżki dla grafu bazowego, a nastepnie wygenerowano trasy zgodnie z powyższym opisem funkcji g. Na wykresie poniżej przedstawiono wartość współczynnika korelacji (oś y) w zależności od i-tej iteracji.

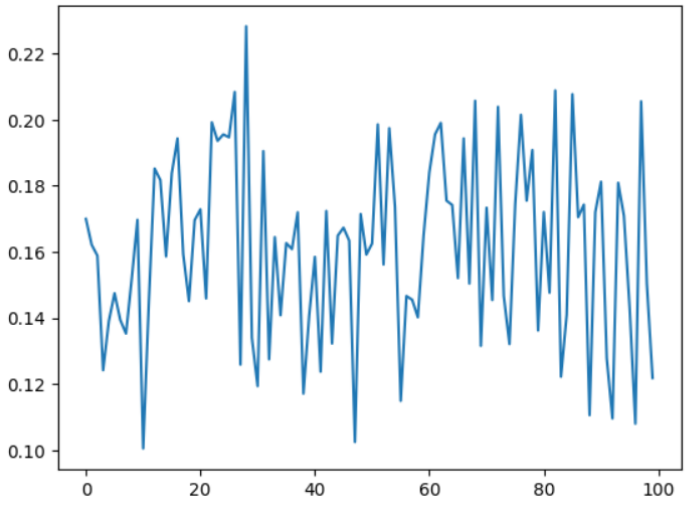

Figure 3: Korelacja Persona

Korelacja przyjmuje wartości z zakresu 0.12 - 0.20. Wskazuje to, że istnieje jedynie niewielka zgodność miedzy trójwymiarowa struktura generowana przez algorytm a rzeczywistymi danymi Hi-C.

5 Symulowane wyżarzanie

W celu optymalizacji i próby ulepszenia wyników, zastosowano algorytm symulowanego wyżarzania. Zadaniem jego było znalezienie trasy maksymalizujacej korelacje z oryginalna macierza Hi-C.

```
    def simmulated_annealing(graph, t_init, matrix, epochs=1000) -> list:
    """
    Performs simulated annealing to find a route that maximizes the correlation with
                                                  the given matrix.

    Parameters:
    graph (nx.Graph): Graph containing the nodes.
    t_init (float): Initial temperature.
    matrix (np.matrix): Original Hi-C matrix.
    epochs (int): Number of epochs for the annealing process (default is 1000).

    Returns:
    list: Final route, list of correlations over epochs, acceptance rates, and steps
                                                  taken.
    """

    route, nodes = initalize_route(graph)
    corr = f_function(route, matrix)
    accept = []
    steps = []
    correlations = [corr]
    for epoch in range(epochs):
        if epoch % 100 == 0:
            print('Epoch: ', epoch)
        prob = random.uniform(0,1)
```

```
        t = t_init * (1 - epoch / epochs)
        route_prop, nodes_prop = g_function(graph, route.copy(), nodes)
        corr_prop = f_function(route_prop, matrix)
        accept_rate = __accept_func(corr, corr_prop, t)
        accept.append(accept_rate)
        if  accept_rate > prob:
            if len(route) < len(route_prop):
                steps.append(1)
            elif len(route) == len(route_prop):
                steps.append(0)
            else:
                steps.append(-1)

            route = route_prop.copy()
            nodes = nodes_prop
            corr = corr_prop
        else:
            steps.append(0)
        correlations.append(corr)
    return route, correlations, accept, steps
```

6 Testy i wyniki

Na ostatnim etapie projektu przeprowadzono 50 niezależnych prób. Analiza miała na celu zidentyfikowanie symulacji, dla której współczynnik korelacji był najwyższy oraz tej, w której wzrost korelacji był najbardziej znaczacy (czyli takiej, w której różnica miedzy poczatkowa wartościa korelacji a najwieksza uzyskana była jak najwieksza). Podczas testów zauważono, że za każdym razem jest to ta sama korelacja.

Poniżej przedstawiamy wyresy analizujace najlepsza symulacje dla różnych wartości parametru sterujacego zwanego temperatura poczatkowa T. Dla wyższych wartości parametru spodzi

6.1 T = 1

Wykres liniowy przedstawia zmiany korelacji w zależności od liczby Epoch. Od około 200 Epochs korelacja zaczyna sie stabilizować na poziomie około 0.21.

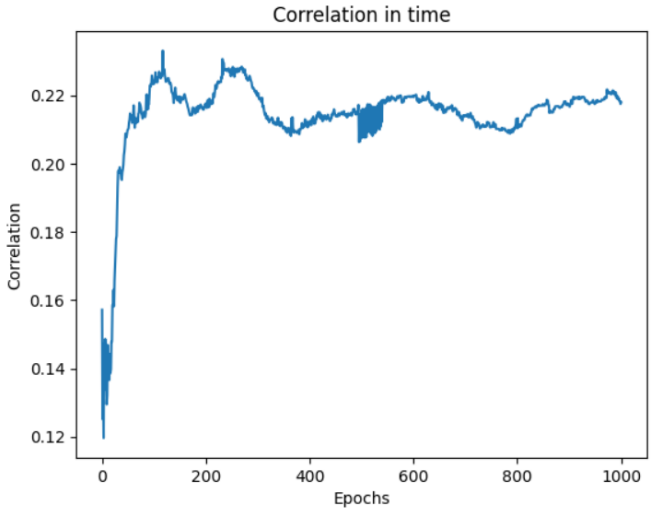

Figure 4: Correlation in time for T = 1

Poniższej wykres przedstawia zależność liczby epoch od rodzaju ruchów, które były najcześciej wykonywane na kolejnych etapach:

- move -1: krok w tył

- move 0: brak zmiany
- move +1: krok w przód

Model wykonywał najwiecej (około 80 % wszystkich ruchów) kroków w przód na każdym z etapów.

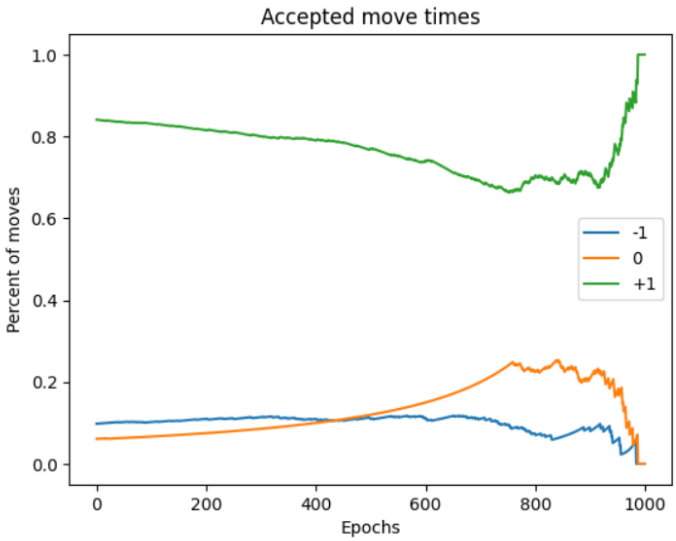

Figure 5: Accepted move times for T = 1

Jak widać na poniższym wykresie, prawdopodobieństwo wyboru nowego modelu, prawie zawsze było maksymalne. Dla takiej wartości temperatury poczatkowej nie zaobserwowano zbieżności modelu w trakcie tych iteracji.

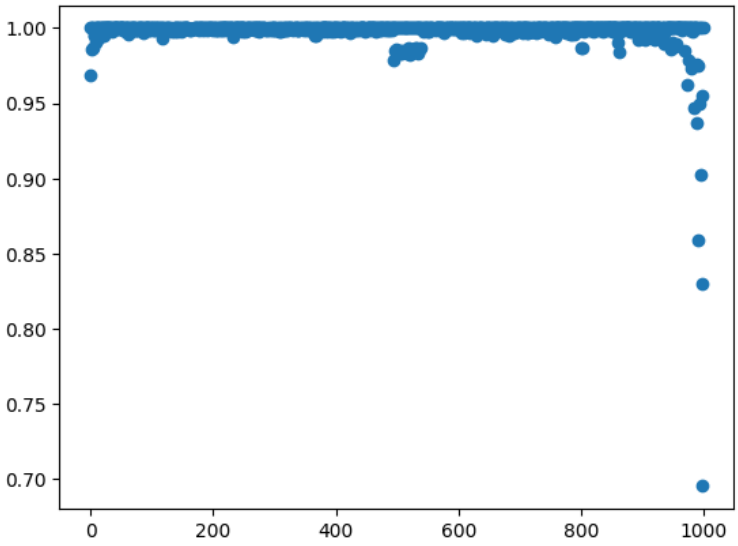

Figure 6:

6.2 T = 0.1

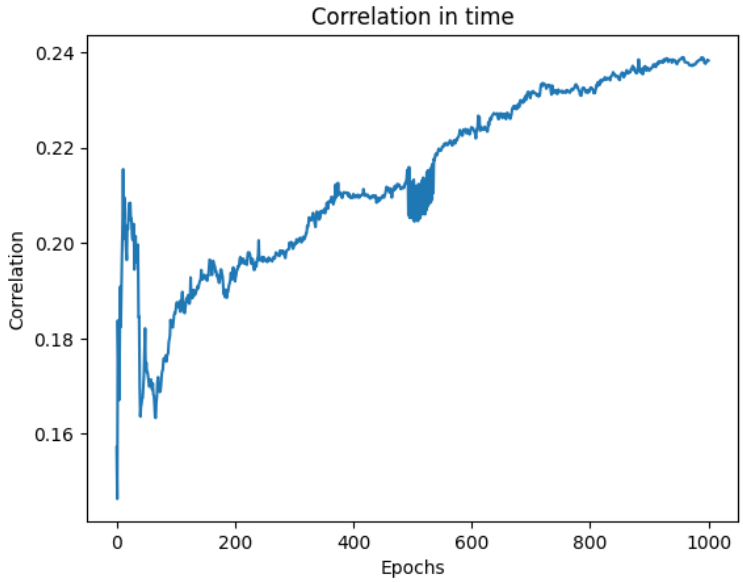

Figure 7: Correlation in time for T = 0.1

Korelacja wzrasta do wartości 0.24. Korelacja rośnie stopniowo, nie stabilizuje sie od pewnego momentu tak jak dla T = 1.

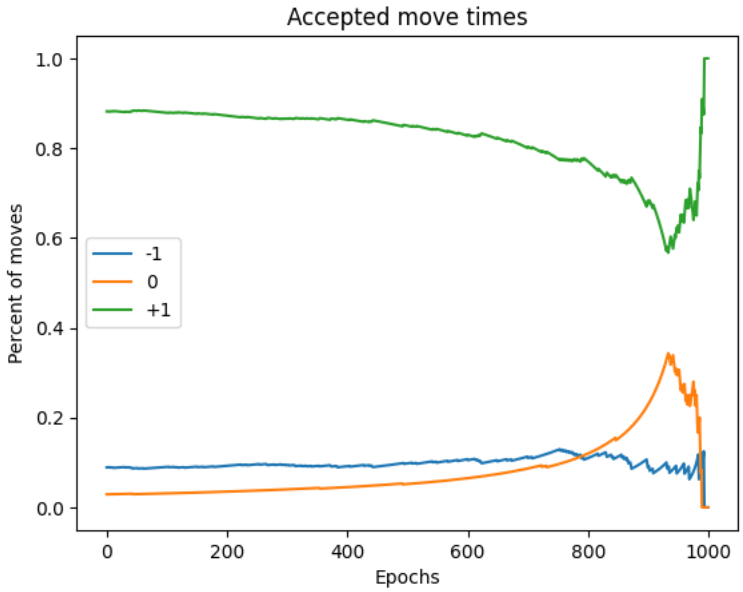

Figure 8: Accepted move time for T = 0.1

Tak jak poprzednio, najwiecej zaakceptowanych ruchów wystepuje dla wartości + 1 (krok w przód)

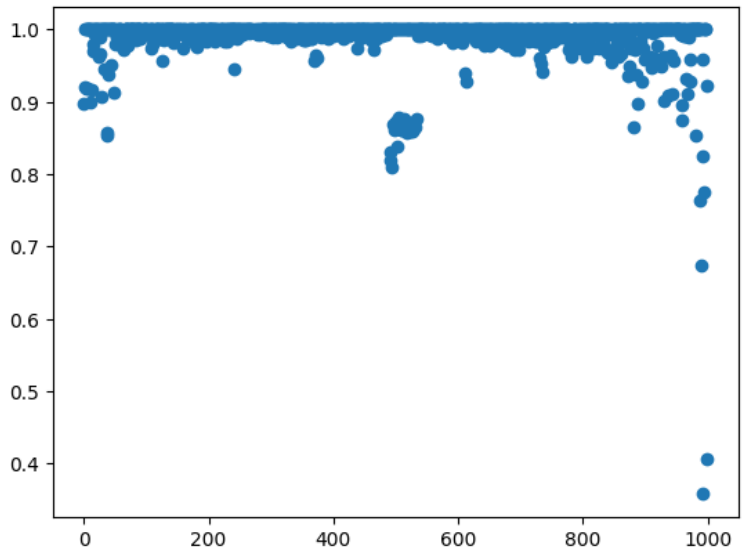

Figure 9:

Jak widać na powyższym wykresie, prawdopodobieństwo wyboru nowego modelu, zachowywało sie już troche lepiej niż w poprzednim przypadku, ale nadal nie zaobserwowano zbieżności.

6.3 T = 0.01

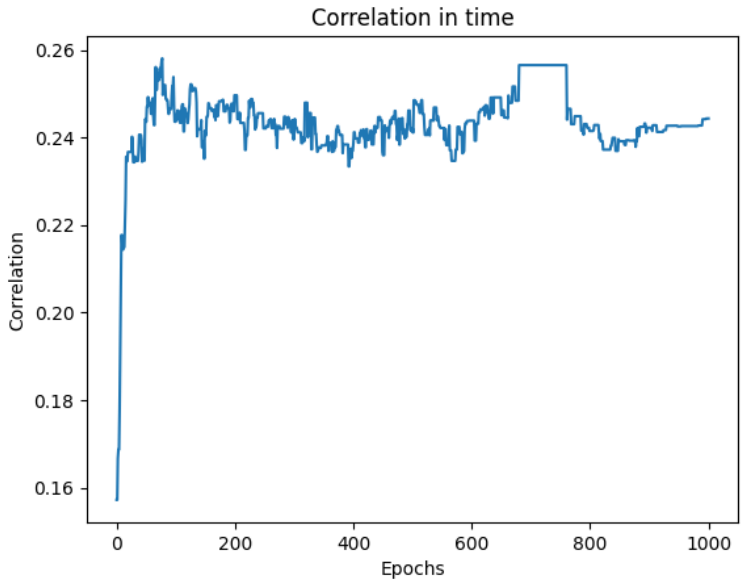

Figure 10: Correlation in time for T = 0.01

Współczynnik korelacji stabilizuje sie bardzo szybko na poziomie 0.25.

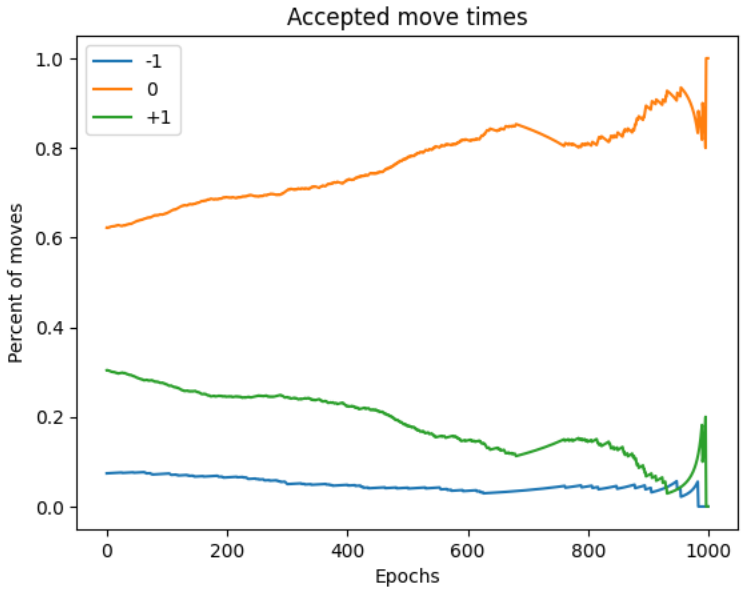

Figure 11: Accepted move times for T = 0.01

W trakcie symulacji najcześciej wystepował brak działania (brak ruch w przód lub w tył).

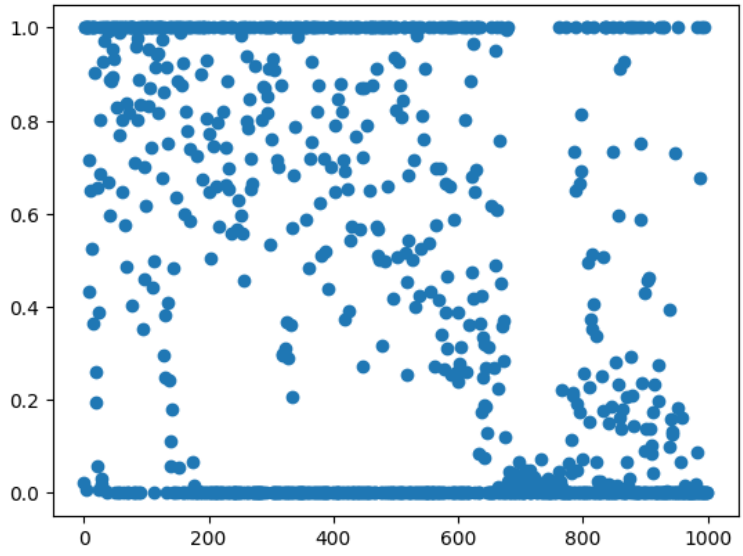

Figure 12:

W przypadku najniższej temperatury poczatkowej wraz z liczba epok, coraz rzadziej był wybierany nowy model co można zauważyć na wszystkich wykresach. Zapewniło to ustabilizowanie sie korelacji na pewnym poziomie oraz zbieżność modelu.

7 Podsumowanie

Zastosowanie algorytmu symulowanego wyżarzania sprawiło, że uzyskano stabilne, utrzymujace sie na jednym poziomie wyniki. Dodatkowo, za każdym okazywało sie, że dwa szukane modele sa tym samym modelem. Najlepsze wyniki uzyskał model w procesie symulowanego wyżarzania z paramtrem tempretatury równym 0.01. Korelacja wyniosła około 0.25, co dało najlepszy wynik spośród wszystkich

testowanych modeli. Na poniższym obrazie przedstawiamy końcowa wizualizacje struktury chromosomu wygenerowanego przez nasz model.

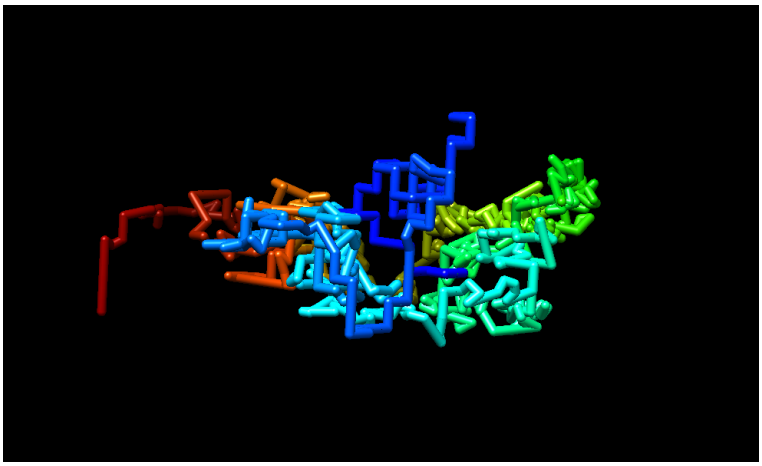

Figure 13: Final Model