Warsztaty Badawcze 2024, Projekt 2
Stworzenie modelu na bazie RPR

Maciej Borkowski, Krzysztof Kosz, Zuzanna Piróg, Jakub Rymarski, Maciej Szpetmański

Zadanie projektowe polegało na stworzeniu trójwymiarowego modelu chromatyny na podstawie danych Hi-C. Należało w tym celu zaimplementować i wykorzystać metodę RPR (Reccurence plot-based reconstruction). Metoda RPR została zaimplementowana na podstawie jej opisu w artykule nr 6 z materiałów JC: *Three-dimensional reconstruction of single-cell chromosome structure using recurrence plots* autorstwa Yoshito Hirata, Arisa Oda, Kunihiro Ohta, Kazuyuki Aihara.

Do projektu wykorzystaliśmy dane z komórki myszy, która ma 19 par chromosomów autosomalnych i jedną parę chromosomów płciowych. Dane zostały pobrane ze strony: https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE48262

1. Opis metody RPR

Hi-C (High-throughput Chromosome Conformation Capture) to technika, która umożliwia analizę przestrzennych interakcji genomu na dużą skalę. Pozwala na mapowanie kontaktów między różnymi regionami chromatyny, co jest kluczowe dla zrozumienia organizacji genomu w przestrzeni trójwymiarowej. Informacje te są istotne dla wielu aspektów biologii komórkowej, w tym regulacji genów, replikacji DNA oraz architektury jądra komórkowego.

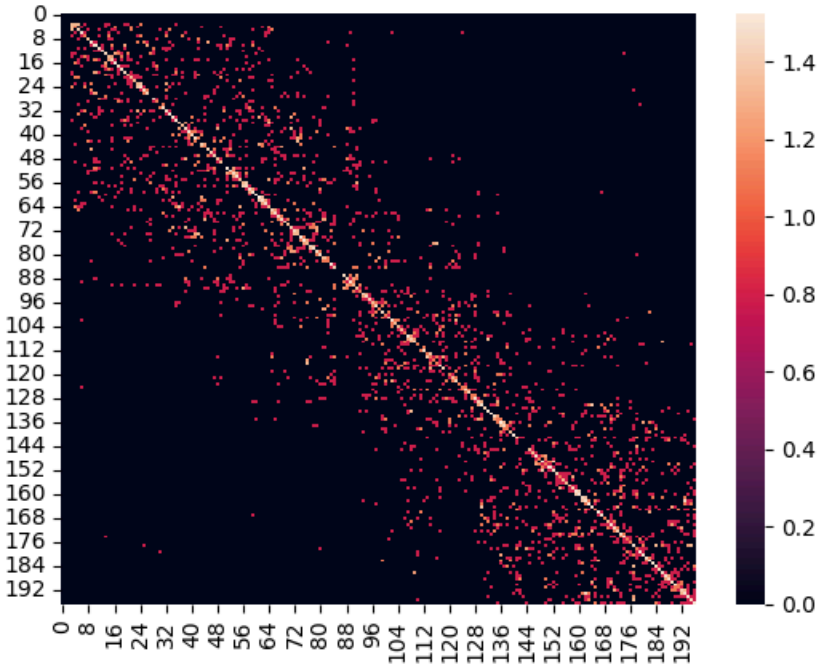
Dzięki Hi-C możemy uzyskać dane dotyczące interakcji chromosomowych w całym genomie, co jest nieocenione w badaniach nad funkcjonowaniem genomu oraz w identyfikacji regionów regulacyjnych, które mogą mieć znaczenie w kontekście chorób genetycznych i nowotworów.

RPR (Reccurence Plot-based Reconstruction) to metoda umożliwiająca trójwymiarową rekonstrukcję struktury chromosomów na podstawie danych Hi-C. Kluczowym elementem tej metody jest wykorzystanie tzw. wykresów powtarzalności (recurrence plots), które pozwalają na wizualizację i analizę dynamiki układów nieliniowych. Daną pozycję w chromatynie możemy interpretować jako pewną chwilę czasową, a kolejne fragmenty chromatyny możemy utożsamić ze zmieniającym się ciągiem zdarzeń.

Zaletami RPR są możliwość dokładnego odwzorowania struktury i efektywność obliczeniowa, co pozwala na stosowanie dużych zestawów danych. Jest także skuteczna przy niskiej rozdzielczości danych single-cell Hi-C. Metoda RPR zapewnia unikalną rekonstrukcję trójwymiarowych struktur chromosomów, co różni ją od metod stochastycznych.

2. Implementacja, wyniki

Pierwszym krokiem było przekształcenie danych wejściowych. Plik csv zawierał informacje o kontaktach między poszczególnymi częściami chromatyny. Na ich podstawie stworzyliśmy biny, w których policzyliśmy kontakty między odpowiednimi fragmentami chromatyny. Następnie stworzyliśmy mapę Hi-C.



Mapa Hi-C dla pierwszego chromosomu

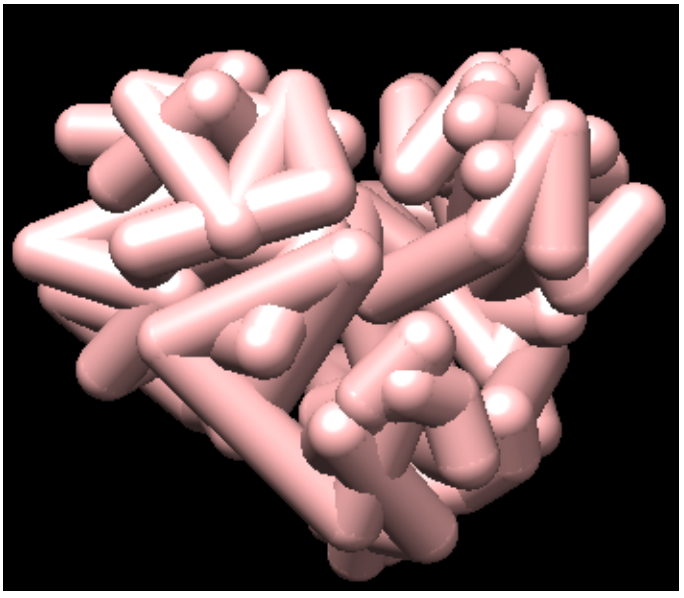
Kolejnym krokiem była już implementacja metody RPR. Na początku stworzyliśmy binarną macierz kontaktów, w której łączyliśmy biny leżące blisko siebie. Następnie obliczyliśmy wagi według wzoru z artykułu nr 6, co pozwoliło na odpowiednie uwzględnienie siły oddziaływania między leżącymi blisko siebie danymi fragmentami.

```python
def calculate_weight_matrix(contact_matrix):
    N = contact_matrix.shape[0]
    weight_matrix = np.zeros((N, N))

    for i in range(N):
        for j in range(N):
            if contact_matrix[i, j]==1:
                intersection = np.sum((contact_matrix[i]==contact_matrix[j]) & (contact_matrix[i]==1))
                union=np.sum(contact_matrix[i])+np.sum(contact_matrix[j])-intersection
                weight_matrix[i, j] = 1 - (intersection / union)
            else:
                weight_matrix[i, j] = 1

    return weight_matrix
```

Następnym krokiem było stworzenie macierzy i grafu odległości. Odległości między każdymi dwoma wierzchołkami liczyliśmy na podstawie algorytmu Dijkstry do znajdowania najkrótszych ścieżek. Ostatnim krokiem była konstrukcja trójwymiarowego modelu chromatyny za pomocą metody multidimensional scaling.

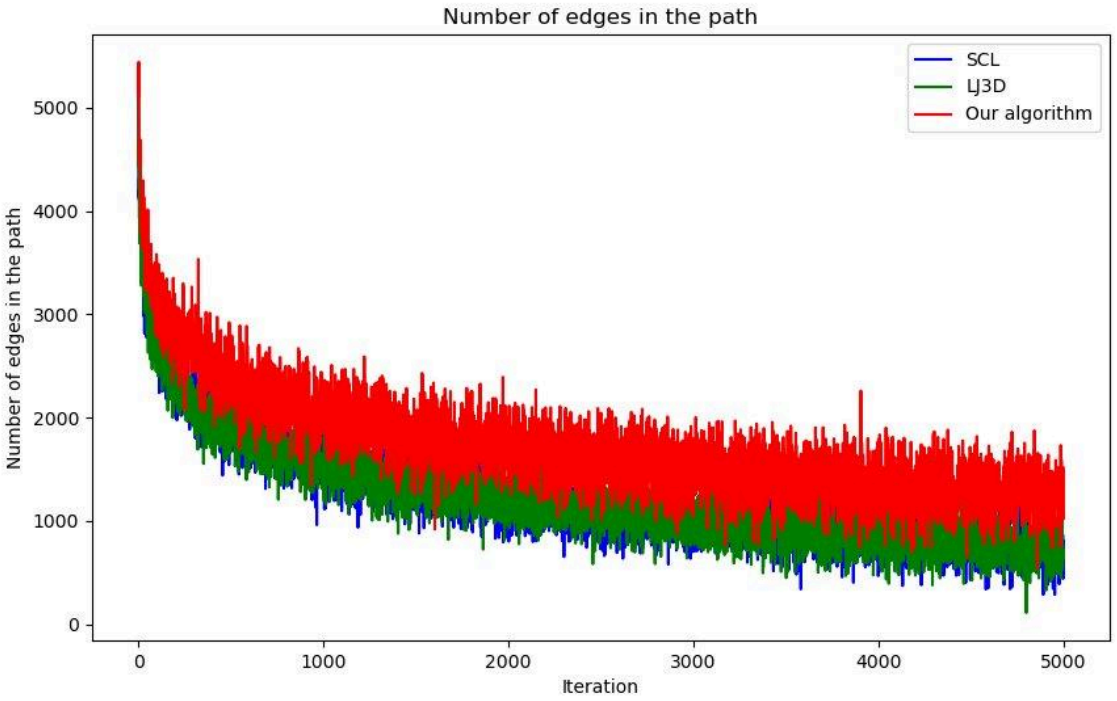
```python
def rpr_visualisation(data_hic, num_of_chromosomes, bin_size):
    cm=calc_contact_matrix(data_hic, num_of_chromosomes, bin_size)
    wr=recurrence_plot_matrix(cm, 0)
    wm=calculate_weight_matrix(wr)
    wg=create_weighted_graph(wm)
    distances_matrix=calculate_distances(wg)
    mds = MDS(n_components=3, dissimilarity='euclidean')
    embedding = mds.fit_transform(distances_matrix)
    return embedding
```

Ostateczna funkcja zawierająca kolejne kroki algorytmu



Wizualizacja dla pierwszego chromosomu

3. Porównanie z innymi metodami



Dokonaliśmy również porównania naszego algorytmu z innymi metodami, takimi jak SCL i LJ3D. Jak widać na powyższym wykresie, algorytmy SCL i LJ3D poradziły sobie lepiej w przypadku tworzenia grafu ścieżki, jednak nasze rozwiązanie nie było dużo gorsze. Aby dokonać tego porównania dla każdej z metod stworzyliśmy 20 modeli budujących graf. Każdy ze słupków na wykresie jest tworzony w następujący sposób: górna krawędź słupka jest najwyższym wynikiem spośród naszych modeli, natomiast dolna krawędź jest wynikiem najniższym w danej iteracji.