

Warszawa, 9 grudnia 2009r.

Ołtarzewski Przemysław
Szczepański Paweł
Wieleba Piotr

Dokumentacja wstępna projektu z przedmiotu Metody Bioinformatyki



Implementacja algorytmu Needlemana-Wunscha,
badającego podobieństwo dwu sekwencji

Przedstawienie algorytmu

Algorytm Needlemana-Wunsha jest jedną z pierwszych prób zaprzęgnięcia programowania dynamicznego w celu wsparcia inżynierii biomedycznej w porównywaniu dwóch sekwencji. Opiera się on na zastosowaniu tzw. macierzy podobieństwa (ang. similarity matrix), która opisuje „nagrody” za znalezienie części wspólnej dwóch sekwencji oraz „kary” naliczane w wypadku wystąpienia niezgodności (różniącego się symbolu).

W macierzy podobieństwa należy umieścić po jednym wierszu dla każdego unikalnego symbolu znajdującego się w pierwszej sekwencji oraz po jednej kolumnie dla każdego unikalnego symbolu występującego w sekwencji drugiej. Przykładowa macierz podobieństwa stosowana przy porównywaniu łańcuchów DNA mogłaby wyglądać następująco:

-	A	G	C	T
A	10	-1	-3	-4
G	-1	7	-5	-3
C	-3	-5	9	0
T	-4	-3	0	8

Na przecięciu się każdego wiersza i kolumny znajduje się liczba, która określa nagrodę lub karę, w zależności od tego, czy znajdujące się w wierszu i kolumnie symbole są jednakowe, czy też różnią się między sobą.

Dodatkowo, w celu umożliwienia porównania sekwencji o różnych długościach, należy uwzględnić tzw. karę za przerwę (ang. gap penalty). Kara za przerwę naliczana jest w momencie, gdy uznajemy, że w jednej z sekwencji znajduje się symbol nie posiadający odpowiednika w drugiej sekwencji.

W celu znalezienia dopasowania dwóch sekwencji (nazwijmy je A i B) cechującego się największym stopniem zgodności tworzymy macierz F zawierającą po jednej kolumnie dla każdego kolejnego symbolu znajdującego się w sekwencji A (plus jedna kolumna przerw) oraz po jednym wierszu dla każdego symbolu w sekwencji B (plus jeden wiersz przerw). Gdy sekwencja A zawiera n symboli, zaś sekwencja B – m symboli powstała macierz cechuje się rozmiarem $(n+1)*(m+1)$. Algorytm Needlemana-Wunsha służy do wypełnienia tej macierzy.

W lewym górnym rogu macierzy F wpisujemy 0. Kolejne komórki macierzy wyznaczamy jako wartość maksymalną z trzech wartości:

1. $F(i-1, j-1) + e(b_i, a_j)$ - w przypadku połączenia
2. $F(i, j-1) + d$ - w przypadku przerwy na sekwencji A
3. $F(i-1, j) + d$ - w przypadku przerwy na sekwencji B

Gdzie:

a_i – jest to i-ty symbol sekwencji A

b_j – jest to j-ty symbol sekwencji B

$e(b_i, a_j)$ – jest wartością wyznaczoną przez macierz podobieństwa

d – kara za przerwę

W powyższy sposób wypełniamy całą macierz F. Warto zauważyć, iż w prawym dolnym rogu otrzymamy komórkę z maksymalną wartością ze wszystkich wyborów. Kolejnym krokiem jest wyznaczenie optymalnych ścieżek, które doprowadziły do jej powstania. Wykonuje się to poprzez analizę tego, z jakiego wyboru (1,2,3) powstała każda komórka, co odpowiednio wskaże nam kierunek ścieżki. Może się tak zdarzyć, iż niektóre wybory w ramach jednej komórki będą dawały ten sam rezultat, co będzie skutkowało więcej niż jedną optymalną ścieżką.

Przykład działania algorytmu

Mając dane:

2 sekwencje $S_1 = \text{ATGC}$ i $S_2 = \text{ATTGC}$,


karę za przerwę = -2,

oraz funkcję podobieństwa opisaną wzorem:

$$\text{sim}(a,b) = \begin{cases} -2, & \text{gdy } a = '-' \text{ lub } b = '-' \\ -1, & \text{gdy } a \neq b \\ 1, & \text{gdy } a = b \end{cases}$$

Macierz F po wypełnieniu będzie wyglądała następująco:

		A	T	G	C
	0	-2	-4	-6	-8
A	-2	1	-1	-3	-5
T	-4	-1	2	0	-2
T	-6	-3	0	1	-1
G	-8	-5	-2	1	0
C	-10	-7	-4	-1	2



		A	T	G	C
	0	-2 ←	-4 ←	-6 ←	-8 ←
A	-2 ↑	1 ↘	-1 ←	-3 ←	-5 ←
T	-4 ↑	-1 ↑	2 ↘	0 ←	-2 ←
T	-6 ↑	-3 ↑	0 ↘	1 ↘	-1 ↘
G	-8 ↑	-5 ↑	-2 ↑	1 ↘	0 ↘
C	-10 ↑	-7 ↑	-4 ↑	-1 ↑	2 ↘

Jak widać na powyższym rysunku, kolorem zielonym zostały wyznaczone dwie optymalne ścieżki, co odzwierciedla się w dwóch optymalnych dopasowaniach:

$S_1 = \text{AT- GC}$

$S_2 = \text{ATTGC}$

$S_1 = \text{A- TGC}$

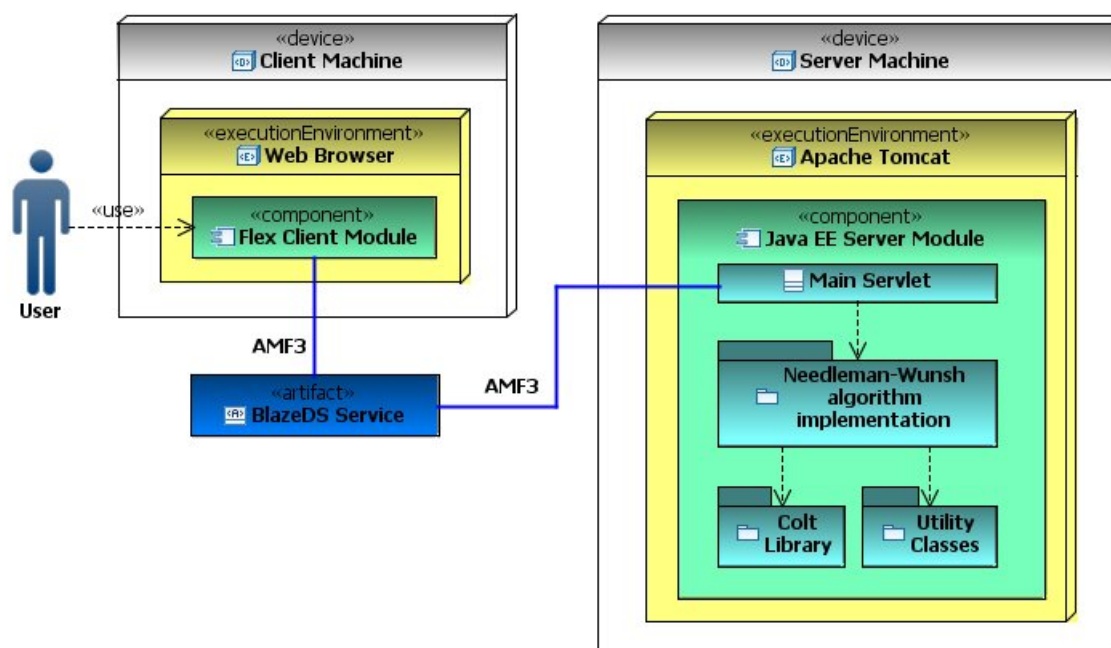
$S_2 = \text{ATTGC}$

Projekt rozwiązania

Projekt architektury rozwiązania przedstawiono na diagramie 1. Użytkownik ma do dyspozycji moduł klienta zrealizowany w technologii Adobe Flex. Korzysta z niego za pośrednictwem przeglądarki sieciowej, która pobiera z serwera plik SWF i uruchamia go lokalnie na komputerze użytkownika.

Moduł klienta komunikuje się z serwerem za pośrednictwem protokołu AMF3. Za serializację danych przekazywanych pomiędzy front-endem i back-endem odpowiada zestaw usług Adobe BlazeDS.

Moduł serwera zostanie zrealizowany w technologii Java EE i będzie funkcjonował w kontenerze aplikacji Apache Tomcat. W celu optymalizacji wydajności, do implementacji algorytmu zostanie wykorzystana biblioteka Colt, oferująca zestaw struktur danych



umożliwiających szybkie operacje między innymi na macierzach.

Diagram 1 – Projekt architektury

Przewidywaną interakcję użytkownika z systemem można opisać według następującego scenariusza:

1. Użytkownik wpisuje w przeglądarce URL, pod którym została osadzona aplikacja
2. Przeglądarka pobiera i uruchamia komponent klienta w postaci pliku SWF
3. Użytkownik korzystając z interfejsu wprowadza porównywane sekwencje DNA, macierz podobieństwa oraz karę za przerwę. Opcjonalnie, wczytuje dane z pliku.

4. Użytkownik zatwierdza dane, inicjując proces obliczeń. Dane wysyłane są przez kanał AMF do modułu serwera.
5. Moduł serwera otrzymuje żądanie użytkownika, przetwarza dane z wykorzystaniem implementacji algorytmu Needlemana-Wunsha. Opcjonalnie, częściowe wyniki lub aktualny stan obliczeń (np. w procentach wykonania), może być przekazywany do klienta i prezentowany użytkownikowi.
6. Moduł serwera po przetworzeniu danych odsyła wyniki do klienta: macierz kar dla ścieżek częściowych oraz optymalne dopasowania sekwencji.
7. Moduł klienta wyświetla wyniki. Opcjonalnie pozwala na zapisanie rezultatu obliczeń do pliku.