Warszawa, 17.04.2014

Marcin Kubik

Krzysztof Opasiak

Jacek Sosnowski

**METODY BIOINFORMATYKI**

**Implementacja algorytmu BLOSUM do obliczania macierzy substytucji**

Prowadzący: Marcin Lewandowski

1. **Treść projektu**

Tematem projektu jest implementacja algorytmu BLOSUM do obliczania macierzy substytucji. W ramach implementacji dostarczone będą macierze prezentujące kolejne etapy działania algorytmu oraz końcową macierz BLOSUM.

1. **Opis implementacji**

Implementacja algorytmu wyznaczania macierzy BLOSUM została wykonana w języku C++ z wykorzystaniem elementów biblioteki Qt. Aby zapewnić uniwersalność przygotowanego rozwiązania zostało ono wykonane z wykorzystaniem mechanizmu szablonów. Szablon algorytmu jest parametryzowany:

1. Typem dla pojedynczego symbolu (np. char lub wchar)
2. Typem kontenera w którym znajduje się sekwencja
3. Typem stałopozycyjnym używanym do obliczeń (np. int lub long)
4. Typem zmiennopozycyjnym używanym do obliczeń (np. float lub double)

Podczas implementacji wykorzystano dość licznie klasy kontenerowe oparte o funkcje haszujące, co pozwoliło ograniczyć złożoność poszczególnych etapów do następujących wartości:

N – liczba sekwencji

L – długość sekwencji

M – liczba unikalnych symboli

Zliczanie par:

L\*(N + M2)

Obliczenie liczby wszystkich par:

1

Normalizacja liczby par:

M2

Obliczanie prawdopodobieństw:

M2

Estymacja prawdopodobieństw:

M2

Obliczenie logarytmów:

M2

Obliczenie macierzy BLOSUM:

M2

W przypadku białek, M jest ograniczone i bardzo małe, przez co największy wkład w złożoność algorytmu ma krok pierwszy. Ogólna zatem złożoność wykonanej implementacji to w tym przypadku:

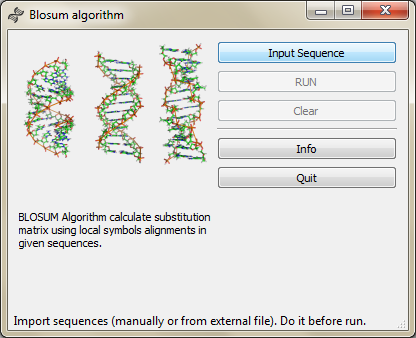
O(N\*L)

1. **Opis interfejsu graficznego**

Aplikacja wyposażona jest w intuicyjny interfejs graficzny na który składa się zestaw okien. W momencie uruchomienia, prezentowany jest widok panelu sterującego, który pozwala uruchomić algorytm. W ogólności dozwolony jest następujący przebieg zdarzeń:

1. Uruchomienie aplikacji.
2. Wciśnięcie przycisku „*Input Sequence*”, co spowoduje wyświetlenie okna o takim samym tytule (*rysunek 2*).
3. Za pomocą nowego okna należy wprowadzić poprawną sekwencję symboli i decyzję zatwierdzić przyciskiem *DONE*. (Szczegółowy opis tych czynności znajduje się niżej).
4. Uaktywnia się przycisk *RUN* z głównego panelu (*rysunek 1*). Naciśnięcie go spowoduje uruchomienie algorytmu i prezentację wyników za pomocą nowego okna „*Results*” (*rysunek 3*).
5. Wyniki obliczeń można usunąć wybierając przycisk *Clear* z panelu głównego aplikacji. Ale okno „*Results*” pozostanie otwarte (przydatne na przykład do późniejszego porównywania wyników). W tym momencie można wczytać nowe sekwencje i zacząć jeszcze raz od kroku 1.

Panel sterowania zawiera również przycisk pozwalający na zamknięcie okna oraz przycisk „*Info*” wyświetlający informacje na temat aplikacji oraz jej autorów.

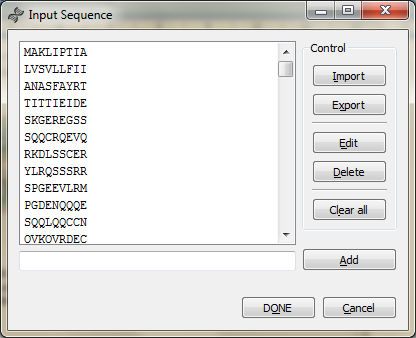


Rysunek

Okno *Input Sequence* pozwala na wprowadzenie sekwencji znaków, które zostaną podane na wejście algorytmu Blosum. Mogą być wpisane ręcznie za pomocą linii wejściowej znajdującej się obok przycisku *Add*. Innym sposobem jest wczytanie sekwencji z pliku za pomocą formularza wyświetlanego po wyborze przycisku *Import*. Jako format zostało przyjęte rozszerzenie „.blo” ponieważ aplikacja zakłada, że każda sekwencja będzie tworzyć osobną linię w danym pliku bez symboli rozdzielających ani końcowych. Dodatkowym założeniem sprawdzanym dopiero po wciśnięciu przycisku *DONE* jest fakt iż wszystkie sekwencje muszą mieć tą samą długość. Funkcja Importu umożliwia również wczytanie plików z rozszerzeniem „.mbi”, których podstawowym zastosowaniem było jednak testowanie działania algorytmu, w module „*tests*”.

Każdą z sekwencji można edytować – przycisk *Edit*, oraz usunąć – przycisk *Delete*. Sekwencje można zaznaczać w sposób dowolny i usuwać od razu całą grupę.

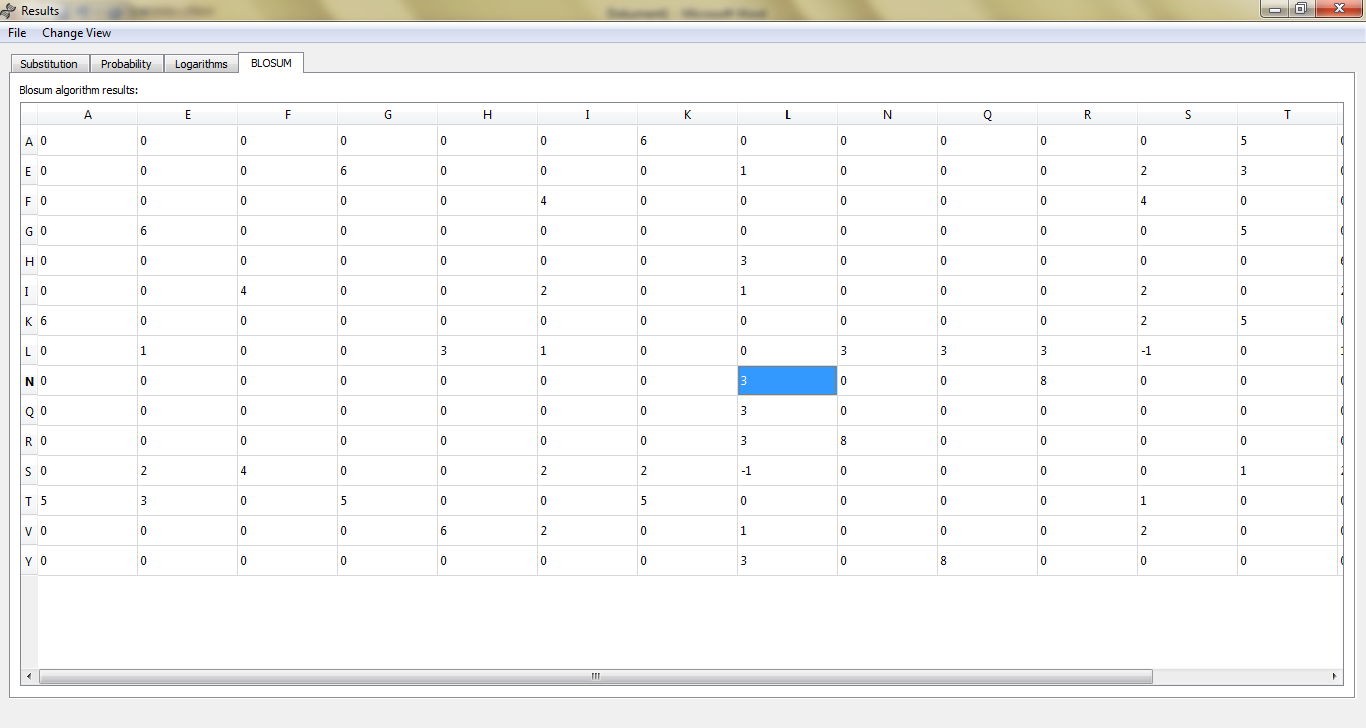
Przygotowany zbiór sekwencji można zapisać do nowego pliku za pomocą opcji *Export*. Wynik tej operacji jest zgodny z formatem pliku „.blo”.



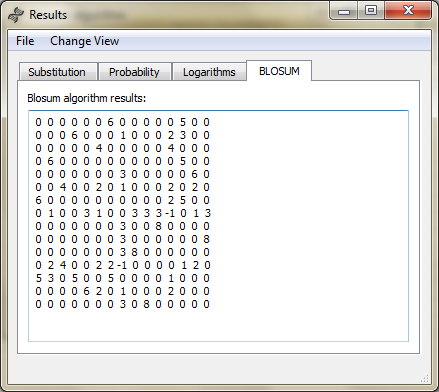
Rysunek

Okno wyników zawiera macierze z poszczególnych kroków algorytmu: macierz podstawień, prawdopodobieństw, logarytmów oraz ostateczną macierz Blosum. Menu „*Change views*” umożliwia zmianę sposobu prezentacji wszystkich macierzy. Dostępne są dwa widoki: w tabeli (*rysunek 3*) oraz za pomocą zwykłego tekstu (*rysunek 4*).

Każdą z macierzy można zapisać do pliku za pomocą menu *File*. Są tam zarówno możliwości zapisu wyników do plików CSV jak również jako zwykły tekst.



Rysunek



Rysunek

1. **Wyniki przeprowadzonych testów**

Głównym parametrem, jaki można wykorzystywać przy pomiarach i analizie algorytmu prowadzącego do uzyskania macierzy BLOSUM jest długość sekwencji. Do przeprowadzenia testów wykorzystano zatem manipulację długością analizowanych sekwencji jak i ich ilością. Dane pobierane były z bazy danych <http://www.uniprot.org/uniprot/> - analizowane białka to:

* Albuminy
* Lizyny
* Keratyny
* Protaminy

Z każdej z nich do testów wybranych zostało około kilkudziesięciu białek.

Wszystkie pomiary (dla każdej długości sekwencji czy dla każdej ilości sekwencji) były wykonywane 10-krotnie a ich wyniki były uśredniane w celu uniknięcia nieścisłości wynikających z wahań czasów wykonania.

Sprzęt na jakim wykonywano testy:

Intel Core i7, Radeon HD 5800, 4 GB RAM, System – Linux Ubuntu 12.10

1. Wpływ długości sekwencji na czas obliczeń

Pierwszym z modyfikowanych parametrów jest długość sekwencji. Do celów testowych wykorzystano dane o łącznej liczbie 1400 sekwencji (im większa długość sekwencji tym danych było odpowiednio więcej). Wyniki w formie wykresu przedstawiono poniżej. Świadczą one o wyraźnym wpływie długości sekwencji na czas pomiarów. Największa różnica w wydajności ma swoje źródło w pierwszym kroku algorytmu, w którym następuje zliczenie ilości par aminokwasów. Ponieważ zliczanie to odbywa się kolumnami, im większa liczba kolumn (czyli większa liczba sekwencji) tym dłużej trwają obliczenia. Jest to krok algorytmu, który najlepiej nadaje się do możliwości zrównoleglenia obliczeń (na przykład z podziałem na procesory).

1. Wpływ ilości sekwencji na czas działania algorytmu

Drugim parametrem mającym wpływ na czas działania algorytmu jest ilość sekwencji. Podobnie jak w poprzednim przypadku jest to związane przede wszystkim z pierwszym krokiem algorytmu, w którym następuje zebranie informacji o częstości występowania par takich samych aminokwasów w poszczególnych kolumnach. Im dłuższa jest kolumna, w której trzeba dokonać obliczeń tym dłuższy jest czas działania całego algorytmu. Na poniższych diagramach przedstawiono wpływ ilości sekwencji na czas działania algorytmu. Analogicznie jak poprzednio wszystkie testy przeprowadzono 10-krotnie – ich wyniki zostały uśrednione.

1. Wpływ rodzaju podziału danych na czas działania algorytmu

Algorytm do tworzenia macierzy BLOSUM operuje przede wszystkim na ciągach aminokwasów. Można zatem ciąg wielu sekwencji traktować jako jedną długą sekwencję, którą można w dowolny sposób dzielić. Dzięki temu można rozważać, w jaki sposób podzielić ciąg aminokwasów na sekwencje aby uzyskać jak najbardziej optymalne wyniki (pod względem wydajności). Oczywiście z biologicznego punktu widzenia badanie takie nie ma znaczenia (pozycja aminokwasu ma znaczenie w kontekście rozważania sekwencji białkowych), z punktu widzenia informatycznego testy takie mogą być przydatne dla osób, które rozważają wydajność algorytmu, czy poszukują dróg jego optymalizacji. Decyzja, czy wprowadzać sekwencje dłuższe, ale mniej czy dużą liczbę krótkich sekwencji może mieć także znaczenie przy decydowaniu, jakie dane posłużą za dane testowe do tworzenia macierzy BLOSUM. Na poniższym diagramie przedstawiono wyniki działania dla sekwencji o łącznej liczbie około 40 tys. aminokwasów. Dzielone były one na coraz dłuższe sekwencje (co za tym idzie było ich coraz mniej).

Na przedstawionym diagramie nie można zauważyć żadnej korelacji między podziałem na sekwencje określonej długości a czasem wykonania algorytmu. Wszystkie wartości mieszczą się w zakresie 15-20 ms (za wyjątkiem pierwszej, granicznej wartości nieznacznie wykraczającej ponad 20 ms). Świadczy to o małym znaczeniu podziału dostarczonych danych wejściowych. Dzieje się tak, ponieważ pierwszy krok algorytmu wymaga iterowania zarówno po wszystkich elementach danej kolumny jak i po wszystkich kolumnach przynajmniej raz (w zaimplementowanej na potrzeby niniejszego projektu wersji). Oznaczając *L*  jako długość sekwencji i N jako liczbę tych sekwencji:

L × N = const

Dalsze kroki algorytmu nie biorą już pod uwagę samych sekwencji a symbole w nich występujące. Powoduje to, że nie są one czułe na zmiany długości sekwencji czy zmianę ich ilości.

1. Wnioski

Algorytm do tworzenia macierzy BLOSUM wykorzystywany jest zazwyczaj jednokrotnie dla danego zestawu danych. Dane te są następnie wykorzystywane dla dalszych badań. Złożoność algorytmu jest liniowo zależna od liczby symboli zawartych we wszystkich sekwencjach. Oznacza to, że nie ma znaczenia czy przetwarzamy dużo krótkich sekwencji czy też niewiele ale długich sekwencji. Krokiem najbardziej wpływającym na czas działania algorytmu jest krok pierwszy (zebranie informacji o częstości występowania poszczególnych par aminokwasów), wymaga on bowiem przeglądania wszystkich dostarczonych sekwencji. Dalsze kroki algorytmu w praktyce w ogóle nie zależą od danych wejściowych, są zatem dużo szybsze. Czas działania algorytmu jest zależny również od liczby unikalnych symboli w sekwencjach. Im mniej różnych symboli obecnych jest w danych wejściowych, tym mniej jest par aminokwasów, jakie należy rozważać a co za tym idzie zmniejsza się także macierz, na której algorytm operuje.

Algorytm BLOSUM jest bardzo przydatnym narzędziem do badania podobieństw białek. Blokowość operacji może powodować konieczność wstępnej obróbki dostarczanych danych, z drugiej jednak strony pozwala na stworzenie łatwych, powtarzalnych kroków pozwalających na otrzymanie oczekiwanych rezultatów. Zalety macierzy BLOSUM powodują, że są one szeroko wykorzystywane w różnego rodzaju badaniach białek.