MBI dokumentacja końcowa

Adam Niziński Michał Przyłuski

25 stycznia 2010

Projekt miał na celu implementację algorytmu badającego podobieństwo dwu sekwencji o liniowej złożoności pamięciowej.

1 Algorytm

Badanie podobieństwa (globalnego) dwu sekwencji jest częstym i ważnym zadaniem, przed którym stoi biologia obliczeniowa, zwana także, bioinformatyką. Niestety, klasyczny algorytm Needlemana-Wunscha, oparty o ideę programowania dynamicznego, charakteryzuje się złożonością pamięciową rzędu $O(m \star n)$, gdzie m i n są odpowiednio długościami rozpatrywanych sekwencji. Algorytm ten został szczegółowo opisany w [1].

Warto jednak zauważyć, że w danym przebiegu pętli algorytmu Needlemana-Wunscha wykorzystywane są dane tylko z bezpośrednio poprzedzającej kolumny. Wynika to z faktu, iż ona właśnie zawiera najlepsze dopasowanie, które udało się znaleźć do bieżącej pozycji. Co więcej, algorytm ten nie wymaga „patrzenia w przyszłość", gdyż do wyznaczenia wartości dopasowania na danej pozycji, potrzebne są tylko informacje z jej najbliższego otoczenia. Takie spostrzeżenia poczynili autorzy pierwszej wzmianki o tym algorytmie [2] w 1988 roku. Ta prosta obserwacja, pozwoliła stworzyć algorytm badający podobieństwo dwu sekwencji o liniowej złożoności pamięciowej, który wstępnie omówiono w [3]. Warto dodać, że opis zawarty w [4, r. 12.1] jest pełny i obszerny, a także na niego powołują się w [5] autorzy biblioteki ncbi-lib, która jest referencyjną implementacją bardzo wielu algorytmów bioinformatyki. Nas najbardziej interesuje klasa **CMMAligner** omówiona w [6].

Sam algorytm wykorzystuje fakt, iż kolumna, która już nie będzie wykorzystana zostaje zwolniona, a kolumny, które nie są jeszcze potrzebne, nie zostały jeszcze zaalokowane w pamięci. Takie podejście pozwoliło znacznie zmniejszyć wymagania pamięciowe algorytmu.

Z drugiej strony, brak pełnej tablicy podobieństw znacznie utrudnia odtworzenie najlepszego dopasowania. Z tego powodu druga faza algorytmu — powrotu — jest rekurencyjna. Polega ona na rekurencyjnym podziale tablicy podobieństw na bloki realizujące „górną" i „dolną" połowę ścieżki odpowiadającej za optymalne dopasowanie. Możemy tak zrobić w myśl zasady optymalności Bellmana, która w naszym przypadku oznacza, że wyznaczone optymalne „podścieżki" razem tworzą optymalną ścieżkę.

Podsumowując, można powiedzieć, że zmniejszona złożoność pamięciowa została uzyskana kosztem znacznej komplikacji algorytmu.

2 Funkcjonalność

Program wczytuje dane wejściowe na dwa sposoby:

- z pliku zawierającego sekwencję w formacie plain,
- z pola tekstowego.

Możliwość wczytywania z pola tekstowego ma kluczowe znaczenie w fazie uruchamiania i testowania projektu, gdyż umożliwia łatwe testowanie poprawności działania na krótkich danych. Możliwość wczytywania pliku jest istotna dla właściwego użycia programu, gdyż prawdziwe sekwencje (zazwyczaj) są znacznej długości.

Użytkownik ma możliwość podania dowolnej macierzy substytucji — odpowiadającej zapotrzebowaniom użytkownika, oraz swobodnego wyboru kary za przerwę. Ponadto, dla celów porównawczych (i potwierdzenia poprawności działania) zaimplementowany został algorytm klasyczny — o kwadratowej złożoności pamięciowej.

Interakcja z programem kończy się poprzez wybór wersji algorytmu: standardowej lub o liniowej złożoności pamięciowej.

Oznacza to, że zrealizowano pełną zakładaną funkcjonalność programu, uzyskując poprawną implementację algorytmu oraz spójne środowisko do jego obsługi.

3 Testowanie

Podstawowym kryterium poprawnego działania programu jest zwracanie przez niego takiej samej wartości oceny dopasowania oraz samego dopasowania, jak przez program bazujący na klasycznym algorytmnie Needlemana-Wunscha. Poprawność implementacji algorytmu Needlemana-Wunscha jest tu kluczowa. Szczęśliwie, algorytm ten jest stosunkowo prosty i intuicyjny, a zatem jesteśmy przekonani, że nasza implementacja jest prawidłowa.

Testowanie wersji z liniową złożonością pamięciową jest ważnym krokiem w trakcie tworzenia programu. Stwierdziliśmy, że program wyznacza prawidłowe wartości oceny dopasowania w każdym przypadku (prawidłowe oznacza tu, zgodne z wynikami algorytmu Needlemana-Wunscha). Z drugiej strony, jednoznaczne stwierdzenie poprawności uzyskanego najlepszego dopasowania jest nieco trudniejsze, gdyż (w pewnych przypadkach) możliwych jest wiele dopasowań o równej ocenie. Jednakże, jeśli dopasowanie otrzymane w wyniku działania algorytmu o liniowej złożoności pamięciowej jest takie samo jak algorytmu Needlemana-Wunscha świadczy to ponad wszelką wątpliwość o poprawnej implementacji algorytmu.

Program został przetestowany przy pomocy kilku sekwencji testowych o różnej długości i podobieństwie. Część z nich była (fragmentami) sekwencji dostępnych z bazy NCBI [7]. Przykładowo, wykorzystano fragmenty genu kodującego oksydazę

cytochromu c [8], który jak wiadomo jest homologiczny między wieloma organizmami.

Pozostałe fragmenty kodu zostały przetestowane w podstawowym zakresie, gwarantującym ich poprawne działanie.

4 Interfejs i implementacja

Interfejs użytkownika został wykonany w oparciu o technologię Adobe AIR — było to sugerowane środowisko. Takie podejście umożliwiło stworzenie „bogatego" interfejsu, przy wykorzystaniu wysokopoziomowej technologii, który dobrze komponuje się ze środowiskiem graficznym, w jakim użytkownik uruchamia narzędzie. Program zawiera pola do wprowadzania sekwencji, macierzy substytucji oraz innych parametrów wejściowych algorytmu.

Po uruchomieniu algorytmu (liniowego bądź klasycznego) program wyświetli ocenę dopasowania oraz dopasowane sekwencje.

Sam program jest zaimplementowany w Javie, gdyż takie narzędzie umożliwia najpełniejszą i najbardziej naturalną integrację ze środowiskiem Adobe AIR. Warto dodać, że poprzez wykorzystanie Javy oraz środowiska Adobe AIR projekt jest przenośny. Adobe AIR (od wersji 1.5) jest dostępny na platformie linux oraz MacOS. Sam zaś algorytm, zaimplementowany w Javie umożliwia uruchomienie na praktycznie dowolnej platformie wyposażonej w maszynę wirtualną Javy.

Literatura

[1] J. Tiuryn: *Wstęp do biologii obliczeniowej*, notatki do wykładu, http://www.mimuw.edu.pl/~tiuryn

[2] E. W. Myers, W. Miller: *Optimal alignment in linear space* Comp. Appl. Biosciences, 4:11-17, 1988

[3] Neil C. Jones, Pavel A. Pevzner: *Introduction to Bioinformatics Algorithms*, The MIT Press, 2004

[4] Dan Gusfield: *Algorithms on Strings, Trees, and Sequences*, Cambridge University Press

[5] http://www.ncbi.nlm.nih.gov/bookshelf/br.fcgi?book=toolkit&part=ch_algoalign#ch_algoalign.divide_and_conquer

[6] http://0-www.ncbi.nlm.nih.gov.millennium.unicatt.it/IEB/ToolBox/CPP_DOC/doxyhtml/mm__aligner_8cpp-source.html

[7] http://www.ncbi.nlm.nih.gov/

[8] http://www.ncbi.nlm.nih.gov/sites/homologene/5016