Implementacja algorytmu Nussinov do obliczania struktury drugorzędowej DNA i RNA

Jerzy Balcerzak, Oleg Druzhynets

June 11, 2014

1 CEL PROJEKTU

Celem projektu było opracowanie programu obliczającego strukturę drugorzędową cząsteczki DNA lub RNA wykorzystującego algorytm Nussinov.

2 UŻYTE TECHNOLOGIE

Stworzony program został całkowicie przygotowany w języku Python w standardowej wersji 2.7. Wybór padł właśnie na ten język programowania przede wszystkim ze względu na jego obiektowy charakter, przejrzystość składni oraz bogactwo bibliotek. Duża część kodu głównego algorytmu oraz modułu obsługi wejścia została przygotowywana w duchu metodyki Test Driven Developement (TDD). Świetnym narzędziem wykorzystanym podczas prac okazał się więc framework pyUnit. Ponaddto, do testów gotowego już programu wykorzystano dodatkowe narzędzia opisane w dalszych częściach poniższego sprawozdania. Wizualizacja otrzymanych wyników została oparta o bibliotekę networkx.

3 OSIĄGNIĘTA FUNKCJONALNOŚĆ APLIKACJI

W ramach projektu stworzono prostą w obsłudze aplikację konsolową (skrypt) nussinovCalculator.py. Podstawową funkcjonalność programu można opisać następująco:

- Pomoc dla użytkownika - wywołanie programu z parametrem -h wyświetla pulę obsługiwanych opcji oraz ich domyślne parametry.

- Wejście - wywołanie programu z opcją -i oraz nazwą pliku (np. ./nussinovCalculator.py -i "NazwaPliku.txt") spowoduje odczytanie z pliku ciągów nukleotydów (można podać ich wiele w kilku linijkach) i przeprowadzenie przewidywania struktury drugorzędowej dla każdego z nich. Program można wywołać bez tej opcji (wtedy domyślnie odczytywane są dane z pliku "input.txt"), jak również podać ciąg symboli nukleotydów prosto do konsoli używając opcji -r (np. -r GGGAAAACCC)
- Wyjście - domyślnie program wypisuje wynik na konsolę oraz zwraca plik "output.txt" w postaci listy par indeksów nukleotydów, które łączą się w strukturze drugorzędowej. Uzywając opcji -o można wskazać plik do którego dane mają być zapisane (np. -o outputs/out1.txt).
- Parametryzacja algorytmu - przy uruchamianiu programu użytkownik może dodatkowo wprowadzić:
  - minimalną ilość nukleotydów w pętli - używając opcji -m (domyślnie: -m 4),
  - oraz wartości w macierzy energii (opcja -e) w formacie listy list (np. -e [[0,0,0,2],[0,0,3,0],[0,3,0,3],[2,0,1,0]]). Przy czym kolejne kolumny i wiersze odnoszą się do nazw nukleotydów w kolejności alfabetycznej - A, C, G, U.
- Wizualizacja - po uruchomienu programu otwiera się ponaddto okno z prostą graficzną wizualizacją połączeń pomiędzy cząsteczkami.

4 Opis zastosowanego algorytmu

Prezentowany program rozwiązuje następujące zadanie: "Określ strukturę drugorzędową dla danej sekwencji nukleotydów". Do rozwiązania tego problemu użyto algorytmu zaproponowanego przez profesor Nussinov [1][2] opierającego swoje działanie na metodyce programowania dynamicznego. Ogólnikowo, przebieg algorytmu przedstawiają się następująco:

- Przygotowanie macierzy Nussinov oraz uzupełnienie jej wartości - implementacja tej części w głównej mierze została zainspirowana informacjami przekazanymi w wykładzie dr Roberta Nowaka [3])
- Przejście po otrzymanej macierzy w celu znalezienia par nukleotydów, tzw. Traceback. Istnieją różne możliwości implementacji tej procedury - w projekcie została wykorzystana jej elegancja wersja rekurencyjna (przedstawiona w materiałach udostępnionych przez uniwersytet w Tübingen [4])

5 Przeprowadzone testy

5.1 Zrealizowane testy

Zrealizowane zostały cztery testy typu „black box”: dwa testy czasu wykonania programu, profilowanie kodu linijka po linijce oraz profilowanie pamięci wykorzystywanej przez program:

- timeTest.py - testowanie czasu wykonania, korzysta z pomocniczej klasy Timer;
- timetest.sh - oszacowanie czasu wykonania programu z wykorzystaniem standardowej UNIX komendy time;
- line_profiler_script.py - profilowanie kodu linijka po linijce;
- memory_profiler_script.sh - profilowanie pamięci wykorzystywanej programem.

Przeprowadzone testy wskazują na poprawne wykonanie procedury i stosunkowo oszczędne zużycie zasobów.

5.2 Kontrola wydajności kodu linijka po linijce

Do testowania wydajności kodu skorzystaliśmy z profilera autorstwa Roberta Kerna [7]. Za pomocą tego narzędzia można zobaczyć jak szybko i jak często wykonywana jest każda linijka kodu w skrypcie. Żeby móc korzystać z line_profiler, najpierw należy go zainstalować:

```
pip install line_profiler
```

Zainstalowane zostaną: moduł line_profiler oraz skrypt kernprof.py. Dla testowania kodu korzysta się z dekoratora @profil, który należy ustawić przed interesującymi nas funkcjami. Nic więcej nie potrzeba importować, ponieważ skrypt kernprof.py samodzielnie wkleja potrzebny kod w trakcie uruchomienia naszego skryptu. Po uruchomieniu skryptu line_profiler_script.sh odbywa się testowanie interesujących nas funkcji. Wyniki testu zostają zapisane do pliku line_profiler_test.txt.

5.3 Profiler pamięci

Na podstawie profilera Roberta Kerna[7], Fabian Pedregose stworzył profiler pamięci memory_profiler [6], który też cechuje się prostym i niezawodnym działaniem. Z danego profilera korzysta się, aby sprawdzić ilość pamięci wykorzystywanej przez dany program. Żeby móc korzystać z memory_profiler, musimy go zainstalować:

```
pip install -U memory_profiler
pip install psutil
```

Instalujemy także psutil (skutkuje on lepszą wydajnością profilera). Dla testowania kodu również korzystamy z dekoratora @profile, którego ustawiamy przed interesującymi nas funkcjami. Po uruchomieniu skryptu memory_profiler_script.sh odbywa się testowanie interesujących nas funkcji. Wyniki testowania zostają zapisane do pliku memory_profiler_test.txt.

6 PORÓWNANIE DZIAŁANIA APLIKACJI Z KOMERCYJNYM ODPOWIEDNIKIEM

W celu dodatkowego potwierdzenia skuteczności naszego programu skonfrontowano jego działanie z komercyjnie dostępnym narzędziem: RNA Secondary Structure Prediction and Analysis on the web [5]
Porównywanie wyników predykcji struktury drugorzędowej sekwencji nukleotydów RNA pokazało, iż zaimplementowany przez nas algorytm Nussinov działa poprawnie. W porównaniu do wyżej wymienionego odpowiednika nasz program działa szybciej, ponieważ przeprowadza znacznie prostszą analizę danych (przeprowadza jedynie predykcję struktury drugorzędowej). Przygotowana przez nas wizualizacja wyników również pozostawia wiele do życzenia z porównaniem do swojego odpowiednika.

7 WNIOSKI

W ramach projektu został zaimplementowany algorytm Nussinov dla predykcji struktury drugorzędowej cząsteczek RNA. Stworzony program jest aplikacją konsolową, która pozwala użytkownikowi wprowadzać sekwencje nukleotydów oraz inne ważne parametry algorytmu. Wynikiem działania programu jest lista par indeksów tych nukleotydów które łączą się ze sobą w strukturze drugorzędowej - zwrócona na konsole oraz do pliku. Ponadto wyświetlona zostaje również graficzna interpretacja otrzymanych wyników. Jako źródło przykładowych danych zostały wykorzystane wybrane ciągi z bazy danych RNA STRAND v2.0 [8]. Ponadto przeprowadzone zostały różnego rodzaju testy wydajnościowe (czasowe i porównujące zużycie zasobów komputera). Porównując wyniki działania programu oraz już istniejących rozwiązań wnioskujemy, iż program działa poprawnie, a przyjęty algorytm jest skuteczny w teoretycznych rozważaniach nad strukturami drugorzędowymi DNA.

Literatura i źródła

[1] Nussinov, Ruth; Pieczenik, George; Griggs, Jerrold R.; Kleitman, Daniel J. (1 July 1978). "Algorithms for Loop Matchings". SIAM Journal on Applied Mathematics 35 (1): 68–82. doi:10.1137/0135006.

[2] Nussinov, R; Jacobson, AB (Nov 1980). "Fast algorithm for predicting the secondary structure of single-stranded RNA.". Proceedings of the National Academy of Sciences of the United States of America 77 (11): 6309–6313. doi:10.1073/pnas.77.11.6309. PMC 350273. PMID 6161375.

[3] Dr Robert Nowak. Slajdy do wykładu "Metody Bioinformatyki (MBI)"). Politechnika Warszawska, `http://staff.elka.pw.edu.pl/~rnowak2/dyd/mbi2014L/index.html`

[4] Prof. Dr. Daniel Huson, Dr. Christoph Dieterich . "Sript for Algorithms in Bioinformatics I Lecture". The University of Tübingen, `http://ab.inf.uni-tuebingen.de/teaching/ws06/albi1/script/protstruct2d_20Dec2006.pdf`

[5] RNA Secondary Structure Prediction and Analysis on the web http://rna.urmc.rochester.edu/RNAstructureWeb/Servers/Predict1/Predict1.html

[6] Memory profiler by Fabian Pedregose, http://ab.inf.uni-tuebingen.de/teaching/ws06/albi1/script/protstruct2d_20Dec2006.pdf

[7] Line profiler by Robert Kern, https://pythonhosted.org/line_profiler/

[8] RNA STRAND v2.0 - The RNA secondary STRucture and statistical ANalysis Database, http://www.rnasoft.ca/strand/