Marek Bajor
Kamil Gołofit

MBI - DOKUMENTACJA PROJEKTU

08.06.2015

1. Temat projektu

Temat nr 8: *Dopasowywanie sekwencji za pomocą algorytmów Gotoha oraz Altschula-Ericksona. Porównanie z inną dostępną implementacją (np. z pakietem Biostrings w języku R).*

2. Zakres projektu

W ramach projektu z przedmiotu MBI zaimplementowane zostały algorytmy Gotoha oraz Altschula-Ericksona. Skrypt testowy w języku R pozwala na wygenerowanie sekwencji DNA o zadanej długości do dopasowywania z alfabetu {A,C,T,G} oraz uruchamia implementacje algorytmów Gotoha, Altschula-Ericksona oraz implementację dostępną w pakiecie *Biostrings* języka R. Wyniki dopasowania oraz czasy wykonania są prezentowane użytkownikowi. Możliwe jest także wielokrotne generowanie sekwencji do dopasowania i wyznaczenie sumarycznego czasu wykonania dla poszczególnych algorytmów. Przykładowe testy, wyniki i wnioski zostały zamieszczone w rozdziale 4.

3. Problem dopasowywania sekwencji

Dopasowanie sekwencji polega na porównywaniu sekwencji nukleotydów w kwasach nukleinowych lub sekwencji aminokwasów w białkach w celu zidentyfikowania regionów wykazujących podobieństwo, mogące być konsekwencją powiązań pomiędzy sekwencjami. Jeśli dopasowywane sekwencje mają wspólne pochodzenie, niedopasowania mogą być interpretowane jako mutacje, delecje lub insercje, które zaszły w jednej lub obu liniach od momentu rozdzielenia sekwencji. Dopasowanie sekwencji może być także stosowane dla sekwencji niebiologicznych, np. danych finansowych, statystycznych lub sekwencji występujących w językach naturalnych.

Można wydzielić dwa podejścia do dopasowania sekwencji: dopasowanie globalne oraz dopasowanie lokalne. Dopasowanie globalne obejmuje całą długość wszystkich analizowanych sekwencji. Dopasowanie lokalne identyfikuje podobne rejony w obrębie długich sekwencji, które w całości nie muszą wykazywać znacznego podobieństwa.

4. Zaimplementowane algorytmy

a. Algorytm Gotoha

Algorytm Gotoha opisany w artykule *“An improved algorithm for matching biological sequences”* jest modyfikacją algorytmu Smitha-Watermana. Przedstawiona implementacja umożliwia globalne dopasowanie sekwencji DNA. Algorytm operuje na trzech macierzach definiujących punktację w zależności od wykrycia operacji mutacji, insercji lub delecji. Podczas dopasowywania sekwencji uwzględnia on nieliniową (afiniczną) karę za przerwy, przyjętą jako $f(x) \sim u*x + v$, gdzie u,v - parametry wejściowe algorytmu.

b. Algorytm Altschula-Ericksona

Algorytm Altschula-Ericksona jest rozwinięciem algorytmu Gotoha. Autorzy zauważyli, że algorytm Gotoha znajduje jedynie jedno z możliwie optymalnych rozwiązań, przy czym wskazali, że może się to kończyć niepowodzeniem.

Algorytm (nazwany w artykule SS-2) wprowadza bardziej złożone zależności pomiędzy kolejnymi porównaniami sekwencji, wykorzystywanymi do odtwarzania ścieżki do optymalnego rozwiązania. Wykorzystywanych jest 8 macierzy sterujących pracą algorytmu, przechowujących informacje na temat kierunków przejść w poszczególnych krokach algorytmu. Determinują one zachowanie algorytmu w przypadku wystąpienia kilku możliwości dalszego działania (np. identyczne wartości w porównywanych komórkach poprzedzających), co w innych algorytmach może być zachowaniem niedeterministycznym.

5. Testy i porównanie algorytmów

a. Porównanie jakości

Oba zaimplementowane algorytmy mają za zadanie znalezienie najlepszego globalnego dopasowania dwóch podanych sekwencji. Algorytmy te działają w odmienny sposób. Mimo, że oba wykorzystują afiniczną funkcję kary za niedopasowanie, algorytm Altschula-Ericksona stosuje bardziej zaawansowane metody wyszukiwania najlepszego rozwiązania. Działanie algorytmów zostało także porównane z resultatami funkcji *pairwiseAlignment()* zawartej w pakiecie *Biostrings*.
Funkcja kary użyta w testach to $F(k) = k+2$, k - numer kolejny przerwy.
Poniżej przedstawionych zostało kilka przykładowych rezultatów wykonania algorytmów dla wygenerowanych ręcznie (3, 4) lub losowo (1, 2) sekwencji DNA:
Przykład 1:
Porównywane ciągi:

```
A T G T C A A A C T T T A C T T G C C A A G A A C T A A G G
G C T A G A C T A T
```

Algorytm Gotoha:

```
C C A A G A A C T A A G G
G C T A G A - C T A T - -
```

Algorytm Altschula-Ericksona:

```
A T G T C A A A C T T T A C T T G C C A A G A A C T A A G G
G - - - - - - - - - - - - - - - - - C T - A G A C T - - A T
```

Algorytm pakietu Biostrings:

```
G T C A A A C T T T
G C T A G A C T A T
```

Przykład 2:
Porównywane ciągi:
```
G T A T C A C G C C G T C T G A T C T C
A T T A A
```
Algorytm Gotoha:
```
A T C T C
A T T A A
```
Algorytm Altschula-Ericksona:
```
G T A T C A C G C C G T C T G A T C T C
A - - - - - - - - - - - - T T A - - - - A
```
Algorytm pakietu *Biostrings*:
```
A T C T C
A T T A A
```

Przykład 3:
Porównywane ciągi:
```
G A A T T C
G A T T C
```
Algorytm Gotoha:
```
G A A T T C
G A - T T C
```
Algorytm Altschula-Ericksona:
```
G A A T T C
G - A T T C
```
Algorytm pakietu *Biostrings*:
```
G A A T T C
G A --T T C
```

Przykład 4:
Porównywane ciągi:
```
A C T G G T C A T
A C G G A T C G T A T
```
Algorytm Gotoha:
```
A C T G G T C - - A T
A C G G A T C G T A T
```
Algorytm Altschula-Ericksona:
```
A C T G G T - - C A T
A C G G A T C G T A T
```
Algorytm pakietu *Biostrings*:
```
A C T G G T C ----A T
A C G G A T C G T A T
```

Na podstawie uzyskanych wyników można stwierdzić, że w przypadku porównywania niewiele różniących się sekwencji, każdy z testowanych algorytmów daje podobny wynik. Niewielkie różnice mogą wynikać np. z preferowania jednego rodzaju przejść w przypadku algorytmu Gotoha, a użycia bardziej złożonych porównań w algorytmie Altschula-Ericksona.

Przy porównywaniu sekwencji losowych o niewielkich fragmentach dopasowanych, algorytmy dają różne wyniki, ale w uzyskanych wynikach można zaobserwować dobrze dopasowane fragmenty.

b. Porównanie wydajności

Czas wykonania zaimplementowanych algorytmów oraz implementacji dostępnej w ramach pakietu *Biostrings* języka R został zmierzony za pomocą funkcji *proc.time(),* która mierzy czas całkowity oraz czas zajętości procesora przez program napisany w języku R. W tabeli poniżej uwzględniony został czas zajętości procesora wykonaniem programu. Testy dla różnych długości sekwencji zostały przeprowadzone wielokrotnie, a ich czas zsumowany, aby zniwelować wpływ błędu pomiarowego na wyniki.

W testach wykorzystana została funkcja kary $F(k) = 10k+12$ jako najlepiej sprawdzająca się w testach jakościowych.

Długości sekwencji	Ilość powtórzeń	Gotoh [s]	Altschul-Erickson [s]	Biostrings [s]
10, 10	10	0.08	0.14	1.00
10, 10	100	1.14	1.20	11.75
10, 10	1000	10.76	13.36	113.99
100, 20	10	0.83	2.32	1.05
100, 20	100	7.86	23.22	10.18
100, 20	1000	78.87	237.22	95.11
500, 40	10	7.47	17.61	0.86
500, 40	50	37.71	88.11	4.42
500, 40	100	74.91	176.96	9.33
10000, 500	10	-	-	7.52
10000, 500	100	-	-	66.16

Na podstawie pomiarów można stwierdzić, iż dla małej długości ciągów (nieprzekraczających 100) algorytm Gotoha jest najbardziej wydajny. Przy dłuższych sekwencjach (a więc w praktycznym zastosowaniu) implemetacja zawarta w pakiecie *Biostrings* wykonuje się wyraźnie szybciej. Co więcej, czas wykonania dla implementacji dostępnej w pakiecie *Biostrings* rośnie wolniej niż liniowo przy zwiększaniu długości sekwencji do dopasowania. Niestety, zarówno implementacja Gotoha, jak i Altschula-Ericksona wypada na tym polu gorzej. Wraz ze wzrostem długości sekwencji wyraźnie szybciej rośnie czas wykonania.

Obserwując wyniki badań można również zauważyć że czas wykonania algorytmu Altschula-Ericksona jest 2-3 razy dłuższy niż algorytm Gotoha. Taka sytuacja wynika ze złożoności samego algorytmu A-E.

6. Podsumowanie

Algorytmy Gotoha oraz Altschula-Ericksona (A-E) dzięki zastosowaniu afinicznej funkcji kary pozwalają dużo dokładniej określić swoje priorytety przy operacji dopasowywania sekwencji niż algorytm Smitha-Watermana, z którego się wywodzą. Mogą więc być dopasowane do konkretnego zastosowania, co podnosi jakość ich działania w konkretnych przypadkach.

W ogólnych przypadkach dopasowywania sekwencji algorytm A-E sprawdza się lepiej niż algorytm Gotoha - wykorzystuje on bardziej zaawansowane metody wyszukiwania optymalnego rozwiązania, co przekłada się także na jego wolniejsze działanie. W przypadku dłuższych sekwencji o wiadomym dużym stopniu podobieństwa korzystniejsze ze względu czasu działania może się okazać jednak użycie algorytmu Gotoha - działa on szybciej, a dla takich sekwencji rezultaty są porównywalne z rezultatami uzyskanymi z algorytmu A-E.

Algorytm dostępny w ramach pakietu *Biostrings* języka R jest wydajniejszy niż zaimplementowane algorytmy. Zarówno dla dłuższych jak i krótszych sekwencji rozwiązania przez niego zwracane są poprawne. Dużym jego plusem jest także skalowalność, jednak dla krótkich sekwencji (o długości poniżej ok. 100) czas jego wykonania jest zaskakująco długi. Może to wynikać ze stopnia obliczeń koniecznych do uzyskania poprawnych wyników.