Sprawozdanie z laboratorium: Bioinformatyka (szablon)

Część I: Analiza teoretyczna

28 marca 2017

Prowadzący: prof. dr hab. inż. Marta Kasprzak

Autorzy:	**Damian Jurga**	inf.....	I2	jasiu@serwer.domena.poczta.pl
	Grzegorz Miebs	inf122453	I2	grzegorz.miebs@student.put.poznan.pl

Zajęcia środowe, 11:45.

Oświadczamy, że niniejsze sprawozdanie zostało przygotowane wyłącznie przez powyższych autorów, a wszystkie elementy pochodzące z innych źródeł zostały odpowiednio zaznaczone i są cytowane w bibliografii.

1 Wstęp

Celem tego sprawozdania jest przedstawienie teoretycznego opracowania metody heurystycznej rozwiązującej problem sekwencjonowania łańcuchów DNA z błędami pozytywnymi oraz negatywnymi w czasie wielomianowym. Alogrytm mając dany na wejściu zbiór oligonukleotydów (tj. ciągów nukleotydów: adeniny, tyminy, guaniny i cytozyny), długość sekwencji oryginalnej, powinien zwrócić sekwencję jak najbardziej zbliżoną do oryginalnej.

2 Sformułowanie problemu

Elementy zbioru oligonukleotydów S zostały zakodowane jako słowa o długości l nad alfabetem nukleotydów $\{A, C, G, T\}$. Moc zbioru S wynosi n.

W przypadku idealnym kolejne słowa pokrywają się na $l-1$ pozycjach; innymi słowy słowa powstałe przez usunięcie odpowiednio pierwszej lub ostatniej litery z dwóch następujących po sobie elementów sekwencji wynikowej są równe. W takim wypadku sekwencja wyjściowa ma długość równą mocy zbiory S.

Ze względu na charakter źródła zbiorów wejściowych, S może zawierać błędy dwojakiej natury. Pojawienie się dodatkowych słów, nieobecnych w przypadku idealnym, nazywa się błędem pozytywnym, a brakujące dane (tj. brak słów bezpośrednio następnych, brak kolejnych wystąpień słów) — negatywnym. W przypadku obecności jedynie błędów negatywnych sekwencja wyjściowa nie jest dłuższa niż n i zawiera wszystkie słowa z S; alternatywnie, w obecności jedynie błędów pozytywnych, sekwencja ma długość n i zawiera $n - l + 1$ słów z S. W przypadku ogólnym, powinna zawierać maksymalną liczbę słów z S oraz nie powinna być dłuższa niż n.

Problem można przedstawić jako skierowany graf pełny $G = (S, \{(s_i, s_j) : (s_i, s_j) \in S \times S \wedge s_i \neq s_j\})$, który z każdym wierzchołkiem ma utożsamiony zysk równy 1, a z każdym łukiem — koszt odpowiadający liczbie niepokrywających się pozycji $k_{i,j}$ (w przypadku idealnym słowa pokrywają się na $l-1$ pozycjach, a ogólnie na $l - k_{i,j}$; $k_{i,j} \in \{1, 2, ..., l\}$). Jako taki znalezienie najlepszej sekwencji sprowadza się do rozwiązania wariantu selektywnego problemu komiwojażera nie wymagające utworzenia cyklu, a jedynie ścieżkę.

3 Algorytm

3.1 Opis

Do rozwiązania tego problemu zbudujemy graf, którego wierzchołkami będą słowa ze zbioru S, a wartości łuku między wierzchołkami będą równe przesunięciu między tymi słowami. Przykładowo łuk z wierzchołka ACCGT do wierzchołka CCGTC będzie miał wartość 1 a do wierzchołka GTCGT wartość 3. Na skonstruowanym w ten sposób grafie rozwiązujemy problem komiwojażera maksymalizujący liczbę odwiedzonych wierzchołków przy ograniczeniu na sumę wartości wykorzystywanych łuków, która nie może być większa od $n-l$, gdyż w przeciwnym razie na wyjściu otrzymamy sekwencję dłuższą niż oryginalna. Aby ograniczyć ponowne odwiedzanie tych samych wierzchołków, będziemy zwiększać wartość łuków prowadzących do odwiedzonych już wierzchołków o pewną stałą C. Do rozwiązania problemu komiwojażera posłużymy się przeszukiwaniem wiązkowym oraz algorytmem wspinaczki.

3.2 Lista kroków

1. Zbudowanie grafu

2. Zaczynamy z losowego wierzchołka
3. Znajdujemy k najbliższych wierzchołków
4. Do każdej z k dotychczasowych ścieżek liczymy odległość po dodaniu każdego z wierzchołków z uwzględnieniem kary za powtórne odwiedzenie tego samego wierzchołka.Wybieramy k najkrótszych ścieżek
5. Powtarzamy krok 4 aż koszt ścieżek przekroczy krytyczną wartość $n - l$
6. Dla każdej z k wygenerowanych ścieżek sprawdzamy które przestawienie parami pozwoli maksymalnie zmniejszyć koszt ścieżki i wykonujemy je
7. Powtarzamy krok 6 i razy, bądź aż do uzyskania lokalnego optimum
8. Jeśli długość ścieżki jest mniejsza od $n-l$ o więcej niż z to ponownie wykonujemy przeszukiwanie wiązkowe w celu rozszerzenia aktualnych ścieżek. W przeciwnym wypadku dokonujemy pełnego przeglądu.

3.3 Złożoność obliczeniowa

Podczas każdej iteracji przeszukiwania wiązkowego badane jest n możliwych rozszerzeń każdej z k ścieżek ($k < n$), a ponieważ maksymalna długość ścieżki jest równa maksymalnej długości sekwencji — złożoność pierwszego przeszukiwania wiązkowego jest równa $O(k * |S|^2)$. Złożoność drugiego przeszukiwania wiązkowego wynosi $O(k * z * |S|)$, dlatego że w najgorszym przypadku należy znaleźć ścieżkę o długości z. Aby były spełnione warunki zadania, z powinno spełniać następującą zależność: $kn(z-1) >= (z-1)!$.

Algorytm wspinaczkowy wymaga co najwyżej $\binom{n}{2} \sim n(n-1)$ zamian podczas każdej iteracji; ponieważ w ogólności istnieje nie więcej niż $n!$ możliwych permutacji sekwencji, zaproponowano by ograniczyć liczbę iteracji od góry pewną stałą i ($i < n!$). Złożoność takiego algorytmu wspinaczkowego jest równa $O(i * |S|^2)$.

Złożoność całego algorytmu jest więc równa $O((k * |S| + k * z + i * |S|) * |S|)$. Stałe k,z oraz i najłatwiej byłoby wyznaczyć eksperymentalnie.

4 Usprawnienia

4.1 Tylko błędy pozytywne

Gdyby założyć, że w danym zestawie danych jedyne błędy jakie występują to błędy pozytywne, moglibyśmy wówczas podczas przeszukiwania wiązkowego wycofywać się z wierzchołków, które są odległe o więcej niż 1 od każdego innego wierzchołka. Pozwoliłoby to ominąć również część błędów pozytywnych. Moglibyśmy także brać pod uwagę tylko i wyłącznie łuki o wartości 1, jednak wówczas algorytm przeszukiwania wiązkowego mógłby wielokrotnie przechodzić po tych samych łukach, odwiedzając kilkukrotnie te same wierzchołki, pomijając jednocześnie inne. Z tego powodu sensowne wydaje się pozostawienie także łuków o większych wartościach. Próg powyżej którego łuki nie będą brane pod uwagę, zwiększający skuteczność algorytmu będzie zależny od topologii sieci utworzonej z łuków o wartości mniejszej bądź równej temu progowi. Jeśli sieć ta będzie odpowiednio gęsta, wówczas szansa na wielokrotne odwiedzanie tych samych wierzchołków spada. Optymalną gęstość takiej sieci najłatwiej będzie ustalić eksperymentalnie.

4.2 Tylko błędy negatywne

Przy założeniu, że w danym zestawie danych występują tylko i wyłącznie błędy negatywne algorytm powinien odwiedzić wszystkie wierzchołki, a więc nie mogą pojawić się powtórzenia i stała C, karająca za ponowne odwiedzenie tego samego wierzchołka może być wówczas równa na przykład n. Ponieważ przeszukiwanie wiązkowe nie zapewnia uzyskania optymalnej sekwencji umożliwienie powtórzeń, choć utrudniając je tak, by algorytm znacznie rzadziej, bądź nigdy nie odwiedzał tych samych wierzchołków, jeśli będą dostępne do tej pory nieodwiedzone, może okazać się korzystne, pod warunkiem eliminacji ich z finalnego rozwiązania. Nawet jeśli spowoduje to, że ścieżka przebiegać będzie w sposób odległy od optymalnego, istnieje spore prawdopodobieństwo, że zostanie ona poprawiona w drugim etapie czyli algorytmie wspinaczki.