Spis rzeczy

7 Sekwencjonowanie na dużą skalę (składanie sekwencji) 108
7.1 Skierowane sekwencjonowanie 108
7.2 Strategia sekwencjonowania na dużą skalę 109
7.3 Składanie sekwencji dla metody 'shotgun' 110
7.3.1 Wykrywanie przecięć 110
7.3.2 Układanie podsłów 111
7.3.3 Decydowanie konsensu 111
7.3.4 Scalanie całej sekwencji 112
7.4 Problem najkrótszego wspólnego nadsłowa 112
7.4.1 Permutacje a nadsłowa 113
7.4.2 Cykliczne słowa i cykliczne pokrycia 113
7.4.3 Cykliczne pokrycia a nadsłowa 114
7.4.4 Konstrukcja cyklicznych pokryć 114
7.4.5 Algorytm aproksymacyjny 115
7.5 Sekwencjonowanie przez hybrydyzację 115

7 Sekwencjonowanie na dużą skalę (składanie sekwencji)

Podstawowym problemem z istniejącymi metodami sekwencjonowania (Maxam-Gilberta, Sangera) jest to, że można sekwencjonować za jednym razem tylko nici DNA o małej długości (300-1000 zasad). Tak więc długie odcinki DNA trzeba dzielić na małe kawałki i te, po zsekwencjonowaniu składać z powrotem. Omówimy kilka metod sekwencjonowania i związane z nimi problemy kombinatoryczne.

7.1 Skierowane sekwencjonowanie

Przyjmijmy, że możemy zsekwencjonować w jednym laboratoryjnym doświadczeniu 400 zasad. Metoda *kroczenia po primerach* polega tym, że najpierw sekwencjonujemy najbardziej lewy odcinek L_1 zawierający 400 zasad. Wybieramy jako primer odcinek położony blisko prawego końca L_1 i używamy go w procesie PCR do zrobienia kopii. Kopia ta będzie się zaczynała od

tego primera, czyli od miejsca w okolicy zasady nr. 400. Proces powtarzamy, sekencjonując za każdym razem najbardziej lewy odcinek nowej kopii zawierający 400 zasad. Problemy z tą metodą są następujące:

- Proces jest sekwencyjny i dlatego bardzo wolny.
- Jeśli z jakiegoś powodu proces skierowanego sekwencjonowania zatrzyma się w pewnym miejscu nici DNA (np. przez wystąpienie powtarzających sie podsłów lub zapętlenie się nici DNA w tzw. szpilkę do włosów (ang. hairpin), to nie daje się zsekwencjonować części nici DNA leżącej na prawo od tego miejsca.

Główne zastosowanie tej metody to sekwencjonowanie krótkich nici bez powtarzających się podsłów lub wypełnianie małych dziur powstałych w wyniku sekwencjonowania inną metodą.

7.2 Strategia sekwencjonowania na dużą skalę

Zaczynamy od całej nici DNA (np. długości 10^8 bp). Tworzymy bibliotekę dużych klonów (10^5-10^6 bp każdy) przez cięcie enzymami, które rzadko tną. Każdy klon jest namnażany poprzez wektory wprowadzane do biologicznego układu, który replikuje klony raz z ze swoim materiałem genetycznym. Typowe wektory dla dużych klonów to *sztuczne chromozomy drożdża* (YAC, Yeast Artificial Chromosome). Są one otrzymywane z drożdża *S. cerevisiae* przez pozostawienie jedynie tych składników chromozomu, które są niezbędne do replikacji. Natępnie wybiera się tyle klonów aby zapewnić pokrycie całej wyjściowej nici DNA i tworzy się mapę fizyczną.

W kolejnym kroku zajmujemy się każdym z dużych klonów z osobna. Taki klon jest powielany i trawiony enzymem o większej częstości cięcia. Tak otrzymane fragmenty (zwykle o długości około 40.000 bp) wstawiamy do wektorów (zwykle są to kosmidy otrzymane z faga λ) i wstrzykujemy do *E. coli* w celu replikacji. Następnie wybieramy tyle klonów (tych pochodzących od kosmidów) aby zapewnić pokrycie całego klonu pochodzącego od YAC'a i budujemy mapę fizyczną poszczególnych klonów pochodzących od YAC'a.

Natępnie stosujemy metodę sekwencjonowania zwaną *shotgun* (nazywaną też metodą losowego sekwencjonowania) do każdego klonu kosmidowego. Metoda ta polega na losowym pocięciu DNA takiego klonu na odcinki o małej długości (około 400 bp). Oczywiście w procesie tym całkowicie gubimy porządek w jakim te odcinki występują na nici. Wybieramy po kolei te odcinki i dla każdego z nich sekwencjonuje się pierwsze 400 bp. Kawałków tych musi być dostatecznie dużo, aby zapewnić sobie możliwość odtworzenia całego klonu.

Jak wynika z analizy probabilistycznej, jeśli L jest długością klonu kosmidowego, to liczba fragmentów wybranych do sekwencjonowania powinna wahać się w granicach od $5L/400$ do $10L/400$.

7.3 Składanie sekwencji dla metody 'shotgun'

Odtwarzanie docelowej sekwencji odbywa się w czterech krokach. Przyjmijmy, że mamy zsekwencjonowane krótkie kawałki.

7.3.1 Wykrywanie przecięć

Dla każdej uporządkowanej pary kawałków chcemy stwierdzić czy sufiks pierwszego pasuje do prefiksu drugiego. Gdyby nie było błędów procesu sekwencjonowania to wystarczyłoby znalezienie najdłuższego dopasowania jak najdłuższego sufiksu z prefiksem (żadnych spacji). To można zrobić w liniowym czasie (zależnym od sumy długości kawałków). Z powodu błedów musimy użyć miary podobieństwa zamiast dopasowania.

Czyli zadanie do rozwiązania wygląda następująco: dane S_1, S_2, należy znaleźć sufiks S_1 oraz prefiks S_2, dla których podobieństwo jest maksymalne. Funkcja podobieństwa jest dobierana w zależności od rodzaju DNA, z którym mamy do czynienia.

Powyższe zadanie rozwiązujemy stosując programowanie dynamiczne (tak jak do lokalnego uliniowienia). Jeśli budowana macierz dla tej metody ma wiersze odpowiadające pozycjom w słowie S_1, a kolumny odpowiadające pozycjom w słowie S_2, to szukamy maksimum w ostatnim wierszu macierzy. Natępnie, używając zapamiętywanych wskaźników, cofamy się aż do pierwszej kolumny. Jeśli (n, i) jest miejscem w ostatniej kolumnie, gdzie znajduje się maksimum, a $(1, j)$ jest miejscem, do którego dojdziemy w pierwszej kolumnie, to optymalne uliniowienie dostajemy dla $S_1[j..n]$ oraz $S_2[1..i]$.

Czas zużyty to $O(n^2)$. Tak więc mając k słów dostajemy pełna informację dla wszystkich par w czasie $O(k^2n^2)$. Ten czas jest wąskim gardłem całej metody.

Heurystyczne ulepszenia (wykrywania przecięć)

Poziom będów powstałych w sekwencjonowaniu jest dość mały (rzędu 5%). Z kolei rozmiary przecięć zachodzących na siebe odcinków powinny być dość duże. Można więc oczekiwać, że nakładające się kawałki mają długie wspólne podsłowo (to jak długie – dobiera się heurystycznie). Znalezienie najdłuższego wspólnego podsłowa można zrobić w czasie $O(n)$. Można to dalej ograniczać do części słów (np. ostatnia ćwiartka pierwszego słowa i pierwsza ćwiartka

drugiego). Tak wię w czasie $O(k^2 n)$ możemy poznać informację o wspólnych podsłowach. Informacja ta pozwala odrzucić dużo par, których nie podejrzewamy o przecięcie. Dla pozostałych par stosujemy metodę dynamicznego programowania.

7.3.2 Układanie podsłów

Stosujemy *metodę zachłanną*. Wybieramy parę o najwyższym podobieństwie sufiks/prefiks i uliniowiamy ją ze sobą (możemy stosować spacje). Następnie wybieramy parę o następnym najwyższym podobieństwie sufiks/prefiks. Możliwe są dwie sytuacje. Albo jedno ze słów tej pary już występuje w zbudowanym uliniowieniu — wówczas 'doklejamy' uliniowienie nowej pary do już utworzonego uliniowienia (w ten sposób powstaje uliniowienie trzech słów, tworzące tzw. *kontig*). W przeciwnym przypadku budujemy osobne uliniowienie dla drugiej pary — powstają w ten sposób dwa kontigi. Proces powtarzamy tak długo aż wyczerpiemy wszystkie pary.

7.3.3 Decydowanie konsensu

Załóżmy, że mamy już zdecydowany układ podsłów (dla danego klonu kosmidowego). Podsłowa te zachodzą na siebie (wielokrotnie) tak, że możemy rozważać znaki występujące w jednej kolumnie. Z całą pewnością będziemy mieli bardzo dużo niezgodności, które trzeba usunąć. Rozważmy jedną kolumnę znaków. Jeśli występuje w niej tylko jeden symbol, to przyjmujemy, że jest to symbol stojący w tym miejscu. W przeciwnym przypadku musimy zdecydować co ma stać w tym miejscu. Stosowane są różne podejścia:

- Dla miejsc niezgodnych podajemy profil takiego miejsca (częstość wystąpienia liter), pozostawiając użytkownikowi ostateczną decyzję.
- Konsens metodą większościowego głosowania: wygrywa litera pojawiająca się największą liczbę razy.
- Wybieramy 'okna' zawierające fragmenty o dużej niezgodności. Dla każdego okna wykonujemy wielokrotne uliniowienie.
- Jeśli jest dużo niezgodności, to poprawiamy całość procesu układania podsłów (por. Sekcja 7.3.2) przez sukcesywne uliniawianie całego otrzymanego dotąd profilu z kolejnym podsłowem. Kolejność wybierania podsłów jest ta sama co w fazie układania podsłów.

7.3.4 Scalanie całej sekwencji

Dla każdego klonu YAC budujemy sekwencję z klonów kosmidowych, stosując mapę tego klonu. Natępnie korzystając z mapy całego DNA, układamy całą sekwencję. Wynik jest kontrolowany (przez człowieka) pod względem zgodności. Możliwe są iteracje procesu, wprowadzające poprawki drobniejszych kawałków (np. metodą sekwencjonowania skierowanego).

Problemy związane z tą metodą:

- Wynik musi być sprawdzany/poprawiany przez człowieka.
- Powtarzające się fragmenty genomu oraz istniejące klony chimeryczne wprowadzają dużo błędów (grubego kalibru).

7.4 Problem najkrótszego wspólnego nadsłowa

Dany zbiór $\mathcal{P} = \{S_1, \ldots, S_k\}$, znaleźć najkrótsze słowo S takie, że każde S_i jest podsłowem S. Oczywiście konkatenacja wszystkich słów $S_1 \cdots S_k$ jest jednym z nadsłów (zwykle nie jest ono najkrótsze).

Przykład 7.4.1 Niech $\mathcal{P} = \{abcc,\ efab,\ bccta\}$. Nadsłowami są

$$bcctabccefab$$

oraz

$$efabccta.$$

□

Motywacją dla tego problemu jest problem scalania dla metody shotgun. Niestety stosuje się tylko wtedy gdy nie ma będów (a więc jest zupełnie teoretyczny).

Twierdzenie 7.4.1 *Problem najkrótszego wspólnego nadsłowa jest NP-zupełny (tzn. jego decyzyjna wersja) oraz MAX-SNP trudny.*

Podamy szkic algorytmu aproksymacyjnego o ilorazie błędu 4. Problem ten był intensywnie badany, a znajdowanie algorytmów aproksymacyjnych o coraz mniejszym błedzie stało sie 'dyscypliną sportową'. Poniżej podajemy listę osiągnieć w tej dyscyplinie.

- Blum, Jiang, Tromp, Yannakakis (1993): 3

- Teng, Yao (1993): 2.89
- Czumaj, Gąsieniec, Piotrów, Rytter (1994): 2.83
- Kosaraju, Park, Stein (1994): 2.71
- Armen, Stein (1994): 2.75
- Jiang, Jiang (1995): 2.67
- Armen, Stein (1996): 2.67
- Breslauer, Jiang, Jiang (1996): 2.596
- Sweedek (1995): 2.5

7.4.1 Permutacje a nadsłowa

Dana permutacja $\pi : \{1, \ldots, k\} \to \{1, \ldots, k\}$. Nadsłowo wyznaczone przez π konstruowane jest następująco. Niech $ov(S_i, S_j)$ oznacza długość najdłuższego słowa W takiego, że W jest sufiksem S_i oraz W jest prefiksem S_j. Niech $pref(S_i, S_j$ będzie słowem spełniającym warunek $S_i = pref(S_i, S_j)W$, gdzie W jest jak wyżej. Wówczas

$$S(\pi) = pref(S_{\pi(1)}, S_{\pi(2)}) \cdots pref(S_{\pi(2)}, S_{\pi(3)}) \cdots pref(S_{\pi(k-1)}, S_{\pi(k)}) \cdot S_{\pi(k)}$$

jest nadsłowem dla $\{S_1, \ldots S_k\}$.

Zauważmy, że jeśli S_* jest najkrótszym nadsłowem, to istnieje permutacja π taka, że $S_* = S(\pi)$. Łatwo jest pokazać, że

$$|S(\pi)| = \sum_{i=1}^{k} |S_i| - \sum_{i=1}^{k-1} ov(S_{\pi(i)}, S_{\pi(i+1)}).$$

Zatem będziemy maksymalizować sumę $\sum_{i=1}^{k-1} ov(S_{\pi(i)}, S_{\pi(i+1)})$.

7.4.2 Cykliczne słowa i cykliczne pokrycia

Dane słowo W, *cykliczne słowo* $\phi(W)$ jest traktowane tak, że ostatni symbol W porzedza pierwszy symbol, tzn. $W(n+1) = W(1)$, gdzie $n = |W|$.

Powiemy, że słowo S *zanurza się* w $\phi(W)$, gdy S jest podsłowem słowa $W^\omega = W \cdot W \cdots$. Niech $\mathcal{P} = \{S_1, \ldots, S_k\}$ będzie skończonym zbiorem słów. *Cykliczne pokrycie* zbioru $\mathcal{P}$ jest to każdy zbiór $C(\mathcal{P})$ cyklicznych słów o tej

własności, że każde S_i zanurza się w pewne ze słów ze zbioru $C(\mathcal{P})$. Długość cyklicznego pokrycia to

$$||C(\mathcal{P})|| = \sum_{\phi(W)\in\mathcal{P}} |W|.$$

Lemat 7.4.2 *Jeśli S_* jest najkrótszym nadsłowem dla $\mathcal{P}$ oraz $C(\mathcal{P})$ jest cyklicznym pokryciem o minimalnej długości, to*

$$||C(\mathcal{P})|| \leq |S_*|.$$

7.4.3 Cykliczne pokrycia a nadsłowa

Załóżmy, że $C(\mathcal{P}) = \{\phi(W)\}$ jest jednoelementowym cyklicznym pokryciem zbioru $\mathcal{P} = \{S_1, \ldots, S_k\}$. Pokrycie to wyznacza pewną permutację (a zatem i nadsłowo) w następujący sposób. Niech $(i_1, \ldots, i_k)$ oznacza kolejność, z jaką słowa $S_1, \ldots, S_k$ zanurzają się w $\phi(W)$ (bierzemy pod uwagę pozycje w W początku każdego ze słów S_i). Zakładamy, że żadne S_i nie jest podsłowem żadnego S_j, dla $i \neq j$. Zatem $(i_1, \ldots, i_k)$ jest permutacją.

7.4.4 Konstrukcja cyklicznych pokryć

Budujemy macierz A taką, że $A(i,j) = ov(S_i, S_j)$, dla dowolnych $i, j \leq k$. Dla macierzy A rozwiązujemy *problem pełnego przypisania*. Polega on na tym, że wybieramy k pozycji w macierzy tak, aby żadne dwie pozycje nie stały w jednym wierszu ani w jednej kolumnie macierzy oraz aby suma wartości stojących na tych pozycjach była maksymalna.

Uwaga: macierz A można zbudować w czasie $O(\sum_{i=1}^{k} |S_i| + k^2)$, natomiast zwykły algorytm zachłanny rozwiązujący problem pełnego przypisania działa w czasie $O(n^3)$ (jeśli nie liczyć czasu porównywania elementów).

Tak wybrane pozycje wyznaczają pewnien zbiór cykli w grafie o k wierzchołkach. Krawędź od i do j istnieje w tym grafie, gdy pozycja (i,j) jest wybraną pozycją macierzy A. Każdy cykl $i_1, \ldots, i_t$ wyznacza słowo cykliczne $\phi(W)$, gdzie

$$W = pref(S_{i_1}, S_{i_2}) \cdots pref(S_{i_{t-1}}, S_{i_t} \cdots pref(S_{i_t}, S_{i_1}).$$

Wówczas $\phi(W)$ tworzy pokrycie dla $\{S_{i_1}, \ldots, S_{i_t}\}$. Jeśli $W_1, \ldots, W_m$ są słowami utworzonymi dla wszystkich cykli pochodzących z rozwiązania problemu przypisania, to $\{\phi(W_1), \ldots, \phi(W_m)\}$ jest minimalnym cyklicznym pokryciem dla $\{S_1, \ldots, S_k\}$.

7.4.5 Algorytm aproksymacyjny

Dane k słów $S_1, \ldots, S_k$.

1. Znajdź minimalne pokrycie cykliczne $\{\phi(W_1), \ldots, \phi(W_m)\}$.
2. Dla każdego z cyklicznych słów utwórz nadsłowa $S_*^{(1)}, \ldots, S_*^{(m)}$.
3. Jako wynik podaj konkatenację $W = S_*^{(1)} \cdots S_*^{(m)}$.

Twierdzenie 7.4.3 *Jeśli S_* jest najkrótszym nadsłowem dla $\{S_1, \ldots, S_k\}$ oraz W jest słowem znalezionym przez powyższy algorytm, to*

$$\frac{|W|}{|S_*|} < 4$$

.

W dowodzie powyższego twierdzenia wykorzystane są dwa ważne narzędzia. Zacznijmy od definicji. Powiemy, że słowo S ma *okres* p, gdy istnieje słowo W takie, że $|W| = p$ oraz S można przedstawić w postaci $U \cdot W^k$, gdzie $k > 0$ oraz U jest niepustym sufiksem słowa W.

Twierdzenie 7.4.4 (O największym wspólnym dzielniku okresów) *Jeśli S ma okresy p oraz q i $|S| > p + q$, to S ma okres będący największym wspólnym dzielnikiem liczb p i q,*

Twierdzenie 7.4.5 (Lemat o zazębianiu) *Jeśli $\phi(W_1)$ oraz $\phi(W_2)$ są cyklicznymi słowami oraz U_i zanurza się w $\phi(W_i)$, dla $i = 1, 2$, to*

$$ov(U_1, U_2) < |W_1| + |W_2|.$$

7.5 Sekwencjonowanie przez hybrydyzację

Metoda ta polega na tym, że przygotowujemy fizyczną matrycę *chip* zawierającą wszystkie kombinacje nukleotydów o długości k (parametr k jest dobierany, zwykle ok. 10). Tak więc mamy 4^k słów. Mając nić DNA, którą chcemy zsekwencjonować robimy dużo kopii tej nici, znakujemy je radioaktywnie, a następnie hybrydyzujemy te kopie w obecności matrycy. Pewne cząsteczki DNA hybrydyzują się z pewnymi słowami z matrycy, pozostałe (wolne) nici DNA usuwamy. Informacja jaką w ten sposób zdobywamy to zbiór wszystkich słów k elementowych, których komplementarne kopie są zawarte w nici DNA. W informacji tej nie jest zawarte to ile razy dane słowo

długości k jest zawarte w nici. Jeśli k jest dostatecznie duże to każdy odcinek wystąpi co najwyżej jeden raz.

Tak więc mamy następujący *problem sekwencjonowania przez hybrydyzację (SBH)*. Dany zbiór słów k elementowych, które są podsłowami pewnego słowa S (S nie znamy). Znaleźć S lub, jeśli S nie jest jednoznacznie wyznaczone, jak najwięcej informacji o S. Przedstawiona redukcja problemu SBH do znajdowania ścieżek Eulera pochodzi od P. Pevznera.

Dane k i pewien zbiór L wszystkich k elementowych podsłów pewnego słowa S. Tworzymy graf skierowany $G(L)$:

- wierzchołkami są wszystkie słowa długości $k-1$.
- krawędź od wierzchołka W_1 do W_2 istnieje i jest etykietowana literą a, gdy istnieje słowo $U \in L$ oraz litera b takie, że $U = W_1 \cdot a$ oraz $U = b \cdot W_2$.

Usuwamy izolowane wierzchołki z $G(L)$. Przypmnijmy, że *ścieżka Eulera* to dowolna droga w grafie przechodząca przez każdą krawędź grafu dokładnie jeden raz. Każda droga w grafie $G(L)$ opisuje pewne słowo: najpierw bierzemy etykietę wierzchołka początkowego tej drogi, a później wszystkie etykiety krawędzi, w kolejności ich przechodzenia. Powiemy, że słowo S jest *kompatybilne* z L, gdy S zawiera jako podsłowo każde ze słów z L i nie zawiera żadnych innych podsłów. Natępujące twierdzenie jest łatwe do dowodu.

Twierdzenie 7.5.1 *Słowo S jest kompatybilne z $L \iff S$ można otrzymać z drogi Eulera w grafie $G(L)$.*

Okazuje się, że pomiędzy każdymi dwoma słowami kompatybilnymi z L istnieje przejście przy pomocy elementarnej operacji zwanej rotacją. Dane słowo S. Powiemy, że słowo S' otrzymuje się z S przez *rotację*, gdy S można przedstawić $S = S_1S_2S_3S_4S_5$ tak, że $S' = S_1S_4S_3S_2S_5$.

Twierdzenie 7.5.2 (Ciągłość SBH)
Jeśli S i S' są kompatybilne z L, to S'' można otrzymać z S przez ciąg rotacji.

Dowód: Podamy szkic dowodu. Niech P będzie drogą Eulera dającą słowo S i niech P' będzie drogą Eulera dla słowa S'. Pokazuje się, że można przyjąć, że obie drogi zaczynają sie w tym samym wierzchołku. Niech A_1 będzie najdłuższym początkowym fragmentem wspólnym dla obu dróg. Pokazuje się, że dalszą cześć drogi P można rozbić na fragmenty B_1, B_2, C_1, C_2 (tzn. $P = AB_1B_2C_1C_2$) o tej własności, że B_1B_2 powraca do wierzchołka, w którym kończy się A oraz fragmenty B_1 i C_1 kończą się w tym samym wierzchołku.

Wówczas jeśli weźmiemy drogę Eulera $P'' = AC_1B_2B_1C_2$, to pokrywa się ona z P' w początkowym fragmencie na odcinku o długości co najmniej o jeden większej niż $|A|$. ■