Grupa Wtorek 9.15	Michał Jaworek, Marcin Kaciuba	11/01/2015

Algorytm Smitha-Watermana - poszukiwanie optymalnych lokalnych dopasowań sekwencji

Abstrakt

Algorytmy dopasowania sekwencji znajduje swoje zastosowanie m. in. w bioinformatyce do poszukiwań dopasowań sekwencji nukleotydów i aminokwasów. Algorytm Smitha-Watermana należy do podgrupy rozwiązań zajmujących jest tzw. dopasowaniem lokalnym. W poniższym dokumencie przedstawiono opis prób zrównoleglenia tego algorytmu z wykorzystaniem technologii CUDA.

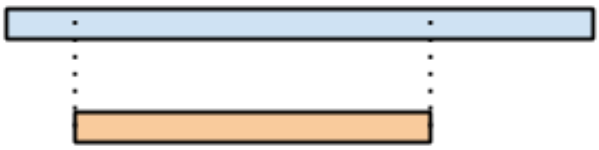
Przedostawienie problemu

Problem dopasowania sekwencji przyjmuje na wejściu dwa ciągi znaków. W ogólnym przypadku, ciągi te mogą składać się z liter dowolnego alfabetu. W przypadku zastosowań bioinformatyczych zazwyczaj ten alfabet jest relatywnie niewielki (np. czteroznakowy "TGAC").

Problem tej klasy można interpretować na dwa sposoby. Istnieją rozwiązania analizujące:

- dopasowanie globalne
- dopasowanie lokalne

W przypadku dopasowania globalnego dwa ciągi porównywane są wzdłuż całej sekwencji. Takie rozwiązanie jest wykorzystywane przy analizie jednodomenowych białek. Algorytmem tego typu jest na przykład algorytm Needlemana-Wunscha. Schemat takiego dopasowania przedstawiono poniżej:

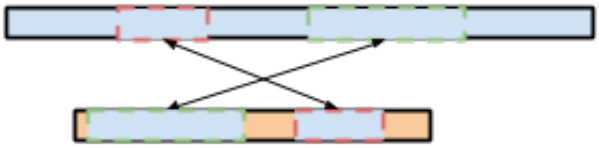


Rysunek 1: Schemat dopasowania globalnego

Lokalny typ dopasowania polega na rozszerzeniu możliwości algorytmów pierwszego typu o zdolność do zauważenia podobieństw w małych obaszarach. Dla przykładu pewne sekwencje mogę być zamienione kolejnością. Ten typ rozwiązania znajduje zastosowanie w analizowaniu białek wielodomenowych. Poniżej przedstawiono schemat dopasowania tego typu.

Dla uwidocznienia różnicy w działaniu tych dwóch typów algorytmów posłużmy się przykładem następujących ciągów:

- TGGAACCA



Rysunek 2: Schemat dopasowania lokalnego

- ACCATGGA

Powyższa sekwencja składa się z dwóch czteroliterowych sekwencji umieszczony w różnej kolejności. Poniżej przedstawiono macierze podobieństawa uzyskane przez oba algorytmy wraz ze znalozionymi rozwiązaniami. Proces powstawania macierzy tego typu zostanie opisany w dalszej częście tego dokumentu.

		T	G	G	A	A	C	C	A
		0	0	0	0	0	0	0	0
A	0	0	0	0	0	5	3	1	5
C	0	0	0	0	3	3	10	8	6
C	0	0	0	0	1	1	8	15	13
A	0	0	0	0	5	6	6	13	20
T	0	5	3	1	3	4	4	11	18
G	0	3	10	8	6	4	2	9	16
G	0	1	8	15	13	11	9	7	14
A	0	0	6	13	20	18	16	14	12

```
ACCA
||||
ACCA
```

```
TGGA
||||
TGGA
```

		T	G	G	A	A	C	C	A
		-4	-2	0	10	12	-3	-10	-9
C	-2	-5	-4	-3	-2	7	0	-15	
C	0	-4	-2	-1	-8	1	2	-13	
A	2	-4	-9	1	2	-5	-4	-3	
T	12	-2	-8	-6	-4	-3	-2	-9	
G	-3	7	1	-5	-3	-1	0	-7	
G	-10	0	2	-4	-2	0	2	-5	
A	-17	-15	-13	-3	-1	-7	-5	5	

```
TGGAACC----A
x     ||    |
A----CCATGGA
```

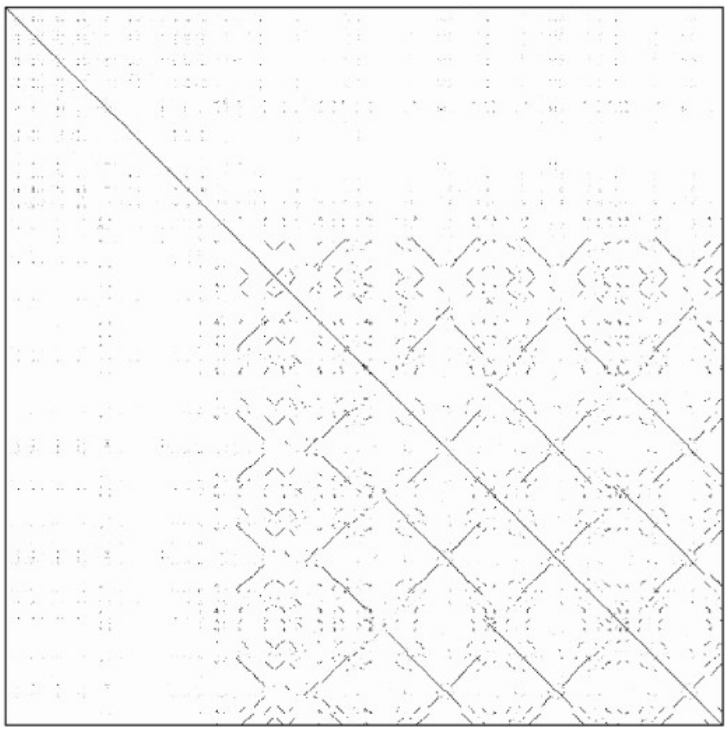
Rysunek 3: Przykład różnicy w działaniu dopasowania globalnego (po lewo) i dopasowania lokalnego (po prawo)

W przypadku globalnego dopasowania najlepszy uzyskany wynik jest jeden. Algorytm uzaje, że za najpesze dopasowanie należy uznać następującą interpretacje: Dla uwidocznienia różnicy w działaniu tych dwóch typów algorytmów posłużmy się przykładem następujących ciągów:

- Pierwszy znak został podmieniony
- Natępnie brakuje 4 znaków w drugim ciągu

- Kolejne dwa znaki pasują do siebie
- Natępnie brakuje 4 znaków w pierwszym ciągu
- Ostatnie znaki pasują do siebie

Jak widać, takie rozwiązanie nie jest w stanie wykryć istoty zadanego przykładu. Dla odmiany dopasowanie lokalne nie narzuca jednego najlepszego rozwiązania. Po zbudowaniu macierzy podobieństwa możemy zauważyć że istnieją dwie ścieżki punktowane w ten sam sposób. Jedna z nich repreztuje informacje o znalezieniu dopasowania podciągów ĄCCA", druga o znalezieniu dopasowania podciągów "TGGA". W przypadku dużych ciągów wejściowych powyższe macierze reprezentuje się w odmienny sposób. Przyjmując pewną wartość progową można utworzyć wykres tego typu:



Rysunek 4: Przykład wizualnej reprezentacji macierzy podobieństwa dla algorytmu dopasowania lokalnego

Dzięki takiemu przedstawieniu wyników możliwe jest zwrócenie uwagi na fragmenty zawierające istotne podobieństwo.

Algorytm Smitha-Watermana w ujęciu sekwencyjnym

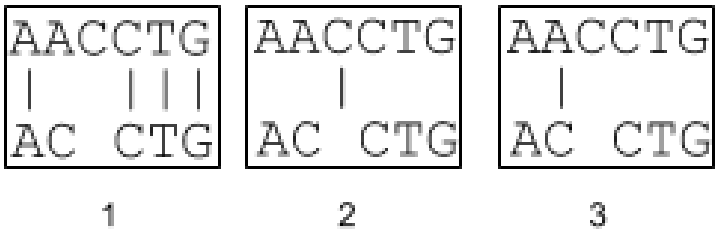
Algorym Smitha-Watermana należy do klasy algorytmów dynamicznych. Składa się z się z dwóch etapów:

- Tworzenie macierzy podobieństwa
- Otwarzanie optymalnej ścieżki (ang. backtracking)

W pierwszym etapie zostaje utworzona pusta macierz. Jej wiersze odpowiadają kolejnym znakom pierwszego ciągu, kolumny kolejnym znakom drugiego ciągu. Komórki

znajdujące się na przecięcie opisują punktacje określającą w jakim stopniu dopasowanie danych dwóch znaków jest poprawne.

W celu wypełnienia powyższej macierzy należy zauważyć, że przy porównywaniu dwóch ciągów można mieć miesce trzy sytuacje przedstawione na poższym schemacie.



Rysunek 5: Możliwe sytacje w trakcie porównywania ciągów: (1 - dopasowanie, 2 - przerwa, 3 - zamiana)

Dwa ciągi są do siebie podobne gdy mamy więcej sytuacji typu 1 (”dopasowanie”) niż sytuacji typów 2 (”przerwa”), 3 (źamiana”). W związku z tym spostrzeżeniem w trakcie wypełniania macierzy wartościami będziemy dodatnio punktować ”dopasowania”, podczas gdy ”przerwy”i źamiany”będę punktowane ujemnie. Dokładne wartości punktacji nie są elementem specyfikacji algorytmu i są dobierane w zależności od rozpatrywanego problemu. Daje to możliwość porównywania ciągów w sposób traktujący ”przerwy”mniej restrykcyjnie niż źamiany”(lub odrotnie).

W celu opisy kroków algorytmy posłużono się przykładem. Schemat przedstawiony na rysunku 6. prezentuje pierwsze kroki wykonywane w celu wypełnienia macierzy podobieństwa. Przedstawiono przypadek dla danych wejściowych: TGGA, TGA oraz dla parametrów:

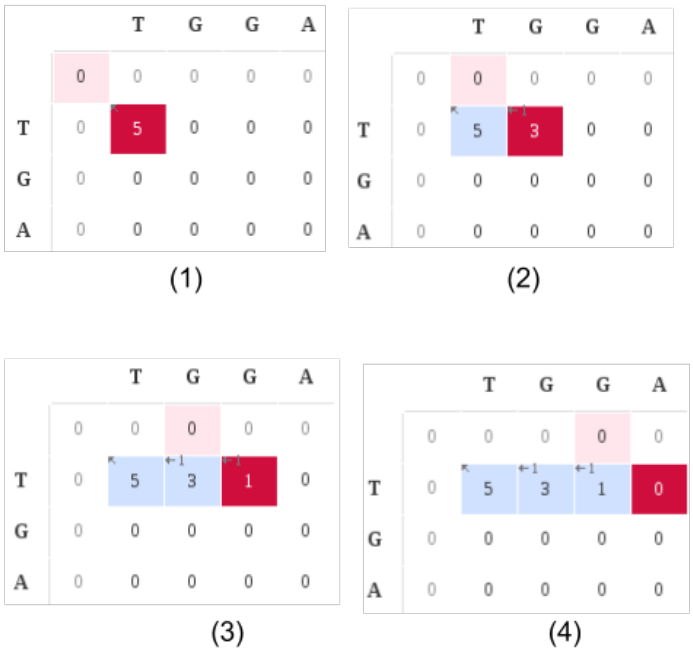
- Punktacja za dopasowanie: 5
- Punktacja za zamiane: -3
- Punktacja za przerwe: -2

Na początku macierz wypełniona jest zerami. Należy zwrócić uwagę na istnienie dodatkowego wiersza i kolumny (oznaczone na schematach szarą czcionką). Komórki rozpatrywane są wiersz po wierszy. Dla każdej komórki wykonywana jest sekwencja kroków mająca odpowiedzieć na następujące pytanie: ”Przez którego z sąsiadów należy poprowadzić ścieżkę dopasowania tak, żeby w obecnej komórce osiągnąć najlepszy wyniki?”. Przy odpowiedzi na to pytanie rozpatrywane są komórki:

- Na lewo - odpowiadająca wprowadzeniu przerwy w pierwszym ciągu
- Powyżej - odpowiadająca wprowadzeniu przerwy w drugim ciągu
- Na skos (powyżej i na lewo) - w zależności od przypadku odpowiadająca wprowadzeniu zamiany lub dopasowania.

Tą procedurę można przedstawić w pseudokodzie w następujący sposób.

```
1   int fromLeft = valueOfCellOnLeft - penaltyForGap;
2   int fromUp = valueOfCellOnUp - penaltyForGap;
3   int fromDiagonal;
```



Rysunek 6: Pierwsze cztery kroki wykonaniu algorytmu Smitha-Watermana dla danych wejściowych: TGGA, TGA

```
4    if(letterInRow == letterInCol){
5        fromDiagonal = valueOfCellInDiagonal + bonusOfMatch;
6    }else{
7        fromDiagonal = valueOfCellInDiagonal + penaltyForReplacement;
8    }
9    int    valueOfThisCell = max(fromLeft, fromUp, fromDiagonal);
10   rememberDecision();
```

Prześledźmy kolejne kroki wykonania przykładu z rysunku 6. Warto przy tej okazji zwrócić uwagę na fakt, że w algorytmie Smitha-Watermana celowo unika się wprowadzania do macierzy ujemnych wartości. Względu na to w poniższym opisie zastosowano znak $\simeq$ wszędzie tam, gdzie zamiast wartości ujemnej podstawiane jest 0.

W kroku 1:

- Wybranie drogi z lewej strony dawałoby: 0 + (-2) $\simeq$ 0
- Wybranie drogi z góry dawałoby : 0 + (-2) $\simeq$ 0
- Ze względu na to, że litery w kolumnach (T) i rzędach (T) są takie same przejście na skos dawałoby: 0 + 5 = 5
- Najbardziej opłacalnym ruchem jest przejście na skos więc zapamiętujemy ten ruch i nadajemy komórce wartość 5

W kroku 2:

- Wybranie drogi z lewej strony dawałoby: 5 + (-2) = 3
- Wybranie drogi z góry dawałoby : 0 + (-2) $\simeq$ 0
- Ze względu na to, że litery w kolumnach (G) i rzędach (T) niesą takie same na skos dawałoby: 0 + (-3) $\simeq$ 0

- Najbardziej opłacalnym ruchem jest przejście z lewej strony więc zapamiętujemy ten ruch i nadajemy komórce wartość 3

W kroku 3:

- Wybranie drogi z lewej strony dawałoby: 3 + (-2) = 1
- Wybranie drogi z góry dawałoby : 0 + (-2) $\simeq$ 0
- Ze względu na to, że litery w kolumnach (G) i rzędach (T) niesą takie same na skos dawałoby: 0 + (-3) $\simeq$ 0
- Najbardziej opłacalnym ruchem jest przejście z lewej strony więc zapamiętujemy ten ruch i nadajemy komórce wartość 1

W kroku 4:

- Wybranie drogi z lewej strony dawałoby: 1 + (-2) $\simeq$ 0
- Wybranie drogi z góry dawałoby : 0 + (-2) $\simeq$ 0
- Ze względu na to, że litery w kolumnach (A) i rzędach (T) niesą takie same na skos dawałoby: 0 + (-3) $\simeq$ 0
- W tym momencie wszystkie drogi dają taką samą wartośc nie zapamiętujemy kierunku i nadajemy komórce wartość 0.

Po wykonaniu analogicznych kroków dla wszystkich komórek macierzy otrzymamy następujący stan:

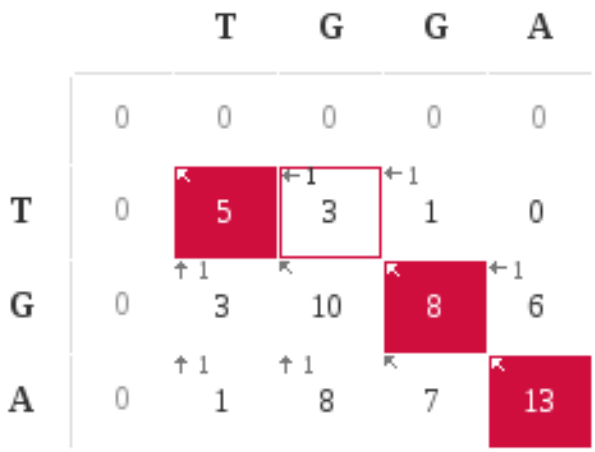
		T	G	G	A
	0	0	0	0	0
T	0	5	3	1	0
G	0	3	10	8	6
A	0	1	8	7	13

Rysunek 7: Całkowicie wypełniona macierz dopasowania algorytmu Smitha-Watermana dla danych wejściowych: TGGA, TGA

W tym momencie macierz jest gotowa do wykonania drugiej fazy - backtrackingu. Polega ona na znalezieniu maksymalnej komórki i zapisaniu wszystkich kroków, które doprowadziły to ustalenie jej wartości. Poniżej przedstawiono macierz z nasiesioną ścieżką metodą backtrackingu.

Po wykonaniu analogicznych kroków dla wszystkich komórek macierzy otrzymamy stan przedstawiony na rysunku 8.

W efekcie uzyskany wyniki to: [$\nwarrow, \leftarrow, \nwarrow, , \nwarrow$] co należy odczytać jako: [dopasowanie, przerwa, dopasowanie, dopasowanie]. Odpowiada do dopasowaniu przedstawionym na rysunku 9.



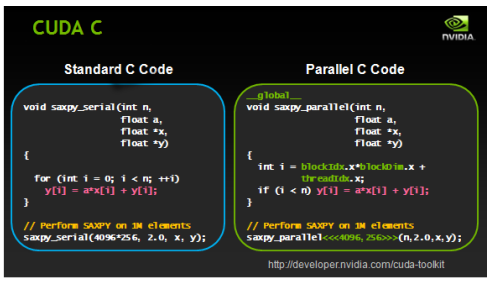
Rysunek 8: Backtracking dla algorytmu Smitha-Watermana dla danych wejściowych: TGGA, TGA

```
TGGA
|  ||
T-GA
```

Rysunek 9: Wynika działania algorytmu Smitha-Watermana dla danych wejściowych: TGGA, TGA

Zarys technologii CUDA

W ostatnich czasach zwiększanie wyników CPU przez zwiększanie częstotliwości zegara jednak przestało to być skutenczne. Zaczęto iść w stronę zwiększanie liczby rdzeni. Co aktualnie w przypadku procesorów jest kontynuowane. Mimio tych starań okazuję się ze karty graficzne są szybsze. Atualnie karty graficzne posiadają setki rdzeni i bardzo szybką pamięć co pozwala przy odpowiednio napisanym algorytmie uzyksanie bardzo dużego przyspieszenia. Widać to na przykład w przypadku łamania haseł, dzięki obliczeniom na karcie graficzne czas metody bruteforce bardzo się skraca. Przyspieszenie to nie jest tak



Rysunek 10: Porównanie progamu sekewncyjnego z równoległy

łatwe do uzyskania jak przykładowo w OpenMP gdzie stosuje się dyrektywy. Aby uzyskać

znaczne przyszpiesznie na kartach graficznych należy odpowiednio przygotować algortym. W przypadku uruchamiania go na GPU kod kernela(pojedynczej funkcji) wykonywany jest przez setki wątków. Ważne jest też używanie szybkiej pamięci karty.

Model zwrónoleglenia Algorytmu Smitha-Watermana

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Cras sed velit nec nibh varius suscipit. Curabitur nisi purus, porttitor in urna ut, tincidunt aliquam leo. Integer sit amet nisi egestas, congue tellus nec, gravida tellus. Nulla facilisi. Quisque ultrices sem sed arcu mattis, in eleifend lectus imperdiet. Donec ullamcorper cursus tortor, in interdum sem imperdiet non. Aliquam sit amet viverra nisl, vitae ullamcorper dolor. Pellentesque varius ex a urna blandit, ac volutpat sapien feugiat. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Nulla posuere ligula ligula, ut ullamcorper nulla tempor vitae.

Vestibulum faucibus ex dolor, non suscipit enim sagittis eget. Cras et condimentum elit. Integer porta, quam ac posuere efficitur, magna mauris finibus nisi, at egestas augue diam sed risus. Ut nec nisi quis nunc sagittis volutpat at viverra eros. Donec ut porttitor orci. Mauris eget eleifend neque, id condimentum tortor. Nunc lobortis quam mi, aliquet varius dolor tempus non. Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Nam sed auctor ex. Praesent tempus ipsum sit amet eros tempus, id ornare nulla vulputate. Nullam lobortis mi leo, ac tincidunt est volutpat dapibus. Pellentesque faucibus maximus suscipit. Duis vitae turpis in nisi condimentum finibus quis a arcu. Donec magna massa, elementum rutrum ligula at, dignissim blandit sem. Vivamus est nisl, aliquam ac tellus ac, egestas consectetur erat. Morbi commodo dui non ipsum vehicula dapibus.

Wnioski

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Cras sed velit nec nibh varius suscipit. Curabitur nisi purus, porttitor in urna ut, tincidunt aliquam leo. Integer sit amet nisi egestas, congue tellus nec, gravida tellus. Nulla facilisi. Quisque ultrices sem sed arcu mattis, in eleifend lectus imperdiet. Donec ullamcorper cursus tortor, in interdum sem imperdiet non. Aliquam sit amet viverra nisl, vitae ullamcorper dolor. Pellentesque varius ex a urna blandit, ac volutpat sapien feugiat. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Nulla posuere ligula ligula, ut ullamcorper nulla tempor vitae.

Vestibulum faucibus ex dolor, non suscipit enim sagittis eget. Cras et condimentum elit. Integer porta, quam ac posuere efficitur, magna mauris finibus nisi, at egestas augue diam sed risus. Ut nec nisi quis nunc sagittis volutpat at viverra eros. Donec ut porttitor orci. Mauris eget eleifend neque, id condimentum tortor. Nunc lobortis quam mi, aliquet varius dolor tempus non. Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Nam sed auctor ex. Praesent tempus ipsum sit amet eros tempus, id ornare nulla vulputate. Nullam lobortis mi leo, ac tincidunt est volutpat dapibus. Pellentesque faucibus maximus suscipit. Duis vitae turpis in nisi condimentum finibus quis a arcu. Donec magna massa, elementum rutrum ligula at, dignissim blandit sem. Vivamus est nisl, aliquam ac tellus ac, egestas consectetur erat. Morbi commodo dui non ipsum vehicula

dapibus.

Źródła

Łukasz Ligowski, Witold Rudnicki - AN EFFICIENT IMPLEMENTATION OF SMITH WATERMAN ALGORITHM ON GPU USING CUDA, FOR MASSIVELY PARALLEL SCANNING OF SEQUENCE DATABASES

E. Banachowicz Bioinformatyka - wykład monograficzny http://opal.przyjaznycms.pl/