一個簡單的反轉排序演算法

A simple algorithm for sorting by reversal

張民欣
慈濟大學醫學資訊系
花蓮市中央路三段 701 號
95325114@stmail.tcu.edu.tw

許弘駿
慈濟大學醫學資訊系
花蓮市中央路三段 701 號
hchsu@mail.tcu.edu.tw

摘要

計算兩組符號排列間反轉距離的問題在近十年引起計算分子生物學對於基因重組問題的重視。給定兩個基因組，為含有相同元素的符號排列，找出一組排列藉由一連串反轉後，轉變成另一組排列所需的最少反轉次數。我們提出新的演算法，不需要複雜的結構，對兩組具有相同元素個數的符號排列做反轉排序。

關鍵詞：計算生物學；基因重組；反轉排序；符號排列。

Abstract

Computing reversal distance of two signed permutations has given rise to computational molecular biology with regard to genome rearrangement in the recent decade. Given two genomes represented as signed permutations of the same elements, the problem consists in finding the minimum number of reversals that transforms one genome into the other. In this paper we present a new algorithm for the problem of sorting signed permutations by reversals without complex construction.

Keyword. Computational biology, Genome rearrangement, Reversal sorting, signed permutations

1、簡介

生物演化過程的建構是以基因重組為基礎，在基因重組的方式中，反轉已證明為最好的建構工具之一。計算兩組符號排列(signed permutations)間反轉距離(reversal distance)的問題在近十年引起計算分子生物學對於基因重組問題的重視。因為兩組符號排列的反轉距離對於不同物種的演化關係及檢查基因序列的相似度是很重要的。生物學家利用計算兩組符號排列的反轉距離來評估基因組的演化距離以建構物種祖先的基因組。

基因序列經過反轉會改變基因組裡基因的順序及方向，我們以整數 1, …, n 來表示基因組裡的基因，以加號或減號來表示基因的方向，基因的順序及方向以符號排列 P=(1, …, n)來表示。兩組符號排列間反轉距離的問題是指一組排列藉由一連串反轉後，轉變成另一組排列所需的最少反轉次數。我們所探討的反轉距離是將一組排列經過數次反轉後成為 identity signed permutation(簡稱 Id)：(+1, +2, …, +n)。

在 1995 年，Hannenhalli-Pevzner(簡稱 HP)首先提出時間複雜度為多項式的演算法來解決兩組符號排列間反轉距離的問題，並提出一個雙重定理計算反轉距離[7]。HP 理論將符號排列擴大為 2n 個符號，並以 breakpoint graph 來表示，此圖是由圓圈(cycle)及連通的元件(connected components)所組成。

HP 理論是複雜的，它將 breakpoint

graph 所形成的結構區分為 hurdles、superhurdles 和 fortresses；HP 所提的多項式演算法，時間複雜度為 $O(n^4)$。

另外，在[6]中定義特定的排列種類，使用其演算法以線性時間計算出反轉距離，他們的方法不須以HP理論為基礎；並應用於計算人類、大鼠和小鼠的X染色體之間的反轉距離。他們將符號排列區分為三種，依各種符號排列所需的時間複雜度由小到大順序為$O(n)$、$O(n\sqrt{n\log(n)})$、$O(2^k \times n\sqrt{n\log(n)})$。

在[2]中提出簡單的演算法計算出反轉距離，其演算法所需的時間複雜度為$O(n)$，它是以一個PQ-tree來表示一個符號排列的各種特徵，並以不同的參數表示hurdles、superhurdles和fortresses，利用這些參數簡化HP所提出的雙重定理。

在這篇文章中我們以 A. Bergeron.等人的理論[2]為基礎提出簡單的演算法對含有 n 個元素的符號排列做反轉排序，我們的演算法不需要複雜的結構，只需要利用 elementary interval 作反轉，它可用於大多數的符號排列，而且生物基因序列所產生的符號排列幾乎都可以利用此結構作反轉排序。

在下一個章節，我們會介紹符號排列、反轉距離和 elementary interval 的定義，以及 elementary interval 的一些特性。第三個章節呈現我們的演算法，提出相關證明，應用 elementary interval 的結構來做反轉排序。在第四個章節以我們的演算法做出結論。

2、基本定義

2.1 signed permutation and reversal distance

符號排列是指一組排列以整數{0, 1, 2, …, n}來表示每個元素，且每個元素都具有符號。假設一個排列含有 n 個元素，我們給定此排列的起點為 0，以 n+1 結尾，例如：P = (0 -3 5 2 -4 1 6)；Id = (0 1 2 …n+1)。點(point)是指排列中一對連續的元素，例如：0 · -3 和 -3 · 5 是 P 的前兩個點。相鄰點(adjacency)意指點是由 i · i+1 或-(i+1) · -i 所構成的；否則就稱為間斷點(breakpoint)，例如 P 裡面的-3 · 5 和 5 · 2。上述例子中的 P 裡面所有的點都是間斷點。

符號排列的區間(interval) $[i, j] \subseteq [1, n]$ $(i < j)$反轉後，以$\rho_{i,j}$=(0 … i-1 -j … -i j+1 … n+1)表示。那麼$P \cdot \rho_{i,j}$表示P在interval [i, j]作反轉，改變[i, j]所有元素的順序及符號：

$$P \cdot \rho_{i,j} = (P_0 \ldots P_{i-1}\ -P_j \ldots -P_i\ P_{j+1} \ldots P_{n+1}).$$

排列 P 的反轉距離，以 d(P)表示，意指將 P 轉變成 Id 所需的最少反轉次數(圖 1)。

0 [2 -5 -4 -3 -1] 7 6 8
0 1 [3 4 5 -2] 7 6 8
0 1 2 [-5 -4 -3 7] 6 8
0 1 2 [-7 3 4 5 6] 8
0 1 2 [-6 -5 -4 -3] 7 8
0 1 2 3 4 5 6 7 8

(圖1)　對P = (0 2 -5 -4 -3 -1 7 6 8)作反轉排序，其d(P) = 5：(2 -5 -4 -3 -1)，(3 4 5 -2)，(-5 -4 -3 7)，(-7 3 4 5 6)，(-6 -5 -4 -3)。

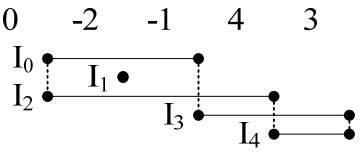
2.2 Elementary intervals

首先將符號排列P以另一種不具符號的排列來表示，我們給定P的每個元素一個左邊點及一個右邊點。每對不具符號的元素(k, k+1)，$0 \le k \le n$用一個elementary interval I_k來表示，interval的端點為：

(1) 若k為正數，則 I_k 的端點為k的右邊點；否則為k的左邊點。

(2) 若k+1為正數，則 I_k 的端點為k+1的左邊點；否則為k的右邊點。

所以元素k和k+1都是elementary interval I_k 的端點(圖2)。



(圖2)　$I_0, I_1, \ldots, I_4$為P = (0 -2 -1 4 3 5)的所有elementary interval。

兩個elementary interval的端點相遇之處為間斷點，如圖2之虛線。elementary interval I_k所包含的端點有以下三種情況：

(1) 同時具有端點k和k+1，如圖2，I_3；

(2) 只包含其中一個端點k或k+1，如圖2，I_0、I_2；

(3) 兩者均不包含，如圖2，I_1、I_4。

此外，若elementary interval的兩個端點為相同時，即是相鄰點，如圖2，I_1。

elementary interval依其端點的符號分為兩種：

(1) Oriented elementary interval：interval的兩個端點符號不同，如圖2，I_0、I_2。

(2) Unoriented elementary interval：interval的兩個端點符號相同，如圖2，I_3、I_4。

在反轉排序的問題中，Oriented elementary interval是比較重要的，因為它可以產生相鄰點。

Lemma 1.[2] 反轉一個Oriented elementary interval會使得符號排列出現相鄰點 k · k+1 或 -(k+1) · -k。

如圖2，I_0反轉：

0 -2 -1 4 3 5

0 1 2 4 3 5

產生相鄰點0 · 1。

將一組符號排列以elementary interval表示時，可將其結構作劃分，以圓圈(cycle)的形式呈現。圓圈意指一連串的點$b_1, b_2, \ldots, b_k$所形成，這些點就是elementary intervals的端點[2]，如圖2是由兩個圓圈所構成，分別是：I_0、I_2、I_3、 I_4的每個端點和I_1的端點。

Lemma 2 [2] 若P的所有elementary interval皆是Oriented，則d(P)=n-c，其中c是指cycle個數。

Lemma 3.[2] 反轉一個Unoriented elementary interval會產生Oriented elementary interval，且符號排列的cycle個數不變。

Theorem 1.[2] 一組排列P由{0, …, n}所組成，有c個cycles，其Tp tree有最小的cost t，則反轉距離：

$$d(P) = n-c+t.$$

因此，cycle個數會影響reversal distance，有關Tp的說明請參考[2]。

3、演算法

在此章節，呈現完整的演算法，以符號排列的 elementary intervals 結構來運作。由於每個符號排列 elementary intervals 的組成不盡相同，有些不含任何的

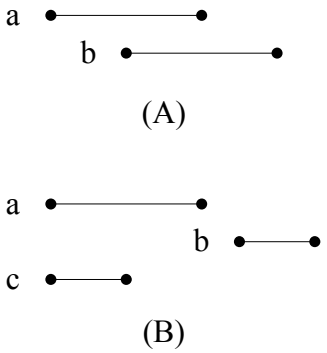
Oriented elementary intervals；因為 Id 只有相鄰點，所以我們利用 Oriented elementary intervals 的反轉來產生相鄰點。我們的演算法中將相鄰點視為一個點而非一個 interval。

排序演算法：

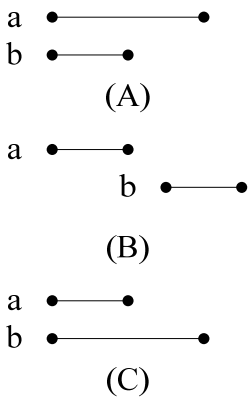
輸入：一組 n 個元素的符號排列 P。
輸出：P 的反轉過程；最後以 Id 結束。

```
 1: S ← Interval(P)
 2: R ← ∅
 3: while( S≠∅)
 4:     OS ← Oriented(S)
 5:     if( OS==∅) then
 6:         I ← Max_Overlap(S)
 7:     else
 8:         for each OI[i] in OS
 9:             OR[i] ← Reversal( P, OI[i] )
10:         I ← Safe(S, OS, OR)
11:     R ← R+I
12:     P ← Reversal(P, I)
13:     S ← Interval(P)
14: end while
15: Output(R)
```

在演算法第一行，先求出排列 P 的所有 elementary intervals 儲存於 S，我們以傳統的排序方式先將 P 的內容按照無符號(unsigned)方式排列由小到大排序，因為 P 的元素為連續的整數，所以可以利用 Hash table 的方式找出所有的 elementary intervals，因此時間複雜度為 O(n)。Interval 最多有 n+1 個，即 $I_0, I_1, \ldots, I_n$。在第二行中，給定一個集合 R 用來紀錄反轉過的 intervals。第三行到最後一行，由於最後的排列為 Id，Id 不含任何的 interval，所以作為 while 迴圈的終止條件；由 Lemma 1 及 Lemma 3 可知此迴圈最多執行 2n 次；在第四行找出排列 P 的所有 Oriented elementary intervals 以利我們做反轉之用。如果排列 P 不含任何 Oriented elementary intervals，則我們任意挑選一個 Unoriented elementary interval 與其他的 interval 有 overlap，且 overlap 的個數最多，將之做反轉，以產生 Oriented elementary intervals 來增加相鄰點；重疊(overlap)意指一個 interval 與其他 interval 有部分區域相同，否則就稱為 commute(圖 3)；commute 有三種情況(圖 4)(假設有兩個 interval a、b)：



(圖 3)　(A)a 與 b 是 overlap；(B)a 與 b、c 是 commute。



(圖 4)　a 與 b 是 commute，則：(A)a 包含 b；(B)a 與 b 不相交；(C) a 包含於 b。

由 Lemma 3 可知，反轉任一個 Unoriented elementary interval，可得到新的 Oriented elementary intervals，且不會再額外增加 reversal distance (因 cycle 個數不變)。

第八行到第十行採用暴力法找出最好的 Oriented elementary intervals，如果排列 P 含有 Oriented elementary intervals 則我們

以 Link List 為資料結構，每個 Node 代表 P 的一個元素，指標則代表每個元素的符號，將每個 interval 都做反轉，將反轉後的排列儲存，因此反轉的時間複雜度為 O(1)；第十行判斷每個 interval 反轉後的排列，哪一個是最好的反轉，所謂最好的反轉為：

(1) 反轉後的每個排列中所含的 interval 個數為最少，且含有 Oriented elementary intervals；
(2) 反轉後的每個排列中沒有 Oriented elementary intervals，則選擇 interval 個數最少的。

根據上面兩種情況挑選出最好的反轉。

我們的演算法所需的時間複雜度為 $O(n^3)$。

Safe(S, OS, OR):

```
Min1 ← Min2 ← |S|
X1 ← X2 ← -1
for i ← 0 to |OS|
   M ← Interval(OR[i])
   T ← Orinted(M)
   if( Min1 >= |M| and T≠∅)
      Min1 ← |M|
      X1 ← i
   else if( Min2 >= |M| )
      Min2 ← |M|
      X2 ← i
if (X1 == -1)
   I ← OR[X2]
else
   I ← OR[X1]
return I
```

4、結論

生物演化是以基因重組為基礎，其中又以反轉為演化過程最好的建構工具。在這裡我們提出新的演算法，描述如何利用 elementary intervals 來做反轉排序，它不需要複雜的結構。在未來仍有一些問題需要考慮，演算法時間複雜度之改善，另外，由於生物的基因序列所產生的符號排列，少部分無法利用 elementary intervals 做反轉排序，此類符號排列所形成的 breakpoint graph 為 fortresses 的結構[7]，因此希望針對此類排列做進一步的探討。

參考文獻

[1] A. Bergeron, C. Chauve, F. de Montgolfier, and M. Raffinot, “Computing Common Intervals of K Permutations, with Applications to Modular Decomposition of Graphs,” *Proc. 13th Ann. European Symp. Algorithms (ESA '05),* pp. 779-790, 2005.

[2] A. Bergeron, J. Mixtacki, and J. Stoye, “Reversal distance without hurdles and fortresses.”, *In CPM 2004 Proceedings, volume 3109 of LNCS*, pp. 388-399, 2004.

[3] Anne Bergeron, “A very elementary presentation of the Hannenhalli–Pevzner theory”, *Discrete Applied Mathematics*, Vol. 146, pp. 134-145, 2005.

[4] Eric Tannier, Anne Bergeron and Marie-France Sagot, ”Advances on sorting by reversals”, *Discrete Applied Mathematics*, Vol. 155, pp. 881-888, 2007.

[5] Max A. Alekseyev and Pavel A. Pevzner, “Colored de Bruijn Graphs and the Genome Halving Problem”, *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, Vol. 4, No. 1, pp. 98-107, 2007.

[6] Se` verine Be´ rard, Anne Bergeron, Cedric Chauve, and Christophe Paul, “Perfect Sorting by Reversals Is Not Always Difficult”, *IEEE/ACM Transactions on*

Computational Biology and Bioinformatics, Vol. 4, No. 1, pp. 228-238, 2007.
[7] Sridhar Hannenhalli and Pavel A. Pevzner, “Transforming Cabbage into Turnip: Polynomial Algorithm for Sorting Signed Permutations by Reversals”, *Proc. of the twenty-seventh annual ACM symposium on Theory of computing*, Vol. 46, No. 1, pp. 178-189, 1999.
[8] Yoan Diekmann, Marie-France Sagot, and Eric Tannier, “Evolution under Reversals: Parsimony and Conservation of Common Intervals”, *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, Vol. 4, No. 2, pp. 301-309 2007.