與DNA 有關的演算法問題

李家同

我們人類是生物, 其他動物和植物也都是生物, 生物非生物最大的不同, 不僅在於生物有生命, 非生物沒有生命, 最重要的是: 生物會產生下一代, 而非生物是無法產生下一代的。

要產生下一代, 就必須將遺傳的訊息傳給下一代, 如何傳遞遺傳訊息呢? 生物傳遞遺傳訊息的方法是經由 DNA。

DNA由四種鹼基組成, 這四種鹼基就是 adenine (A), guanine (G), cytosine(C) 和 thymine(T)。以人類爲例, 我們的 DNA 是一個長達30億長的 A, G, C, T 序列, 舉例來說, 以下的序列就可能是 DNA 的一部份:

CATGATGGTTAT

DNA存在於生物的細胞裡面, 每一個細胞裡的 DNA 都一樣, 但細胞不僅組織不一樣, 功能也不一樣, 爲什麼同樣的 DNA 會造成不一樣的細胞呢? 這是因爲不同的細胞會產生不同的蛋白質, 舉例來說, 紅血球細胞就要產生一種可以帶氧氣的蛋白質。

細胞怎麼知道如何製造他們需要的蛋白質呢? 他們是根據 DNA 中間的一段序列來製造的, 這一段序列就是某個基因, 人體中總有十萬個不同的蛋白質, 而也就有十萬個基因。

蛋白質由20個氨基酸所合成, 每一個氨基酸對應一個 codon, 每一個 codon 由 A, T, G, C 中的三個字母所構成, 舉例來說, TTT 代表 Phenylalanine, CGT 代表 Arginine, 最精彩的是 TAA, TAG 和 TGA都代表 “end of program”。

這篇文章的主要目的在於介紹和 DNA 有關的幾個演算法問題, 我們不可能詳細地介紹如何設計這些演算法, 因爲這太複雜了, 讀者也必須有相當好的演算法知識, 才能懂其中的細節, 但讀者看了這篇文章以後, 至少會發現, 演算法的應用實在是無遠弗屆的。

序列對齊問題 (Sequence Alignment Problem): 因爲 DNA 是序列, 蛋白質也是序列, 有時我們要比較兩個序列, 就只有將他們對齊一下, 請看以下的例子:

$$S_1 = \text{abbcad}$$

$$S_2 = \text{aecb}$$

我們至少有以下的三種將兩個序列排列的方法:

ab - bcad	abbcad	abbcad
aecb - - -	ae - cb -	aecb - -
(a)	(b)	(c)

對齊 (b) 要比對齊 (a) 及對齊 (c) 要好, 對齊 (b) 有四個不配的地方, 他們是:

b b a d

e - b -

對齊 (a) 有五個不當的配對, 他們是:

b - c a d

e c - - -

對齊 (c) 也有五個不當的配對。

習慣上, 我們有一種計分的制度:

(1) 如果 a_i 和 b_j 對齊, 而 $\mathrm{a}_i = \mathrm{b}_j$, 加二分。
(2) 如果 a_i 和 b_j 對齊, 而 $\mathrm{a}_i \neq \mathrm{b}_j$, 扣一分。
(3) 如果 a_i 和 b_j 對齊, 而 a_i 和 b_j 是一個 blank, 扣一分。

所謂序列對齊問題, 可以正式定義如下: 我們有兩個序列: $S_1 = \mathrm{a}_1\mathrm{a}_2\cdots\mathrm{a}_n$ 和 $S_2 = \mathrm{b}_1\mathrm{b}_2\cdots\mathrm{b}_n$, 找一個分數最高的對齊。

如果我們沒有演算法, 是無法解決這個問題, 因爲我們不能用窮舉法, 一旦用了窮舉法, 我們的演算法就需要 exponential 的時間了。

可是我們可以用 ① dynamic programming 的方法來解這個問題, dynamic programming 的另一個名稱是 ② tabular method, 我無法在此詳細解釋 dynamic programming, 但是我可以很簡單地描述一下如何用 dynamic programming 來解這個問題:

令 $A(i,j)$ 爲 $\mathrm{a}_1\mathrm{a}_2\cdots\mathrm{a}_i$ 及 $\mathrm{b}_1\mathrm{b}_2\cdots\mathrm{b}_j$ 之間的最佳對齊的分數, 則 $A(i,j)$ 可以用以下的公式求得:

$$A(o,o) = o$$
$$A(i,o) = -i$$
$$A(o,j) = -j$$
$$A(i,j) = \begin{cases} \max\left\{\begin{array}{l} A(i-1,j-1)-1 \\ A(i-1,j)-1 \\ A(i,j-1)-1 \end{array}\right\} \\ \qquad\qquad \text{if } \mathrm{a}_i \neq \mathrm{b}_j \\ A(i-1,j-1)+2 \\ \qquad\qquad \text{if } \mathrm{a}_i = \mathrm{b}_j \end{cases}$$

一旦有了這個公式, $A(m,n)$ 就可以很快地求得了。下圖就是用 dynamic programming 求 abbcad 和 eacb 最佳化對齊的經過。

i \ j	0	1 a	2 b	3 c	4 c	5 a	6 d
0	0	−1	−2	−3	−4	−5	−6
1 e	−1	−1	−2	−3	−4	−5	−6
2 a	−2	1	0	−1	−2	−2	−3
3 c	−3	0	0	0	1	0	−1
4 b	−4	−1	2	2	1	0	−1

我們有的時候還要解決更複雜的對齊問題, 比方說, 我們有時會不鼓勵太多的 gap, 我們寧願有一個較長的 gap, 也不要太多小的 gap, 這個考慮 "gap penalty"的序列對齊問題, 也可以用 dynamic programming 來解。

我們要做序列對齊, 主要的目在於決定序列之間的相似形, 經過對齊以後, 如果對齊的分數高, 表示序列之間相似, 對齊的分數底, 表示序列之間不相似。

以上所講的全是兩個序列的對齊問題, 大多數時間, 我們所要比對的序列數目都會超過2, 以下的就是一個例子:

$S_1 =$ A T G C T C
$S_2 =$ A G A G C
$S_3 =$ T T C T G
$S_4 =$ A T T G C A T G C

一個還不錯的對齊如下:

$S_1 =$ A T - G C - T - C
$S_2 =$ A - - G A - G - C
$S_3 =$ - T - T C - T - G
$S_4 =$ A T T G C A T G C

一個多元對齊問題解答的好壞, 可以從每一對序列對齊後所得到的分數來看, 以我們的四個序列爲例, 我們會有六對序列: (S_1, S_2), (S_1, S_3), (S_1, S_4), (S_2, S_3), (S_2, S_4), (S_3, S_4), 每一對, 現在都有一 個配對, 也都有一個分數, 以 S_1 和 S_4 爲例, 現在有6個正確的配對, 3個不正確的配對。而 S_3 和 S_4 只有3個正確的配對, 6個不正確的配對。

多元序列對齊問題, 當然希望各對序列對齊得分的總和要最大, 可惜這是一個 NP-complete 的問題。NP-complete 是一個演算法上的名詞, 如果我們說某某問題是 NP-complete, 就是說這是一個非常難的題目, 極不可能有任何演算法, 可以在 polynomial 的時間內解掉這個問題。如果一個 NP-complete 的問題能夠有 polynomial 時間的演算法, 那麼所有的問題都會有 polynomial 時間的演算法, 這顯然是不太可能的事。

因爲多元序列對齊問題是 NP-complete 的問題, 我們就只好尋找所謂 "近似解演算法" (approximation algorithm), 近似解當然不保證可以得到最佳化的答案, 但是近似解答案的誤解不能太大。在下面, 我大略地介紹一個近似解解法。

以四個序列

$S_1 =$ A T G C T C
$S_2 =$ A G A G C
$S_3 =$ T T C T G
$S_4 =$ A T T G C A T G C

爲例, 我們首先用一種方法決定 S_1 是四個序列中最和別的序列相似者, 要決定這個並不難, 只要 polynomial 時間內即可完成, 一旦決定了這一點, 我們的下一步就是每一個序列都和 S_1 對齊, 對齊的時候, 只顧到 S_1 而不顧到其他的序列, 其結果就是一個近似解。

我們首先對齊 S_1 和 S_2, 其結果如下:

$S_1 =$ A T G C T C
$S_2 =$ A - G A G C

我們再對齊 S_1 和 S_3

$S_1 =$ A T G C T C
$S_3 =$ - T T C T G

換言之, S_1, S_2 和 S_3 的對齊成了以下的情形:

$S_1 =$ A T G C T C
$S_2 =$ A - G A G C
$S_3 =$ - T T C T G

最後, 我們對齊 S_1 和 S_4

$$S_1 = \text{A T - G C - T - C}$$
$$S_4 = \text{A T T G C A T G C}$$

四個序列的最後對齊如下, 如示:

$$S_1 = \text{A T - G C - T - C}$$
$$S_2 = \text{A - - G A - G - C}$$
$$S_3 = \text{- T - T C - T - G}$$
$$S_4 = \text{A T T G C A T G C}$$

以上的對齊不一定是最佳化的, 因爲我們忽略了 S_2 和 S_3, S_2 和 S_4 等等序列的對齊。令這個近似解的得分爲 App, 令最佳對齊的得分爲 Opt, 我們可以證明

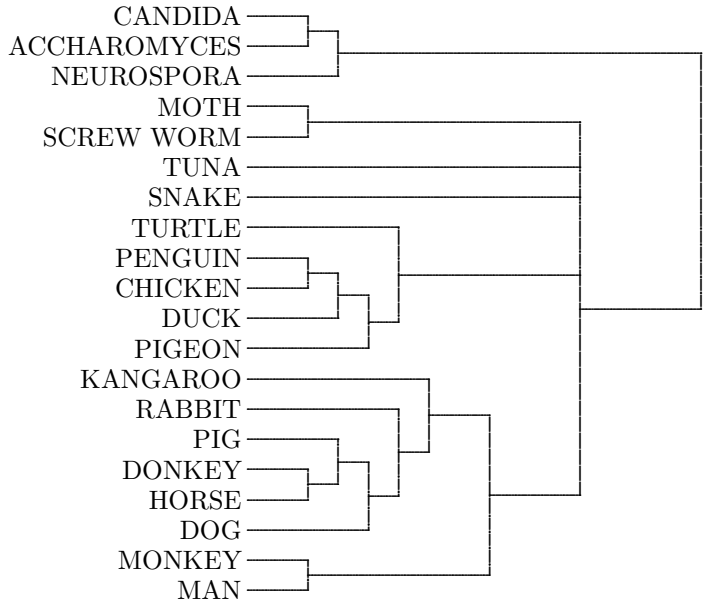
$$\text{App} \leq 2\text{Opt}$$

以上的近似解已經不錯了, 但是有很多新的研究方向更值得介紹。我在以下要簡單地介紹一種叫做“PTAS (Polynomial Time Approximation Scheme)” 的研究領域。

所謂 PTAS, 乃是指我們可以指定某一種誤差, 對應這個指定的誤差, 我們有一個近似解演算法, 這個演算法一定可以產生一個在指定誤差內的近似解, 不僅如此, 這個演算法所需要的執行時間仍是 polynomial 的。設計這種 PTAS 是我們做演算法研究者的夢想, 只有少數的問題有這種 PTAS, 而最近有一個問題, 叫做 minimum routing cost spanning tree problem, 這個問題也可以用來解多元序列的對齊問題。

以下我們將介紹另一個問題, 就是演化樹的問題, 假設我們有幾個物種, 物種和物種之間有一個距離, 我們的任務是建造一個演化樹, 演化樹必須能夠適宜地反應出物種和物種之間的距離。

以下是20個物種的一個演化樹, 從這個演化樹, 我們可以看出物種演化的過程, 這一個演化樹是根據生物細胞內的 C 色素氨基酸序列所建造的。


CANDIDA
ACCHAROMYCES
NEUROSPORA
MOTH
SCREW WORM
TUNA
SNAKE
TURTLE
PENGUIN
CHICKEN
DUCK
PIGEON
KANGAROO
RABBIT
PIG
DONKEY
HORSE
DOG
MONKEY
MAN

演化數的規定很複雜, 我在此只舉幾個特別的規定:

(1) 物種只能出現在演化樹的 leaf nodes 上。
(2) 演化樹有兩種: rooted 和 unrooted。
(3) 如果演化樹是 rooted, 則從 root 到 leaf nodes 的距離都完全相同。
(4) 物種 x 和物種 y 在演化樹上的距離必須大於等於 x 和 y 在 distance matrix 上的距離。

在以下 $dt(x,y)$ 是 x 和 y 在演化樹上的距離, $d(xy)$ 是 x 和 y 在 distance matrix 裡的距離。

到目前爲止, 我們有三種演化樹的規格:

(1) Minimax 演化樹
在 minimax 演化樹, $(dt(x,y) - d(x,y))$ 的最大者要最小。
(2) Minisum 演化樹
在 Minisum 演化樹, 每一對物種在樹上的距離總和要最小。
(3) Minisize 演化樹
在 Minisize 演化樹, 樹上的距離和要最小。

以下的表顯示很多演化樹的問題都不容易得到答案的:

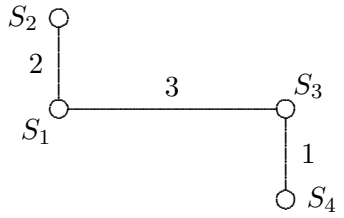
	Minimax	Minisum	Minisize
unrooted	NP-complete	NP-complete	Unknown
rooted	$O(n^2)$	NP-complete	NP-complete

在下面, 我們要介紹一個爲 rooted 演化樹所設計的演算法, 我們用一個例子來解釋。

我們的 distance matrix 如下:

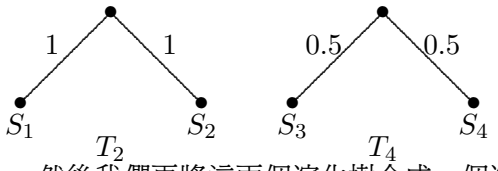
	S_1	S_2	S_3	S_4
S_1	0	2	3	3.1
S_2		0	3.6	5
S_3			0	1
S_4				0

我們首先看到 distance matrix 中的最大距離是 S_2 和 S_4 之間的距離。然後我們再根據 distance matrix 建造一個 minimal spanning tree。以下就是這個 minimal spanning tree。

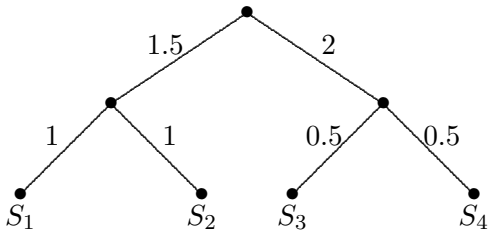


我們在這個 minimal spanning tree 裡找一個從 S_2 到 S_4 的路徑, 這一條路徑就是 $S_2 - S_1 - S_3 - S_4$。在這條路徑內, 最長的邊 (edge) 是 (S_1, S_3), 我們將這個 edge 割斷, 就可以得到兩個 set: $\{S_1, S_2\}$ 和 $\{S_3, S_4\}$。

我們分別替, $\{S_1, S_2\}$ 及, $\{S_3, S_4\}$ 做兩個演化樹, 而且完全保持 S_1 和 S_2 及 S_3 和 S_4 之間的距離, 這兩演化樹如下:



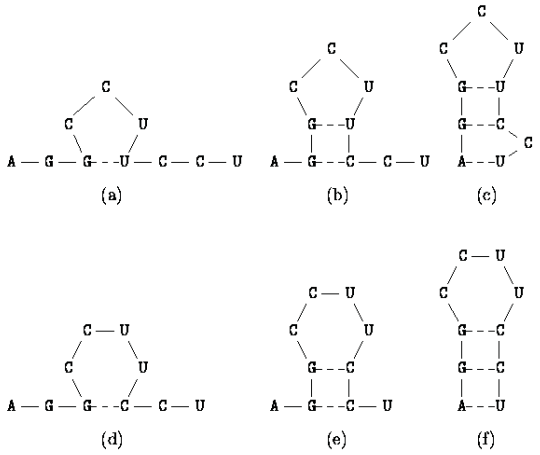
然後我們再將這兩個演化樹合成一個演化樹, 而使 S_2 和 S_4 之間的距離得以完全的保存。這個演化樹如下圖:



因爲很多演化樹問題都是 NP-complete 的, 很多演化樹的問題都必需求救於近似解, 在這方面, 可做的研究實在很多。

在以下, 我們將要介紹一個所謂 RNA 和蛋白質的第二級結構, 我們以 RNA 爲例, RNA 和 DNA 不同, DNA 有雙螺旋的結構, RNA 只有一個線列, 我們可以說一個 RNA 是一連串 A, G, C, U 的序列, 但是 A, G, C, U 中有三種可以配對的, 那就是 A 可以和 U 配對, G 和 C 可以配對, G 也可以和 U 配對。

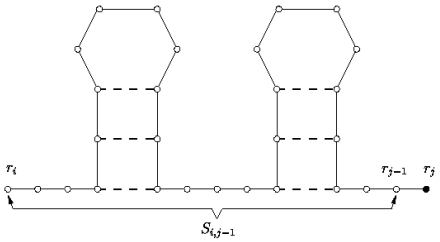
以 A − G − G − C − C − U − U − C − C − U 爲例, 下圖是這一個序 列結構的可能性。



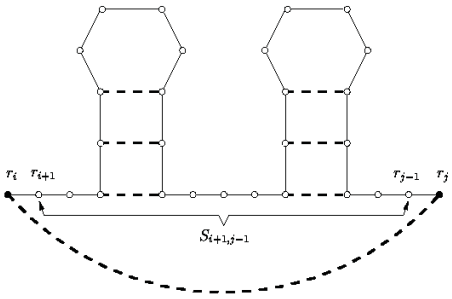
所謂 RNA 第二級結構問題, 就是根據一個 RNA 的序列, 找一個配對最多的第二級結構。這一個問題也可以用 dynamic programming 來解。

考慮 $S_{i,j} = r_i r_{i+1} \cdots r_j$ 這一個序列, 令此序列的最佳化結構有 $M_{i,j}$ 的配對。

Case 1: 在最佳化的答案中, r_j 不和任何其他的 base 相配。在此情形下, 我們應該爲 $r_i \cdots r_{j-1}$ 找一個最佳化的結構。$M_{i,j} = M_{i,j-1}$

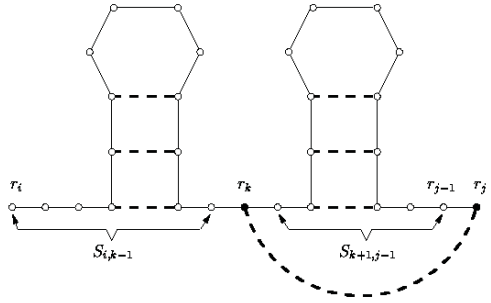


Case 2: 在最佳化情況下, r_j 和 r_i 相配, 在此情形下, 我們應該爲 $r_{i+1} \cdots r_{j-1}$ 找一個最佳化結構, 而 $M_{i,j} = M_{i+1,j-1}$



Case 3: 在最佳化結構下, r_j 和某 r_k 相配。在此情形下, 我們應替 $r_i \cdots r_{k-1}$ 及 $r_{k+1} \cdots r_{j-1}$ 找最佳化的結構, 而

$$M_{i,j} = \max_{i+1 \le k \le j-4} \{1 + M_{i,k-1} + M_{k+1,j-1}\}$$



由以上的法則看來, $M_{i,j}$ 可以很有條理地一步一步得到。

我們應該瞭解, DNA 和演算法有密切的關係, 越來越多的 DNA 問題, 都可以用熟知的演算法來解決, 作者本人在作者的網站上 (http://www.csie.ncnu.edu.tw/~rctlee/Biology/index.html) 有這一方面的講義, 對此有興趣的讀者不妨上網去下載這份講義。

—本文作者任教於暨南國際大學資訊工程系—