有效率的找尋 DNA 序列中的所有串聯重複序列

羅有隆　　絲一倫
朝陽科技大學資訊管理系
副教授　　研究生
yllo@cyut.edu.tw　　s9414625@cyut.edu.tw

摘要

在 DNA 的序列中，串聯重複序列(tandem array)指的是，一段連續核甘酸序列(nucleotide sequence)連續重複兩次或以上且是相連的連續出現。而透過分析串聯重覆序列可幫助我們診斷單一基因疾病、鑑定親子關係等等於醫學上的應用。雖然串聯重複序列在生物學上的研究或與學上的應用已經很多年，但如何有效率的找尋出一 DNA 序列中所有的串聯重複序列，已被發表的相關研究並不多。由於 DNA 序列可以由核甘酸鹼基的 A、T、G、C 四個符號來表示，如此，透過字串比對，可以發展找出一 DNA 序列中串聯重複序列的發生，這也是現有技術所採行的辦法。本研究報告即在提出一新的演算法，利用重複片段的串接，以幫助找尋 DNA 序列中所有的串聯重複序列，並透過實驗證明所提出的方法比現有的技術更為有效率。

關鍵詞：DNA 序列、重覆片段、串聯重覆序列、銜接重覆

Abstract

A tandem array is two or more contiguous copies of a pattern of nucleotides in a DNA sequence. Tandem arrays can be applied in disease diagnosis, human identity testing, and so forth. Although, the applications of tandem arrays have been widely used by biology and medical sciences many years, the proposed schemes for efficiently discovering all the tandem arrays in a DNA sequence are rare. A DNA sequence can be represented by four standard nucleotide bases -- A、T、G and C, such that, for existing approaches, the string matching techniques can be used to discover all the tandem arrays in a DNA sequence. In this paper, we will propose a new algorithm by concatenating repeating patterns for finding all the tandem arrays from a DNA sequence. Our experimental results also show that our approach is more efficiently than existing schemes.

Keywords：DNA sequence, repeating pattern, tandem repeat, tandem repeat

1. 前言

在生物資訊學中，人類基因體計劃(The Human Genome Projcet)已於 2000 年公佈初稿。因此生物資訊學已經由如何快速準確的定序，轉變為誰能有效的利用這些序列資訊以成為生物、醫學上的領先者。公司、藥廠、對序列解讀有興趣的使用者，只要透過網際網路就可到美國的GenBank，歐洲的EMBL，日本的DDBJ等三大資料庫擷取DNA序列進行研究。研究的議題相當的廣泛，包括序列比對分析、基因全序列定序、基因地圖、蛋白質結構、分子模型建立、演化樹、重覆子序列的找尋等等[6]。至於串聯重覆序列(tandem array)在DNA序列中扮演的角色，在相關報告中指出，DNA是由A、T、G、C四種核甘酸鹼基所組成，隨著不同的排序而決定不同的蛋白質合成[1]。人與人之間雖然有 99.9%以上的序列是相同的，原先占非常大比重的重覆子序列，被科學界視為垃圾序列，但1000個核甘酸就有可能出現DNA變異，為何人不可能長的一模一樣也可由此得到解譯。變異可分成兩種，一種為單一核甘酸變異，另一種為重複序列次數的改變，其中最常看到的為串聯重覆序列次數的改變。而所謂串聯重複序列(tandem array)指的是，一段連續核甘酸序列(nucleotide sequence)連續重複兩次或以上且是相連的連續出現。例如：假設一核甘酸序列為“ATTCGATTCGATTCG”，而其中子序列“ATTCG”連續重複了三次，則“ATTCGATTCGATTCG”為一串聯重複序列。

此外，如果串聯重覆序列的重覆次數剛好為 2 時，亦稱為銜接重覆(tandem repeat)，例如：“ATTCGATTCG”。 透過分析串聯重覆序列可幫助我們診斷單一基因疾病、鑑定親子關係等等。以癌症來說，以往我們常取得細胞切片研究哪些可能是致癌因子，但隨著樣本的不同，許多不確定因素有可能導致實驗的正確性不足。我們可以知道癌細胞的特性有大量重覆複製，使細胞不斷的分裂生長因而失去了原有的功能，最終導致病變[8]。透過上述特性我們可在DNA序列當中，先找出串聯重覆序列，再檢查它們的出現次數，是否在正常範圍內。因此，找尋DNA序列中串聯重覆序列的重要性，遂受到相當的重視[3][4][13][14][18]。

在現有找尋串聯重覆序列的技術中，大多是利用字串比對的方式，來偵測串聯重覆序列的發生，因此皆需耗費相當多的計算時間。而此篇研究報告，將透過利用重複片段(repeating pattern)的串接，提出一更有效率的演算法，以幫助串聯重覆序列的搜尋。本文共有五節，內容概述如下：除第一節的前言外，在第二節中我們將介紹現有找尋串聯重覆序列的技術；接著在第三節詳述我們所要發表新的演算法；之後在第四節，我們設計了實驗以驗證所提出方法的效率。而最後一節則為我們的結論與未來的研究方向。

2. 現有文獻探討

近年來有關找尋串聯重覆序列(tandem array)或銜接重覆(tandem repeat)的代表性研究報告，有Benson於1998年發表了*k*-tuple matches的演算法，他利用了統計準則於或然率模型中來偵測銜接重覆的發生[2]；Stoye與Gusfield於2002年發表了利用Optimized Basic Algorithm在suffix tree的非葉節點的走訪與比對，來找尋DNA中的所有串聯重覆序列[14]；Buchner與Janjarasjitt於2003，提出了periodicity explorer algorithm，利用處理DNA序列週期的轉換，來產生每個核甘酸(nucleotide)位置中最接近的週期序列[3]，但此研究報告是以找尋所有的重覆序列(repeat sequences)為主要目的。接著於2005年，Chen提出了利用Findsquares演算法，先找出銜接重覆，再從它們建構出所有不可延伸的串聯重覆序列[4]。接下來，我們就針對[14]與[4]可找尋所有串聯重覆序列的技術做進一步的說明。

2.1 Optimized Basic Algorithm

Stoye與Gusfield於 2002 年發表了利用simple time- and space-optimal algorithm先找到DNA序列中所有的tandem repeats，再推演出所有的tandem arrays [14]。他們將演算法稱為Optimized Basic Algorithm，可簡要摘錄為如下的 6 個步驟：

步驟 1. 首先將DNA序列建構成implicit suffix tree[7]。例如圖 1的implicit suffix tree是由序列“ACAACAACA”所構成，其中‘#’為字串結束符號。suffix tree常用在建立字串的索引結構[7][15]。

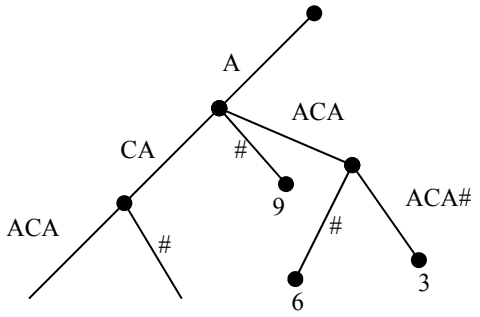

圖1 “ACAACAACA”之 implicit suffix tree

步驟 2. 走訪implicit suffix tree，當經過每個非葉節點時，則將該非葉節點標記為V_1, V_2, V_3, ..., V_n且記錄它們的長度為$D(V_1)$, $D(V_2)$, $D(V_3)$, …, $D(V_n)$。

步驟 3. 走訪每個非葉節點V_i的分支，直至葉節點，並檢查V_i至葉節點之每個距離與$D(V_i)$是否相等。若相等，則表示發現了長度為$2 \cdot D(V_i)$之tandem repeats。

步驟 4. 在現有已找到之 tandem repeats 推演出所有之 tandem repeats。方法為，若一 tandem repeat 發生始於第 j 位置，則比對第 j-1 之符號與此 tandem repeat 之最後一符號是否相同，若相同，則可能又發現一新的 tandem repeat，啟始位置在 j-1。接著再以同樣方法繼續檢驗此新發現的 tandem repeat 與尚未檢驗過之每一 tandem repeat。

步驟 5. 對每一發生兩次或以上之tandem repeat，再做檢查，假設其長度為 $2 \cdot D(V_i)$，將其發生的位置做兩兩相減，如果有差值為$D(V_i)$或$D(V_i)$的倍數，表示有 $3 \cdot D(V_i)$或更長的tandem array被發現。

步驟 6. 刪除重覆找到之 tandem arrays。經過前五個步驟後，已找尋出所有的 tandem arrays，但卻可能有重覆的發生，必須做一過濾與剔除的動作。

Optimized Basic Algorithm 的時間複雜度，也在研究報告中做了討論，為 $O(n\log n)$。

2.2 Findsquares Algorithm

由於tandem repeat亦稱為square[14]，在Chen的報告中[4]，是先利用一Findsquares algorithm找到所有的squares (tandem repeats)，接著再利用squares來建構出所有串聯重覆序列(tandem arrays)。假設tandem repeat的長度為$2p$，作者先定義了一p週期子字串，它是由一段最大長度m的子字串Y所組成，使得對所有的$i\in\{1, \ldots, m-p\}$而言，Y的第i個字元等於第$i+p$個字元($Y_i=Y_{i+p}$)，並且$\left\lfloor \frac{m}{p} \right\rfloor \geq 2$。其演算法可概分為2個步驟，分述如下：

步驟 1. 使用Findsquares algorithm，以遞迴方式切割字串(序列)，再找尋所有tandem repeats。做法是對一字串S，從中間切割成Q與R子字串，再單獨對Q與R切割，成為Q_1、Q_2與R_1、R_2等 4 個子字串，繼續對子字串做切割，直到切割後各子字串的長度等於 1 為止。遞迴程式接著依序從長度為1、2、4、…等子字串中，找尋跨子字串之tandem repeats，再依照p週期子字串定義，由左往右規則性的找尋長度為$2p$橫跨 2 個子字串的tandem repeats。例如，以序列**"**ATATATAT**"**為例，切割到子字串長度為1，則**"**A**"**與**"**T**"**並無法形成一跨 2 個子字串之tandem repeat，但切割至長度為 2 的子字串時，將可形成一跨子字串之tandem repeat。如圖 3，**"**ATAT**"**為跨子字串之 2 週期子字串形成的tandem repeat。繼續如圖 3，於子字串長度為 4 時，依p週期子字串定義，依然可推導出tandem repeats **"**ATAT**"**始於 1、3、5 的位置與**"**TATA**"**始於 2、4、6 的位置。

圖2 長度為 2 之子字串

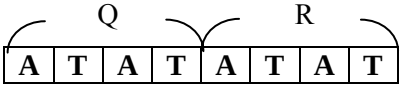

圖3 "ATATATAT"切割為 Q 與 R

步驟 2. 由步驟 1 所找到的 tandem repeats，其結束位址(字串尾)若繼續出現連續的 p 週期子字串，亦即表示它是屬於 tandem array。如此藉由找出所有不可延伸的 p 週期子字串，以決定出所有的 tandem arrays。再以 2 週期子序列**"**ATATATAT**"**來說，tandem repeats **"**ATAT**"**始於 1、3、5 的位置，以及**"**TATA**"**始於 2、4、6 的位置。而最左 tandem repeat **"**ATAT**"**的右邊也可以繼續串接 2 週期子字串**"**AT**"**，以產生 tandem arrays 有長度 $3p$ 的 **"**ATATAT" 與 $4p$ 的**"**ATATATAT**"**。同樣的最左 tandem repeat **"**TATA**"**的右邊也可以繼續串接 2 週期子字串**"**TA**"**，以產生 $3p$ 長度的 tandem array **"**TATATA"。

由於 tandem repeat 也是屬於長度 $2p$ 的 tandem arrays，透過前兩步驟，可找到全部的 tandem arrays。而 Chen 的報告中也提到，此演算法的時間複雜度亦為 $O(n\log n)$。

3. Repeating Pattern Alignment Approach

在本節中，我們詳述所新提出的在字串中找尋所有 tandem arrays 的方法，而我們將之命名為 Repeating Pattern Alignment Approach。

3.1 演算法說明

Repeating Pattern Alignment Approach 是利用重覆片段(repeating pattern)的串接來找尋所有的 tandem arrays。重覆片段的發生，只要在一字串的任何位置重覆出現即可，不需要如同 tandem array 一樣，必須串聯發生。我們新設計的演算法包括有如下 3 個步驟：

步驟 1. Implicit suffix tree construction：如同Optimized Basic Algorithm[14]的步驟 1 一樣，首先將DNA序列建構成implicit suffix tree[7]。

步驟 2. Repeating pattern discovering：從implicit suffix tree中，找出所有的重覆片段，並記錄下重覆片段的長度與發生位置。因在implicit suffix tree中，從根節點至任一非葉節點所經過的字串組成，即形成一重覆片段，重覆的次數可由分支度與子樹的分支度統計而得，相關研究於[7][10][11]。

步驟 3. Alignment and tandem arrays finding：針對每一個重覆片段依發生位置順序做排列，以判斷相鄰重覆片段間可能連續串接或有部份重疊情形，最後找出所有 tandem arrays。假設 repeat pattern 的長度為 p，一重覆片段所發生之兩啟始位置相減的差，可能有如下三種狀況：

*狀況一：*差值大於 p，則此重覆片段所發生的位置並不相連，將之忽略。

*狀況二：*差值等於 p，則此重覆片段所發生的位置剛好相連，將之串接以形成長度為 $2p$ 之 tandem repeat。接著可繼續再判斷下一發生位置，以發現是否可以找到長度為 $3p$ 或更長的 tandem array。

*狀況三：*差值小於 p，表示此重覆片段發生的兩位置有部份重疊。而假設差值為 d，且 $d > p/2$，則發現一長度為 $2d$ 之 tandem repeat。同樣的繼續再判斷下一發生位置，以發現是否可以找到長度為 $3d$ 或更長的 tandem array。但如果 $d \leq p/2$，則將之忽略，因必定有更短的重覆片段以產生 tandem array。

3.2 範例說明

同樣的我們以 DNA 序列"ACAACAACA"為範例，以說明如何用 Repeating Pattern Alignment Approach 來找到所有的 tandem arrays。執行過程如下：

步驟 1. 以序列建構implicit suffix tree其結果與圖 1相同。

步驟 2. 在 implicit suffix tree 中拜訪所有的節點，將所找到的重覆片段、長度與發生位置記錄於表 1 中。

步驟 3. 在表 1 中針對每一個重覆片段發生位置計算差值，將結果記錄於「發生位置差值比對情況」欄位中，再將比對狀況所找到的 tandem arrays 記錄於最後一欄位。如此表 1 中的最後一欄位記載著所有的 tandem arrays。

表 1. 重覆片段處理過程

重覆片段	長度	發生位址	發生位置差值比對情況	Tandem array
ACAACA	6	1, 4	重疊且差值為長度 1/2	忽略
ACA	3	1, 4, 7	差值等於長度且可連續串聯 3 次	ACAACA (1-6, 4-9) ACAACAACA (1-9)
A	1	1, 3, 4, 6, 7, 9	2 次出現差值等於長度	AA(3-4, 6-7)
AACA	4	3 6	重疊且差值大於長度 1/2	AACAAC(3-8)
CAACA	5	2 5	重疊且差值大於長度 1/2	CAACAA(2-7)
CA	2	1,4,7	差值大於長度	忽略

3.3 時間複雜度分析

我們所提出的 Repeating Pattern Alignment Approach，若 DNA 序列的長度為 n，則逐步分析每一步驟所需的時間複雜度如下：

步驟一：implicit suffix tree可於$O(n)$的時間內建構完成[7]。

步驟二：將implicit suffix tree的每個節點拜訪過一次，以找出所有重覆片段，則時間複雜度為$O(n)$[10][11]。

步驟三：最差狀況為某重覆片段發生 n 次，因必須依位置順序做比對，可能有需要先做排序，所需時間為 $O(n\log n)$。而之後的依序比對只需 $O(n)$次的時間。

由以上的分析，Repeating Pattern Alignment Approach 的時間複雜度亦為 $O(n\log n)$。

4. 效能評估

在Optimized Basic Algorithm (OBA)[14]與Findsquares Algorithm[4]所發表的研究報告中，都僅做了時間複雜度的分析，而沒有提供實驗數據做為效能分析。連同我們於此篇報告中所提出的Repeating Pattern Alignment Approach (RPA)，三個演算法的時間複雜度同為$O(n\log n)$，但這並不代表三個方法找尋所有tandem arrays所需的時間成本也會一樣。因此，在本節中，我們將對RPA、OBA、以及Findsquares做一系列的效能評估與比較。在實驗中我們自National Center for Biotechnology Information (NCBI)[19]取得了 500 個DNA序列樣本，做為本實驗的資料庫，並分別讓三個演算法來找尋所有的tandem arrays，以比較個別所需的執行時間。我們考量可能影響執行效能的因素，將分別從下列三個方向來探討：

(1) DNA 序列的長度
(2) Tandem arrays 的個數
(3) 重複片段的個數

以下我們分別將上述三種因素，做為實驗的參數，分析兩演算法的差異。

4.1 DNA 序列的長度

因三個找尋全部tandem arrays演算法的時間複雜度都是一樣的$O(n\log n)$，因此DNA序列的長度n，對每個演算法的執行時間，都是個重要的影響因素。在此節中，我們就以實際的實驗，來探討DNA序列的長度，對三個演算法的影響。於此實驗裡，我們將實驗資料庫依DNA序列樣本的長度來分組，分組方式為長度

小於 200 的分配為 100 這組，長度介於 200 與 300 之間的分配於 200 這組，餘此類推。但長度大於 700 者，全部分配於 700 這組。如此，我們總共分成 7 組，又因資料樣本有按長度特別挑選進入實驗資料庫，以使每組有差不多一致的樣本數。三個演算法分別對 7 組樣本實驗，計算平均找尋一DNA序列之全部tandem arrays所需的時間，實驗結果則呈現於圖 4。由實驗圖中，我們發現三個演算法找尋全部tandem arrays的所需時間，確實受到DNA序列長度的影響，且隨長度增加而需要更長的執行時間。而其中Optimized Basic Algorithm (OBA)受到的影響最為嚴重，隨DNA序列長度的增加，執行時間也急速的增加。在DNA序列長度為 500 時，已超過我們實驗圖所設定的 1 秒鐘執行時間範圍。反觀Findsquares Algorithm與我們所發表的 Repeating Pattern Alignment Approach (RPA)，實驗到了DNA序列長度為 700 時，所需的執行時間仍然維持在 0.4 秒以下，且RPA比較於Findsquares，還可以節省約有 40%的執行時間。由此實驗驗證了，雖然三個演算法的時間複雜度是相同，但執行效率卻有很大的差異，而我們所設計的RPA，則是三者中最為有效率者。

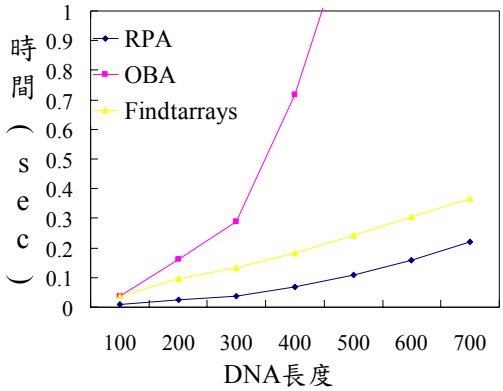

圖4 DNA 序列長度的影響

4.2 Tandem Arrays 的個數

因OBA與Findsquares兩演算法，皆是先找到所有的tandem repeats，再去推衍出所有的tandem arrays。而依tandem repeat與tandem array的定義，tandem repeat是包含於tandem array之中。因此，在這個實驗中，我們將探討tandem array的個數，對三個演算法的影響。於此實驗，我們將實驗樣本重新分組，以每個DNA序列全部的tandem array數目來分組，tandem array數目在 200 以下的分配在 100 這組，個數介於 200 與 300 之間者分配於 200 這組，餘此類推。而因為個數超過 700 者很有限，因此個數超過 600 者，全部分配於 600 這組，共分成 6 組。這樣的分組，各組的樣本數，無法如DNA序列長度分組一樣的分配平均，但每組的樣本數仍介於 59~109 之間，還是有足夠的代表性。我們仍然統計三個演算法分別對 6 組樣本實驗，平均找尋一DNA序列之全部tandem arrays所需的時間。實驗結果如圖 5。於圖 5中，我們發現，DNA序列的全部tandem array個數，對三個演算法的影響，更勝於DNA序列的長度，三曲線以更陡峭的趨勢向上走。OBA在這個實驗中，更快的超出我們時驗圖所設定的時間範圍，表現得較不理想。而RPA與Findsquares的實驗結果，也是隨tandem array個數增加而曲線向上走，但都比OBA來得平緩。在tandem array個數為 600 這組時，RPA仍然可比Findsquare節省約 35%的執行時間。此實驗再次的驗證了，我們所提出的RPA演算法優於現有的OBA與Findsquares algorithm。

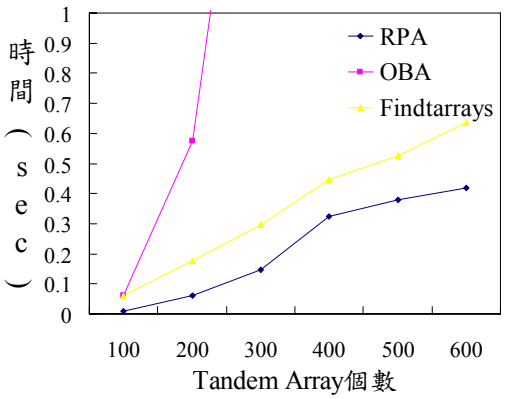

圖5 tandem array 個數的影響

4.3 重覆片段出現的個數

我們所設計的Repeating Pattern Alignment Approach(RPA)演算法，是透過重覆片段(repeating pattern)的比對，以找到所有的tandem arrays。重覆片段的出現，不需要如同tandem array一樣，必須串聯發生，也就是tandem array必須包含於重覆片段中，但重覆片段不一定就是tandem array。因此，對於一DNA序列，重覆片段的個數，往往會比tandem array的個數還多。重覆片段個數的多寡，很可能對RPA有較大的影響。此節的實驗，即在探討重覆片段的個數，對三個演算法執行效率的影響。我們先將資料庫中每個DNA序列樣本做重覆片段的統計，再將重覆片段個數 200 以下的樣本分配於 100 這組中，個數介於 200 與 300 之間者分配於 200 這組，餘此類推。而重覆片段個數超

過 700 者，全分配於 700 這組。這樣的分組，讓每組的樣本數差異更大些，介於 39~124 之間，且個數多的發生在 100 與 200 這兩組。實驗結果表現於圖 6。此實驗結果與前兩個實驗結果類似，仍然是RPA與Findsquares遠優於OBA，而且還是RPA表現得最好。然而，此實驗的結果，在重覆片段個數 100 這組，RPA相對於OBA與Findsquares都有 60%以上的改善率。但是在 700 這組，RPA只比Findsquares約有 23%的改善率。此結果證實了我們的預期，重覆片段的多寡，對RPA會有較大的影響。不過總結來說，我們的RPA演算法，仍然比OBA與Findsquares表現優異。

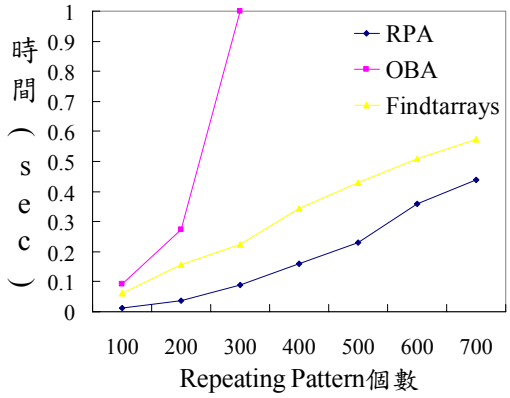

圖6 重覆片段個數的影響

5. 結論與未來研究方向

在現有找尋 DNA 序列中所有串聯重複序列(tandem arrays)的演算法，多採用先找到所有的銜接重覆(tandem repeat)，再透過對 tandem repeat 的分析比對，以找到所有的 tandem arrays。而於此研究報告中，我們新提出了 Repeating Pattern Alignment Approach 來找尋 DNA 序列中所有的 tandem arrays。我們的演算法，是先找出 DNA 序列中的所有重覆片段(repeating patterns)，再透過這些重覆片段發生的位置，依序做串接比對，以找出所有的 tandem arrays。因現有技術對找尋重覆片段已相當的有效率，我們利用先找重覆片段，而非找 tandem repeat，可以預期執行上將會較為快速。雖然我們所提出的新演算法與現有技術有著相同的時間複雜度，皆為 $O(n\log n)$，但我們經由實驗驗證了所提出的演算法，比現有演算法還來得更有效率，可更快速的找到 DNA 序列中所有的 tandem arrays。

未來研究方向，我們可朝找尋近似串聯重複序列(approximate tandem array)來發展，希望利用重覆片段來推演，也可以設計出更有效率的解決方案。

參考文獻

[1] D. Anastassiou, "Genomic Signal Processing," *IEEE Signal Processing Magazine,* Vol.81, 2003, pp.349-355.

[2] G. Benson, "An Algorithm for Finding Tandem Repeats of Unspecified Pattern Size," *In Proc. of the 2nd annual international conference on Computational molecular biology*, New York, Mar. 1998, pp. 20-29.

[3] M. Buchner and S. Janjarasjitt, "Detection and Visualization of Tandem Repeats in DNA Sequences," *IEEE TRANS. ON SIGNAL PROCESSING,* VOL. 51, NO.9, SEP. 2003.

[4] J. G. Chen, "Finding All Tandem Arrays in DNA Sequences," *master thesis*, Dept. of Computer Science and Information Engineering, National Chi-Nan University, Taiwan, 2005.

[5] C. F. Cheung, J. X. Yu, and H. Lu., "Constructing Suffix Tree for Gigabyte Sequence with Megabyte Memory," *IEEE TRANSATCTION ON KNOWLEDGE AND DATA ENGINEERING,* Vol. 17, No. 1, JAN. 2005.

[6] J. Cohen, "Bioinformatics-An Introduction for Computer Scientists," *ACM Computing Surveys*, Vol. 36, No.2, June 2004, pp. 122-158.

[7] D. Gusfield, "Algorithms on Strings, Trees, and Sequences," *Computer Science and Computational Biology,* the Press Syndicate of the University of Cambridge, New York 1997.

[8] H. Hammond, L. Jin, Y. Zhong, C. T. Caskey, and R. Chakraborty, "Evaluation of 13 short tandem repeat loci for use in personal identification applications." *Am J Hum Gent*, 1994, 55:175-189.

[9] E. Hunt, M. P. Atkinson, and R. W. Irving, "Database indexing for large DNA and protein sequence collections," *the VLDB Journal*, 11:256-271/Digital Object Identifier (DOI), 2002.

[10] Y-L. Lo, W-L. Lee, and L-H. Chang, "True Suffix Tree Approach for Discovering Non-trivial Repeating Patterns in a Music

Object," accepted by *Journal of Multimedia Tools and Applications*, Springer, to appear.

[11] Y. L. Lo and W. L. Lee, "Linear Time for Discovering Repeating Patterns in Music Databases," *the 2004 IEEE International Conference on Multimedia and Expo (ICME),* 2004.

[12] M. Main, and R. Lorentz, "An $O(n\log n)$ Algorithm for Finding All Repetitions in a String," *Journal of Algorithms*, Vol. 5, 1984, pp.422-432.

[13] M. Sammeth and J. Stoye, "Comparing Tandem Repeats with Duplications and Excisions of Variable Degree," *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, Vol. 3, Issue 4, Oct. 2006, pp. 395-407.

[14] J. Stoye, and D. Gusfield, "Simple and flexible detection of contiguous repeats using a suffix tree," *Theoretical Computer Science*, Vol. 270, Issues 1-2, 2002, pp. 843-856.

[15] E. Ukkonen, "On-Line Construction of Suffix Tree," *In Proc. of the 14th Ann. Symp. on Switching and Automata Theory*, 1995, pp. 1-11.

[16] P. Weiner, "Linear Pattern Matching Algorithms," *In Proc. of the 14th Ann. Symp. on Switching and Automata Theory*, 1973, pp. 1-11.

[17] H. E. Williams, and J. Zobel, "Indexing and Retrieval for Genomic Database," *IEEE Trans. on Knowledge and Data Engineering*, Vol. 14, NO. 1, Jan./Feb. 2002.

[18] H-X. Zhou and H. Yan, "A Fast Method for Determining the Repeat Pattern Size in DNA Sequences," *In Proc. of the 6th Int'l Conf. on Machine Learning and Cybernetics*, Vol.6, Hong Kong, Aug. 2007, pp. 3314-3318.

[19] National Center for Biotechnology Information (NCBI), http://www.ncbi.nlm.nih.gov/.