應用演化式演算法解決多重蛋白質序列排比之問題

賴智錦　何誠禎
樹德科技大學資訊管理系
E-mail: cclai@mail.stu.edu.tw

摘要

近幾年來，多重序列排比的問題已成為分子生物學上重要的研究議題。多重序列排比是目前分子生物序列分析上最為常見且重要的技術，其主要目的在於呈現結構上的差異。在本論文中，我們應用演化式演算法與啟發式微調技術來解決多重序列排比的問題，多個演化式運算元將重新設計來協助改善排比的結果。在實驗的資料上，我們採用真實的蛋白質序列資料做為測試對象。為了驗證所提方法之可行性，我們分別與 *Clustal W*、*DiAlign* 及 *DCA* 等序列排比軟體進行比較，在實驗數據上呈現出本論文所提出的方法確實能夠找到更好的序列排比解答。

關鍵詞：多重序列排比、演化式演算法

一、緒論

近幾年來，多重序列排比(multiple sequence alignment, *MSA*)在計算分子生物學上被視為最常使用且相當重要的技術。多重序列排比通常應用於三條或三條以上的序列中，其目的在於呈現出序列間的特徵模式、結構間的差異、相似程度和基本架構。因此，生物學家常以多重序列排比為方法，從眾多的序列中找出相同或相似性極高的部分，這些部分可提供生物學家分析未知序列的特徵及有助於發掘出隱含的生物資訊與意義。多重序列排比可以應用的範圍相當的廣泛，例如，親族間的特徵模式、察覺基因疾病、犯罪的鑑定、親緣樹的判定、預測新序列的二級結構或三級結構、及藥物的研究等。因此，在分子序列分析相關問題上，多重序列排比一直被視為資料分析的必要前置處理動作。

分子生物序列在排比的過程中在於求得最佳的排列組合，序列長度與數量多寡會影響問題解決的難易度。因此，多重列序排比問題在序列數量太多或序列太長時，其存在解是否能在合理的時間內找到，已被視為 *NP*-hard [5] 之問題。到目前為止，已經有許多方法應用於多重序列排比之問題上 [7]，較受囑目的方法可分為：漸進式演算法 (progressive algorithms) 與迭帶式演算法 (iterative algorithms)。

漸進式演算法是一種啟發式演算法，既不會浪費時間與記憶體空間，且排比的效果也不錯。*Clustal W* [9] 是根據漸進式演算法所設計的一套序列排比軟體。排比的過程是先以兩條序列進行排比，排比後的結果再從尚未排比的序列中取出一條序列接續排比的動作。依此程序，直到把全部的序列排比完成。Notredame 等 [6] 提出分子序列排比包含全域排比 (global alignment) 和區域排比 (local alignment)。序列排比的整個流程可分為二個部分：(1)將結合全域性排比與區域性排比後的結果透過配對分數的準則作初步的評估，(2)將評估後的排比結果利用一致性演算法調整其排比的組合。Thomsen 等 [10] 利用現存的分子序列排比工具(*Clustal V*)產生一個排比結果當作演化式演算法初始解答的種子(seed)。根據演化式演算法特性，再把種子解答導入跟其它隨機產生的解答一起演化。在演化的過程中，其可能解答將與所提出的交配運算元與突變運算元進行修正與微調。實驗結果顯示可以改善 *Clustal V* 的排比效果，演化速度也大大提昇。

迭帶式演算法亦是另一種啟發式演算法。此觀念的用意在於針對解答以重覆計算、調整及修正來跳脫局部最佳解，以致於找出問題的近似最佳解。Wayama 等 [11] 在使用遺傳演算法解決氨基酸序列排比的問題。在染色體編碼部分以"1"與"0"分別表示空白及序列間的字母位置。Notredame 和 Higgins [5]表示使用一般方法來處理分子序列排比問題時，會發生在序列數量增加的情形下，導致無法找到較好的解答；因此，提出利用推測性的方式解決分子序列排比之問題。他們提出的 *SAGA* 方法是利用遺傳演算法輔以所設計的二十個突變運算元來解決多重序列排比之問題。

Zhang 和 Wong [13] 建置一個結合遺傳演算法與兩條序列比對的動態規劃法(pairwise dynamic programming)的系統。在此系統中，遺傳演算法應用在搜尋完全對位的區塊(match blocks)；兩條序列比對的動態規劃法用於處理在兩個完全對位區塊之間尚未排比好的序列。

由於多重序列排比之問題可視為最佳化的問題之一；因此，在本論文中我們採用演化式演算法來解決多重序列排比的問題。我們將以演化式演算法中的遺傳演算法為基礎，重新設計遺傳演算法中所用到的遺傳運算元以應用於解決多重蛋白質序列排比之問題。

本論文中，第二節將回顧多重序列排比的問題，並敘述所提出的方法。實驗結果與討論置於第三節，最後則是結論與未來的研究方向。

二、多重序列排比問題與所提方法

2.1 多重序列排比問題

多重序列排比的基本定義如下：

假設有一組序列 $X=\{X_1, X_2, ..., X_m\}\,(m\geq 3)$，其序列尚未進行排比的狀況如下：

$$X_1 = X_{1,1}X_{1,2}...X_{1,n_1}$$
$$X_2 = X_{2,1}X_{2,2}...X_{2,n_2}$$
$$\cdots$$
$$X_m = X_{m,1}X_{m,2}...X_{m,n_m}$$

其中，

- $\forall X_{p,q} \in \sum$, $X_{p,q}$代表在第 p 條序列中第 q 個位置。
- $1 \leq p \leq m$, m 表示為序列的個數。
- $1 \leq q \leq n_p$, n_p 表示為第 p 條序列長度。

多條序列在經過排比後，其結果呈現如下：

$$X'_1 = X'_{1,1}X'_{1,2}...X'_{1,N}$$
$$X'_2 = X'_{2,1}X'_{2,2}...X'_{2,N}$$
$$\cdots$$
$$X'_m = X'_{m,1}X'_{m,2}...X'_{m,N}$$

在 m 條序列中，$\max\{n_1, n_2, ..., n_m\} \leq N \leq \sum_{p=1}^{m} n_p$

- $\forall X'_{p,q} \in \sum \cup \{-\}$，$1 \leq p \leq m$，$1 \leq q \leq N$。
- N 代表多重序列排比後其序列的長度，但不包含在排比後整個欄位都是空白的部份。

2.2 所提方法

演化式演算法是一種迭帶式的方法，它提供了平行且強韌性的搜尋方式。在本論文中，我們利用演化式演算法中的遺傳演算法 [3] 為基礎，加上重新設計的交配、突變與啟發式微調運算元，以增加多重序列排比結果的正確性。在演化的過程中，序列排比的組合會根據選擇、交配與突變運算元的調整或修正呈現出不同的排列組合。

■ 染色體編碼方式

序列排比是在比對的過程中，適當的加入空白(space 或 gap)以增加序列組合(解答)的多樣性。由此得知，我們利用遺傳演算法的目的便是搜尋每條序列中合適的空白位置。因此，可將排比結果中的空白位置予以編碼成遺傳演算法中的染色體。

假設有 k 條序列且序列長度分別為 l_1, l_2, …, l_k。我們可以將每條染色體的空白數定為：

$$\sum_{i=1}^{k}(w = l_i) \quad (1)$$
$$w = (1 + r_{sp}) \times l_{\max} \quad (2)$$

其中 w 代表每一條序列加上插入空白數後的長度，$l_{\max}$ 代表著在 k 條序列中最長的序列長度，r_{sp} 則表示要插入序列中空白數的比率。插入序列中空白數的比率會影響整體序列排比的結果。若是插入空白數的比率太大，在排比的過程會浪費許多時間與記憶體去調整多餘的空白數量。若是插入空白數的比率太小，在排比的過程會影響到排比結果的正確性或失去較好的排比結果。因此，在解決這類問題時，r_{sp} 的預設值通常會設定為 0.2 至 0.5[1]。

■ **初始化族群**

在演化式計算的過程中，首要的步驟就是先初始化族群。在本論文中，我們以隨機的方式來產生其可能的解答。

■ **適應函數**

適應函數是針對染色體作出適當評估及判斷其優劣。Sum-of-pairs (*SP*)是非常廣泛地應用在多重序列排比的問題上用以計算分子序列間對位分數的一種函數 [8]，其標準架構如公式(3)與(4)所示：

$$SP-Score(X_1\#X_2\#\cdots\#X_k) = \sum_{i=1}^{N} C_{score}(X'_{1,i}, X'_{2,i}, \ldots, X'_{k,i}) \quad (3)$$

$$C_{score}(X'_{1,i}, X'_{2,i}, \ldots, X'_{k,i}) = \sum_{1\le p<q\le k} P_{score}(X'_{p,i}, X'_{q,i}) \quad (4)$$

$SP-Score(X_1\#X_2\#...\#X_k)$代表著將所有長度為 N 的序列進行兩兩逐對排比並根據分數矩陣表計算其分數的加總。$C_{score}(X'_{1,i}, X'_{2,i}, \ldots, X'_{k,i})$代表著在多重序列排比中, 針對每條序列的第 i 個欄位的字母(符號)做比對及依據分數矩陣表進行評分與計算加總分數之動作。其中，$X'_{u,i}(1 \le u \le k)$便是代表在多重列排比後的第 u 條序列中的第 i 個欄位。$P_{score}(X'_{p,i}, X'_{q,i})$則代表著在第 p 條序列與第 q 條序列中的第 i 個欄位的字母對應到分數矩陣表上的相對分數。在此問題上有許多不同的分數矩陣表，例如：*PAM250*、*Blosum62* 與 *Gonnet250* 等 [2]；其中，*PAM250* 與 *Blosum62* 最常被各式序列排比工具所使用。

■ 選擇運算元

選擇運算元是依據適應函數進行評估染色體的優劣後，選取較為優秀的染色體作為子代的候選人。在擁有優秀染色體的族群環境下，透過演化的過程能夠有較高的機率產生出優秀的子代染色體。在此，我們採用競爭法(tournament selection)的方式來挑選較為優秀的染色體。

■ 交配運算元

交配運算元是將族群中任取兩條染色體交換彼此所擁有的資訊，其目的在於產生具有更多可能性與多樣性的染色體。我們使用了標準的單點交配(Standard One-Point Crossover, *SOPC*)與改良式的單點交配(Modified Standard One-Point Crossover, *MSOPC*)。改良式單點交配的運算方式是：在進行交配過程之前，必須針對本身染色體中的空白位置進行對位。在兩條染色體中的空白位置分別對位完後，才可進行標準的單點交配。其目的在於不破壞本身染色體中原有對位完成的空白位置。

■ 突變運算元

在經過複製與交配的過程後，突變運算元可幫助染色體內的空白位置微調(移動)到適當的位置。我們提出三種不同的突變運算元，分別為 MergeColumnSpace 運算元、ShiftRowSpace 運算元與 ModifyShiftRowSpace 運算元。

- MergeColumnSpace 運算元

此運算元的目的在於縮小突變的範圍與產生更多相同欄位的空白位置。因此，突變的可能範圍來自於染色體本身所具有的空白位置。此方法的步驟如下：

步驟一：以隨機的方式選擇一列為標準列(base row)。

步驟二：接著根據突變率來判斷每個基因是否要突變。基因若要突變，便再判斷其基因(空白位置)是否為標準列其中一員，若是，此基因便不須再突變，若不是，此基因的突變範圍便從染色體中挑選。

步驟三：在突變的過程，標準列是其他序列的基準，所以不需進行突變的動作。

- ShiftRowSpace 運算元

ShiftRowSpace 運算元的目的在於針對序列排比中的小範圍，進行空白位置的調整以求得較適當的序列排比。*ShiftRowSpace* 運算元將會以隨機的方式從序列排比中挑選一部分且為連續的欄位，利用挑選出來的部分製造一個樣版序列(template sequence)。然後，根據突變的機率來判斷每一序列是否有突變的可能。若是有序列被挑選到要進行突變，可利用樣版序列與被挑選的序列(target sequence)以動態規劃的方式進行序列排比。此運算元的運作描述如下：

步驟一：將染色體內空白位置的座標與測試資料所組成的序列排比，如圖 1(a)所示。

步驟二：以隨機的方式從原排比的序列中選取某個區段，假設從第 22 個欄位取至第 36 個欄位，如圖 1(b)所示。此區段我們定義成子序列排比(sub-alignment)。

步驟三：我們將子序列排比中的每一個欄位裡出現頻率最高的符號組合起來，所得到的序列即為樣版序列。如果在欄位中有出現相同頻率的情況，以隨機的方式挑選其中一個符號為主。接著，我們根據突變的機率從子序列排比中選擇要突變的序列。在樣版序列與要突變的序列作排比之前，要突變的序列必須先將序列中的空白位置抽離掉，以達到序列中的空白位置可重新調整至適當的位置。在圖 1(c)中，s'_1 與 s'_4 是從子序列排比中根據突變的機率所選擇，再經過抽離本身的空白位置所得的序列，而 t 序列就是我們所塑造出來的樣版序列。

步驟四：將 s''_1 與 s''_4 分別與 t 序列以動態規

劃的方式重新調整其空白位置。不過，在以此方式進行排比時，唯一的限制就是在排比對位時不得在 *t* 樣版序列中插入或調動任何一個空白位置。在排比的過程中，空白會被適當的插入其正確位置以得到最佳的排列組合。圖 1(d)則代表著與 *t* 樣版序列排比完所呈現的結果。

步驟五：針對序列重新排比的結果，可觀察到 s''_1 序列與原本的子序列相同，而 s''_4 序列跟原來的已有所不同。其表示 s_1 序列原本已呈現排列組合的最佳化，而 s_4 尚未達到最佳的排列組合。圖 1(e)代表著序列排比後的新結果。

- ModifyShiftRowSpace 運算元

在演化的過程中，*ShiftRowSpace* 運算元於某些的情況下會導致求得的結果陷入區域最佳解。因此，我們提出此運算元之目的在於修正 *ShiftRowSpace* 運算元的缺點。*ModifyShiftRowSpace* 運算元可幫助空白位置的調整及呈現出更多具有可能性的解答。此方法的運作流程大致與 *ShiftRowSpace* 運算元相同，其不同之處在於塑造樣版序列與挑選序列的步驟。在塑造樣版序列的方式上，是將 *ShiftRowSpace* 的樣版序列內的空白予以抽離，而挑選要突變的序列為子序列排比中全部且已抽離空白的序列。

三、實驗結果與討論

在本節中，我們透過實驗來驗証所提方法之正確及實用性。我們採用真實蛋白質序作為測式樣本，並與 *Clustal W*、*DiAlign* 及 *DCA* 等序列排比軟體進行比較。

本論文採用實際的生物分子序列資料做為測試資料(請參閱附錄)，這些測試資料的數量、長度及相似程度等資訊如表 1 所示。在實驗上，我們連續執行所提方法二十次，在這二十次結果中，取出最好的解答與其他軟體的結果進行比較。在實驗中，空白數出現的比率(r_{sp})設定為 0.2 [1]。在分數矩陣表上，我們採用 *PAM250* 的分數矩陣表，並且在分數表上作些微的修改 [12]。我們把這個分數矩陣表內的數值同等的提升，使得表內的數值都大於 0，除了字元對位空白時給予-1 與空白對位空白給予 0。進行測試所使用的 *PAM250* 分數矩陣表見表 2 所示。

在測試的結果(見表 3)中可發現，我們的方法在解決蛋白質序列上有不錯的表現。但是仍有些資料的表現上略低於其他的測試軟體。在我們的測試資料中，排比結果比較不理想的資料集(P7, P8, P14)幾乎都是序列數量較多且相似程度較低，這與 Chellapilla 及 Fogel [1]的觀點正好相符。在 Chellapilla 及 Fogel 的文章中指出，影響序列排比結果的原因有三：(1)序列的數量、(2)序列的平均長度及(3)序列的整體相似度 [1]；此三項因素往往也是其他排比軟體所亟欲克服的。除了上述因素影響排比結果，在所提方法上可能受限於突變運算元較少或是所設計的運算元較不適用於相似性過低的序列，導致所提的方法在序列相似度較低且序列數量較大的雙重影像時，比較不能得到適當的解答。整體而言，相較於 Notredame 與 Higgins [5] 設計了二十個運算元來解決多重序列排比的問題，我們能夠以較少的運算元配合後處理的微調機制，獲得如此的解答，足以表示我們所提出的方法確實可行。

表 1 蛋白質序列之測試資料表

ID	AveLength(max,min)	SeqNum	Similarity
P1	228(230,227)	6	0.64
P2	227(228,227)	7	0.65
P3	228(229,228)	6	0.60
P4	321(323,318)	4	0.72
P5	323(324,321)	12	0.46
P6	163(166,161)	4	0.42
P7	380(386,379)	15	0.21
P8	261(267,260)	15	0.57
P9	141(142,141)	9	0.49
P10	115(118,115)	15	0.35
P11	198(208,186)	4	0.30
P12	603(605,603)	6	0.74
P13	348(352,337)	10	0.50
P14	150(163,145)	15	0.17

Note: ID: sequence id number, AveLength(max, min): average, maximum, and minimum sequence length, SeqNum: sequence number, Similarity: percentage of matched columns based on the best alignment using Clustal W.

表 2 實驗所採用的 *PAM250* 分數矩陣表

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V	-
A	10																				
R	6	14																			
N	8	8	10																		
D	8	7	10	12																	
C	6	4	4	3	20																
Q	8	9	9	10	3	12															
E	8	7	9	11	3	10	12														
G	9	5	9	9	5	7	8	13													
H	7	10	10	9	5	11	9	6	14												
I	7	6	6	6	6	6	6	5	6	13											
L	6	5	5	4	2	6	5	4	6	10	14										
K	7	11	9	8	3	9	8	6	8	6	5	13									
M	7	8	6	5	3	7	6	5	6	10	12	8	14								
F	5	4	5	2	4	3	3	3	6	9	10	3	8	17							
P	9	8	8	7	5	8	7	8	8	6	5	7	6	3	14						
S	9	8	9	8	8	7	8	9	7	7	5	8	6	5	9	10					
T	9	7	8	8	6	7	8	8	7	8	6	8	7	5	8	9	11				
W	2	10	4	1	0	3	1	1	5	3	6	5	4	8	2	6	3	25			
Y	5	4	6	4	8	4	4	3	8	7	7	4	6	15	3	5	5	8	18		
V	8	6	6	6	6	6	6	7	6	12	10	6	10	7	7	7	8	2	6	12	
-	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0

四、結論

多重序列排比在分子生物學上是一個很重要的議題。具有效率與效能的序列排比方法，不僅可幫助分子生物學家快速、準確的找出序列間的差異，更可以幫助分子生物學家進行分析與判斷序列間彼此的特性。

本論文是以演化式演算法結合啟發式的微調機制以解決多重蛋白質序列排比的問題。我們針對問題的特性設計了交配運算元、突變運算元，期望透過演化式演算法的多點搜尋特性，能夠同時提供多組不同的可能解答，再經過世代的演化以搜尋到最佳近似解。其目的在於找出序列更好的排比組合，且能提供給分子生物學家作為一個可進行分析的依據。在實驗過程中，所提方法與 *Clustal W*、*DiAlign*、*DCA* 等排比軟體以蛋白質的實際資料進行測試。在實驗數據顯示上，所提方法在大部分的測試資料都可以找到優於排比軟體的結果。

誌謝

本研究承蒙國家科學委員會補助部分研究經費(計畫編號 NSC 90-2213-E-366-004 及 NSC 92-2213-E-366-005)，特此誌謝。

參考文獻

[1] K. Chellapilla and G. B. Fogel, “Multiple sequence alignment using evolutionary programming,” *Congress on Evolutionary Computation*, pp.445-452, 1999.

[2] C. Gibas and P. Jambeck, *Developing Bioinformatics Computer Skills*, O’REILLY, Reading, 2001.

[3] D. E. Goldberg, *Genetic Algorithms in Search*, *Optimization*, and *Machine Learning*, Addison-Wesley, Reading, 1989.

[4] H. D. Nguyen, I. Yoshihara, K. Yamamori and M. Yasunaga, “ Align multiple protein sequences by parallel hybrid Genetic Algorithm, ” *Genome Informatics*, vol. 13, pp.123-132, 2002.

[5] C. Notredame and D. G. Higgins, “SAGA: sequence alignment by genetic algorithm,” *Nucleic Acids Research*, vol. 24, no. 8, pp.1515-1524, 1996.

[6] C. Notredame, D. G. Higgins and J. Heringa, “T-Coffee: A novel method for fast and accurate multiple sequence alignment,” *Journal of Molecular Biology*, pp.205-217, 2000.

[7] C. Notredame, “Recent progresses in multiple sequence alignment: a survey,” *Pharmacogenomics*, vol. 3, no. 1, pp.1-14, 2002.

[8] J. Setubal and J. Meidanis, *Introduction to Computational Molecular Biology*, PWS, Boston, MA, 1997.

[9] J. D. Thompson, D. G. Higgins and T. J. Gibson, “CLUATAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position specific gap penalties and weight matrix choice,” *Nucleic Acids Research*, vol. 22, no. 22, pp.4673-4680, 1994.

[10] R. Thomesn, G. B. Fogel and T. Krink, “A Clustal alignment improver using evolutionary algorithms,” *Proc. of the Fourth Congress on Evolutionary Computation*, vol. 1, pp.121-126, 2002.

[11] M. Wayama, K. Takahashi and T. Shimizu, “An approach to amino acid sequence alignment using a genetic algorithm,” *Genome Informatics*, vol. 6, pp.122-123, 1995.

[12] B.-H. Yang, *An Approach to Multiple Protein Sequence Alignment Using A Genetic Algorithm*, Master thesis, Department of Computer Science and Information Engineering, National Central University, July, 2001.

[13] C. Zhang and A. K. C. Wong, “Toward efficient multiple molecular sequence alignment: A system of genetic algorithm and dynamic programming,” *IEEE Transactions on Systems, Man*, and *Cybernetics* Part B, vol. 27, no. 6, pp.918-932, 1997.

附錄

P1 Data Set：

NP_008345 NP_008215 NP_008228
BAA95619 AAK38692 AAB00992

P2 Data Set：

AAK08554 AAK08578 NP_071645
NP_068785 NP_007371 NP_008098
NP_007384

P3 Data Set：

AAK14328 AAK14327 AAG49582
NP_008649 NP_008279 AAK00949

P4 Data Set：

AAK08547 AAK08571 NP_071642
NP_068782

P5 Data Set：

NP_066555 NP_112132 NP_112522

NP_007562 NP_110504 NP_114224
BAB40348 AAF61378 BAB20733
BAB20720 BAB20707 AAK52942

P6 Data Set：

LGHO2 S33878 S14719
S11538

P7 Data Set：

NP_007574 NP_112532 NP_007380
NP_008107 NP_007393 NP_008237
NP_008055 NP_115361 NP_115439
NP_071654 NP_068794 BAA85284
AAG28215 BAA95628 AAK38699

P8 Data Set：

NP_007568 NP_112528 NP_007374
NP_008101 NP_007387 NP_008231
NP_008049 NP_115356 NP_115434
NP_071648 NP_068788 BAA85281
AAG28222 BAA95622 AAK38695

P9 Data Set：

HAAQ HACQ A26543
HAHOD P01923 P06635
HAOR P01965 A45964

P10 Data Set：

NP_007569 NP_112717 NP_007375
NP_008102 NP_007388 NP_115357
NP_115435 NP_071649 NP_068789
NP_008232 NP_008050 AAG28223
BAA95623 AAK38696 BAA85282

P11 Data Set：

P41048 P41049 P25027
P41045

P12 Data Set：

P70389 P35859 P35858
Q9DBI7 O70211 O02833

P13 Data Set：

AAD01995 O70160 P00978
P04366 P02760 Q60559
Q64240 Q62577 Q07456
Q9DBJ9

P14 Data Set：

AAB24577 O02480 P02212
P02213 P04251 P14395
P14821 P14822 P925165
Q17155 Q17156 Q17157
Q25269 Q26505

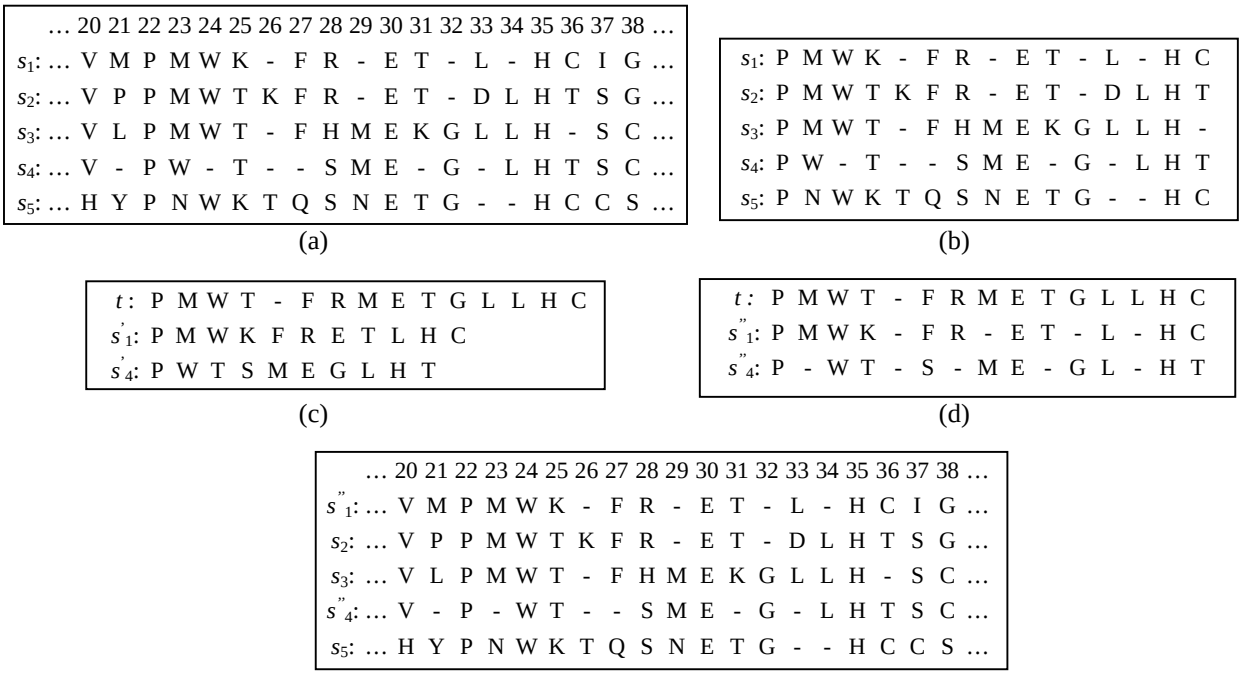

圖 1 ShiftRowSpace 運算元之運作流程 (a)序列排比，(b)子序列排比，(c) *t* 樣版序列與要突變的子序列之表示方式，(d)子序列重新排比的結果，及(e) 重新排比的可能結果。

表 3 所提方法與其他方法之比較表

Data set	The proposed approach		*Clustal W*		*DiAlign*		*DCA*	
	Score	M.C.	Score	M.C.	Score	M.C.	Score	M.C.
P1	42336	147	42304	147	42304	147	42304	147
P2	59937	150	59922	149	59902	149	59922	149
P3	43081	139	43081	139	43081	139	43081	139
P4	23925	233	23895	232	23900	232	23927	233
P5	260934	149	260433	148	261756	148	260717	148
P6	11065	69	11018	69	10913	69	11062	69
P7	494253	172	498316	180	498584	184	498842	182
P8	355171	142	357010	153	357043	153	357010	153
P9	60508	67	60508	67	60508	67	60508	67
P10	68496	8	68125	9	67269	9	68365	9
P11	12532	64	12481	62	11974	62	12471	61
P12	113639	448	113639	448	113718	448	113639	448
P13	193692	178	193602	178	193619	178	193619	178
P14	162303	25	165983	29	167621	29	166505	29

Clustal W, progressive-based method, http://www.csc.fi/mobio/progs/clustalw/clustalw.html
DiAlign, consistency-based method, http://bibiserv.techfak.uni-bielefeld.de/dialign/submission.html
DCA, exact-based method, http://bibiserv.techfak.uni-bielefeld.de/cgi-bin/dca_submit
M.C.: matched columns