模糊理論與混合式遺傳演算法應用於基因序列排比

研究生：鄭尊仁　　指導教授：張炳騰 博士
曾宗瑤 博士

東海大學工業工程研究所

摘要

本研究提出一個新的方法，用來建立蛋白質成對序列排比。目前在蛋白質序列排比上一直存在一個致命的問題，那就是當使用明確值資料去進行分析時，太多的不確定因子與敏感性資訊的遺失，導致序列排比問題出現瓶頸。由於使用不同的軟體和演算法會造成不同的結果，對於正在研究生物基因的科學家們，不同演算法亦很難廣泛地運用於基因序列上。因此基於這個最重要的前提下，本研究提出模糊的概念，將250單點突變矩陣(point accepted mutations, PAMs)與62區塊突變矩陣(blocks substitution matrix, BLOSUM)利用基因演算法(genetic algorithm, GA)於序列排比上。最主要的目的是用來減少不確定因子的影響，避免利用明確值或權重的方式，造成重要資訊的遺失，以及增加解的正確性與適用性。

實驗果顯示，不論是利用PAM250還是BLOSUM62矩陣，利用GA演算法皆能找到更長且配對的蛋白質序列，並在不同矩陣的運用上，利用模糊矩陣所產生解的變動性要比明確值小，也就是說，將模糊概念運用於序列排比，確實能夠減少不確定性的影響並且增加解在區域相似上的利用性。在混合式基因演算法上，本研究利用基因演算法每代所搜尋出的最佳解，再去進行禁忌搜尋法(Tabu search)，但結果顯示要比基因演算法差。

關鍵詞：蛋白質序列排比、基因演算法、模糊理論、仿射性間格懲罰函數

Application of Fuzzy Set Theory and Hybrid Genetic Algorithm to Gene Sequence Alignment Problems

Student：Tsun-Jen Cheng　　Advisor：Dr. Ping-Teng Chang
Dr. Tsueng-Yao Tseng

Institute of Industrial Engineering & Enterprise Information
Tunghai University

ABSTRACT

In this paper a novel way to construct pairwise alignment of protein sequence is proposed. Currently in protein sequence alignment the vital problem is having too many uncertain factors and causes significant data loss while using crisp data. Due to using different software and algorithms that will bring about different results, for scientists researching in protein, different algorithms will be difficult to use widespread. Therefore, for this important premise, fuzzy concept is introduced and fuzziness is implemented in the matrix for 250 point accepted mutations (PAMs) and matrix for 62 blocks substitution matrix (BLOSUM) in sequence aligning, and integrated with the Genetic algorithm (GA) and Hybrid Genetic algorithm (HGA). The purpose for this implementation is to reduce the effects of uncertain factor, avoid making use of crisp values or weights resulting in significant data loss, and increase solution accuracy and method suitability.

Wether experimental results of fuzzy matrix for 250 PAM or BLOSUM62 can find more continuous and identical protein sequence after sequence alignment by GA. And using differet fuzzy matrices can make results of variation trivial than crisp matrices. We utilize HGA, using optimal solution of every epoch to combine with tabu search. the results show that it is not better than GA.

Keywords: protein sequence alignment、genetic algorithm、fuzzy theory、affine gap cost.

誌謝

鳳凰花開，又到了畢業的季節。回首兩年來在研究所的點點滴滴，猶如昨日而歷歷在目。在此要感謝指導教授 張炳騰博士在研究領域上的諄諄教誨，使我獲益良多。我要特別感謝我最親愛的父母親，在我求學的過程中始終是我最佳的後援，給我最大的支持和鼓勵，讓我可以盡情揮灑。還有一路上與我共同成長的女友，總能適時給我最中肯的建議，幫助我度過無數挑戰。研究室學長 國楨、國平時常撥空與我討論論文細節及研究上的細心協助，研究伙伴文偉、瑋珊、敬淳、俊嘉兩年來的一同努力打拼及學弟妹的幫忙，一併附上最真摯的謝意。

最後感謝口試委員 白炳豐博士、陳琨太博士與曾宗瑤博士對研究論文上缺失的指正與建議，讓此篇論文能更趨於完善。

鄭尊仁 謹誌於

東海大學工業工程與經營資訊研究所

民國九十五年六月

目錄

表目錄

圖目錄

第一章 序論

1.1 研究背景

西元 1953 年，Watson 與 Crick 兩位學者共同解開了 DNA 雙股螺旋結構開始，便已經決定敞開分子生物學領域的大門，到了 2000 年 6 月 26 日，人類聯合發表對於完成整個人類基因體序列草圖的繪製，從此便進入了後基因體時代。從過去幾十年，生物學家不論是用化學的方式定序或配合科技(MicroArray)，累積了非常大量的基因序列資料，因此只是利用人力的分析是無效率且不切實際的。於是，生物資訊學(Bioinformatics)便應孕而生，生物資訊學意指所有電腦處理、分析生物學研究過程，包含利用電腦、網路進行資料的收集與整合、利用影像處理設備將研究過程所得的圖像比對，及大多數分子生物學者比較熟悉的巨分子序列分析比對、結構預測等。

所以，生物資訊學的成員是一個大集合，包含生物學家、病理家、數學家及程式設計者等，集合多個不同領域的專家。在後基因體時代，面對如此龐大的資料，便要依靠高度的電腦運算能力。在基因定序階段，主要依賴資訊業人才從事生物資訊工具的開發，然而在蛋白質體與基因功能的分析，就需要生物學家的協助，生物學家面臨了資訊整理與資訊分析的挑戰。生物資訊的資料日漸擴增，生物、醫學、藥學等領域研究人員想藉由生物資訊資料庫加快實驗目標的設定，一般而言研究人員會以已定序但未知功能的生物序列，從已知的資料庫內查詢此序列是否存在，或者是否存在相似的序列。另外，基因分析和資訊的管理，更是需要資訊與生物科技的整合性人才，這也是現階段各國在發展生物科技上最需要解決的問題。

由於基因工程與基因計畫的發展，大量的蛋白質與 DNA 的排序資料被發現出來。排序資料發現後，進一步則是要找出特定排序所決定的功能為何。而這些則需依賴生物技術。生物技術是利用生物程序、生物細胞、或其代謝物質來製造產品，改進傳統生產程序及提升人類生活素質之科學技術，不但是研究生命科學、醫學之基本工具，更是一種跨學科之整合性科學。同時生物技術具有廣泛深遠之應用潛力，其應用範圍包括醫學、農學、海洋、能源、環保、化工、礦冶等領域，可說是上個世紀繼石油化學、航空、核能及資訊科技後之另一次技術革命。所謂的基因工程就像技術科學的工程設計，按照人類的需要把這種生物的這個“基因”與那種生物的那

個“基因”重新“施工”，“组装”成新的基因组合，創造出新的生物。這種完全按照人的意願，由重新组裝基因到新生物產生的生物科学技術，就稱為“基因工程”，或者說是“遺傳工程”。以下將對去氧核醣核酸(Deoxyribo Nucleic Acid, DNA)做一簡單介紹。

生命體是由數十種不同種類的元素所組成，其中碳(C)、氫(H)、氧(O)、氮(N)等元素占了 99%。所以生命體是由上述等元素所製作出複雜的有機化合物，如碳水化合物(醣類)、脂質、蛋白質、核酸等物質，這些都是生物體組成的根本。

假設我們把 DNA 想像成兩條細長的線，互相纏繞(圖 1-1(a)) [1]，就像由兩股細繩編成的繩子。可是仔細觀察 DNA 的生化性質，就知道兩條線都是由小的次單元所組成，有如珠子串成的長鏈(圖 1-1(b)) [1]，而珠子數目可能有好幾百萬個。化學家稱為核苷酸，而每一個核苷酸有 3 部分(圖 1-2)：名為鹼基的扁平環狀結構，稱為去氧核糖的環狀糖分子、以及磷酸。DNA 的骨幹由去氧核糖和磷酸鹽交互連結而成。鹼基則和去氧核糖相連，位在 DNA 兩條骨幹之間，和骨幹的主軸垂直(圖 1-1(c)和圖 1-1) [1]。兩條 DNA 的骨幹互相纏繞，形成我們常常聽到的「雙螺旋」(圖 1-1(c)和圖 1-3) [1]。

鹼基一層搭在一層之上，有如螺旋狀樓梯的一個個階梯。鹼基有四種(圖 1-4) [1]，通常用 A、T、G、C 來簡稱，它們真正的化學名稱是腺嘌呤（adenine, A）、鳥糞嘌呤（guanine, G）、胞嘧啶（cytosine, C）及胸腺嘧啶（thymine, T）。由於每個核苷酸酯含有一個鹼基，所以我們可以用鹼基代號來代表核苷酸。

DNA 的核苷酸排列有一定的順序，細胞就是利用這種性質來儲存一傳資訊，原理和利用點與線組合來傳遞消息的摩斯電碼很類似。

如果進一步研究 DNA 的資訊如何轉換成蛋白質，就可以更進一步了解基因的本質。基因的資訊並不是在鹼基字母上，而是儲存在鹼基拼出的字。DNA 的資訊可以製造出蛋白質長鏈分子，蛋白質鏈由胺基酸所組成，然而胺基酸有 20 種，核苷酸只有 4 種，因此我們可以知道若每三個核苷酸為一組，就有 64 種可能的組合，足夠代表 20 種胺基酸，而且也能包括怡傳資訊的標點符號，例如代表開端與結尾的訊號。

這種相當於一個胺基酸的一組三個核苷酸，稱為密碼子(codon)。細胞讀取遺傳密碼時，DNA 的資訊會先轉成核糖核酸（ribonucleic acid, RNA）。

RNA 與 DNA 很類似，但是 RNA 的鹼基組合中沒有 T，而是 U(尿嘧啶)，而且醣分子也稍微不同。圖 1-5 列出轉換成 RNA 形式的遺傳密碼。把 DNA 的資訊轉換成 RNA 的過程，稱為轉錄（transcript），也就是合成 RNA 的意思。這種 RNA 叫做信使 RNA（message RNA, 簡稱 mRNA）。接著，信使 RNA 的資訊在轉換成蛋白質的胺基酸序列，這個過程稱為轉譯(translation)。圖 1-6 為 DNA 轉譯成蛋白質的概念圖。

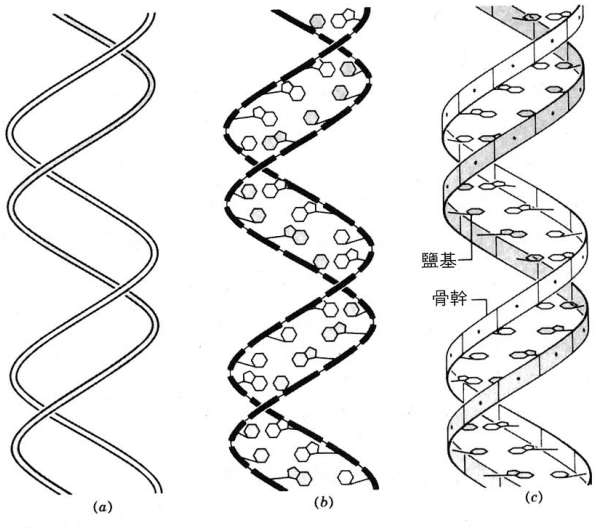


圖 1- 1DNA 結構圖

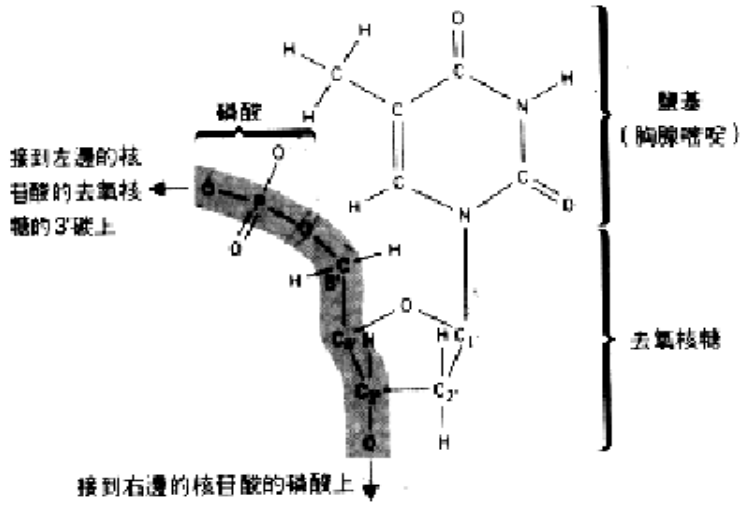


圖 1- 2 去氧核醣核苷酸的構造

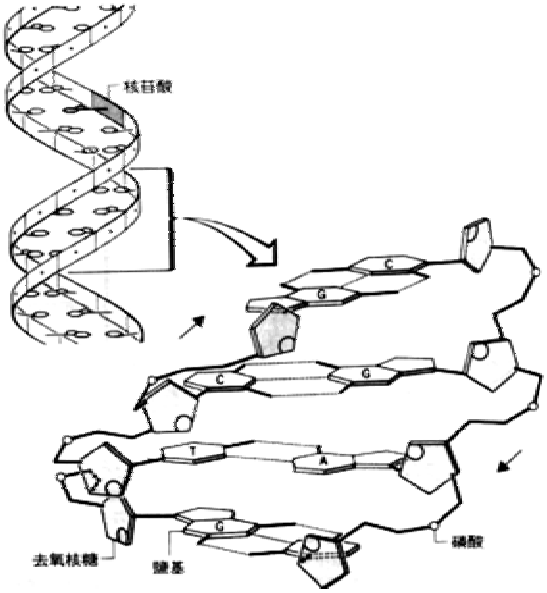


圖 1- 3 一段 DNA 雙螺旋

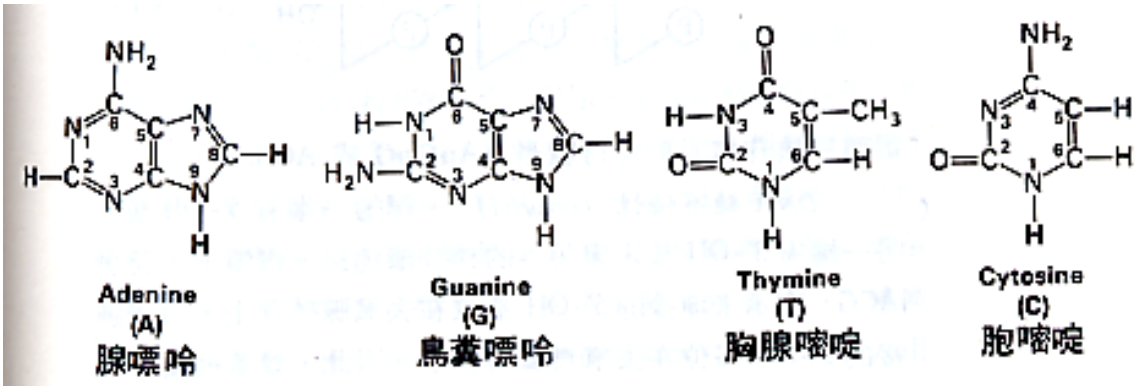


圖 1- 4 四種鹼基結構

UUU, UUC } Phe UUA, UUG } Leu	UCU, UCC, UCA, UCG } Ser	UAU, UAC } Tyr UAA, UAG } TERM	UGU, UGC } Cys UGA TERM UGG Trp	
CUU, CUC, CUA, CUG } Leu	CCU, CCC, CCA, CCG } Pro	CAU, CAC } His CAA, CAG } Gln	CGU, CGC, CGA, CGG } Arg	
AUU, AUC, AUA } Ile AUG Met	ACU, ACC, ACA, ACG } Thr	AAU, AAC } Asn AAA, AAG } Lys	AGU, AGC } Ser AGA, AGG } Arg	
GUU, GUC, GUA, GUG } Val	GCU, GCC, GCA, GCG } Ala	GAU, GAC } Asp GAA, GAG } Glu	GGU, GGC, GGA, GGG } Gly	

圖 1- 5 基因轉譯碼

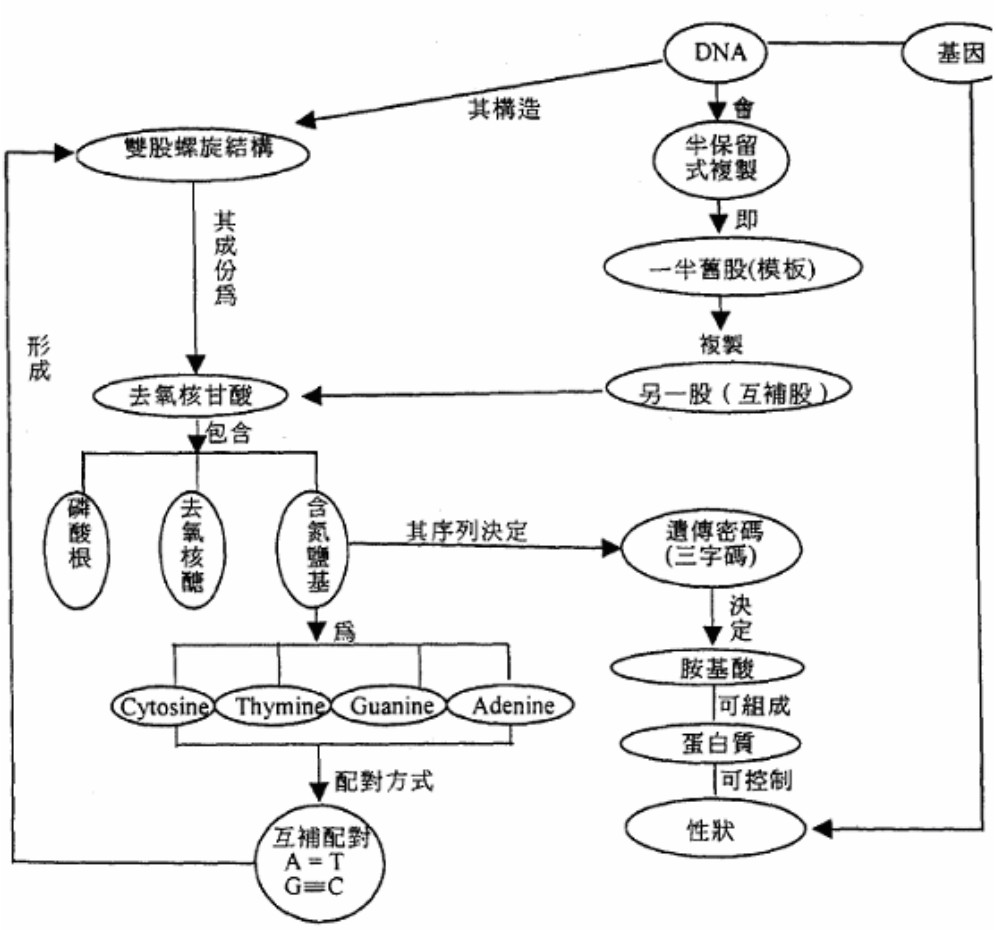


圖 1- 6DNA 概念圖

首先基因內包含了許許多多的 DNA，而 DNA 是以一種名為雙螺旋的結構存在，其主要成分去氧核苷酸，雙螺旋結構則是利用半保留式複製的方式形成(圖 1-7)，而半保留式的複製方式，就是將兩股中的一股留下，留下的為舊股，另一新股則以互補配對方式形成，及鹼基 A 會與鹼基 T 配對，鹼基 G 則會與鹼基 C 配對。由於每個核苷酸只含有一個鹼基，因此我們可以用鹼基來代替核苷酸說明，而鹼基的組合方式則會決定產生出何種遺傳密碼(密碼子,是一種胺基酸)，換句話說，每 3 個鹼基可組成一個密碼子，而許多的密碼子則可以形成蛋白質進而控制人體的性狀表現。

蛋白質通常是由 20 種胺基酸(amino acid)所組合而成的，如附錄二所示，胺基酸的基本結構可分為三個部分，分別為胺基(-NH2)、羧基(-COOH)、支鏈(R)，就如圖 1-8 所示。

胺基酸的基本結構：

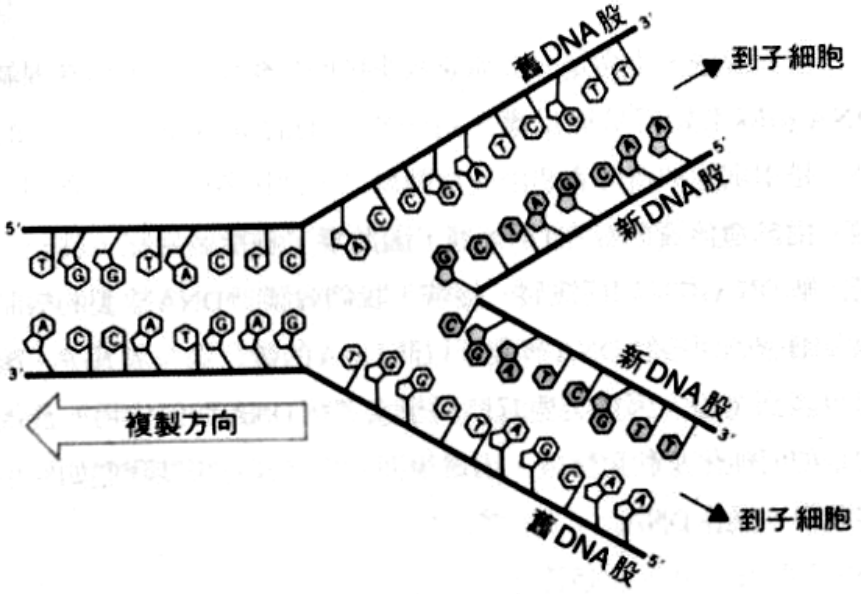


圖 1- 7 半保留式複製

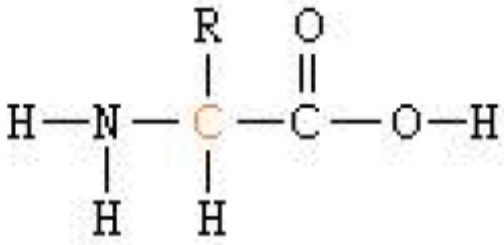


圖 1- 8 胺基酸的基本結構

綜合上述所說，我們可以由圖 1-9 清楚的看出上述內容的流程：

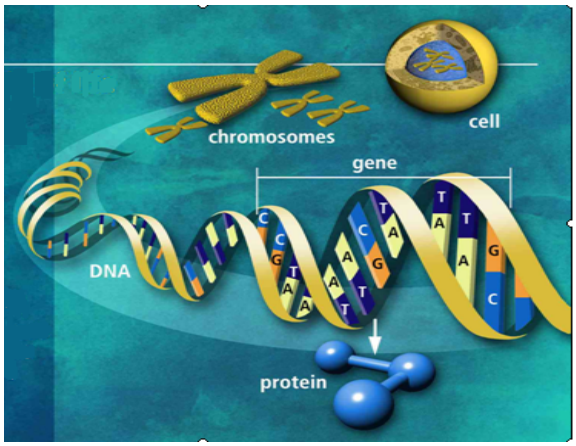


圖 1- 9DNA 轉譯成蛋白質

生物技術的定義是利用生物或其產物來生產對人類醫學或農業有用的物質或生物。依歷史發展或所用方法的不同，可分成以下兩大類：

傳統生物技術：應用釀造醱酵、配育新種等傳統的方法來達致上述目的。

現代生物技術：以生物化學或分子生物方法改變細胞或分子的遺傳形質以達上述目的。本研究簡單將現代生物技術分成以下四個範疇：

1. 基因操作:把外來基因經重組後導入宿主細胞中，則可表現並生產此基的有用產物。
2. 細胞培養:人工培養生物細胞，可大量生產所代謝的有用物質，或經再成為新個體。
3. 單株抗體:可生產有用抗體的淋巴細胞若與癌細胞融合，則形成穩定而培養細胞株
4. 酵素工技:將酵素固定化或修飾，可增加穩定性或專一性，也有人造酵或催化抗體。

以下為現代生物技術的整體關聯圖：

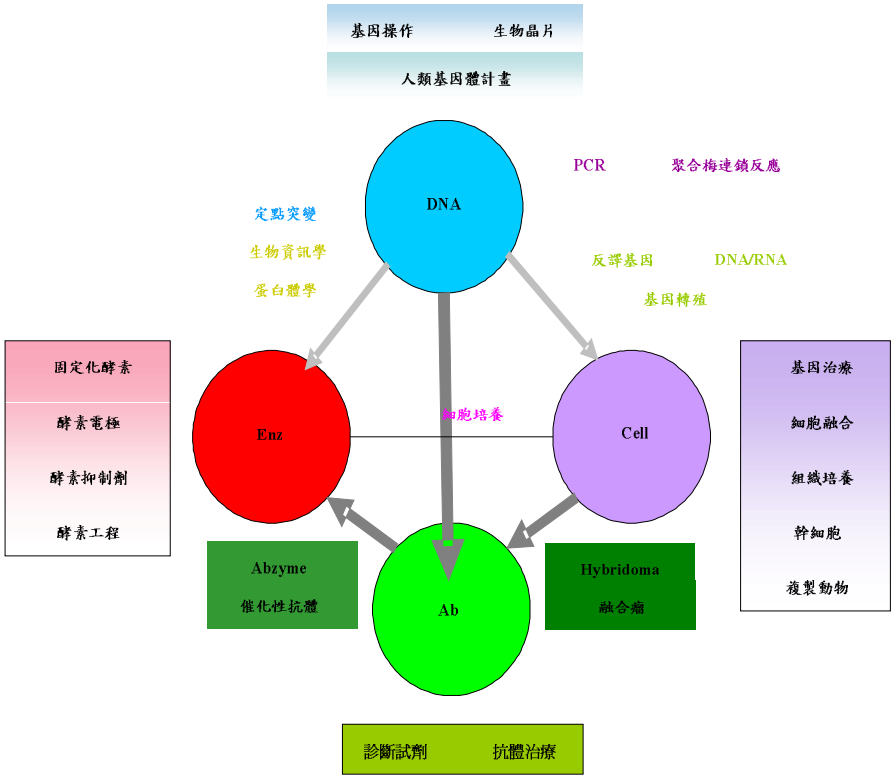


圖 1- 10 現代生物技術整體關聯圖

然而爲了要進行上述現代生物技術的研究，生物學家多半是靠生物以及醫學方面的實驗及觀察，但由於每次實驗的時間與金費會因實驗次數與變數的多寡快速增加。爲了解決這個問題，許多科學家便投入了大量的人力在解決這個問題。由於這個跨學科的領域跟計算機技術有關，因此我們稱為計算生物學。

計算生物學(Computational Biology)主要是研究生物學應用上具計算複雜度的問題，(Computational Molecular Biology)。而計算分子生物學的主要課題包括了序列組合、序列分析、生物資訊資料庫、基因認定、種族樹建構及蛋白質三維結構推測等，在此簡要地敘述如下：

1. 序列組合

由於以目前的技術是無法將人細胞中整個長度三十億的 DNA 序列一次讀出。所以，分子生物學家所用的方式是先將這序列分成一些較小的片段，然後再逐一兜成原來的整個序列，目的就是相要將各個生物體的完整序列拼湊出來。

2. 序列分析

在我們得到一些序列片段後，我們也希望能藉由序列間的比較分析來看看它們的相似程度、找出一些基因規則、或甚至於用來推測它們的演化關係，主要目的在於發現基因的變化是否有規則性可言，以及各物種間的演化相關程度的多寡

3. 生物資訊資料庫

由於愈來愈多的生物序列已被決定出來，以資料庫協助管理是最為有效的方式。其中，美國國家衛生研究院生物科技資訊中心(NCBI; National Center for Biotechnology Information)所支援的 GenBank 已廣被各實驗室所採用，GenBank 是一個儲存核酸序列及蛋白質序列的資料庫，它與英國的 EMBL(European Molecular Biology Laboratory)資料庫及日本的 DNA 資料庫互相合作，截至 1997 年 8 月止，它已儲存了 492,483 種不同的序列，總共有 353,713,490 個核酸字符。雖然其中有 54%是人類的核酸字符，但它也包含了超過 15,500 個不同種類的生物序列。也由於其龐大的資料量，因此有很多人探討如何有效地表示資料，以及如何有效地搜尋資料。

4. 基因認定

在人類三十億長的 DNA 序列中，約只有 3%是基因（所謂基因是指那些會轉換成蛋白質的 DNA 序列，我們人類約有五萬到十萬種基因），如何在 DNA 序列中決定基因所在位置仍是未解的問題，已有很多研究提供了一些有效的方法，但仍未能完全精確地預測出所有基因位置，倘若能正確的預測出基因位置，現今所多的疾病將可迎刃而解。

5. 蛋白質三維結構推測

蛋白質的很多特性與功能是和它實際的三維結構非常相關的，然而目前直接去決定某種蛋白質的結構，通常不是不可行就是代價太高，藉由一些方法的設計與協助，生物學家可以用較低的代價求得蛋白質可能的結構，然後再以實驗加以驗證。這些推測很多是基於最低熱能結構或蛋白質序列比較來進行的。

6. 演化樹建構

演化樹的建構是一門有悠久歷史的研究領域，近年來由於生物序列的協助，我們可藉由這種更精細的分析來建構那些較為模稜兩可的種族間之演化樹；同時，我們也可藉由這種細部分析來驗證以前所建構的演化樹。通常這方面的研究都會先以生物序列的比較來求得種族之間的兩兩距離，然後基於某些要件下，試著去建構一個最符合需求的演化樹。

序列分析在生物資訊的應用非常廣泛，在序列組合方面，它常被用來判斷是否要連接某些序列片段；在生物資訊資料庫上，它是搜尋工具的引擎，扮演著最關鍵的角色；對基因認定而言，它常用來協助決定那些區段是基因所在；而在種族樹的建構上，它可用來推測種族間的兩兩距離，以做為建構種族樹的基礎；在蛋白質三維結構的推測上，一維序列的分析有時也提供了不少的資訊。綜合以上所說，我們可以由圖 1-11 做一簡單說明：

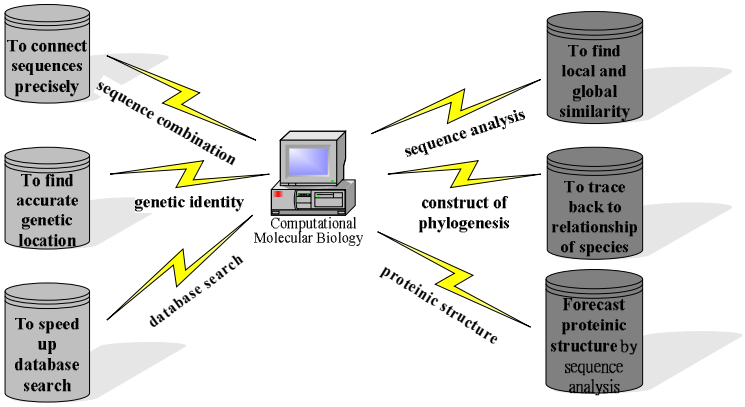


圖 1- 11 計算生物學應用領域與目的

1.2 研究動機

『工業工程』(IE)的目的是將人員、物料、設備、資訊和能源環境等整體系統近似設計、改良與規劃並探討效率與效能的一門學科。它利用數學的、自然科學的以及人文與社會科學的特殊知識與技能，並應用工程分析及設計的原理和方法，對於上述系統所可能得到的績效，予以規定(規劃)、預測和評估。工業工程(IE)起初為提高生產力、降低成本及改善品質之重要學科。創始於美國，其應用初以製造業為主，隨後普及工商業，舉凡行政機關、軍事單位、交通運輸、以及學校、醫院、公私立團體等，以為有效運用人力資源、財力分配，及其它相關資源之管理。此技術在美歐及日本等國普遍採用，我國近年來工商日益發達，工業結構改變與技術水準不斷提高，對外貿易歷年來高度成長，由於競爭日趨激烈展以及顧客要求亦日益嚴格，若干大中型廠商逐漸重視如何提高生產力方面的問題，工業工程對於工業管理之改進、工業技術之升級、以及設備自動化等，均可發揮輔助配合功效，於是工業工程之推展配合其顯著效果，乃漸露曙光。綜合上述，工業工程的目的可簡述如下:

1. 是工程與非工程領域的協調者。
2. 工程與科學管理間之『橋樑』。
3. 利用數學、自然科學以及人文與社會科學的特殊知識與技能，並應用工程分析及設計的原理和方法，對於系統進行最佳化、預測和評估。

基於上述理由與研究背景中所提到的問題不謀而合，因此非常適合工業工程人員進入生科領域來加速它的進步。而在生科領域中經評估，我們挑選計算生物學為最適合工業工程人員優先進入的領域其原因如下:

1. 不像其他領域需要非常多的生物知識。
2. 非常需要最佳化設計、預測和評估方面的人才。
3. 計算生物學是生科領域與其他領域間的橋樑，恰巧跟工業工程的理念相符。

由於計算生物學領域包函範圍非常廣泛，且序列分析在未來基因(Genome)的研究扮演很關鍵的角色，因此本研究將針對相似度序列分析進行研究。

1.3 研究目的

經上述介紹我們可知，還有許多問題等待不同領域的學者共同加入研究。其中序列排比又是生科研究的根本，因此若能直接從根本改進，相信將會對整個生科領域帶來不小的效益。因此本研究的方向就是利用將模糊導入序列比對中，讓得分函數模糊化，大幅降低蛋白質序列在極度不確定條件下所造成的影響。其目的如下：

1. 在專業生科研究人員進行實驗前，可利用更加優越的創新模糊架構去進行實驗設計，提高正確率。
2. 降低研究人員的操作時間與金錢。
3. 提高相似度分析的正確性與速度，加速疫苗的開發。
4. 利用創新的模糊懲罰函數與先進的演算法，試著將種族演化過程更加詳細精確的描述出來，並且也發展一套介面，提供研究人員然做初步的判斷。
5. 找出可能有害基因族群，將可使研究人員在判斷上更全面。

1.4 研究方向包括

有鑒於目前序列相似性問題在許多序列上仍然無法提高其相似度，其中最大的原因不外乎是不確定因素實在太高，除了使用的軟體與演算法不同會造成不一樣的相似度外，對於不夠了解這些生物基因的科學家們，便很難再突破。因此在這個最重要的前提下，我們試著導入模糊的概念來提出一個更適用的方法，以下為本研究之方向。

1. 我們將利用模糊架構與各種懲罰函數並從中比較找出各個函數的優缺點，進而整合出一個以模糊為基礎的全新理論架構。
2. 我們將整合出的模糊架構與啟發式演算法結合，預期將會縮短求解時間、提高正確率。

圖 1-12 為本研究論文架構：

1. 問題探討與確立目標

   探討傳統基因序列比對的處理方式，提出改善的方法。確立研究範圍與目標。

2. 文獻探討

藉由相關文獻的回顧，了解目前相關研究領域上的成果，作為研究過程的參考。

3. 組織架構與研究方法

實際建立模糊基因序列比對模型和其演算方法。

4. 模式實作與結果分析

以實際例子進行「模糊基因序列比對」演算與「傳統基因序列比對」演算，並將模擬所得數據進行分析，比較兩種方法的優缺點。

5. 結論與建議

將研究工作做一總結，並提出後續研究方向，以供日後研究參考。

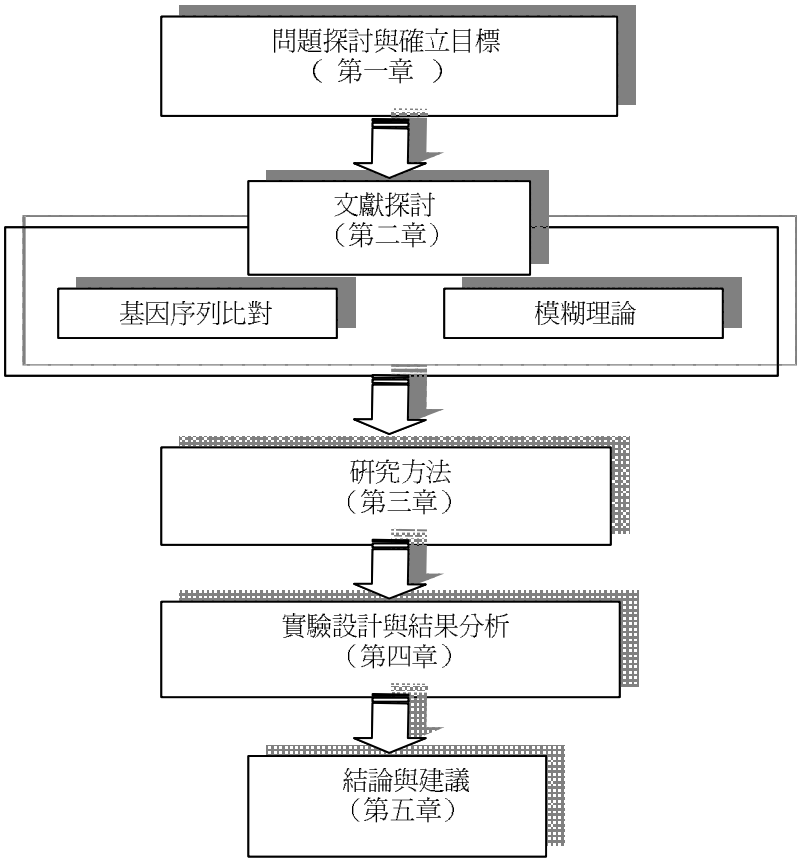


圖 1- 12 論文架構

第二章 文獻探討

2.1計算生物學相關文獻探討

爲了迎接 21 世紀這個基因世紀的來臨，基因功能的研究為當務之急，而要了解基因功能則需仰賴計算生物學。計算生物學除了能大幅降低生物學家進行實驗的成本外，對於社會更可以利用演化樹完成電腦病毒追溯、建立動植物的演化關係使下一代更了解地球的演化、重建基因網路使更多未知疾病的原因重現並加速疫苗製作與統計分析等等。另外還可用序列分析與蛋白質結構預測找出基因的正確位置與功能。

計算生物學的研究發展中各方向均有很多文章探討過。序列組合[44]、序列分析[6] [7] [8] [16] [17] [18] [19] [30]基因認定[42]、演化樹建構[24]蛋白質結構預測[9] [20] [23]其中在序列分析之探討方面又可分為 2 條序列[5] [6] [19]，與多條序列[3] [5] [51] [54]。2 條序列分析在 1970 年代，分子生物學家 Needleman 及 Wunsch [41]以動態程式設計方法(Dynamic Programming)技巧分析了氨基酸序列的相似程度；由於序列分析在未來基因組(Genome)的研究扮演很關鍵的角色，因此分析工具可說是五花八門[7]。目前有三種構想是較受大家所青睞的：第一種是 Smith-Waterman [46]的方法，這種方法很精細地計算兩序列間最好的 n 個區域排比(Local Alignment)，雖然這個方法很精確，但因耗時較久，所以多半應用在較短序列間的比較，然而，也有一些學者試著去改善它的一些計算複雜度，使它在長序列的比較上也有一些實際的應用。第二種是 Pearson 的 FASTA 方法[43]，這種方法先以較快方式找到一些有趣的區域，然後再以 Smith-Watermann [46]的方法應用在那些區域中。如此一來，它的計算速度就比 Smith-Waterman [46]快，而且在很多情況下，它的精細程度也不差。第三種是 Altschul 等人所製作的 BLAST [7]，它的最初版本完全沒有考慮間隔(gap)，所以在計算上比其他方式快了許多。雖然它不夠經細，但它的計算速度使得它在生物序列資料庫的搜尋上有很大的優勢，也因此它可說是目前最受歡迎的序列分析工具。此外，最近剛出爐的 Gapped BLAST [8]及 PowerBLAST [54]也已針對精細程度做了很大的改進，且在計算速度上仍維持相當的優勢。多條序列分析一直是計算生物學上很重要的課題，但由於問題複雜度，因此，在計算方法上有兩種不同的流派：一種是 Lipman 等人提出的方式[3] [5]，它們做的是同時比較多個序列，但試著去降低計算時所

用的動態程式設計矩陣點，據論文指出，這種方式比較 10 個長度為 200 的序列也不會遭遇太大的問題；另一種方式是 Feng 及 Doolittle [25]所採用的，它根據序列遠近程度的演化樹來做序列排比，一旦 gap 在某個比較中出現後，它就會被保留到最後，這種方法用來比較 k 個長度皆為 n 的序列所需時間較短，所以非常廣受歡迎。

2.2 序列分析相關文獻

數據庫搜索的基礎是序列的相似性比對，而尋找同源序列則是數據庫搜索的主要目的之一。所謂同源序列，簡單地說，是指從某一共同祖先經趨異進化而形成的不同序列。必須指出，相似性(similarity)和同源性(homology)是兩個完全不同的概念。相似性是指序列比對過程中用來描述檢測序列和目標序列之間相同DNA鹼基或氨基酸殘基順序所占比例的高低。當相似程度高於50%時，比較容易推測檢測序列和目標序列可能是同源序列；而當相似性程度低於20%時，就難以確定或者根本無法確定其是否具有同源性，不能把相似性和同源性混為一談。所謂“具有50%同源性”，或“這些序列高度同源”等說法，都是不確切的。

相似性概念的含義比較廣泛，除了上面提到的兩個序列之間相同鹼基或殘基所占比例外，在蛋白質序列比對中，有時也指兩個殘基是否具有相似的特性，如側鏈基團的大小、電荷性、親疏水性等。在序列比對中經常需要使用的氨基酸殘基相似性分數矩陣，也使用了相似性這一概念。此外，相似性概念還常常用於蛋白質空間結構和折疊方式的比較。

序列排比在分子生物學分析上，扮演了一個很重要的腳色。根據不同序列間的排比，可以幫助預測出未知的蛋白質功能或結構，若找出相似的部份越多，所能提供的資訊也就越多，因此兩條或多條以上的序列排比主要是盡可能找出最大的序列間的排比相似。兩條序列的排比我們通常稱為成對序列排比(pairwise)，而其他的則稱為多條序列排比(multiple sequence alignment) [32]，圖2-1成對序列排比的例子取自Thompson et al. [48]於1999年所發表的文章。

在圖2-1中，間格(gap, ‘-’)是用來插入在兩條序列間，目的是為了找兩條序列最大的相似範圍，為了解決這個問題，我們需要有計量的標準來衡量排比出的序列。一般最常用的方法是利用成對序列排比總和分數/成本(sum-of-pairs score/cost) 或突變機率矩陣(mutation probability matrix)。

1 -NLFVALYDfvasgdntlsitkGEKLRVLgynhn - - - - - - - - - -- -- - -gE

36 WCEAQt--kngqGWVPSNYITPVN- - -- - -

1 kGVIYALWDyepqnddelpmkeGDCMTIIhrede-- - -- - -- - -- - -- - deiE

39 WWWARl- -ndkeGYVPRNLLGLYP-- - - - -

圖 2- 1 成對序列排比範例

在1970年，Needleman and Wunsch [41]提出了一種演算法來解決2條蛋白質序列的排比。這個演算法是藉由計算找出排比最大的得分，而利用的衡量標準則是將所有排比對應的得分總和減去所有間格的懲罰分數總和。成對序列排比可以利用動態規劃(dynamic programming, DP)正確的解決。然而，由於排比問題屬於NP-hardness問題，因此最常使用啟發式的演算法來進行排比問題。從1970年到最近，許多的分析技術相繼被提出[7] [8] [16] [17] [18] [19] [30]。

有一些技術能用來替代像蛋白質的成本矩陣，例如 Dayhoff [21]的PAM(Point Accepted Mutation)矩陣，PAM 矩陣是用來描述胺基酸在不同演化距離間的取代機率，通常結合間格成本(gap cost)或權重來產生成本函數。一般來說，不同成本函數會造成不同的最佳排比。因此，除了可利用已知的成本函數發展演算法去搜尋最佳解或近似最佳解外，另一方面也可朝更接近真實生物學的替代矩陣發展[28]。由於矩陣的建立是利用實驗的方式找出胺基酸自然突變的比率，藉此來接近實際現象。因此本研究利用混合式演算法加上得分矩陣來找尋最佳解排比時，必定會包含一些不確定因素在內。因此本研究利用模糊集合與邏輯將得分矩陣內的每個元件模糊化。

模糊集合和邏輯最早是由 Zadeh [53]於 1965 年所提出用來解處理不確定環境的理論，尤其適用在複合系統(complex system)上。在模糊算數的領域裡，Zadeh 的擴張原則(extension principle)被廣泛的應用[15] [22] [40]近年來，有很多分子生物學上的文章都有應用模糊理論[10] [11] [29] [31] [42] [50]。在序列排比方面，Blankenbecler et al. (2003) [10]提出結構排比(structure -alignmen)的方法。這個問題所利用的成本函數包含模糊配對變數與自動配位結合。結構排比是根據 Needleman-Wunsch 的演算法，利用連續變數取代 2 元決策樹。

2.3 兩條序列的排比

在計算機上通常要解決兩條序列的排比是利用一種稱為"動態程式設計"(Dynamic programming)技術來設計的演算法。在做序列排比之前通常會定義一個算分數的方法稱為得分函數(score function)，用來定義序列排比時字元互相對齊的情況時所應得到的分數。例如兩個字元如果對齊且相同則得兩分，如果不同則得-1 分，如果對應到空白(俗稱開 gap)則得-1 分。這個 score function 並非絕對的，它能夠應映不同生物學家的需求定義不一樣的算分數方法。例如對連續開空白增加扣分，或則當對齊後最前端或做後端連續開的空白不予扣分。所以 score function 的做法被廣泛的應用在序列排比上。而常用的而兩條序列排比的問題定義如下：

定義一：有兩條序列 S1 及 S2，我們要找到一種序列排比(alignment)的情況，使得在這種情況下所得的分數是最高的。例如：假設兩條序列為 S1= "GCACAGC",S2= "CATGCG"，則利用動態程式設計的方法來解決兩條序列排比的作法如下：首先，先定義 score function ，假設我們在此定義相同得+2 分，不同得-1 分，對應到空白得-1 分。起始值 A(0,0)=0 代表 gap 和 gap 對應。A(i,0)= -i 代表 S1 和 gap 對齊。A(0,j)= -j 代表 gap 和 S2 對齊。接著我們即可列出如圖 2-1-1 的式子

$$A(i,j)=\max\begin{cases}\max\begin{cases}A(i-1,j-1)-1\\A(i-1,j)-1\\A(i,j-1)-1\end{cases} & if\ a_i\neq b_j\\ A(i-1,j-1)+2 & if\ a_i=b_j\end{cases} \tag{1}$$

再來我們建立一個矩陣用來儲存 A(i,j)的結果，如下圖 2-2，然後從左上角開始，依據我們剛才定義的 score function，來決定由那個格子下來，且分數為多少。而矩陣最右下角格子的值即是兩條序列做排比最高的分數。計算結果如下圖所示，得到如圖 2-3 的圖之後，再由最右下角倒推回左上角即可知最好的排比結果為

GCACAGC－
－CA－TGCG 和 GCACAGC－
－CAT－GCG 兩種結果

在相似序列分析上，最先是由 Needleman and Wunsch [41]於 1970 年代，利用最小替代成本的概念計算分數並結合動態規劃解決 2 條序列的排比問題，運用在多條序列排比卻是 NP-hardness。爲了解決這個問題，有些

學者利用定義搜尋解的上下界來減少不必要的搜尋空間且同時進行複合式序列排比，也有些學者延續這種想法，利用重新定義邊際權重或邊際成本來改善速度上述問題。其次是利用啟發式方法來加速求解，雖然解決了速度問題，但卻容易落入區域解造成正確性不足。另一派學者利用階段式的精鍊，可避免掉入區域解但卻造成速度慢的問題。也因如此，在計分方式上也有了不同的方式，致可分為 2 類:

	i	0	1	2	3	4	5	6	7
j			G	C	A	C	A	G	C
0		0	-1	-2	-3	-4	-5	-6	-7
1	C	-1							
2	A	-2							
3	T	-3							
4	G	-4							
5	C	-5							
6	G	-6							

圖 2- 2 初始矩陣

	i	0	1	2	3	4	5	6	7
j			G	C	A	C	A	G	C
0		0	-1	-2	-3	-4	-5	-6	-7
1	C	-1	-1	1	0	-1	-2	-3	-4
2	A	-2	-2	0	3	2	1	0	-1
3	T	-3	-3	-1	2	2	1	0	-1
4	G	-4	-1	-2	1	1	1	3	2
5	C	-5	-2	1	0	3	2	2	5
6	G	-6	-3	0	0	2	2	4	4

圖 2- 3 結果矩陣

1. 依賴 Dayhoff [21]於 1978 年所發展出的 PAM(Point Accepted Mutation)矩陣與 Henikoff and Henikoff [28]於 1992 年所發展出的 BLOSUM(Blocks Substitution Matrices)矩陣建立的特殊替代矩陣，加上 Altschu [3]於 1989 年所提出閥值懲罰(gap penalties)的成本，與 Altschul et al. [5]於 1989 年、Thompson et al. [47]於 1994 年提出的一些序列權重，其最佳解定義在最低的替代、插入或刪除成本。
2. 依據 Altschul and Erickson [4]於 1986 年提出成對權重加總(weighted sums of pairs)裡，利用反射閥值懲罰(affine gap penalties)加上一般的替代矩陣，通常統計分析被建立在大量排比(alignment)上。但這並不適用在序列中若有特殊(interseted)序列存在時，爲了解決這個問題，Gribskov [27]於 1987 年與 Krogh [34]於 1995 年 重新設計計分系統，此時允許設計特定序列的計分系統(take into account patterns of conservation)和獨特的替代(substitution characteristic of each position in the multiple alignment of a given family.)。

綜合上述研究，我們可以整理出以下幾點問題為本研究動機。

1. 利用明確值(crisp)與權重兩種方式並不符合實際情形。
2. 計算出的最高分數不一定是最好的相似度分析。
3. 不確定因素太多，利用實數計算無法滿足環境現制。
4. 由於環境變數太大，序列的界定太過主觀。
5. 無法同時兼顧速度與正確性。

為了要解決這些問題，本研究則是利用模糊矩陣的方式，期望藉由矩陣模糊化來降低不確定因子的影響。

2.4得分矩陣(score function)文獻

最初在計算序列間距離時都只有考慮到完全相同的殘基，相同的殘基配對得一分，這種計算序列距離的矩陣，稱為單位矩陣(圖 2-4)，這種矩陣的特性是它很「稀疏」，因為矩陣中大部分的組成份子都是 0，這同時也意味著這種矩陣缺乏辨別力。所以，為了提高辨別力，且在不增加背景雜訊的前提下，加入其它較弱但具有生物意義的胺基酸配對訊息。這個改變正是蛋白質序列分析的重點，我們必須在僅有數學意義的高分排比方式，及具有生物意義但分數較低的排比之間作出抉擇。

為了解決這位問題,科學家們設計出可以對相似(但不相同)的胺基酸鹼配對評分的矩陣基，而給分的標準是參考許多不同親疏遠進的蛋白質序列之間其胺基酸互相取代的比率(substitution rates)而定出來的矩陣。這跟DNA 序列為最不相同的一點。正確地使用種矩陣，可以提高對不同排比間差異的敏感度，尤其是在兩條序列間的全等的胺基酸配對很少時，其辨識力可以更清楚的突顯出來。

	A	C	G	T
A	1	0	0	0
C	0	1	0	0
G	0	0	1	0
T	0	0	0	1

圖 2- 4 單位矩陣(Unitary matrix)

因此在 1978 年，Dayhoff 利用基因間的遺傳距離，提出了 PAM [21]的得分矩陣。在序列比對中，通常希望使用能夠反映一個氨基酸發生改變的概率與兩個氨基酸隨機出現的概率的比值的矩陣。這些比值可以用相關機率（relatedness odds）矩陣表示。這就是突變數據相似性分數矩陣產生的基礎，在序列比對過程中，兩個序列從頭到尾逐個殘基進行比對，所得機率值的乘積就是整個比對的分值。在實際使用時，通常取機率值的對數以簡化運算。因此，常用的突變數據矩陣 PAM250 實際上是機率值的對數矩陣。矩陣中值大於 0 的元素所對應的兩個殘基之間發生突變的可能性較大，值小於 0 的元素所對應的兩個殘基之間發生突變的可能性較小。

突變數據矩陣 PAM 即可接受點突變(Point Accepted Mutation，簡稱 PAM)。1 個 PAM 的進化距離表示 100 個殘基中發生一個殘基突變的概率。對應於一個更大進化距離間隔的突變概率矩陣，可以通過對初始矩陣進行適當的數學處理得到[21]，如常用的 PAM250 矩陣，PAM250 相似性分數

矩陣相當於在兩個序列之間具有 20%的殘基匹配。主對角線上分數值是指兩個相同殘基之間的相似性分數值，有些殘基的分值較高，如色氨酸 W 為 17、半胱氨酸 C 為 12，說明它們比較保守，不易突變；有的殘基的分值較低，如絲氨酸 S、丙氨酸 A 兩種胺基酸均為 2，這些氨基酸則比較容易突變。不同氨基酸之間的分數值越高，它們之間的相似性越高，進化過程中容易發生互相突變，如苯丙氨酸 F 和酪氨酸 Y，它們之間的相似性分數值是 7。而相似性分數值為負數的氨基酸之間的相似性則較低，如甘氨酸和色氨酸之間為-7，它們在進化過程中不易發生互相突變。此外，表中把理化性質相似的氨基酸按組排列在一起，如鹼性氨基酸組氨酸 H、精氨酸 R 和賴氨酸 K。

在 Dayhoff [21]的文中提到，在蛋白質中，所謂被接受的突變點(accepted point mutation in protein)即為蛋白質中的某一個胺基酸利用自然的方式置換掉另一個胺基酸。產生這個結果時會有 2 個明顯的過程，第一，突變的發生會導致病變的基因模板，在蛋白質中產生一個胺基酸，第二，新的卓越突變形式將被接受。通常所產生的新胺基酸，功能與舊的相似時，突變的次數也會較頻繁。文中假設所有胺基酸的突變機率固定，並且利用實驗的方式，對大量且相似性高達 85%以上的樣本進行實驗。所利用的公式如下：

在非對角線的公式如下：

$$M_{ij} = \frac{\lambda m_i A_{ij}}{\sum_i A_{ij}}$$

A_{ij} : 可接受點突變個數

λ: 比例常數 (2)

μ_j : 第j個胺基酸的mutability

(is the mutability of the jth amino acid)如表

在對角線的公式如下：

$$M_{jj} = 1 - \lambda m_j \quad (3)$$

A_{ij} 為單點突變被接受的個數如圖 2-5，表格內的數字即為實驗時的突變次數，若以 A_{31} 為例，則代表胺基酸 Asn 突變成胺基酸 Ala 的次數。λ 則為一個常數，當 PAM 矩陣為 1 時，λ 則為 1，本例則以 1PAM 矩陣為例，因此 λ 值為 1。μ_j 則代表第 j 個胺基酸的關連性突變比例，若以圖 2-6 為例，被

排的序列分別為 ADA 與 ADB，因此在這兩條序列內的胺基酸種類有 A、D、B 三種，而以第一行來看，A 對應到 A，第二行則是 D 對 D，第三行則是 A 對 B，其中一、二行並無產生突變，第三行則是有胺基酸 A 突變成胺基酸 B，因此關聯性突變的算法以 A 為例，出現次數為 3 次，突變次數則為 1 次，因此關聯性突變比例為 1/3=0.33，若以 D 為例，出現次數為 2 次，突變次數 1 次，關聯性突變比例則為 0/2=0，若以 B 為例，出現次數為 1 次，突變次數 1 次，關聯性突變比例則為 1/2=0.5，圖 2-5 則是 1PAM 矩陣所求得的關聯性突變比例。在得到圖 2-5 與圖 2-7 的資料後經由公式計算，即可得到圖 2-8 的 1PAM 矩陣，以紅色箭頭所指的位置為例，設 λ 為 1，A_{ij} 為 109，$\sum_i A_{ij} = 30 + 109 + ...365 = 3634$，帶入公式 $M_{ij} = \frac{\lambda m_i A_{ij}}{\sum_i A_{ij}} = \frac{1 \times 109 \times 134}{3634} \doteq 4$

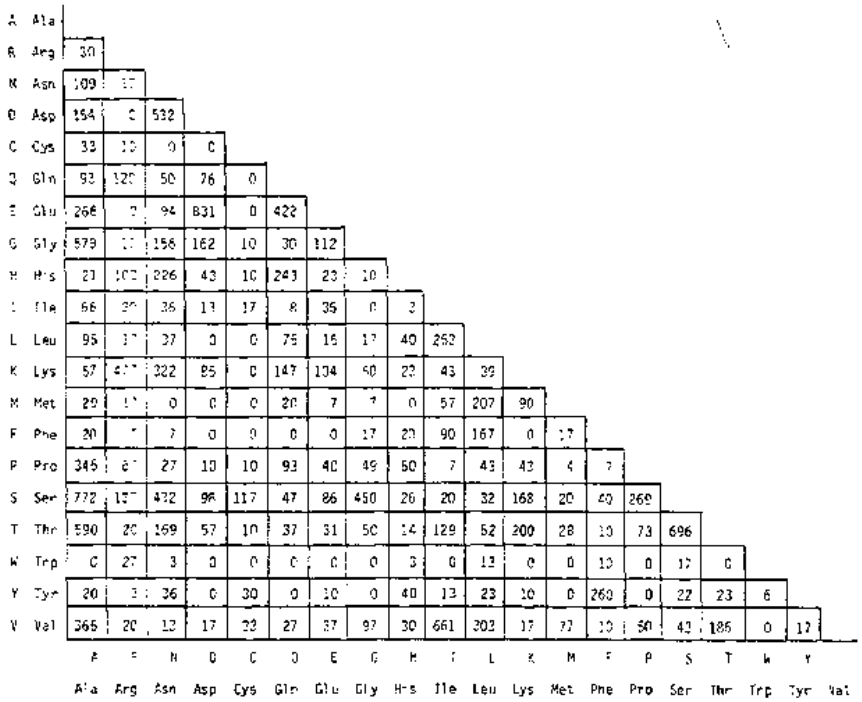


圖 2- 5 單點突變被接受的個數

Aligned sequences	A D A A D B		
Amino acids	A	B	D
Changes	1	1	0
Frequency of occurrence (total composition)	3	1	2
Relative mutability	.33	1	0

圖 2- 6 關聯性突變範例

Relative Mutabilities of the Amino Acids[a]

Asn	134	His	66
Ser	120	Arg	65
Asp	106	Lys	56
Glu	102	Pro	56
Ala	100	Gly	49
Thr	97	Tyr	41
Ile	96	Phe	41
Met	94	Leu	40
Gln	93	Cys	20
Val	74	Trp	18

[a]The value for Ala has been arbitrarily set at 100.

圖 2- 7 關聯性突變比例

$$\frac{1\times109\times134}{3634}\doteq4$$

ORIGINAL AMINO ACID

REPLACEMENT AMINO ACID	A Ala	R Arg	N Asn	D Asp	C Cys	Q Gln	E Glu	G Gly	H His	I Ile	L Leu	K Lys	M Met	F Phe	P Pro	S Ser	T Thr	W Trp	Y Tyr	V Val
A Ala	9867	2	9	10	3	8	17	21	2	6	4	2	6	2	22	35	32	0	2	18
R Arg	1	9913	1	0	1	10	0	0	10	3	1	19	4	1	4	6	1	8	0	1
N Asn	4	1	9822	36	0	4	6	6	21	3	1	13	0	1	2	20	9	1	4	1
D Asp	6	0	42	9859	0	6	53	6	4	1	0	3	0	0	1	5	3	0	0	1
C Cys	1	1	0	0	9973	0	0	0	1	1	0	0	0	0	1	5	1	0	3	2
Q Gln	3	9	4	5	0	9876	27	1	23	1	3	6	4	0	6	2	2	0	0	1
E Glu	10	0	7	56	0	35	9865	4	2	3	1	4	1	0	3	4	2	0	1	2
G Gly	21	1	12	11	1	3	7	9935	1	0	1	2	1	1	3	21	3	0	0	5
H His	1	8	18	3	1	20	1	0	9912	0	1	1	0	2	3	1	1	1	4	1
I Ile	2	2	3	1	2	1	2	0	0	9872	9	2	12	7	0	1	7	0	1	33
L Leu	3	1	3	0	0	6	1	1	4	22	9947	2	45	13	3	1	3	4	2	15
K Lys	2	37	25	6	0	12	7	2	2	4	1	9926	20	0	3	8	11	0	1	1
M Met	1	1	0	0	0	2	0	0	0	5	8	4	9874	1	0	1	2	0	0	4
F Phe	1	1	1	0	0	0	0	1	2	8	6	0	4	9946	0	2	1	3	28	0
P Pro	13	5	2	1	1	8	3	2	5	1	2	2	1	1	9926	12	4	0	0	2
S Ser	28	11	34	7	11	4	6	16	2	2	1	7	4	3	17	9840	38	5	2	2
T Thr	22	2	13	4	1	3	2	2	1	11	2	8	6	1	5	32	9871	0	2	9
W Trp	0	2	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	9976	1	0
Y Tyr	1	0	3	0	3	0	1	0	4	1	1	0	0	21	0	1	1	2	9945	1
V Val	13	2	1	1	3	2	2	3	3	57	11	1	17	1	3	2	10	0	2	9901

圖 2- 8 1PAM 矩陣

突變數據矩陣的產生基於相似性較高（通常為 85%以上）的序列比對，那些進化距離較遠的矩陣，如 PAM250(圖 2-9)是從初始模型中推算出來而不是直接計算得到的，其準確率受到一定限制。而序列分析的關鍵是檢測進化距離較遠的序列之間是否具有同源性，因此突變數據矩陣在實際使用時存在著一定的局限性。

C Cys	12																			
S Ser	0	2																		
T Thr	-2	1	3																	
P Pro	-3	1	0	6																
A Ala	-2	1	1	1	2															
G Gly	-3	1	0	-1	1	5														
N Asn	-4	1	0	-1	0	0	2													
D Asp	-5	0	0	-1	0	1	2	4												
E Glu	-5	0	0	-1	0	0	1	3	4											
Q Gln	-5	-1	-1	0	0	-1	1	2	2	4										
H His	-3	-1	-1	0	-1	-2	2	1	1	3	6									
R Arg	-4	0	-1	0	-2	-3	0	-1	-1	1	2	6								
K Lys	-5	0	0	-1	-1	-2	1	0	0	1	0	3	5							
M Met	-5	-2	-1	-2	-1	-3	-2	-3	-2	-1	-2	0	0	6						
I Ile	-2	-1	0	-2	-1	-3	-2	-2	-2	-2	-2	-2	-2	2	5					
L Leu	-6	-3	-2	-3	-2	-4	-3	-4	-3	-2	-2	-3	-3	4	2	6				
V Val	-2	-1	0	-1	0	-1	-2	-2	-2	-2	-2	-2	-2	2	4	2	4			
F Phe	-4	-3	-3	-5	-4	-5	-4	-6	-5	-5	-2	-4	-5	0	1	2	-1	9		
Y Tyr	0	-3	-3	-5	-3	-5	-2	-4	-4	-4	0	-4	-4	-2	-1	-1	-2	7	10	
W Trp	-8	-2	-5	-6	-6	-7	-4	-7	-7	-5	-3	2	-3	-4	-5	-2	-6	0	0	17
	C	S	T	P	A	G	N	D	E	Q	H	R	K	M	I	L	V	F	Y	W
	Cys	Ser	Thr	Pro	Ala	Gly	Asn	Asp	Glu	Gln	His	Arg	Lys	Met	Ile	Leu	Val	Phe	Tyr	Trp

圖 2- 9 PAM250 矩陣

於是 S. Henikoff 和 J.G. Henikoff 在 1992 年 11 月發展一種運算胺基酸序列排比時計算替換發生機率的 BLOSUM 矩陣[32]，可以更直接地表示序列間的遺傳距離。它是以比對蛋白質序列的 BLOCKS 資料庫為基礎，從其中擷取出來的序列推演而得。因此以 BLOcks SUbstitution Matrix 來命名。BLOSUM 矩陣法目標在於取代 Dayhoff 矩陣法(後者的優點在於用來比較 85%以上相似度的序列)，所以 S. Henikoff 和 J.G. Henikoff 計算在任何位置成對的胺基酸數目與全部期望出現成對的胺基酸數目的比率，結果以對數來表示。Henikoff 考慮到遺傳距離相近的蛋白質序列中，彼此間胺基酸的差異不大，因此在計算每一點的各種胺基酸比例時，在遺傳距離相近的蛋白質間共有的同一種胺基酸在其點上所佔的比例會偏高，而導致計算上的偏差。所以布洛森矩陣的方法是將各個序列片段依照最低相同比例(minimum percentage identity)給予分組，每組分別計算其中每個點各種胺基酸的比例，事實上就像在計算一條序列的每個點各種不同的胺基酸出現的機率一樣，這樣就可以修正因為某些蛋白質彼此間遺傳距離較近而造成的偏差。以下以簡單例子說明 BLOSUM 公式:

假設有一個行列為 1×10 的區塊，其中包含的殘基分別為 AAAAAAAAAS，因此會有 9 個 AA 配對(match)與 1 個 AS 錯誤配對(mismatch)，而這 9 個 AA 殘基則會有(8+7+6+…1)=36 種可能的 AA 配對、AS 或 SA 配對則會有 9 種，並且沒有任何的 SS 配對，個過程會重複出現在所有的區塊中，並將結果總和儲存在不同的表格(table)。我們可得知，所有配對組合會有(36+9+0)=45 種，因此可推得，假設有 w 個氨基酸 s 條序列，利用 ws(s-1)/2 可得到 1*10(10-1)/2=45 的結果。利用上述例子，我們可得到發生比率與期望比率，在將發生機率/期望發生機率以對數表示，發生比率的公式如下：

$q_{ij} = f_{ij} / \sum_{i=1}^{20}\sum_{j=1}^{i} f_{ij}$　$1 \le j \le i \le 20$，f_{ij} 為胺基酸 i 第 j 種配對的次數，若以上述為例，$f_{AA} = 36$，$f_{AS} = 9$，$q_{AA} = 36/45 = 0.8$，$q_{AS} = 9/45 = 0.2$。而期望機率的公式為 $p_i = q_{ii} + (\sum_{j\neq i} q_{ij}/2)$，$p_A = [36 + (9/2)]/45 = 0.9$，$p_S = (9/2)/45 = 0.1$。

則 AA 的期望發生機率為 0.9*0.9=0.81，AS+SA 的期望發生機率為 2*(0.9*0.1)=0.18，SS 則為 0.1*0.1=0.01，最後取對數後則得最後結果，以若對數值=0，表示觀察值=期望值，若對數值<0，表示觀察值<期望值，若對數值>0，表示觀察值>期望值。

當用不同的最低相同比例來對蛋白質序列分組時，就會得到不同的布洛森矩陣，用 62%最低相同比例做出來的就是布洛森 62 (BLOSUM62)矩陣，請參考圖 2-10，圖 2-11 所示，使用的布洛森矩陣值越大，辨識率就越高，另外還有布洛森 30 矩陣，布洛森 60 矩陣等以供使用，詳見附錄一。

	C	S	T	P	A	G	N	D	E	Q	H	R	K	M	I	L	V	F	Y	W	
		0	-1	1	0	2	1	1	2	1	2	0	0	2	4	1	5	1	2	-2	5	C
			2	0	-2	0	-1	0	0	0	1	0	0	0	1	0	1	-1	1	1	-1	S
C	9			2	-1	-1	-1	0	0	0	0	0	0	-1	0	-1	1	0	1	1	3	T
S	-1	4			2	-2	-1	-1	0	0	-1	-1	-1	1	1	0	-1	0	0	2	1	P
T	-1	1	5			2	-1	-2	-2	-1	0	0	1	1	0	0	1	0	1	1	2	A
P	-3	-1	-1	7			2	0	-1	-2	0	1	1	0	0	-1	0	-1	1	2	4	G
A	0	1	0	-1	4			3	-1	-1	0	0	1	-1	0	-1	0	-1	0	0	0	N
G	-3	0	-2	-2	0	6			2	-1	-1	-1	0	-1	0	0	0	0	2	1	3	D
N	-3	1	0	-2	-2	0	6			1	0	0	2	2	1	-1	0	0	2	2	4	E
D	-3	0	-1	-1	-2	-1	1	6			0	-2	0	1	1	-1	0	0	1	3	3	Q
E	-4	0	-1	-1	-1	-2	0	2	5			2	-1	0	1	0	-1	0	1	2	2	H
Q	-3	0	-1	-1	-1	-2	0	0	2	5			-1	-1	0	-1	1	0	1	3	-4	R
H	-3	-1	-2	-2	-2	-2	1	-1	0	0	8			1	-2	-1	1	1	2	3	1	K
R	-3	-1	-1	-2	-1	-2	0	-2	0	1	0	5			-2	-1	-1	0	1	2	4	M
K	-3	0	-1	-1	-1	-2	0	-1	1	1	-1	2	5			-1	1	0	0	1	3	I
M	-1	-1	-1	-2	-1	-3	-2	-3	-2	0	-2	-1	-1	5			-1	0	-1	1	2	L
I	-1	-2	-1	-3	-1	-4	-3	-3	-3	-3	-3	-3	-3	1	4			0	1	2	4	V
L	-1	-2	-1	-3	-1	-4	-3	-4	-3	-2	-3	-2	-2	2	2	4			-1	-2	1	F
V	-1	-2	0	-2	0	-3	-3	-3	-2	-2	-3	-3	-2	1	3	1	4			-1	2	Y
F	-2	-2	-2	-4	-2	-3	-3	-3	-3	-3	-1	-3	-3	0	0	0	-1	6			-1	W
Y	-2	-2	-2	-3	-2	-3	-2	-3	-2	-1	2	-2	-2	-1	-1	-1	-1	3	7			
W	-2	-3	-2	-4	-3	-2	-4	-4	-3	-2	-2	-3	-3	-1	-3	-2	-3	1	2	11		
	C	S	T	P	A	G	N	D	E	Q	H	R	K	M	I	L	V	F	Y	W	

圖 2- 10BLOSUM62 matrix

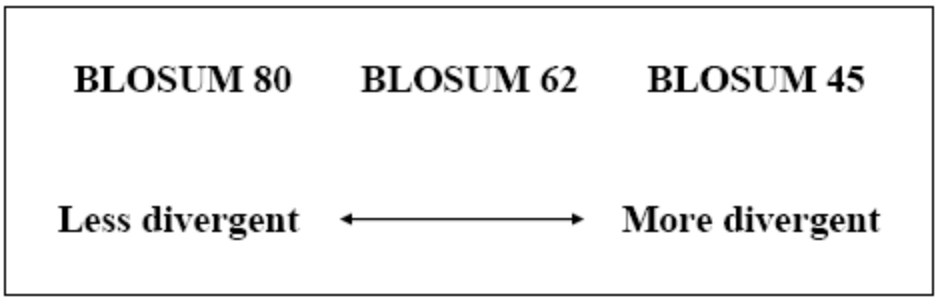


圖 2- 11 各種 BLOSUM 矩陣的關係圖

2.5間格懲罰函數與衡量標準

假如排比僅局限於將完全相同的序列排在一起，那麼很容易的可以寫出一個合理的程式來處理。但是，通常排比是要比對整段完整的序列，一個考慮完善的

排比程式必須將兩條完整的序列中全部的胺基酸殘基都列入考慮，這意思是說有時候必須將不相同的胺基酸配對在一起。在這個情形下，缺洞位置的安排就變的十分難以取捨。雖然可以將相同的胺基酸殘基配對在一起，而在不同的地方都插入缺洞，這樣的排比分數將是最高的，但卻沒有任何生物意義。另一種較好的方式就是每增加一個缺洞則在總分上扣分(penalty)。於是配合動態程式規劃的全面性或區域性排比演算法，在插入空白時則處罰扣分。在DNA序列排比時，因為鹼基的比對較為單純，不用考慮突變(取代)的因素，所以插入空白造成缺洞所扣的分數都是一樣的。但在蛋白質序列中胺基酸比對時，因為使用的評分矩陣的不同，所以插入空白所扣的分數也會有所不同，而常用的間格懲罰函數有兩種，在2002年，Hung等人所提出的文章中有提到[32]，一種為線性間格懲罰函數(Linear gap penalty)，其想法是其源於在不同的插入間格時會造成不同的結果，因此須給定一個懲罰函數，其公式為$f(G)=O*G$，其中G為間格數，O則為一個間格的開啟成本(Open cost)，而開啟成本可視不同需要做改變。另一種則叫做仿射性間格懲罰函數(affine gap penalty)，原理則是以線性間格懲罰函數為基礎，為了要達到不同間格應代表不同的懲罰，因此訂定了一個新的公式為$f(G)=O+(G-1)*E$，其中O與G所代表的意義與之前相同，E則代表延伸成本(Extend cost)，通常開啟成本要比延伸成本來的大。

通常在衡量序列的好壞，我們會以序列所得到的分數高低來作為指標，但有時最佳的序列並不一定出現在分數最高的序列上，因此在1999年，

Thompson[1][49]的文中提到另一種叫做Cloum Score(CS)計量指標，其公式如下：

$$CS = \sum_{i=1}^{M} C_i / M \tag{4}$$

其中M代表比對的行數(colum)，亦即比對中最常序列長度，C_i則代表第i行的殘基比對後是否相同，若為同一個殘基，則C_i=1，否之則為0，因此CS是利用2元分數(binary score)的一種指標

2.6模糊理論相關文獻

2.6.1模糊理論簡介

1965年美國加州大學柏克萊分校查德L.A.Zadeh [53]提出模糊理論，傳統科學都是以排除研究對象的模糊性為研究前提，模糊理論卻是承認存在研究對象內的模糊性為研究前提。現今科學所欲研究的對象之結構複雜性日益增加，人類的知識語言因人類本身的主觀意識、不同的時間、環境的變遷、研判事件的角度等等條件下，而具備模糊性，將使得科學家無法清楚研究對象的真實本質，以進而適當地建立假設的數學模式，所以模糊理論的想法應運而生，乃是參考人類思維方式對環境所使用模糊測度與分類原理，給予較為穩健的描述方法處理多元複雜的曖昧、不確定現象，總而言之，人類在知識中對不確定事件的處理，關於文字意義的不確定性，可以用模糊理論來處理。

2.6.2模糊集合(fuzzy set)

對通常集合論進行擴充之後而形成的。傳統數學的集合的基本定義，為屬於或不屬於此集合，即「非此即彼」，明確的判斷對象所屬之集合。欲適當的描述模糊事物，必須放棄傳統數學的集合的基本定義，將對象物所屬之集合的概念模糊化，即「亦此亦彼」，無明確的判斷對象物所屬之集合即「非此即彼」，承認論域上存在既非完全屬於集合A又非完全不屬於集合A，使傳統集合的絕對屬於的概念變更為相對屬於的概念。

模糊集合的主要特色，是將人類思維中不確定的、屬質的事物用隸屬度函數來表示。使生活上週遭的模糊情境與事物，能透過多元邏輯觀念與運算方法，進行數理分析。模糊集合泛指論域中具有特定性質的事物，且定義區域的界限並不分明，模糊集合論的主要特質，是在對於樣本的多重

可能偏好，做出更深入更精確的解說。

2.6.3模糊數與隸屬度函數

2.6.3.1 三角型隸屬度函數

假設$\tilde{A}$是模糊集合或模糊數（FN）在實數$\Re$上，則以三角型隸屬度函數（triangular membership function）表示為$(a_1, a_2, a_3), x \in \Re$ to $\tilde{A}$，如圖2-12。

$$\tilde{A}(x) = \begin{cases} 0 & x < a_1 \\ (x-a_1)/(a_2-a_1) & a_1 \le x \le a_2 \\ (a_3-x)/(a_3-a2) & a_2 \le x \le a_3 \\ 0 & x > a_3 \end{cases} \tag{5}$$

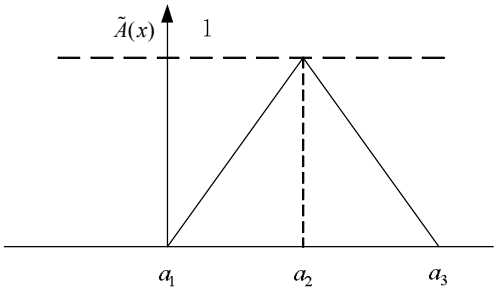


圖 2- 12 三角型隸屬度函數(a_1, a_2, a_3)

在一信賴區間α水準，$\alpha \in (0, 1]$，我們定義A_α：

$$A_\alpha = \left\{ x \middle| \tilde{A}(x) \ge \alpha \right\} \tag{6}$$

三角型隸屬度函數以α-截集表示為：

$$A_\alpha = [a_1^{(\alpha)}, a_2^{(\alpha)}] = [a_1 + \alpha(a_2 - a_1), a_3 - \alpha(a_3 - a_2)] \tag{7}$$

$a_1^{(\alpha)}$表示A_α的下界，而$a_2^{(\alpha)}$表示A_α的上界。

2.6.3.2 梯型隸屬度函數

假設$\tilde{A}$是模糊集合或模糊數（FN）在實數$\Re$上，則以梯型隸屬度函數（triangular membership function）表示為$(a_1, a_2, a_3, a_4), x \in \Re$ to $\tilde{A}$，如圖2-13。

$$\tilde{A}(x) = \begin{cases} 0 & x < a_1 \\ (x-a_1)/(a_2-a_1) & a_1 \le x \le a_2 \\ 1 & a_2 \le x \le a_3 \\ (a_4-x)/(a_4-a_3) & a_3 \le x \le a_4 \\ 0 & x > a_4 \end{cases} \tag{8}$$

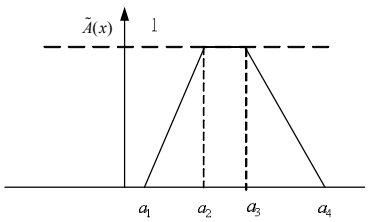


圖 2- 13 梯型隸屬度函數(a_1, a_2, a_3, a_4)

在一信賴區間α水準，$\alpha \in (0, 1]$，我們定義A_α：

$$A_\alpha = [a_1^{(\alpha)}, a_2^{(\alpha)}] = [a_1 + \alpha(a_1 - a_2), a_4 - \alpha(a_4 - a_3)] \tag{9}$$

2.6.3.3 α-截集（α-cut）模糊算術

α-截集是將模糊集合轉換為明確集合的工具，在模糊理論中佔有相當重要的地位，茲定義α-截集如下：

對一模糊集合A而言，若給定一實數值α，$\alpha \in [0, 1]$，則對模糊集合A取α-截集所形成的明確集合 $A_\alpha = \left\{ x \middle| \tilde{A}(x) \geq \alpha \right\}$。其中，$\alpha$被稱為信賴標準（Confidence Level）或門檻值（Threshold Value），當α值愈大，表示置信標準或門檻值愈高，則其所對應的區間就愈小；同理，當α值愈小，表示置信標準或門檻值愈低，則其所對應的區間就愈大，如圖2-14所表示。本文中三角型隸屬度函數和梯型隸屬度函數皆使用α-截集計算。

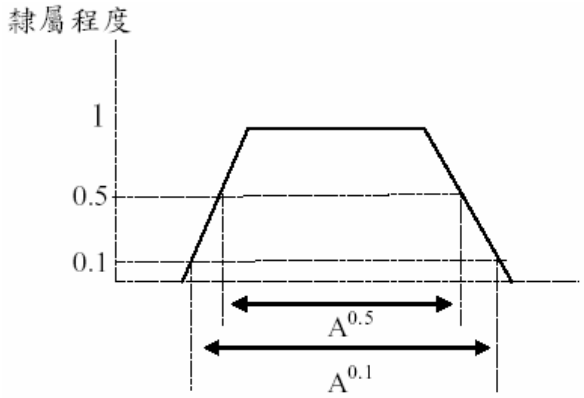


圖2- 14α-截集示意圖

根據模糊算術方法，Zadeh's sup-min方法表示如下

$$(\tilde{A} \circ \tilde{B})(z) = \sup_{x \circ y = z} \min(\tilde{A}(x), \tilde{B}(y)), \tag{10}$$

其中。表示任何模糊算術。Mizumoto and Tanaka [38]於（1976）使用相同方法在α-截集模糊數和區間算術，Chang [12]於（2003）整理使用α-截集模糊算術方法的文獻。

α-截集*模糊*算術中，模糊數基本運算公式（加法、減法、乘法、除法）可以計算的更快，在每一個α-level區間中，使用區間算術（Kaufmann and Gupta, 1988 [32]）。

1. 模糊數加法

令A和B分別為兩模糊數$\tilde{A}$和$\tilde{B}$，在一信賴區間α水準，$B_\alpha = [b_1^{(\alpha)}, b_2^{(\alpha)}]$, $\alpha \in (0, 1]$, $\forall \tilde{A}, \tilde{B} \subset \Re$，表示如下：

$$A_\alpha + B_\alpha = [a_1^{(\alpha)}, a_2^{(\alpha)}] + [b_1^{(\alpha)}, b_2^{(\alpha)}] = [a_1^{(\alpha)} + b_1^{(\alpha)}, a_2^{(\alpha)} + b_2^{(\alpha)}], \forall \alpha \in (0, 1]. \tag{11}$$

2. 模糊數減法

在一信賴區間α水準，$\forall \tilde{A}, \tilde{B} \subset \Re$ $\alpha \in (0, 1]$。

$$A_\alpha - B_\alpha = [a_1^{(\alpha)}, a_2^{(\alpha)}] - [b_1^{(\alpha)}, b_2^{(\alpha)}] = [a_1^{(\alpha)} - b_2^{(\alpha)}, a_2^{(\alpha)} - b_1^{(\alpha)}] \tag{12}$$

3. 模糊數乘法

在一信賴區間α水準，$\forall \tilde{A}, \tilde{B} \subset \Re$ $\alpha \in (0, 1]$。

$$\begin{aligned} A_\alpha \times B_\alpha = [a_1^{(\alpha)}, a_2^{(\alpha)}] \times [b_1^{(\alpha)}, b_2^{(\alpha)}] = [&\min(a_1^{(\alpha)} b_1^{(\alpha)}, a_1^{(\alpha)} b_2^{(\alpha)}, a_2^{(\alpha)} b_1^{(\alpha)}, a_2^{(\alpha)} b_2^{(\alpha)}), \\ &\max(a_1^{(\alpha)} b_1^{(\alpha)}, a_1^{(\alpha)} b_2^{(\alpha)}, a_2^{(\alpha)} b_1^{(\alpha)}, a_2^{(\alpha)} b_2^{(\alpha)})]. \end{aligned} \tag{13}$$

4. 模糊數除法

在一信賴區間α水準，$\forall \tilde{A}, \tilde{B} \subset \Re$ $\alpha \in (0, 1]$。

$$\begin{aligned} A_\alpha \div B_\alpha = [a_1^{(\alpha)}, a_2^{(\alpha)}] \div [b_1^{(\alpha)}, b_2^{(\alpha)}] = [&\min(a_1^{(\alpha)} / b_1^{(\alpha)}, a_1^{(\alpha)} / b_2^{(\alpha)}, a_2^{(\alpha)} / b_1^{(\alpha)}, a_2^{(\alpha)} / b_2^{(\alpha)}) \\ &\max(a_1^{(\alpha)} / b_1^{(\alpha)}, a_1^{(\alpha)} / b_2^{(\alpha)}, a_2^{(\alpha)} / b_1^{(\alpha)}, a_2^{(\alpha)} / b_2^{(\alpha)})], \text{ for } b_1^{(\alpha)}, b_2^{(\alpha)} > 0, \forall \alpha \in [0,1]. \end{aligned} \tag{14}$$

2.6.3.4 解模糊化（Defuzzification）

將模糊數轉換成一個明確值的過程即為「解模糊化」，解糢糊是將模糊集合$\tilde{A}$以最具代表性的一點表示，與隨機變數的平均值類似。解模糊化所使用之方法很多，本論文使用重心法（the center of area ，COA）解糢糊化。重心COA(x)公式表示如下：

$$COA(x) = \frac{\int_{\Re} \tilde{A}(x) x dx}{\int_{\Re} \tilde{A}(x) dx} \tag{15}$$

以往在進行序列分析時，研究人員多是利用得分模式與懲罰函數來進

行相似度分析。但面對序列這種充斥不確定因素的環境中，我們整理出幾個當利用明確值所面臨的問題：

1. 利用明確值與權重兩種方式並不符合實際情形。
2. 計算出的最高分數不一定是最好的相似度分析。
3. 不確定因素太多，利用實數計算無法滿足環境現制。
4. 由於環境變數太大，序列的界定太過明確。
5. 無法同時兼顧速度與正確性。

由於上述三點相似度分析的限制，更能顯示出若將其全部模糊化來重新建構相似度分析的潛在機會與優點。

2.7演算法介紹

基因演算法（Genetic Algorithm）是由密西根大學教授 John Holland 在 1975 年於 Adaption in Natural and Artifical System 文中所提出的一般性最佳化演算法則。理論基礎可回到自 1985 年達爾文（Charles Darwin）的「物種演化」（On the Origin of Species by Means of Nature Selection）書中的「物競天擇，適者生存」的演化及淘汰觀念。在此原則下，透過一些人工的運算，例如，複製、交配以及突變等方式對可能的解進行演化，並根據適應函數來評估，以求得最佳解或近似最佳解。

基因演算法是近年來發展快速以及具有潛力的最佳化方法之一，它類似傳統搜尋方法之漫步法（Random Walk Method）是全域搜尋法的一種。由於它是同時以多點方式搜尋最佳解，而非點對點的搜尋，對於多峰谷之函數而言，基因演算法較傳統演算法更可以較快找出整體最佳解（Global Optimum ），同時也能避免陷入區域最佳解（Local Optimum ）。此特性是基因演算法的最大優點。不過由於計算所需的時間過長、不易掌控，也是另一個缺陷。如何改善傳統基因演算法在最佳點附近收斂速度緩慢的缺點，並希望能快速求得全域最佳解，是目前 GA 最需加強的。Goldberg [26] 於(1989)說明了基因演算法不同於傳統的最佳化方法主要可歸納為下列幾項特性：

1. 基因演算法的運作是關於編碼後的參數，而不是參數的本身。因此可使適用的範圍更廣。
2. 基因演算法在搜尋的過程中是對整個母體做搜尋，也就是對多個點同時

做搜尋，而不是單點而已。因此能避免落入區域最佳解（Local Optimum）。

3. 基因演算法使用的是目標函數（Objective Function）的資訊，不需要微分或其他輔助的資訊。因此能使用在各種型態的適應函數，例如多目標或非線性函數等。

4. 基因演算法利用機率性（Probabilistic）方式來求解，而不是使用明確的（Deterministic）規則。這樣一來可使它更具彈性。基因演算法之運算機制一般基因演算法的運算流程可由圖 2-15 表示：

因此，若要以基因演算法來搜尋最佳解的問題時，首先需考慮以下要素：（1）決定參數編碼（Encode）方式；（2）產生初始族群，並決定族群大小（Population Size）；（3）根據問題的特性，設定適應函數（Fitness Function）；（4）執行基因運算子（Genetic Operator），產生子代（Offsprings）；（5）設定控制參數，如交配率、突變率以及停止運算的條件等。（6）進行解碼（Decode）的動作。以下將對上述步驟加以說明。

1. 編碼及解碼：基因演算法所要運作的對象通常是未能表示可行解的字串，而非決策變數本身。所以必須先預估每個參數的搜尋範圍，再將每個參數予以編碼，在求解適應函數值後，再將字串解碼成實際的變數。一般參數的編碼方式有二進位編碼（Binary Encoding）、實數編碼（Real Encoding）。而實數編碼則是將參數集合完全不變的來做運算。

   (1) 二進位編碼（Binary Encoding）：由於基因演算法在初始發展時就是以此來運算，因此二進位編碼是最常用且最普遍的編碼方式。在二進位編碼中，每一條染色體都是由實數參數編碼而成，並由 0 與 1 的字串所組成。表 2-1 為二進位編碼的例子：

   (2) 實數編碼（Real Encoding）：對於一些比較複雜或變數很多的問題，採用實數編碼的方式來求解可以比二進位編碼更有效率，在（c）裡我們將有更詳細的探討。表 2-2 為實數編碼的例子：

   (3) 兩者的差異：兩者的差異在Michalewicz [36]於(1992)的書中提到：傳統的基因演算法所使用的表示法是以二進位（Binary）為主，但

當遇到多維度（Multidimensional）或者需要高精確度（High-precision）的問題時就會有一些缺點存在。例如，對於一個擁有 100 個變數，範圍介於[-500,500]之間且需要精確到小數點第六位以上的問題時，以二進位的表示法來說就需要長度為 3000 的字串，對於這樣的問題，以進位為主的基因演算法可能就無法有效的進行運算，且需浪費許多的計算時間。

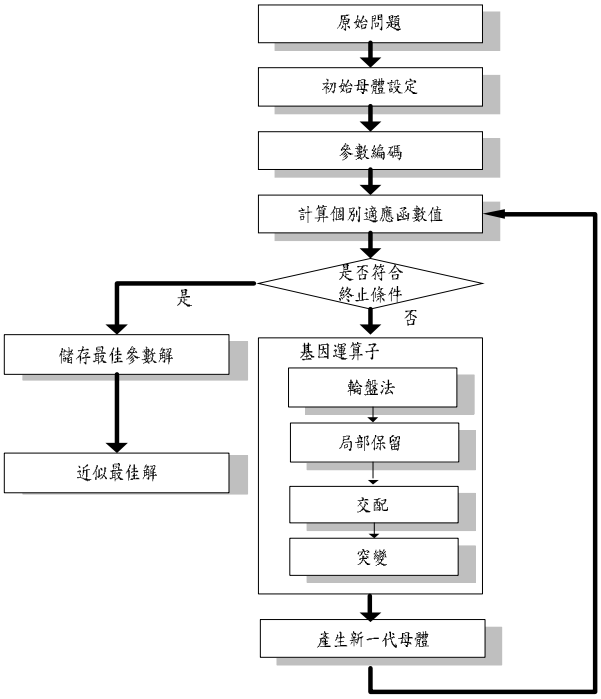


圖 2- 15 基因演算法之運算流程圖

表 2- 1 以二進位編碼為例的染色體

染色體 A	10110011100101011100101
染色體 B	11111110000011000001111

在連續的區間裡要找出全域最佳解（Global Optimum）對基因演算法是一大挑戰。傳統上使用二進位表示法的基因演算法最後是將真實的設計空間給分離。雖然這樣以二進位為編碼方式的基因演算法已經很成功的運用

在最佳化的問題上，但他仍然有些缺點：也就是當運用在真實世界裡的問題時，這些問題通常都會有很多的設計變數。另一個缺點是當運用在連續空間裡的參數最佳化問題時，二進位的表示法就會與真實問題的空間有所差異。例如，在真實空間裡很接近的兩個點可能在以二進位為表示法的空間裡就顯的離得很遠。

表 2- 2 以實數編碼為例的染色體

染色體 A	1.234　5.3243　0.4556　2.3293　2.4545
染色體 B	ABDJEIFDJYDSABXDSTRBVDMD
染色體 C	（back），（back），（right），（forward），（left）

對於這樣的問題，有一種簡單的表示法就是浮點（Floating-point）表示法。在這樣以實數為編碼（Real-coded）的基因演算法裡，個體相對於設計變數是以實數來表示；因為浮點表示法接近真實的設計空間，所以這樣的基因演算法是結實的（Robust）、準確的（Accurate）和有效的（Efficient），而且，字串的長度相對於二進位表示法減少了許多。不過，並非實數編碼的方式一定比二進位來的好，因為對於不同的問題必須常常發展一些新的交配（Crossover）或突變（Mutation）方式。

2. 初始母體的產生：在產生初始母體前必須先決定母體的大小，如果數目太大的話，會耗費計算時間，而太小則有可能太早收斂。母體產生的方式是以隨機的方式產生或是以啟發式解產生。
3. 適應函數：應用基因演算法在求解最佳化問題之前，需將所遇到的問題轉換為適應函數，而適應函數代表著系統對外在環境的適應能力。適應函數值越高，表示該染色體具有較優的特質，將來被複製（Reproduction）或選取（Selection）的機率就越大；反之，則越容易被淘汰。適應函數的設定會影響最佳解的好壞，若設定的理想，結果就好；反之，若設定的不好，則結果就會有偏差。

(1) 複製（Reproduction）或選取（Selection）；根據每個個體的的適應程

度，來決定個體被複製或選取的機率。因此，擁有較高適應值的染色體就有較高的機率被選出來進行複製。也因如此，適應函數值較低的個體，也會漸漸的消失被取代掉。目前常用的複製技術有兩種：(i) 輪盤法（Roulette Wheel)：在輪盤上依據適應函數值的大小來劃分區域面積，也就是與適應函數值成正比。(ii) 等機率法：每個個體被複製的機率皆相同。

(2) 交配（Crossover）

a 二進位表示法（Binary Representation)：此過程是隨機選取複製後的兩個母體，藉由彼此交換基因來產生新的兩個個體。交配過程發生的機率由交配機率所控制。基本上，常見的交配方式有三種形式：

(a) 單點交配：在所選取的兩個個體內，隨機選取一交配點後，將這兩個交配點後的基因全部交換。如圖 2-16 所示：

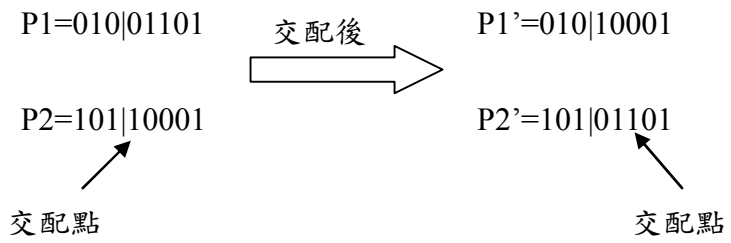


圖 2- 16 單點交配

(b) 雙點交配：在所選取的兩個個體內，隨機選取二個交配點後，將這兩個交配點內的基因全部交換。如下所示：

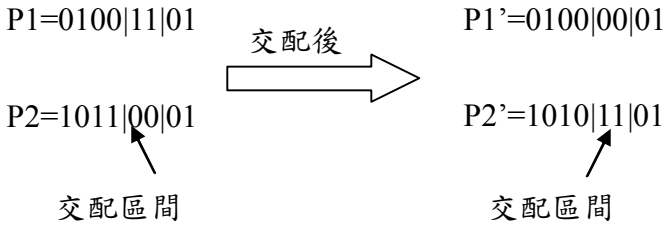


圖 2- 17 雙點交配

(c) 字罩交配：先產生一與染色體長度相同的字串，即稱為字罩。字罩由 0 與 1 隨機產生的字串組合，當字罩位元為 1 時，則兩個染色體於字罩為 1 的位置互相交換，以形成新的染色體。如下所示。

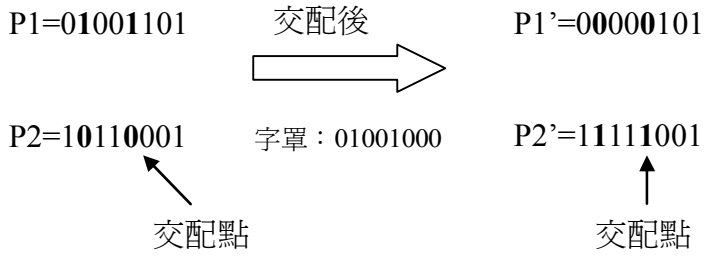


圖 2- 18 字罩交配

b 實數表示法：除了上述的交配方式外，也有很多的文獻在討論交配的方式，包括多點交配等等。由於二進位的編碼方式無法滿足現實生活中所需解決的問題，且無法保證子代一定會比母代好，因此以實數為編碼方式的基因演算法因而產生。以方向為主（Direction-based）的運算子就是為了改善交配過程的演化，使子代能比母代更好(Michalewicz, 1994 [37])。以方向為主的交配運算子使用目標函數的值來決定搜尋的方向。假設有兩個母代 B_1和B_2 產生子代 B'，則 $B' = r \cdot (B_2 - B_1) + B_2$ 其中 r 是介於 1 和 0 的隨機變數，且假設 B_2 不比 B_1 差，也就是說對於最大化問題，$fitness(B_2) \geq fitness(B_1)$。Murata(1994) [39]另提出 10 種不同的交配方式對於流程式排程問題進行電腦模擬測試。

(3) 突變（Mutation）

a 二進位表示法：在演算的過程中，程式會隨機產生一個突變的機率值，若此值比原先定義突變率低，則染色體會進行突變的程序。所謂的突變，是隨機選取染色體上的某一基因，將此基因做 0 或 1 的交換。舉例來說，假設預設的突變率為 0.01，若產生的亂數小於 0.01 則進行突變，否則繼續檢查下一字元。如圖 2-19 所示，僅第 3 個位元產生的亂數小於 0.01，因此將它變更。此目的是避免過早收

斂，以跳脫區域解。

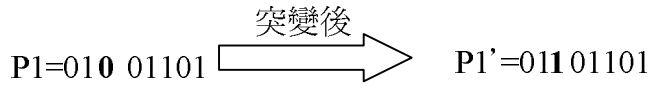


圖 2- 19 基因突變

b 實數表示法：假設 a_i 和 b_i 是每個變數的上下界，$\overline{X}$ 和 $\overline{Y}$ 是其中的兩條染色體，且有 n 個變數，則 $\overline{X}=(x_1,x_2,.....,x_i,....x_n)$，$\overline{Y}=(y_1,y_2,.....,y_i,.....y_n)$

(a) 均勻突變（Uniform Mutation）：隨機選擇一變數 j，使它屬於均一分配（Uniform Distribution）且介於 a_i 和 b_i 之間，也就是 $U(a_i,b_i)$。

$$x_i'=\begin{cases}U(a_i,b_i), & if\ i=j\\ x_i, & otherwise\end{cases} \tag{16}$$

(b) 界線突變（Boundary Mutation）：隨機選擇一變數 j，使它等於上界或下界。

$$x_i'=\begin{cases}a_i, & if\ i=j, r<0.5\\ b_i, & if\ i=j, r\geq 0.5\\ x_i, & otherwise\end{cases} \tag{17}$$

其中 r=U（0,1）。

(c) 非均勻突變（Non-uniform Mutation）：隨機設定一變數 j，使它等於非均勻的隨機變數。

$$x_i'=\begin{cases}x_i+(b_i-x_i)f(G), & if\ \ r_1<0.5\\ x_i-(x_i+a_i)f(G), & if\ \ r_1\geq 0.5\\ x_i, & otherwise\end{cases} \tag{18}$$

其中

$$f(G) = (r_2(1 - \frac{G}{G_{\max}}))^b$$

r_1, r_2 = *a uniform random number between* (0,1)

G = *the current generation*

$G_{\max}$ = *the* max *imum number of generations*

b = *a shape parameter* (19)

(d) 多點非均勻突變（Multi-non-uniform Mutation）：將母代 $\overline{X}$ 中所有的變數做非均勻突變。(Michalewicz, 1992 [36])

4. 終止條件：因演算法在正常的運作下，如何使其停止演化，常用的方式有下列幾種方法：

(1)設定演化代數：我們可以在程式開始執行之前就先設定演化代數，當達到設定值後，則停止演化。至於應該設定幾代，則需視問題複雜度以及資料量而定。

(2)設定演化時間：如（1）所言，先給予演化的時間，時間到了則停止。不過（1）和（2）並無法得知是否真的以達到最佳解。

(3)當最佳解經幾個世代的演化後，並無明顯的差異時，則可視為找到最佳解。

Steven 於 1996 年提出將基因演算法應用於多條序列排比上，但當時並未利用編碼的方式來進行求解，而 Hung 等人[32]於 2002 年的文章中提出，將平行式基因演算法導入到多條序列排比中，且利用行進方向做為編碼的依據來進行求解，以下將對 Hung 所提出的平行式混合基因演算法中的參數編碼、交配、突變加以說明。

1. 參數編碼：以圖 2-20 為例，(a)表示排比序列為 KPDN 與 KDM，由(b)可知，分別將往橫軸移動的路徑編為 1，往縱軸移動的路徑編為 2，往斜邊移動的路徑則編為 3，再由圖 2-21 中可得知，假設初始解為(a)，利用座標的方式則可表示成(b)，而(c)則為編碼後排比序列。
2. 交配：圖 2-22 中，(a),(b)皆為隨機選出，在利用隨機的方式找出交配點 P，因此子代將會從父代-1 與父代-2 得到 31*31，而*的位置則以隨機的方式產生，若*=2，則子代將為 31231。

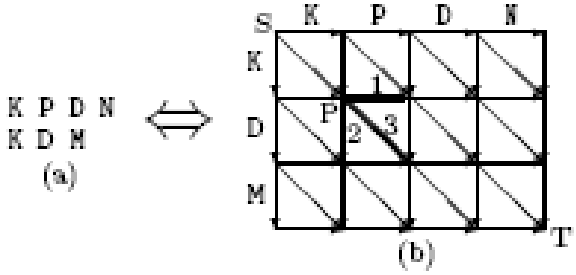


圖 2- 20 編碼說明

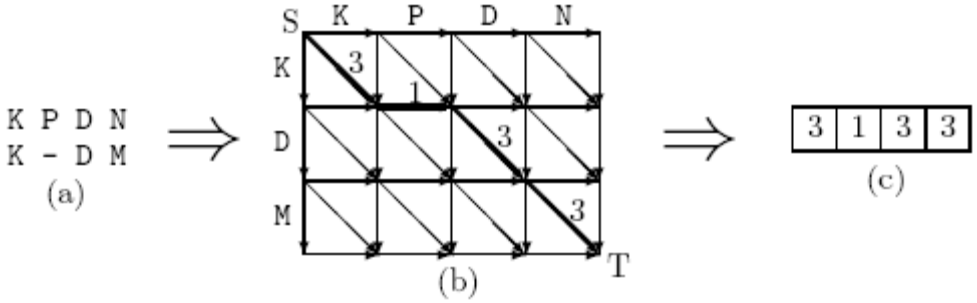


圖 2- 21 參數編碼

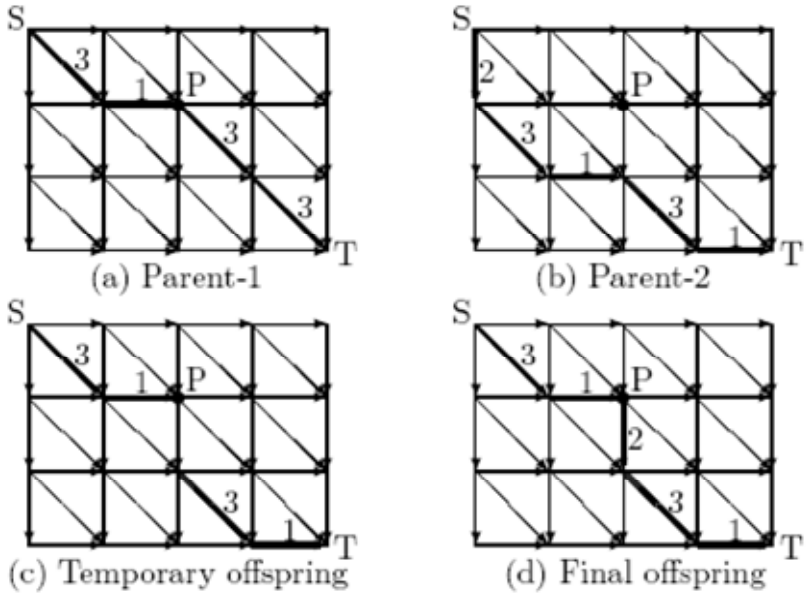


圖 2- 22 交配

3. 突變：由圖 2-23 可知，首先將(a)中父代的路徑上，選取 P、Q2 點作為突變的片段，在經由隨機的方式產生 P、Q 之間的路徑，最後可得(b)中的編碼為 3313。

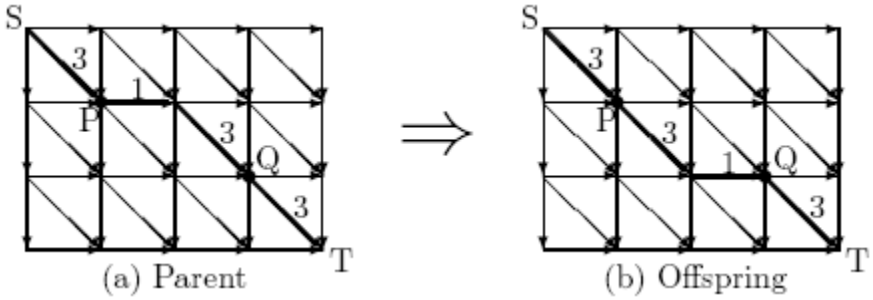


圖 2- 23 突變

禁制搜尋法(Tabu Search, TS)為一高階的萬用啟發式解法，專門用來求解組合最佳化的問題其流程如圖 3-4 所示。其以求得初始解出發，再以搜尋鄰近解或符合渴望準則的鄰近解作為下次搜尋的依據，並且記錄曾經搜尋的路徑，以避免陷入區域最佳解之中。禁制搜尋法中主要包含移步（Move）、禁制名單(Tabu List)、渴望準則(Aspiration Level)、候選名單(Candidate List)及搜尋停止準則(Stopping Criterion)五種組成要素，以下為禁制搜尋法的說明說明。

(1) 移步：每個解都會存在著相對應的鄰近解，而將目前的解轉換成其中一個最好的鄰近解的步驟則稱為移步。求解相關問題時，通常藉由隨機或是其他方式產生一起始解，再藉由各種移步方式搜尋鄰近解，並選擇最好的鄰近解做移步。

(2) 候選名單：記錄在搜尋過程中所找到的可行移步的屬性，禁制搜尋過程中所做的移步，即是從候選名單中挑選最佳的移步。

(3) 禁制名單：在移步過程中，記錄前幾次的移步，以避免求解過程中可能發生重複相同的路徑或倒退到前一個解，而導致循環搜尋，因此較不容易陷入區域最佳解。而禁制串列記錄的法則是採先進先出法，亦即一個新的移步被列入禁制名單時則把串列中最舊的移步移去，只存放最近幾次的記錄。

(4) 渴望準則：由於禁制名單的設計可能使得搜尋時錯過找到最佳解的機會，因此，禁制限制機制不能單獨存在，必須配合渴望準則的機制使得處於禁制狀態的移步有第二次被選到機會，亦即當某一移步處於禁制狀態時，但其花費的成本小於目前所搜尋到的最佳解時，將可以

透過渴望準則解除其禁制的狀態，並選擇此移步。

(5) 搜尋停止準則：包括最大允許運算的遞迴數、預設目標函數值持續未改善次數、達到預設可接受之目標函數值範圍、及最大允許CPU 計算時間等四種。

以下為禁忌搜尋法流程架構圖：

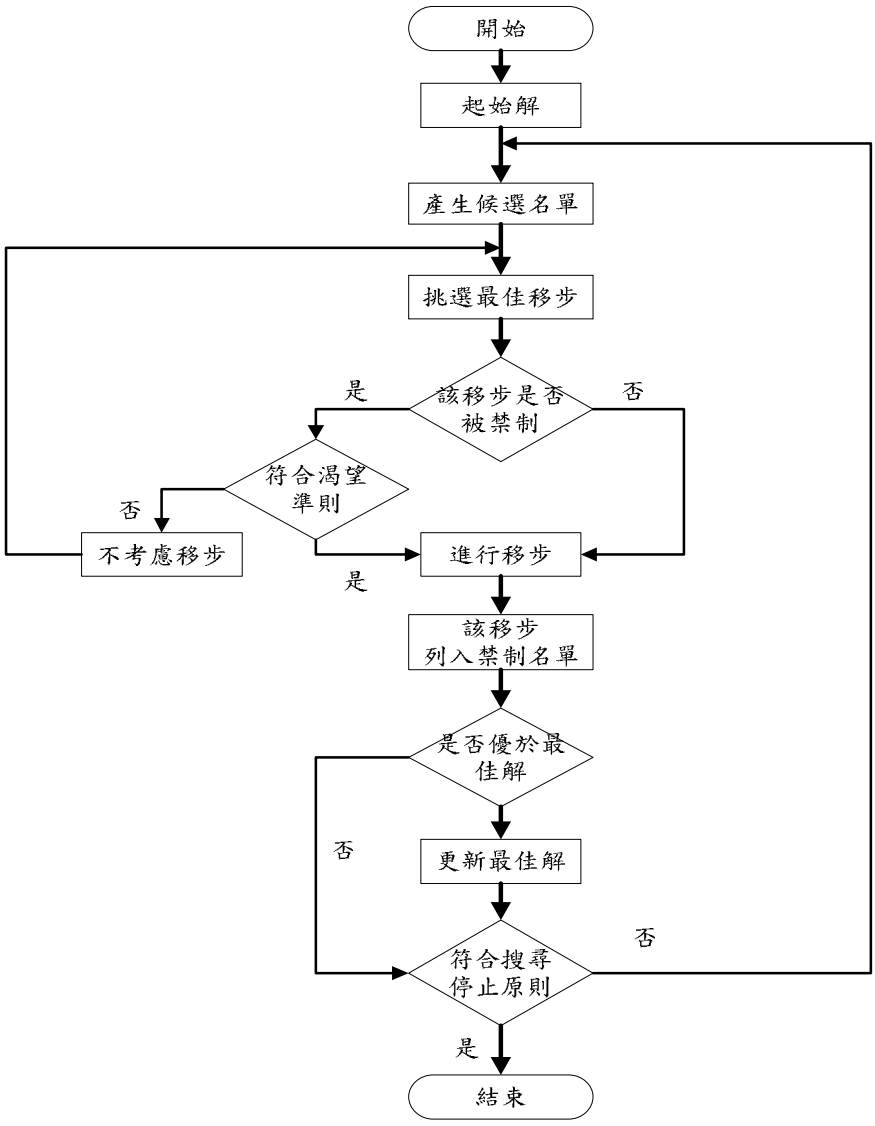


圖 2- 24 禁忌搜尋法流程架構

Raiz 等人[45]在 2005 年將 Tabu 演算法應用於多條基因序列排比，其中所使用的移步策略有兩種，分別為單一序列移動於區塊移動，以下將對兩種方式分別加以說明：

1. 單一序列移動：單一序列移動是利用隨機的方式找尋間格數目來進行移步的動作，且所挑選的間格數最小值為 1。移步的位置也是利用隨機的方式取得，由圖 2-25(a)可得知，在第一條序列中，隨機挑選其中的 2 個間格來進行移步，而移步的位置則是往右移動 3 個位置。
2. 區塊移動：間格移動時，包含了一條以上的序列，因此區塊的寬度最少可為 1，但區塊的深度至少須為 2，若深度為 1 時，則與單一序列移動方式相同，以圖 2-25(b)為例，所選擇的間格數為 2，深度則為 3。

圖 2- 25(a)單一序列移動, (b)區塊移動

第三章 研究方法

3.1 系統架構原理

序列比對的基本思想，是找出檢測序列和目標序列的相似性。比對過程中需要在檢測序列或目標序列中引入空位，以表示插入或刪除（圖 3-1）。序列比對的最終實現，必須依賴於某個數學模型。不同的模型，可以從不同角度反映序列的特性，如結構、進化關系等。很難斷定，一個模型一定比另一個模型好，也不能說某個比對結果一定正確或一定錯誤，而只能說它們從某個角度反映了序列的生物學特性。此外，模型參數的不同，也可能導致比對結果的不同。

```
Query: 179 ENGFRYIPFRIY------------QTTTER--------PFIQKLFRPVAADGQLHTLGDL 218
             F+ IP RIY              T +R         F  ++    A  G   T
Sbjct: 181 LESFKNIPLRIYTDDVRLHVHPETDFTDQRGRTKEEFGRFNGRIIDTCAQSGSFGTRIGA 240
```

圖 3- 1 序列比對，圖中“–”表示插入和刪除，用字符表示相同的殘基，“+”表示相似殘基。

序列比對的數學模型大體可以分為兩類，一類從全長序列出發，考慮序列的整體相似性，即整體比對；第二類考慮序列部分區域的相似性，即局部比對。局部相似性比對的生物學基礎是蛋白質往往是由較短的序列片段組成的，這些部位的序列具有相當大的保守性，盡管在序列的其它部位可能有插入、刪除或突變。此時，局部相似性比對往往比整體比對具有更高的靈敏度，其結果更具生物學意義。

有鑒於目前序列相似性問題在許多序列上仍然無法提高其相似度，其中最大的原因除了演算法的發展不足外，另一個重點就是得分矩陣的發展。本研究則是以找出最長的局部相似為出發點，將 PAM 與 BLOSUM 矩陣模糊化，並利用仿射性間格懲罰函數與混合式基因演算法來進行兩條序列的排比，預期能達到上述要求。

首先將每個基因序列所代表的數值模糊化，目的在減少不確定性環境的影響，期望在正確性與求解時間中找尋一個平衡點。

其次再將模糊後的矩陣配合仿射性間格懲罰函數與混合式基因演算法求解，在每次求解後都會進行相似度的分析，若相似度不足時即表示所求的解不夠正確，因此我們在重新回到演算法中計算，如此重複直到相似度達到設定值或遞回次數到達預定次數。

在這個系統中，另一個重點就是生物研究人員的額外資訊，由於序列的不確定性，因此我們利用模糊概念，但若有生物研究人員的額外資訊時，我們會將其加入在得分函數或其他系統架構中。

最後利用模糊基因序列加懲罰函數架構，利用傳統 GA、Tabu 及混合式 GA 演算法整合導入，進而創造出一個以模糊為基礎的理論架構，最終目的在找出最長的局部相似、加基因序列正確性及方法的適用性。

3.2 系統發展

全域序列排比(global sequence alignment)：所謂的全域序列比對就是將兩段長度為 n 及 m 的序列做比對，假設 m＝n，經過調整比對後，取得全域整體序列比對最佳的結果，換言之，比對結果之長度必定大於或等於 m 值；1970 年，Needleman 及 Wunsch [41]提出應用動態規劃（dynamic programming）及遞迴參數（recursive argument）來進行計分；對應突變的三種現象：替代（substitution）、插入（insert）、刪除（delete）。

區域序列排比(local sequence alignment)：在全域序列比對中，我們必須找出兩序列完整的比對結果，但生物序列長度大多都非常長，生物學家需要其中相似部分來做研究，且兩序列中必定可能包含相似及不相似的部分，不相似部份即可能造成扣分而影響到相似部份之呈現，故全域序列比對變不符使用,於是在 1981 年便由 Smith 及 Waterman [46]兩位發展出最初的局部序列比對，就是找尋兩序列中最為相似的子序列，使其比對分數獲得最高分，其作法與全域比對極為相似，其差別在於局部比對、刪除比對果為負分的片段，換言之，在動態規劃中，凡出現負分者，皆以零分取代之，此舉用意在，當序列在比對時，比對分數一但小於零，及判斷此片段為及不相似，故將其以零分取代，再開始找尋另一段相似片段，在計分完成後，便可在動態規劃矩陣中找尋最高分者，及為相似之子序列比對分數。

其實，2 大類序列相似度比較之下，其中還可分割；一般而言有下列幾種形式。

長序列分析：在分析大型序列時，最關鍵的因素就是求解時間。

一般序列長度分析：雖然求解時間不像大型序列那麼久，但中長度序列分析必須同時注重時間與正確性。

短序列分析：最重要的地方是在進行相似度時如何將其擴展到多條序列(multiple sequences)的比較。

本研究假設，所有生物序列資料的取得，皆是正確無誤且計算時間在合理等待範圍內，若求解時間太長則須調整相似度或序列的複雜性。

在限制條件方面，因為本系統首先運用在單機板上，因此在序列取得上只能擷取特定種類的基因序列群來進行實驗分析。所以，此系統所存在的限制很多，整體來說系統中包括的以下限制：

1. 序列取得的正確性
2. 同類序列蒐集的完整性
3. 硬體設備的運算速度
4. 生物知識的了解程度

在研究進行方面，本研究將依圖 3-2 所列之系統架構圖細分成下列各階段進行系統建構及運算機制設計。

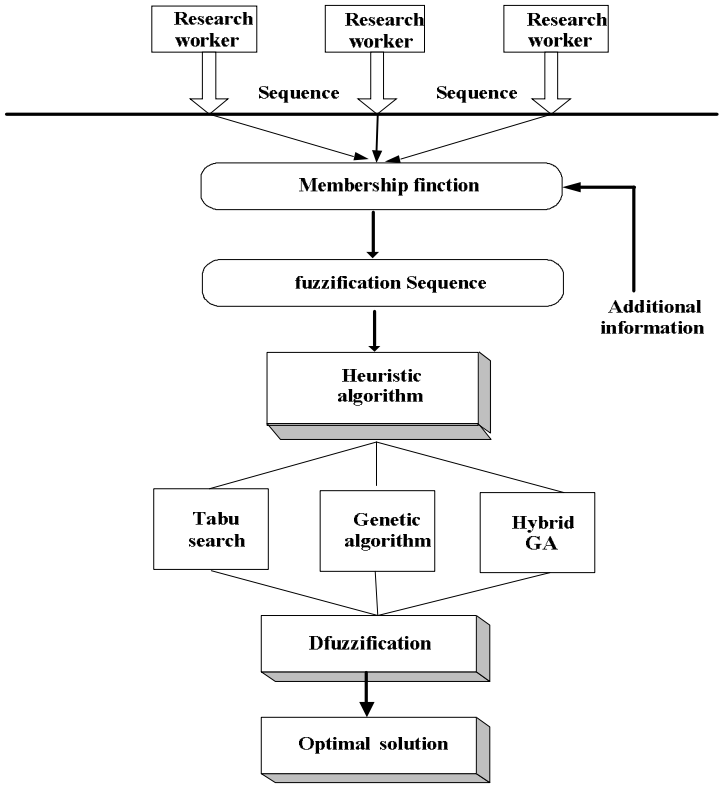


圖 3- 2 系統架構圖

1. 序列模糊化架構

模糊歸屬函數*(Fuzzy numbers)*—有一個模糊集合或模糊數（FN）在實線$\Re$上，可表示為(a_1,a_2,a_3),a_2表示眾數，a_1與a_3分別表示$\tilde{A}(x)$的左界與右界，而歸屬函數$\tilde{A}(x)$可被定義為$x \in \Re$ to $\tilde{A}$

$$\tilde{A}(x)=\begin{cases}0, & x<a_1,\\ L((x-a_1)/(a_2-a_1)), & a_1\le x\le a_2,\\ R((a_3-x)/(a_3-a_2)), & a_2\le x\le a_3,\\ 0, & x>a_3,\end{cases} \tag{20}$$

*L*和*R*分別表示$\tilde{A}(x)$中左邊與右邊的*(shape functions)*。若以三角型隸屬度函數（triangular membership function）可表示成如圖3-3

$$\tilde{A}(x)=\begin{cases}0, & x<a_1,\\ (x-a_1)/(a_2-a_1), & a_1\le x\le a_2,\\ (a_3-x)/(a_3-a_2), & a_2\le x\le a_3,\\ 0, & x>a_3.\end{cases} \tag{21}$$

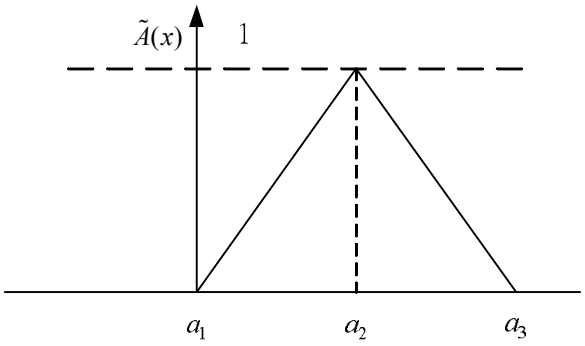


圖 3- 3 三角隸屬度函數 (a_1, a_2, a_3).

在一信賴區間α水準，$\alpha \in (0, 1]$，我們定義A_α：

$$A_\alpha=\left\{x\middle|\tilde{A}(x)\ge\alpha\right\}, \tag{22}$$

三角型隸屬度函數以α-截集表示為：

$$A_\alpha=[a_1^{(\alpha)},a_2^{(\alpha)}]=[a_1+\alpha(a_2-a_1),a_3-\alpha(a_3-a_2)], \tag{23}$$

$a_1^{(\alpha)}$表示A_α的下界，而$a_2^{(\alpha)}$表示A_α的上界。

α-截集（α-cut）模糊算術(*The α-cut fuzzy arithmetic)* 根據模糊算術方法，

Zadeh's sup-min方法表示如下

$$(\tilde{A} \circ \tilde{B})(z) = \sup_{x \circ y = z} \min(\tilde{A}(x), \tilde{B}(y)), \tag{24}$$

其中，$\circ$表示任何的算數操作(arithmetic operation)。公式(5)代表Mizumoto and Tanaka [38]於（1976）使用相同方法在α-截集模糊數和區間算術。模糊算數的結果可被稱作α-截集算數，並且α-截集算數的發展在Chang (2005) [15]有做介紹

α-截集*模糊*算術中，模糊數基本運算公式（加法、減法、乘法、除法）可以計算的更快，在每一個α-level區間中，使用區間算術（Kaufmann and Gupta, 1988 [32]）以下為本研究所利用的模糊算數。

5. 模糊數加法

令A和B分別為兩模糊數$\tilde{A}$和$\tilde{B}$，在一信賴區間α水準，$B_\alpha = [b_1^{(\alpha)}, b_2^{(\alpha)}]$，$\alpha \in (0, 1]$, $\forall \tilde{A}, \tilde{B} \subset \Re$，表示如下：

$$A_\alpha + B_\alpha = [a_1^{(\alpha)}, a_2^{(\alpha)}] + [b_1^{(\alpha)}, b_2^{(\alpha)}] = [a_1^{(\alpha)} + b_1^{(\alpha)}, a_2^{(\alpha)} + b_2^{(\alpha)}], \forall \alpha \in (0, 1]. \tag{25}$$

6. 模糊數減法

在一信賴區間α水準，$\forall \tilde{A}, \tilde{B} \subset \Re$ $\alpha \in (0, 1]$。

$$A_\alpha - B_\alpha = [a_1^{(\alpha)}, a_2^{(\alpha)}] - [b_1^{(\alpha)}, b_2^{(\alpha)}] = [a_1^{(\alpha)} - b_2^{(\alpha)}, a_2^{(\alpha)} - b_1^{(\alpha)}] \tag{26}$$

7. 模糊數乘法

在一信賴區間α水準，$\forall \tilde{A}, \tilde{B} \subset \Re$ $\alpha \in (0, 1]$。

$$\begin{aligned} A_\alpha \times B_\alpha = [a_1^{(\alpha)}, a_2^{(\alpha)}] \times [b_1^{(\alpha)}, b_2^{(\alpha)}] = [&\min(a_1^{(\alpha)} b_1^{(\alpha)}, a_1^{(\alpha)} b_2^{(\alpha)}, a_2^{(\alpha)} b_1^{(\alpha)}, a_2^{(\alpha)} b_2^{(\alpha)}), \\ &\max(a_1^{(\alpha)} b_1^{(\alpha)}, a_1^{(\alpha)} b_2^{(\alpha)}, a_2^{(\alpha)} b_1^{(\alpha)}, a_2^{(\alpha)} b_2^{(\alpha)})]. \end{aligned} \tag{27}$$

8. 模糊數除法

在一信賴區間α水準，$\forall \tilde{A}, \tilde{B} \subset \Re$ $\alpha \in (0, 1]$。

$$\begin{aligned} A_\alpha \div B_\alpha = [a_1^{(\alpha)}, a_2^{(\alpha)}] \div [b_1^{(\alpha)}, b_2^{(\alpha)}] = [&\min(a_1^{(\alpha)} / b_1^{(\alpha)}, a_1^{(\alpha)} / b_2^{(\alpha)}, a_2^{(\alpha)} / b_1^{(\alpha)}, a_2^{(\alpha)} / b_2^{(\alpha)}) \\ &\max(a_1^{(\alpha)} / b_1^{(\alpha)}, a_1^{(\alpha)} / b_2^{(\alpha)}, a_2^{(\alpha)} / b_1^{(\alpha)}, a_2^{(\alpha)} / b_2^{(\alpha)})], \text{ for } b_1^{(\alpha)}, b_2^{(\alpha)} > 0, \forall \alpha \in [0,1]. \end{aligned} \tag{28}$$

在上述的討論中，我們有介紹到PAM250矩陣。1個PAM的演化距離表示100個残基中發生一個残基突變的機率，而PAM250矩陣則是將1PAM矩陣自乘250次得知。由於PAM矩陣是建立在胺基酸的改變上，並且是資訊來自許許多多的蛋白質實驗上，因此我們可以知道，其中必定包括一些不確

定的因子，因此模糊歸屬函數被使用來替換PAM250矩陣中的元素，而模糊PAM250矩陣可被定義如下：

$$\widetilde{M}_{ij} = (m_{ij} - l_{ij}, m_{ij}, m_{ij} + r_{ij}) \tag{29}$$

其中M_{ij}為 PAM250 矩陣中的元素，代表在給定演化區間的情況下，第 i 行胺基酸被第 j 列胺基酸所取代的機率，l_{ij}表示在M_{ij}中，演化距離的下界值，r_{ij}表示在M_{ij}中，演化距離的上界值。

而模糊 PAM250 矩陣則可參考附錄三。

選擇適當的解模糊方法：解模糊的過程剛好與模糊化相反，它是將一模糊集合$B(y)$，$y \in Y$，轉換至一個明確值y^*的動作，即找一最適合代表模糊集合$B(y)$的確定點$y^* \in Y$。至於，如何才是最適合的解模糊化法？大可用以下三個準則來決定。[2] [35]

1. 合理性：至少在人的直覺上，y^*代表$B(y)$是合理的，可被接受的。譬如y^*在$B(y)$之底集中間，或y^*的歸屬度值比較高。倘若y^*在$B(y)$之底集以外，直覺上就不合理了。

2. 計算簡單：這是為了在模糊控制問題上使用方便。

3. 連續性：$B(y)$之形狀有稍許的變化，y^*之位置變化不會太大。

只要合於上面的準則及定義的解模糊化方法均可被接受。而在文獻上所提到的解模糊化方法有許多種，以下幾種解模糊化的方法是較常被應用的 [2] [35]：(1)重心解模糊化法(center of gravity defuzzification)、(2)面積和之中心解模糊化法(center of sum defuzzification)、(3) 最大面積之中心解模糊化法(center of largest area defuzzification)、(4)第一個最大值解模糊化法(first of maxima defuzzification)、(5)最後一個最大值解模糊化法(last of maxima defuzzification)、(6)最大值之平均值解模糊化法(middle of maxima defuzzification)及(7)高度解模糊化法(height defuzzification)。

本研究利用三角模糊數將PAM與BLOUSM矩陣模糊化詳細可見附錄三，並利用重心法（the center of area ，COA）解糢糊化。重心COA(x)公式表示為

$$COA(x) = \frac{\int_{\Re} \tilde{A}(x)\, x\, dx}{\int_{\Re} \tilde{A}(x)\, dx} \tag{30}$$

2. 演算法選擇

選擇 Tabu 演算法

(1) 起始解：隨機產生 10 組起始解，假設要比對的基因分別叫做 KPDNG 與 KDN，而隨機產生的 10 組解中，KDN--為其中一組解，另一組為 KD-N-。

(2) 產生候選名單：假設初始解有 KDN--與 KD-N-兩種，將 PAM250、BLOUSM62 矩陣與 affine gap cost(open cost=5,extened cost=2)利用 gap 移步，將每次移步後所求得的解加入在候選名單中，而本研究的 gap 移步方式有兩種，以下將作簡單的介紹。若以 KDN--為初始解，則移步方式則移步方式如圖 3-4。由圖 3-4 可知，若 gap 左右緊臨下一個 gap，則我們將 2 個 gap 看做一個去做移步，原因在於當利用仿射性間格懲罰函數時，將間格成本分為開啟成本(open gap cost)與延伸成本(extend gap cost)，當間格越連續時，所受到的懲罰會比間格數分散的時候低，因此基於這個原因，我們將移步方式分開。

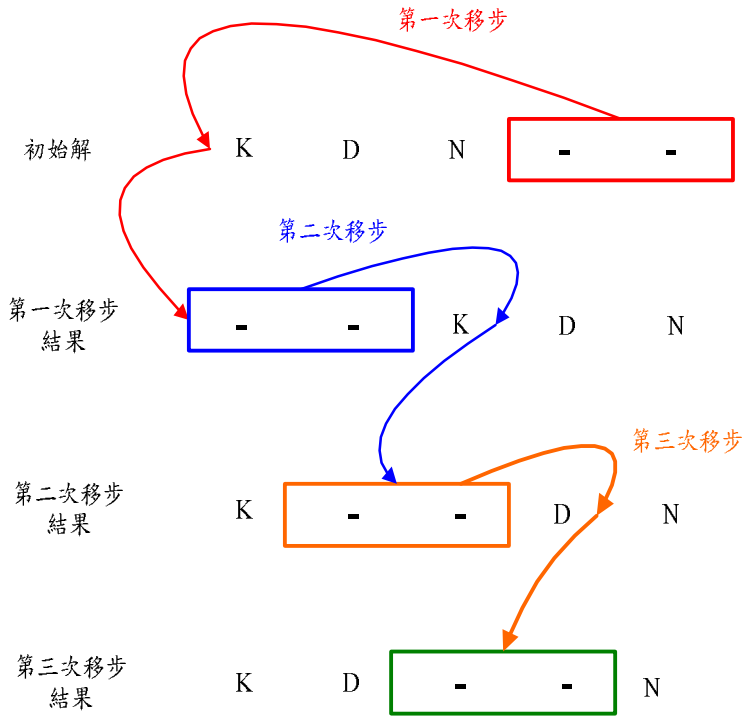


圖 3- 4 區塊間格移步

若初始解為 KD-N-是屬於 gap 不緊鄰，則移步方式則移步方式如圖 3-5。

由圖 3-5 可看出，若 gap 不緊鄰，我們將把 2 個 gap 當作獨立的看待，因此移步方式，會先由第一個 gap 做完移步後，第 2 個 gap 才會繼續，若出現 3 個以上則以此類推，且兩種移步方式可能會同時出現。

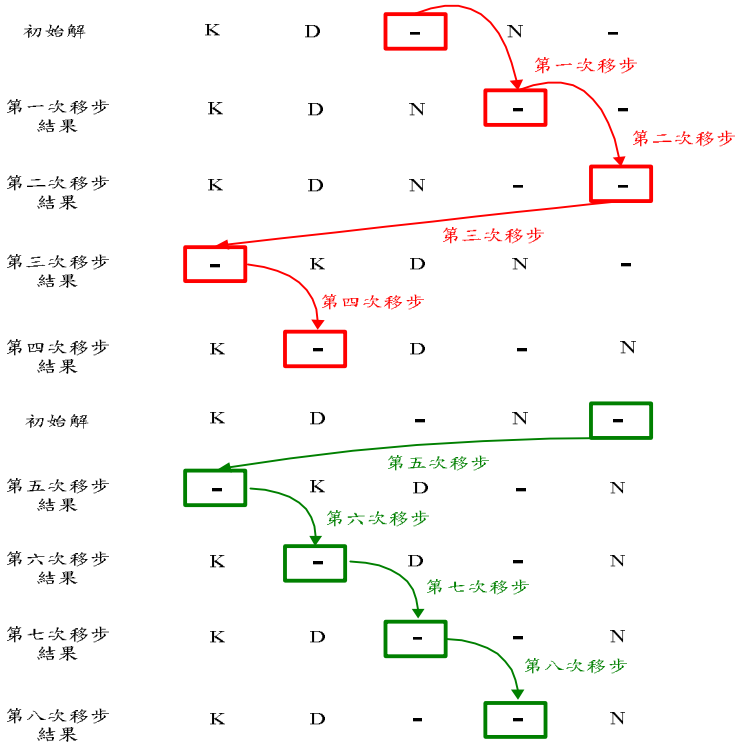


圖 3- 5 單一間格移步

選擇 GA 演算法(A multi-simplexes and genetic algorithm hybrid method , 2003 [13], A genetic algorithm for solving a fuzzy economic lot-size scheduling problem, 2004 [14])下圖 3-6 為本研究的演算架構：

1. 原始問題：本研究是將基因演算法應用在基因序列的排比上。
2. 參數編碼：本研究參考 Hung D.N (2002) [32]在 Aligning Multiple Protein Sequences by Parallel HybridGenetic Algorithm 中所用的編碼方式，利用向量的方式將編碼範圍分成 3 個緯度去進行求解搜尋，圖 3-7 為參數編碼圖的簡單介紹：

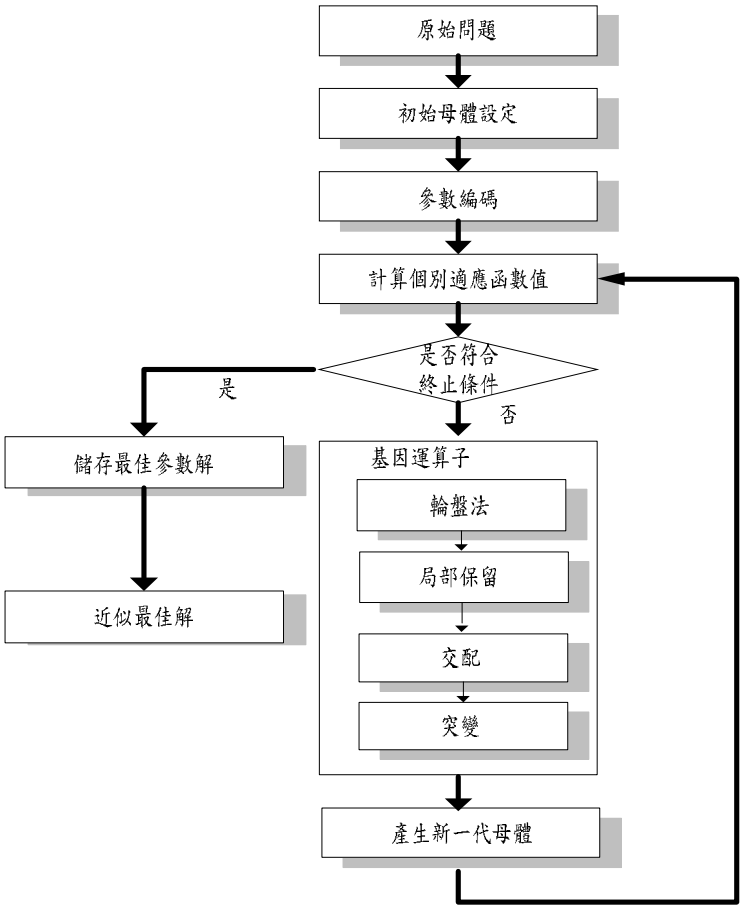


圖 3- 6 基因演算法架構圖

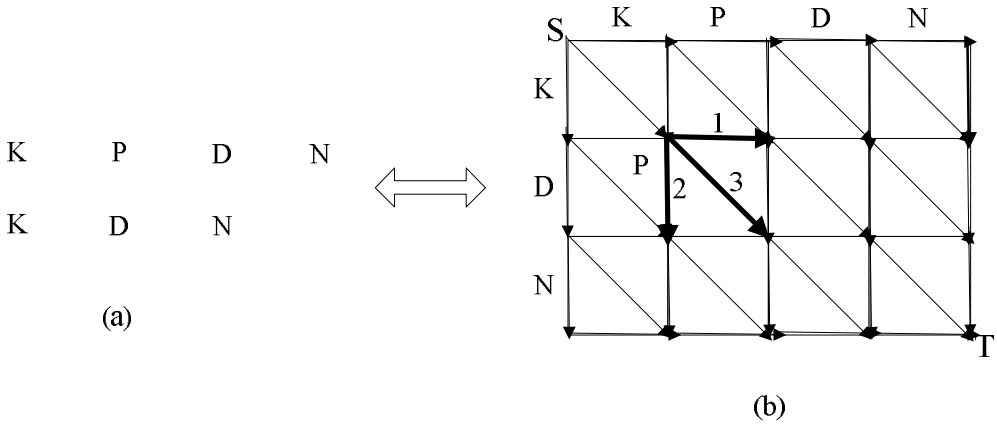


圖 3- 7 編碼方式

S 為起點，T 為終點，KPDN 與 KDN 為兩條要進行比對的基因，從起點至終點皆有 3 個方向可前進，因此我們在每個 x 與 y 軸的交點上可發現皆須進行方向的抉擇，以圖(b)中的 P 點為例，若向緯度 1 前進，則 P 點並不能對應到 y 軸的基因，因此 P 點會對應到一個 gap，若向緯度 2 前進，則 P 點會保留，因此 D 點會對應到一個 gap，若向緯度 3 前進，則 P 點會

與 y 軸的基因 D 相對應，因此 P 點會對應到一個 D，如此便能從起點至終點完整的對兩條基因行進方向進行編碼。

3. 初始母體設定：本研究的初始解(popsize)採隨機方式選擇 10 條進行求解，若以

上圖（a）為例，編碼位元為 4，假設隨機產生的初始解中的一組解為 KDN-，其中-所代表的意義為 gap 在進行編碼後即變成 3331，若初始解為-KDN，進行編碼後即變成 1333。

4. 計算各別適應函數值：在基因序列排比上我們利用 PAM250、BLOUSM62 矩陣與 affine gap cost 來進行適應函數值的計算，而 open gap cost 設為 5，extened gap cost 設為 2，以下為適應函數值簡單介紹：

假設初始解有 $f(x_1)$,$f(x_2)$..$f(x_{10})$ 10 組，且 10 組的值在利用得分矩陣與懲罰函數後，分數分別為[10，50，15，-10，-3，9，10，10，10，10]，則適應函數值計算步驟如下；

1. 判斷所有組解的分數是否包含負值。
2. 若其中包含負值，則將所有解分數調成正值，以上述解為例，經調整後所有解的分數則變成[20，60，25，0，7，19，20，20，20，20]，調整方法則是以最小值為基準，每組解同時加上-10。
3. 計算此帶的最佳適應函數值，公式為 fitness=$\frac{\max f(x_i)}{\sum_{i=1}^{10} f(x_i)} \doteq 0.3$

5. 基因運算子：整個基因演算法的核心部份，可分成以下幾個部份

(1)輪盤法：假設利用每一組解所計算出的適應函數值為 $f(x_i), i=1\sim 6$，

(2)則每一組解的複製個數 n_i 為 $\frac{f(x_i)}{\sum_{i=1}^{10} f(x_i)} \times 10$，當 $\sum_{i=1}^{10} n_i$ >popsize 時，從適應函數值最低的開始刪除，直到 $\sum_{i=1}^{10} n_i$ =popsize 才停止。

(3)局部保留：由於基因演算法屬於全域搜尋的方法，因此本研究希望除了依靠交配(crossover)來進行解的收斂外，另外再加上其他原則以便使收斂能夠更快速。若以上圖(a)為例，若初始解為 KDN-則與 KPDN 比較後發現，第一個位置的基因 K 與初始解相同，因此我們將

其保留，原因在於當相同基因對應時所獲得的分數，遠高於不同基因對應時所獲得的分數，因此若將其保留，可加速求解收斂的速度，此步驟圖例將會與交配(crossover)步驟一起說明。

(4) 交配(crossover)：由於編碼是以行進方向為基礎，因此在進行交配時，本研究不採用以往的兩兩交配方式，因此父代與母代皆為同一條染色體，圖 3-8 即為局部保留與交配的簡單圖例介紹。

(5) 突變：傳統的基因演算法中，為了避免收斂速度太慢，通常會給定一個較小突變機率，而本研究則是避免收斂過快，因此給定一個較高的突變機率，且在局部保留時所留下的基因也有可能突變。圖 3-9 為突變的簡單介紹。

綜合上述所說，圖 3-10 為基因演算法應用於基因序列排比上的完整圖例說明。

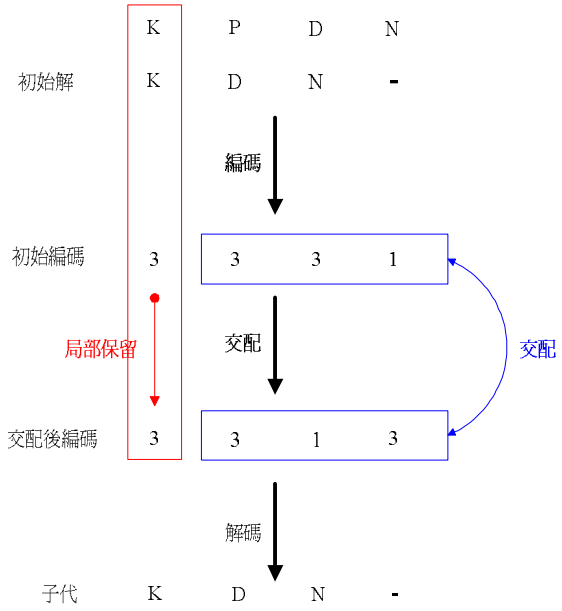


圖 3- 8 交配流程圖

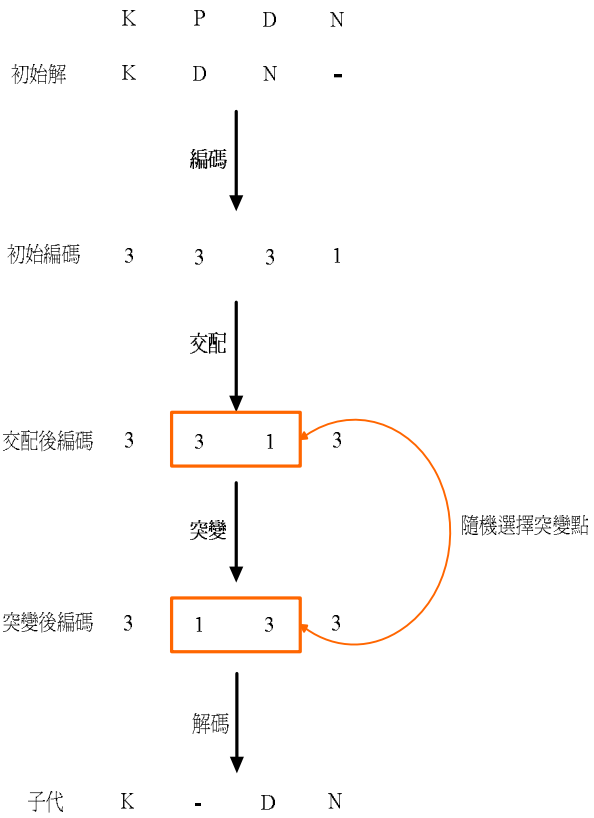


圖 3- 9 突變示意圖

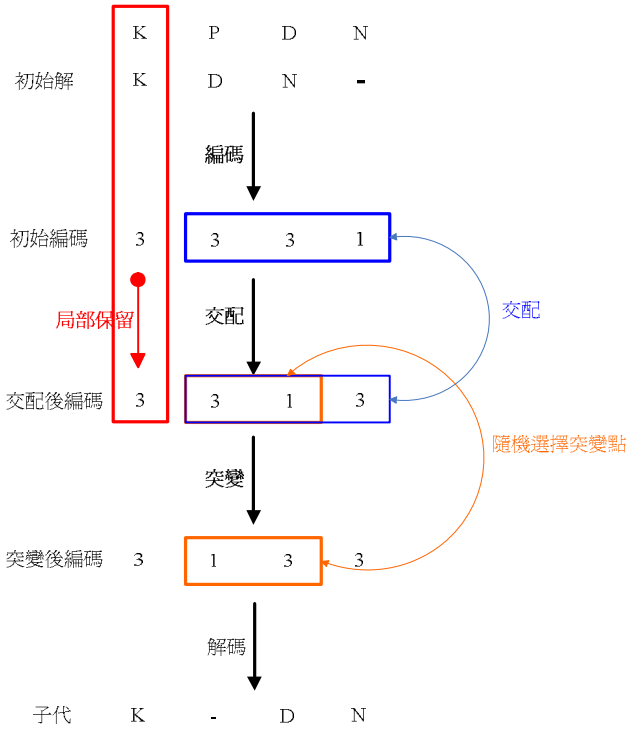


圖 3- 10 基因演算法

選擇混合式 GA 演算法:

本研究架構的混合式演算法是將禁忌搜巡法與基因演算法混合，演算法流程圖如圖 3-11。

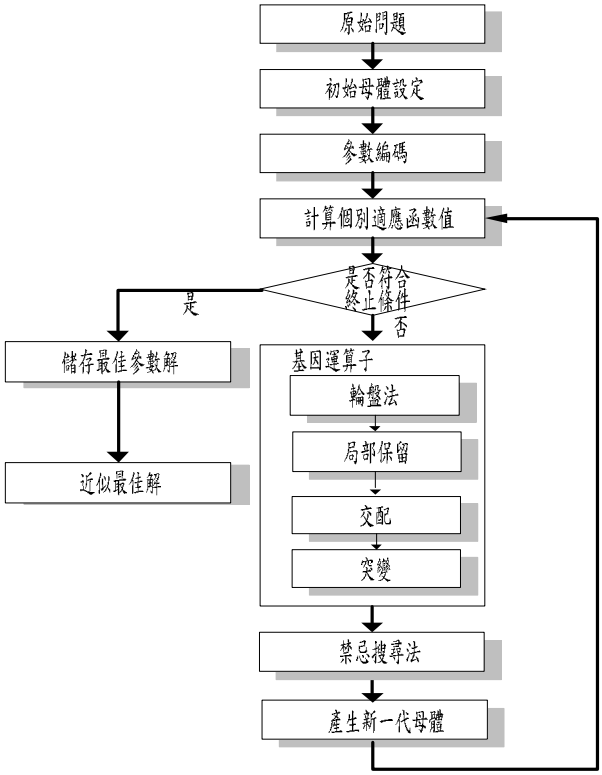


圖 3- 11 混合式基因演算法流程圖

假設以基因演算法的例子說明，簡單範例如圖 3-12。將基因演算法所搜尋到的解與禁忌搜尋法的解作比較，若禁忌演算法的解較好，則將代替掉基因演算法的解，反之則維持原解，然後在不斷重複直到到達終止條件。

6. 先進序列排比：將序列利用模糊得分矩陣，懲罰函數(open cost 設 5，extend cost 設 2)與混合式基因演算法進行序列分析。
7. 最佳解獲得：當解模糊的解能夠被我們所接受或到達設定的遞迴次數時，我們即停止運算。

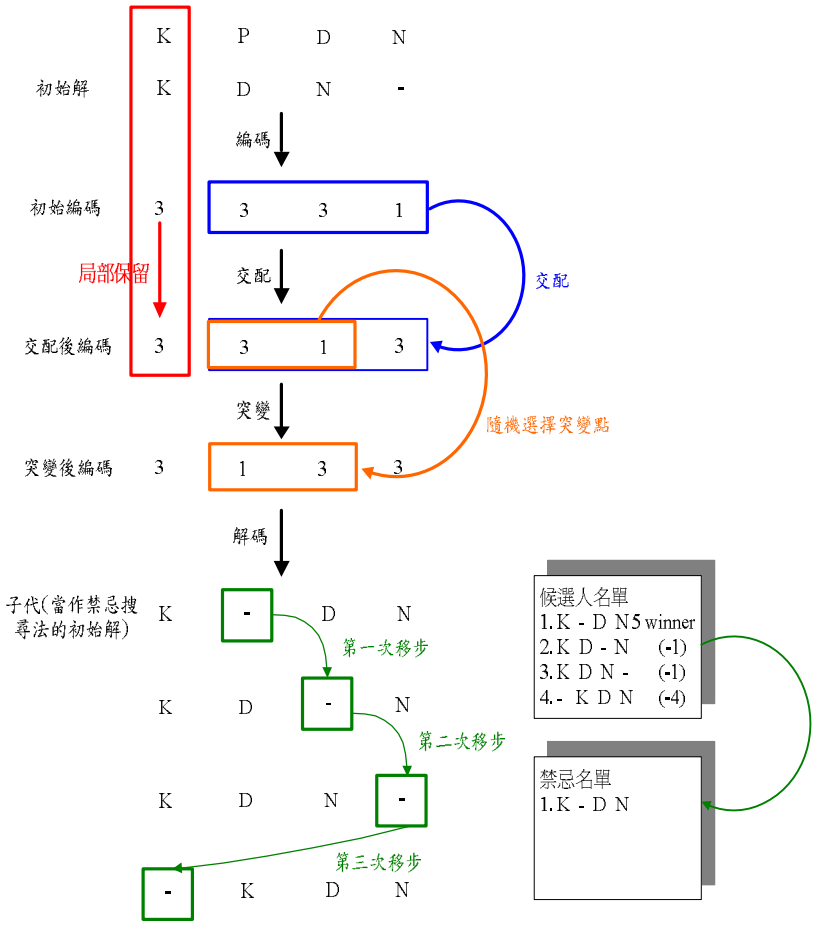


圖 3- 12 混合式基因演算法範例

第四章 實驗設計與結果分析

4.1 基因演算法應用於明確值結果分析

在眾多的資料中，我們利用 BAliBASE(Thompson et al., 1999) [48]所提供的蛋白質序列作為序列排比的樣本，其中利用 reference 1 中的 SH3 來進行排比與分析，並且使用模糊 PAM 矩陣作為評分依據。除此之外，為了解決排比序列長度不同的問題，本研究利用間隔性懲罰函數(affine gap penalties)，主要原因在於此函數有較低的複雜性，因此求解快速。而在本研究所使用的間隔性懲罰函數中，開啟成本(open gap cost)設 5，延伸成本(extended gap cost)則設 2。.

在本研究中，我們使用了不同的三角歸屬度函數，分別為右偏、對稱、左偏的三角歸屬度函數。而右偏歸屬度函數為 $\widetilde{M}_{ij}$=(mij-1, mj, mij+3)，對稱歸屬度函數為 $\widetilde{M}_{ij}$=(mij-3, mij, mij+3)，左偏歸屬度函數為 $\widetilde{M}_{ij}$=(mij-3, mij, mij+1)。基因演算法求解代數為 500 代，重複次數為 20 次。基因演算法利用明確值求解(GA-crisp)與基因演算法利用模糊值求解(GA-fuzzy)比較結果參考表 4-9、表 4-18、圖 4-1、圖 4-2 與圖 4-3 。表 4-9 中我們包含平均 CS、最低 CS、最高 CS 以及連續最長配對 CS。其中，平均 CS 與 GA-crisp 沒有顯著差異，而最低 CS 與最高 CS 皆優於 GA-crisp，而在表 4-18 中，平均 CS 與 GA-crisp 沒有顯著差異，最低 CS 優於 GA-crisp，最高 CS 在序列 lpht-lihvA 與 lpht-lvie 有顯著差異，圖 4-1 則是針對最高 CS 所畫出的圖形，空心且圖形為圓形與三角形分別代表模糊 PAM 與模糊 BLOSUM 矩陣應用於基因演算法，而實心且圖形為圓形與三角形分別代表明確值 PAM 與明確值 BLOSUM 矩陣應用於基因演算法。除此之外，連續最長配對 CS(The longest continued sequence CS)也是一項非常重要的指標，若我們能找出越長的連續配對基因，代表兩序列間的特徵、功能較有可能出現相似的情形。不管從表 4-9 或表 4-18 中，顯示出我們的方法所獲得的連續最長配對 CS，皆優於傳統的基因演算法，圖 4-2 則是針對連續最長配對畫出的圖形，且由圖 4-3 可得知，應用不同矩陣時，模糊矩陣所得的結果波動要比利用明確值矩陣來的小，這証明模糊理論確實可以降低不確定性環境的影響，其中圖 4-1 是利用表 4-9 與表 4-18 中的連續最長配對 CS 所畫出，以 lpht-lycsB 為例，兩

模糊矩陣的誤差值為 35%-33.75%=1.25%，而兩明確值的誤差值為 28.75%-25%=3.75%，以此類推畫出 lpht-lycsB、lpht-laboA、lpht-lihvA 及 lpht-lvie 4 點。

4.1.1 基因演算法將 PAM250 矩陣應用於明確值結果分析

在利用基因演算法求解時，我們的基因代數為 500 代，重複次數為 20 次，以下為實驗結果。

表 4- 1 PAM250 矩陣應用於明確值之結果

序列名稱	使用方法	求解代數	平均 CS(%)	最高 CS(%)	最低 CS(%)	連續最長 CS(%)
1pht-1ycsB	GA	500	36.25	51.25	18.75	25
lpht-1aboA	GA	500	32.5	45	18.75	17.5
lpht-1ihvA	GA	500	22.5	33.75	11.25	10
lpht-1vie	GA	500	27.5	50	10	18.75

4.1.2 基因演算法將 PAM250 矩陣應用於三角模糊數結果分析

在利用基因演算法求解時，我們的基因代數為 500 代，重複次數為 20 次，右偏歸屬函數範圍為[a-1,a,a+3]之間，左偏歸屬函數範圍為[a-3,a,a+1]之間以下，對稱歸屬函數範圍為[a-3,a,a+3]之間，以下為實驗結果。

表 4- 2 不同歸屬函數在基因 1pht 與基因 1ycsB 的排比

序列名稱	使用方法	求解代數	歸屬函數	平均 CS(%)	最高 CS(%)	最低 CS(%)	連續最 CS(%)
lpht-1ycsB	GA	500	右偏	35	65	21.25	33.75
lpht-1ycsB	GA	500	左偏	32.5	41.25	21.25	23.75
lpht-1ycsB	GA	500	對稱	31.25	35	17.5	23.75

表 4- 3 不同歸屬函數在基因 1pht 與基因 1aboA 的排比

序列名稱	使用方法	求解代數	歸屬函數	平均 CS(%)	最高 CS(%)	最低 CS(%)	連續最長 CS(%)
lpht-1aboA	GA	500	右偏	22.5	48.75	11.25	17.5
lpht-1aboA	GA	500	左偏	33.75	51.25	16.25	18.75
lpht-1aboA	GA	500	對稱	33.75	48.75	20	20

表 4- 4 不同歸屬函數在基因 1pht 與基因 1ihvA 的排比

序列名稱	使用方法	求解代數	歸屬函數	平均 CS(%)	最高 CS(%)	最低 CS(%)	連續最長 CS(%)
lpht-1ihvA	GA	500	右偏	21.25	32.5	7.5	13.75
lpht-1ihvA	GA	500	左偏	22.5	35	11.25	16.25
lpht-1ihvA	GA	500	對稱	22.5	33.75	12.5	12.5

表 4- 5 不同歸屬函數在基因 1pht 與基因 1vie 的排比

序列名稱	使用方法	求解代數	歸屬函數	平均 CS(%)	最高 CS(%)	最低 CS(%)	連續最長 CS(%)
lpht-1vie	GA	500	右偏	27.5	47.5	12.5	25
lpht-1vie	GA	500	左偏	25	31.25	8.75	17.5
lpht-1vie	GA	500	對稱	26.25	41.25	12.5	21.25

表 4- 6 右偏三角歸屬函數

序列名稱	使用方法	求解代數	平均 CS(%)	最高 CS(%)	最低 CS(%)	連續最長 CS(%)
lpht-1ycsB	GA	500	35	65	21.25	33.7
lpht-1aboA	GA	500	22.5	48.75	11.25	17.5
lpht-1ihvA	GA	500	21.25	32.5	7.5	13.75
lpht-1vie	GA	500	27.5	47.5	12.5	25

表 4- 7 左偏三角歸屬函數

序列名稱	使用方法	求解代數	平均 CS(%)	最高 CS(%)	最低 CS(%)	連續最長 CS(%)
lpht-1ycsB	GA	500	32.5	41.25	21.25	23.75
lpht-1aboA	GA	500	33.75	52.5	16.25	18.75
lpht-1ihvA	GA	500	22.5	23	11.25	16.25
lpht-1vie	GA	500	25	31.25	8.75	17.5

表 4- 8 對稱三角歸屬函數

序列名稱	使用方法	求解代數	平均 CS(%)	最高 CS(%)	最低 CS(%)	連續最長 CS(%)
lpht-1ycsB	GA	500	31.25	47.5	17.5	23.75
lpht-1aboA	GA	500	33.75	48.75	20	20
lpht-1ihvA	GA	500	22.5	33.75	12.5	12.5
lpht-1vie	GA	500	26.25	41.25	12.5	21.25

4.1.3三角模糊數應用於 PAM250 矩陣與明確值比較結果分析

以下數據是根據上表所整理，我們分別將模糊歸屬函數所求得的最佳解找出並整理資料如下表。

表 4- 9 利用 PAM250 時的模糊最佳解與明確值最佳解比較表(連續最長 CS)

序列名稱	使用方法	求解代數	最佳歸屬函數	平均 CS(%)	最高 CS(%)	最低 CS(%)	連續最長 CS(%)
lpht-1ycsB	GA-fuzzy	500	右偏	35	65	21.25	33.75
lpht-1aboA	GA-fuzzy	500	對稱	33.75	48.75	20	20
lpht-1ihvA	GA-fuzzy	500	左偏	22.5	35	11.25	16.25
lpht-1vie	GA-fuzzy	500	右偏	27.5	47.5	12.5	25
lpht-1ycsB	GA-crisp	500	無	36.25	51.25	18.75	25
lpht-1aboA	GA-crisp	500	無	32.5	45	18.75	17.5
lpht-1ihvA	GA-crisp	500	無	22.5	33.75	11.25	10
lpht-1vie	GA-crisp	500	無	27.5	50	10	18.75

4.1.4基因演算法將 BLOUSM62 矩陣應用於明確值結果分析

在利用基因演算法求解時，我們的基因代數為 500 代，重複次數為 20 次，以下為實驗結果。

表 4- 10 利將 BLOUSM62 時應用於明確值之結果

序列名稱	使用方法	求解代數	平均 CS(%)	最高 CS(%)	最低 CS(%)	連續最長 CS(%)
lpht-1ycsB	GA	500	36.25	53.75	27.5	28.75
lpht-1aboA	GA	500	36.25	51.25	25	25
lpht-1ihvA	GA	500	21.25	35	10	15
lpht-1vie	GA	500	26.25	36.25	13.75	18.75

4.1.5基因演算法將 BLOUSM62 矩陣應用於三角模糊數結果分析

在利用基因演算法求解時，我們的基因代數為 500 代，重複次數為 20 次，右偏歸屬函數範圍為[a-1,a,a+3]之間，左偏歸屬函數範圍為[a-3,a,a+1]之間以下，對稱歸屬函數範圍為[a-3,a,a+3]之間，以下為實驗結果。

表 4- 11 不同歸屬函數在基因 1pht 與基因 1ycsB 的排比

序列名稱	使用方法	求解代數	歸屬函數	平均 CS(%)	最高 CS(%)	最低 CS(%)	連續最長 CS(%)
1pht-1ycsB	GA	500	右偏	36.25	51.25	30	35
1pht-1ycsB	GA	500	左偏	35	40	30	22.5
1pht-1ycsB	GA	500	對稱	36.25	57.5	26.25	22.5

表 4- 12 同歸屬函數在基因 1pht 與基因 1aboA 的排比

序列名稱	使用方法	求解代數	歸屬函數	平均 CS(%)	最高 CS(%)	最低 CS(%)	連續最長 CS(%)
lpht-1aboA	GA	500	右偏	35	48.75	30	26.25
lpht-1aboA	GA	500	左偏	37.5	53.75	30	25
lpht-1aboA	GA	500	對稱	32.5	47.5	15	21.25

表 4- 13 不同歸屬函數在基因 1pht 與基因 1ihvA 的排比

序列名稱	使用方法	求解代數	歸屬函數	平均 CS(%)	最高 CS(%)	最低 CS(%)	連續最長 CS(%)
lpht-1ihvA	GA	500	右偏	25	50	10	22.5
lpht-1ihvA	GA	500	左偏	27.5	43.75	2.5	17.5
lpht-1ihvA	GA	500	對稱	22.5	45	7.5	12.5

表 4- 14 同歸屬函數在基因 1pht 與基因 1vie 的排比

序列名稱	使用方法	求解代數	歸屬函數	平均 CS(%)	最高 CS(%)	最低 CS(%)	連續最長 CS(%)
lpht-1vie	GA	500	右偏	25	38.75	7.5	17.5
lpht-1vie	GA	500	左偏	27.5	47.5	13.75	23.75
lpht-1vie	GA	500	對稱	27.5	52.5	11.25	25

表 4- 15 右偏三角歸屬函數

序列名稱	使用方法	求解代數	歸屬函數	平均 CS(%)	最高 CS(%)	最低 CS(%)	連續最長 CS(%)
1pht-1ycsB	GA	500	右偏	36.25	51.25	30	35
lpht-1aboA	GA	500	右偏	35	48.75	30	26.25
lpht-1ihvA	GA	500	右偏	25	50	10	22.5
lpht-1vie	GA	500	右偏	25	38.75	7.5	17.5

表 4- 16 左偏三角歸屬函數

序列名稱	使用方法	求解代數	歸屬函數	平均 CS(%)	最高 CS(%)	最低 CS(%)	連續最長 CS(%)
1pht-1ycsB	GA	500	左偏	35	40	30	22.5
lpht-1aboA	GA	500	左偏	37.5	53.75	30	25
lpht-1ihvA	GA	500	左偏	27.5	43.75	2.5	17.5
lpht-1vie	GA	500	左偏	27.5	47.5	13.75	23.75

表 4- 17 對稱三角歸屬函數

序列名稱	使用方法	求解代數	歸屬函數	平均CS(%)	最高CS(%)	最低CS(%)	連續最長CS(%)
1pht-1ycsB	GA	500	對稱	36.25	57.5	26.25	22.5
lpht-1aboA	GA	500	對稱	32.5	47.5	15	21.25
lpht-1ihvA	GA	500	對稱	22.5	45	7.5	12.5
lpht-1vie	GA	500	對稱	27.5	52.5	11.25	25

4.1.6三角模糊數應用於 BLOUSM62 矩陣與明確值比較結果分析

以下數據是根據上表所整理，我們分別將模糊歸屬函數所求得的最佳解找出並整理資料如下表。

表 4- 18 利用 BLOUSM62 時的模糊最佳解與明確值最佳解比較表(連續最長 CS)

序列名稱	使用方法	求解代數	最佳歸屬函數	平均CS(%)	最高CS(%)	最低CS(%)	連續最長CS(%)
1pht-1ycsB	GA-fuzzy	500	右偏	36.25	51.25	30	35
lpht-1aboA	GA-fuzzy	500	右偏	35	48.75	30	26.25
lpht-1ihvA	GA-fuzzy	500	右偏	25	50	10	22.5
lpht-1vie	GA-fuzzy	500	對稱	27.5	52.5	13.75	25
1pht-1ycsB	GA-crisp	500	無	36.25	53.75	27.5	28.75
lpht-1aboA	GA-crisp	500	無	36.25	51.25	25	25
lpht-1ihvA	GA-crisp	500	無	21.25	35	10	15
lpht-1vie	GA-crisp	500	無	26.25	36.25	13.75	18.75

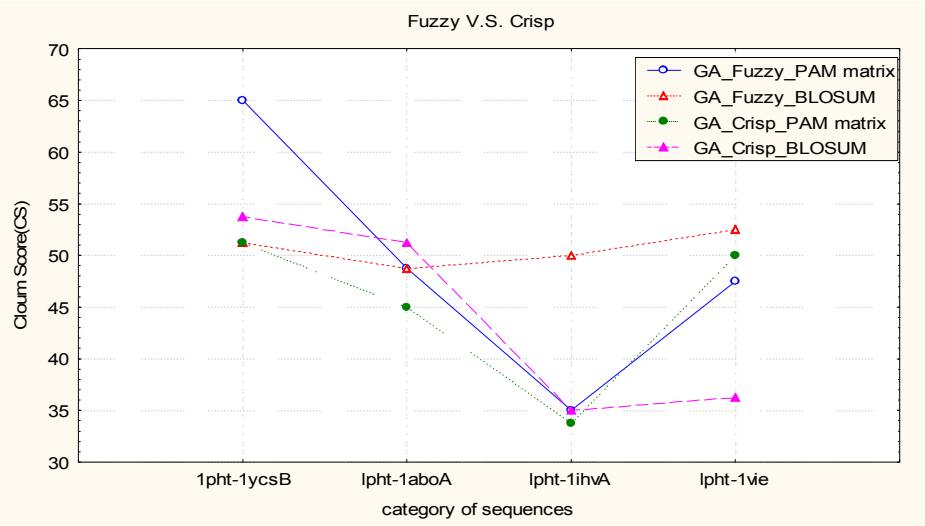


圖 4- 1GA-crisp 與 GA-fuzzy 應用不同矩陣比較圖(最高 CS)

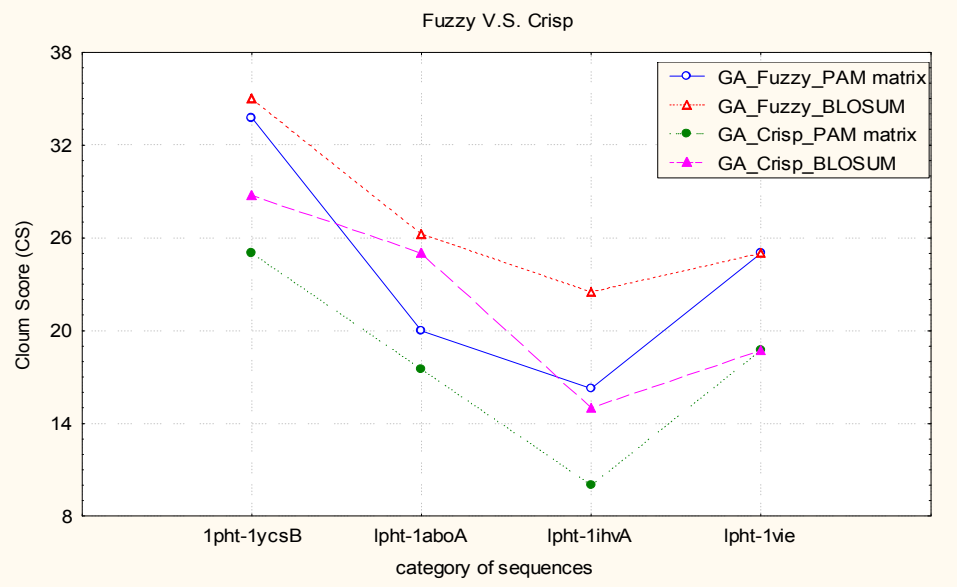


圖 4- 2GA-crisp 與 GA-fuzzy 應用不同矩陣比較圖(連續最長配對 CS)

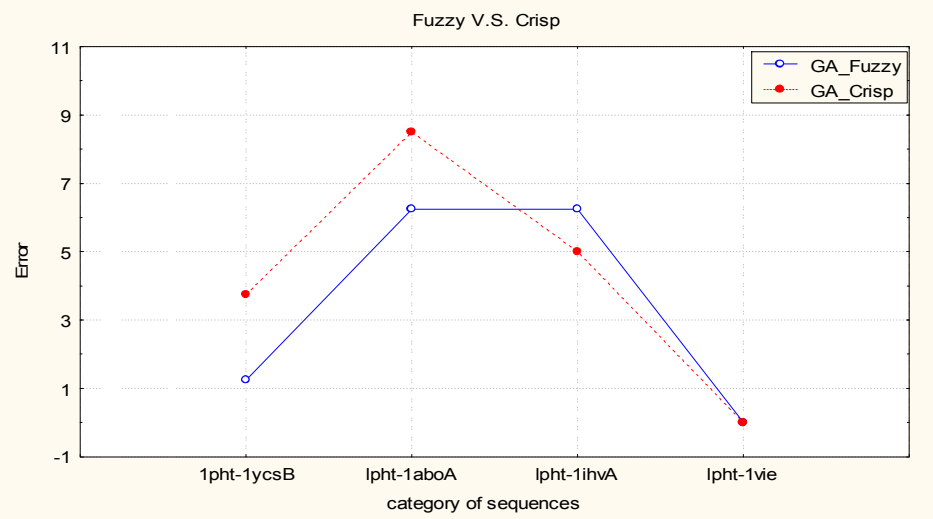


圖 4- 3GA-crisp 與 GA-fuzzy 誤差比較圖(連續最長配對 CS)

4.2 混合式基因演算法應用於明確值結果分析

本研究的混合式基因演算法，是將基因演算法每代所搜尋到的最佳解，在去進行禁忌搜尋，以下便為混合式演算法的實驗結果。

4.2.1 將 BLOUSM62 矩陣應用於明確值結果分析

在利用基因演算法求解時，我們的基因代數為 500 代，重複次數為 5 次，以下為實驗結果，從實驗結果發現，利用 BLOUSM62 矩陣時，較容易出現 2 段較長的連續 CS，因此在以下實驗數據我們加比一項名為連續次長 CS，而以下括號內的值即為連續次長的值 CS，而括號旁的為連續最長 CS。

表 4- 19 BLOUSM62 矩陣應用於明確值之結果

序列名稱	使用方法	求解代數	平均 CS(%)	最高 CS(%)	最低 CS(%)	連續最長 CS(%)
lpht-1ycsB	GA	500	35	47.5	26.25	22.5(7.5)
lpht-1aboA	GA	500	35	37.5	31.25	15(8.75)
lpht-1ihvA	GA	500	18.75	27.5	8.75	7.5(3.75)
lpht-1vie	GA	500	26.25	47.5	15	12.5(3.75)

4.2.2將 BLOUSM62 矩陣應用於三角模糊數結果分析

在利用基因演算法求解時，我們的基因代數為 500 代，重複次數為 5 次，右偏歸屬函數範圍為[a-1,a,a+3]之間，左偏歸屬函數範圍為[a-3,a,a+1]之間以下，對稱歸屬函數範圍為[a-3,a,a+3]之間，以下為實驗結果。

表 4- 20 不同歸屬函數在基因 1pht 與基因 1ycsB 的排比

序列名稱	使用方法	求解代數	歸屬函數	平均 CS(%)	最高 CS(%)	最低 CS(%)	連續最長 CS(%)
1pht-1ycsB	GA	500	右偏	42.5	51.25	40	30
1pht-1ycsB	GA	500	左偏	38.75	45	26.25	27.5
1pht-1ycsB	GA	500	對稱	21.875	26.25	17.5	20

表 4- 21 不同歸屬函數在基因 1pht 與基因 1aboA 的排比

序列名稱	使用方法	求解代數	歸屬函數	平均 CS(%)	最高 CS(%)	最低 CS(%)	連續最長 CS(%)
lpht-1aboA	GA	500	右偏	35	52.5	23.75	18.75
lpht-1aboA	GA	500	左偏	35	45	25	12.5
lpht-1aboA	GA	500	對稱	32.5	35	30	11.25

表 4- 22 不同歸屬函數在基因 1pht 與基因 1ihvA 的排比

序列名稱	使用方法	求解代數	歸屬函數	平均 CS(%)	最高 CS(%)	最低 CS(%)	連續最長 CS(%)
lpht-1ihvA	GA	500	右偏	28.75	32.5	25	12.5
lpht-1ihvA	GA	500	左偏	22.5	35	15	10
lpht-1ihvA	GA	500	對稱	22.5	27.5	18.75	12.5

表 4- 23 不同歸屬函數在基因 1pht 與基因 1vie 的排比

序列名稱	使用方法	求解代數	歸屬函數	平均 CS(%)	最高 CS(%)	最低 CS(%)	連續最長 CS(%)
lpht-1vie	GA	500	右偏	20	37.5	12.5	12
lpht-1vie	GA	500	左偏	15	16.25	13.75	3.75
lpht-1vie	GA	500	對稱	31.25	36.25	25	18.75

表 4- 24 右偏三角歸屬函數

序列名稱	使用方法	求解代數	歸屬函數	平均CS(%)	最高CS(%)	最低CS(%)	連續最長CS(%)
1pht-1ycsB	GA	500	右偏	42.5	51.25	40	30
lpht-1aboA	GA	500	右偏	35	52.5	23.75	18.75
lpht-1ihvA	GA	500	右偏	28.75	32.5	25	12.5
lpht-1vie	GA	500	右偏	20	37.5	12.5	12

表 4- 25 左偏三角歸屬函數

序列名稱	使用方法	求解代數	歸屬函數	平均CS(%)	最高CS(%)	最低CS(%)	連續最長CS(%)
1pht-1ycsB	GA	500	左偏	38.75	45	26.25	27.5
lpht-1aboA	GA	500	左偏	35	45	25	12.5
lpht-1ihvA	GA	500	左偏	22.5	35	15	10
lpht-1vie	GA	500	左偏	15	16.25	13.75	3.75(

表 4- 26 對稱三角歸屬函數

序列名稱	使用方法	求解代數	歸屬函數	平均CS(%)	最高CS(%)	最低CS(%)	連續最長CS(%)
1pht-1ycsB	GA	500	對稱	21.875	26.25	17.5	20
lpht-1aboA	GA	500	對稱	32.5	35	30	11.25
lpht-1ihvA	GA	500	對稱	22.5	27.5	18.75	12.5
lpht-1vie	GA	500	對稱	31.25	36.25	25	18.75

4.2.3三角模糊數應用於 BLOUSM62 矩陣與明確值比較結果分析

以下數據是根據上表所整理，我們分別將模糊歸屬函數所求得的最佳解找出並整理資料如下表。

表 4- 27 利用 BLOUSM62 時的模糊最佳解與明確值最佳解比較表(連續最長 CS)

序列名稱	使用方法	求解代數	最佳歸屬函數	平均CS(%)	最高CS(%)	最低CS(%)	連續最長CS(%)
1pht-1ycsB	GA-fuzzy	500	右偏	42.5	51.25	40	30
lpht-1aboA	GA-fuzzy	500	右偏	35	52.5	23.75	18.75
lpht-1ihvA	GA-fuzzy	500	右偏	28.75	32.5	25	12.5
lpht-1vie	GA-fuzzy	500	對稱	31.25	36.25	25	18.75
1pht-1ycsB	GA-crisp	500	無	35	47.5	26.25	22.5
lpht-1aboA	GA-crisp	500	無	35	37.5	31.25	15
lpht-1ihvA	GA-crisp	500	無	18.75	27.5	8.75	7.5
lpht-1vie	GA-crisp	500	無	26.25	47.5	15	12.5

第五章 結論與建議

5.1 研究總結

在本研究中提出一種先進的方法來進行蛋白質序列排比。首先，我們使用模糊邏輯去建立模糊PAM250與BLOSUM62矩陣，接著我們使用模糊算數所估計出的分數去計算基因演算法中的適應函數值。實驗結果發現，不論應用哪種矩陣，GA-fuzzy皆能夠找到較長的連續基因配對序列，且由於PAM與BLOSUM矩陣中包含一些不確定因子，因此本實驗結果證明模糊邏輯確實能有效的處理不確定性問題並成功的應用在蛋白質序列排比上。這個先進的方法可在相關研究中，提供不同的觀點，以下為本研究之結論重點。

1. 模糊理論確實可以減少不確定環境的影響，因此即使利用較不適合的矩陣，所得到的結果波動不會過大，因此可知模糊矩陣確實可減少不確定因子的影響。
2. 我們進行基因序列排比時，全域排比(global alignment)所得到的解，並無明顯的差異，但在區域排比(local alignment)所得到的解明顯要比明確值矩陣好，因此我們可知道模糊矩陣因為考慮不確定因子的關係，可得到較好的區域排比解。
3. 基於不同歸屬函數會得到不同結果的原因下，每一種基因應該要有屬於自己的歸屬函數，如此才能更符合現實。
4. 若只對基因演算法每代的最佳解進行禁忌搜尋法，則所得結果並不會如預期的比單純基因演算法好。
5. 可在相關研究中，提供不同的觀點

5.2 後續研究建議

未來研究方向應可往以下幾點發展：

1. 為每個基因訂定專屬的歸屬函數。
2. 將模糊架構導入蛋白質結構分析領域內。
3. 將模糊架構導入演化數分析領域內。
4. 進行混合式 GA 演算法時，不只對每代最佳解進行禁記搜尋，而是將其他解同時考慮在內。

參考文獻

[1] 德利卡著，周業仁譯，(2004)。DNA 的 14 堂課，天下文化。

[2] 王文俊(1997)，認識 Fuzzy，全華科技圖書股份有限公司。

[3] Altschul, S.F. (1989). Gap costs for multiple sequence alignment. *Journal of Theoretical Biology*, 138, 297-309.

[4] Altschul, S.F. and Erickson, B.W. (1986). Optimal sequence alignment using affine gap costs. *Bulletin of Mathematical Biology*, 48, 603-616.

[5] Altschul, S.F. and Lipman, D. (1989). Trees, stars, and multiple biological sequence alignment. *SIAM Journal on Applied Mathematics*, 49, 197-209.

[6] Altschul, S.F., Carroll, R.J. and Lipman, D.J. (1989). Weights for data related by a tree. *Journal of Molecular Biology*, 207, 647-653.

[7] Altschul, S.F., Gish, W., Miller, W., Myers, E. and Lipman, D. (1990). A basic local alignment search tool. *Journal of Molecular Biology*, 215, 403-410.

[8] Altschul, S.F., Madden, T.L., Schaffer, A.A., Zhang, J., Zhang, Z., Miller, Z.W. and Lipman, D. (1997). Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research*, 25, 3389-3402.

[9] Behe M.J., Lattman E.E., Rose G.D. (1991). The protein-folding problem: The native fold determines packing, but does packing determine the native fold? Proceedings of the National Academy of Sciences of the United States of America, May 15, 88(10), 4195-4199.

[10] Blankenbecler, R., Ohlsson, M., Peterson, C. and Ringner, M. (2003). Matching protein structures with fuzzy alignments. Proceedings of the National Academy of Sciences of the United States of America, October 2, 100(21), 11936-11940.

[11] Carleos, C., Rodriguez, F., Lamelas, H. and Baro, J.A. (2003). Simulating complex traits influenced by genes with fuzzy-valued effects in pedigreed populations. *Bioinformatics*, 19, 144-148.

[12] Chang, P.T. (2003). Fuzzy strategic replacement analysis. *To appear in* European *Journal of Operational Research*.

[13] Chang, P.T., Lee, H.H., Chang, C.H. and Hung, K.C. (2003). A multi-simplexes and genetic algorithm hybrid method. *Submitted to European Journal of Operational Research.*

[14] Chang, P.T., Yao, M.J., Huang, S.F. and Chen, C.T. (2004). A genetic algorithm for solving a fuzzy economic lot-size scheduling problem. *Submitted to International Journal of Production Economics.*

[15] Chang, P.-T. (2005). Fuzzy strategic replacement analysis. *European Journal of Operational Research*. 160, 532–559.

[16] Chao, K.M. (1998). On computing all suboptimal alignments. *Information Sciences*, Mar., 105, (1-4), 189-207.

[17] Chao, K.M. and Miller, W. (1995). Linear-space algorithms that build local alignments from fragments. *Algorithmica*, 13, 106-134.

[18] Chao, K.M. Zhang, J., Ostell, J. and Miller, W. (1997). A tool for aligning very similar DNA sequences. *Computer Applications in the Biosciences*, 13, 75-80.

[19] Chao, K.M., Hardison, R.C. and Miller, W. (1994). Recent developments in linear-space alignment methods: A survey. *Journal of Computational Biology*, 1, 271-291.

[20] Creighton, T.E. (1990). Protein folding. *Biochemistry Journal*, 270, 1-16.

[21] Dayhoff, M.O., Schwartz, R.M. and Orcutt, B.C. (1978). A model of evolutionary change in proteins. *Atlas of Protein Sequence and Structure,* 5, 345-352.

[22] Dubois, D., and Prade, H. (1980). Fuzzy Sets and Systems. *Theory and Applications* (Academic Press).

[23] Edwards, Y.J. and Cottage, A. (2003). Bioinformatics methods to predict protein structure and function: A practical approach. *Molecular Biotechnology*, Feb., 23(2), 139-66.

[24] Farach, M., Kannan, S. and Warnow, T. (1995). A robust model for finding optimal evolutionary trees. *Algorithmica*, 13, 155-179

[25] Feng, D.F. and Doolittle, R.F. (1987). Progressive sequence alignment as a prerequisite to correct phylogenetic trees. *Journal of Molecular Biology*, 25, 351-360.

[26] Goldberg D.E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Publishion.

[27] Gribskov, M., McLachlan, M. and Eisenberg, D. (1987). Profile analysis: Detection of distantly related proteins. Proceedings of the National Academy of Sciences of the United States of America, 84, 4355-4358.

[28] Henikoff, S. and Henikoff, J.G. (1992). Amino-Acid Substitution Matrices From Protein Blocks. Proceedings of the National Academy of Sciences of the United States of America, 89(22), 10915-10919.

[29] Heger, A. and Holm, L. (2003). Sensitive pattern discovery with 'fuzzy' alignments of distantly related proteins. *Bioinformatics*, 19, i130-i137.

[30] Huang, X. and Miller, W. (1991). A time-efficient, linear-space local similarity algorithm. *Advances in Applied Mathematics*, 12, 337-357.

[31] Huang, Y. and Li, Y, (2004). *Prediction of protein subcellular locations using fuzzy k-NN method. Bioinformatics, 20, 21-28.*

[32] Hung D.N., Ikuo Y., Kunihito Y. and Moritoshi Y. (2002). Aligning Multiple Protein Sequences by Parallel HybridGenetic Algorithm. *Genome Informatics*, 13, 123-132.

[33] Kaufmann, A., Gupta, M.M. (1988). *Fuzzy Mathematical Models in Engineering and Management Science* (Elsevier).

[34] Krogh, A. and Mitchison, G.. (1995). Maximum entropy weighting of aligned sequences of proteins or DNA. Third International Conference on Intelligent Systems for Molecular Biology (ISMB), Cambridge, England. AAAI Press, Menlo Park, CA.

[35] Lee, E.S. and Li, R.L. (1988). Comparison of fuzzy numbers based on the probability measure of fuzzy events. *Computers and Mathematics with Applications*, 15, 887-896.

[36] Michalewicz Z. (1992). *Genetic Algorithms + Data Structures = Evolution Programs, Springer Verlag*, New York, NY.

[37] Michalewicz Z., et al. (1994). Evolutionary operations for continuous convex parameter spaces. Proceedings of the Third Annual Conference on Evolutionary Programming, 84-97.

[38] Mizumoto, M., and Tanaka, K. (1976). The four operations of arithmetic on fuzzy numbers. *Systems Computers Controls*, 7, 73–81.

[39] Murata T. and Ishibuchi, H. (1994). Performance evaluation of genetic algorithms for flowshop scheduling problems. Proceedings of 1st IEEE Conf. Evolutionary Computation, Orlando.

[40] Nahmias, S., (1978), Fuzzy variables. *Fuzzy Sets and Systems*, 1, 97-111.

[41] Needleman, B. and Wunsch, C.D. (1970). A general method applicable to the search for similarities in the amino acid sequences of two proteins. *Journal of Molecular Biology*, 48, 443-453.

[42] Nieto, J.J. and Torres, A. (2003). Midpoints for fuzzy sets and their application in medicine. *Artificial Intelligence in Medicine*, 27, 81-101.

[43] Pearson, W.R. and Lipman, D. (1988). Improved tools for biological sequence comparison. Proceedings of the National Academy of Sciences of the United States of America, 85, 2444-2448.

[44] Pevzner, P.A. (1995). DNA physical mapping and alternating Eulerian cycles in colored graphs. *Algorithmica*, 13, 77-105.

[45] Raiz, T., Yi, W. and Li, K.B. (2005). A tabu search algorithm for post-processing multiple sequence alignment. *Journal of Bioinformatics and Computational Biology*, 3(1), 145-156.

[46] Smith T.F. and Waterman M.S. (1981). Identification of common molecular subsequences. *Journal of Molecular Biology*. 197, 723-728.

[47] Thompson, J.D., Higgins, D.G. and Gibson, T.J. (1994). Improved sensitivity of profile searches through the use of sequence weights and gab excision. *Computer Applications in the Biosciences*, 10, 19-29.

[48] Thompson, J. D., Plewniak, F., and Poch, O. (1999). BAliBASE: A benchmark alignment database for the evaluation of multiple alignment programs, *Bioinformatics*, 15, 87-88.

[49] Thompson, J.D., Plewniak, F. and Poch, O. (1999). A comprehensive comparison of multiple sequence alignment programs, *Nucleic Acids Research*, 27, 2682-2690.

[50] Torres, A. and Nieto, J.J. (2003). The fuzzy polynucleotide space: basic properties. *Bioinformatics*. 19, 587-592.

[51] Wu, Y., Lancia, G., Bafna, V., Chao, K.M., Ravi, R. and Tang, C.Y. (1998). A

polynomial-time approximation scheme for minimum routing cost spanning trees. *Symposium on Discrete Algorithms*, 8, 21-32.

[52] Xu, Y., Mural, R. and Uberbacher, E.C. (1994). Constructing gene models from a set of accurately-predicted exons: An application of dynamic programming. *Computer Applications in the Biosciences*, 10, 613-623.

[53] Zadeh, L.A. (1965). Fuzzy Sets. *Information and Control*, 8, 338–353.

[54] Zhang, J. and Madden, T.L. (1997). PowerBLAST: a new network BLAST application for interactive or automated sequence analysis and annotation. *Genome Methods*, 649-656.

附錄

附錄一 得分矩陣

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

BLOUSM30 矩陣

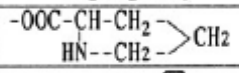
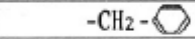
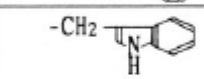
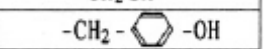
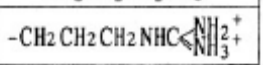
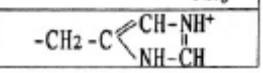
	A	C	D	E	F	G	H	I	K	L	M	N	P	Q	R	S	T	V	W	Y	*
A	4	0	-2	-1	-2	0	-2	-1	-1	-1	-1	-1	-1	-1	-1	1	0	0	-3	-2	-4
C	0	9	-3	-3	-2	-2	-3	-1	-3	-1	-1	-2	-3	-3	-3	-1	-1	-1	-2	-2	-4
D	-2	-3	6	2	-3	-1	-1	-3	-1	-3	-3	1	-1	0	-1	0	-1	-3	-4	-3	-4
E	-1	-3	2	5	-3	-2	0	-3	1	-3	-2	0	-1	2	0	0	-1	-2	-3	-2	-4
F	-2	-2	-3	-3	6	-3	-1	0	-3	0	0	-3	-4	-3	-3	-2	-2	-1	1	3	-4
G	0	-2	-1	-2	-3	6	-2	-3	-1	-4	-2	0	-2	-2	-2	0	-2	-3	-2	-3	-4
H	-2	-3	-1	0	-1	-2	7	-3	-1	-3	-1	1	-2	1	0	-1	-2	-3	-2	2	-4
I	-1	-1	-3	-3	0	-3	-3	4	-3	2	1	-3	-3	-3	-3	-2	-1	3	-2	-1	-4
K	-1	-3	-1	1	-3	-1	-1	-3	4	-2	-1	0	-1	1	2	0	-1	-2	-3	-2	-4
L	-1	-1	-3	-3	0	-4	-3	2	-2	4	2	-3	-3	-2	-2	-2	-1	1	-2	-1	-4
M	-1	-1	-3	-2	0	-2	-1	1	-1	2	5	-2	-2	0	-1	-1	-1	1	-1	-1	-4
N	-1	-2	1	0	-3	0	1	-3	0	-3	-2	6	-2	0	0	1	0	-3	-4	-2	-4
P	-1	-3	-1	-1	-4	-2	-2	-3	-1	-3	-2	-2	7	-1	-2	-1	-1	-2	-4	-3	-4
Q	-1	-3	0	2	-3	-2	1	-3	1	-2	0	0	-1	5	1	0	-1	-2	-2	-1	-4
R	-1	-3	-1	0	-3	-2	0	-3	2	-2	-1	0	-2	1	5	-1	-1	-2	-3	-2	-4
S	1	-1	0	0	-2	0	-1	-2	0	-2	-1	1	-1	0	-1	4	1	-2	-3	-2	-4
T	0	-1	-1	-1	-2	-2	-2	-1	-1	-1	-1	0	-1	-1	-1	1	4	0	-2	-2	-4
V	0	-1	-3	-2	-1	-3	-3	3	-2	1	1	-3	-2	-2	-2	-2	0	4	-3	-1	-4
W	-3	-2	-4	-3	1	-2	-2	-2	-3	-2	-1	-4	-4	-2	-3	-3	-2	-3	10	2	-4
Y	-2	-2	-3	-2	3	-3	2	-1	-2	-1	-1	-2	-3	-1	-2	-2	-2	-1	2	6	-4
*	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	1

BLOUSM60 矩陣

	A	C	D	E	F	G	H	I	K	L	M	N	P	Q	R	S	T	V	W	Y	*
A	5	-1	-3	-1	-3	0	-2	-2	-1	-2	-2	-2	-1	-1	-2	1	0	-1	-4	-3	-6
C	-1	9	-5	-6	-3	-4	-5	-2	-4	-2	-2	-4	-4	-4	-5	-2	-2	-2	-4	-4	-6
D	-3	-5	7	1	-5	-2	-2	-5	-1	-5	-4	1	-3	-1	-3	-1	-2	-5	-6	-4	-6
E	-1	-6	1	6	-5	-3	-1	-4	0	-4	-3	-1	-2	2	-1	-1	-1	-3	-5	-4	-6
F	-3	-3	-5	-5	7	-5	-2	-1	-4	0	-1	-4	-4	-4	-4	-3	-3	-2	0	3	-6
G	0	-4	-2	-3	-5	6	-3	-5	-2	-5	-4	-1	-3	-3	-3	-1	-3	-5	-4	-5	-6
H	-2	-5	-2	-1	-2	-3	8	-4	-1	-4	-3	0	-3	1	0	-2	-2	-4	-3	1	-6
I	-2	-2	-5	-4	-1	-5	-4	5	-4	1	1	-4	-4	-4	-4	-3	-1	3	-4	-2	-6
K	-1	-4	-1	0	-4	-2	-1	-4	6	-3	-2	0	-2	1	2	-1	-1	-3	-5	-3	-6
L	-2	-2	-5	-4	0	-5	-4	1	-3	5	2	-4	-4	-3	-3	-3	-2	0	-3	-2	-6
M	-2	-2	-4	-3	-1	-4	-3	1	-2	2	7	-3	-3	0	-2	-2	-1	0	-2	-2	-6
N	-2	-4	1	-1	-4	-1	0	-4	0	-4	-3	7	-3	0	-1	0	0	-4	-5	-3	-6
P	-1	-4	-3	-2	-4	-3	-3	-4	-2	-4	-3	-3	8	-2	-3	-2	-2	-3	-5	-4	-6
Q	-1	-4	-1	2	-4	-3	1	-4	1	-3	0	0	-2	7	1	-1	-1	-3	-3	-3	-6
R	-2	-5	-3	-1	-4	-3	0	-4	2	-3	-2	-1	-3	1	6	-1	-2	-3	-4	-3	-6
S	1	-2	-1	-1	-3	-1	-2	-3	-1	-3	-2	0	-2	-1	-1	5	1	-2	-4	-3	-6
T	0	-2	-2	-1	-3	-3	-2	-1	-1	-2	-1	0	-2	-1	-2	1	6	-1	-4	-2	-6
V	-1	-2	-5	-3	-2	-5	-4	3	-3	0	0	-4	-3	-3	-3	-2	-1	5	-3	-3	-6
W	-4	-4	-6	-5	0	-4	-3	-4	-5	-3	-2	-5	-5	-3	-4	-4	-4	-3	11	2	-6
Y	-3	-4	-4	-4	3	-5	1	-2	-3	-2	-2	-3	-4	-3	-3	-3	-2	-3	2	8	-6

BLOUSM90 矩陣

附錄二 胺基酸名稱、支鏈構造及縮寫

氨基酸側鏈(R)的結構			
氨基酸	3字符號	1字符號	R
不含極性的氨基酸			
甘氨酸	Gly	G	-H
丙氨酸	Ala	A	$-CH_3$
纈氨酸	Val	V	$-CH(CH_3)_2$
亮氨酸	Leu	L	$-CH_2CH(CH_3)_2$
異亮氨酸	Ile	I	$-CH(CH_3)CH_2CH_3$
蛋氨酸	Met	M	$-CH_2CH_2SCH_3$
脯氨酸	Pro	P	
苯丙氨酸	Phe	F	
色氨酸	Trp	W	
含有極性的中性氨基酸			
絲氨酸	Ser	S	$-CH_2OH$
蘇氨酸	Thr	T	$-CHOHCH_3$
天冬醯胺	Asn	N	$-CH_2CONH_2$
谷氨醯胺	Aln	Q	$-CH_2CH_2CONH_2$
半胱氨酸	Cys	C	$-CH_2SH$
酪氨酸	Tyr	Y	
酸性氨基酸			
天冬氨酸	Asp	D	$-CH_2COO^-$
谷氨酸	Glu	E	$-CH_2CH_2COO^-$
碱性氨基酸			
賴氨酸	Lys	K	$-CH_2CH_2CH_2NH_3^+$
精氨酸	Arg	R	
組氨酸	His	H	

二十種胺基酸之名稱、支鏈構造及縮寫

附錄三 模糊計分矩陣

ORIGINAL AMINO ACID

REPLACEMENT AMINO ACID	A Alg	R Arg	N Asn	D Asp	C Cys	Q Gln	E Glu	G Gly	H His	I Ile	L Leu	K Lys	M Met	F Phe	P Pro	S T Ser	W Thr	Y Trp	V Tyr	Val
A	[1,2,5]	[-3,-2,1]	[-1,0,3]	[-1,0,3]	[-3,-2,1]	[-1,0,3]	[-1,0,3]	[0,1,4]	[-2,-1,2]	[-2,-1,2]	[-3,-2,2]	[-2,-1,2]	[-2,-1,2]	[-4,-3,0]	[0,1,4]	[0,1,4]	[0,1,4]	[-7,-6,-3]	[-4,-3,0]	[-1,0,3]
R		[5,6,9]	[-1,0,3]	[-2,-1,2]	[-5,-4,-1]	[0,1,4]	[-2,-1,2]	[-4,-3,0]	[1,2,5]	[-3,-2,1]	[-4,-3,0]	[2,3,6]	[-1,0,3]	[-5,-4,-1]	[-1,0,3]	[-1,0,3]	[-2,-1,2]	[1,2,5]	[-5,-4,-1]	[-3,-2,1]
N			[1,2,5]	[1,2,5]	[-5,-4,-1]	[0,1,4]	[0,1,4]	[-1,0,3]	[1,2,5]	[-3,-2,1]	[-4,-3,0]	[0,1,4]	[-3,-2,1]	[-4,-3,0]	[-1,0,3]	[0,1,4]	[-1,0,3]	[-5,-4,-1]	[-3,-2,1]	[-3,-2,1]
D				[3,4,7]	[-6,-5,-2]	[1,2,5]	[2,3,6]	[0,1,4]	[0,1,4]	[-3,-2,1]	[-5,-4,-1]	[-1,0,3]	[-4,-3,0]	[-7,-6,-3]	[-2,-1,2]	[-1,0,3]	[-1,0,3]	[-8,-7,-4]	[-5,-4,-1]	[-3,-2,1]
C					[11,12,15]	[-6,-5,-2]	[-6,-5,-2]	[-4,-3,0]	[-4,-3,0]	[-3,-2,1]	[-7,-6,-3]	[-6,-5,-2]	[-6,-5,-2]	[-5,-4,-1]	[-4,-3,0]	[-1,0,3]	[-3,-2,1]	[-9,-8,-5]	[-1,0,3]	[-3,-2,1]
Q						[3,4,7]	[1,2,5]	[-2,-1,2]	[2,3,6]	[-3,-2,1]	[-3,-2,1]	[0,1,4]	[-2,-1,2]	[-6,-5,-2]	[-1,0,3]	[-2,-1,2]	[-2,-1,2]	[-6,-5,-2]	[-5,-4,-1]	[-3,-2,1]
E							[3,4,7]	[-1,0,3]	[0,1,4]	[-3,-2,1]	[-4,-3,0]	[-1,0,3]	[-3,-2,1]	[-6,-5,-2]	[-2,-1,2]	[-1,0,3]	[-1,0,3]	[-8,-7,-4]	[-5,-4,-1]	[-3,-2,1]
G								[4,5,8]	[-3,-2,1]	[-4,-3,0]	[-5,-4,-1]	[-3,-2,1]	[-4,-3,0]	[-6,-5,-2]	[-1,0,3]	[0,1,4]	[-1,0,3]	[-8,-7,-4]	[-6,-5,-2]	[-2,-1,2]
H									[5,6,9]	[-3,-2,1]	[-3,-2,1]	[-1,0,3]	[-3,-2,1]	[-3,-2,1]	[-1,0,3]	[-2,-1,2]	[-2,-1,2]	[-4,-3,0]	[-1,0,3]	[-3,-2,1]
I										[4,5,8]	[1,2,5]	[-3,-2,1]	[1,2,5]	[0,1,4]	[-3,-2,1]	[-2,-1,2]	[-1,0,3]	[-6,-5,-2]	[-2,-1,2]	[3,4,7]
L											[5,6,9]	[-4,-3,0]	[3,4,7]	[1,2,5]	[-4,-3,0]	[-4,-3,0]	[-3,-2,1]	[-3,-2,1]	[-2,-1,2]	[1,2,5]
K												[4,5,8]	[-1,0,3]	[-6,-5,-2]	[-2,-1,2]	[-1,0,3]	[-1,0,3]	[-4,-3,0]	[-5,-4,-1]	[-3,-2,1]
M													[5,6,9]	[-1,0,3]	[-3,-2,1]	[-3,-2,1]	[-2,-1,2]	[-5,-4,-1]	[-3,-2,1]	[1,2,5]
F														[8,9,12]	[-6,-5,-2]	[-4,-3,0]	[-4,-3,0]	[-1,0,3]	[6,7,10]	[-2,-1,2]
P															[5,6,9]	[0,1,4]	[-1,0,3]	[-7,-6,-3]	[-6,-5,-2]	[-2,-1,2]
S																[1,2,5]	[0,1,4]	[-3,-2,1]	[-4,-3,0]	[-2,-1,2]
T																	[2,3,6]	[-6,-5,-2]	[-4,-3,0]	[-1,0,3]
W																		[16,17,20]	[-1,0,3]	[-7,-6,-3]
Y																			[-4,-3,0]	[-1,0,3]
V																				[-3,-2,1]

模糊右偏 PAM250 矩陣

ORIGINAL AMINO ACID

REPLACEMENT AMINO ACID	A Alg	R Arg	N Asn	D Asp	C Cys	Q Gln	E Glu	G Gly	H His	I Ile	L Leu	K Lys	M Met	F Phe	P Pro	S Ser	T Thr	W Trp	Y Tyr	V Val
A	[-1,2,3]	[-5,-2,-1]	[-3,0,1]	[-3,0,1]	[-5,-2,-1]	[-3,0,1]	[-3,0,1]	[-3,1,2]	[-4,-1,0]	[-4,-1,0]	[-5,-2,1]	[-4,-1,0]	[-4,-1,0]	[-6,-3,-2]	[-2,1,2]	[-2,1,2]	[-2,1,2]	[-9,-6,-5]	[-6,-3,-2]	[-3,0,1]
R		[3,6,7]	[-3,0,1]	[-4,-1,0]	[-7,-4,-3]	[-3,1,2]	[-4,-1,0]	[-6,-3,-2]	[-1,2,3]	[-5,-2,1]	[-6,-3,-2]	[0,3,4]	[-3,0,1]	[-7,-4,-3]	[-3,0,1]	[-3,0,1]	[-4,-1,0]	[-1,2,3]	[-7,-4,-3]	[-5,-2,1]
N			[-1,2,3]	[-1,2,3]	[-7,-4,-3]	[-3,1,2]	[-3,1,2]	[-3,0,1]	[-1,2,3]	[-5,-2,1]	[-6,-3,-2]	[-3,1,2]	[-5,-2,1]	[-6,-3,-2]	[-3,0,1]	[-3,1,2]	[-3,0,1]	[-7,-4,-3]	[-5,-2,1]	[-5,-2,1]
D				[1,4,5]	[-8,-5,-4]	[-1,2,3]	[0,3,4]	[-2,1,2]	[-3,1,2]	[-5,-2,1]	[-7,-4,-3]	[-3,0,1]	[-6,-3,-2]	[-9,-6,-5]	[-4,-1,0]	[-3,0,1]	[-3,0,1]	[-10,-7,-6]	[-7,-4,-3]	[-5,-2,1]
C					[9,12,13]	[-8,-5,-4]	[-8,-5,-4]	[-6,-3,-2]	[-6,-3,-2]	[-5,-2,1]	[-9,-6,-5]	[-8,-5,-4]	[-8,-5,-4]	[-7,-4,-3]	[-6,-3,-2]	[-3,0,1]	[-5,-2,1]	[-11,-8,-7]	[-3,0,1]	[-5,-2,1]
Q						[1,4,5]	[-1,2,3]	[-4,-1,0]	[0,3,4]	[-5,-2,1]	[-5,-2,1]	[-3,1,2]	[-4,-1,0]	[-8,-5,-4]	[-3,0,1]	[-4,-1,0]	[-4,-1,0]	[-8,-5,-4]	[-7,-4,-3]	[-5,-2,1]
E							[1,4,5]	[-3,0,1]	[-3,1,2]	[-5,-2,1]	[-6,-3,-2]	[-3,0,1]	[-5,-2,1]	[-8,-5,-4]	[-4,-1,0]	[-3,0,1]	[-3,0,1]	[-10,-7,-6]	[-7,-4,-3]	[-5,-2,1]
G								[2,5,6]	[-5,-2,1]	[-6,-3,-2]	[-7,-4,-3]	[-5,-2,1]	[-6,-3,-2]	[-8,-5,-4]	[-3,0,1]	[-3,1,2]	[-3,0,1]	[-10,-7,-6]	[-8,-5,-4]	[-4,-1,0]
H									[3,6,7]	[-5,-2,1]	[-5,-2,1]	[-3,0,1]	[-5,-2,1]	[-5,-2,1]	[-3,0,1]	[-4,-1,0]	[-4,-1,0]	[-6,-3,-2]	[-3,0,1]	[-5,-2,1]
I										[2,5,6]	[-1,2,3]	[-5,-2,1]	[-1,2,3]	[-3,1,2]	[-5,-2,-1]	[-4,-1,0]	[-3,0,1]	[-8,-5,-4]	[-4,-1,0]	[1,4,5]
L											[3,6,7]	[-6,-3,-2]	[1,4,5]	[-1,2,3]	[-6,-3,-2]	[,-3,]	[-5,-2,1]	[-5,-2,1]	[-4,-1,0]	[-1,2,3]
K												[2,5,6]	[-3,0,1]	[-8,-5,-4]	[-4,-1,0]	[-3,0,1]	[-3,0,1]	[-6,-3,-2]	[-7,-4,-3]	[-5,-2,1]
M													[3,6,7]	[-3,0,1]	[-5,-2,1]	[-5,-2,1]	[-4,-1,0]	[-7,-4,-3]	[-5,-2,1]	[-1,2,3]
F														[6,9,10]	[-8,-5,-4]	[-6,-3,-2]	[-6,-3,-2]	[-3,0,1]	[4,7,8]	[-4,-1,0]
P															[3,6,7]	[-3,1,2]	[-3,0,1]	[-9,-6,-5]	[-8,-5,-4]	[-4,-1,0]
S																[-1,2,3]	[-3,1,2]	[-5,-2,1]	[-6,-3,-2]	[-4,-1,0]
T																	[0,3,4]	[,-5,]	[-6,-3,-2]	[-3,0,1]
W																		[,17,]	[-3,0,1]	[,-6,]
Y																			[,-3,]	[-3,0,1]
V																				[,-2,]

模糊左偏 PAM250 矩陣

ORIGINAL AMINO ACID

REPLACEMENT AMINO ACID

	A Alg	R Arg	N Asn	D Asp	C Cys	Q Gln	E Glu	G Gly	H His	I Ile	L Leu	K Lys	M Met	F Phe	P Pro	S T Ser	W Thr	Y Trp	V Tyr	Val
A	[-1,2,5]	[-5,-2,1]	[-3,0,3]	[-3,0,3]	[-5,-2,1]	[-3,0,3]	[-3,0,3]	[-2,1,4]	[-4,-1,2]	[-4,-1,2]	[-5,-2,1]	[-4,-1,2]	[-4,-1,2]	[-6,-3,0]	[-2,1,4]	[-2,1,4]	[-2,1,4]	[-9,-6,-3]	[-6,-3,0]	[-3,0,3]
R		[3,6,9]	[-3,0,3]	[-4,-1,2]	[-7,-4,-1]	[-2,1,4]	[-4,-1,2]	[-6,-3,0]	[-1,2,5]	[-5,-2,1]	[-6,-3,0]	[0,3,6]	[-3,0,3]	[-7,-4,-1]	[-3,0,3]	[-3,0,3]	[-4,-1,2]	[-1,2,5]	[-7,-4,-1]	[-5,-2,1]
N			[-1,2,5]	[-1,2,5]	[-7,-4,-1]	[-2,1,4]	[-2,1,4]	[-3,0,3]	[-1,2,5]	[-5,-2,1]	[-6,-3,0]	[-2,1,4]	[-5,-2,1]	[-6,-3,0]	[-3,0,3]	[-2,1,4]	[-3,0,3]	[-7,-4,-1]	[-5,-2,1]	[-5,-2,1]
D				[1,4,7]	[-8,-5,-2]	[-1,2,5]	[0,3,6]	[-2,1,4]	[-2,1,4]	[-5,-2,1]	[-7,-4,-1]	[-3,0,3]	[-6,-3,0]	[-9,-6,-3]	[-4,-1,2]	[-3,0,3]	[-3,0,3]	[-10,-7,-4]	[-7,-4,-1]	[-5,-2,1]
C					[9,12,15]	[-8,-5,-2]	[-8,-5,-2]	[-6,-3,0]	[-6,-3,0]	[-5,-2,1]	[-9,-6,-3]	[-8,-5,-2]	[-8,-5,-2]	[-7,-4,-1]	[-6,-3,0]	[-3,0,3]	[-5,-2,1]	[-11,-8,-5]	[-3,0,3]	[-5,-2,1]
Q						[1,4,7]	[-1,2,5]	[-4,-1,2]	[0,3,6]	[-5,-2,1]	[-5,-2,1]	[-2,1,4]	[-4,-1,2]	[-8,-5,-2]	[-3,0,3]	[-4,-1,2]	[-4,-1,2]	[-8,-5,-2]	[-7,-4,-1]	[-5,-2,1]
E							[1,4,7]	[-3,0,3]	[-2,1,4]	[-5,-2,1]	[-6,-3,0]	[-3,0,3]	[-5,-2,1]	[-8,-5,-2]	[-4,-1,2]	[-3,0,3]	[-3,0,3]	[-10,-7,-4]	[-7,-4,-1]	[-5,-2,1]
G								[2,5,8]	[-5,-2,1]	[-6,-3,0]	[-7,-4,-1]	[-5,-2,1]	[-6,-3,0]	[-8,-5,-2]	[-3,0,3]	[-2,1,4]	[-3,0,3]	[-10,-7,-4]	[-8,-5,-2]	[-4,-1,2]
H									[3,6,9]	[-5,-2,1]	[-5,-2,1]	[-3,0,3]	[-5,-2,1]	[-5,-2,1]	[-3,0,3]	[-4,-1,2]	[-4,-1,2]	[-6,-3,0]	[-3,0,3]	[-5,-2,1]
I										[2,5,8]	[-1,2,5]	[-5,-2,1]	[-1,2,5]	[-2,1,4]	[-5,-2,1]	[-4,-1,2]	[-3,0,3]	[-8,-5,-2]	[-4,-1,2]	[1,4,7]
L											[3,6,9]	[-6,-3,0]	[1,4,7]	[-1,2,5]	[-6,-3,0]	[-6,-3,0]	[-5,-2,1]	[-5,-2,1]	[-4,-1,2]	[-1,2,5]
K												[2,5,8]	[-3,0,3]	[-8,-5,-2]	[-4,-1,2]	[-3,0,3]	[-3,0,3]	[-6,-3,0]	[-7,-4,-1]	[-5,-2,1]
M													[,6,]	[-3,0,3]	[-5,-2,1]	[-5,-2,1]	[-4,-1,2]	[-7,-4,-1]	[-5,-2,1]	[-1,2,5]
F														[6,9,12]	[-8,-5,-2]	[-6,-3,0]	[-6,-3,0]	[-3,0,3]	[4,7,10]	[-4,-1,2]
P															[3,6,9]	[-2,1,4]	[-3,0,3]	[-9,-6,-3]	[-8,-5,-2]	[-4,-1,2]
S																[-1,2,5]	[-2,1,4]	[-5,-2,1]	[-6,-3,0]	[-4,-1,2]
T																	[0,3,6]	[-8,-5,-2]	[-6,-3,0]	[-3,0,3]
W																		[14,17,20]	[-3,0,3]	[-9,-6,-3]
Y																			[-6,-3,0]	[-3,0,3]
V																				[-5,-2,1]

模糊對稱 PAM250 矩陣