直交調和搜尋最佳化演算法

學生：郭禎祥　　指導教授：張炳騰 博士
曾宗瑤 博士

東海大學工業工程與經營資訊研究所

摘要

調和搜尋演算法(Harmony Search; HS)是將樂團演奏表現調整至最協調且美妙的現象引用到最佳化演算系統當中，而發展出一套全新的啟發式演算法。然而在最佳化問題的求解過程中，如何有效率的搜尋到目標函數的全域最佳解並不是一件容易的事，而調和搜尋演算法有不錯的搜尋精確度，但調和搜尋演算法由於微調方向依賴隨機化的方式求得，使得搜尋速度較慢，求解時間上消耗多，這對於決策者而言，是求解過程中的最大問題。

本研究提出直交調和搜尋演算法(Orthogonal Harmony Search; OHS)，主要透過結構上重新修正並應用直交表實驗設計(Orthogonal Experimental Design; OED)之技術以改良調和搜尋演算法。直交調和搜尋演算法主要以三項機制運作搜尋：(1)以直交交配來做全域搜尋(exploration)，(2)以直交微調來做區域搜尋(exploitation)，(3)以隨機方式尋找其他可行解。因此，直交調和搜尋演算法承襲調和搜尋演算法的搜尋精確度，結合直交表實驗設計具有優良經驗的推理能力與主效果分析，促使演算法在廣大的搜尋空間中，能夠為子代判斷正確的搜尋區域與方向，更快速向最佳解逼近以達到收斂並強化搜尋最佳解的能力。

本研究於直交表改良演算法的測試函數中，選出八個函數做實驗，實驗函數包含單一區域解與多重區域解問題。實驗結果除了與調和搜尋演算法比較之外，並與直交基因演算法(Orthogonal Genetic Algorithm; OGA)、直交模擬退火演算法(Orthogonal Simulated Annealing Algorithm; OSA)比較，整體結果直交調和搜尋演算法優於其他演算法，並證實本研究所提出之演算法在各種測試函數中能迅速且穩定向最佳解逼近。

關鍵字：調和搜尋、最佳化、收斂效率、直交表實驗設計、全域搜尋

An Orthogonal Harmony Search for Function Optimization

Student: Chen-Hsiang Kuo Advisor: Dr. Ping-Teng Chang
Dr. Tsueng-Yao Tseng

Institute of Industrial Engineering and Enterprise Information
Tunghai University

ABSTRACT

Harmony search is a new heuristic algorithm, and it is conceptualized using the musical process of searching for a perfect state of harmony. How to search the global optimization solution efficiently in object function is difficult in the process of solving optimization problems. Harmony search have ability to find solutions closer to the optima but it consumes a lot of time. And it's a big problem of HS for decision makers.

This paper brings up an Orthogonal Harmony Search (OHS). The main focal point of OHS is to revise harmony search and apply method of orthogonal experimental design (OED) to enhance it. OHS has three operators: (1) orthogonal clossover for global search, (2) orthogonal pitch adjustment for local search, (3) random search for feasible space. OHS use the systematic reasoning methods, OED and factor analysis, to find the right direction to approach the optimal solution and speed the search ability.

We execute the proposed algorithm to solve eight test functions include of unimodal and multi-modal. Comparing with HS, OGA and OHS, we can find that OHS can quicker slove problems than them and more stable find optimal or close-to-optimal solutions.

Keywords: Harmony Search; optimization; convergence efficiency; orthogonal experimental design; global search.

誌謝

首先要感謝我的指導教授張炳騰老師，在他細心的指導下，給予我在寫論文時有相當大的幫助，在未來工作態度及待人處事上，給予很多建議，亦師亦友的情誼與師恩，永誌難忘。口試期間，承蒙曾宗瑤老師、白炳豐老師及陳琨太老師在百忙之中特別撥空審閱論文，提供寶貴的意見，使我的論文得以更加完整，也讓我瞭解做研究所必須有的嚴謹態度，由衷的感謝各位老師。

在研究過程中，感謝國禎學長與龍廷全力協助與支持，國平學長、志昇學長、欣怡學姊與上屆學長敬淳、尊仁、俊嘉、文偉、瑋珊學姊精神上的鼓勵及兩年來的照顧。感謝研究室同學國丞、香君、鼎翰，大家互相鼓勵一同走過最艱難的時光，有你們真好！此外，感謝學弟妹季詮、敬予、亦群、正明、曉嬋一年來的支持與陪伴，真的挺到了！感謝！

最後，也是最重要的，感謝父母家人及女友的體貼與關懷，使我能在溫馨無慮的環境下，順利完成學業。

在此，僅將此喜悅與所有關心我及支持我的人共同分享，謝謝你們！

郭禎祥 謹致於

東海大學工業工程與經營資訊學系研究所

民國九十五年六月

目錄

表目錄

圖目錄

第一章 緒論

1.1 研究背景與動機

在最佳化問題的求解過程中，要搜尋到全域解並不是一件容易的事，基因演算法(Genetic Algorithms; GAs)是採用隨機搜尋的方式，雖然可以很快的對全域做搜尋，找到極值附近的區域解，但想從最佳解附近逼近到最佳解上時，卻需要耗費很多的時間去演化最佳的解。另一方面，調和搜尋演算法雖然可以穩定的往最佳解的方向逼近且在細部搜尋上有較佳的精確度，但對於分秒必爭的決策者而言，調和搜尋演算法求解速度太慢，乃最佳化求解過程中的最大問題。

由於不同問題和不同領域並不是都能用同一種方法來解決，因此有許多啟發式演算法用以解決實務問題，這些演算法於 2.1 節會做整理與簡介，雖然這些演算法已有許多研究證實其優異的特性，但是如何能設計出更簡易、更有搜尋效能的演算法來解決實務問題，為本研究最大的動機。

1.2 研究目的

為了讓調和搜尋演算法在搜尋時間上能夠更有效率，因此結合了直交表實驗設計的方法與調和搜尋演算法成為本研究之研究模型-直交調和搜尋演算法。OHS 主要是承襲 HS 本來具有的全域及區域解搜尋能力，結合 OED 具有優良經驗的推理能力與主效果分析，促使在廣大的搜尋空間中，能夠為子代判斷正確的搜尋方向，以達到快速收斂並強化搜尋最佳解的能力。

從其他直交演算法的文獻中，我們找出許多常被引用的測試函數，期望從這些不同性質的測試函數中，實驗出 HS 的特性與本研究所提出的 OHS 之差異與精進的效益，進而比較近年來發表與直交表相關的改良演算法，包括有直交基因演算法、直交模擬退火演算法，比較其搜尋效能，期望 OHS 搜尋能力上有更優異的表現。

1.3 研究架構

本研究分為以下幾個流程步驟，研究的流程與架構如圖1.1：

1. 問題探討與確立目標

   探討調和搜尋演算法對於函數最佳解搜尋方法的問題，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. 研究方法

   本章將詳細描述本研究所提出的直交調和搜尋演算法之執行步驟。

4. 數值實驗與分析

   本研究以八個測試函數做範例來進行實驗驗證與實驗設計，實驗結果也將與近年來發表在國際期刊上並有優秀表現且結合直交表之演算法一併做比較。

5. 結論與未來建議

   本章為本研究論文之結論，並對本研究之建議及未來之研究方向作一概括描述。

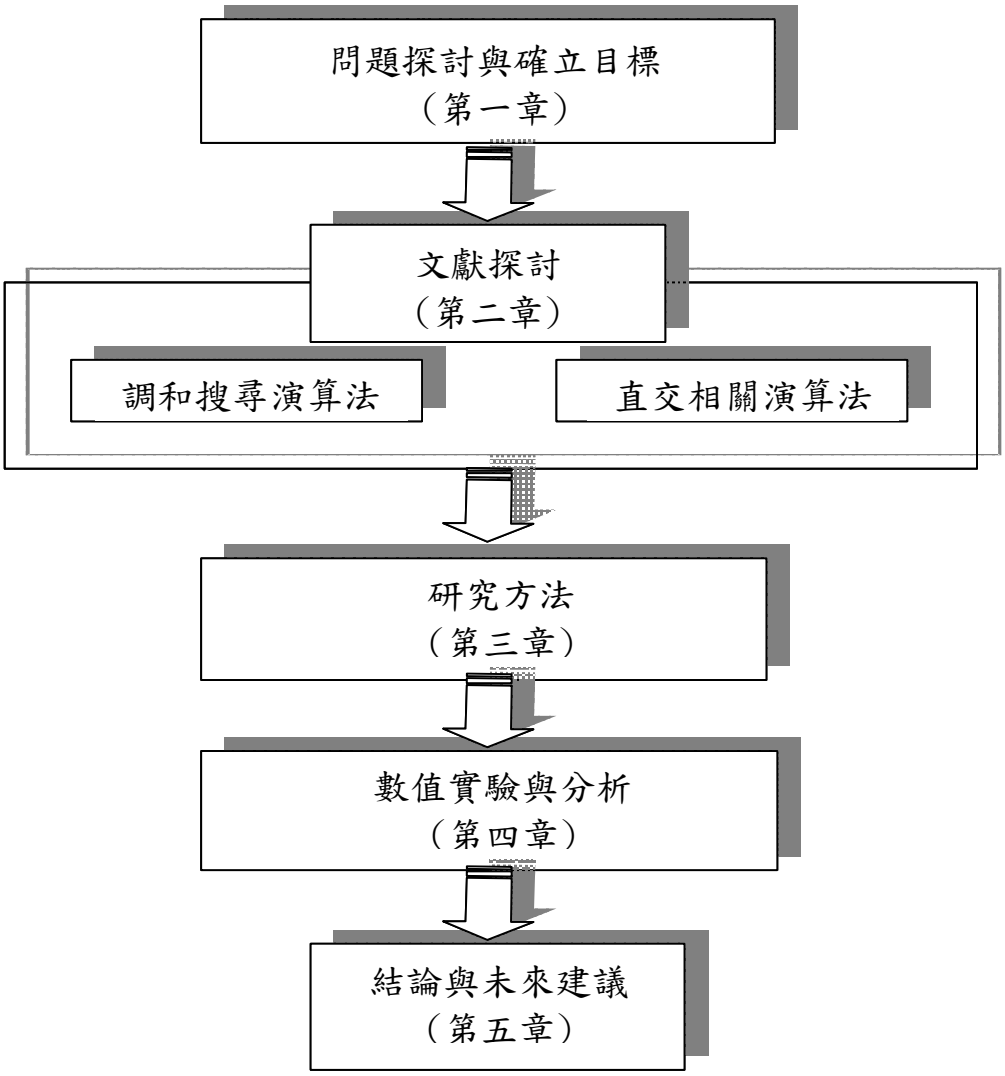


圖 1.1 研究方法與步驟

1.4 研究工具

本研究使用 Matlab7.0 來作為運算工具，利用 Matlab 的特性，開發出一套合理且可行的程式，並利用本研究中的範例求出最佳的解。Matlab 是 Matrix Laboratory 的簡寫，它是一個直譯式的語言程式，與其他的程式語言比較起來，因為語法較於單純，所以較容易學習與瞭解。Matlab 的主要特色在於數值分析、矩陣的運算與資料型態的轉換，它能夠讓使用者即使面對大量的資料以及冗長的運算問題時，也能有效的解決，因此，本研究選擇 Matlab 來作為演化最佳化問題的解決工具。

第二章 文獻探討

2.1 啟發式演算法相關文獻

從 1970 年代開始，啟發式演算法藉由一些規則(rules)與隨機現象(randomness)以仿效自然現象(natural phenomena)的方式設計演算法結構，而這樣的演算法雖然不見得能夠找到最佳解，但是比起傳統的數理求解方法來說，啟發式演算法能以較快的速度找到滿意解。啟發式演算法透過電腦計算搜尋複雜問題的最佳解，目前被大量應用在搜尋(search)、最佳化(optimization)、排程(scheduling)等各種工程問題求解[4][6][29][44][45][51]。

近年來，常被使用的啟發式演算法大致可分為下列幾類：

1. 仿效生物演化過程(biological evolutionary process)的基因演算法(Genetic Algorithm; GA)、演化策略(Evolution Strategies; ESs)、演化規劃(Evolution Programming; EP)
2. 仿效動物行為(animal behavior)的蟻拓尋優法(Ant Colony Optimization; ACO)、禁忌搜尋演算法(Tabu Search; TS)、粒子群演算法(Particle Swarm Optimization; PSO)
3. 仿效物理退火過程(physical annealing process)的模擬退火演算法(simulated annealing)

而本研究改良的調和搜尋演算法，乃是仿效樂團演奏表現調整至最協調且美妙的改善過程，將此概念引用到最佳化演算系統當中，此演算法屬於仿效動物行為的啟發式演算法。

以下將針對上述之啟發式演算法做介紹，調和搜尋演算法將於 2.2 節詳加說明。

2.1.1 基因演算法

基因演算法是由 John Holland[36][37][38]所提出的一般性最佳化演算法則，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)。

基因演算法的主要操作(operator)為基因交換(crossover)、天擇(selection)

與突變(mutation)。天擇主要係依據母代(Parent)中每一染色體之適應值來決定子代(offspring)中，哪些染色體該被淘汰或複製及保留至下一代的一種運算操作，常見的方法有輪盤法(roulette wheel)、競賽法(tournament selection)、比例法(proportionate selection)、排序法(ranking selection)[31]。基因交換的目的是在整個族群中製造差異，藉以產生優良且具適應能力的下一代，常見的基因交換方式有單點交換(One-point crossover)、尾尾交換(Tail-tail crossover)、首尾交換(Head-tail crossover)、複合式交換(Multiple crossover)、均勻交換(Uniform crossover)[1][56]。突變是為了增加族群個體的差異性而使用的一種基本運算子，藉由突變過程的變動運作，增加更多新的染色體進入搜尋空間中，以避免過早收斂至局部最佳解。

除了傳統的GAs模型外，近來也發展出一些特殊的GAs模型，如：Darrell Whitley提出的Genitor[9][10][11]、Eshelman提出的CHC[19][20]、Goldberg提出的Messy GAs[30][32]、Cantù-Paz提出的Island based GAs[5]等。

2.2.2 演化策略

演化策略是 Rechenberg[46][46]及 Schwefel[49][50]發展出來的演化模型。演化策略與基因演算法最大的差異在於演化策略著重的是族群個體的表現型態(phenotype)的變化，基因演算法著重的是個體基因型態(genotype)的擾動。

演化策略主要的演化方法為天擇(selection)、突變(mutation)與重構(recombination)。最常見的天擇方法為$(\mu \overset{+}{,} \lambda)$ [49]的型態，Schwefel and Rudolph[52]提出(μ,κ,λ)-ES 架構，κ代表演化代數，若母代存活代數大於κ則保留母體，$1 \le \kappa \le \infty$，若$\kappa = 1$，則天擇方法會成為(μ,λ)-ES 的型態；若$\kappa = \infty$，則天擇方法會成為$(\mu + \lambda)$-ES 的型態。為了使演化策略的演化更具有彈性，針對突變強度(mutation strength)的調整而發展出自我適應能力的演化策略(self-adaptation evolution strategies)，使用學習參數τ_0以及突變強度(mutation strength)的更新規則增強演化效率。此外，為了在整個族群中製造差異，藉以產生優良且具適應能力的下一代，發展出類似基因交換(crossover)的操作方式，稱為重構。

2.2.3 演化規劃

演化規劃是 L. Fogel 於 1960 年代為了發展出可以作到自我演化的人工

智能所提出的演化模型[23]。Fogel 認為人工智能需要能預測環境的變化並作出適當的改變以接近目標，因此他將環境當作一連串的符號(symbol)，找出預測環境的機制用以產生出下一個相關環境的符號。

演化規劃對於族群中的個體利用有限狀態機(finite state machine)當作演化的單元，每一次對環境的預測都給訂一個 pay off function 用以評斷個體演化的好壞。在演化操作方面，演化規劃並不使用重構的方法，而是每一個母體經由突變產生出單一的子代，主要的突變作法有：改變輸出符號(change of an output symbol)、改變轉換狀態(change of a state transition)、增加狀態(addition of a state)、刪除狀態(deletion of a state)以及改變初始狀態(change of the initial state)；在天擇的方法方面，演化規劃可以視為$(\mu+\mu)$形式的天擇，由於每一個母體產生出單一的子代，再從母體與子代中選擇出其中的一半做為存活的個體。

2.2.4 蟻拓尋優法

蟻群系統(Ant system)為 Dorigo et al.[14]提出。蟻群系統乃是根據自然界中螞蟻覓食的行為模式所發展出來的演算法，其最大的特色為人造螞蟻(artificial ant)根據路徑上的費洛蒙(pheromone)多寡來搜尋求解組合最佳化的問題。

蟻群尋優法在 Colorni et al.[7][8]提出，稱之為蟻群系統（Ant System）。其原理是藉由自然界螞蟻尋找食物的精神發展出一套演算法則，並求解蟻群系統驗證旅行銷售員(traveling sales man ; TSP)問題。而 Dorigo and Maria[14]提出改良於蟻群系統的蟻拓法(Ant Colony System; ACS)，蟻拓法被廣泛被使用於求解各種最佳化問題。Dorigo et al.[13]。將蟻群系統、蟻拓法與相關應用的法則歸納成一套啟發式演算法 Ant Colony Optimization Meta-Heuristic。以這套演算法為精神的統稱為蟻拓尋優法(Ant Colony Optimization; ACO)。

2.2.5 禁忌搜尋演算法

禁制搜尋法為 Fred Glover(1977)所提出[28]，它是一種用來克服掉入區域最佳解化的搜尋程序，此方法係透過彈性記憶體(Flexible Memory)之應用，藉以避免陷入循環解的產生，並於一合理時間內求得一近似解(或最佳解)。

禁制搜尋法其彈性記憶體結構可分為短期與長期二種階段。在短期階

段利用禁制限制式(Tabu Constraints)來限制搜尋的狀態，以避免搜尋的重複與反覆，而渴望準則(Aspiration Level)則用以釋放禁制限制，避免停滯不前。短期階段的重點在加速到達區域最佳化；長期階段則用加強性(Intensification)與多樣性(Diversification)將搜尋帶入新的區域以求得更佳的解，使得搜尋得以跳脫區域最佳解，進而尋獲總體最佳解。

2.2.6 粒子群演算法

最早是由 Kennedy & Eberhart(1995, 1996)[15]提出來的最佳化推測演算技術，其概念乃是模仿自然界中鳥類搜尋食物的機制而設計出的演算法。後續不斷有相關的改良方法提出，例如，Shi & Eberhart(1998)提出降低粒子移動速度的方法，在演化更新時乘上一個慣性因子(inertia weight)，該研究指出當慣性因子大於 1.2 時較具有探索的能力，當小於 0.8 時較容易產生群聚效應。之後在 2001 年他們又提出一個方法，以緊縮因子(constriction factor)來取代之前所提出的慣性因子，亦可達到不錯效果(Eberhart & Shi, 2001) [16]；後續另有 Bergh & Engelbrecht(2004)以協同作業的概念改良 PSO 的方法。

最佳粒子群演算法建立在群體(Swarm)間協同呼應的組織性運作機制上，演化的過程中群體的每個粒子(Particle)會不斷地在解空間內探索，而且本身能記憶自己在嘗試過程中最好的值，我們稱之為最佳值(Particle best value ;pbest)，此外，粒子群彼此之間也能夠傳遞群體中最好的粒子的位置，這種全域最佳粒子我們稱之為群體最佳值(Global best value ; gbest)，或是鄰近地區最好的粒子所在位置，這種區域最佳粒子我們稱之為區域最佳值(Local best value; lbest)，最佳粒子群集演算法尋優步驟就是依據 pbest 及 gbest(或 lbest)來決定每個粒子所要探索的方向，藉由尋優步驟使得整體的粒子群集逐漸接近在解空間內的目標位置。

2.2.7 模擬退火演算法

模擬退火法最早由 Metropolis et al.(1953)提出，當時並沒有受到研究者的重視。後來 Kirkpatrick et al. [40]將其應用於求解組合性最佳化問題後，才引起注意。模擬退火法是由模擬物理現象於物理系統，在降溫結晶能量降低的過程所衍生出的一種演算法，故稱為模擬退火法。退火過程是物理上為使物質中晶體排列達到最低能量狀態所使用的降溫步驟。

模擬退火法的最大特徵在於除了接受較優解外，仍然給一定機率以接

受較差的解。最初在 T 值較大(溫度較高)時，接受較多較差的解，隨著 T 值減少，接受較差解的機率就越來越小，這個特徵使得模擬退火法不同於鄰近搜尋法(neighbor search)，而具有跳出局部最佳解而達到全域最佳解的能力。

2.2 調和搜尋演算法相關文獻

2.2.1 調和搜尋演算法介紹

韓國學者 Zong Woo Geem 於 2001 年發表一套全新的最佳化啟發式演算法，其原理是將樂團演奏表現調整至最協調且美妙的現象引用到最佳化演算系統當中，而發展出一套全新的啟發式演算法，本研究稱之為調和搜尋演算法。

調和搜尋演算法的概念就像一個樂團如何奏出協調且美妙的曲子一樣。在樂團演奏表演當中，我們可以知道曲子演奏表現的好壞在於每一個樂器所發出來的聲音是否能夠讓曲子聽起來協調且美妙，然而協調且美妙的音樂當然是透過樂團不斷的重複練習與各個樂器音調的修正來達到整體有更好的表現。對於一個樂團演奏表現的好與壞，主要就是在於每一個樂器所表現出來的素質。而這樣的人工現象(artificial phenomenon)引用到最佳化演算系統裡頭，一個樂團演奏的表現就如同最佳化當中的目標函數，每一個樂器就像代表著一個變數，而如何讓演奏達到最佳就如同於如何讓目標函數到達最佳值的問題，樂團藉由每一次的練習以達到最完美的境界就如同演算法藉由每一次的評估函數來更接近最佳解。樂團演奏的練習程序與最佳化問題的演化程序有著相似概念，其比較如下表 2.1 所示。

表 2.1 最佳化演化程序與音樂表現練習程序之比較表

Comparison Factor	Optimization Process	Performance Process
Best state	Global Optimum	Fantastic Harmony
Estimated by	Objective Function	Aesthetic Standard
Estimated with	Values of Variables	Pitch of Instruments
Process unit	Each Iteration	Each Practice

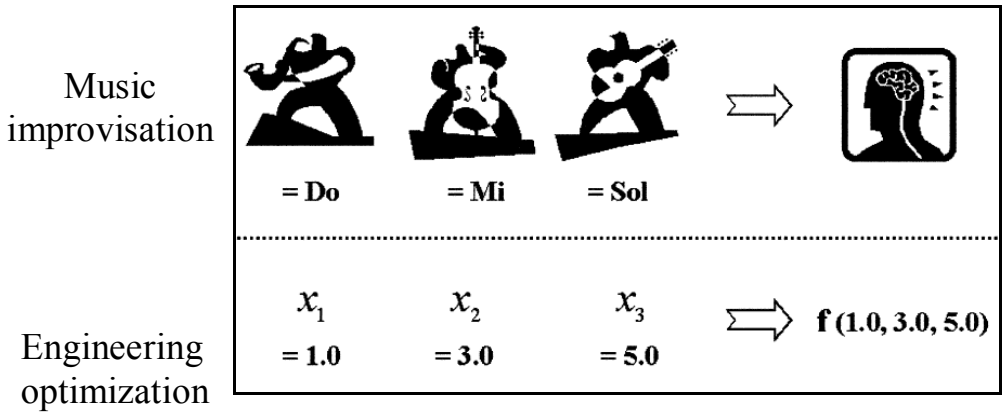


圖 2.1 音樂改善與工程最佳化之概念對照圖

調和搜尋演算法之概念程序如下：

Step 1：隨機產生初始的調和記憶空間(Harmony Memory)，此記憶空間裡保存著 *HMS* (Harmony Memory Size)個演奏組合。

Step 2：下一次的演奏練習每個樂器就隨機(random)產生一個音調、或隨機抽取自記憶空間(do nothing)、或是由記憶空間裡的音調作微調(pitch adjustment)，根據此原則，產生一組新的音調組合。

Step 3：如果新的一組音調演奏後聽起來有比調和記憶空間裡其中一組音調組合還要好，那麼就把記憶空間裡最差的一組淘汰，並將新的音調組合更新至調和記憶空間。

Step 4：演奏後達到中止條件，即終止；否則回到 Step 2。

舉例來說，假設某一首爵士樂曲由小提琴(Fiddle)、薩克斯風(Saxophone)、鋼琴(Keyboard)三種樂器所演奏。在一開始時，三種樂器都隨機採用一種音調分別為(C, E, G)，同理重複運作三次，三組解分別為(C, E, G)、(C, F, A)、(B, D, G)，此三組音調即為調和記憶空間裡的三組解，如下圖所示：

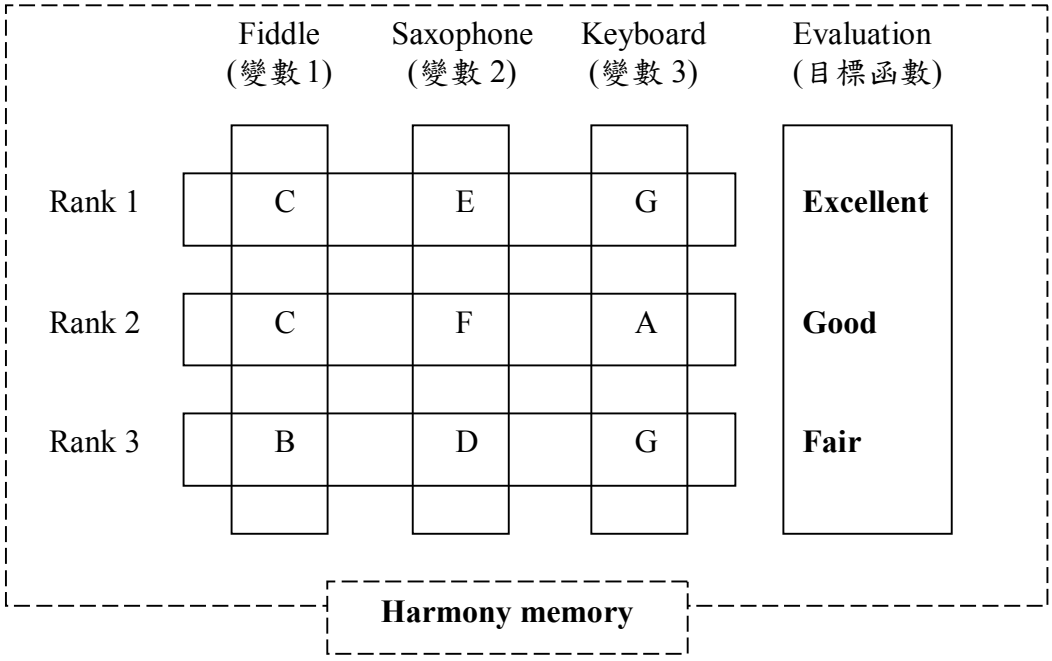


圖 2.2 調和記憶體示意圖

在音樂改善過程當中，利用三種樂器的音調產生一個新的組合。譬如，小提琴的音調採用{C, C, B}中的 C，薩克斯風的音調採用{E, F, D}中的 D，鋼琴的音調採用{G, A, G}中的 A，組合成一個新的音調組合(C, D, A)。進而評估新的音調組合是否優於調和記憶空間裡的其中一種組合。如果是，則將(C, D, A)儲存到調和記憶空間裡，並刪除調和記憶空間中最差的音調組合(B, D, G)，否則就保留原本的調和記憶體。

樂器音調的搭配組合決定了音樂表現的成果，調和搜尋演算法掌握的這樣現象創造全新的演算結構。調和搜尋演算法執行程序如下：

Step 1: 定義問題與參數值

假設問題為最小化問題，其模式如下：

$$\text{minimize} \quad f(x)$$

$$\text{subject to} \quad x_i \in X_i, \text{ for } i = 1,2,...,N$$

則 $f(x)$ 為目標函數值，x_i 為第 i 個決策變數值，X_i 為第 i 個決策變數之變數範圍集合。也就是說對於離散變數 X_i，$X_i = (x_i(1), x_i(2),...,x_i(K))$，而連續變數 X_i，$x_i^L \le X_i \le x_i^U$。N 為決策變數個數，K 為離散變數的可能值個數。

調合搜尋演算法之參數包括有：(1) 記憶體空間大小(*HMS*)，也就是每

一代中保留 *HMS* 組可行解在記憶體中以供下一代進行演化，(2) *HMCR*(Harmony memory considering rate)，其中 $0 \le HMCR \le 1$，(3) *PAR* (pitch adjustment rate)，其中 $0 \le PAR \le 1$，(4) *bw* 為控制變數移動步距大小之參數。

Step 2: 建立初始解

隨機產生 *HMS* 組初始解，並計算各別的目標函數值，依照 $f(x)$ 大小排序後，解儲存在調和記憶體(harmony memory; *HM*)中，如下所示。

$$HM = \begin{bmatrix} x^1 \\ x^2 \\ \vdots \\ x^{HMS} \end{bmatrix}$$

其中， x^i 為 *HMS* 個函數評估值排序後的第 i 組解(solution vector) $x^i = (x_1^i, x_2^i, ..., x_N^i)$。

Step 3: 根據調和記憶體產生逐漸改善

在調和搜尋演算法中，產生一組新解(harmony vector) $x' = (x_1', x_2', ..., x_N')$，$x'$ 中的每個變數主要依靠著三個機制所產生：(1) memory consideration: 變數值隨機抽取自 *HM*，(2) pitch adjustments: 變數值抽取自 *HM* 後進行微調，(3) randomization: 隨機化產生。

舉例說明，新解的第一個變數 x_1' 有 *HMCR* 的機率可以選自於 *HM* 中 $(x_1^1 \sim x_1^{HMS})$其中一個值，有 1－*HMCR* 的機率自 x_1 的變數範圍中隨機產生。同樣的，其他變數產生如下通式：

$$x_i' \leftarrow \begin{cases} x_i' \in \{x_i^1, x_i^2, ..., x_i^{HMS}\} & \text{w.p. } HMCR \\ x_i' \in X_i & \text{w.p. (1-}HMCR\text{)} \end{cases}$$

對於每個變數值有 *HMCR* 的機率為 *HM* 中儲存的歷史數值中選取出來，有 1－*HMCR* 的機率由變數範圍中隨機產生。所以，當 *HMCR*=1 時，選取的變數值則主要來自於 *HM* 中儲存的歷史數值，也就是放棄 *HM* 中數值以外的機會。而 1－*HMCR* 的機率由變數範圍中隨機產生，這個作法就與基因演算法突變的意義相似。

然而對於新解中取自於 *HM* 的變數值有 *PAR* 的機率作微調(pitch adjustment)找尋鄰近解。

Pitch adjusting decision for x_i'

$$x_i' \leftarrow \begin{cases} \text{Yes} & \text{w.p. PAR} \\ \text{No} & \text{w.p. (1-PAR)} \end{cases}$$

也就是說，變數執行微調的機會只有 $\text{HMCR} \times \text{PAR}$。

假設 $x_i(k)$ 將進行微調，其微調的數值公式如下：

$x_i' \leftarrow x_i(k+m)$ for discrete decision variables

$x_i' \leftarrow x_i' + \alpha$ for continuous decision variables

其中 m 為鄰近解的指數，$m = \{..., -2, -1, 1, 2, ...\}$。而 $\alpha = bw \times u(-1,1)$，其中，bw 為決策者給定的微調步距，$u(-1,1)$ 為均勻分配隨機數值於(-1,1)之間。

因此，*HMCR* 與 *PAR* 在調和搜尋演算法中，分別扮演著全區域搜尋(global search)與區域解搜尋(local search)權重的角色。

Step 4: 更新調和記憶體

在步驟 3 中所產生的新解，經由函數評估後，優於 *HM* 中函數評估值最差的一組解，則將新解更新至 *HM* 中。

Step 5: 若未滿足終止條件，回到步驟 3

滿足終止條件則停止，否則回到步驟 3 和步驟 4。

調合搜尋演算法在 Step 3 產生新解 $x' = (x_1', x_2', ..., x_N')$ 時，每個變數 x_i' 主要依靠著三個機制所產生：(1) 採用自記憶體中的變數值 memory consideration: 變數值隨機抽取自 *HM*，(2) pitch adjustments: 變數值抽取自 *HM* 後進行微調，(3) randomization: 隨機化產生。每個變數 x_i' 形成的機制與機率，可經由簡單的樹枝圖呈現，如圖 2.3。

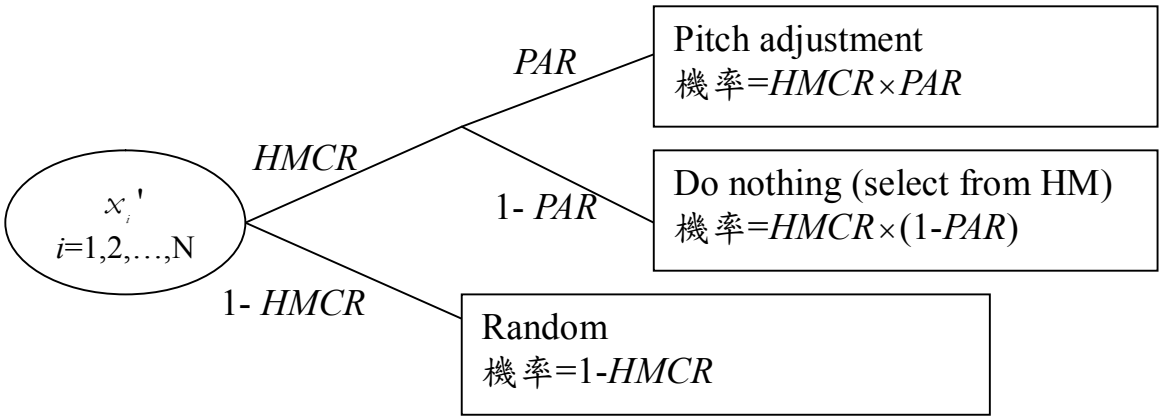


圖 2.3 HS 新解產生機制與機率關係圖

2.2.2 調和搜尋演算法與基因演算法之比較

由於調和搜尋演算法與基因演算法在結構上有些許的相似，本節將針對兩演算法進行比較。而綜合過去的各種最佳化搜尋演算法，歸納出大多數搜尋演算法通常有以下的程序：

1. 記憶體中的解隨機初始化，或特定的啟發式演算法，求出某一個(些)解，完成後執行演化程序。
2. 對記憶體內的每一組解(HS 為 harmony vectors；GA 為 chromosomes)計算函數評估值，函數評估值與最佳解的距離直接有關。
3. 根據過去記憶體內的解進行演化，演化過程中通常具有突變機制以跳脫區域解。
4. 定義終止條件準則，滿足的話，就停止，否則回到步驟 2。

藉由以上步驟，我們可以歸納 HS 與 GA 的共同特點有：

1. 兩者初始解採用隨機的方式求得。
2. 都具有適者生存，不適者淘汰的精神。
3. HS 有著類似於 GA 交配與突變的機制。

而兩者相異之處有：

1. HS 有類似於交配與突變的機制，只是運作方式與運作程序與 GA 不同。
2. HS 具有微調機制(pitch adjustment)，相較 GA 有較強的區域搜尋最佳解能力。

表 2.2 GA 與 HS 搜尋機制差異比較表

	GA	HS (以 GA 術語描述 HS 搜尋現象)
複製	自母體中挑選出成對的染色體	*HM* 中的 *HMS* 組解皆被選出
交配	1. 任兩條染色體進行交配 2. 通常採用單點或雙點交配	1. *HMS* 組解進行交配-每個變數值皆可挑選自 *HM* 2. 採用多點交配模式
突變	交配後進行突變	變數有 1－*HMCR* 的機率跳脫到可行解區域，此機制與突變相似，*HMCR* 的機率採用 *HM* 中的值
微調	無微調機制	進行交配的變數有 *PAR* 的機率在進行微調

2.2.3 調和搜尋演算法之相關文獻

調和搜尋演算法起源於 2001 年[24]由韓國學者 Geem Z.W.所發表，當時提出時引用三個問題以證實演算法求解能力，包括有(1)銷售員旅行問題(Traveling Salesman Problems; TSP)，(2)有限制式的最小化問題，並與 GA、EP 等演算法做比較，(3)水資源分配輸送管網路設計問題(pipe network design)。

Geem Z.W. (2002)應用 HS 於輸送管網路設計問題[25]，在滿足水資源流量需求及最小化瓶頸輸送管的情形下，以 HS 求解輸送管設計總成本最小化之研究。

Geem Z.W. et al. (2004)應用 HS 在工程結構最佳化(structural optimization)問題上[26]，為了測試 HS 求解的能力與穩定性，測試問題大小包括最小的10條平面桁架系統問題(10-bar planar truss)至最大的200條平面桁架系統問題等等。

Geem Z.W. (2005)將整理三大類型問題以 HS 作為求解工具，包括有無限制式之最小化問題、有限制式之最小化問題及數種工程結構最佳化問題

等等。

2.3 直交表結合搜尋演算法之相關文獻

近幾年來，演算法的改良通常以下列兩種型式的方法做結合：啟發式演算法和傳統的理論最佳化手法搭配做改良。啟發式演算法如 2.1 節所述，而傳統的理論最佳化手法如直交表實驗設計、梯度搜尋(gradient)、簡捷法(simplex)等等，而這些方法的結合都有相當不錯的成效，其中梯度搜尋與簡捷法常被用來強化演算法區域解搜尋(local search)的能力，而直交表常用以降低實驗次數並判斷搜尋的方向，加強搜尋速度，本研究將以直交表實驗設計作為改良手法，而近年來著名期刊中以直交表作為改良手法的演算法整理如表 2.3。

Ho S.-Y. and Chen J.-H. (2000)以 GA 為演算基礎，利用直交表實驗設計技巧使 GA 交配，並應用在銷售員旅行問題上[33]，透過直交表可以更有效率的從母代的路徑中找到更好的解，而在此篇文章中，針對 TSP 問題使用直交表所改良的 GA 屬於離散類型的。

Leung Y.-W. and Wang Y. (2001)將直交表交配的方法以量化的形式改良 GA，以便處理實數型態的問題，稱之為直交表基因演算法[43]。較特別的是，OGA 的初始解乃透過直交表在可行解區域裡頭均勻的分佈實驗點中，取較佳的解作為初始解。OGA 使用了十五個測試函數分別與近年來五個知名的演算法進行比較，以證實 OGA 的搜尋能力。

吳子逢(2003)提出一個直交演化策略(orthogonal evolution strategy; OES)，此演算法使用以直交實驗設計的新型突變機制來找到連續型變數最佳化問題的近似最佳解。OES 加強原本演化策略中突變機制的統計分析能力，有效率地縮短實驗時間及成本，進而達到改善並穩定搜尋能力。

Ho S.-Y. et al. (2004)發表了直交模擬退火演算法[34]。在 OSA 中，透過直交表實驗設計，在空間中均勻的分佈點實驗後，再經由主效果分析，判斷每一緯度之最佳水準，此為子代移動之方向與距離。並應用 OSA 於八個測試函數及設計混合 H_2/H_∞ 最佳化控制等最佳化問題中。而 OSA 陸續應用在大型的元件配置問題[35](Large Floorplanning Problems)、電磁體的最佳化問題[34] (Electromagnetic Problems)等等。

林宏穗(2004) 提出直交粒子群最佳化演算法，主要是將傳統粒子群最佳化演算法的粒子移動策略取代，由一個以直交實驗設計為基礎的智慧型產生新解機制來強化其搜尋的能力。

表 2.3 直交表結合演算法之相關文獻與應用

作者	年代	主要參數	問題種類	與其他方法比較
Shim-Ying Ho and Jian-Hung Chen	2000 [33]	1. *population size* 2. *Pc* 3. *Pm* 4. *ps*	Traveling Salesman Problems	OX, UX2, EER, and OAX
Yiu-Wing Leung and Yuping Wang	2001 [43]	1. *population size* 2. *pc* 3. *pm* 4. *Q* 5. *F*	15 test functions	FES, FSA, PSO, EO, CEP
Shinn-Ying Ho et al.	2004 [35]	1. *Temperature* 2. *N* 3. *Cooling rate* 4. *Q (level)*	Large Floorplanning Problems	Fast-SP
Shinn-Jang Ho et al.	2004 [34]	1. *Temperature* 2. *N* 3. *Cooling rate* 4. *Q (level)*	Electromagnetic Problems	FSA and IGA
Li-Sun Shu et al.	2004 [55]	1. *Temperature* 2. *N* 3. *Cooling rate* 4. *Q (level)*	(1) A. Large Parameter Optimization Problem (2) Designing Mixed H_2/H_∞ Optimal Controllers	(1) ESGA and FSA (2) GA-based method and GA

第三章 研究方法

本研究主要將調和搜尋演算法利用直交表實驗設計方法加以改良，透過直交表實驗設計具有優良經驗的推理能力與主效果分析，為子代判斷正確的搜尋區域與方向，使求解過程能在決策者有限的時間上找到最好的求解品質。本研究提出直交調和搜尋演算法主要以三項機制運作搜尋：(1)以直交交配來做全域搜尋，(2)以直交微調來做區域搜尋，(3)以隨機方式尋找其他可行解，其結構如圖 3.1，細部流程如圖 3.7，於本章 3.2 節將詳細介紹 OHS 之操作方式。而本章將於 3.1 節介紹演算法中使用的直交表實驗設計技術，包括直交表、主效果分析及最陡路徑法等。

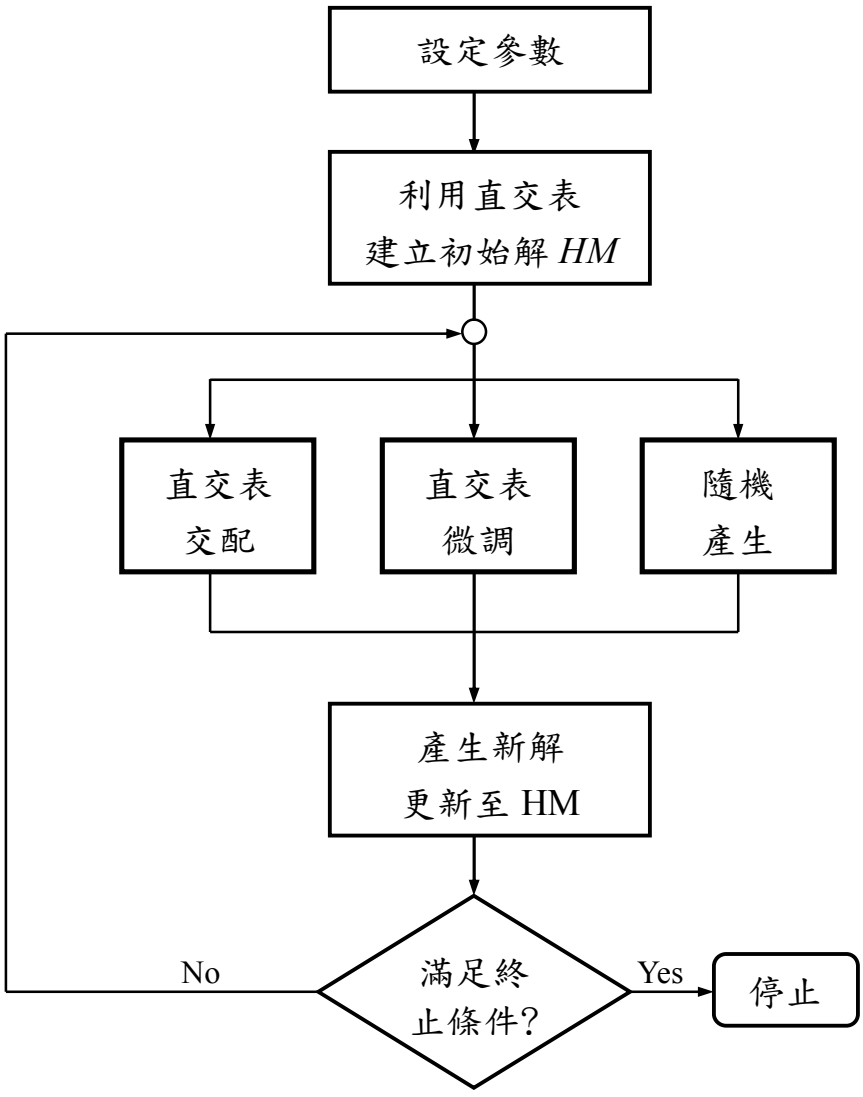


圖 3.1 直交調和搜尋演算法演算流程

3.1 直交表實驗設計方法

3.1.1 直交表

直交表(orthogonal array)是由 R. A. Fisher(1951)最先提出的。「直交」所代表的意思是平衡(balance)而不混合(mix)，亦即統計上的獨立(statistically independence)，因此直交表中每一欄的各水準值(level)出現次數是相同的。應用直交表分析資料的好處是可以獨立且均衡的求出每一個因素的主效果(main effect)，由主效果就可以推論每一個因素對於該實驗結果的影響程度。使用直交表，事實上僅是進行部份因素實驗(fractional-factorial experiment)，因此能較完全因素實驗(full-factorial experiment)節省大量執行的時間。並且直交表實驗具有系統推理的特性，因此只需進行部份因素實驗就可以求得最佳解的近似解(near optimum)。

直交表的組成特性包括：

(1)任一列，每個水準值的出現機率相同，

(2)任兩列其中兩個因素，每次組合兩因素的存在機率相同，

(3)任兩列其中兩個因素，會出現所有水準組合方式，

(4)若將任兩列對調，其正交性質依然不變，

(5)若將任意列刪除，其正交性質依然不變。

以三個因素，每個因素都有兩個水準值的例子來說，若要進行完全因素實驗，應執行 8(即 2^3)個實驗，如表 3.1 所示。如果現在只能執行 8 個實驗中的 4 個實驗，依傳統的單因素實驗方法，在一次只改變一個因素的方式下，則會選擇圖中實驗編號為 1、2、4 與 8 的實驗點進行實驗。但是點 1、2、4 與 8 並不均衡，也就是說，在六面體的每一面所選取之實驗點數不盡相同。如果選擇的實驗點是 1、4、6 與 7 (或是 2、3、5 與 8)，如圖 3.2 所示的黑點，則六面體的每一面都有兩個實驗點，且都相互對稱，因此最佳解皆在其所包圍的立方體之中，藉由此均勻取樣的實驗，來推測全部實驗的最佳值。而這樣的採樣方式，即稱為部份因素設計(fractional factorial design)。

表 3.1 三因子二水準完全實驗表

實驗編號	x_1	x_2	x_3
1	-1	-1	-1
2	+1	-1	-1
3	-1	+1	-1
4	+1	+1	-1
5	-1	-1	+1
6	+1	-1	+1
7	-1	+1	+1
8	+1	+1	+1

表 3.2 三因素二水準值部份因素實驗表

實驗編號	x_1	x_2	x_3
1	-1	-1	-1
2	+1	+1	-1
3	+1	-1	+1
4	-1	+1	+1

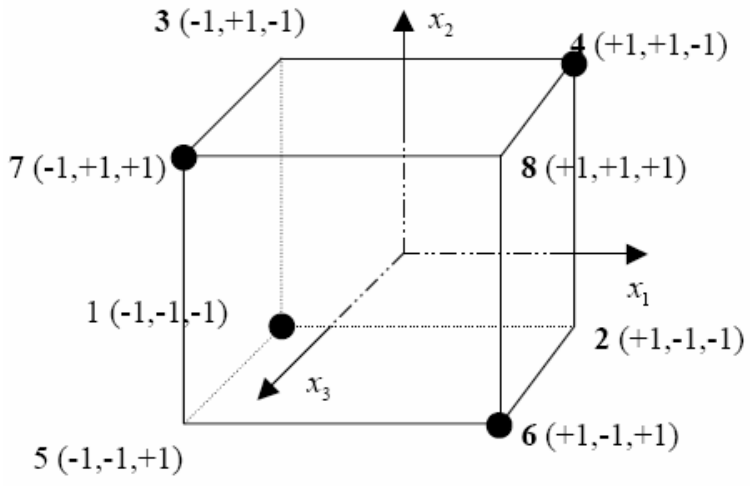


圖 3.2 三因素二水準值部份因素實驗空間分佈圖

若以直交表來組合實驗因素，這些因素組合會均勻的在求解空間中取樣，則可以大量減少計算次數的複雜度。然後根據實驗的結果，進一步計算出每個因素在不同參考值下的主效果，而該主效果則直接反映各因素在評估函數中的優劣程度。將各因素中主效果較佳的參考值加以組合，便可推理得到最佳組合解。

3.1.2 建立直交表

創造直交表必須先設定水準數(Q)與因素個數(N)，得到的直交表為 $M \times N$ 矩陣，其中 M 為實驗次數。直交表的建立方式整理自文獻[43]。

Algorithm:
建立 N 個因素、Q 個水準、M 次實驗的直交表為 $L_M(Q^N)$，其中 $M= Q^J$。
step 1: 決定 J 值，亦即決定實驗次數 $M= Q^J$。

J 值為滿足 $\frac{Q^J-1}{Q-1} \geq N$ 的最小整數。

step 2: 建立直交表基本欄(basic columns):

FOR k=1 To J DO
Begin

$$j = \frac{Q^{k-1}-1}{Q-1} + 1;$$

FOR i=1 TO Q^J DO

$$a_{i,j} = \left\lfloor \frac{i-1}{Q^{J-k}} \right\rfloor \bmod Q;$$

END.

step 3: 建立直交表非基本欄(nonbasic columns):

FOR k=2 To J DO
Begin

$$j = \frac{Q^{k-1}-1}{Q-1} + 1;$$

FOR s=1 TO j-1 DO
FOR t=1 TO Q-1 DO

$$a_{j+(s-1)(Q-1)+t} = (a_s \times t + a_j) \bmod Q;$$

END.

step 4: Increment $a_{i,j}$ by one for all $1 \leq i \leq M$ and $1 \leq j \leq N$.

例如：水準數 Q=3、變數個數 N=3，產生直交表如下：

$$\mathrm{OA}=\begin{bmatrix}1&1&1\\1&2&2\\1&3&3\\2&1&2\\2&2&3\\2&3&1\\3&1&3\\3&2&1\\3&3&2\end{bmatrix}$$

直交表呈現出來的屬於一個離散類型的矩陣，為了將直交表應用至演算法結構裡頭，Leung Y.-W. and Wang Y. (2001) 提出將空間量化的手法。透過量化的過程可以將某一特定的空間，對應至各個水準數，如某一變數之變數範圍為(0, 10)，在水準數為 3 時，第一個水準值為 0，第二個水準值為 5，第三個水準值為 10。針對每一個變數上下界分割為 Q 個水準並對照直交表進行量化。定義變數 x_i 為第 i 個因素(i=1, 2, …, N)，其中 x_i 的變數範圍為$[l_i, u_i]$，將 x_i 分割成 Q 個水準分別為$\alpha_{i,1}$, $\alpha_{i,2}$, ..., $\alpha_{i,Q}$，而$\alpha_{i,j}$ 定義如下：

$$\alpha_{i,j}=\begin{cases}l_i & ,j=1\\ l_i+(j-1)\left(\dfrac{u_i-l_i}{Q-1}\right) & ,2\le j\le Q-1\\ u_i & ,j=Q\end{cases} \qquad \text{(公式 3. 1)}$$

將$\alpha_{i,1}$, $\alpha_{i,2}$, ..., $\alpha_{i,Q}$ 對應直交表之水準別填入其中，產生新表。舉例來說，假設變數 $0\le x_i\le 10$ $(i=1,2,3)$ 則新表型式如下：

$$\mathrm{OA}=\begin{bmatrix}1&1&1\\1&2&2\\1&3&3\\2&1&2\\2&2&3\\2&3&1\\3&1&3\\3&2&1\\3&3&2\end{bmatrix} \rightarrow \mathrm{OA}_{\text{quantize}}=\begin{bmatrix}0&0&0\\0&5&5\\0&10&10\\5&0&5\\5&5&10\\5&10&0\\10&0&10\\10&5&0\\10&10&5\end{bmatrix}$$

3.1.3 主效果分析

透過直交表來進行主效果分析，來了解哪些因素對於事件影響的效果為何，稱之為主效果分析或因素分析。定義 y_t 為直交表實驗中第 t 次實驗的函數評估值，第 j 個因素水準 k 的主效果為 S_{jk}，其定義為 $S_{jk}=\sum_{t=1}^{M} y_t \times F_k$，其中 F_k 為一個旗標值，若第 t 次實驗中第 j 個因素選用水準為 k，則 $F_k=1$，否則 $F_k=0$。

表 3.3 主效果分析計算範例表

實驗次數	因素			評估函數
	F-1	F-2	F-3	y_t
E-1	L-1	L -1	L -1	y_1
E-2	L -1	L -2	L -2	y_2
E-3	L -1	L -3	L -3	y_3
E-4	L -2	L -1	L -2	y_4
E-5	L -2	L -2	L -3	y_5
E-6	L -2	L -3	L -1	y_6
E-7	L -3	L -1	L -3	y_7
E-8	L -3	L -2	L -1	y_8
E-9	L -3	L -3	L -2	y_9
S_{j1} S_{j2} S_{j3}	S_{11} S_{21} S_{31}	S_{12} S_{22} S_{32}	S_{13} S_{23} S_{33}	

(註) F 為因素別，E 為實驗次數別，L 為水準別，S 為主效果值

因此，如果目標函數屬於最大化問題，主效果值越大則對於目標函數越有貢獻；反之，如果目標函數是最小化問題，主效果值越小則對於目標函數越有貢獻。主效果計算結果可顯示出因素中水準的個別影響。例如，在最大化問題中，如果主效果計算後得知 $S_{\mathrm{j1}} > S_{\mathrm{j2}} > S_{\mathrm{j3}}$ 則表示在第 j 個因素中的整體貢獻度，第一個水準對於大於第二水準，第二個水準大於第三個水準，因此在第 j 個因素中，選用第一個水準作為方向。

此外，如果因素間的交互作用影響如果很大，主效果分析的結果會因為交互作用而產生干擾，如果能交存在的交互作用消除或降至最低，主效果分析的精確度就能相對提升。

3.1.4 最陡路徑法

在最適化的過程中，如何以迅速而有條理、有系統的方法逼近極值點所在區域，為一般研究者常遭遇到的困擾。最陡路徑法即為一探索極值點所在區域的有效方法。

最陡路徑之設計是以因子設計的中心點作為起點，沿主效果分析所得知之方向設定實驗點，並以固定距離逐漸向前延伸。然後，比較前後實驗點之回應值，若次一個實驗點的函數評估值較前一個函數評估值為小(以最小化問題為例)，則表示仍在往極值點逼近，於是依此路徑繼續延伸，直到實驗點的函數評估值上升為止。

在本研究中，利用最陡路徑法於直交表微調機制中強化區域解搜尋的求解速度，於 3.2 節詳述。

3.2 直交調和搜尋演算法

本小節將介紹調和搜尋演算法的演算步驟，主要分為初始解與演算法搜尋兩個部分進行說明，並以最小成本最佳化問題做為探討對象。

3.2.1 初始解

在利用直交表創造初始解(initial harmony memory)時，先將可行空間進行分割，分割方式是將變數可行範圍最大之變數挑選出來，針對此緯度 x_p 切割為 S 等分[43]。

$$u_p - l_p = \max_{1\le i\le m}(u_i - l_i)$$

$$S = \begin{cases} 10, \text{ if } \max\limits_{1\le i\le m}(u_i - l_i) \le 100 \\ 20, \text{ if } \max\limits_{1\le i\le m}(u_i - l_i) > 100 \end{cases}$$

其中，u_p 與 l_p 分別為變數 x_p 的上界與下界。將此緯度切割為 S 等分，此緯度的每個等分上下界分別為 $[l(1), u(1)]$, $[l(2), u(2)]$,....,$[l(S), u(S)]$。

$$l(i) = l_p + (i-1)\left(\frac{u_p - l_p}{S}\right)$$

$$u(i) = u_p - (S-i)\left(\frac{u_p - l_p}{S}\right) \qquad \text{其中，} i = 1, 2, ..., S$$

並在 S 個子空間裡頭，分別利用水準數(Q)為 3、變數個數為 m 之直交

表 $L_M(Q^m)$，在均勻的空間裡頭進行 M 次實驗，總實驗次數為 $M \times S$ 次，由 $M \times S$ 次實驗中選取出最好的 HMS 次實驗解作為初始解 HM。

舉例說明，假設一個最佳化問題有三個變數，變數範圍為 $0.5 \le x_1 \le 10.5$、$3.5 \le x_2 \le 6.5$、$4.5 \le x_2 \le 7.5$，參數設定水準數 $Q = 3$，子空間個數 $S = 5$。則 5 個子空間如下：

$$[l(i),\ u(i)] = [(2i - 1.5,\ 3.5,\ 4.5),\ (2i + 0.5,\ 6.5,\ 7.5)] \quad \text{for} \quad i = 1,\ 2,\ ...,\ 5$$

在第一個子空間 $[l(1),\ u(1)] = [(0.5,\ 3.5,\ 4.5),\ (2.5,\ 6.5,\ 7.5)]$ 進行量化後可得三個變數的各個水準值：

$$\begin{cases} \alpha_1 = (0.5,\ 1.5,\ 2.5) \\ \alpha_2 = (3.5,\ 5.0,\ 6.5) \\ \alpha_3 = (4.5,\ 6.5,\ 7.5) \end{cases}$$

並應用直交表 $L_{3^2}(3^3)$ 進行 9 次實驗。

$$\mathrm{OA} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 2 \\ 1 & 3 & 3 \\ 2 & 1 & 2 \\ 2 & 2 & 3 \\ 2 & 3 & 1 \\ 3 & 1 & 3 \\ 3 & 2 & 1 \\ 3 & 3 & 2 \end{bmatrix} \rightarrow \mathrm{OA}_{\mathrm{quantize}} = \begin{bmatrix} 0.5 & 3.5 & 4.5 \\ 0.5 & 5.0 & 6.0 \\ 0.5 & 6.5 & 7.5 \\ 1.5 & 3.5 & 6.0 \\ 1.5 & 5.0 & 7.5 \\ 1.5 & 6.5 & 4.5 \\ 2.5 & 3.5 & 7.5 \\ 2.5 & 5.0 & 4.5 \\ 2.5 & 6.5 & 6.0 \end{bmatrix}$$

同樣步驟在其他子空間各進行 9 次實驗，共進行了 $9 \times 5 = 45$ 次實驗，從 45 次實驗中挑選出函數評估值最好的 HMS 個作為初始解 HM。

3.2.2 演算法搜尋

得到初始解 HM 後，調和搜尋演算法開始進行搜尋。OHS 主要有三項搜尋機制：直交交配(orthogonal crossover)、直交微調(orthogonal pitch adjustment)、隨機搜尋(random)。演算法每一代進行都有機率執行三項機制之一，有 $HMCR \times PAR$ 的機率執行直交微調，有 $HMCR \times (1 - PAR)$ 的機率執行直交交配，有 $1 - HMCR$ 的機率在可行空間隨機產生一組解，OHS 新解產生機制與機率關係圖如圖 3.3，進一步說明如後。

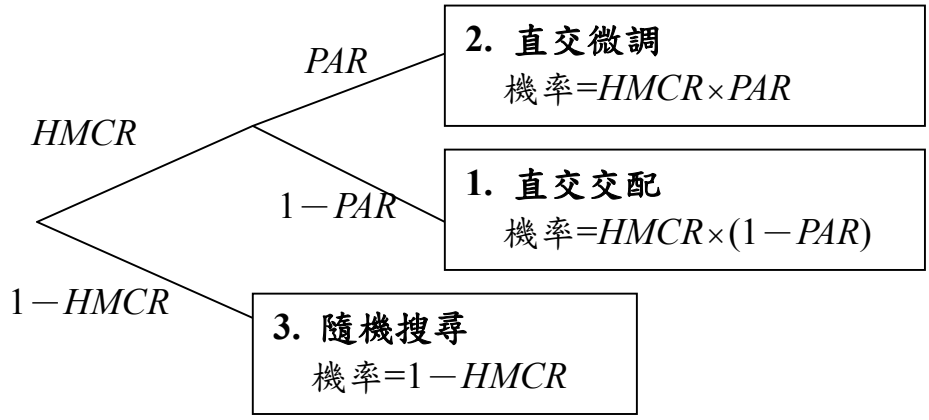


圖 3.3 OHS 新解產生機制與機率關係圖

由於直交表實驗次數可能會因為因素個數的增加而上升，因此採用變數分群的方法，控制直交表之實驗次數。

● **變數分群**

首先，將 m 個變數分為 F 群，即直交表的因素個數為 F，產生水準數 Q=3、因素個數為 F 之直交表 $L_M(Q^F)$。

舉例說明，假設有 5 個變數 $x_1, x_2, ..., x_5$，當 F=4 時隨機將 5 個變數分為 4 群 $\{(x_1), (x_2), (x_3), (x_4, x_5)\}$，則原始直交表 OA 擴充為 $\mathrm{OA}_{\mathrm{new}}$，而實驗次數控制為 9 次，實驗次數不會因變數個數而上升。

$$\mathrm{OA}=\begin{bmatrix}1&1&1&1\\1&2&2&2\\1&3&3&3\\2&1&2&3\\2&2&3&1\\2&3&1&2\\3&1&3&2\\3&2&1&3\\3&3&2&1\end{bmatrix}\rightarrow \mathrm{OA}_{\mathrm{new}}=\begin{bmatrix}1&1&1&1&1\\1&2&2&2&2\\1&3&3&3&3\\2&1&2&3&3\\2&2&3&1&1\\2&3&1&2&2\\3&1&3&2&2\\3&2&1&3&3\\3&3&2&1&1\end{bmatrix}$$

OHS 在每一代演化時，一旦使用到直交表且變數個數大於 F 值，則將變數隨機分群，否則則採用因素個數為 m 之直交表 $L_M(Q^m)$。

以下針對 OHS 三項機制操作進行說明：

1. 直交交配

(1) 執行機率：每一代進行都有 $HMCR\times(1-PAR)$的機率執行直交交配。

(2) 功能：透過直交交配在空間中做全域搜尋。

(3) 操作方式：

首先，由 *HM* 中選出 2 組解 $X_1=[x_1^1,x_2^1,...,x_m^1]$ 與 $X_2=[x_1^2,x_2^2,...,x_m^2]$，由這兩個解則可在求解空間中包圍出一個子空間，空間範圍為 $[l_{parents}, u_{parents}]$。

$$l_{parents}=[\min(x_1^1,x_1^2),\min(x_2^1,x_2^2),...,\min(x_m^1,x_m^2)]$$

$$u_{parents}=[\max(x_1^1,x_1^2),\max(x_2^1,x_2^2),...,\max(x_m^1,x_m^2)]$$

透過(公式 3.1)將此空間進行量化，利用直交表 $L_M(Q^F)$ 均勻在此空間均勻產生實驗點。再經由直交表的實驗結果進行主效果分析，找出各變數使函數評估值最小的水準值 X_{new}。

如果 X_{new} 函數評估值優於 *HM* 中的其中一組解，則保留 X_{new} 至調和記憶空間(*HM*)裡，並淘汰 *HM* 中最差的一組解，此步驟完成，即更新完 *HM*。如果滿足終止條件則停止，否則進行下一代演算。

(4) 圖形概念：

以二維空間為例，如圖 3.4 與圖 3.5 所示。自 HM 中隨機選取兩點 A 與 B，若兩點位於相同的區域解時，如圖 3.4 可發現以 A 與 B 點之各緯度最大值與最小值所包圍出的空間，由直交表在此空間執行均勻實驗，實驗後經由主效果分析得知 X_{new} 點為主效果推薦之最佳實驗點，然而 A、B 與 X_{new} 都位於相同的區域解裡頭。若兩點位於不同的區域解時，如圖 3.5 可發現與上述相同步驟執行後，主效果推薦之最佳實驗點 X_{new} 點位於不同的區域解裡頭，此乃此機制跨越其他區域解之概念。

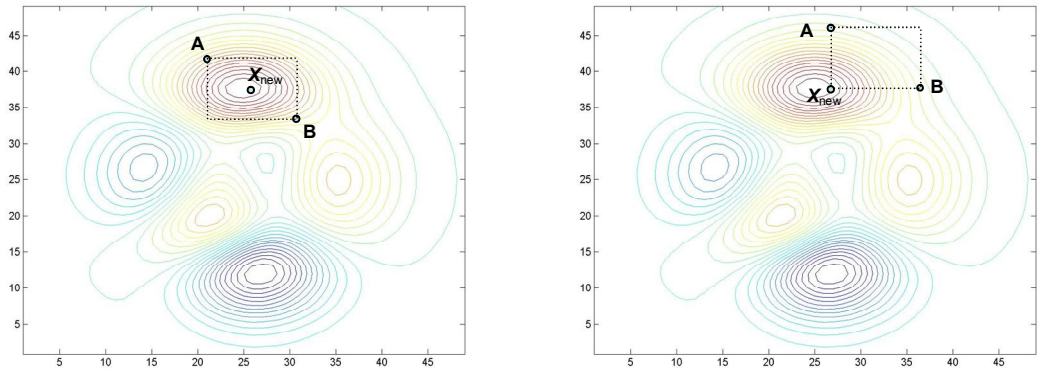


圖 3.4 直交交配選取的兩點於相同區域解時之搜尋

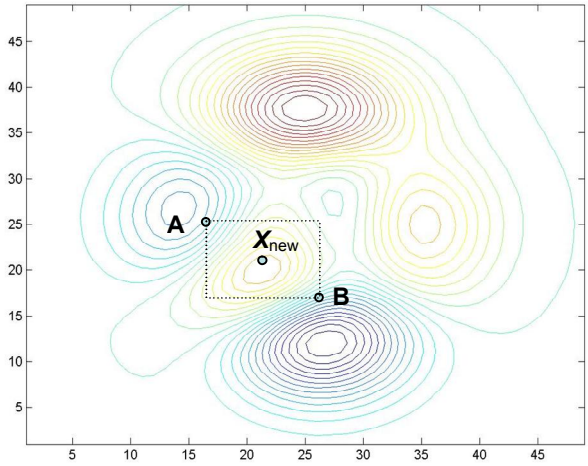


圖 3.5 直交交配選取的兩點於不同區域解時之搜尋

2. 直交微調

(1) 執行機率：每一代進行都有 *HMCR*×*PAR* 的機率執行直交微調。

(2) 主要功能：透過直交微調做單一區域搜尋，以求得靠近此區域最佳解的值。

(3) 操作方式：

從調和記憶空間(*HM*)中隨機選出一組解 $X=[x_1,x_2,...,x_m]$，利用小幅度的變動產生 $X_1=[x_1^1,x_2^1,...,x_m^1]$ 與 $X_2=[x_1^2,x_2^2,...,x_m^2]$，其中 x_i^1 與 x_i^2 產生公式為：

$$x_i^1=x_i+\overline{x_i};\ \ x_i^2=x_i-\overline{x_i}\quad \text{for } i=1, 2, \ldots, m$$

其中，$\overline{x_i}$ 服從平均數為 0、標準差為 *bw* 的常態分配，*bw* 為演算法參數。亦即，$\overline{x_i}\rightarrow N(0,\ bw)$。

所產生上界 X_1 與下界 X_2 就圍出個搜尋空間，透過直交表 $L_M(Q^F)$ 均勻在此空間均勻產生實驗點，再經由直交表的實驗結果進行主效果分析，找出各變數使函數評估值最小的水準值 X_{new}。

從 X 透過直交表找到新解 X_{new}，等同於在一個反應區面找到最佳化的前進方向與距離 $\Delta X=X_{new}-X$。因此透過最陡路徑法的概念，尋找 $X+a\Delta X$ 的解是否更好(for a=1,2, …,5)，如果有比較好，則依此方向繼續搜尋，直到沒有改善或 a>5 為止。此處設定 a 最大值為 5 是因為在微調機制底下，微調的步距如果過小，而演算法可能在此處永無止盡搜尋，為了避免這樣的

現象，每代採用最陡路徑法的次數以 5 次為上限。

將最終搜尋的較優解 $X+a\Delta X$ 保留至調和記憶空間(*HM*)裡，並淘汰 *HM* 中最差的一組解，此步驟完成，即更新完 *HM*。如果滿足終止條件則停止，否則進行下一代演算。

(4) 圖形概念：

以二維空間為例，如圖 3.6 所示，O 點為由 *HM* 中隨機選出之一組解，經過常態分配隨機找出各緯度之上界 X_1 與下界 X_2，由直交表在此空間執行均勻實驗，實驗後經由主效果分析得知 A 點為此空間最好的前進方向，以最陡路徑法從 O 點至 A 點相同的路徑與距離持續前進，直到函數評估值不再改善為止，路徑為 O➔A➔B➔C➔D，由於 E 點函數評估值較 D 點差，因此前進到 D 點為止，並考量是否更新 D 點進入 *HM* 中。

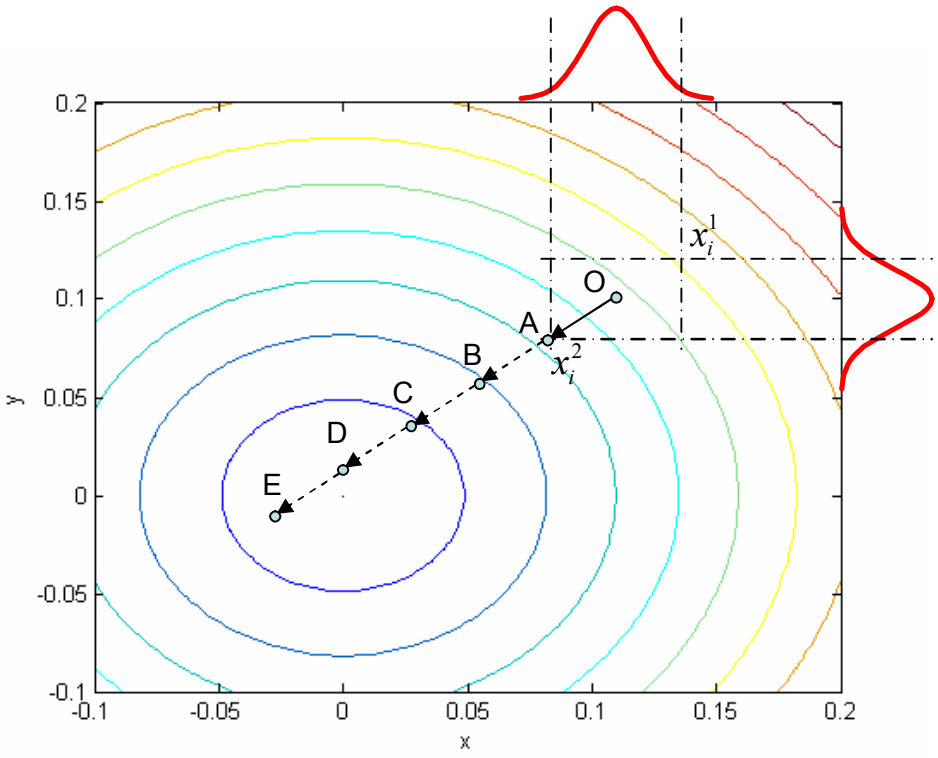


圖 3.6 OHS 直交微調機制示意圖

3. 隨機搜尋

(1) 執行機率：每一代進行都有 1－*HMCR* 的機率執行隨機搜尋。

(2) 主要功能：透過隨機搜尋在可行的空間持續搜尋，避免遺漏最佳解可能存在的區域。

(3) 操作方式：

演算法每一代進行都有 1－*HMCR* 的機率在可行空間隨機產生一組

解。如果此解優於 *HM* 中的其中一組解，則此隨機產生的解保留至 *HM* 中，並淘汰 *HM* 中最差的一組解，此步驟完成，即更新完 *HM*，此代演算完成。如果滿足終止條件則停止，否則進行下一代演算。

在本章中，我們可以歸納出數點結論：

1. 使用直交表方法搜尋時，每一變數必須先找出其上下界範圍，才可對應至其各水準值。每一變數的上下界決定後，相對的也在搜尋空間中圍出一個子空間，根據直交表在此空間中均勻的產生實驗點，再經由直交表的實驗結果進行主效果分析，找出各變數使函數評估值最小的水準值 X_{new}。
2. OHS 主要以機率型式執行三項機制搜尋：(1)直交交配：隨機取自 HM 中兩點，於兩點所包圍的空間中，執行直交表實驗及主效果分析，來做全域搜尋，(2)直交微調：隨機取自 HM 中一點，根據此點附近以常態隨機取一空間，執行直交表實驗、主效果分析及陡升路勁法，來做區域搜尋，(3)以隨機方式尋找其他可行解。
3. OGA 與 OSA 都是經由直交表改良過的演算法，結構操作上與 OHS 有些不同的地方，比較如表 3.4。
4. 本研究所提出 OHS 之虛擬碼如下：

step 1. ***(Initialization)***
Get initial solution vectors (*HM*) by OED in the feasible space.
step 2. If RN1 < *HMCR*
2.1 If RN2 < *PAR*
(Orthogonal Pitch)
Select one solution vector from HM and use OED to generate a candidate solution X in the pitch space.
2.2 If RN2 > *PAR*
(Orthogonal clossover)
Select two harmony vectors from HM and use OED to generate a candidate solution X in the space between them.
step 3. If RN1 > *HMCR*
(Random) Random generate a solution *X*.
step 4. Evaluate *X* and update *HM*.
step 5. ***(Termination)***
If stopping criterion is met, stop the algorithm. Otherwise, go to Step 2.

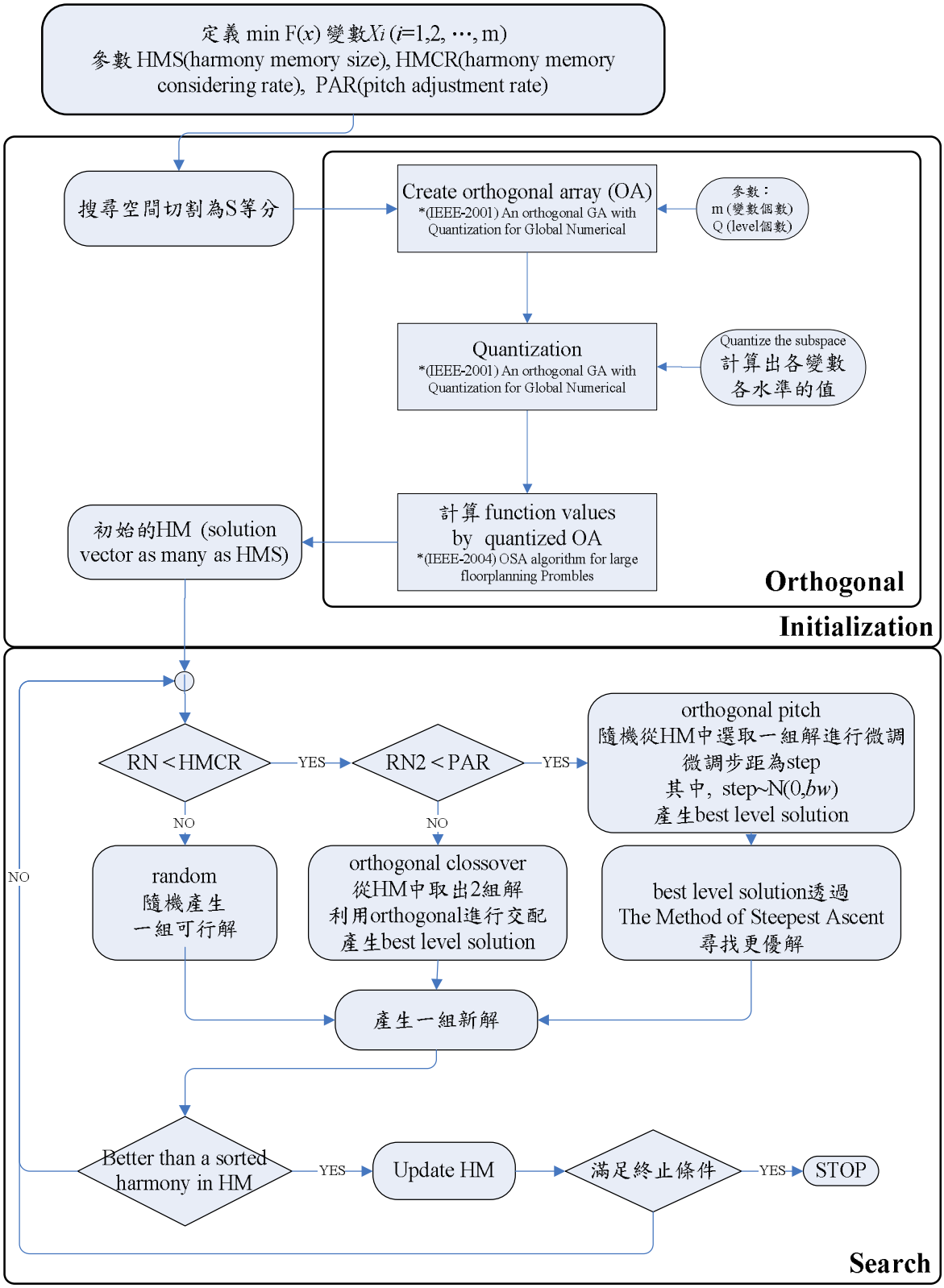


圖 3.7 OHS 細部流程圖

表 3.4 OGA、OSA、OHS 演算機制比較表

	OGA	OSA	OHS
初始解	透過直交表產生 保留最好的 G 條染色體至母體當中	隨機產生一組初始解	透過直交表產生 保留最好的 HMS 組解至記憶體當中
交配機制	由母體中選出成對的數條染色體進行直交表交配	無交配機制	由記憶體中選出兩組解進行直交交配。 OHS 每一代有 $HMCR\times(1-PAR)$ 的機率執行直交表交配
突變機制	母體中每條染色體都有機率被選出進行突變，被選出的染色體僅隨機突變某一變數之變數值	無突變機制 跳脫區域解的方式是以特定機率接受較差解的形式	無突變機制 跳脫區域解的方式是以隨機產生一組可行解的形式
微調機制	無微調機制	利用科西分配(t=1)隨機產生步距，利用此步距決定以決定直交表之高低水準數值	步距乃利用常態分配 N~(0, bw)隨機產生，利用此步距決定以決定直交表之高低水準。 OHS 每一代有 $HMCR\times PAR$ 的機率執行直交微調。 微調後如果有改善，則利用最陡路徑法深入搜尋
變數分群	變數分群為 F 群 F 可為實驗參數	變數分群為 F 群 $F=(3^{\lceil\log_3(2m+1)\rceil}-1)/2$ F 值決定於變數量 m	變數分群為 F 群 F 可為實驗參數
主效果分析	無	透過主效果分析求得一組可能為更好的解	透過主效果分析求得一組可能為更好的解

第四章 數值實驗與分析

4.1 實驗測試函數

本章節將依第三章所提的研究架構，再根據文獻中找出數個最佳化中代表性的測試函數進行實驗。本研究的實驗函數主要參考自比較對象OSA[55]、OGA[43]等文獻所使用的測試函數。本研究以最小化問題做為主要探討對象，測試的實驗將測試函數歸納為兩大類，分別為單一區域解函數與多重區域解函數，透過這兩大類函數的實驗以測試本研究所提 OHS 於各種函數類型之穩定性，測試函數詳見表 4.1。測試函數之特性與二維空間之示意圖如表 4.2。

表 4.1 本研究實驗之測試函數

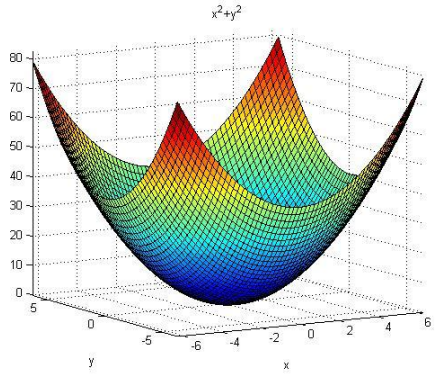
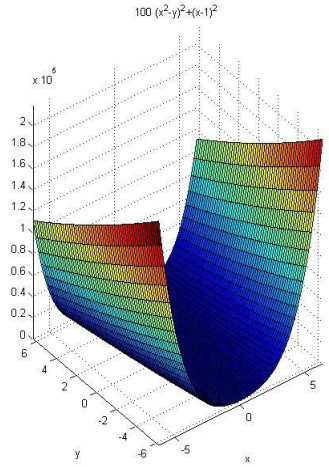
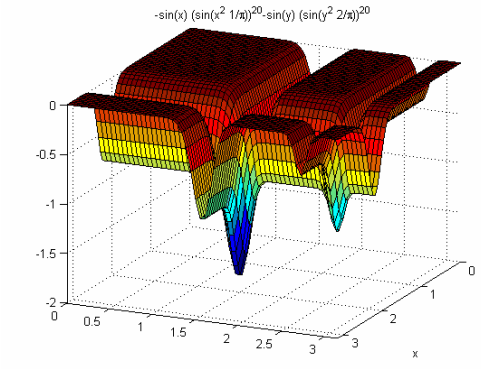
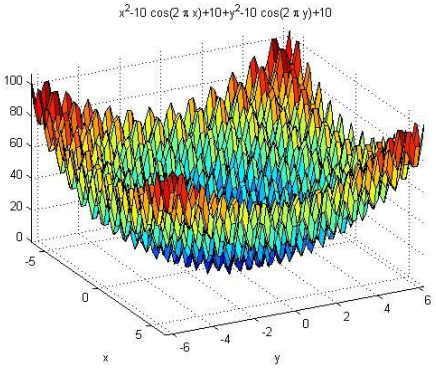
	測試函數	變數範圍	最佳值
單一區域解函數 (unimodal)	$f_1=\sum_{i=1}^{m}x_i^2$	$[-5.12, 5.12]^m$ m=30	0
	$f_2=\sum_{i=1}^{m-1}(100(x_i^2-x_{i+1})^2+(x_i-1)^2)$	$[-5.12, 5.12]^m$ m =10	0
多重區域解函數 (multi-modal)	$f_3=-\sum_{i=1}^{n}\sin(x_i)\sin^{2m}(\frac{i\times x_i^2}{\pi})$ and $n=10, m=10$	$[0, \ \pi]^m$ m =10	-9.66
	$f_4=\sum_{i=1}^{m}[x_i^2-10\cos(2\pi x_i)+10]$	$[-5.12, 5.12]^m$ m =30	0
	$f_5=20+e-20e^{-0.2\sqrt{\frac{\sum_{i}^{m}x_i^2}{m}}}-e^{\frac{\sum_{i}^{m}\cos(2\pi x_i)}{m}}$	$[-30, 30]^m$ m =30	0
	$f_6=\frac{1}{4000}\sum_{i=1}^{m}x_i^2-\prod_{i=1}^{m}\cos(\frac{x_i}{\sqrt{i}})+1$	$[-600, 600]^m$ m =10	0
	$f_7=\sum_{i=1}^{m}[\sin(x_i)+\sin(\frac{2x_i}{3})]$	$[3, 13]^m$ m =30	-36.4794
	$f_8=\sum_{i=1}^{m-1}[\sin(x_i+x_{i+1})+\sin(\frac{2x_ix_{i+1}}{3})]$	$[3, 13]^m$ m =10	-20

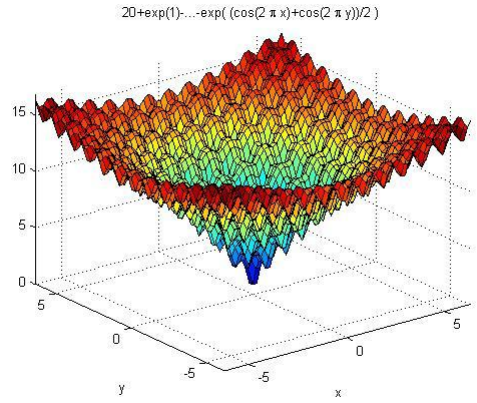
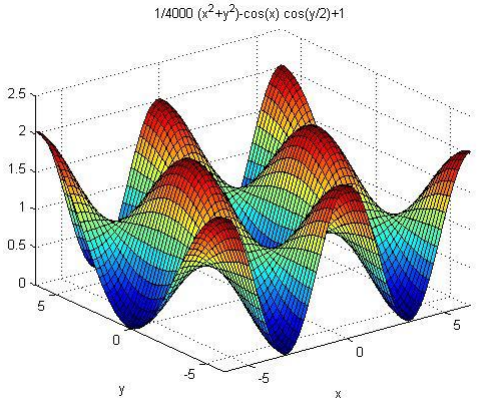
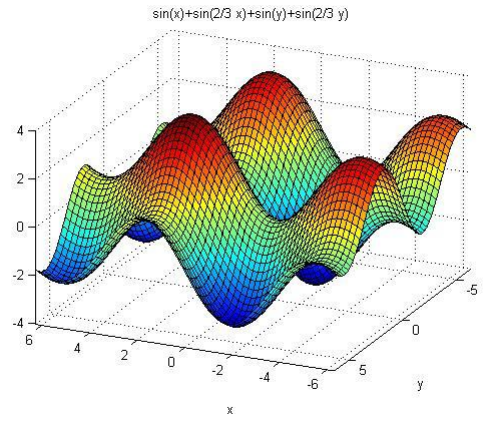
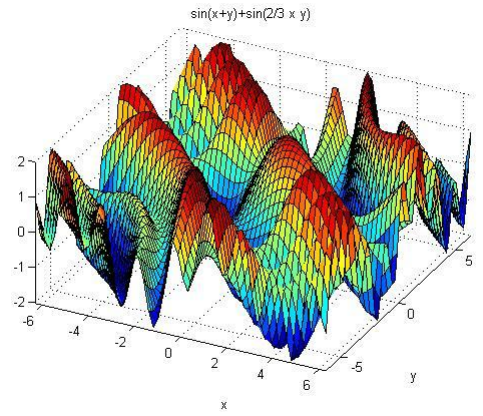
各演算法在各函數尋找最佳參數時，應在相同的終止條件下，進行參數實驗與比較。在實務上，決策者所在乎的是求解所消耗的時間與求解的精確度，因此本研究採用時間及誤差值作為實驗的終止條件。當函數評估值與最佳值之誤差值在規定範圍內則視為收斂，當演算法執行滿足時間終止條件或誤差終止條件(收斂條件)時，程式終止。本研究所採用各函數之終止條件如下，函數 1、函數 4、函數 6 與最佳值誤差範圍小於 1e-5 視為收斂，函數 2、函數 5、函數 7、函數 8 與最佳值誤差範圍小於 1e-3 視為收斂，函數 3 與最佳值誤差範圍小於 0.005 視為收斂，而時間終止條件除函數 2 於 120 秒明顯比較出勝負外，其他函數時間中止條件為 60 秒，即可判斷演算法搜尋能力之優劣。

本研究實驗環境之軟體與硬體規格如下：

1. Microsoft Windows XP
2. Intel Pentium IV
3. CPU 3.2 GHz
4. 512MB RAM
5. 使用程式語言 Matlab 7.0

表 4.2 測試函數之特性與二維空間之示意圖

測試函數及其特性	二維空間之示意圖	相關文獻
1. $f_1 = \sum_{i=1}^{m} x_i^2$ (Sphere Model) 2. 屬於單一區域解		[58] [43]
1. $f_2 = \sum_{i=1}^{m-1}(100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2)$ (Generalized Rosenbrock's Function) 2. 屬於單一區域解 3. 變數之間具有交互作用 4. 由圖形可知，此為較尖陡的函數，因此函數評估值對於變數敏感度較高		[58] [55] [43]
1. $f_3 = -\sum_{i=1}^{n}\sin(x_i)\sin^{2m}(\frac{i \times x_i^2}{\pi})$ and $n = 10, m = 10$ 2. 屬於多重區域解 3. 變數之間具有交互作用 4. 區域解個數多達約 10！個		[39] [48]
1. $f_4 = \sum_{i=1}^{m}[x_i^2 - 10\cos(2\pi x_i) + 10]$ (Generalized Rastrigin's Function) 2. 屬於多重區域解		[58] [55] [43]

測試函數及其特性	二維空間之示意圖	相關文獻
1. $f_5 = 20 + e - 20^{-0.2\sqrt{\frac{\sum_i^m x_i^2}{D}}} - e^{\sum_i^m \frac{\cos(2\pi x_i)}{D}}$ (Ackley's Function) 2. 屬於多重區域解		[58] [55] [43]
1. $f_6 = \frac{1}{4000}\sum_{i=1}^m x_i^2 - \prod_{i=1}^m \cos(\frac{x_i}{\sqrt{i}}) + 1$ (Generalized Griewank Function) 2. 屬於多重區域解		[58] [55] [43]
1. $f_7 = \sum_{i=1}^m [\sin(x_i) + \sin(\frac{2x_i}{3})]$ 2. 屬於多重區域解		[55]
1. $f_8 = \sum_{i=1}^{m-1} [\sin(x_i + x_{i+1}) + \sin(\frac{2x_i x_{i+1}}{3})]$ 2. 屬於多重區域解 3. 變數之間具有交互作用		[55]

4.2 參數實驗設計及說明

本研究參數實驗採用一次變動一個參數[57]的方式(one-variable-at-a-time strategy)進行實驗，每種參數組合獨立進行實驗 30 次。各函數經實驗後所採用的參數組合之優先性如下：

1. 收斂比例(convergence percentage)為 100%時，採用平均執行時間最短之參數。
2. 收斂比例在 0%與 100%之間時，採用收斂比例較高之參數；如果參數之收斂比例相同，則採用函數評估值最接近最佳解之參數。
3. 當收斂比例為 0%，表示執行時間消耗 100%，此時採用函數評估值最接近最佳解之參數。

其中，收斂比例的計算方式為：(達到收斂條件的次數/設定的測試次數)*100%。

本實驗採用參數之優先性原則亦為後續各演算法比較時之排名準則。

以 $f1$ 為例，實驗後採用之參數組合為 *HMS*=20、*HMCR*=0.9、*PAR*=0.1、*bw*=0.01、*F*=8、*Q*=3。以下將針對 *HMS*、*HMCR*、*PAR*、*bw* 等參數做變動並繪出其收斂圖形，以便探討該參數變動對於收斂速度所帶來的影響。

1. 參數 *HMS*

在多重區域解裡頭，參數 *HMS* 越小表示在空間裡頭保留的子代越少，相對對於搜尋的風險較大，可能遺失掉最佳解所在的區域。而 $f1$ 為單一區域解問題，因此所保留的子代個數不需要太多，因此當參數 *HMS* 越大收斂速度相對較慢些，如圖 4.1。

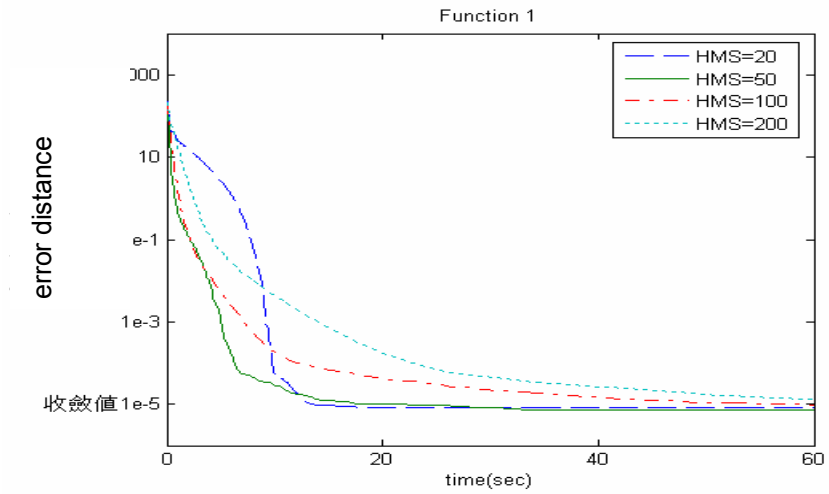


圖 4.1 參數 *HMS* 變化對於函數評估值之收斂影響

2. 參數 *HMCR*

參數 *HMCR* 控制下一代從記憶體中繼續演化的機率，相對的 1－*HMCR* 也控制了搜尋其他可行空間解的機率。在 $f1$ 中，*HMCR*=0.9 雖然在搜尋時間約第 10 秒以前的收斂速度相對較慢，但是約在第 10 秒以後的收斂速度反而較快些。這也說明，在搜尋的早期下一代由記憶體中的解進行演化的機率越高，其進步速度較慢，而後期由記憶體中的解進行演化時反而能夠以較快速度收斂，如圖 4.2。

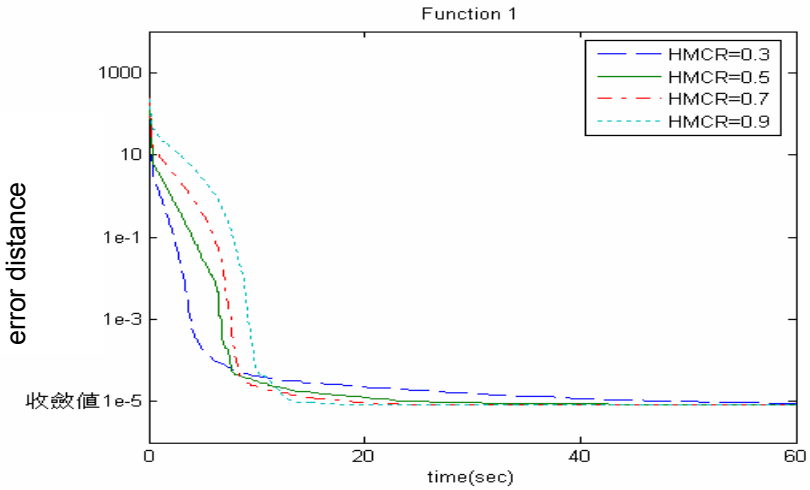


圖 4.2 參數 *HMCR* 變化對於函數評估值之收斂影響

3. 參數 *PAR*

參數 *RAR* 控制下一代從記憶體中挑出解進行直交微調的機率，相對的 1－*RAR* 也控制直交交配的機率。在測試 f 1 中，*RAR*=0.1 與 0.8 時能在十幾秒內快速收斂，表示此函數在直交微調機率大些或直交交配機率大些時，能有較快的收斂速度，如圖 4.3。

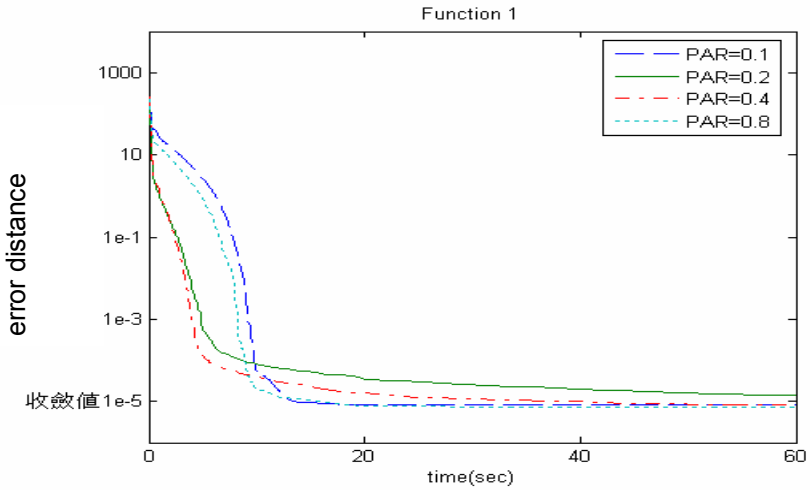


圖 4.3 參數 *PAR* 變化對於函數評估值之收斂影響

4. 參數 *bw*

參數 *bw* 控制直交微調移動時的步距。參數 *bw* 太大或太小都會使收斂速度較慢，選擇適當步距參數值，能使解快速向最佳解逼近。以 $f1$ 為例，當參數 *bw* 為 1(太大)或 0.001(太小)皆會影響求解速度，當參數 *bw* 為 0.01 收斂速度最快，如圖 4.4。

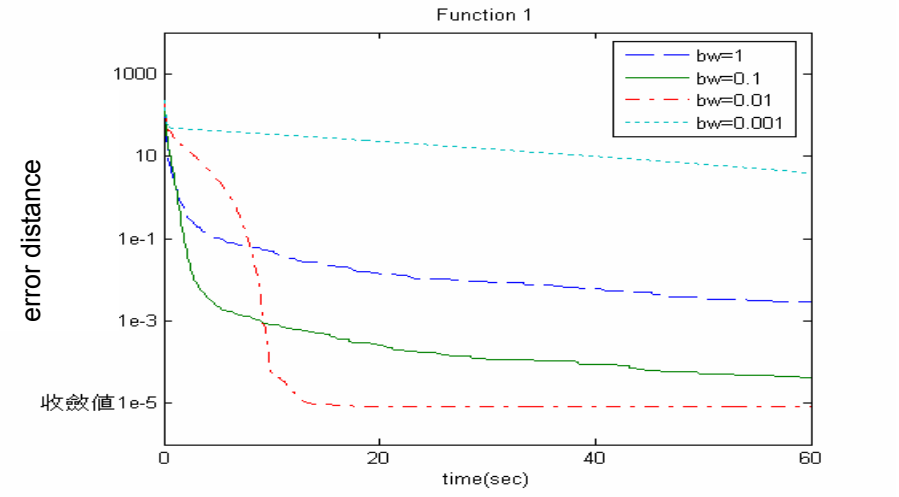


圖 4.4 參數 *bw* 變化對於函數評估值之收斂影響

在探討參數變動所帶來的影響後，對各參數使用說明整理如表 4.3。

表 4.3 各參數使用說明表

參數	參數說明與使用建議
HMS	調和記憶空間大小(Harmony memory size) 區域解個數越多，建議採用 *HMS* 值越大。
HMCR	採用調和記憶空間保留值的比例(Harmony memory considering rate)，其中 $0 \le HMCR \le 1$。 此參數間接掌握演算法搜尋 *HM* 以外的可行解比例，比例為 1－*HMCR*。*HMCR* 採用值通常在 0.5~0.9 之間。
PAR	微調比例(pitch adjustment rate)，其中 $0 \le PAR \le 1$。 此參數控制採用調和記憶空間保留值進行微調的比例。
bw	此參數控制決策變數微調步距，步距大小(step length)決定於平均數為 0，標準差為 *bw* 之常態分配。
F	*F* 為變數分群數。變數個數較少時，建議採用 *F*=4。 本研究中，變數個數為 10 採用 *F*=4，變數個數為 30 採用 *F*=4 或 8 作實驗。
Q	*Q* 為水準數，一般採用 *Q*=3。

4.3 OHS 與其他演算法實驗結果之比較

實驗結果除了與調和搜尋演算法比較之外，並與近年來著名期刊中與直交表相關且具有相當不錯的表現之演算法進行比較，如直交基因演算法、直交模擬退火演算法等，實驗結果如表 4.4。

後續將針對八個測試函數與四個演算法進行比較與探討，為了突顯各演算法搜尋的績效差異，就演算法之間的函數評估值與最佳值差距的倍數關係，及演算法之間所耗費的搜尋時間比較，定義下列兩個公式：

誤差倍數=(該演算法與最佳值之差距)÷(各演算法與最佳值差距之最小值)

若各演算法與最佳值差距之最小值為零，則不計算誤差倍數。

時間倍數=(該演算法搜尋消耗時間)÷(各演算法搜尋消耗時間之最小值)

以下將針對各函數會出四個演算法的函數評估值與最佳解誤差關係與搜尋消耗時間之倍數關係，並將演算法收斂情形依時間變化繪出。

函數 1 ：$f_1 = \sum_{i=1}^{m} x_i^2$

OGA 與 OHS 在 $f1$ 的收斂比例都為 100%，而 OHS 耗用時間 12.77 秒較 OGA 的 13.70 秒少些，時間差透過雙尾 T 檢定得 p-value=0.21，在 $\alpha = 0.05$ 時表示想者時間差異並不顯著。從收斂圖形可看出，OHS 於約 10 秒以前收斂速度比 OGA 慢，約 10 秒以後快速下降而比 OGA 更快達到中止條件。

而 HS 與 OSA 的收斂比例都為 0%，而 OSA 函數評估值 0.058 比 HS 函數評估值 0.03 較為接近最佳值。HS 與 OSA 耗用一分鐘後求得的解，距離最佳值的誤差與 OHS 相較之下約有數百倍之多，明顯落後 OHS 與 OGA。

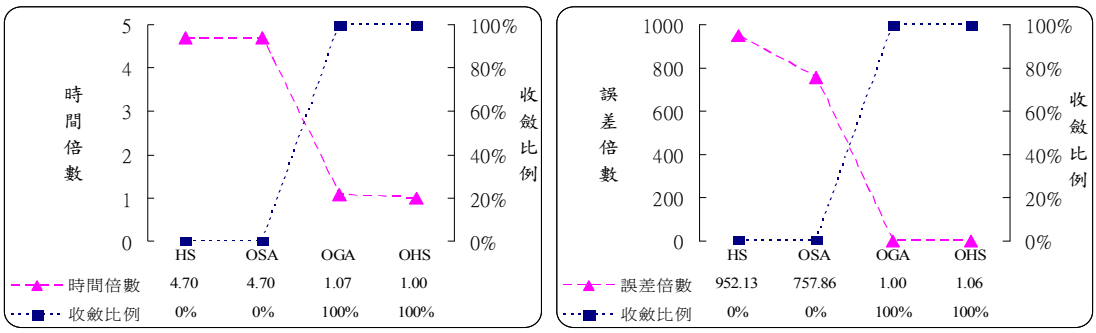


圖 4.5 各演算法執行 $f1$ 之時間倍數關係圖與誤差倍數關係圖

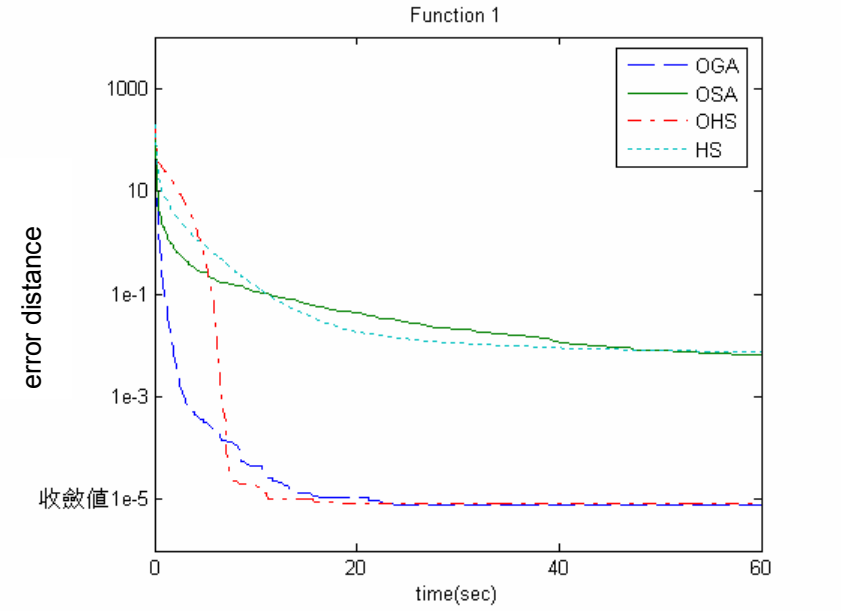


圖 4.6 各演算法於 $f1$ 之收斂圖形

函數 2 ：$f_2=\sum_{i=1}^{m-1}(100(x_i^2-x_{i+1})^2+(x_i-1)^2)$

各演算法在 $f2$ 收斂比例皆為 0%，因此比較各演算法在兩分鐘內的搜尋績效，最後各函數評估值與最佳值距離以 OHS 最接近，其次分別為 HS、OGA、OSA，OHS 對應 HS、OGA、OSA 之函數評估值差異，透過雙尾 T 檢定計算 p-value 分別為 0.016、7.65E-11、7.17E-20，在 α=0.05 時，OHS 與其他三者之差異顯著。

在搜尋時間約 5~30 秒時，HS 優於其他的演算法，在時間約 30~60 秒時，OHS 與 HS 相當接近，而 OGA 與 OSA 收斂速度趨近於零，在 60 秒之後，OHS 搜尋速度仍維持相當不錯的收斂速度，並比 HS 更接近最佳解，直到 120 秒時，OHS 仍保持一定的收斂速度優於 HS，而 OGA 與 OSA 在時間 30 秒以後收斂速度趨於 0，後續收斂能力略差。

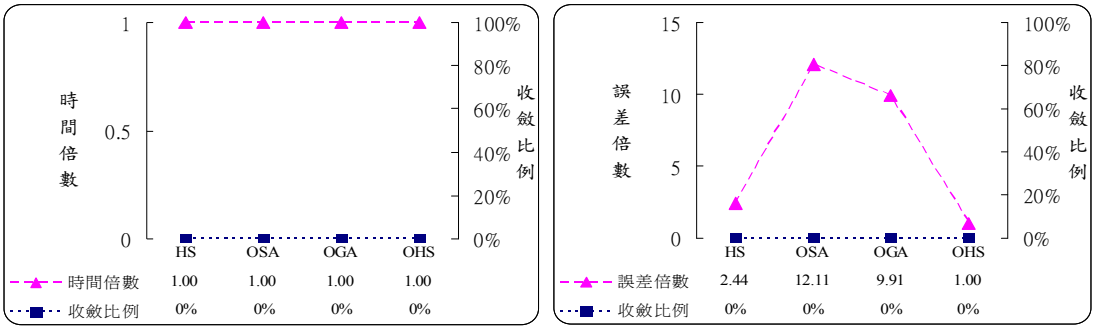


圖 4.7 各演算法執行 $f2$ 之時間倍數關係圖與誤差倍數關係圖

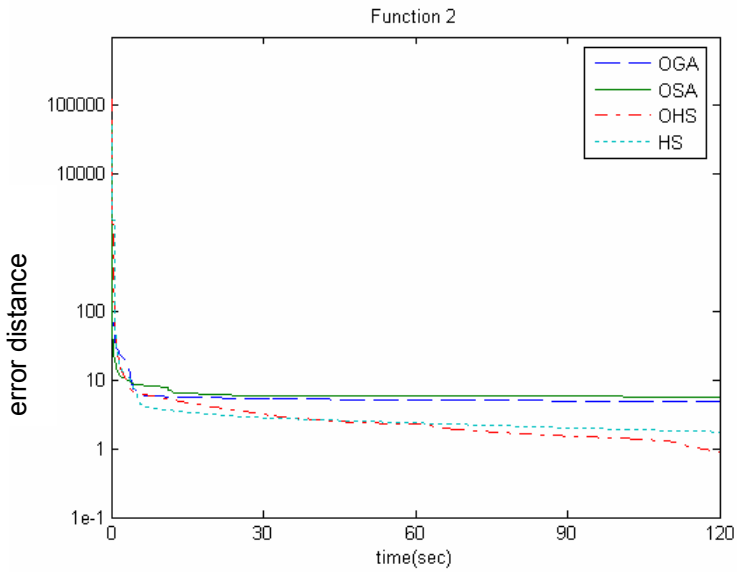


圖 4.8 各演算法於 $f2$ 之收斂圖形

函數 3 ： $f_3 = -\sum_{i=1}^{n} \sin(x_i)\sin^{2m}\left(\frac{i \times x_i^2}{\pi}\right)$

HS、OGA 及 OHS 在 $f3$ 收斂比例都達 100%，消耗時間分別為 7.73 秒、1.23 秒、5.28 秒，OGA 耗用時間都較 HS 與 OHS 少，OHS 時間差透過雙尾 T 檢定計算 p-value，OHS 對應 HS、OGA 之函數評估值差異，透過雙尾 T 檢定計算 p-value 分別為 0.17、1.90E-10，在 $\alpha = 0.05$ 的顯著水準下，OHS 與 OGA 差異顯著，OHS 與 HS 差異不顯著。OHS 搜尋所耗用時間較 OGA 多，OHS 與 HS 消耗時間為 OGA 的四至六倍。而 OSA 收斂比例為 0%，收斂比例最低，與最佳解的距離為 OHS 的十幾倍，耗用時間為 OHS 的數十倍。

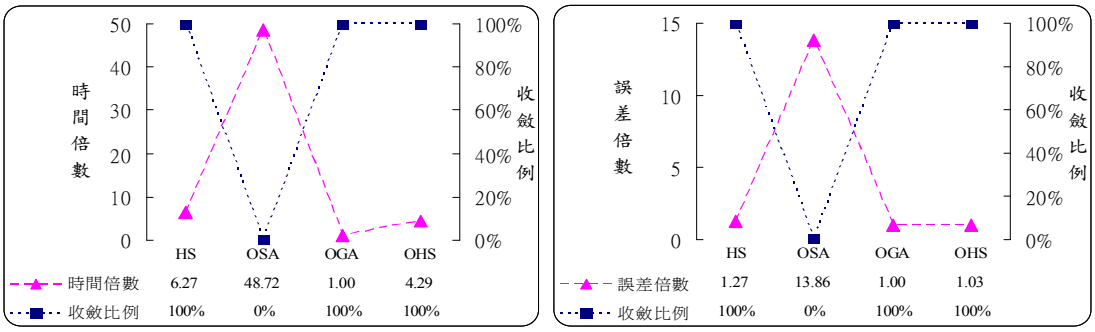


圖 4.9 各演算法執行 $f3$ 之時間倍數關係圖與誤差倍數關係圖

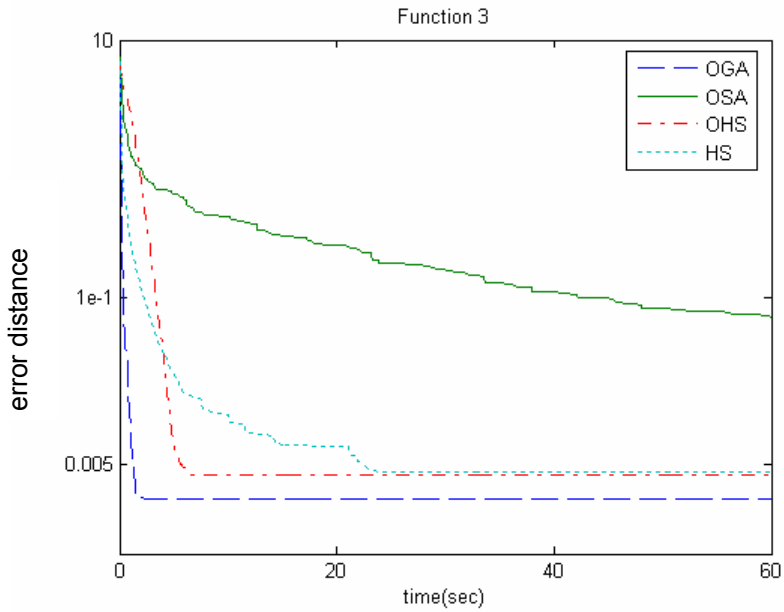


圖 4.10 各演算法於 $f3$ 之收斂圖形

函數 4 ： $f_4 = \sum_{i=1}^{m}[x_i^2 - 10\cos(2\pi x_i) + 10]$

在 $f4$ 中 OGA 與 OHS 經過直交表求得初始解後，能在搜尋演算數代內利用直交表交配的方式，恰巧切割到最佳解，此現象會在下一小節進一步說明與探討。而 OGA 平均耗用時間 0.41 秒找到最佳解，而 OHS 平均耗用時間 0.46 秒找到最佳解，時間差異經過雙尾 T 檢定得 p-value=0.0027，在 $\alpha = 0.05$ 的顯著水準下，OGA 平均耗用時間少於 OHS 平均耗用時間。

而 HS 與 OSA 收斂比例皆為 0%，OSA 函數評估值比 HS 更為接近最佳解，而兩者耗用時間為 OHS 與 OGA 的百倍之多。

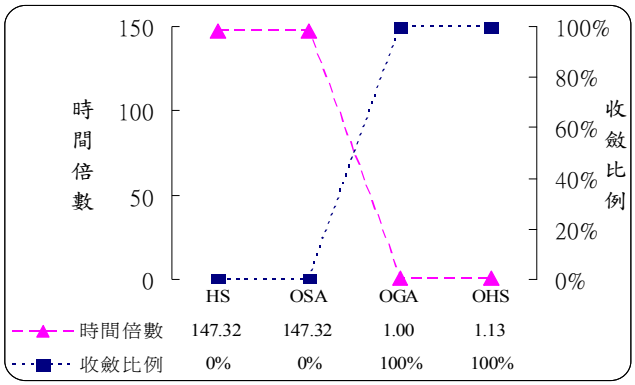


圖 4.11 各演算法執行 $f4$ 之時間倍數關係圖

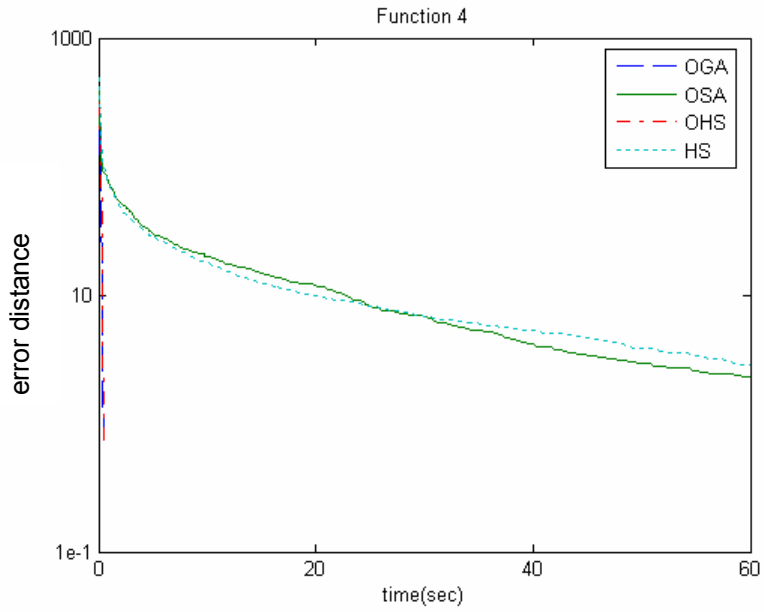


圖 4.12 各演算法於 $f4$ 之收斂圖形

函數 5 ：$f_5 = 20 + e - 20e^{-0.2\sqrt{\frac{\sum_i^m x_i^2}{m}}} - e^{\frac{\sum_i^m \cos(2\pi x_i)}{m}}$

OHS 在 $f5$ 的收斂比例為 100%，且耗用時間僅需 4.27 秒，而 OGA 收斂比例 10%，平均耗用時間 54.27 秒。HS 與 OSA 收斂比例為 0%，但 OSA 較 HS 函數評估值較為接近最佳解。

HS、OSA、OGA 所耗用時間為 OHS 的十幾倍之多，而與最佳解的距離分別為 OHS 的數萬倍、數千倍、數百倍之多。

從收斂圖形觀察，OHS 雖然在搜尋起步較慢些，但在系統化的主效果分析判斷搜尋區域後，求解速度快速逼近最佳解，因此，此函數 OHS 明顯優於其他各演算法。

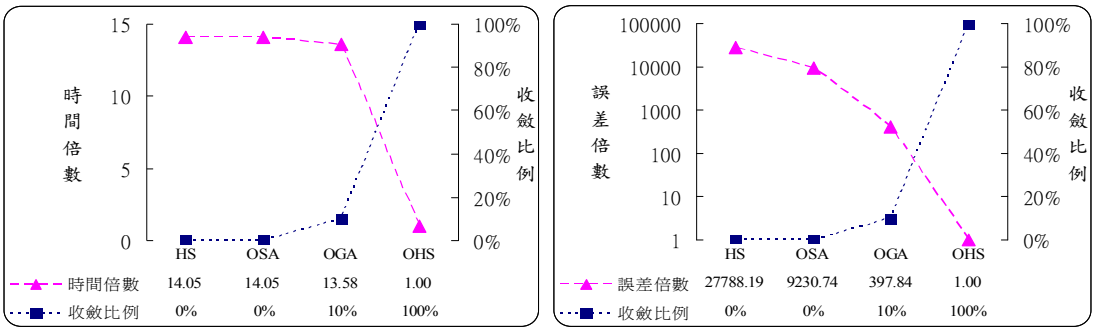


圖 4.13 各演算法執行 $f5$ 之時間倍數關係圖與誤差倍數關係圖

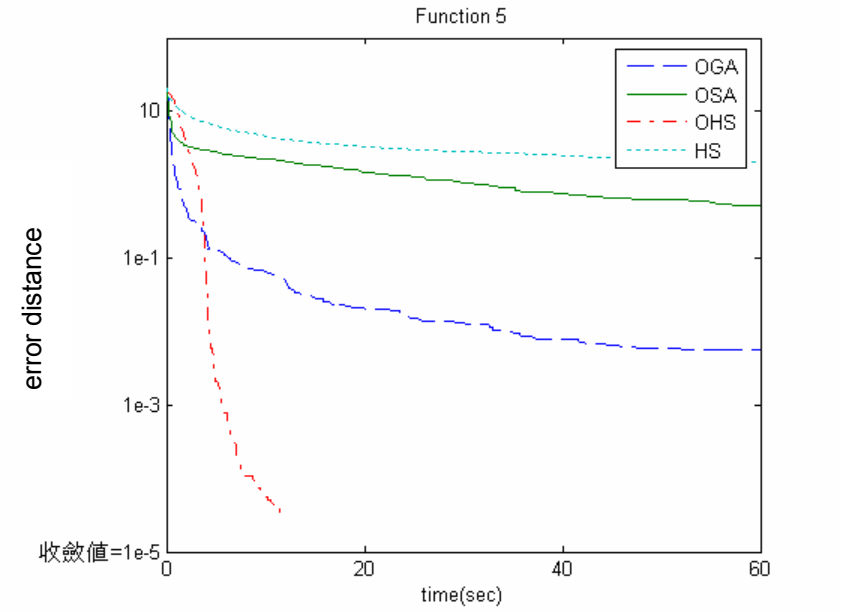


圖 4.14 各演算法於 $f5$ 之收斂圖形

函數 6 ：$f_6 = \frac{1}{4000}\sum_{i=1}^{m} x_i^2 - \prod_{i=1}^{m}\cos(\frac{x_i}{\sqrt{i}}) + 1$

在$f6$中 OGA 與 OHS 經過直交表求得初始解後，能在搜尋演算數代內利用直交表交配的方式，恰巧切割到最佳解，此現象會在下一小節進一步說明與探討。其中 OGA 平均耗用時間 0.24 秒找到最佳解，而 OHS 平均耗用時間 0.12 秒找到最佳解，時間差經過雙尾 T 檢定得 p-value= 1.71e-009，在$\alpha = 0.05$的顯著水準下，OHS 平均耗用時間少於 OGA 平均耗用時間。

而 HS 與 OSA 收斂比例皆為 0%，OSA 函數評估值比 HS 更為接近最佳解，而兩者耗用時間為 OHS 或 OGA 的數百倍之多。

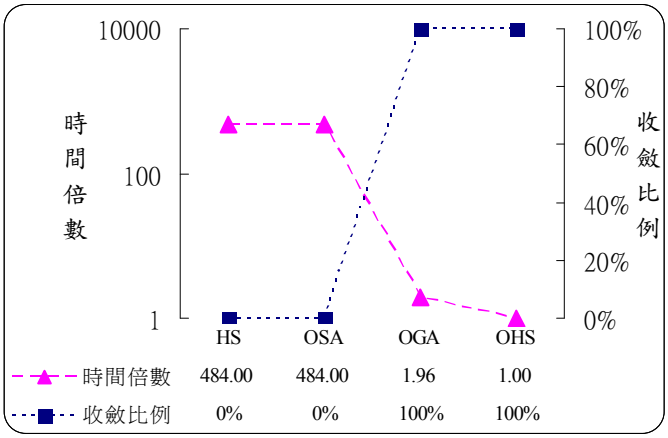


圖 4.15 各演算法執行$f6$之時間倍數關係圖

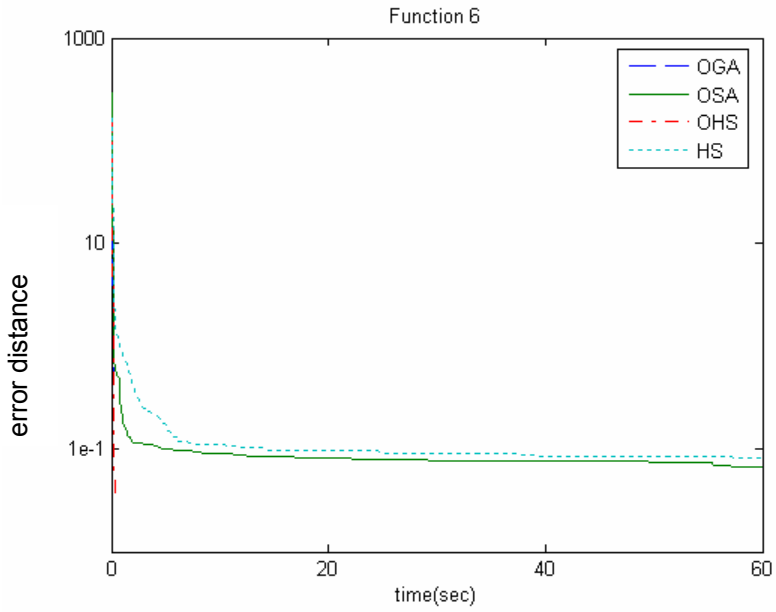


圖 4.16 各演算法於$f6$之收斂圖形

函數 7 ：$f_7 = \sum_{i=1}^{m}[\sin(x_i) + \sin(\frac{2x_i}{3})]$

在 f 7 中，OGA 與 OHS 收斂比例皆為 100%，OGA 平均耗用時間 7.6 秒找到最佳解，而 OHS 平均耗用時間 5.8 秒找到最佳解，時間差經過雙尾 T 檢定得 p-value=5.96e-07，在 $\alpha = 0.05$ 的顯著水準下，OHS 平均耗用時間少於 OGA 平均耗用時間。而 HS 與 OSA 收斂比例皆為 0%，HS 函數評估值比 OSA 更為接近最佳解。

觀察收斂圖形可發現，OHS 在早期起步速度略比 OGA 慢，在約 2~3 秒後，OHS 函數評估值才比 OGA 較為接近最佳解，比起 OGA 快達到收斂，而 HS 與 OSA 搜尋一分鐘後，離最佳解的誤差距離分別為 OHS 的數倍至一千多倍。

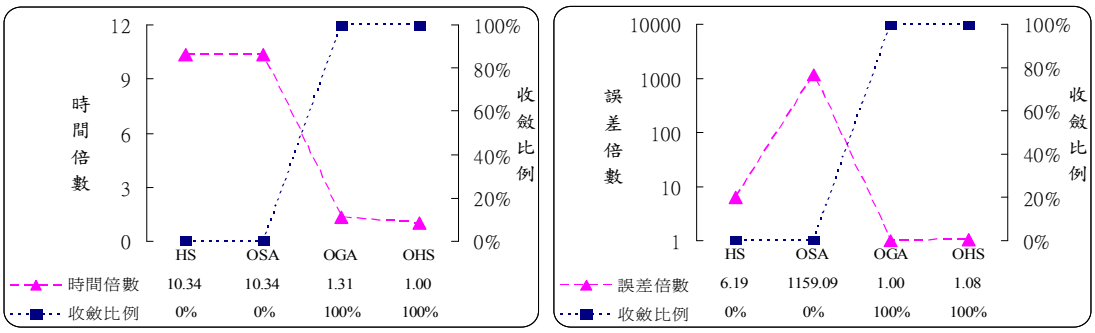


圖 4.17 各演算法執行 f7 之時間倍數關係圖與誤差倍數關係圖

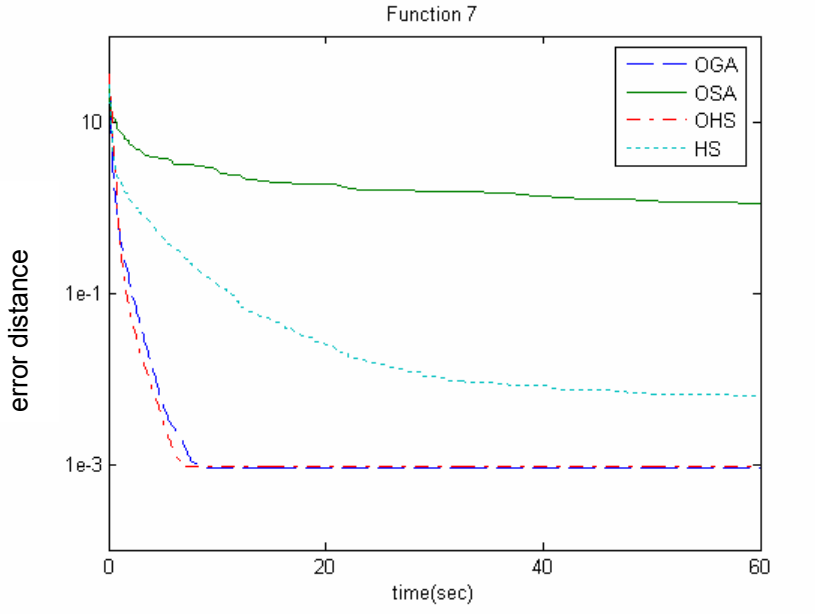


圖 4.18 各演算法於 f7 之收斂圖形

函數 8 ：$f_8 = \sum_{i=1}^{m-1}[\sin(x_i + x_{i+1}) + \sin(\frac{2x_i x_{i+1}}{3})]$

在 $f8$ 中，四個演算法的收斂比例皆為 0%，因此比較各演算法在 1 分鐘內的搜尋績效。而各函數評估值與最佳值距離以 OGA 最接近，其次分別為 OHS、OSA、HS，OHS 與 HS、OSA、OGA 之函數值差異透過雙尾 T 檢定計算 p-value 分別為 1.67E-04、7.92E-04、0.09。

OGA 雖然比起 OHS 接近最佳解，但是在 $\alpha = 0.05$ 的信心水準下，OHS 和 OGA 差異並不顯著，而 OHS 和 HS 與 OSA 之差異是顯著的。

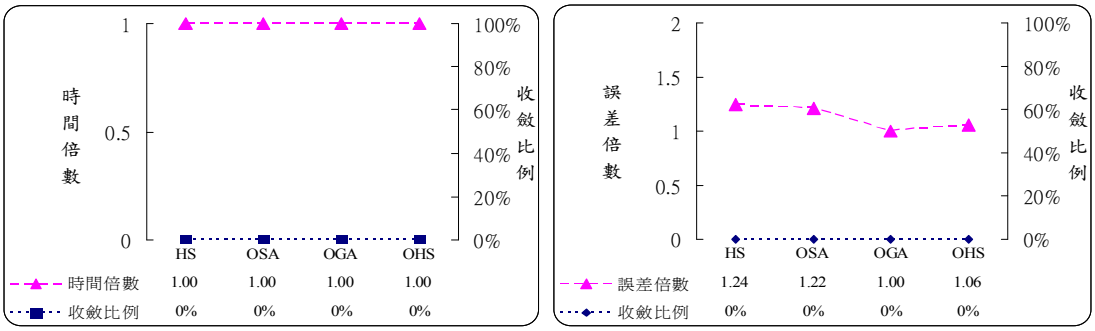


圖 4.19 各演算法執行 $f8$ 之時間倍數關係圖與誤差倍數關係圖

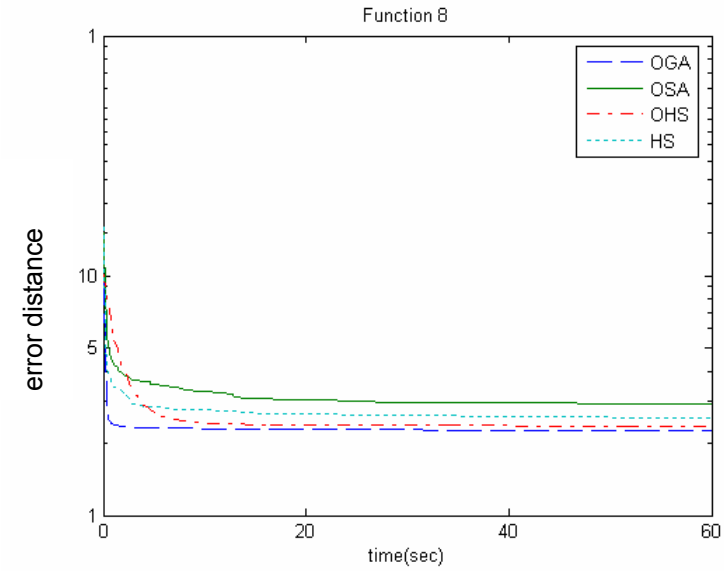


圖 4.20 各演算法於 $f8$ 之收斂圖形

由於在 $f4$ 與 $f6$ 中，OGA 與 OHS 都能夠在一秒內恰好切割到最佳解，本研究將於 4.4 節中針對 OGA 與 OHS 初始解之特徵進行探討。

表 4.4 OHS 與其他演算法在八個測試函數實驗結果與排名

	HS				OGA				OSA				OHS				目標函數之最佳值
	函數值 Avg. (STD.)	CPU time (sec.)	收斂比例	排名	函數值 Avg. (STD.)	CPU time (sec.)	收斂比例	排名	函數值 Avg. (STD.)	CPU time (sec.)	收斂比例	排名	函數值 Avg. (STD.)	CPU Time (sec.)	收斂比例	排名	
$f1$	0.0073 (0.0014)	60 (0)	0%	4	7.66E-06 (2.0E-06)	13.7 (5.18)	100%	2	0.0058 (0.0024)	60 (0)	0%	3	8.12E-06 (1.3E-06)	12.77 (3.61)	100%	1	0
$f2$	1.16 (1.11)	120 (0)	0%	2	4.6976 (2.7309)	120 (0)	0%	3	5.7422 (1.842)	120 (0)	0%	4	4.74E-01 (1.01)	120 (0)	0%	1	0
$f3$	-9.6557 (0.0013)	7.73 (5.42)	100%	3	-9.6566 (0.0014)	1.23 (0.49)	100%	1	-9.6129 (0.0261)	60 (0)	0%	4	-9.6565 (0.0011)	5.28 (0.78)	100%	2	-9.66
$f4$	1.7938 (0.9577)	60 (0)	0%	4	0 (0)	0.41 (0.07)	100%	1	1.7014 (0.6574)	60 (0)	0%	3	0 (0)	0.46 (0.04)	100%	2	0
$f5$	1.5375 (0.4422)	60 (0)	0%	4	4.84E-03 (0.0039)	54.27 (18.17)	10%	2	0.5107 (0.1334)	60 (0)	0%	3	5.53E-05 (2.1E-06)	4.27 (1.62)	100%	1	0
$f6$	0.0679 (0.0336)	60 (0)	0%	4	3.26E-07 (1.8E-06)	0.24 (0.14)	100%	2	0.0405 (0.0187)	60 (0)	0%	3	0 (0)	0.12 (0.05)	100%	1	0
$f7$	-36.474 (0.0005)	60 (0)	0%	3	-36.4785 (1.0E-04)	7.6 (0.81)	100%	2	-35.468 (0.242)	60 (0)	0%	4	-36.47846 (6.5E-05)	5.8 (0.87)	100%	1	-36.4794
$f8$	-17.186 (0.4686)	60 (0)	0%	4	-17.739 (0.1982)	60 (0)	0%	1	-17.2492 (0.4453)	60 (0)	0%	3	-17.6139 (0.3464)	60 (0)	0%	2	-20
Average Rank	3.5				1.75				3.375				1.375				
Final Rank	4				2				3				1				

4.4 OHS 與 OGA 初始解之特徵探討

在 $f4$ 與 $f6$ 中，OGA 與 OHS 經過直交表求得初始解後，能在搜尋演算數代演化內利用直交交配的方式，恰巧切割到最佳解，將此現象於本節說明。

在 OHS 與 OGA 搜尋前求初始解時，首先會先將空間進行切割，切割後的每個空間使用直交表均勻在該切割空間中找出分佈的點，從所有切割空間中的分佈點選取出 *HMS*(*G*)組較優的解作為初始解 *HM*(*population*)，進而進行演算法搜尋。但是這樣求初始解的作法對於某些特殊函數而言有機會發生恰巧切割到最佳解的情況。這些特殊函數通常具有的特徵，如函數圖形於最佳解兩端呈現對稱的情形，且最佳解變數值位於該變數範圍的中點等。本研究中，$f1$、$f4$~$f6$ 都具有這樣的特徵，而發生恰巧切割到最佳解的情形以 $f4$ 與 $f6$ 發生的機率最大，$f1$ 與 $f5$ 發生機率很小。

例如本研究中測試 $f6$，其變數範圍為-600~600 之間，最佳解於發生在點[0,0,0,0,0,0,0,0,0,0]，最佳值為 0，此函數發生恰巧切割到最佳解的情況的機率極高，以 OGA 求解 $f6$ 為例說明如下：

參數設定：*population size*=100、*pc*=0.9 、*pm*=0.1、*Q*=3、*F*=4

當 OGA 進行演算時，會從初始解中各緯度保留具有元素 0 的解，如下列的兩條染色體進行直交表交配

染色體 1	0	0	0	0	-600	-600	-600	0	600	0
染色體 2	-300	-600	-600	-600	0	0	0	0	0	0

其對應的直交表如下：

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 2 & 2 \\ 1 & 3 & 3 & 3 \\ 2 & 1 & 2 & 3 \\ 2 & 2 & 3 & 1 \\ 2 & 3 & 1 & 2 \\ 3 & 1 & 3 & 2 \\ 3 & 2 & 1 & 3 \\ 3 & 3 & 2 & 1 \end{bmatrix}$$

變數隨機分群，分為 4 群(F=4)：

{(X1,X2,X3), (X4,X5,X6,X7), (X8), (X9,X10)}

10 個變數分成 4 群後的直交表：

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \\ 1 & 1 & 1 & 3 & 3 & 3 & 3 & 3 & 3 & 3 \\ 2 & 2 & 2 & 1 & 1 & 1 & 1 & 2 & 3 & 3 \\ 2 & 2 & 2 & 2 & 2 & 2 & 2 & 3 & 1 & 1 \\ 2 & 2 & 2 & 3 & 3 & 3 & 3 & 1 & 2 & 2 \\ 3 & 3 & 3 & 1 & 1 & 1 & 1 & 3 & 2 & 2 \\ 3 & 3 & 3 & 2 & 2 & 2 & 2 & 1 & 3 & 3 \\ 3 & 3 & 3 & 3 & 3 & 3 & 3 & 2 & 1 & 1 \end{bmatrix}$$

對照直交表量化後的 9 次實驗及函數評估值如下：

$$\begin{bmatrix} -300 & -600 & -600 & -600 & -600 & -600 & -600 & 0 & 0 & 0 \\ -300 & -600 & -600 & -300 & -300 & -300 & -300 & 0 & 300 & 0 \\ -300 & -600 & -600 & 0 & 0 & 0 & 0 & 0 & 600 & 0 \\ -150 & -300 & -300 & -600 & -600 & -600 & -600 & 0 & 0 & 0 \\ -150 & -300 & -300 & -300 & -300 & -300 & -300 & 0 & 300 & 0 \\ -150 & -300 & -300 & 0 & 0 & 0 & 0 & 0 & 600 & 0 \\ 0 & 0 & 0 & -600 & -600 & -600 & -600 & 0 & 300 & 0 \\ 0 & 0 & 0 & -300 & -300 & -300 & -300 & 0 & 600 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{matrix} 563.499 \\ 315.994 \\ 293.493 \\ 411.625 \\ 164.142 \\ 141.165 \\ 383.496 \\ 180.804 \\ 0 \end{matrix}$$

像這樣兩條染色體，進行直交交配後，就會恰巧切割到該函數最佳解，然而在初始解中存在像這樣類型的兩條染色體組合相當多，恰巧切割到最佳解的狀況最容易發生搜尋第十代以前，第十代以前如果沒有恰巧切割到最佳解，那第十代以後母體結構像這樣成對的染色體比例就大幅下降。

由於真實世界中的決策問題求解不一定恰巧屬於這類型函數的特徵，因此為了讓 OHS 與 OGA 能夠在更貼近真實問題的函數中進行比較，而非比較演算法恰巧切割到最佳解的速度，本研究將函數 4 與 6 的變數範圍隨機偏移五個範圍，在相同大小的空間進行實驗。範圍變動後觀察 OHS 與 OGA 在非特殊狀況下比較兩個演算法的搜尋能力。函數 4 與函數 6 之五個隨機變動之變數範圍偏移如下表 4.5。

表 4.5 隨機對 $f4$ 與 $f6$ 之變數範圍做偏移

		變數範圍偏移	變數範圍
$f4$	Range 1	向左偏移 7.9173	(-37.9173, 22.0827)
	Range 2	向右偏移 3.7555	(-26.2445, 33.7555)
	Range 3	向左偏移 14.8544	(-45.1456, 15.1456)
	Range 4	向左偏移 18.5833	(-48.5833, 11.4167)
	Range 5	向右偏移 19.7963	(-10.2037, 49.7963)
$f6$	Range 1	向左偏移 267.7273	(-867.7273, 332.2727)
	Range 2	向右偏移 380.0517	(-219.9483, 980.0517)
	Range 3	向右偏移 219.2023	(-380.7977, 819.2023)
	Range 4	向右偏移 91.9923	(-508.0077, 691.9923)
	Range 5	向左偏移 315.3458	(-915.3458, 284.6542)

表 4.6 OGA 與 OHS 於 $f4$ 與 $f6$ 之五個變數範圍實驗結果

	OGA				OHS			
	函數值 Avg. (STD.)	CPU time (sec.)	收斂比例	排名	函數值 Avg. (STD.)	CPU Time (sec.)	收斂比例	排名
$f4$ Range1	0.0018 (0.0008)	60 (0)	0%	2	3.74e-05 (1.22e-05)	60 (0)	0%	1
$f4$ Range2	0.0014 (0.0011)	60 (0)	0%	2	3.08e-05 (1.08e-05)	60 (0)	0%	1
$f4$ Range3	0.0020 (0.0016)	60 (0)	0%	2	3.44e-05 (1.47e-05)	60 (0)	0%	1
$f4$ Range4	0.0019 (0.0011)	60 (0)	0%	2	0.0003 (0.0001)	60 (0)	0%	1
$f4$ Range5	0.0016 (0.0011)	60 (0)	0%	2	0.0003 (0.0001)	60 (0)	0%	1
$f6$ Range1	0.0216 (0.0152)	60 (0)	0%	2	0.0054 (0.0097)	37.00 (20.88)	56.67%	1
$f6$ Range2	0.0295 (0.0199)	60 (0)	0%	2	0.0064 (0.0053)	59.44 (2.53)	6.67%	1
$f6$ Range3	0.0443 (0.0265)	60 (0)	0%	2	0.0082 (0.009)	53.67 (14.50)	16.67%	1
$f6$ Range4	0.0263 (0.0175)	60 (0)	0%	2	0.0027 (0.0059)	31.11 (20.92)	66.67%	1
$f6$ Range5	0.0340 (0.0208)	60 (0)	0%	2	0.0034	44.53 (17.92)	46.67	1

$f4$ 與 $f6$ 各經過五次隨機偏移後進行實驗，實驗結果如表 4.6。由實驗結果可發現 OGA 與 OHS 於 $f4$ 之收斂比例皆為 0%，但在五個偏移的變數範圍中，OHS 皆能夠比 OGA 更為接近最佳解。在測試函數 6 的五個偏移的變數範圍中，可發現 OHS 的收斂比例都比 OGA 高些。

經由 $f4$ 與 $f6$ 的偏移實驗可發現 OHS 比起 OGA 能更接近最佳解，而 OGA 與最佳解的誤差倍數約為 OHS 的數十倍至數百倍。使得 OHS 能更逼近最佳解的原因除了系統化的主效果分析外，直交微調使得求解的精確度相對提升，經由本節實驗證實，$f4$ 與 $f6$ 在隨機求得的五個變數範圍底下，OHS 的搜尋能力或收斂比例皆優於 OGA。

4.5 實驗結論

根據 4.3 節的實驗結果，依函數別列表排名，而排名原則與 4.2 節中選用參數之原則相同。比較後的排名結果簡化後如表 4.7 所示，OHS 對各演算法差異之顯著性如表 4.8。

表 4.7 各演算法於各函數之排名與總排名

函數別	HS	OSA	OGA	OHS
$f1$	4	3	2	1
$f2$	2*	4	3	1*
$f3$	3	4	1*	2*
$f4$	4	3	1*	2*
$f5$	4	3	2*	1*
$f6$	4	3	2*	1*
$f7$	3	4	2*	1*
$f8$	4	3	1	2
Average Rank	3.5	3.375	1.75	1.37
Final Rank	4	3	2	1

(註) *為在 α=0.05 時，第一名與第二名之差異顯著

表 4.8 OHS 對其他演算法之差異顯著性

p-value	OHS to HS	OHS to OSA	OHS to OGA	備註
$f1$	3.88E-93**	3.88E-93**	0.21	時間差異
$f2$	0.016*	7.17E-20**	7.66E-11**	函數值差異
$f3$	0.17	4.03E-31**	1.91E-10**	時間差異
$f4$	1.56E-255**	1.56E-255**	2.68E-03**	時間差異
$f5$	1.10E-213**	1.10E-213**	2.98E-18**	時間差異
$f6$	7.18E-267**	7.18E-267**	1.71E-09**	時間差異
$f7$	2.06E-152**	2.06E-152**	5.96E-07**	時間差異
$f8$	1.67E-04**	1.06E-04**	0.283	函數值差異

(註) *與**分別為在α=0.05 與α=0.01 時，檢定差異顯著

演算法比較後結論：

1. OHS 在八個測試函數中，有五個函數排名第一，而三個函數排名第二(包括$f3$、$f4$、$f8$)。
2. OHS 在f 3 排名第二，搜尋平均時間較 OGA 多出了 4.05 秒鐘，在$\alpha = 0.05$ 的信心水準下是顯著的。
3. OHS 雖然在f 4 排名第二，搜尋時間較 OGA 多出了 0.05 秒鐘，但是在 4.4 節中，f 4 範圍經過隨機偏移五次進行實驗後，發現 OHS 的搜尋能力都比 OGA 好，表示 OGA 在$f4$ 變數範圍(-600, 600)的表現只是的特例。
4. OHS 在$f8$ 排名第二，OGA 平均的函數評估值比 OHS 接近最佳值，但是在$\alpha = 0.05$ 的信心水準下並不顯著，表示 OGA 與 OHS 的函數評估值兩者之差異不顯著。
5. 在各函數的收斂圖形當中，可以發現 OHS 在搜尋時間早期收斂速度比起其他演算法來的慢，但是 OHS 在中後期持續改善的能力較強。這樣的現象對照 OHS 的演算結構，我們可以發現，由於參數 *HMCR* 與 *PAR* 控制了 OHS 微調機制發生機率，因此 OHS 在搜尋早期直交微調機制可能造成些許時間上的浪費，但是微調機制在搜尋時間後期充分發揮區域搜尋的能力。
6. OSA 於早期搜尋效率不差，但對後期搜尋時由於其移動步距太大，導致搜尋最佳解的品質較差，另外，OSA 跳脫區域解採用機率的形

式進行，跳脫的速度相對也較慢，此兩點為 OSA 最大缺陷。

7. OGA 若初期搜尋時有母代可快速包圍住最佳解，OGA 搜尋速度就會快速收斂，但是一旦母代經過演化後如果能保留住包圍最佳解子代組合減少了，那 OGA 後續搜尋能力就大幅度衰退。反觀，OHS 初期搜尋亦有快速包圍最佳解的能力，經過演化後如選出能包圍最佳解的兩組解執行直交交配的機率下降了，OHS 尚具有直交微調機制做細部搜尋，透過微調機制 OHS 能夠利用接近最佳解的母代群一步一步逼近最佳解。
8. 在八個函數總排名後，OHS 排名第一，表示 OHS 的搜尋能力及搜尋穩定性優於其他三者。

第五章 結論與未來建議

本研究提出的直交調和搜尋演算法，主要是結合直交表實驗設計與調和搜尋演算法兩者之特色。直交調和搜尋演算法主要以三項機制運作搜尋：(1)以直交交配來做全域搜尋，搜尋最佳解可能存在的任一個區域解，(2)以直交微調來做區域搜尋，搭配最陡路徑法在區域解中向最佳解逼近，(3)以隨機方式尋找其他可行解。因此，直交調和搜尋演算法結合直交表實驗設計具有優良經驗的推理能力與主效果分析，促使演算法在廣大的搜尋空間中，能夠為子代判斷正確的搜尋區域與方向，快速向最佳解逼近以達到收斂並強化搜尋最佳解的能力。

經由本研究與各種類型函數的實驗結果可歸納出數點結論：

1. OHS 將 HS 之搜尋結構重新設計後，不再像 HS 搜尋一組解需要 m 次迴圈求得(m 為變數個數)，取而代之的是每一代只執行 OHS 三種機制之一，求解消耗時間大幅降低，此現象在變數個數增加時更為明顯。
2. 透過直交表推理能力與主效果分析為子代判斷正確的搜尋區域與方向，取代 HS 以隨機方式判斷搜尋方向，使得 OHS 能比 HS 以更快的速度找到更好的解。
3. 在單一區域解的函數中，由於 OHS 直交微調機制的主效果分析與最陡路徑法，使得 OHS 能選擇正確搜尋方向以達到更好的求解品質。
4. 在多重區域解問題中，OHS 在搜尋早期透過直交交配機制能快速包圍住可能存在最佳解之區域，到了搜尋中後期將保留在記憶體中的解選出，利用微調機制充分發揮區域搜尋的能力，快速逼近最佳解。
5. OHS 藉由直交交配全域搜尋與直交微調的區域搜尋，使得速度上能夠優於現有的優良直交演算法(OGA、OSA)，以證實本研究所提出之方法在求解這些函數的效率比其他比較演算法有更佳的效果。

本研究提出之調和搜尋演算法，未來如果能將參數 *HMCR*、*PAR*、*bw* 等學習模式導入本研究之演算法中，可以在演算的過程中自動調整，相信對於實驗結果或許會有幫助。

參考文獻

[1] 吳子逢，何信瑩。*使用直交實驗設計的高效率演化策略及其應用*，逢甲大學碩士論文，2003。

[2] 林宏穗，何信瑩。*設計一種新型的直交粒子群最佳化演算法*，逢甲大學碩士論文，2004。

[3] Arabas, J., Mulawka, J. and Pokrasniewicz, J., "A new class of the crossover operators for the numerical optimization," *Proceedings of the 6th International Conference on Genetic Algorithms*, Morgan Kaufmann, San Mateo, CA, pp. 42-48, 1995.

[4] Bäck, T., Evolutionary Algorithms in Theory and Practice. New York: Oxford Univ. Press, 1996.

[5] Cantù-Paz, E ., *A summary of research on parallel genetic algorithms*. IlliGAL Report No. 95007, University of Illinois at Urbana-Champaign, July, 1995.

[6] Carlos Andrés Peña-Reyes, Moshe Sipper, "Evolutionary Computation in Medicine: an overview," *Artificial Intelligence in Medicine*, 19, pp. 1-23, 2000.

[7] Colorni, A., Dorigo, M. and Maniezzo, V., "An investigation of some properties of an ant algorithm," Proceedings of the *Parallel Problem Solving from Nature Conference*, R. Manner and B. Manderick Eds. Brussels, Belgium: Elsevier, pp.509-520, 1992.

[8] Colorni, A., Dorigo, M. and Maniezzo, V., "Distributed optimization by ant colnies," Proceedings of the *First European Conference Artificial Life*, F. Varela and P. Bourgine, Eds. Paris, France: Elsevier, pp.134-142, 1991.

[9] Darrell Whitley, L. and Kauth, J., GENITOR: a different genetic algorithm, Proceedings of the *1988 Rocky Mountain Conference on Artifical Intelligence*, 1988.

[10] Darrell Whitley, L. and Starkweather, T., "GENITOR II: a distributed genetic algorithm," *Journal of Experimental and Theoretical Artifical Intelligence 2*, pp. 189-214, 1990.

[11] Darrell Whitley, L., The GENITOR algorithm and selective pressure: why rank based allocation of reproductive trials is best, in: J. D. Schaffer (Ed.), Proceedings of *the Third International Conference on GAs, Morgan Kaufmann, Los Atlos*, CA, pp. 116-121., 1989

[12] Dorigo, M., Bonabeau, E. and Theraulaz, G., "Ant algorithm and stigmergy," *Future Generation Computer Systems*, 16, pp. 851-871, 2000.

[13] Dorigo, M., Caro, G.D. and Gambarsella, L.M., "Ant algorithms for discrete optimization," *Artificial Life*, 5, pp. 137-172, 1999.

[14] Dorigo, M., Maniezzo, V. and Colorni, A., "Positive feedback as a research strategy," *Technology Report 91-016*, Politecnico di Milano, 1991.

[15] Eberhart, R.C. and Kennedy, J. (1995) "A new optimizer using particle swarm theory," Proc. 6th Int. Symp. Micro Machine and Human Science, Nagoya, Japan, pp.39-43.

[16] Eberhart, R.C. and Shi, Y. (2001) "Particle Swarm Optimization: Developments Application and Resources," Proceedings of the 2001 Congress on Evolutionary Computation, Vol.1, pp. 81–86.

[17] Engelbrecht, A.P. and Ismail, A. (1999) "Training product unit neural networks," Stability Control: Theory Appl., Vol. 2, No. 1-2, pp. 59-74.

[18] Eberhart, R.C. and Hu, X. (1999) "Human tremor analysis using particle swarmoptimization," in Proc. Congr. Evolutionary Computation, Washington, DC, pp. 1927-1930.

[19] Eshelman, L. J. and Schaffer, D., Preventing premature convergence in genetic algorithms by preventing incest, in: L. Booker, R. Belew (Eds.), Proceedings of the Fourth International Conference on GAs, Morgan Kaufmann, Los Atlos, CA, 1991.

[20] Eshelman, L. J. and Schaffer, J. D., Real-coded genetic algorithms and intervalschemata. In *Foundations of Genetic Algorithms 2 (FOGA-2)*, pp.187-202, 1993.

[21] Fogel, L. J., "Autonomous automata," *Ind. Res.*, vol. 4, pp. 14–19, 1962.

[22] Fogel, L. J., "On the organization of intellect," Ph.D. dissertation, University of California, Los Angeles, 1964.

[23] Fogel, L. J., Owens, A. J., & Walsh, M. J., Artificial intelligence through simulated evolution. New York: Wiley, 1966.

[24] Geem Z.W., Kim J.-H., Loganathan GV, "A new heuristic optimization algorithm: harmony search," *Simulation 76*(2), pp. 60-68, 2001.

[25] Geem Z.W., Kim J.-H., Loganathan GV, "Harmony search optimization: Application to pipe network design," *Int J Modell Simulat 22*(2), pp. 125-133, 2002.

[26] Geem Z.W., Lee S.-K., "A new structural optimization method based on the harmony search algorithm," *Computers and Structures 82*, 781-798, 2004.

[27] Geem Z.W., Lee S.-K., "A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice," *Comput. Methods Appl. Mech. Engrg. 194*, pp. 3902-3933, 2005.

[28] Glover F., "Heuristic for integer programming using surrogate constraints," *Decision Sci. 8*(1), pp. 156-166, 1977.

[29] Goldberg, D. E., Genetic Algorithms in Search, Optimazation and Machine Learning, Addison-Wesley, Reading, MA, 1989.

[30] Goldberg, D. E., Deb, K. and Korb, B.,Messy genetic algorithms revisited: Nonuniform size and scale. *Complex Systems 4*(4), pp. 415-444, 1990.

[31] Goldberg, D. E. and Deb, K., A comparison of selection schemes used in genetic algorithms. In *Foundations of Genetic Algorithms 1 (FOGA-1)*, pp. 69-93, 1991.

[32] Goldberg, D. E., Korb, B. and Deb, K., Messy genetic algorithms: Motivation, analysis and first results. *Complex Systems 3*(5), pp. 493-530, 1989.

[33] Ho S.-Y. and Chen J.-H., "A GA-based systematic reasoning approach for solving traveling salesman problems using an orthogonal array crossover," *High Performance*

Computing in the Asia-Pacific Region, 2000. Proceedings. *The Fourth International Conference/Exhibition on*, vol.2, no.pp.659-663 vol.2, 2000.

[34] Ho S.-J., Ho S.-Y, and Shu L.-S., "A Novel Orthogonal Simulated Annealing Algorithm for Optimization of Electromagnetic Problems," *IEEE TRANSACTIONS ON MAGNETICS 40*(4), 2004.

[35] Ho S.-Y., Ho S.-J., and Lin Y.-K., "An orthogonal simulated annealing for floorplanning problems," *IEEE Trans. VLSI Syst. 12*, 2004.

[36] Holland, J. H., "Outline for a logical theory of adaptive systems," *J. Assoc. Comput. Mach.*, vol. 3, pp. 297–314, 1962.

[37] Holland, J. H., Adaptation in Natural and Artificial Systems. Ann Arbor, MI: Univ. of Michigan Press, 1975.

[38] Holland, J. H. and Reitman, J. S., "Cognitive systems based on adaptive algorithms," in Pattern-Directed Inference Systems, D. A. Waterman and F. Hayes-Roth, Eds. New York: Academic, 1978.

[39] J. Yen, J. C. Liao, B. Lee and D. Randolph, A hybrid approach to modeling metabolic systems using a genetic algorithm and simplex method, *IEEE Transactions on System, Man, and Cybernetics-Part B: Cybernetics*, 28(2), pp. 173-191, 1998.

[40] Kirkpatrick S., Gelatt C. and Vecchi M., "Optimization by simulated annealing," *Science* 220(4598), pp. 671-680, 1983.

[41] Koza, JR., Genetic programming: a paradigm for genetically breeding populations of computer programs to solve problems. Technical Report STANCS-90-1314, Department of Computer Science, Stanford University, June 1990.

[42] Koza, JR., Genetic Programming. Cambridge, MA: MIT Press, 1992.

[43] Leung Y.-W. and Wang Y., "An orthogonal genetic algorithm with quantization for global numerical optimization," *IEEE Trans. Evol. Comput.* 5, pp. 41–53, 2001.

[44] Mitsuo Gen, Runwei Cheng, Genetic Algorithms and Engineering Design, Wiley, New York, 1997.

[45] Pierreval, H., Caux, C., Paris J.L. and Viguier, F., "Evolutionary Approaches to the Design and Organization of Manufacturing Systems," *Computers & Industrial Engineering*, 44, pp. 339-364, 2003.

[46] Rechenberg, I., Evolutions strategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution. Stuttgart, Germany: Frommann-Holzboog, 1973.

[47] Rechenberg, I., Evolutionstrategie '94. Frommann-Holzboog, Stuttgart, 1994.

[48] Renders, J.-M. and H. Bersini, "Hybridizing genetic algorithms with hill-climbing methods for global optimization: Two possible ways, " In Z. Michalewicz, J. D. Scha_er, H.-P. Schwefel, D. B. Fogel, and H. Kitano (Eds.), *Proceedings of the First IEEE International Conference on Evolutionary Computation*, pp. 312-317, IEEE Press, 1994.

[49] Schwefel, H.-P., Numerical Optimization of Computer Models. Chichester: Wiley, 1981.

[50] Schwefel, H.-P., Evolution and Optimum Seeking. New York: Wiley, 1995

(Sixth-Generation Computer Technology Series).

[51] Schwefel, H.-P., Evolution and Optimum Seeking, Wiley , New York, 1995.

[52] Schwefel, H.-P.,and Rudolph, G., "Contemporary evolution strategies," in *Advances in Artificial Life. 3rd Int. Conf. on Artificial Life* (Lecture Notes in Artificial Intelligence, vol. 929), F. Mor´an, A. Moreno, J. J. Merelo, and P. Chac´on, Eds. Berlin, Germany: Springer, pp. 893–907, 1995.

[53] Shi, Y. and Eberhart, R.C. (1999) "Empirical study of particle swarm optimization," in *Proc. Congr. Evolutionary Computation*, Washington, DC, pp. 1945-1949.

[54] Shi, Y. and Eberhart, R. C. (1998) "A Modified Particle swarm optimizer," Proc. IEEE 1nt. Conf. Evolutionary Computation, Anchorage, AK, pp. 69-73.

[55] Shu L.-S., Ho S.-Y., and Ho S.-J., "OSA: Orthogonal Simulated Annealing Algorithm and Its Application to Designing Mixed H_2/H_∞ Optimal Controllers," *IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS 34*(5), 2004.

[56] Syswerda, G., "Uniform crossover in genetic algorithms," *Proceedings of the 3rd International Conference on Genetic Algorithms*, pp.2-9, 1989.

[57] Wang T.-Y., Wu K.-B., A parameter set design procedure for the simulated annealing algorithm under the computational time constraint. Computer & Operations Research (26), pp. 665-678, 1999.

[58] Yao X., Liu Y., "Fast evolution strategies," in Evolutionary Programming VI, P. J. Angeline, R. Reynolds, J. McDonnell, and R. Eberhart, Eds. Berlin, Germany: Springer-Verlag, pp. 151-161, 1997. (Available at ftp://www.cs.adfa.edu.au/pub/xin/yao_liu_ep97.ps.gz).