國立交通大學

電機與控制工程學系

碩士論文

以關聯法則對於合作式粒子族群演算法訂立族群分類規則

Using Association Rules to Set the Classified Rules of Swarms for Cooperative Particle Swarm Optimization

研 究 生：高昆義

指導教授：林昇甫 博士

中華民國九十八年七月

以關聯法則對於合作式粒子族群演算法訂立族群分類規則

Using Association Rules to Set the Classified Rules of Swarms for Cooperative Particle Swarm Optimization

研 究 生：高昆義　　Student: Kun-Yi Kao

指導教授：林昇甫 博士　　Advisors: Dr. Sheng-Fuu Lin

國 立 交 通 大 學

電機與控制工程學系

碩士論文

A Thesis

Submitted to Department of Electrical and Control Engineering

College of Electrical Engineering

National Chiao-Tung University

in Partial Fulfillment of the Requirements

for the Degree of Master

in

Electrical and Control Engineering

July 2009

Hsinchu, Taiwan, Republic of China

中華民國九十八年七月

以關聯法則對於合作式粒子族群演算法訂立族群分類規則

研究生：高昆義　　指導教授：林昇甫 博士

國立交通大學 電機與控制工程學系

摘要

本論文主要目的在於訂立合作式粒子族群演算法的族群分類規則，以頻率項成長(FP-growth)找尋函數各維度間的關聯性，關聯性的強弱將決定族群分類規則是否成立。合理的族群分類規則可使演算法在效能上有良好的表現。因此對合作式粒子族群演算法來說，族群分類應為整個演化過程中相當重要的一部分。此外，使用最廣泛的粒子族群演算法及分割因子合作式粒子族群演算法，在使用上皆有其限制，兩者皆有適合解決的問題種類，但也有表現較差的部分，並無法以一種方法涵蓋所有最佳化的問題。本論文因族群分類規則具有彈性，且可根據維度關聯性訂立族群分類規則，可視問題不同做調整，在使用上可適用於各種問題類型。

關鍵字：頻率項成長、合作式粒子族群演算法、粒子族群演算法。

Using Association Rules to Set the Classified Rules of Swarms for Cooperative Particle Swarm Optimization

Student: Kun-Yi Kao **Advisor: Dr. Sheng-Fuu Lin**

Department of Electrical and Control Engineering

National Chiao-Tung University

Abstract

The purpose of this thesis is to define the group-based evolution classification rules for cooperative particle swarm optimization (CPSO). FP-growth is adopted in this thesis to find the connectivity between each dimension, in which the strength of the connectivity determines whether the classification rules apply or not. Generally, reasonable classification rules cause the algorithm work effectively. Hence, for the algorithm, the group classification plays a very important role in the evolutionary process. Besides, there are both restrictions between the most widely-used particle swarm optimization (PSO) and the CPSO. These two algorithms both work well under certain optimization tasks, but weak under others. None of these algorithms can cope with all the optimization tasks. In this thesis, the group classification rules are more flexible. The classification rules defined in this thesis are determined according to the connectivity between each dimension, and they are adjustable under different tasks, which greatly enhance the applicability of proposed mining based CPSO.

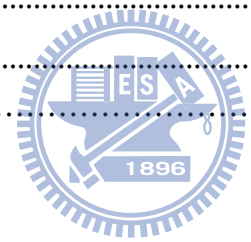
Keywords：FP-growth, CPSO, PSO.

誌謝

首先誠摯感謝指導教授林昇甫博士兩年來的教導，老師的細心指導及糾正學生課業上的問題，讓學生在研究所的路上不至於迷網困惑。同時感謝本實驗室學長永吉、逸章、士哲、晉嘉、國育、弦澤、啟耀、俊偉，能適時給予幫助及建議；以及一起努力的同學長安、道智、子航、煜維以及學弟們，大家一起同甘共苦，在彼此需要的時候伸出援手，一起成長，一起完成夢想，使論文能夠順利完成，除了學業上，生活上互相幫忙，協助，也因為有大家的參與，研究所這兩年充滿許多寶貴的回憶，能夠成為 806 實驗室的一份子，讓學生充滿著驕傲。

最後感謝我的父母，從小到大對我的鼓勵及支持，在我不順遂時，總是可以告訴我未來的方向在哪，當我生活上遇到挫折時，總是給我最大的幫助，完成研究所學業這份榮耀，將與他們共享。

目錄



圖目錄

表目錄

第一章 緒論

電腦的發明造就了生活上許多的便利，隨著日新月異的進步與發展，電腦成為幫助做龐大運算的重要工具，有別於人類有限的運算能力，快速及準確是電腦最大的優點也是如今為何會被應用如此廣泛的重要因素，經由電腦的幫助，許多需要大量運算的理論陸陸續續出現，其中包含本論文所提的最佳化理論。

人類研究的目的在於得到更好的生活，在某些部分來說，意即迫切希望可以得到比目前更好的最佳解，但遺憾的是，多數情況我們並無法輕易得到想要或是最好的結果，在過去人類只能利用一次又一次的嘗試，累積經驗告訴後人怎樣的結果是最好的，不同於以往完全仰賴於經驗法則，現今有許多的方法及理論皆可用於尋找近似的最佳解，像是粒子族群最佳化 (particle swarm optimization) 演算法以及基因演算法(genetic algorithm)即是兩種相當出色的最佳化演算法理論，基因演算法是以判斷優勝劣敗的機制，演化方式類似於基因遺傳過程，選擇較優良的基因進行交配(crossover)及突變(mutation)，較差的基因則為淘汰，若演化後基因可達到優良基因標準，則永遠遺傳。粒子族群最佳化演算法是模擬生物覓食行為所發展而來的一套最佳化理論，生物間透過互相聯繫，彼此告知食物所在位置的資訊，慢慢逼近食物的所在地，意即透過聯繫的方式，使各點皆可達到最佳解的狀態。

本章節中，將對本論文做整體架構介紹，1.1 節為介紹目前的相關研究；1.2 節為研究動機，告知為何要做此方面研究，此研究又可帶來什麼好處；1.3 節會介紹本論文的整體架構，讀者可以藉此了解整篇論文的編排架構，在之後論文的閱讀上，也可更容易進入各章節所提及的部分。

1.1 相關研究

本論文將以粒子族群最佳化演算法做為整篇論文主軸，本小節中將大略提及此演算法的起源與簡介。另外本論文另一重點為關聯性法則的尋找，透過資料探勘找出最佳化過程中的關聯項目。第一章中只會大略稍做介紹，詳細內容將於第二章及第三章所提及。

1.1.1 粒子族群演算法概述

粒子族群最佳化(particle swarm optimization, PSO)於 1995 年由 Kennedy 和 Eberhart 所提出[1, 2]，最佳化的過程為參考生物覓食模式，發展至今並不算是相當久的最佳化演算法，但如今卻使用廣泛，主要原因為粒子族群演算法彈性大、適用範圍廣且種類眾多，除了可用於函數最佳化外[3, 4]，也能使用於類神經網路的訓練[5]-[7]，近幾年許多學者紛紛提出改良粒子族群最佳化的方式，使整體演算法在使用上更為成熟。

圖 1.1 大略為各類型粒子族群演算法分類圖，大致上可分為四大類：

1. 原始粒子族群演算法：即為最初於 1995 年由 Kennedy 和 Eberhart 所提出的最佳化演算法[1, 2]，此時演算法具有收斂速度方向，全域最佳解(*gbest*)、個體最佳解(*pbest*)，以及擾動量等觀念，但並無族群分類的概念，較適合用於維度不大，且各維度間具有關連性的函數。

2. 改良粒子族群演算法：改良式的粒子族群最佳化有很多種型式，其中較重要的包含導入慣性比重(inertia weight)的概念[4]以及群體的想法[8]，慣性比重的導入可加快粒子收斂的速度，而群體的想法類似於族群，以群體的中心點位

置代替全域最佳解或個體最佳解，此可延伸至合作式粒子族群演算法的族群分類。

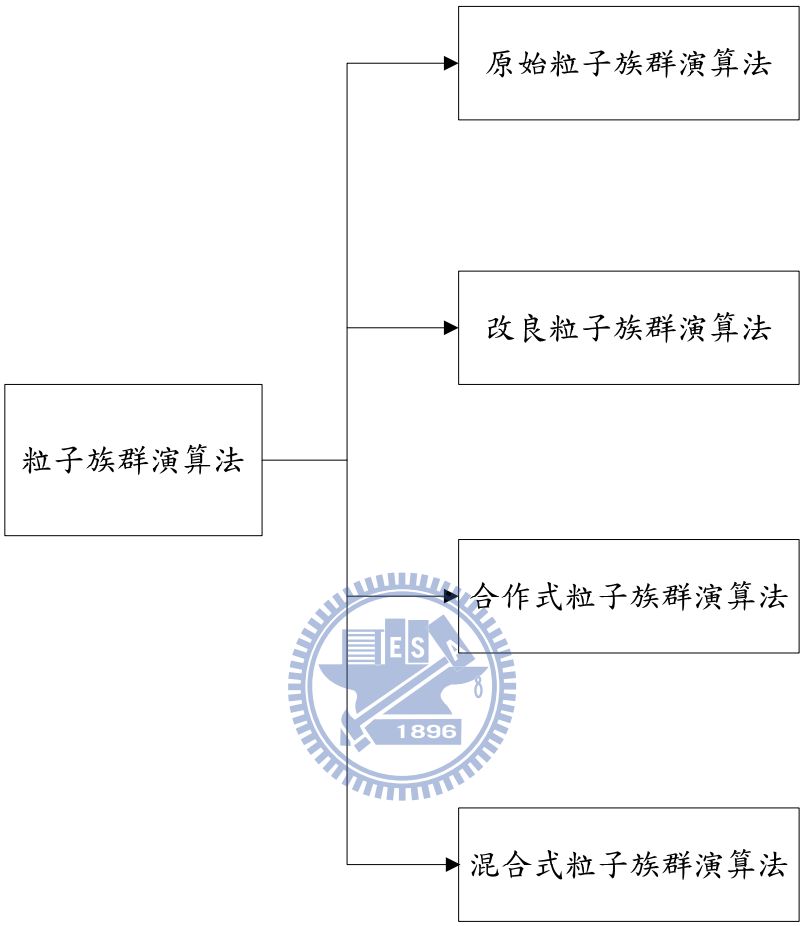


圖 1.1 粒子族群演算法分類圖。

3. 合作式粒子族群演算法：2004 年由 van den Bergh 與 Engelbrecht 提出族群分類的概念[9]，讓各族群所包含的維度不與其他族群內的維度一起演化，改善的原始粒子族群演算法在演化上的缺陷，此一想法為原始粒子族群演算法的重要改進，本論文也將以合作式粒子族群演算法中的其中一類做為研究主軸。

4. 混合式粒子族群演算法：為原始粒子族群演算法與合作式粒子族群演算法混用[9]-[11]，目的是希望保有兩者優點，原始粒子族群演算法可幫助演化過程

逃離非真的最佳解。

1.1.2 結合關聯法則與合作式粒子族群演算法之研究探討

合作式粒子族群演算法可算是一種較大改變的改良型式，導入的重要的族群分類概念，本論文將針對其中的一種演算法 CPSO-S_K 做深入的研究，因此方法彈性較大，較適合用於各類問題的解決，但最大問題為沒有族群分類的依據，以至於使上用並不廣泛。

對 CPSO-S_K 來說，族群分類規則為最重要的部分，將具有關聯性的維度分到同一族群，不具關聯性的維度全部分散，因此本論文將以資料探勘中的頻率項成長(FP-growth)[12, 13]，尋找各維度間的關聯法則，以做為族群分類的依據。

資料探勘可在大筆資料庫中找尋有要的資訊，包含了五種模式，分類（classification）、趨勢分析（trend analysis）、分群（clustering）、關聯（association）以及循序特徵（sequence pattern）。上述將於 2.3 節中詳細介紹此一部分。

關聯法則為資料探勘的其中一種模式，於 1993 年由 Agrawal、Imielinski 以及 Swami 所提出[14, 15]，主要用於搜尋項目間的關聯程度，透過最小支持值度及最小信任度為門檻，可找出使用者所需要的關聯資訊。

1.2 研究動機

粒子族群最佳化演算法與基因演算法[16]可說是目前使用上最廣泛的兩種最佳化理論，基因演算法因提出時間較早，因此最佳化的過程中較常以基因演算法做為工具，但近年粒子族群最佳化演算法發展迅速，許多改良後版本陸續被提

出，這些改良方式使原始粒子族群演算法更具有競爭力，其中改良的種類中以合作式粒子族群演算法最受青睞，因此粒子族群演算法也逐漸受到重視且廣泛的使用。

合作式粒子族群演算法較廣泛使用的演算法有兩種，分別為 CPSO-S 以及 CPSO-S_K，後者比前者更有彈性，但被使用的次數卻遠遠不如前者，主要原因是 CPSO-S_K 並無一套有效的分類依據，導致一般使用者在初期即無法進行使用，因此本論文希望透過關聯法則的搜尋，找出相關的維度項目，以做為 CPSO-S_K 族群分類的標準，透過這組訂立的規則，可幫助未來使用者在族群分類上有跡可循。

比較 PSO、CPSO-S 及 CPSO-S_K，前兩者在族群分類上皆有嚴格的規定，PSO 將所有維度分到同一族群裡，CPSO-S 則是把所有維度各自分到不同族群，但這兩種方法卻會受到使用對象的不同，在效能表現上差異大，各有不同的適當使用時機，並無法通用於所有最佳化的例子，在這一點上 CPSO-S_K 的確比這兩者來的有彈性，可適用於所有最佳化的對象，因為 CPSO-S_K 族群分類規則完全由使用者設定，族群分類情況只要依照不同題型做改變便可達到好的效果，但現今的文獻中並較好的方法來規範族群分類準則，因此訂立可靠且合理的族群分類規則可視為 CPSO-S_K 最重要的一部分。

1.3 整體架構

本篇論文在以下章節的安排，第二章將介紹各類型粒子族群演算法內容和演算方式，並將資料探勘做較詳細介紹，包含資料探勘模式、流程、關聯法則以及建構頻率項樹；第三章為本文核心，提供本篇論文的想法及架構；第四章為實驗數據，所有重要的資訊將詳列於此；第五章為本文的結論。

第二章 背景知識

本文中所提及相關知識，包含最佳化理論、資料探勘以及模糊類神經網路三大領域。在最佳化演算法部分，使用粒子族群最佳化(PSO)，以及合作試粒子族群最佳化(CPSO)，兩者皆有其重要性以及適當的使用時機；資料探勘的部分，利用頻率項目成長(FP-growth)尋找所需關聯性；而模糊類神經網路，本文使用 TSK 模糊類神經系統，此部分是用於函數模擬 (function approximation)，以下章節將詳細介紹各理論及其相關知識。2.1 節為原始粒子族群最佳化介紹，以及幾種改良方式；2.2 節為合作式粒子族群最佳化的介紹；2.3 節為資料探勘的部分，介紹搜尋關聯法則的工具；2.4 節介紹 TSK 模糊類神經系統，本論文中函數模擬實驗需以此模糊類神經網路做為架構，在此節中將詳細介紹。

2.1 粒子族群最佳化

粒子族群最佳化(particle swarm optimization, PSO)於 1995 年由 Kennedy 和 Eberhart 所提出[1, 2]，可成功用於大範圍連續函數的最佳化演算法[17]-[20]，以模擬動物覓食所產生行為模式的演算法，如同動物間彼此聯繫聯絡食物所在位置，每個粒子會受到本身以及其他粒子所影響，不同粒子擁有各自的速度及方向，最終往最佳解處靠近。

2.1.1 原始粒子族群最佳化

粒子族群最佳化(PSO)中粒子的移動方式如圖 2.1 所示，假設系統為二維系統，圖中目標為系統中最佳解，A、B 與 C 為系統中三個粒子，A_n 表示 A 粒子

在每一代演化後所在的位置，每一筆粒子資料中含有兩個維度(x, y)，每個維度受本身位置、其他維度位置、全域最佳解(*gbest*)、個體最佳解(*pbest*)以及擾動量所影響。

粒子族群最佳化演化公式如公式 2.1、2.2、2.3 及公式 2.4，其中，V 為演化速度，X 為粒子位置， i 為各維度所含粒子數，d 為各不同維度，c_1 與 c_2 為系統加速係數(acceleration coefficients)，此加速係數用來控制每一代演化中粒子的移動量，某些時候不同的值會影響整個系統的效能[8]；*rand*1 與 *rand*2 為兩個隨機變數，在演化中產生擾動，目的為避免系統落於局部最佳解(local minima)。

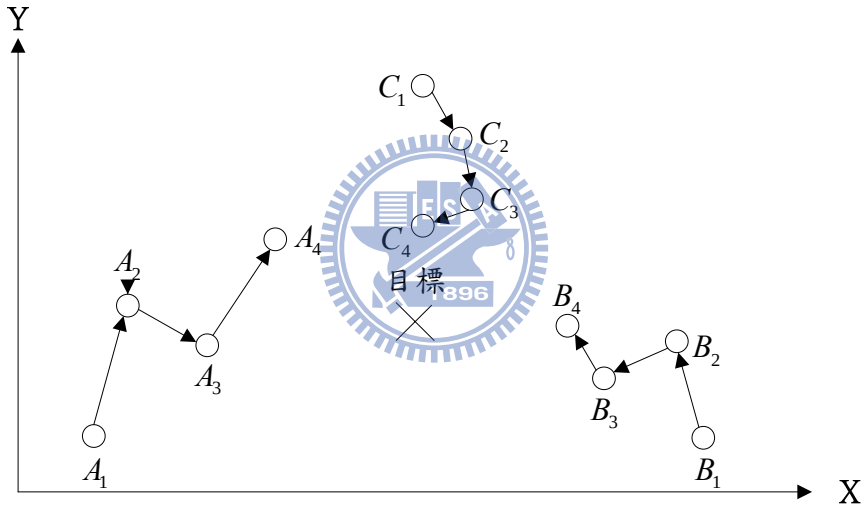


圖2.1 粒子族群最佳化移動示意圖。

pbest 為每個粒子在演化中得到的最佳解， *gbest* 為所有粒子在演化中最佳解， *pbest* 與 *gbest* 關係如圖 2.2 所示。

$$V_i^d(t+1) = V_i^d(t) + S_1 + S_2, \tag{2.1}$$

$$S_1 = c_1 \cdot rand1_i^d \cdot [pbest_i^d(t) - X_i^d(t)], \tag{2.2}$$

$$S_2 = c_2 \cdot rand2_i^d \cdot [gbest^d(t) - X_i^d(t)], \tag{2.3}$$

$$X_i^d(t+1) = X_i^d(t) + V_i^d(t+1). \tag{2.4}$$

假設空間中有 a～e 五個粒子，演化代數以 3 為例，a_n 代表演化第 n 代時粒子位置，*pbest* 表示為這三代演化中各粒子最好的解，每個粒子有一組最好的 *pbest*，意即 *pbest* 的數目與初始設定的粒子數相同，如圖 2.2 中 A_3、B_2、C_3、D_1 以及 E_2 等紅色粒子所式；*gbest* 為所有粒子在這三代演化中最好的解，包含所有例子經過演化所產生的全部資料，如本例中粒子個數 5 個，演化代數為 3，因此總共會產生 15 筆比較資料，假設 C_3 為這 15 筆資料中最好的一組解，即離設定目標最近的資料，則 C_3 為本例中的 *gbest*，如圖 2.2 藍色框所示，簡單來說，在本例中，*pbest* 經過演化後產生的資料總共有五筆，而 *gbest* 僅有一組。粒子族群最佳化便是希望透過演化將 *gbest* 落於目標上。

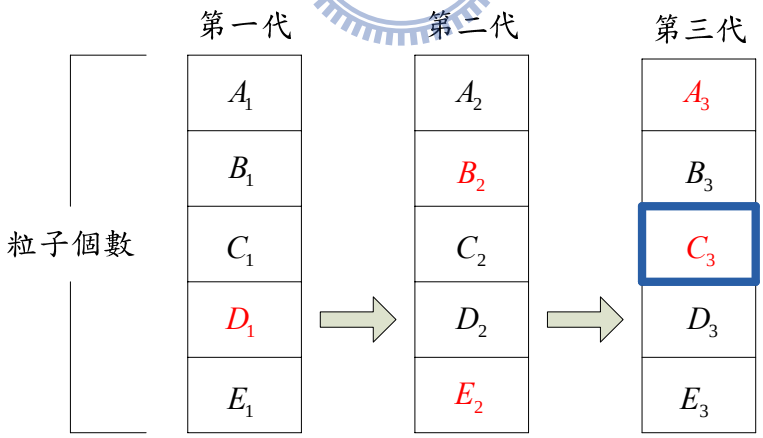


圖 2.2　*pbest* 與 *gbest* 關係圖例。

進行演化中，通常會設定一組固定參數[$-v_{max}$，v_{max}]，用來對演化速度 V_i 產生限制，以避免粒子有過大的演化速度而跑出適當的搜尋空間。此組參數並未意味著粒子的位置 X_i 必須限制落於這段範圍內，僅是用來控制粒子在演化中可移

動的最大量。

粒子族群演算法標準流程可見圖 2.3，k 為演化代數。

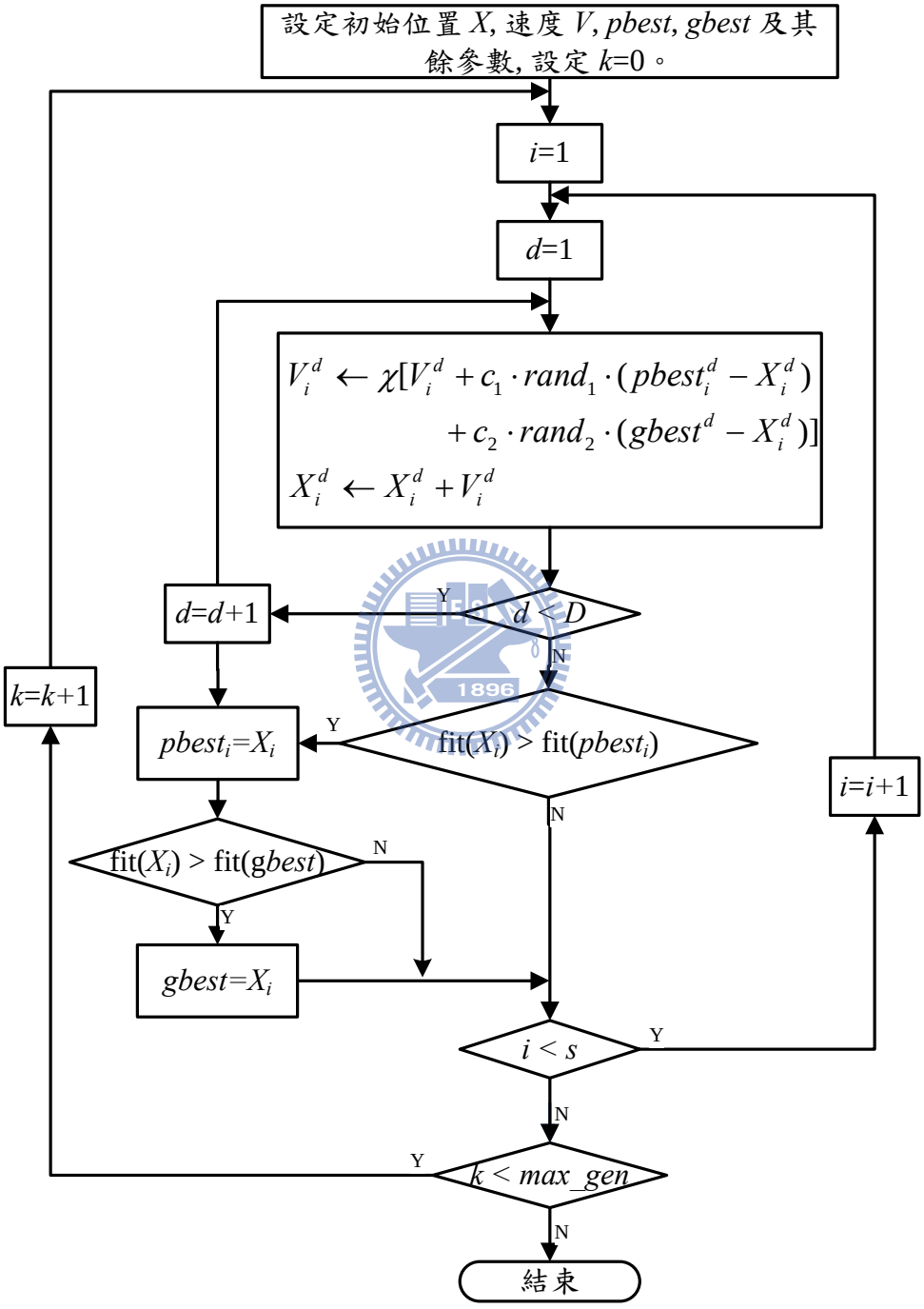


圖 2.3 粒子族群演算法標準流程。

近年由 Clerc 提出加入壓縮因子 χ(constriction factor)概念取代 $v_{\max}$ 限制，用來擔保演算法的收斂性[21]-[23]，如公式 2.5、2.6、2.7 及公式 2.8 所示。

$$V_i^d(t+1) := \chi\{wV_i^d(t) + S_1 + S_2\}, \tag{2.5}$$

$$S_1 = c_1 \cdot rand1_i^d \cdot [pbest_i^d(t) - X_i^d(t)], \tag{2.6}$$

$$S_2 = c_2 \cdot rand2_i^d \cdot [gbest^d(t) - X_i^d(t)], \tag{2.7}$$

$$\chi = \frac{2}{\left|2 - \phi - \sqrt{\phi^2 - 4\phi}\right|}$$ ，其中 $\phi = c_1 + c_2$ 且 $\phi > 4$ 。 (2.8)

以壓縮因子 χ 取代 $v_{\max}$ 限制的改變，有時未必會在效能上得到較好的改進，Shi 和 Eberhart 提出結合兩種限制因素可達到最快的收斂效能[24]。

2.1.2 改良粒子族群最佳化

原始粒子族群最佳化演算法有許多改良的方式。1998 年由 Shi 和 Eberhart 提出導入慣性比重(inertia weight)的概念[4]，即在原始粒子族群最佳化的演算過程中，$V_i(t)$ 被 $wV_i(t)$ 所取代，如公式 2.9、2.10 及公式 2.11 所示,通常會設定此比重的值從第一次演化，到最後一次演化過程中，由 1 到近似 0 做線性遞減，加入慣性比重的結果可加速演算法的收斂速度。

$$V_i^d(t+1) = wV_i^d(t) + S_1 + S_2, \tag{2.9}$$

$$S_1 = c_1 \cdot rand1_i^d \cdot [pbest_i^d(t) - X_i^d(t)], \tag{2.10}$$

$$S_2 = c_2 \cdot rand2_i^d \cdot [gbest^d(t) - X_i^d(t)]. \tag{2.11}$$

1999 年由 Angeline 提出一套改良原始粒子族群最佳化效能的方法[10]，在整體演算法上與原始演算法有較大的不同，以一種粒子間適者生存，不適者取代的

方式進行改進，粒子間以粒子存在的適當程度進行比較，將全部的粒子劃分為相等兩群，較不適當的一群粒子，由表現較好的另一半粒子取代，取代資料包括粒子位置以及演化速度，但卻不包括各粒子本身的 *pbest* ，以這種取代的方式在進行剩餘的演化過程，此技巧在某些例子可確實改善演算法的效能。

還有一種改良方式是以劃分群體的概念[25]，將原本全部粒子在同一個群體中的情況，劃分為數個群體，劃分後每個群體內各粒子保有各自的 *pbest* ，g_j 為劃分後各群體的中心點，在演算法修改上有三種型式，型式一如公式 2.12、2.13 及公式 2.14；型式二如公式 2.15、2.16 及公式 2.17；型式三如公式 2.18、2.19 及公式 2.20。

型式一

$$V_i^d(t+1) := wV_i^d(t) + S_1 + S_2, \tag{2.12}$$

$$S_1 = c_1 \cdot rand1_i^d \cdot [g_i^d(t) - X_i^d(t)], \tag{2.13}$$

$$S_2 = c_2 \cdot rand2_i^d \cdot [gbest^d(t) - X_i^d(t)]. \tag{2.14}$$

型式二

$$V_i^d(t+1) := wV_i^d(t) + S_1 + S_2, \tag{2.15}$$

$$S_1 = c_1 \cdot rand1_i^d \cdot [pbest_i^d(t) - X_i^d(t)], \tag{2.16}$$

$$S_2 = c_2 \cdot rand2_i^d \cdot [g_j^d(t) - X_i^d(t)]. \tag{2.17}$$

型式三

$$V_i^d(t+1) := wV_i^d(t) + S_1 + S_2, \tag{2.18}$$

$$S_1 = c_1 \cdot rand1_i^d \cdot [g_i^d(t) - X_i^d(t)], \tag{2.19}$$

$$S_2 = c_2 \cdot rand2_i^d \cdot [g_j^d(t) - X_i^d(t)]. \tag{2.20}$$

此種改善方式可降低落於局部最佳解的傾向，但在執行速度上因為比原始演算法複雜度來得高，所以相對有需時間也較久。而在效能表現上，在文獻[25]中指出，只有型式一表現上較原始粒子族最佳化演算法來的出色。

粒子族群演算法是一種相當簡單且又有效的最佳化工具，即使是在較大範圍中做最佳化的動作，此演算法也可達到很高的效能，但卻存在一項較大的缺點，容易造成“兩步向前，一步後退”(two steps forward, one step back)的情況，尤其在維度大的例子中情況更為明顯，此種情形可見圖 2.1，以 B_2 和 B_3 兩點為例，假設 $B_2=[x_2, y_2]$ 以及 $B_3=[x_3, y_3]$，若以 x 軸來看這兩點，x_3 比 x_2 更接近設定的目標，因此第三代的演化有比第二代來得好；反觀比較兩點 y 座標，y_3 比 y_2 更遠離設定的目標，嚴格來說這一代的演化對 y 座標是較差的，但在此處 x 的效果比 y 的效果來得顯著，更符合期望的目標，這樣的結果是可被接受的，至於改良方式，將於下章節合作式粒子族群演算法提及。

2.2 合作式粒子族群最佳化

合作式粒子族群[9]最佳化(CPSO)在演算法上並無太大差異，僅在族群的分類上做些修正，加入了分解族群的概念，類似於 GA 中合作式的想法[26],[27]，在種類上可大致分為三類，CPSO-S 、CPSO-S_K ，以及 CPSO-H_K 。以 PSO 在解決問題時，雖有整體優化能力強、收斂速度快，以及算法簡單…等優點，但卻無法避免“two steps forward, one step back”的缺點，對於解決維度大的問題時，一次演化需要更新的維度太多，表現並不理想，而 CPSO 正是用來解決這些缺點。

現今提出的 CPSO 演算法大致上科分為三類，分別為 CPSO-S、CPSO-S_K ，以及 CPSO-H_K ，前兩類的差異只在於族群的選取上不同，而 CPSO-H_K 在演算法上做了一點修改，簡單可視為 CPSO 與 PSO 的混合型式。

CPSO-S：此演算法在族群的分類上與 PSO 可說是相反的型式，傳統粒子族群最佳化演算法會將所有維度的資料放於同一的族群裡，演算過程中各維度的資料一起更新，如圖 2.4 所示，*A*~*E* 為五個維度，*A*(*n*)代表第一個維度的第 *n* 顆粒子，PSO 會將所有維度組成的一組資料一起演化，各維度間無優先順序之分，速度快，但維度大時卻不容易找到最佳解；CPSO-S 正好相反，將每個維度的資料全部分開，有幾個維度便會產生幾組族群，演算方式上，一次只演化一個維度，其餘維度以目前為止最佳的一組資料提供運算，一個維度演化完在繼續演化下一個維度，如圖 2.5 所示，CPSO-S 會先計算第一個維度的第一顆粒子，其餘維度以當時 *gbest* 的二~五維資料提供運算，第一個維度運算結束後，在進行第二個維度，每次運算只改變一筆資料，在計算量上，明顯比 PSO 多上許多，但因為一次只演化一筆資料，收斂的效果表現上會比 PSO 來的好。

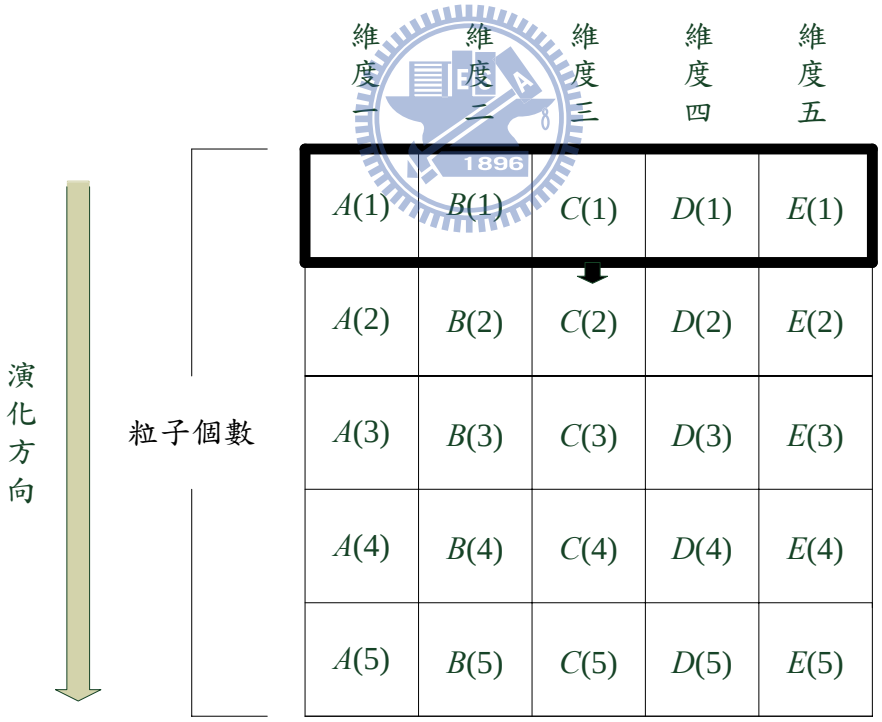


圖 2.4　PSO 演化示意圖。

CPSO-S 最大的優點，就是可以解決 PSO 無法克服的重大缺陷，“two steps forward, one step back”，因為 PSO 一次的演化為所有維度的一組資料，在演化成

功的情形下，雖然對整體來說是一組較好的解，但若將各維度分開來看，有些維度的新位置並未必會比前一代來得好；反觀 CPSO-S，將各維度的資料全部分開，一次只演化一個維度，可以確保每個維度在新的一代一定比過去的任何一代優良。

與傳統 PSO 比較，CPSO-S 雖可解決 PSO 的缺點，但除了演化過程中計算量較大以外，若面對的問題中變數之間存在關聯性，有時效能表現上未必會比 PSO 來得優良，因為 CPSO-S 初始已將所有變數設定為相互獨立，此動作對於具有關連性的部分會產生較差影響，此缺點可透過另一類 CPSO 來解決，即為 CPSO-S_K。

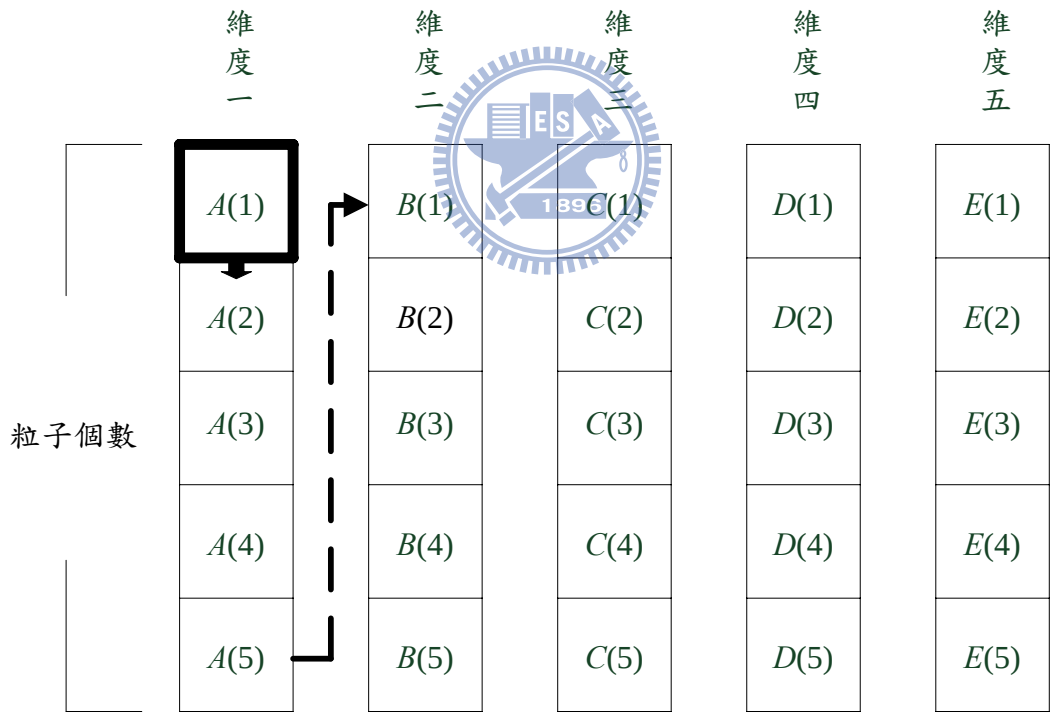


圖 2.5　CPSO-S 演化示意圖。

CPSO-S_K：CPSO-S 解決了“two steps forward, one step back”，但面對有關連性的不同變數卻會造成不好的影響，造成這種現在的主因在於初始的畫分族群上，CPSO-S_K 與 CPSO-S 最大的不同便是在族群分類上做改進，不同於 CPSO-S

一開始便把每個維度獨立開來，CPSO-S_K 給系統較大的彈性來處理維度分族群的問題。

假設系統具有 n 個維度，CPSO-S_K 在初始設定上，會將 n 個維度分成 k 個族群，在此處 k 稱為分割因子(split factor)，分割後的 k 個族群，擁有各自不同的維度，有些族群可能會是一個維度，有些族群會是 c 個維度($c < n$)，透過不等分變數數目的做法，將具有關聯性的變數放於同一個族群，其餘互相獨立的變數，則全部分別放於不同的族群，此種分類法，可避免將相關變數獨立開來造成的不良效果，顯而易見，當分割因子 $k = n$ 時，每個維度將會全部獨立，因此 CPSO-S 其實為 CPSO-S_K 的一種特例。

若可做出適當的族群分類，CPSO-S_K 將可有效改善 CPSO-S 面對相關變數造成的不良影響，但在執行上，卻不如想像中容易，實際應用上，大部分的例子並無法明顯看出哪些變數是具有相關性，因此在分族群上造成極大的困擾，現今在族群分類上並無一套有效的規則可循，使用者只能依靠嘗試錯誤法(trial and error)來判斷族群分類的好壞，因此即使找到比 PSO 或 CPSO-S 好的解，也未必能決定此時找到最好的族群分類結果，相同的，此時所得到的解，也未必會是系統的最佳解，本論文即在此方法上會出修正。

CPSO-H_K：混合式合作式族群最佳化(hybrid CPSOs)簡稱為 CPSO-H_K，此種方法將 CPSO-S_K 與 PSO 在使用上作結合，目的是想要得到這兩種方法各自較好的特性。當 CPSO-S_K 有陷入局部最佳解的傾向時，利用 PSO 可幫助演算法逃離此時假的最佳解。

但演化過程中，並不容易看出 CPSO-S_K 是否具有陷入局部最佳解的傾向，因此使用上 CPSO-S_K 與 PSO 會以接連出現的方式表現，當 PSO 做完一代演化，接下後面的是 CPSO-S_K 也做一代演化，彼此之間循環的出現，透過這種安排可

使演化的收斂方向較為正確。

2.3 資料探勘

資料探勘(data mining)是用來擷取所需資訊一種有效的方法[28]，這些資訊藏於大量的資料庫中，透過資料探勘可找出使用者所需要且有價值的潛藏規則。資料探勘可視為資料庫知識發掘(knowledge discovery in database)的一部分，其中知識發掘的過程包括資料選取、前置處理、轉換、資料探勘及解釋與評估。若資料庫本身具有足夠的完整性，則所得到的結果將會是重要的知識。資料探勘運用利用分類、群集分析、關連性、序列分析、機器自我學習及其它統計方法，從資料庫中找出隱藏、未知且有用的資訊。

資料探勘可分成五種模式，分類（classification）、趨勢分析（trend analysis）、分群（clustering）、關聯（association）及循序特徵（sequence pattern）。

1. 分類：根據一些變數的數值做計算，再依照結果作分類。例如可將全部垃圾資料分為可燃不可燃，以這些完成分類的資料，研究可燃垃圾與不可燃垃圾的性質和特徵，這就是從歷史資料中所得到的知識，再利用這些擷取到的知識，去對未來的資料做預測。
2. 趨勢分析：以現今所得到的資訊，來預測未來的數值。例如工廠利用去年的訂單和員工請假頻率，來衡量今天需要多少雇請多少員工，或是將去年與今年員工數做比較，找出其中的差異性。
3. 分群：是將資料分組，其目的是找出各組之間的差異及同組中成員的相似性，使群組內差異小，群組外差異大。例如百貨公司將商品以使用客戶年齡層、性別、消費能力，分不同樓層區塊擺放，這種分群可使百貨公司更易於統計客戶的需求，也增加客戶的便利性。
4. 關聯：是要在同一個交易中找出隱含其它項目之存在的項目，找出某一事件

或是資料中會同時出現的部分。例如客人買了餅乾和糖果，則在這情況下，多數客人喜歡也買汽水，透過這種關聯性，商家將糖果、餅乾及汽水，放置在同一個區塊，且只針對餅乾和糖果做特價，則可達到將三種商品一起賣出的效果，達到銷售目的。

5. 循序特徵：與關聯關係密切，最大的不同是循序特徵中相關項目是以時間加以區分。例如客人買了餅乾和糖果，在這種情況下，兩天後客人會再來買汽水的機率。

資料探勘流程圖如圖 2.6 所式，各步驟解釋如下：

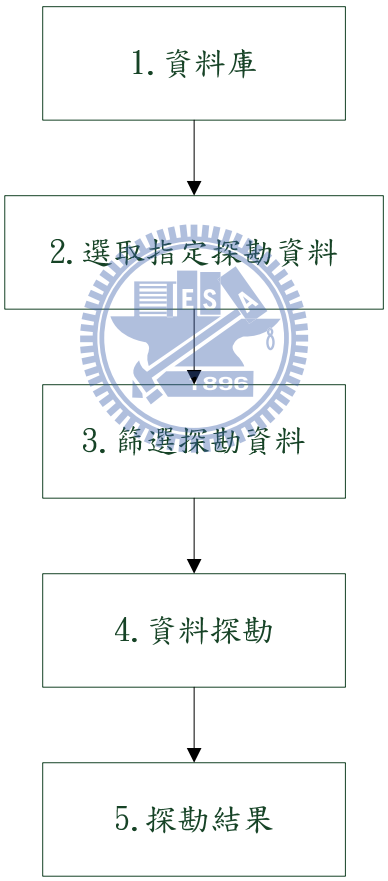


圖 2.6 資料探勘流程圖。

1. 資料庫：要執行資料探勘第一步驟就是須有一個完整且大量的資料庫，這個資料庫中未必所有項目都是使用者所需要，也未必每一筆資料都無瑕疵，因此在使用前應先做前處理。
2. 選取指定探勘資料：資料庫並未必僅依照使用者當下需求所建立，因此許多

種類的資訊是不必要的，探勘的資料可能不會是整個資料庫的全部，需要的僅是其中一個資料庫的一部份，或是數個資料庫裡的一部分資料。

3. 篩選探勘資料：篩選出較好的資料，刪除一些不必要的內容，例如資料重複、資料錯誤、或資料無效...等，目的使探勘過程減少計算量以及確保資料的完整性。
4. 資料探勘：種類如上述資料探勘五種模式，分類、趨勢分析、分群、關聯及循序特徵等。
5. 探勘結果：藉由資料探勘，找出隱藏的特徵或關聯性。

資料探勘憑著可預測未來的趨勢，可對使用者提出有用的建議，但唯有好的資料庫及前置處理，才是資料探勘最重要的一步。

2.3.1 發掘關聯式規則

發掘關聯式規則（mining association rule）最早是 1993 年由 Agrawal 等人所提出[14, 15]，主要用來找出項目間的關聯性，例如用於賣場商品販售上，可找出各商品間販賣的關係，像是買牙刷的人有多少機率會買牙膏、買洗髮精的人有少機率會買潤髮乳等。文獻[29]-[32]提及關聯法則於市場應用的重要性。

支持度(support)和信任度(confidence)是發掘關聯式規則最重要的兩個指標，若以賣場為例，兩者表示如公式 2.21、公式 2.22，支持度(X)表示購買商品 X 的機率，而信任度(X→Y)代表當顧客購買商品 X 時，同時會購買商品 Y 的機率，由這兩種指標可以了解顧客的購買習慣，可成為賣場作商品做促銷時所需重要資訊。

使用上，首先須設定最小支持度(minimum support)以及最小信任度門檻(minimum confidence)，利用這兩個指標為使用者做篩選。發掘關聯式規則有兩

個步驟：步驟一，找出所有頻繁項目。步驟二，由步驟一所得到的頻繁項目，產生高關聯性的規則。

支持度(X)=(包含 X 的銷售紀錄) ／ (資料庫中總交易紀錄). (2.21)

信任度(X→Y)=(包含 X 且包含 Y 的銷售紀錄) ／ (包含 X 的銷售紀錄) (2.22)

如圖 2.7 所示，為一簡單範例，交易項目包含{A, B, C, D, E, F}六項，而超過最小支持度的項目總共有{A}、{D}、{E}、{F}、{A, D}、{A, E}、{D, E}及{A, D, E}等 7 項。

交易編號	項目
100	A、D、E
200	A、C、D、E
300	A、D、E、F
400	B、F

所有項目；A、B、C、D、E、F
最小支持度；50%
最小信心度；60%

頻繁項目	支持度
{A}	75%
{D}	75%
{E}	75%
{F}	50%
{A, D}	75%
{A, E}	75%
{D, E}	75%
{A, D, E}	75%

圖 2.7 發掘關聯式規則簡例。

經由公式 2.22，可算出各種規則組合的信任度，信任度可視為關聯的強度，

如信任度(A→D)=支持度({A}∪{D}) ／ 支持度({A})=100%，以商品為例，表示購買商品 A 的情況下同時會購買商品 D 的機率是 100%，符合使用者一開始所設定的 60%；又如信任度(A→F)=支持度({A}∪{F}) ／ 支持度({A})=33%，很明顯這個關聯式規則並不是使用者所需要的。

2.3.2 Apriori 演算法

Apriori 演算法是使用候選項目集合來找出頻繁項目集合的一種方法，也是一套最有影響力的探勘布林關聯規則頻繁項目集合的演算法，利用逐層搜尋的疊代方法，由第 k 個項目集合，往下推出第(k+1)個項目集合，再以第(k+1)個項目集合推第(k+2)個項目集合，以此類推，直到找不到第(k+n)個項目集合為止。使用上一開始由資料庫找出只有一個項目的集合，視為候選 1-項目集合，若各項目集合有滿足最小支持度，則視為頻繁 1-項目集合，不滿足的項目集合則刪除，再由頻繁 1-項目集合內的各項目互相結合，得出候選 2-項目集合，此時每個集合包含兩個項目，若候選 2-項目集合滿足最小支持度，則視為頻繁 2-項目集合，不滿足項目集合則刪除，以此類推，重覆產生與刪除的動作，直到不再產生候選項目集合。

圖 2.8 為 Apriori 演算法簡例，資料庫中有四筆資料，交易項目包含{A, B, C, D, E, F}六項，首先會找出所有單項集合，得到候選 1-項目集合，{B}與{C}因為支持度未大於最小支持度，所以頻繁 1-項目集合將刪除{B}與{C}兩項而僅剩四組集合，再由這四組集合產生候選 2-項目集合，共六組，此時{A, F}、{D, F}、{E, F}三項不滿足最小支持度，頻繁 2-項目集合刪除不滿足的三項集合而僅存三項，以此類推，最後可得到頻繁 3-項目集合{A, D, E}，且支持度為 75%。其中實線箭頭代表產生項目集合，虛線箭頭代表刪除項目集合。

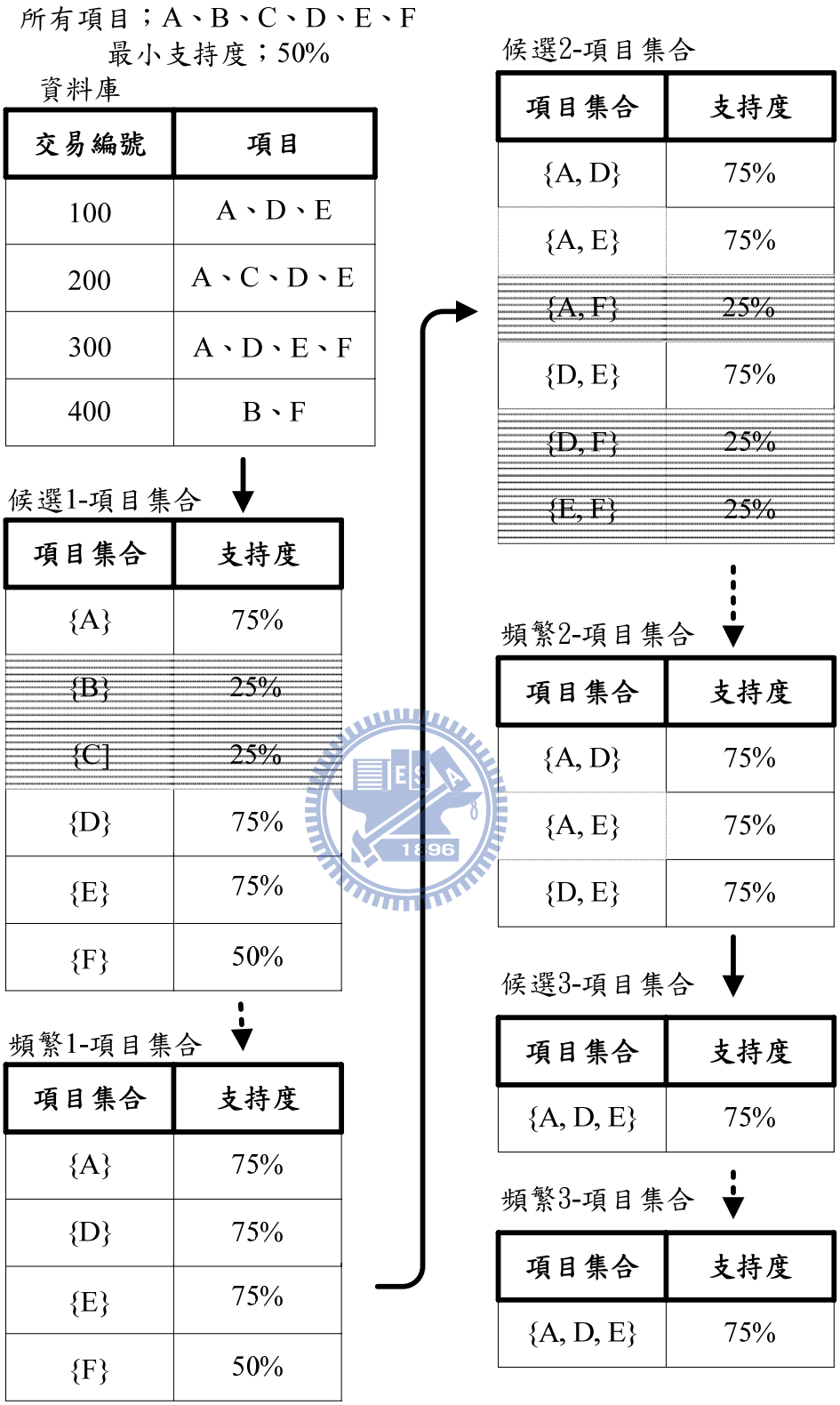


圖 2.8 Apriori 演算法簡例。

Apriori 演算法雖然有效且計算簡單，但卻有幾個較大的缺點：

1. 候選人選取不易：有時面對的資料庫內含資料量相當龐大，要完整找出所有的候選項目存在著較高的困難性。

2. 冗長乏味的支持度運算：每一筆候選項目都必須計算一次支持度，雖然計算容易，但卻容易因為冗長且乏味而產生錯誤。

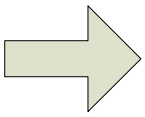
2.3.3 頻率項目成長演算法

有別於 Apriori 演算法，以逐層搜尋的方式找出關聯規則，頻率項目成長(FP-growth)並無延用此一方式，整體核心為頻率項樹(FP-tree)的建構，這也是效能可以優於 Apriori 演算法的主要原因。FP-Growth 在一開始做一次全面的搜尋，先去除小於做小支持度的項目，再將各筆資料內的項目依照支持度的高低進行排序，依此資料建立 FP-tree。

最小支持度：60%

交易編號	交易項目
100	{A, C, D, F, G, I, M, P}
200	{A, B, C, F, I, M, O}
300	{B, F, H, J, O}
400	{A, B, C, K, S, P}
500	{A, C, E, F, L, M, N, P}

排序及指定項目



交易編號	排序後指定項目
100	{A, F, C, M, P}
200	{A, F, C, B, M}
300	{F, B}
400	{A, C, B, P}
500	{A, F, C, M, P}

圖 2.9 資料指定並排序。

如圖 2.9，資料庫中有五筆交易資料，最小支持度為 60%，所以當資料內項目出現少於三次則會在第一次挑選中被刪除，挑選後的項目再依照出現的次數作排序，即為圖 2.9 中右邊的表格，上步驟結束後，建立標頭索引表(header table)，索引表的建立僅依存在項目出現的次數加以排序，目的是在找出 FP-tree 中每個項目的第一個節點，由圖 2.9 右表及標頭索引表，建立 FP-tree，且每個相同項目以虛線連接，如圖 2.10 所示。

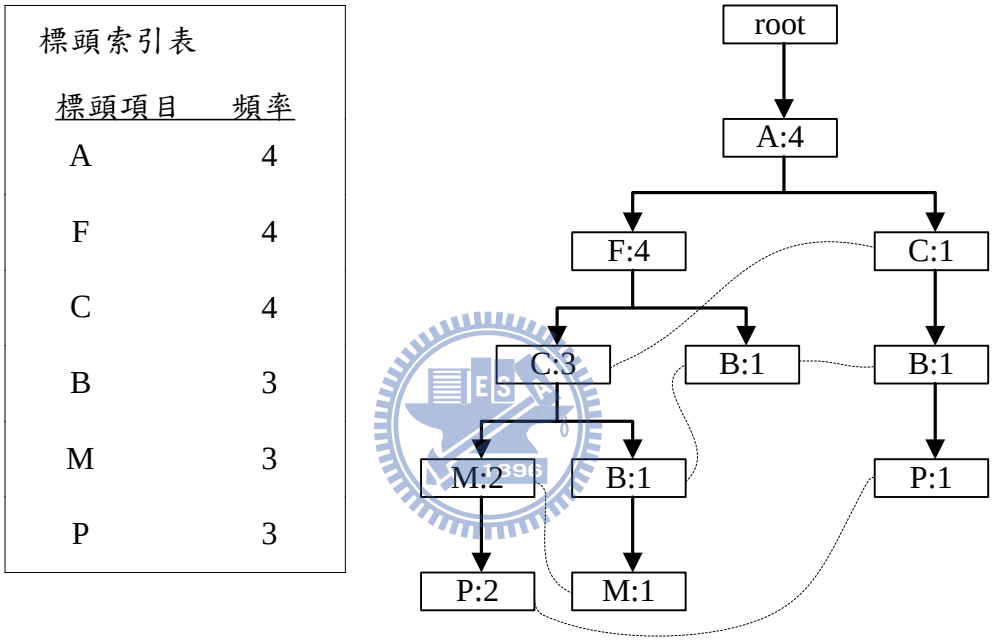


圖 2.10 建構 FP-tree。

建立出 FP-tree 後，可藉由此得到想要的關聯資訊，例如想知道與項目 p 相關的關聯規則，可得到兩條路徑，路徑一與出現次數為 AFCM：2，另一條為 ACB：1，由此兩條路徑，得知與 P 關聯的各項目出現次數分別為{A:2, F:2, C:3, M:2, B:1}，最小支持度為 60%，及最少須出現 3 次，可得出與 p 相關的關聯規則為 CP:3。

相較於 Apriori 演算法，FP-growth 有幾項優點：

1. 不須產生候選項目，因此不需做候選項目測試。
2. 根據頻繁項目的提供來分解探勘目標以及資料庫。
3. 可縮小探勘時的資料庫容量。
4. 不須重複搜尋整個資料庫。

2.4 TSK 模糊類神經網路

近年來模糊類神經網路應用廣泛，常被用在處理自動控制及訊號處理系統、人類邏輯推論、圖形辨識、語音辨識、診斷程序、智慧型系統等，在各領域有十分重要的貢獻，有別於非 0 即 1 的概念，模糊系統提供更具有彈性的學習模式，允許每個元素可同時屬於多個不同集合，僅在程度上做區分，此一概念提供較好推廣度以及錯誤容忍度，與人類思考相當接近，更適合用於解決現實世界中非線性的問題，在效果上也得到相當顯著的成就。表 2.1 提供傳統與模糊兩種不同概念的差異。

表 2.1 傳統與模糊概念比較。

	傳統概念	模糊概念
數值範圍	一固定值	涵蓋區間
數值型態	二值化	多值
數值規則	非是即否	程度差異

TSK(Takagi-Sugeno-Kang)模糊類神經網路由 Takagi 與 Sugeno 於 1985 年提出[33, 34]，利用模糊化切割(fuzzy partition)將輸入的空間是為線性切割的延伸，

其推論法則如公式 2.23，y^{ℓ} 代表第 ℓ 條模糊規則的輸出變數，a_i^{ℓ} 為實數參數，其中 i=0, 1, 2, …, m 。

$$R^{\ell}: If \quad x_1 \quad is \quad A_1^{\ell} \quad and \quad x_2 \quad is \quad A_2^{\ell}, ..., and \quad x_m \quad is \quad A_m^{\ell},$$
$$then \quad y^{\ell} = a_0^{\ell} + a_1^{\ell} x_1 + a_2^{\ell} x_2 + ... + a_m^{\ell} x_m. \tag{2.23}$$

TSK 模糊類神經網路提供多個輸入及一個輸出，輸入的個數視使用者需求調整，整體架構可分為五層。第一層輸入層(input nodes)，提供訊號輸入，但並不作調整，訊號將直接送至第二層；第二層為歸屬函數層(membership function node)，將第一層的輸出當作輸入，對此訊號進行模糊化的處理；第三層為規則層(rule node)，每個神經元的輸入個數與第一層神經元各數相同；第四層為推論層(consequent node)，推論由第三層所得到的資訊；第五層為輸出層(output node)，目的用於解模糊化。整體架構如圖 2.11，各層詳細說明如下，x 為第一層輸入，$u^{(n)}$ 為第 n 層輸出。

1. 第一層：第一層為輸入層，並不做任何運算，目的只是要將輸入訊號帶至下一層，$u_i^{(n)}$ 為第 n 層第 i 個輸入的輸出，計算公式如下：

$$u_i^{(1)} = x_i. \tag{2.24}$$

2. 第二層：此層為歸屬函數層，目的在於將第一層輸出的訊號作模糊化的處理，使訊號不再是單一數值，每個訊號的轉換成一個模糊參數，使訊號屬於多個模糊集合(fuzzy set)，僅在程度上有所不同，此處使用的歸屬函數為高斯歸屬函數(Gaussian membership function)，計算公式如下，其中 m_{ij} 和 σ_{ij} 分別是平均值(mean)和標準差(deviation)，對應到第 i個輸入的第 j個歸屬函數。

$$u_{ij}^{(2)} = exp\left(-\frac{[u_i^{(1)} - m_{ij}]^2}{\sigma_{ij}^2}\right). \tag{2.25}$$

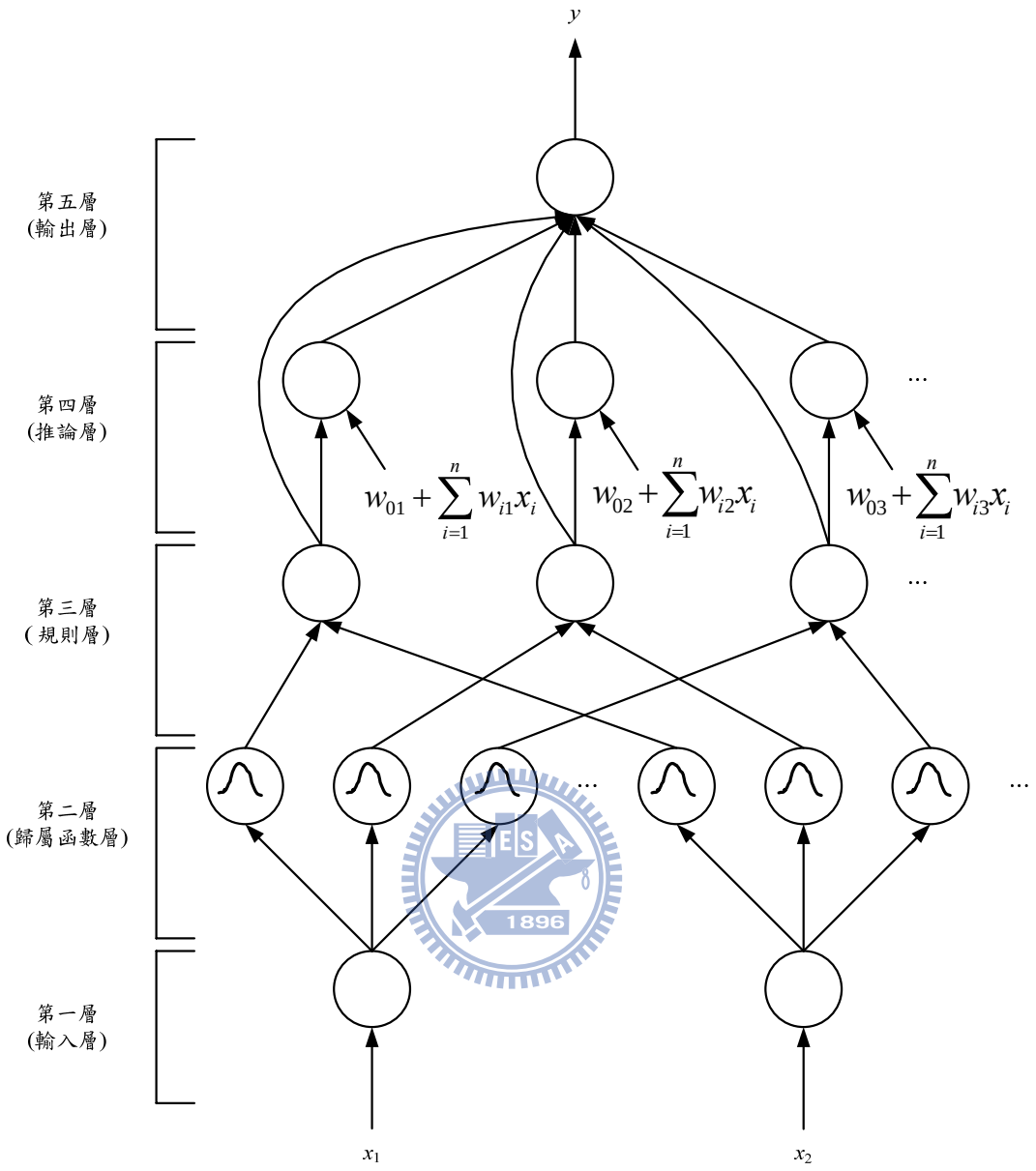


圖 2.11 TSK 整體架構。

3. 第三層：規則層，將第二層輸入以模糊交及運算(fuzzy AND operation)得到一個法則的激發量(fire strength)，公式如下：

$$u_j^{(3)} = \prod_i u_{ij}^{(2)} . \tag{2.26}$$

4. 第四層：推論層，此層的動作，會將第三層得到的輸出值，乘上一組線性組

合式，其中的參數由使用者控制，其公式如下，其中 N 為輸入個數：

$$u_j^{(4)} = u_j^{(3)}\left(w_{0j} + \sum_{i=1}^{N} w_{ij} x_i\right) \quad . \tag{2.27}$$

5. 第五層：輸出層，做解模糊化的動作，轉換成使用者所需要的資訊，其公式如下，其中 R 是規則數，N 為輸入個數。

$$y = u^{(5)} = \frac{\sum_{j=1}^{R} u_j^{(4)}}{\sum_{j=1}^{R} u_j^{(3)}} = \frac{\sum_{j=1}^{R} u_j^{(3)}\left(w_{0j} + \sum_{i=1}^{N} w_{ij} x_i\right)}{\sum_{j=1}^{R} u_j^{(3)}} \quad . \tag{2.28}$$

第三章 CPSO-S_{DM}

本論文分為兩個部分，第一部分為找尋函數的最佳解，第二部分為函數模擬。在尋找函數最佳解上，主要為改進合作式粒子最佳化演算法，所用的合作式粒子族群最佳化種類為 CPSO-S_K，再以資料探勘尋找所面對例子各維度間的關聯性，做為族群分類的依據；在函數模擬上，以 TSK 模糊類神經系統做為主要核心，再以改良過的 CPSO-S_K 找到最佳化的系統參數，使模擬出的資料能夠更接近真實函數的圖形。

使用 CPSO-S_K 的原因，是因為 PSO 與 CPSO-S 皆有其適合解決的問題，但也有各自在最佳化上的盲點，且仔細觀察這兩種方法，PSO 將所有維度放在同一個族群裡，CPSO-S 講所有分度分到不同的族群，皆可視為 CPSO-S_K 的特例，因此 CPSO-S_K 的應用範圍較廣，且彈性也較大，更適合用於面對各種未知情況的問題。如表 3.1 所示，為三者間對應比較。

表 3.1 各演算法比較。

		PSO	CPSO-S	CPSO-S_K
維度型態	關聯維度	佳	差	佳
	非關聯維度	差	佳	佳
	大維度	差	較 PSO 佳	最佳
族群分類規則		單一族群	一維度一族群	使用者設定

章節 3.1 為本文演算法概述；章節 3.2 為基於 FP-growth 的最佳化粒子族群演算法，此處為本文訂立族群分類規則的核心；章節 3.3 為函數模擬架構。

3.1 演算法概述

現今用於最佳化處理的演算法多為基因演算法(GA)以及粒子族群演算法，而粒子族群演算法為近幾年所發表的最佳化演算法，在表現上並不遜色於基因演算法，使用上也逐漸嶄露頭角，許多改良原始例子族群最佳化的演算法也陸續出現，其中值得注意的方法為 CPSO-S_K，CPSO-S_K 是一套應用廣泛，彈性佳且適合用於解決各種問題的方法，但此演算法最大的缺點為族群分類的標準並無依據，使用者在使用上需以嘗試錯誤法(trial and error)找出最佳的分類方式，意即使用者必須嘗試多種族群分類方式，單純比較之間的效果，才能找出判斷哪一種族群分類是好的，這樣的使用方式，並無法有效的發揮演算法該有的優勢，導致多數使用者對此種方法缺乏興趣。

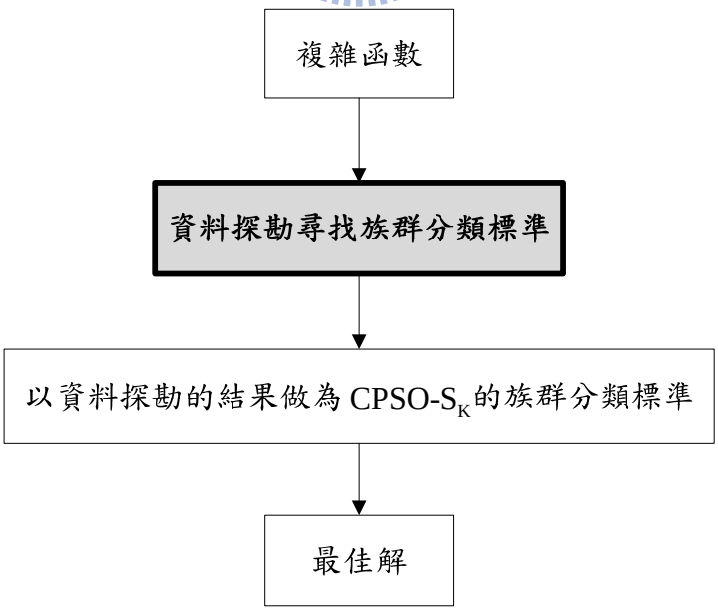


圖 3.1 最佳化基本流程。

本文在最佳化的處理上，同樣使用 CPSO-S_K 做為演算方法，對於無族群分類標準的問題，透過資料探勘來解決並告知使用者哪些維度彼此間具有關聯性，圖 3.1 為最佳化處理時的基本流程，面對複雜函數時，先以資料探勘尋找出個維度關聯性，決定哪些維度該放在同一族群，哪些維度該分開，再依照這個族群分類結果提供 CPSO-S_K 使用，以達到最好的表現。

3.2 基於 FP-growth 的最佳化粒子族群演算法

此章節目的在於訂立 CPSO-S_K 族群分類規則，以各維度間的關聯性做為是否分在同一類的標準，在族群分類上，以 FP-growth 做為搜尋相關性項目的演算法，藉由此方法找出維度之間的關聯程度，以確立族群分類準則。

3.2.1 族群分類規則確立

本論文與原始 CPSO-S_K 最大的差別，就在於加入了資料探勘的技巧，使族群分類方式變為有規則可循，在資料探勘的部分，以 FP-growth 做為找尋關聯性的工具，在使用上首先設定數筆測試資料，再以測試資料的表現為依據定立規則，可分為四個步驟，隨機分組、資料測試、資料選取以及規則確立，說明如下：

1. 隨機分組：CPSO-S_K 最重要的核心便是族群的分類方式，這也是使用上的第一個步驟，但可惜的是，大多數問題並無法察覺該如何將維度做區分，也無法知道分成幾個族群是最佳的方式，也就是說，如果一開始便決定每一筆測試資料的族群各數以及族群內維度各數都是相同的，這樣的設定並不妥當。如圖 3.2 所示，此為一符合實際情況的分類方式，假設問題具有三十個維度，若此時處於最好的分類情況，會發現每個族群內的維度個數並不會相同，例如 swarm_1 包含維度編號{1, 9, 11, 13}共四個維度，swarm_2 只有維度編號{10}一個維度，而 swarm_4 卻含有維度編號{6, 17, 18, 20, 25, 29, 30}共 7 個維

度，除此之外，使用者也無法得知族群個數應如何設定才是最佳解，因此本文在族群個數以及各族群內所包含的維度數目和編號皆以隨機產生，且當一筆測試資料完成後，重新設定族群個數以及各族群內維度數目和編號，增加彼此間配對組合的多樣性。

圖 3.2 實際分類情況。

本文中隨機分組程序將會得到如同圖 3.3 的結果，圖中僅列出三筆測試資料，最適當的測試資料數目需視維度的大小而定，數目取太少，造成配對組合方式不足，這樣的結果意義並不大；反之數目取太多，雖然可更精確得到維度之間彼此的關聯性，但卻會造成資料量龐大，演算時間過長的窘境。此種方式得到的各種配對組合，比較圖 3.2 可發現，是較符合實際情況的分類。

2. 資料測試：將隨機分組後所得到的資料，做為初步族群分組的依據，以這幾種分組情況進行測試，每一筆測試資料經過運算會對應到一筆適切值(fitness value)，這些適切值將構成初步資料庫。

維度總數：30

	第一筆		第二筆		第三筆	
	族群編號	維度編號	族群編號	維度編號	族群編號	維度編號
族群個數	族群_1	2, 5, 8	族群_1	3	族群_1	21, 25, 30
	族群_2	3, 17	族群_2	9, 19, 26, 30	族群_2	3, 19, 27
	族群_3	14, 26	族群_3	8, 10, 23	族群_3	5
	.		.		.	
	.		.		.	
	.		.		.	
	族群_13	9, 16, 25, 30	族群_21	1, 17, 18, 29	族群_9	6, 28, 29

圖 3.3 本文測試資料分類情況。

3. 資料選取：初步資料庫建立後，此步驟主要用來判別資料庫裡各筆資料的好壞，適切值較好的資料，代表分類情況較佳，反之適切值較差的資料，表示當時的分類情況，對最佳化的過程來說是不良的。判別好壞的方式本文以各筆資料適切值的百分之四十最為標準，意即分布於 40%的適切值視為優良，匯入優良的資料成為優良資料庫，且假設一筆資料視為優良，則這筆資料內所有的配對組合都將視為優良，成為未確定規則，其餘資料將視為表現較差的分類情況，刪除這些不必要的資訊，初始資料庫中這些表現較差的分類情況，將不在優良資料庫中出現。資料庫中每一筆資料的型態，將會告知哪些組合是優良的，例如{維度_2, 維度_5, 維度_8}一組為優良、{維度_3, 維度

_17}一組為優良、{維度_9, 維度_16, 維度_25, 維度_30}一組為優良...，這些組合變是未確立規則，需要再透過規則確立的步驟檢驗。

4. 規則確立：得到優良資料庫後，將以這個資料庫裡的各筆資料產生一組或多組規則，以 FP-growth 做為尋找規則的方法，首先將優良資料庫裡的資料建立 FP-tree，由 FP-tree 可找到個維度間的關聯性，其中關聯法則兩個重要的指標支持度和信任度，在本文中所設定最小支持度為整體資料個數的 50%，最小信任度亦為 50%，當完成 FP-tree 後，透過關聯法則可以知道哪些維度分於同一族群裡，可滿足最小支持度及最小信任度，這些組合方式，將成為後續運算上的分群組規則，意即 CPSO-S_K 群族分類規則因此確立。

圖 3.4 為族群分類規則確立的詳細流程圖，族群分類規則訂立後，有可能會產生失敗的情況，若訂立規則後演算過程中，所得到的適切值不滿足優良資料庫的優良標準，則視為規則失敗，將重新訂立族群分類規則，造成規則失敗的原因有兩個：

1. 測試資料錯誤：測試資料在進行演化時，可能陷入局部最佳解，以至於這樣的資料庫並不正確，影響族群分類情況。

2. 測試資料數目過少：測試資料數目太少導致資料庫裡的各筆資料並不完整，此種情況下容易造成族群分類情況產生不客觀的效果。

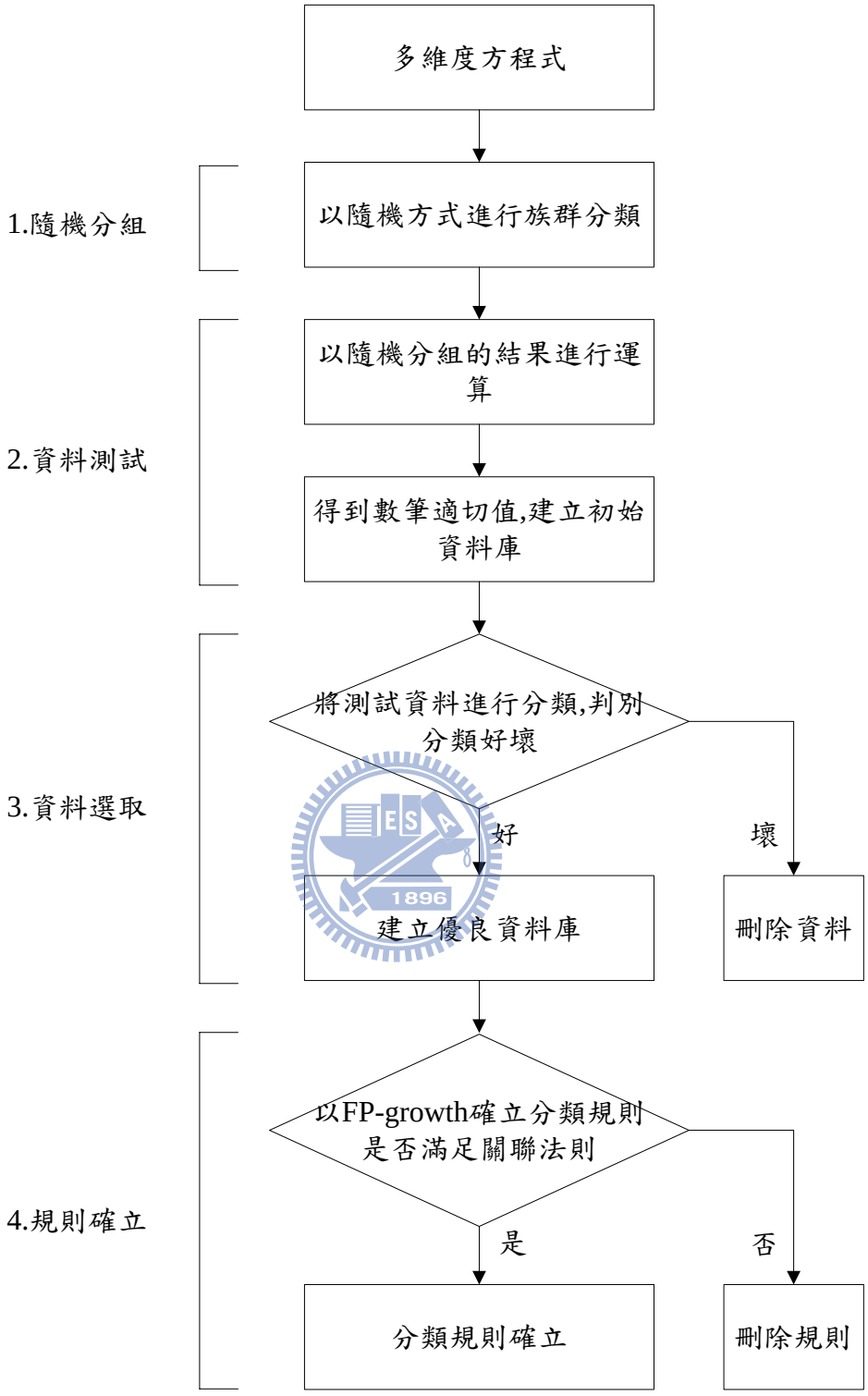


圖 3.4 族群分類詳細流程。

圖 3.5 和圖 3.6 為族群分組簡例，其中包含維度_1 到維度_17 共 17 個維度，在優良資料庫中存放五筆交易資料，圖 3.5 為這五筆交易資料，經過最小支持度

刪除頻率低項目後，再依照頻率的高低由高至低做排序所得的結果，此處最小支持度為 50%，因為資料個數為五筆，所以各項目必須出現至少 3 次才會被選取，其餘出現少於 3 次的項目將被刪除。

優良資料庫

編號	維度項目
1	{維度_1, 維度_3, 維度_4, 維度_6, 維度_7, 維度_9, 維度_13, 維度_16}
2	{維度_1, 維度_2, 維度_3, 維度_6, 維度_9, 維度_15}
3	{維度_2, 維度_6, 維度_8, 維度_10, 維度_15}
4	{維度_2, 維度_3, 維度_11, 維度_16, 維度_17}
5	{維度_1, 維度_3, 維度_5, 維度_6, 維度_12, 維度_13, 維度_14, 維度_16}

最小支持度：50%

編號	維度項目
1	{維度_6, 維度_3, 維度_1, 維度_13, 維度_16}
2	{維度_6, 維度_3, 維度_1, 維度_2}
3	{維度_6, 維度_2}
4	{維度_3, 維度_2, 維度_16}
5	{維度_6, 維度_3, 維度_1, 維度_13, 維度_16}

圖 3.5 族群分類簡例。

圖 3.6 為依據選取後的交易資料所作的頻率項樹，經過最小支持度的篩選後，被選取的維度只剩下{維度_1, 維度_2, 維度_3, 維度_6, 維度_13, 維度_16}

共六個維度，由頻率項樹，可找出個項目的關聯規則，此數最小信任度為 50%，各項目規則如下：

1. 維度_16 相關項目：{維度_6, 維度_3, 維度_1, 維度_13}出現次數 2，{維度_3, 維度_2}出現次數 1，出現次數超過 3 次的項目只有維度_3，因此與維度_16 相關的規則只有{維度_3, 維度 16}這一組，此組規則確立。

   確立規則：{維度_3, 維度_16,}

2. 維度_13 相關項目：{維度_6, 維度_3, 維度_1}出現次數 2，無出現次數超過 3 次的項目，無規則產生。

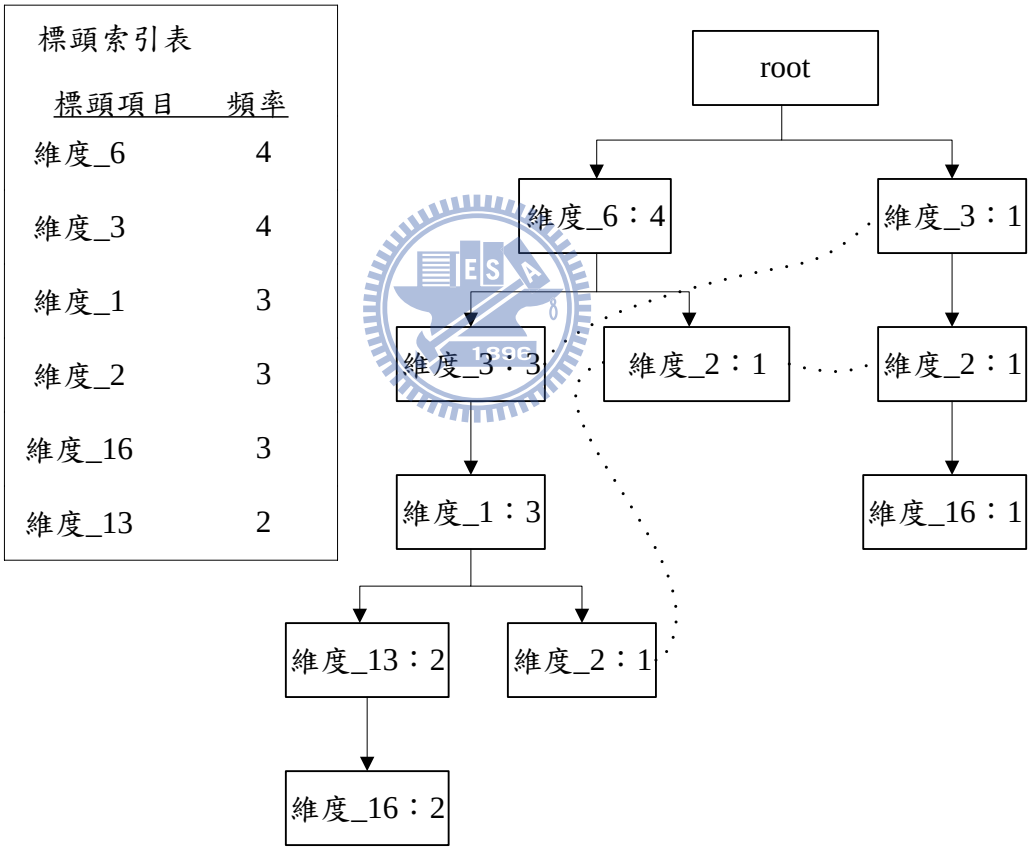


圖 3.6 族群分類簡例頻率項樹。

3. 維度_2 相關項目：{維度_6, 維度_3, 維度_1}出現次數 1，{維度_6}出現次數 1 次，{維度_3}出現次數 1，並無出現次數超過 3 次的項目，規則產生失敗，此處無規則產生。

此範例得到一組規則，規則一為{維度_3, 維度_16}，最後確立的規則含有兩個維度，除了這兩個維度外，其餘維度皆視為獨立的個體，不與其他維度具有關聯性，這些獨立的維度，將各自劃分到不同的族群裡，每個族群中只包含本身一個項目，因此最後可得到的族群分類規則為：族群一{維度_3, 維度_16}、族群二~族群十五分別為{維度_1}、{維度_2}、{維度_4}、{維度_5}、{維度_6}、{維度_7}、{維度_8}、{維度_9}、{維度_10}、{維度_11}、{維度_12}、{維度_13}、{維度_14}及{維度_15}，亦即此處經由關聯性的探勘後，16 個維度將分為 15 個族群，以這 15 個族群做為 CPSO-S_K 族群分類規則，依照此族群分類規則完成後續最佳化的演算。

3.2.2 族群分類規則確立後最佳化演算法

本文所改良最佳化演算法的重心落於族群的分類上，訂立出一套合理的分類規則，規則確立後，以 CPSO-S_K 完成整體最佳化過程，比較圖 2.4 及圖 2.5，PSO 與 CPSO-S 在族群分類規則上較單純，PSO 將所有維度項目分於同一族群裡，每一次的演算若得到較好的最佳解，會一次更新所有維度項目的值。CPSO-S 在族群分類上剛好與 PSO 相反，將所有維度全部獨立，每一個維度會放到一個族群裡，一次只會演化一個維度，其餘的維度皆以 *gbest* 代替。

CPSO-S_K 正好介於 PSO 與 CPSO-S 之間，假設一個維度 6 的例子，各維度以 *A*~*F* 表示，族群個數為三，若分類規則為：族群一{*A, C, F*}、族群二{*E*}、族群三{*B, D* }。如圖 3.7 所示，*A*(*n*)代表 *A* 維度裡第 *n* 個粒子，CPSO-S_K 會先最佳化第一個族群，其中包含三個維度項目{*A, C, F*}，一次的演算法將一起更新這三個維度的最佳解，直到第一個族群最佳化過程完成後，在進行第二個族群的最佳化，此處族群二僅包含一個維度項目，因此族群二的最佳化過程一次只會更新一個維度的最佳解，以此類推，相較於 PSO 與 CPSO-S，CPSO-S_K 會因族群內所包

含的維度數目不同，在最佳化過程一次更新的維度數目也會不同，這也是考慮到維度之間的關聯性所做的改進。

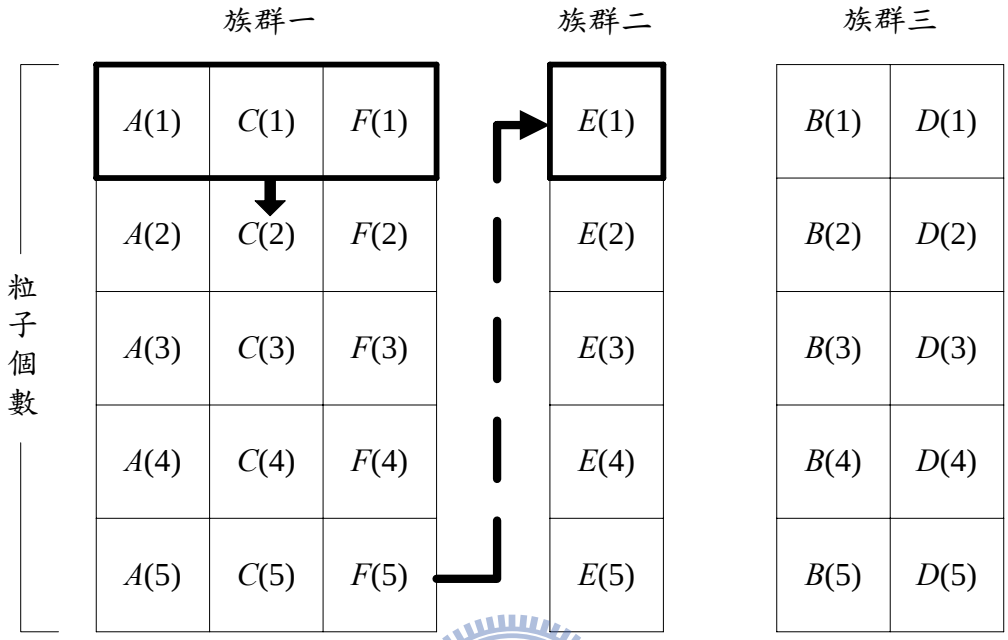


圖 3.7 CPSO-S_K 演化示意圖。

3.3 函數模擬架構

本論文以模糊類神經網路做為函數模擬主要架構，如此的優點在於透過類神經網路，使用上不需建構一組模擬函數的基底函數，可更容易使用此架構達到模擬函數圖型的目標。

3.3.1 架構模糊類神經網路

本文的模糊類神經網路採用 TSK 模糊類神經系統，在架構上分為五層，分別為輸入層、歸屬函數層、規則層、推論層以及輸出層，圖 3.8 為輸入數目 2，

規則數為 5 的 TSK 模糊類神經架構，以下將有各層詳細的介紹：

1. 第一層：輸入層，這一層提供輸入的管道，輸入在第一層中並不做任何運算，或可說輸入訊號與神經元間的權值為 1，僅將輸入傳送至第二層，此層公式如公式 3.1，$u_i^{(n)}$ 為第 n 層中第 i 個輸入的輸出，架構如圖 3.8 所示，輸入個數為 2。

$$u_i^{(1)} = x_i, for\ i=1, 2. \tag{3.1}$$

2. 第二層：歸屬函數層，模糊化第一層的輸出訊號，訊號會由在第一層的單一數值轉換成數個模糊集合，僅程度上不同，本文中使用的歸屬函數為高斯歸屬函數，計算公式如公式 3.2，此層中有兩個重要指標為平均值和標準差，分別以m_{ij}和σ_{ij}來表示，m_{ij}對應到第 i個輸入第 j個歸屬函數的平均值，同理，σ_{ij}對應到第 i個輸入第 j個歸屬函數的標準差。本文在此層中每個輸入對應到 5 個歸屬函數，因為輸入數目為 2，因此第二層中總共 10 個歸屬函數，每個歸屬函數有各自的平均值與標準差，換句話說，此層中所需最佳化的參數數目將有 20 個。

$$u_{ij}^{(2)} = \exp\left(-\frac{[u_i^{(1)} - m_{ij}]^2}{\sigma_{ij}^2}\right), for\ i=1, 2; j=1, 2, \cdots, 5 \tag{3.2}$$

3. 第三層：規則層，執行模糊規則啟動強度的運算，以模糊交及運算得到一個法則的激發量，運算公式如公式 3.3，此層中總共會有 5 個神經元，但此層中並無需要最佳化的參數產生。

$$u_j^{(3)} = \prod_i u_{ij}^{(2)}, for\ i=1, 2; j=1, 2, \cdots, 5. \tag{3.3}$$

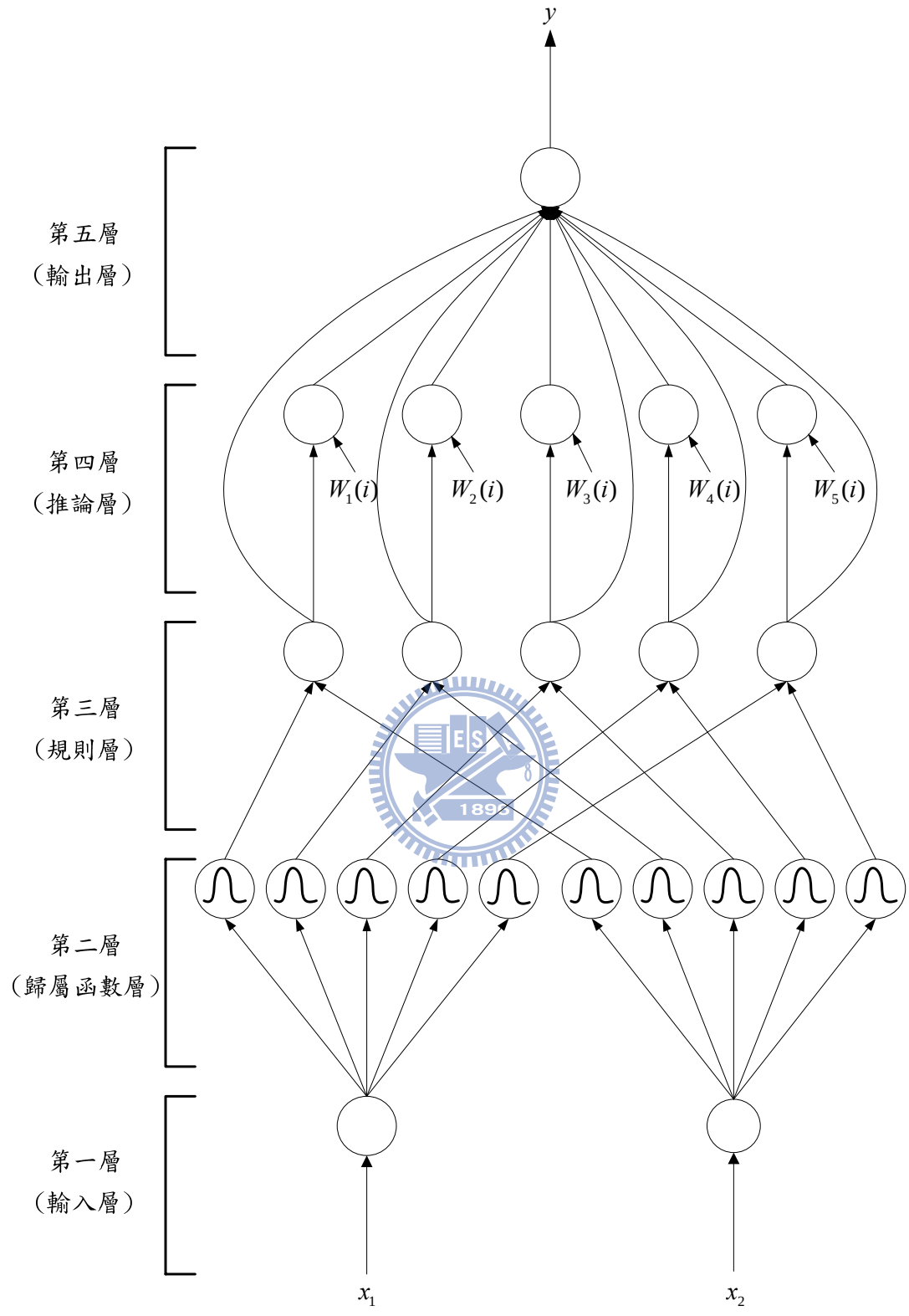


圖 3.8 本論文所使用 TSK 模糊類神經網路架構。

4. 第四層：推論層，此層中會將第三層的輸出，乘上一組線性組合式，以$W_j(i)$表示，推論層用於推論規則層所得到的資訊，如公式 3.4 所示，其中 N 為輸

入個數，每一組$W_j(i)$會產生三個可調變數，例如$W_1(i)=w_{01}+(w_{11}x_1+w_{21}x_2)$，其中$w_{01}$、$w_{11}$以及$w_{21}$為使用者須調整參數，因此在此層中，總共會有 15 個參數須做最佳化的處理。

$$u_j^{(4)}=u_j^{(3)}W_j(i), \tag{3.4}$$

其中$W_j(i)=w_{0j}+\sum_{i=1}^{N}w_{ij}x_i$, *for* i=1, 2; j=1, 2, ⋯, 5.

5. 第五層：輸出層，將系統做解模糊化的動作，成為使用者所需要的資訊，如公式 3.5 所示，R 為規則數，N 為輸入數目。

$$y=u^{(5)}=\frac{\sum_{j=1}^{R}u_j^{(4)}}{\sum_{j=1}^{R}u_j^{(3)}}=\frac{\sum_{j=1}^{R}u_j^{(3)}(w_{0j}+\sum_{i=1}^{N}w_{ij}x_i)}{\sum_{j=1}^{R}u_j^{(3)}}, \tag{3.5}$$

其中 i=1, 2; j=1, 2, ⋯, 5.

以 TSK 進行函數模擬具有三項優點：

1. 以模糊類神經模擬函數，不需建構模擬函數時所需的基底函數，更容易達到函數模擬的目標。
2. 類神經網路具備由輸入及輸出推論非線性函數的能力。
3. 常見模糊類神經網路為 TSK 與 Mamdani 兩類架構，差別在於 TSK 增加第四層即為推論層，此層乘上一組線性組合式，對於解決非線性問題具有較佳的能力。

模糊類神經系統表現的好壞，將直接受到上述提及的幾個參數所影響，參數的個數即為最佳化過程中變數的數目，即為系統的維度，意即系統維度的大小，將取決於規則數和輸入數目，參數個數的算法如公式 3.6 所示，本文中規則數為 5，輸入數為 2，因此總共會產生 35 個參數須做最佳化的調整，分別為{m_{11}, σ_{11}, m_{12}, σ_{12}, m_{13}, σ_{13}, m_{14}, σ_{14}, m_{15}, σ_{15}, m_{21}, σ_{21}, m_{22}, σ_{22}, m_{23}, σ_{23}, m_{24}, σ_{24}, m_{25}, σ_{25}, w_{01}, w_{11}, w_{21}, w_{02}, w_{12}, w_{22}, w_{03}, w_{13}, w_{23}, w_{04}, w_{14}, w_{24},

w_{05}, w_{15}, w_{25}}共 35 個。

$$\text{參數個數}=2\times\text{規則數}\times\text{輸入數}+(1+\text{輸入數})\times\text{規則數}. \quad (3.6)$$

輸入數目由題型所決定，而規則數由使用者所訂定，規則數太少會使演算法無法學習到所要的目標，但規則數越多並不代表系統會表現越好，過多的規則數容易造成演算過程中計算量的沉重負擔，例如若設定 7 個規則數，2 個輸入數，則最佳化過程將會面對將近 50 個維度的問題，這樣的維度大小，計算量相當龐大，將會造成演算上太過於沉重的負擔。

3.3.2 最佳化過程

上述 TSK 架構所得到需最佳化參數為 35 個，本文以 CPSO-S_K 做為調整參數的最佳化工具，此處族群分類的依據，透過章節 3.2 來確立規則，整體流程如圖 3.9。

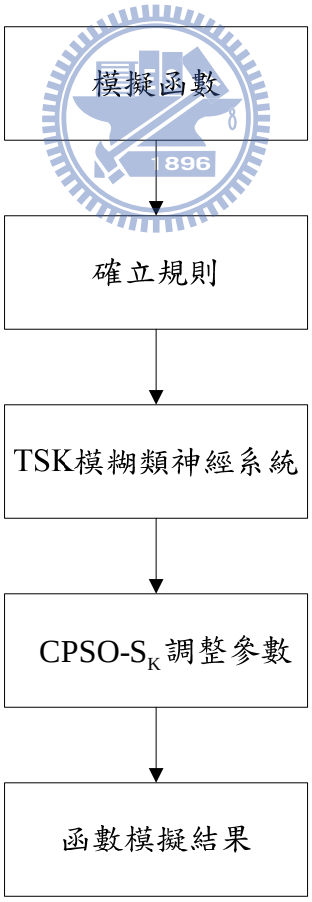


圖 3.9 函數模擬流程圖。

第四章 實驗數據

本章節將提及兩部分，第一部分為函數最佳解的尋找，改良原有 CPSO-S_K 無分類依據的窘境，以 FP-growth 找尋關聯維度，提供演算法中的分類準則。第二部分將運用於函數模擬，以訂立規則後的 CPSO-S_K 結合 TSK 模糊類神經網路，將模糊類神經網路中的數個參數做最佳化的處理，使模擬後的函數圖形可接近於實際函數圖型。本研究實驗用設備為 Intel Core Duo 1.86GHz 中央處理器，1.5GB 記憶體的筆記型電腦，使用 Matlab R2008b 做為撰寫程式及實驗模擬軟體。各節敘述如下，4.1 節為函數最佳化實驗，4.2 節為函數模擬圖型實驗，4.3 節為實驗模擬分析。

4.1 函數最佳化

本節將提供數個函數進行最佳化的模擬，函數的種類包含各維度間相關或各維度間獨立以及不全相關及不全獨立的不同類型，透過 FP-growth 找出關聯的維度，確立 CPSO-S_K 族群分類標準，使合作式粒子族群最佳化演算法可達到更好的效果。

4.1.1 最佳化方程式選取

本文中用於最佳化模擬的方程式有四種，其中包含 Rosenbrock function、Ackley's function 以及 generalized Rastrigin function 等三種，以 f_0~ f_3 表示，各方程式如方程式 4.1 至方程式 4.4 所示，n 為各方程式中維度個數，本實驗中 n 值皆為 30，意即各方程式皆具有 30 個維度。

f_0 各維度間為完全相關，每個維度之間都具有關聯性，意即關聯的情況並不只是其中某幾個維度，而是所有的維度都是關聯項目，不同的維度將彼此影響其它維度的演化，也就是說此種方程式若族群分類結果較接近 PSO，能把所有維度分在同一類，在演化上的效果會較好； f_1 及 f_2 各維度間為完全獨立，每個維度並不受其它維度的影響，此種情況在族群分類上，若能將所有維度各自分到不同的族群，意即類似於 CPSO-S 的族群分類準則，在表現上會得到較好的效果； f_3 為維度部分相關且部分獨立，可用於判斷本論文演算法是否具備提升粒子族群演算法效能的能力。本文中所提出基於 FP-growth 的 CPSO-S_K 演算法，便是為了達到這樣的結果，對於維度相關方程式或是維度非相關方程式甚至於維度間部分相關且部分獨立方程式，皆能做出最適當的處理。

Rosenbrock function：

$$f_0(x)=\sum_{i=1}^{\frac{n}{2}}(100(x_{2i}-x_{2i-1}^2)^2+(1-x_{2i-1})^2). \tag{4.1}$$

Ackley's function：

$$f_1(x)=-20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^{\,2}})-\exp(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i))+20+e. \tag{4.2}$$

generalized Rastrigin function：

$$f_2(x)=\sum_{i=1}^{n}(x_i^{\,2}-10\cos(2\pi x_i)+10). \tag{4.3}$$

$$f_3=\left(\sum_{i=1}^{m/2}(100(x_{2i}-x_{2i-1}^2)^2+(1-x_{2i-1})^2)\right)+\left(\sum_{i=m+1}^{n}x_i^2-10\cos(2\pi x_i)+10\right). \tag{4.4}$$

4.1.2 函數最佳化實驗結果

本論文在函數最佳化部分比較的對象為 PSO 以及 CPSO-S，三種方法在演算

上每個族群維度大小並不相同，因此若以程式執行速度來比較並不公平，且這三種演算法的演算方式有所差異，每一個演化代中所做的份量亦不相同，因此也不可以演化代數做為相同的限制，因此本論文在停止條件的設定上，以 function evaluations(FEs)的數量做為準則，每運算一次適切值則 FEs 值增加 1。表 4.1 為各方程式初始設定參數。

表 4.1 實驗參數初始設定。

函數編號	函數維度	初始範圍	演化門檻值 (threshold)
f_0	30	2.048	100
f_1	30	30	5.00
f_2	30	5.12	100
f_3	30	30	5

本論文所使用的方法以及比較的兩種演算法在 function evaluations 的數目上皆設定為 2×10^5，所有實驗的數據皆測試 50 次，最後所得函數值為 50 次實驗的平均值，以下為 PSO、CPSO-S 以及本論文在參數上的初始設定，三者在演化公式上如公式 2.5 所示。

1. PSO： c_1=1.49，c_2=1.49，w=0.72，族群個數只有一個，將所有維度皆放於同一維度中。
2. CPSO-S：c_1=1.49，c_2=1.49，w 初始值為 0.72，w 值以線性遞減至運算時間終了，族群個數為 30，每一個族群僅擁有一個維度，及每個維度演化上皆為

獨立的狀態。

3. 本論文：最佳化演算法以 CPSO-S_K 做為基礎，c_1=1.49，c_2=1.49，w 初始值為 0.72，w 值以線性遞減至運算時間終了，族群的個數視 FP-growth 找尋關聯維度的結果而定，具關聯性的維度放於同一維度裡，其餘維度全部分散於不同的族群中。

在粒子數的取法上，為使模擬可以得到較客觀的結果，本論文先做粒子數測試，表 4.2~表 4.5 為 f_0~ f_3 粒子數由 1~50 的實驗結果。

表 4.2 f_0 粒子數測試結果。

	粒子個	函數值		粒子個	函數值		粒子個	函數值
f_0	1	34.6844		18	0.00733		35	0.00764
	2	22.9581		19	0.00791		36	0.00724
	3	4.27145		20	0.00699		37	0.00756
	4	1.54612		21	0.00692		38	0.00761
	5	0.67832		22	0.00698		39	0.00764
	6	0.36900		23	0.00713		40	0.00767
	7	0.30596		24	0.00713		41	0.00771
	8	0.23573		25	0.00729		42	0.00789
	9	0.19382		26	0.00735		43	0.00802
	10	0.15088		27	0.00744		44	0.00792
	11	0.10399		28	0.00762		45	0.00771
	12	0.05529		29	0.00765		46	0.00734
	13	0.01938		30	0.00762		47	0.00823
	14	0.00921		31	0.00768		48	0.00894
	15	0.00703		32	0.00763		49	0.00719
	16	0.00702		33	0.00772		50	0.00772
	17	0.00701		34	0.00742			

表 4.3 f_1粒子數測試結果。

	粒子	函數值		粒子	函數值		粒子	函數值
f_1	1	0.074218		18	0.000177		35	0.000169
	2	0.028183		19	0.000173		36	0.000179
	3	0.007915		20	0.000169		37	0.000159
	4	0.001627		21	0.000169		38	0.000166
	5	0.001061		22	0.000170		39	0.000162
	6	0.000734		23	0.000167		40	0.000172
	7	0.000601		24	0.000163		41	0.000179
	8	0.000355		25	0.000172		42	0.000171
	9	0.000241		26	0.000160		43	0.000173
	10	0.000197		27	0.000173		44	0.000172
	11	0.000183		28	0.000183		45	0.000162
	12	0.000186		29	0.000169		46	0.000181
	13	0.000174		30	0.000168		47	0.000153
	14	0.000176		31	0.000165		48	0.000178
	15	0.000173		32	0.000166		49	0.000162
	16	0.000179		33	0.000167		50	0.000168
	17	0.000177		34	0.000170			

表 4.4 f_2 粒子數測試結果。

	粒子	函數值		粒子	函數值		粒子	函數值
f_2	1	0.01030		18	0		35	0
	2	0.00541		19	0		36	0
	3	0.00283		20	0		37	0
	4	0.00119		21	0		38	0
	5	0.00023		22	0		39	0
	6	0.000052		23	0		40	0
	7	0		24	0		41	0
	8	0		25	0		42	0
	9	0		26	0		43	0
	10	0		27	0		44	0
	11	0		28	0		45	0
	12	0		29	0		46	0
	13	0		30	0		47	0
	14	0		31	0		48	0
	15	0		32	0		49	0
	16	0		33	0		50	0
	17	0		34	0			

表 4.5 f_3 粒子數測試結果。

	粒子個數	函數值		粒子個數	函數值		粒子個數	函數值
f_3	1	3.5399017		18	0.0000153		35	0.0000146
	2	1.3345612		19	0.0000134		36	0.0000147
	3	0.6339643		20	0.0000142		37	0.0000142
	4	0.0235110		21	0.0000133		38	0.0000158
	5	0.0514271		22	0.0000138		39	0.0000132
	6	0.0036362		23	0.0000140		40	0.0000140
	7	0.0007339		24	0.0000143		41	0.0000149
	8	0.0000533		25	0.0000159		42	0.0000147
	9	0.0000493		26	0.0000151		43	0.0000152
	10	0.0000328		27	0.0000149		44	0.0000159
	11	0.0000287		28	0.0000152		45	0.0000169
	12	0.0000231		29	0.0000151		46	0.0000148
	13	0.0000210		30	0.0000142		47	0.0000149
	14	0.0000191		31	0.0000148		48	0.0000152
	15	0.0000193		32	0.0000157		49	0.0000144
	16	0.0000188		33	0.0000148		50	0.0000148
	17	0.0000162		34	0.0000157			

由表 4.2~表 4.5 可發現，各函數在粒子數小於 10 時有較明顯的變化，維度間粒子數過小會導致整體效能嚴重受到影響，當粒子數大於 15 時，表現上變化不大，因此本論文在粒子數的選取上，以個數 5, 10, 15, 20 及 25 等 5 種情況進行模擬。

演算法模擬 f_0 的實驗結果如表 4.6 所示，此方程式為一個維度完全相關的例子，每個維度間彼此都具有關聯性，因此透過本論文的族群分類機制，會將所有的維度皆放於同一族群裡，如此的分類結果與 PSO 的族群分類方式是相同的，所以在演化上會與 PSO 所得到的結果相近，由表 4.2 可以看出，此一維度完全相關的函數，若以 PSO 進行演化會比 CPSO-S 來得有效，雖然 CPSO-S 可避免 PSO 演算法的缺點“兩步向前，一步後退”，但 CPSO-S 將關聯維度劃分開來所造成的缺陷遠大於自身演算法所提供的優點，因此結果如同第二章所述，維度關聯函數仍以 PSO 可得到較好結果。

表 4.6 f_0 於各方法實驗結果。

函數	演算法	各維度粒子數	函數值
f_0	本論文	5	0.67977
		10	0.15088
		15	0.00703
		20	0.00699
		25	0.00692
	PSO	5	0.65712
		10	0.14900
		15	0.00653
		20	0.00757
		25	0.00741
	CPSO-S	5	2.13751
		10	0.73661
		15	0.6684
		20	0.83584
		25	0.85302

此次實驗中，有個較值得注意的部分，表 4.6 中 PSO 演算法當各維度包含 15 個粒子以及 20 個粒子時，由實驗數據看出，維度中包含的粒子數越多，有時並不一定會使演化效果更好，主要原因在於本實驗不以演化代數做為限制，而是

以 function evaluations 代替演化代數做為演算法終止條件，意即當每個維度中粒子數增加，每一次演化代中將消耗更多 function evaluations 的次數，若大略以演化代數來看，維度中粒子數多的實驗，最佳化過程中起始至演化結束，執行的演化代數會比維度中粒子數較少的實驗來得少，以致於會出現粒子數較多反而較差的情況。但粒子數的影響主要在於個數小於 10 的情況，因此當粒子個數足夠時，彼此間差異並不大。

表 4.7 為方程式 f_1 於各種實驗方法所得到的結果，f_1 的各維度間彼此獨立，所有維度皆不相關，因此本論文在此實驗中，族群分類後的結果為所有維度皆放於不同的族群裡，總共產生族群個數為 30，此種分法與 CPSO-S 相同，意即本論文的方法在實驗數據將會於 CPSO-S 相近。由實驗數據可明顯看出，當函數的類型為維度間具有相關性，且維度數目較大時，CPSO-S 的表現會明顯優於 PSO，本論文藉由 FP-growth 尋找關聯項目，應用於此類型方程式也可達到與 CPSO-S 相近的結果。

表 4.7 f_1 於各方法實驗結果。

函數	演算法	各維度粒子數	函數值
f_1	本論文	5	0.001066
		10	0.000197
		15	0.000173
		20	0.000169
		25	0.000168
	PSO	5	25.8342
		10	6.84017
		15	4.81892
		20	3.70377
		25	3.60661
	CPSO-S	5	0.001103
		10	0.000257
		15	0.000197
		20	0.000186
		25	0.000172

表 4.8 為方程式 f_2 於各種實驗方法所得到的結果，f_2 與 f_1 在維度關聯性上具有相同特性，每個維度皆是互相獨立，彼此之間並不影響，因此本論文在族群分

類結果上會與 CPSO-S 相同，將所有維度各自分到不同的族群裡，總共會產生 30 個族群，在實驗結果上也會與 CPSO-S 相近。此次實驗可更明顯看出族群分類法則應用於各維度獨立方程式的重要性，PSO 在此實驗表現並不理想，平均函數值皆相當大，以最佳化的觀點來說並不成功，反觀本論文以及 CPSO-S，函數值達到 0，此處函數值並不為接近 0 的數字，而是真正使函數值達到 0，在最佳化過程中這種結果並不常見，本論文將於章節 4.3 實驗結果分析中說明。

表 4.8 f_2 於各方法實驗結果。

函數	演算法	各維度粒子數	函數值
f_2	本論文	5	0.00023
		10	0
		15	0
		20	0
		25	0
	PSO	5	138.337
		10	86.5612
		15	67.3586
		20	56.9372
		25	51.4623
	CPSO-S	5	0.00019
		10	0
		15	0
		20	0
		25	0

表 4.9 為 f_3 於各實驗方法所得結果，f_3 為一函數部分相關且部分獨立的方程式，PSO 與 CPSO-S 因為在族群分類的設定上，都無法與實際情況得到較接近的族群分類結果，因此在實驗結果上，因為本論文在族群分類規則上較自由，並沒

有一開始便設定好的情況，可依照題型的不同做自主調整，可得到較佳的最佳解搜尋，尤其是如函數 f_3 這類 PSO 與 CPSO-S 皆表現不好的題型，本論文對於提升粒子族群演算法的優點，可明顯的表現出來。

表 4.9 f_3 於各方法實驗結果。

函數	演算法	各維度粒子數	函數值
f_3	本論文	5	0.0514
		10	0.000032
		15	0.000019
		20	0.000013
		25	0.000014
	PSO	5	17.988
		10	2.3090
		15	2.0079
		20	1.5442
		25	1.6213
	CPSO-S	5	3.2959
		10	0.00879
		15	0.00631
		20	0.00517
		25	0.00553

4.2 函數模擬

本節將以 TSK 模糊類神經網路進行函數模擬，利用上述所提到確立規則後 CPSO-S_K 尋找 TSK 模糊類神經網路的各參數值，在演算過程中，藉由 FP-growth 找尋 TSK 各參數間的關聯性，以此改良後的 CPSO-S_K 使模糊類神經網路在函數模擬上更接近實際函數的圖形表現。

4.2.1 函數類型選取

本論文中提供三組函數進行函數圖型模擬，如以下方程式 4.5、方程式 4.6 及方程式 4.7，此處與函數最佳化不同，執行函數最佳化時，FP-growth 用於尋找函數間各變數的關聯性，如方程式 4.2，族群分類規則的訂立為方程式中 x_i 之間是否具有關連性，而函數模擬在訂立族群分類規則上對方程式裡的各變數並不感興趣，在此處 FP-growth 將尋找 TSK 模糊類神經系統中各變數的關聯性質，如各平均值及標準差，因此雖然函數模擬所提供的方程式看起來較函數最佳化方程式來的簡單且容易，但實際上函數模擬在演化上所處理的維度數目，大部分都會比函數最佳化來得多，執行上計算量的負擔也較重。

函數一：

$$y = \begin{cases} x, & 0 \le x \le 3, \\ 7.5 - 1.5x, & 3 < x \le 5. \end{cases} \tag{4.5}$$

函數二：

$$y = (1 + x_1^{-2} + x_2^{-1.5})^2 \ , \ 1 \le x_1, x_2 \le 5. \tag{4.6}$$

函數三；

$$y = \begin{cases} (1 + x_1^{-2} + x_2^{-1.5})^2, & 2 \le x_2 \le 3, \\ 5 + \sin(2x_1) + \sin(2x_2), & 3 < x \le 4. \end{cases} \tag{4.7}$$

4.2.2 函數模擬實驗結果

本文中效能比較以均方根誤差(root mean square error, RMSE)為標準，定義如公式 4.8 所示，

$$\text{RMSE} = \frac{\sqrt{\sum_{j=1}^{N}(y_j - \hat{y}_j)^2}}{N}, \tag{4.8}$$

其中 N 為取點個數; y_j 為 j 點的實際值; $\hat{y}_j$ 為 j 點時本系統輸出值。

本文在 TSK 參數最佳化所使用演算法為改良後 CPSO-S_K ，因此在比較對象上，以 PSO 和 CPSO-S 做為參數最佳化的兩種方法，當最佳化演算法為 PSO 時，初使參數設定為 c_1=1.49、c_2=1.49、還有 w=0.72，而當最佳化演算法為本論文與 CPSO-S 時，兩者設定相同，c_1=1.49，c_2=1.49，w 初始值為 0.72，w 值以線性遞減至運算時間終了。包含本論文等三種方法在 function evaluations 數目皆為 1000 次，函數一與函數二設定如表 4.10 所示。

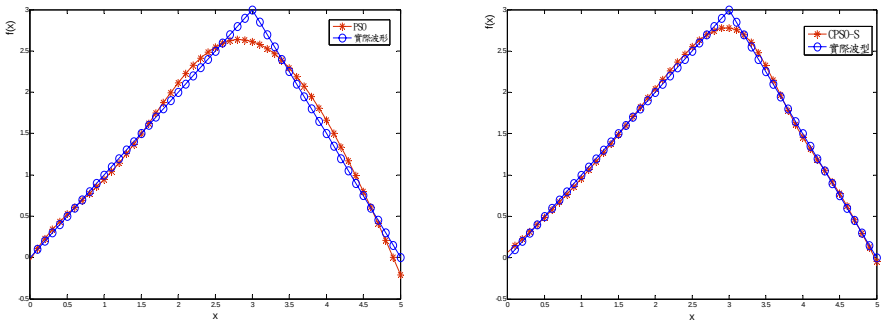
表 4.10 函數初始設定。

函數編號	TSK 規則數目	初始範圍	演化門檻值
函數一	5	5.12	100
函數二	5	10	100
函數三	5	10	100

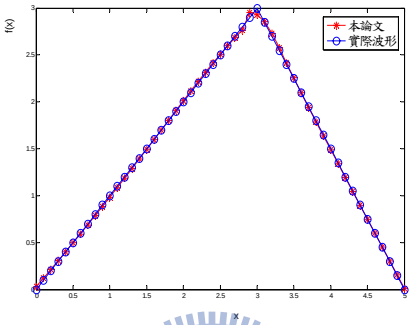
表 4.11 為函數一模擬結果，由表中可看出，以 PSO 為參數最佳化方法在表現上較不理想，而 CPSO-S 雖然表現較 PSO 好，但以 RMSE 來看，仍然是以本文所提出改良後方法較能最好，圖 4.1 為三種方法模擬圖型，本論文與 CPSO-S 較接近實際圖，且在變化較大的部分，本論文明顯優於其他兩種演算法。圖 4.2 為 PSO、CPSO-S 與本論文三種方法與實際同形的誤差值。

表 4.11 函數一模擬結果。

	RMSE
以 PSO 調整 TSK 參數	0.0169
以 CPSO-S 調整 TSK 參數	0.0075
以本論文調整 TSK 參數	**0.0026**



(a) (b)



(c)

圖 4.1 函數一模擬結果：(a)以 PSO 調整 TSK 參數，(b)以 CPSO-S 調整 TSK 參數(c)以本論文調整 TSK 參數。

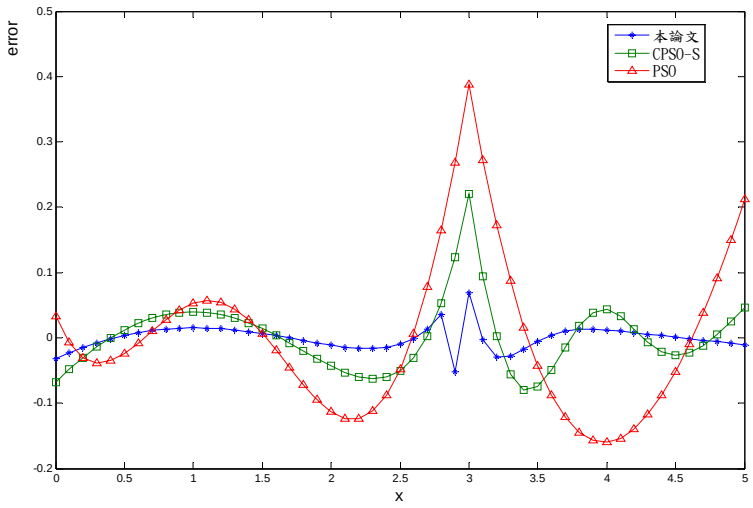


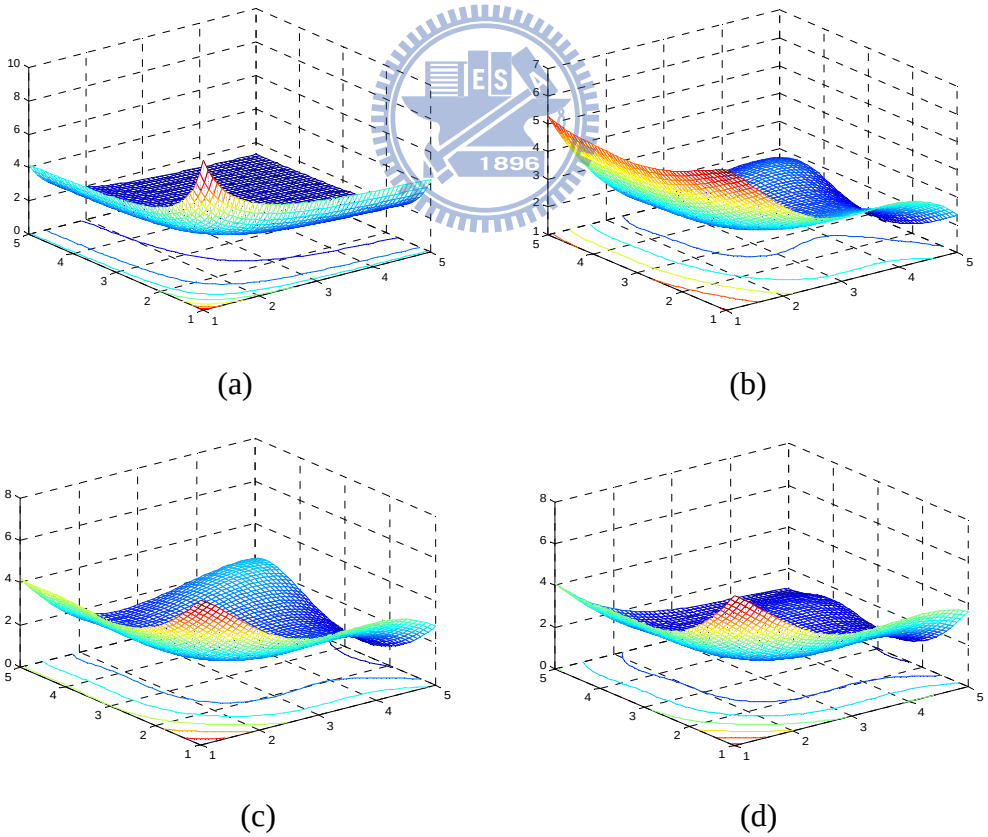
圖 4.2 三種方法誤差值。

表 4.12 為函數二模擬結果，函數二在困難度上較函數一來得高，因此效能表現上會比模擬函數一來的差。由表 4.12 可看出，模擬結果依舊以本文所使用

方法最好，若分別觀察三種方法，PSO 演化速度快，演化初期效果也不錯，但當需要做較準確的微調參數時，此時效果較差，並無法得到好的模擬結果；CPSO-S 雖然速度慢，但其演化效果在各階段較為平均，也可以比較準確調整較細微的部分，在表現上優於 PSO；本文在三中方法中表現最好，在演化速度上介於兩者之間，在演化過程中，本論文演算法擷取 PSO 與 CPSO-S 的優點，無論是在細微處的微調，或是效能表現上皆優於其餘兩種，圖 4.3 為三種方法用於模擬函數二圖型。

表 4.12 函數二模擬結果。

	RMSE
以 PSO 調整 TSK 參數	0.0106
以 CPSO-S 調整 TSK 參數	0.0086
以本論文調整 TSK 參數	**0.00453**



(a) (b)

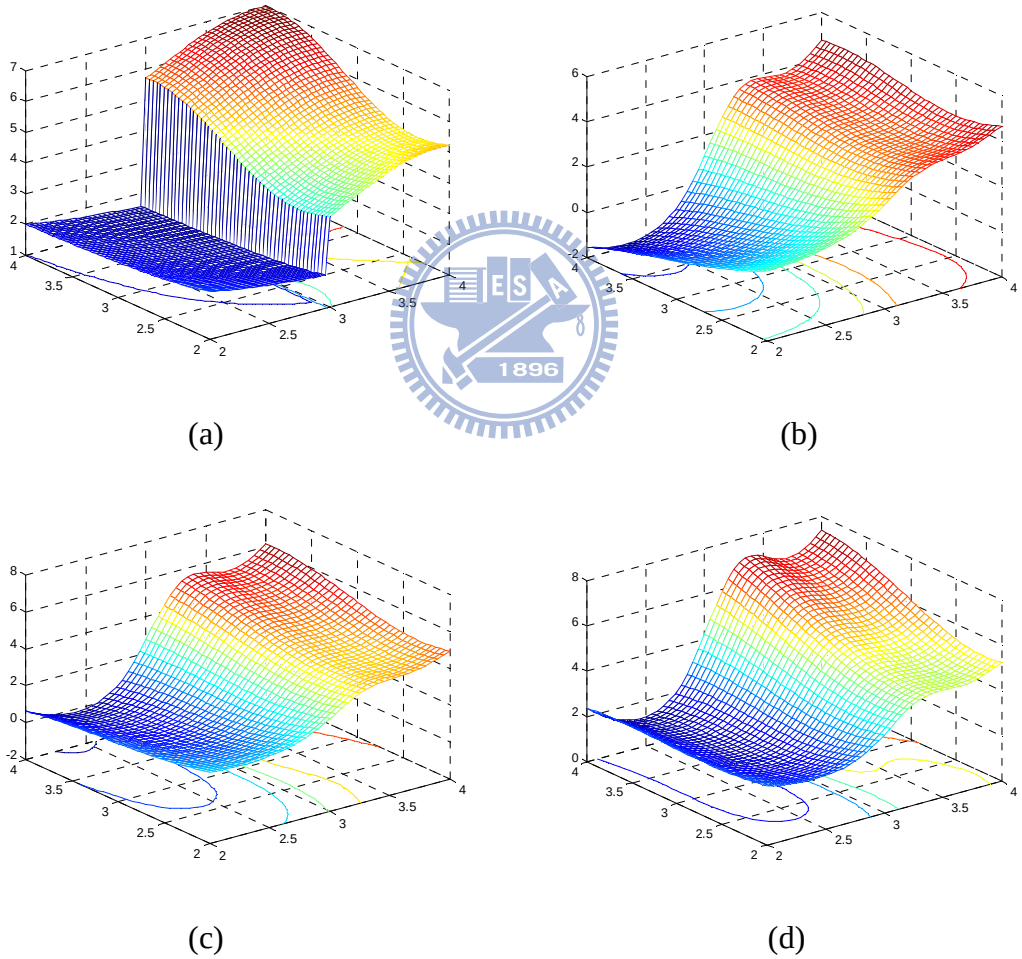
(c) (d)

圖 4.3 函數二模擬結果：(a)實際圖型，(b)以 PSO 調整 TSK 參數，(c)以 CPSO-S 調整 TSK 參數，(d)以本論文調整 TSK 參數。

表 4.13 為函數三模擬結果，函數三為一不連續函數，在圖形中具有一段層面，圖 4.4 為三種方法模擬函數三圖型。

表 4.13 函數二模擬結果。

	RMSE
以 PSO 調整 TSK 參數	0.0558
以 CPSO-S 調整 TSK 參數	0.0279
以本論文調整 TSK 參數	**0.0115**



(a) (b)

(c) (d)

圖 4.4 函數三模擬結果：(a)實際圖型，(b)以 PSO 調整 TSK 參數，(c)以 CPSO-S 調整 TSK 參數，(d)以本論文調整 TSK 參數。

函數三在難度上遠高於函數一和函數二，主要在斷層面上逼近困難，表 4.9

及圖 4.4 可看出，模擬函數三得到的結果並不理想，但本論文在表現上能可優於其他兩種方法。

4.3 模擬結果分析

在函數最佳化的實驗中，維度內粒子的個數會影響效能的表現，尤其當粒子個數過於小時，會造成較大的影響，因此在使用的建議上，可將粒子個數設定在 15~25 之間，彼此之間差異不大。

函數 f_0 各維度間皆為相關，因此本論文在確立規則步驟時，當測試資料族群分類情況與 PSO 相近時，可得到較好的效能，所以本論文經過 FP-growth 尋找關聯項目，可得到類似於 PSO 的族群分類規則，意即把所有的維度置於一個族群內，以實驗結果來看，即使 CPSO-S 不會有“兩步向前,一步向後”的不良效果，但因為函數 f_0 的維度全相關性，因此結果如預期的想法，本論文與 PSO 表現會較 CPSO-S 好。

函數 f_1 與函數 f_2 在實驗中，可明顯看出選擇最佳化演算法的重要性，此兩函數各維度間都是彼此獨立且維度數目大，對於 PSO 來說，“兩步向前，一步向後”的影響成為相當嚴重缺點，且 PSO 本身即對維度較大的函數表現不甚理想，因此在實驗數據上才會有如此明顯的差距，其中值得注意的部分為函數 f_2，在表 4.4 中，存在著較讓人疑惑的數據，因為 f_2 為個維度間獨立的函數，當函數 f_2 以 CPSO-S 及本論文做最佳化演算時，在實驗結果可得到最佳解為 0，此處並不會近似解，而為實際值 0，主要原因在於函數 f_2 如公式 4.3 所示，仔細觀察此函數，可發現當各維度的值皆為 0 時，可得到最後輸出值為 0，意即變數 x_1，x_2，…，x_{29}，x_{30} 皆等於 0 時，則輸出 f_2 =0，但 PSO 卻無法達到這樣的結果，主要為受到關聯維度以及過大的維度所影響。

函數 f_3 實驗中，維度部分相關且部分獨立，因此 PSO 及 CPSO-S 在本實驗中表現並不好，本論文演算法在族群分類上彈性較佳，可使族群分類較接近實驗函數的實際情況，因此本論文在函數 f_3 的實驗上，可達到提升粒子族群演算法效能。

PSO 與 CPSO-S 在最佳化過程中皆有較適合及較不適合的函數類型，這兩種演算法並無法通用於所有函數的最佳化，表 4.14 為本論文與這兩種方法的對照表，由表中可看出，本論文在處理 PSO 及 CPSO-S 表現差的函數時，皆能達到好的效果，因此在使用上，可通用於各種類型的方程式。

表 4.14 各方法應用於三種函數結果對照表。

		本論文	PSO	CPSO-S
f_0	族群分類規則	所有維度置於同一族群	所有維度置於同一族群	所有維度分開至不同族群
	表現優劣	良好	良好	差
f_1	族群分類規則	所有維度分開至不同族群	所有維度置於同一族群	所有維度分開至不同族群
	表現優劣	良好	差	良好
f_2	族群分類規則	所有維度分開至不同族群	所有維度置於同一族群	所有維度分開至不同族群
	表現優劣	良好	差	良好
f_3	族群分類規則	部分分於各不同族群，部分各自獨立	所有維度置於同一族群	所有維度分開至不同族群
	表現優劣	良好	差	差

在函數模擬的實驗中，以 PSO 為參數最佳化演算法雖然速度較快，但卻只能做大略的調整，並無法做到較細微的圖形曲線逼近；而以 CPSO-S 做為最佳化演算法速度上較慢，但卻可以較準確逼近圖型；本論文在速度上介於 PSO 與 CPSO-S 之間，族群分類結果大多會將每一組輸入在歸屬函數層的平均值和標準差置於同一族群內，如公式 3.2 所示，平均值和標準差在 TSK 模糊類神經系統中為相關變數，因此這樣的分類結果有助於演算法效能的表現，此外本文以改良式 CPSO-S_K 做為參數最佳化工具，可擷取 PSO 與 CPSO-S 兩種演算法的良好特性，最重要的部分為 CPSO-S_K 在演化上彈性大，既可改善 PSO“兩步向前,一步向後”的缺點，也可解決 CPSO-S 對相關項目演化困難的窘境，因此本文在模擬結果上，皆比其餘兩者來得好。

第五章 結論

粒子族群演算法因為在使用上簡單且有效，隨然發展至今並不算很長的時間，但在許多的應用上卻是相當廣泛，其中最著名的兩種方法為 PSO 及 CPSO-S，雖然這兩種方法可以解決多數問題，但卻各自存在無法解決的問題類型，而 CPSO-S_K 卻可同時以包含這兩種演算法的方式存在，在彈性和使用性上皆比前兩者來的高，但當使用 CPSO-S_K 時，所遭遇到的問題便是無法確立族群分類規則，雖然演算法看似彈性且實用，但卻讓使用者在使用的第一步就遇到阻礙，若以嘗試錯誤法來做為 CPSO-S_K 使用上的輔助，這樣的結果並不讓人滿意且貢獻並不大，因此若要廣泛使用 CPSO-S_K，族群分類規則的訂立應該最重要的部分，一套有效的族群分類方式，才可使此種演算法更具有價值。

本文所提到將 CPSO-S_K 結合資料探勘，以 FP-growth 尋找各維度間的關聯性，如此一來便可解決無族群分類規則可尋的窘境，使用者在一開始便可訂立出合理的族群分類方法，徹底解決 CPSO-S_K 執行上所遇到的瓶頸，由第四章實驗結果可看出，透過關聯性的尋找，所得到的族群分類結果為合理且實用，省去了很多利用嘗試錯誤法所浪費的時間。

此外，當處理的問題具有過大的維度時，因為 PSO 在維度過大的情況下效果會因“兩步向前，一步後退” 而表現不佳，每一次的演化中，僅考量整體的適切值，各維度是否較上一代來得好，PSO 並無法確保，對於最佳解的找尋造成相當大影響，因此多數使用者在沒有族群分類的規則下，會以 CPSO-S 做為最佳化過程中所使用的演算法，但如此一來，將所有維度分到不同的族群裡，完全不考

慮維度間是否具有關連性，會造成計算量龐大的問題，且對關聯維度來說並不理想，雖然仍可比 PSO 較有效達到找尋最佳解的目的，但卻會因為沉重的計算量而使計算時間相當冗長。透過本論文所提出的方法，將關聯維度找出並置於同一族群裡，可有效的使族群數目減少，在計算量上，可達到縮小的效果，使執行速度上可以更加快速，另外在最佳化過程中，也可避免關聯維度被分到不同族群所造成不好的影響，因此不僅在演化計算量上，以及最佳解的找尋，本論文都可達到較不錯的表現。

未來希望可透過關聯法則，進行設定參數的調整，使參數不需經由一連串的錯誤嘗試法來決定。

參考文獻

[1] R. C. Eberhart and J. Kennedy, “A new optimizer using particle swarm theory,” in *Proc. 6th Int. Symp. Micro Machine and Human Science*, pp. 39-43, 1995.

[2] R. C. Eberhart and J. Kennedy, “Particle swarm optimization,” *Neural Networks*, vol. 4, pp. 1942-1948, 1995.

[3] Y. Shi and R. C. Eberhart, “Empirical study of particle swarm optimization,” in *Proc. Congr. Evolutionary Computation*, pp. 1945-1949, 1999.

[4] Y. Shi and R. C. Eberhart, “A modified particle swarm optimizer,” in *Proc. IEEE Int. Conf. Evolutionary Computation*, pp. 69-73, 1998.

[5] A. P. Engelbrecht and A. Ismail, “Training product unit neural network,” *Stability Control: Theory Appl.*, vol. 2, no. 1-2, pp. 59-74, 1999.

[6] F. van den Bergh and A. P. Engelbrecht, “Cooperative learning in neural networks using particle swarm optimizers,” *South African Computation J.*, vol. 26, pp. 84-90, 2000.

[7] R. C. Eberhart and X. Hu, “Human tremor analysis using particle swarm optimization,” in *Proc. Congr. Evolutionary Computation*, pp. 1927-1930, 1999.

[8] R. C. Eberhart, P. Simpson, and R. Dobbins, *Computational Intelligence PC Tools*, San Diego: Academic Press Professional, ch. 6, pp. 212-226, 1996.

[9] F. van den Bergh and A. P. Engelbrecht, “A cooperative approach to particle swarm optimization,” *Evolutionary Computation*, vol. 8, pp. 225-239, 2004.

[10] P. J. Angeline, “Using selection to improve particle swarm optimization,” in *Proc. Int. Joint Conf. on Neural Networks*, pp. 84-89, 1999.

[11] M. Lovbjerg, T. K. Rasmussen, and T. Krink, “Hybrid particle swarm optimizer with breeding and subpopulations,” in *Proc. Genetic and Evolutionary Computation Conf.*, pp. 469-476, 2001.

[12] J. Han, J. Pei, and Y. Yin, “Mining frequent patterns without candidate generation,” in *Proc. ACM Special Interest Group on Management of Data*, pp. 1-12, 2000.

[13] J. Han, R. Mao, J. Pei, and Y. Yin, “Mining frequent patterns without candidate generation: a frequent-pattern tree approach,” *Data mining and knowledge discovery*, vol. 8, pp. 53-87, 2004.

[14] R. Agrawal, T. Imielinski, and A. Swami, “Mining association rules between sets

of items in large databases," in *Proc. ACM Special Interest Group on Management of Data*, pp. 207-216, 1993.

[15] R. Agrawal and R. Srikant, "Fast algorithm for mining association rules," in *Proc. Very Large Data Bases*, pp. 487-499, 1994.

[16] J. H. Holland, *Adaptation in Natural and Artificial System*, Cambridge: MIT Press, 1992.

[17] P. Angeline, "Evolutionary optimization versus particle swarm optimization: Philosophy and performance differences," in *Evolutionary Programming VII*, pp. 601-610, 1998.

[18] J. Kennedy, "The particle swarm: social adaptation of knowledge," in *Proc. Int. Conf. Evolutionary Computation*, pp. 303-308, 1997.

[19] J. Kennedy, "Methods of agreement: inference among the elementals," in *Proc. IEEE Int. Symp. Intelligent Control*, pp. 883-887, 1998.

[20] Y. Shi and R. C. Eberhart, "Parameter selection in particle swarm adaptation," in *Evolutionary Programming VII*, pp. 591-600, 1997.

[21] M. Clerc, "The swarm and the aueen: toward a deterministic and adaptive particle swarm optimization," in *Proc. Int. Commodities Export Corporation*, pp. 1951-1957, 1999.

[22] D. Corne, M. Dorigo, and F. Glover, *New ideas in optimization*, New York: McGraw-Hill, ch. 25, pp. 379-387, 1999.

[23] M. Clerc and J. Kennedy, "The particle swarm: explosion, stability, and convergence in a multi-dimensional complex space," *IEEE Trans. Evolutionary Computation*, vol. 6, no. 1, pp. 58-73, 2002.

[24] Y. Shi and R. C. Eberhart, "Comparing interia weights and constriction factors in particle swarm optimization," in *Proc. Congr. Evolutionary Computing*, pp. 84-89, 2000.

[25] J. Kennedy, "Stereotyping: improve particle swarm performance with cluster analysis," in *Proc. Congr. Evolutionary Computing*, pp. 1507-1512, 2000.

[26] H. G. Cobb, "Is the genetic algorithm a cooperative learner?," in *Foundations of Genetic Algorithms 2*., pp. 277-296, 1992.

[27] S. H. Clearwater, T. Hogg, and B. A. Huberman, "Cooperative problem solving," in *Computation: The Micro and Macro View*, pp. 277-296, 1992.

[28] P. Cabena, and P. Hadjnian, *Discovering Data Mining*, Upper Saddle River: Prentice Hall, 1998.

[29] J. Han, and M. Kamber, *Data Mining: Concepts and Techniques*, San Francisco:

Elsevier, 2001.

[30] J. Pei and J. Han, “Can we push more constraints into frequent pattern mining?,” in *Proc. Int. Conf. Knowledge Discovery and Data Mining*, pp. 350-354, 2000.

[31] S. Park, W. W. Chu, J. Yoon, and C. Hsu, “Efficient searches for different lengths of FP-growth algorithm,” in *Proc. 4th Int. Conf. Information and Knowledge Management*, pp. 31-36, 2003.

[32] A. Russell and R. S. Probablistic, “Reasoning about data and hash-based algorithm for mining association rules,” in *Proc. ACM Special Interest Group on Management of Data*, pp. 245-263, 2002.

[33] T. Takagi and M. Sugeno, “Fuzzy identification of systems and its applications to modeling and control,” *IEEE Trans. System, Man, and Cybern.*, vol. 15, no. 1, pp. 116-132, 1985.

[34] M. Sugeno and G. T. Kang, “Structure identification of fuzzy model,” *Fuzzy Sets System*, vol. 28, pp. 15-33, 1998.