整合粒子群體演算法與單體法求解多目標問題之研究

何怡偉[1]、高一統[2]、黃冠勳[3]
[1]聖約翰科技大學行銷與流通管理學系
[1]E-mail: erwi@mail.sju.edu.tw
[2]大同大學資訊工程學系
[2]E-mail: kao@cse.ttu.edu.tw
[3]大同大學資訊工程碩士班
[3]E-mail: cjejvmp1@gmail.com.tw

摘要

在企業管理中經常會遇到多種目標需要決策的問題，例如：公司在作生產計劃時，不單單只需考慮利潤問題，同時也要考慮生產力、產品品質、設備利用率等。本研究整合 Nelder-Mead 單體法與粒子群體最佳化演算法的優點，發展出整合粒子群體最佳化演算法與 Nelder-Mead 單體法來解決多目標問題，並運用資料暫存器，保留每次演算過程中所找到的 Pareto 解。由文獻中擷取兩題多目標無限制式問題進行求解，並將測試實驗結果與其他文獻資料比較。由實驗結果顯示本研究方法優於其他演算法，因此證明了本研究演算法能有效的處理多目標最佳化問題並且能快速收斂並取得最佳 Pareto 曲線。

關鍵詞：粒子群體最佳化演算法、Nelder-Mead 單體法、多目標、資料暫存器、Pareto 曲線

1. 前言

多目標最佳化問題的數學模型：

$$\text{Min/Max } f_m(x)=\{f_1(x), f_2(x),\ldots, f_M(x)\} \quad (1)$$

$$\text{s.t.} \quad g_i(x)\le 0, \quad i=1,2,\ldots,I$$

$$h_j(x)=0, \quad j=1,2,\ldots,J$$

$$x_k^{low}\le x_k\le x_k^{up} \quad k=1,2,\ldots,n$$

其中多目標最佳化問題中會有 k 個目標函數問題 $f_m(x)$；而 $g_i(x)$ 代表不等式限制條件；$h_j(x)$ 代表等式限制條件；x^{low} 和 x^{up} 代表變數 x 的上限與下限。

多目標規劃的概念源於1961年由美國數學家 Charnes 和 Cooper 首先提出[1]，在解決多目標最佳化問題前，不像處理單一目標最佳化問題時，只要尋求最好或最接近的解，就可以達成最佳化目的。因為目標函數不只一個，絕大多數的目標問題之間往往都是互相矛盾的，所以在尋求多目標最佳解時，往往會在一個範圍內全都是適合的解，而這些解就看決策者如何去做決策。至於多目標問題的求解有很多種方法，如早期使用模擬退火法[2]、基因演算法(genetic algorithms, GA) [3]等求解多目標問題，經過不斷的發展到近年最常求解多目標的方法有粒子群體演算法(particle swarm optimization, PSO)及改良過後的基因演算法 NSGA-II 等[4][5][6]。

Nelder-Mead (NM)單體法為區域搜尋的技術，而 PSO 演算法為全域搜尋技術，若有辦法整合此兩種演算法，並能取得互補效果，因此本研究整合粒子群體最佳化演算法與 Nelder-Mead 單體法稱為 NM-PSO 演算法，設計出能應對多目標最佳化問題之演算法，並兼顧準確性與效率。本研究使用兩個標準測試模型去驗證 NM-PSO 的表現，並與其他演算法做比較。

2. 文獻探討

2.1. 多目標問題的支配(Dominated)情況

求解多目標最佳化問題，因為問題不只限於一種，在目標函數解出來的所有可行解中，會發現有些目標是互相矛盾、拉扯的，至於如何在所有可行解中找出讓各目標間都能符合的最佳解，就需要把各點的解互相作比較，而會有支配(Dominated)的情況出現，如圖1所示，有二個目標函數 f_1 要極大化、f_2 要極小化，可以看到在可行解區域(Feasible Region)點 A 所有的目標函數值都比點 D 好，這時就可稱 A 支配 D。經過不斷的去搜尋評估可行解，會發現在可以解區域上都沒有任何一點會比 A 來得好，至少都會有一個目標函數值優於對方。例如圖2.1可以看到 A、B、C 三點都會有一個目標函數值比對方好，這些點就可稱為非支配或互不支配(Non-dominated)的情形，而這些非支配的解會在可行解中會形成一個曲段，這曲段就稱為 Pareto 最佳解集合(Pareto Front)[7]，如圖1粗黑線的部分。這些 Pareto 最佳解集合內所有的點也是多目標最佳化所要尋求的解集合，這時可由決策者來決定取用哪一組目標解。

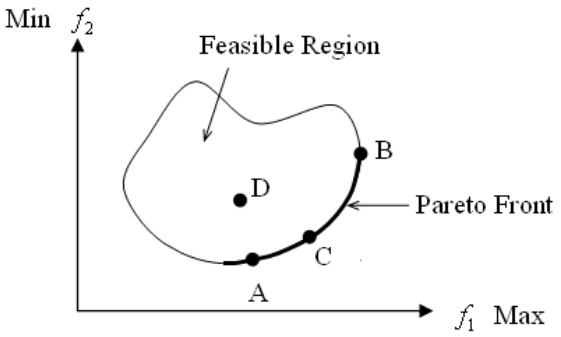


圖 1. 雙目標 Pareto 最佳解集合示意圖

2.2. 資料暫存器(Archive Controller, ARC)

此方法由 Coello *et al* [8]所提出，步驟是將上一步驟所得到的非支配解作資料儲存的動作。首先會產生一個控制機制 ARC，一開始這控制機制為空集合(ARC=ϕ)，可想像為一個空的容器。透過每次 PSO 演算法的更新，慢慢地去搜尋目標解，篩選出非支配解存入此控制機制，並做儲存的控制。一開始當判斷是為非支配解就可以儲存至 ARC 暫存器，此時 ARC 暫存器裡就會有第一筆的資料，接下來往後如果又有目標解進入 ARC 暫存器，就會發生以下三種情形：

1. 如果判斷與 ARC 暫存器的資料皆為非支配解情況，新進入的解就可以直接儲存至 ARC 暫存器裡。
2. 如果判斷新進入的解為被支配解的情況下，此筆資料直接就不做儲存動作。
3. 如果判斷在 ARC 暫存器的資料裡，某些資料是被新進入的解支配的話，此時就必須把 ARC 暫存器裡的被支配的解作刪除的動作。

2.3. Nelder-Mead 單體法

Nelder-Mead 單體法[9]由 Nelder 和 Mead 於 1965 所改良單體法在搜尋時能藉由群體內的資訊往最佳解的方向移動，因此如果它選擇的路徑是正確的，便能將最佳解包圍起來，一旦包圍住便能達到快速收斂的效果。不過如果遇到的問題區域最佳解太多的話，它很難找到真正的最佳解，且很容易陷入區域最佳解，這也是他最大的缺點。 其原理是比較在各點經由運算逐漸向最佳點逼近的一種演算法。Nelder-Mend 單體法在一開始時必須將各點代入評估函數，找出最佳、次佳、最差的目標函數後，經過反射、擴張、收縮、逼近，會產生新的點，在重複上述的步驟後，當各點十分逼近函數值時則達到收斂條件，演化過程便終止。

2.4. 粒子群體演算法

粒子群體最佳化演算法(Particle Swarm Optimization, PSO)，於1995年由 James Kennedy 和 Russell Eberhart 所提出[10]，主要概念起源於群體行為之研究，在觀察鳥群或魚群之間的移動，發現這些群體都能朝著同一方向與目標前進，這種特別的行進方式是透過個體間互相傳遞資訊，使其在移動中不會迷失路途，最終達到目的地。PSO 就像模仿這些群體生物行為，讓每個粒子都像這些生物行為一樣，慢慢地逼近目標，來尋求目標最佳解。

PSO 主要精神就是要讓每個粒子可以交換過去的經驗，使每個粒子都可以獨立去搜尋最佳解出來，再記錄每個粒子的個別最佳解，依照這些粒子找出來的個別最佳解而去修正下一次搜尋的粒子速度與位子，這正是所謂的「認知模式」；若每個粒子只依照群體最佳解來修正下一次的粒子速度與位子，就是所謂的「社會模式」。而在綜合上述兩種模式中所用的數學模型，便是合併了社會模式及認知模式的綜合模式，意即粒子會同時參考群體以及個體的最佳解來修正下次的搜尋方向及速度，並且可以經由參數的設定，改變兩種模式的重視程度。

3. 研究方法

本章節將探討 NM-PSO 演算法如何應用在多目標問題。在多目標的問題上，因為需要考慮各個目標解可能會發生相互衝突的情況，所以會產生許多組 Pareto 最佳解。本章節有三個重點，第一是介紹達到增加搜尋密度之目的分割空間方法；第二重點是對 ARC 內的解作 NM 搜尋，將 simplex 的附近作更精密的搜索，得到更多的解；第三重點是限定粒子的搜尋範圍，讓粒子不會陷在邊界、且讓搜尋範圍更均勻。

以下將詳細說明 NM-PSO 演算法應用於多目標最佳化流程(或詳看圖2)：

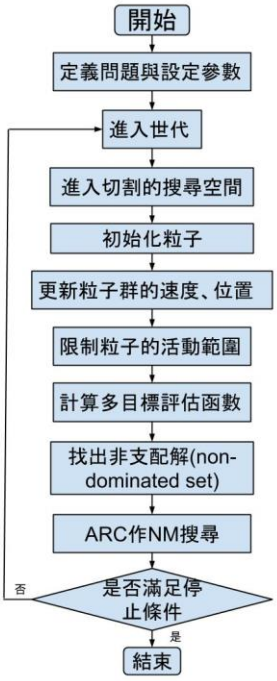


圖 2. NM-PSO 演算法的流程圖

3.1. NM-PSO 的求解程序

以下將詳細說明本論文在 NM-PSO 演算法求解多目標最佳化的流程，以及介紹資料暫存器(Archive Controller)，說明此機能如何保存 Pareto 最佳解的情況。本論文分為七項步驟，如下所示：

1. 定義多目標問題
   選定多個評估函數皆為無限制式問題，評估函數包含目標函數及決定所要取的粒子維度與組數，並且設定多目標問題及 NM-PSO 演算法之各參數以及停止條件。
2. 隨機產生各粒子初始值
   此步驟將隨機產生 i 組粒子的初始值，當起始粒子產生如下

$$x^0=\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1j} \\ a_{21} & a_{22} & \cdots & a_{2j} \\ \vdots & \vdots & \ddots & \vdots \\ a_{i1} & a_{i2} & \cdots & a_{ij} \end{bmatrix}=\begin{bmatrix} 0 & 0 \\ 1 & 1 \\ 2 & 2 \end{bmatrix}$$

   為3×2的矩陣，其中各a_{ij}代表各x值在矩陣的位置，i列代表組數，j行代表目標函數的維度。要產生幾組解出來做運算，使用者則可以自己定義。
3. 分割搜尋空間
   每一個測試函數都會在一定的範圍內尋找非支配解，本演算法會將搜尋空間依據第一維度作切割，將搜尋的範圍變小，使得粒子的搜尋密度變大，此空間粒子會更細膩地作搜尋。
4. 計算多目標評估函數
   把上一個步驟的起始x^0代入到目標函數f_m，假設目標函數f_m有二個f_1與f_2是雙目標的形式，則f_m值就會產生出與x^0對應的數值出來。如下所示：

$$x^0=\begin{bmatrix} 0 & 0 \\ 1 & 1 \\ 2 & 2 \end{bmatrix} \Rightarrow f_m=\begin{bmatrix} 0 & 2 \\ 2 & 2 \\ 1 & 4 \end{bmatrix}$$

5. 找出非支配解(Non-dominated Set)
   計算多目標評估函數後，將判斷各組f_m值是否被支配的，如果 A 的目標函數值全部都比 B 還要佳，就稱 A 支配 B。而被支配的解 B 就不作保留，把 A 篩選出來，之後再去跟其他各組解作比較是否支配，只要被支配就不作保留，而不被支配的解就保留起來。經過篩選過後，可能保留的不只一組解，因為只要其中一個目標函數值優於對方，就無法做比較為互相矛盾的情況，所以可能會產生不只一組解，這些保留下來的解就稱為非支配解(Non-dominated Set)，此步驟將被支配的解都予以剔除，只會留下非支配解。
6. 資料暫存器(Archive Controller, ARC)
   此步驟是將上一步驟所得到的非支配解作資料儲存的動作。首先會產生一個控制機制 ARC，一開始這控制機制為空集合(ARC=ϕ)，可想像為一個空的容器。透過每次 NM-PSO 演算法的更新，慢慢地去搜尋目標解，篩選出非支配解存入此控制機制，並做儲存的控制。經過不斷的反覆搜尋出目標解，只將好的資料儲存至 ARC 暫存器，而將差的資料做刪除動作，最後在 ARC 的所有資料就是多目標所求的 Pareto 最佳解集合(Pareto Front)。
7. 對 ARC 內的解作 Nelder-Mead 搜尋
   在現有解的附近也可能是最佳解的假設下，將 ARC 內的最佳解先作位置的排序，依序的3個最佳解作為 simplex，並對此 simplex 的區域附近作搜尋，增加 Archive Controller 內的精準度和最佳解的數目。
8. 運用 PSO 演算法進行更新
   計算出每個粒子的個體與群體最佳位置，代入本研究所使用的 PSO 演算法公式(2)、(3)，得到每個粒子新的移動位移量，並透過某些機制改變每個粒子的位置，此部分關係到一些處理多目標問題機制與 PSO 參數設定，將保留到下一節再作詳細說明。

$$V_{id}^{New}=w\times V_{id}^{old}+C_1\times rand_1\times(p_{id}-X_{id}^{old})+C_2\times rand_2\times(p_{gd}-X_{id}^{old}) \quad (2)$$

$$X_{id}^{New}=X_{id}^{old}+V_{id}^{New} \quad (3)$$

   其中符號定義為：
   i：粒子的編號
   d：維度
   V_{id}^{new}：透過公式(2)運算所得到的新的位移量
   w：慣性權重值
   V_{id}^{old}：目前的位移量
   $rand_1$、$rand_2$：介於0到1之間的隨機變數
   C_1、C_2：為2的常數
   p_{id}：為個體粒子中得到的區域最佳解
   p_{gb}：全體粒子中達到的全域最佳解
   X_{id}^{old}：一開始粒子所在的位置
   X_{id}^{new}：加上新的位移量(V_{id}^{new})所得到的新位置
9. 限定粒子的搜尋範圍
   由第3點可知道搜尋空間是有限制的，避免因為搜尋密度太大而使結果變差，每一個跌代的粒子群經由 PSO 公式更新其速度和位置後，會依據新的機制將粒子更新後的位置限制在該搜尋範圍內。
10. 停止條件設定
    以上步驟便是 NM-PSO 演算法應用於多目標的程序，並檢視是否達到實驗停止條件，若其目標函數值已達收斂，且實驗所設計的迭代次數已達到，則停止實驗並取出最終資料儲存控制(Archive Controller)的 Pareto 最佳解當作此次多目標的最佳化解集合，否則將回到步驟3繼續重新演算。

3.2. NM-PSO 與 PSO 不同的地方

依據3.1節流程說明可知，NM-PSO 與 PSO 不同的地方為新增了第3步驟會分割搜尋的空間、第7步對 ARC 內作 NM 搜尋、第9步對粒子的搜尋範圍作限制3種方法，以下將詳細說明較原 PSO 不同之處。

1. 分割搜尋空間
   為了減少搜尋空間、增加搜尋的密度，讓每個區域被搜尋的機率增加，本演算法將搜尋空間分為多個，在每個被分割的搜尋空間內會作固定次數的搜尋，即更新 PSO 的速度、位置，本演算法將搜尋過全部被分割的空間視為一次的世代，依據世代數、分割區域數、搜尋次數的增加，得到的 Archive Controller 內的非支配解(Non-dominated Set)也會更精確，數量也會更多，即 Pareto Front 會更為清晰。
2. 對 ARC 內的解作 Nelder-Mead Search
   我們將 ARC 內的最佳解附近的區域認定為食物豐富的區域，即該區的附近即有可能找到更多的最佳解，欲達此目的，首先將這些在 ARC 內的可能解作為單體搜尋的 vertex，我們將一個 simplex 視為一個三角形，在排序後 ARC 的連續 3 個點為三角形的 vertex，每次依據 NM 搜尋的行為：reflection、expansion、contraction、shrinkage，對該 simplex 附近區域作搜尋、並塑造出下一個 simplex，每次的 simplex 的形狀皆可能會變化，simplex 的變化方向即是往解多的地方的方向，並將每次搜尋到新的解經步驟 5 的判斷機制後儲存於 Archive Controller 內。
3. 限定粒子的搜尋範圍
   每一個跌代的粒子群經由 PSO 公式更新其速度和位置後，粒子群的位置有可能超過搜尋空間的邊界、若粒子在邊界作搜尋，也可能會因為原本的邊界限制而陷在邊界附近的搜尋空間不斷的打轉，本演算法使用一個新的機制將粒子的位置限制於搜尋空間內、卻又可以廣泛的搜尋。本演算法依據「地圓說」的原理，地球是圓的，即沿著緯線走一圈會回到原點。本演算法採取的方法是:若粒子更新後的位置超過最大邊界，粒子會由最小邊界為起點，並前進超過原邊界的距離，超過最小邊界的情況則反之。

4. 實驗結果與分析

本研究的核心，主要是以 PSO 演算法來處理多目標最佳化問題，從 Deb et al., 2002年文獻中[5]，擷取研究多目標最佳化的標準方程式，主要為求極小值的最佳化問題，本研究 PSO 演算法會用 Matlab 程式語言[11]來撰寫。除了測試實驗題形，也利用效能量測指標去評估本研究演算法的誤差率、分布性、延展性等，並且將本研究所得出的結果與余宗祐論文[12]所列的結果做比較。最後本論文會做實際的案例分析，看看本研究演算法運用在實例上，能否求得出目標之解出來，以下先介紹本研究所使用的各種效能量測指標。

4.1. 量測方法

1. 與真正 Pareto 解的誤差值(Generational Distance, GD)
   此量測方法是由 Van Veldhuizen 所提出的[13]，功用是將正確的 Pareto 最佳解與求得出的所有解，評估兩者間距離誤差值 d_i，數值越接近0代表與真正的 Pareto 最佳解越相同，此方法運用公式(4)是將求出的每一個解找出與真正的 Pareto 最佳解之最近的距離，再將每個解的差距加總取平均，就得到演算法求得解的誤差值(GD)。

$$\text{GD}=\left(\frac{1}{n}\sum_{i=1}^{n}d_i^2\right)^{1/2} \tag{4}$$

2. 最大延展度(Maximum Spread, MS)

   此方法 K. C. Tan 文獻所提出的[14]，由最大延展度指標可以看出演算法所求的邊界與正確的 Pareto 最佳解的邊界，兩者邊界的差異性比例，比例越接近1說明與正確的 Pareto 最佳解邊界越相似，代表著演算法有很好的搜尋廣度。計算方式為以下公式(5)。

$$\text{MS}=\sqrt{\frac{1}{M}\sum_{i=1}^{M}\left[\frac{\min(f_i^{\max},F_i^{\max})-\max(f_i^{\min},F_i^{\min})}{F_i^{\max}-F_i^{\min}}\right]^2} \tag{5}$$

3. Pareto 解的分布性(Spacing, S)

   除了上述兩種方法外，也要看 Pareto 解的分布性，本論文使用 Schott, J. R. [15]提出公式(6)量測出每個 Pareto 解的分布，方法為將求得出的所有解，經過排序後，計算出每個點與最近距離點的差距 d_i'，算出平均 $\bar{d}'$ 後取標準差就為分布性的評估指標，指標越小代表解與解之間的距離越短，也就是求得出的 Pareto 解密度越高。

$$S=\left[\frac{1}{n}\sum_{i=1}^{n}(d_i'-\bar{d}')^2\right]^{1/2}/\bar{d}',\quad \bar{d}'=\frac{1}{n}\sum_{i=1}^{n}d_i' \tag{6}$$

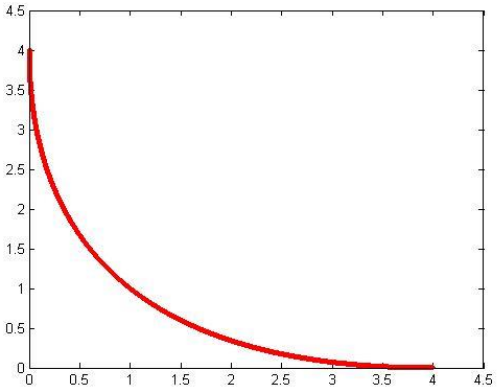
4.2. 標準測試方程式及量測指標數據

Deb et al., 2002年文獻中[5]擷取兩題方程式測試(可見表1)，題目分為 SCH、FON 為本研究演算法計算各標準測試方程式，其中題形包括1及3維度為極小化問題。

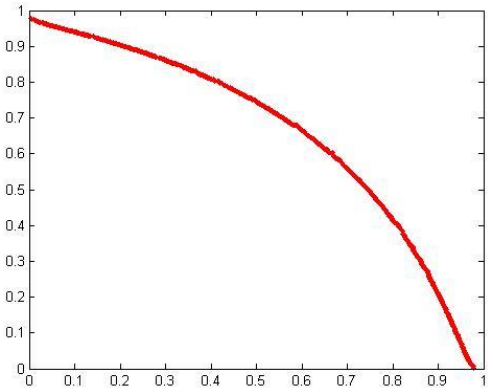
文獻中方程式 SCH 為1維度的題目，搜尋範圍設定在$[-10^3,10^3]$；FON 為3維度的題目搜尋範圍設定在$[-4, 4]$。NM-PSO 設定演化次數為4次，粒子數為400個、分割區域為4個，由圖4跑出來的圖形都有找到一條 Pareto 曲線，與理論的 Pareto 曲線一樣，其中 NM-PSO 求解 SCH 時大約找到14960解，求 FON 時大約找到3995解，明顯優於余宗祐論文所得的結果分別為600與700個解。

表 1. 標準測試方程式問題

題號	n	多目標標準測試題形
SCH	1	$f_1(x)=x^2$ $f_2(x)=(x-2)^2$
FON	3	$f_1(x)=1-\exp\left(-\sum_{i=1}^{3}\left(x_i-\frac{1}{\sqrt{3}}\right)^2\right)$ $f_2(x)=1-\exp\left(-\sum_{i=1}^{3}\left(x_i+\frac{1}{\sqrt{3}}\right)^2\right)$



(a) SCH



(b) FON

圖 3. NM-PSO 測試模型 Pareto 解

而要如何判斷是好或壞的 Pareto 解，就要用以下整理出的量測指標數據進行判斷，可見表2與表3。由表2與表3可知 NM-PSO 處理 SCH 結果在 S 與 MS 指標優於 PSO 解，而在 FON 結果 NM-PSO 在 GD 與 S 指標優於 PSO，由此可知本研究 NM-PSO 演算法可有效處理無限制式多目標的問題。

表 2. SCH 效能評估指標

SCH		NM-PSO	PSO
GD	Mean	0.0037	**9.81E-04**
	Median	0.0037	**9.88E-04**
	Min	0.0036	**8.79E-04**
	Max	0.0037	**0.0011**
	Std	6.67E-05	**6.49E-05**
S	Mean	**0.0053**	0.3177
	Median	**0.0052**	0.3153
	Min	**0**	0.3829
	Max	**0.0096**	0.3875
	Std	**0.0035**	0.1727
MS	Mean	**0.9998**	0.9991
	Median	**0.9999**	0.9988
	Min	**0.9991**	0.9973
	Max	**1**	0.9999
	Std	**3.84E-04**	9.65E-04

表 3. FON 效能評估指標

FON		NM-PSO	PSO
GD	Mean	**0.0011**	0.0012
	Median	**0.0011**	0.0011
	Min	**1.00E-03**	0.001
	Max	**0.0013**	0.0013
	Std	**9.92E-05**	1.09E-04
S	Mean	**0.0118**	0.3177
	Median	**0.0049**	0.3153
	Min	**0**	0.3829
	Max	**0.0118**	0.3875
	Std	**0.0049**	0.1727
MS	Mean	0.9975	**0.9991**
	Median	0.9978	**0.9988**
	Min	0.9954	**0.9973**
	Max	0.9993	**0.9999**
	Std	1.40E-03	**9.65E-04**

5. 結論

本研究整合 Nelder-Mead 單體法與粒子群體最佳化演算法的優點，發展出整合粒子群體最佳化演算法與 Nelder-Mead 單體法來解決多目標問題，並運用搜尋區域的分割、邊界的限制來找到更多的解，由實驗結果可即清楚的得到一條 Pareto Front。此外，在粒子的速度限制、搜尋空間的分割、PSO 參數的設定、Nelder-Mead 搜尋的參數設定等仍需要多加研究，才可以達到更好的效果。

6. 參考文獻

[1] A. Charnes and W. W. Cooper. Management models and industrial applications of linear programming. *Wiley, New York, 1961*.

[2] P. Czyzak and A. Jaszkiewicz. Pareto simulated annealing—a metaheuristic technique for multiple-objective combinatorial optimization. *Journal of Multi-Criteria Decision Analysis, 1998*.

[3] C. M. Fonseca and P. J. Fleming. Genetic algorithms for multiobjective optimization: formulation, discussion and generalization. *Forrest, 1993*.

[4] D. Liu, K. C. Tan, C. K. Goh and W. K. Ho. A multiobjective memetic algorithm based on particle swarm optimization. *IEEE Transaction on Systems, Man and Cybernetics-Part B, 2007*.

[5] K. Deb, A. Pratap, S. Agarwal and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transaction on Evolutionary Computation, 2002*.

[6] R. C. Min and Z. L. Young. A novel elitist multiobjective optimization algorithm: multiobjective extremal optimization. *Optimization Online Digest, 2007*.

[7] D. Kalyanmoy. Multi-objective optimization using evolutionary algorithms. *John Wiley & Sons, NewYork , 2002*.

[8] C. A. C. Coello, G. T. Pulido and M. S. Lechuga. Handling multiple objectives with particle swarm optimization. *IEEE Transaction on Evolutionary Computation, 2004*.

[9] J. A. Nelder and R. Mead. A simplex method for function minimization. *Computer Journal, 1965*.

[10] J. Kennedy and R. C. Eberhart. Particle swarm optimization. *IEEE International Conference on Neural Networks, 1995*.

[11] B. Adrian and B. Moshe. Matlab 5 for engineers. *Prentice Hall, England, 1999.*

[12] 余宗祐. 運用粒子群體演算法求多目標問題之解. *聖約翰科技大學工業工程與管理碩士論文, 2010.*

[13] D. A. Van Veldhuizen. Multiobjective evolutionary algorithms: classifications, analyzes, and new innovations. *Wright-Patterson Air Force Base, 1999*.

[14] K. C. Tan, Y. J. Yang and C. K. Goh. A distributed cooperative coevolutionary algorithm for multiobjective optimization. *IEEE Transactions on Evolutionary Computation, 2006*.

[15] J. R. Schott. Fault tolerant design using single and multicriteria genetic algorithm optimization. *Massachusetts institute of technology, Cambridge, 1995.*