以一個有效的螞蟻演算法解決工作排程問題

蔡正發　楊澤　賈景軒
國立屏東科技大學資訊管理系
cftsai@mail.npust.edu.tw

摘要

智慧型啟發式演算法應用在解決NP-Hard的問題中，包含了無線通訊網路(Wireless Communication Network)、資料探勘(Data Mining)、多媒體傳輸、旅行推銷員問題(Traveling Salesman Problem；TSP)、資源分配(Resource Allocation)及排程問題(Scheduling Problem)…等。其中在排程問題裡，最常被學者提出討論的即工作排程問題(Job-Shop Scheduling Problem；JSP)。有不少學者嘗試將智慧型演算法，應用在解決工作排程的問題上，例如1994年Colorni等人成功的將螞蟻系統(Ant System; AS)運用在工作排程問題上，當時所得到的結果，相較於最佳解(Global Optimum)仍有10%的誤差。因此本研究提出一個新的螞蟻系統演算法以解決工作排程問題的架構，經由實際電腦模擬結果得知，本系統確能更有效的解決工作排程問題。

關鍵詞：螞蟻系統、工作排程、基因演算法

An Effective Job-Shop Scheduling Method by a New Ant Algorithm

Cheng-Fa Tsai, Tzer Yang, Ging-Shing Chia

Department of Management Information Systems, National Pingtung University of Science and Technology, Pingtung, Taiwan

cftsai@mail.npust.edu.tw

Abstract

The NP-hard problems to be solved using intelligent heuristic approaches may include traveling salesman problem, resource allocation problem, data mining, wireless communication network, scheduling problem, and so forth. The most widely studied and increasingly important issue in scheduling problem is the job-shop scheduling problem (JSP). Therefore, many efficient intelligent approaches to solve the job-shop scheduling problem are proposed. In this paper, we present a new and effective algorithm based on ant system for solving the job-shop scheduling problem. According to our simulation results, our proposed method outperforms some already existing approaches.

Keywords: Ant system, job-shop scheduling, genetic algorithm

一、 前言

工作排程(JSP)為近年來學者熱門討論議題之一，JSP問題已經由M. Kolonko證明為NP-Hand的問題[9]，工作排程(JSP)問題包含了m個不同功能的機器，並分別可以完成n個工作中所需的排程表。解決NP-Hand問題的方法，最常見的是智慧型啟發式演算法，其中螞蟻系統是近年來發展出來的智慧型啟發式演算法之一，1991年正式由Marco Dorigo提出，當Dorigo將其運用於解決TSP之後[1]-[4]，已證明螞蟻系統確實是一非常有效率的方法；螞蟻系統的觀念來自於螞蟻運送食物之行為[7]，螞蟻運送食物會留下費洛蒙(pheromone)，之後的蟻蟻會依循先前螞蟻所留下之費洛蒙的濃度行走，因此路徑上的費洛蒙濃度較濃，表示該路徑行走的螞蟻較多。

1996 年 Dorigo 所發表的文章中，他又將費洛蒙更新方法又分成三類[5]：1).Ant-density、2).Ant Q、3).Ant-cycle。文章中說明了 Ant-cycle 的費洛蒙更新方式，比上述其他兩種能獲得較佳的解；接著 Dorigo 等人試著把螞蟻系統應用在 Quadratic Assignment Problem 上[6]；1994 年 Colorni 等人亦成功的把螞蟻系統運用在工作排程問題上[2]，印証了螞蟻系統對於決解工作排程問題的效能頗為理想。

本研究中提出數項策略以提升螞蟻系統的求解能力，經由實際的電腦模擬結果，證明本研究所提出的新方法，比先前學者所提出的方法更具成效。

二、問題定義

工作排程(JSP) 可描述為：一個工作排程(Job Shop)[10]，包含了 m 個不同功能的機器(M={M1,…..Mm})，分別可以完成 n 個工作(J={J1,…...Jn})中所需的某特定步驟，每一個工作皆由一連續的特定作業程序所構成，JSP 就是要將每個工作步驟要在什麼時候，放入哪一台機器的排程表放置排定，所以每一個工作都有不同機器流程及機器執行時間對應。常見的 JSP 限制條件如下所述[8]：

A. 任一工作不能中斷
B. 每一台機器在同一時間只能處理一件工作
C. 每一個工作的作業順序不能被違反
D. 同一工作在一台機器只能使用一次
E. 不同工作間其作業並沒有優先次序的限制

JSP 問題是一個可以滿足上述限制式的排程問題，且能夠將時間流程控制到最短時程(makespan)，其解碼方式之後將再詳述。

三、以螞蟻系統解決工作排程問題

螞蟻系統自從由 Marco Dorigo 提出之後，目前文獻上已有許多學者將螞蟻系統應用在解決各類不同的問題上，亦有學者探討參數對問題的影響力，不同的參數設定，的確會使解的效能有所差異，接下來本論文將先討論螞蟻系統解決推銷園旅行問題(Traveling Salesman Problem; TSP) 的方法，之後再討論螞蟻系統於解決工作排程問題上的編碼/解碼方式及其主要架構，最後將詳述本文所提出的修改方法。

3.1 螞蟻系統解決 TSP 問題

螞蟻系統的演算法一般可以將其視為下列幾部分：

A. 初始化(Initialize)
B. 將螞蟻分配至各個地點
C. 每一隻螞蟻依照公式(1)所求得機率去選擇所要選擇的目的地

$$p_{ij}^{k}(t)=\begin{cases}\dfrac{[\tau_{ij}(t)]^{\alpha}\cdot[\eta_{ij}]^{\beta}}{\sum_{k\in allowed_k}[\tau_{ik}(t)]^{\alpha}\cdot[\eta_{ik}]^{\beta}} & \text{if } j\in \text{allowed}_k \\ 0 & otherwise\end{cases} \quad (1)$$

p_{ij}^{k}：第 k 隻螞蟻選擇從城市 i 到城市 j 這條路徑的機率

i，j：指的是城市，亦可將城市 i 視為目前位置，城市 j 視為目的位置

$\tau_{ij}(t)$：i 到 j 這條路徑於時間 t 時費洛蒙的強度

η_{ij}：$\eta_{ij}=1/d_{ij}$ (距離的倒數)，也就是指從 i 到 j 的這條路徑的能見度

d_{ij}：指的是從 i 城市至 j 城市這條路徑的距離

α 及 β：控制變數(α 越大代表較重視費洛蒙的比例、β 越大代表 η_{ij} (距離的倒數)較重

要)

D. 計算每一隻螞蟻所走過的路徑，並依走過路徑長短增減費洛蒙的濃度。

$$\Delta\tau_{ij}^{k}=\begin{cases}\dfrac{Q}{L_k} & \text{if}(i,j)\in \text{tour described by tabu}_k\\ 0 & \text{otherwise}\end{cases} \qquad (2)$$

Q：為一個常數，一般設為 1；在 JSP 問題中本研究設為 10

L_k：總路徑距離；在 JSP 問題中本研究當成總排程時間的長度

$$\Delta\tau_{ij}+\Delta\tau_{ij}^{k}=\sum_{k=1}^{m}\Delta\tau_{ij}^{k} \qquad (3)$$

$\Delta\tau_{ij}$:由 i 城市到 j 城市這條路徑的費洛蒙值

$\Delta\tau_{ij}^{k}$:由 i 城市到 j 城市這條路徑的第 k 隻螞蟻所釋放費洛蒙值

E. 依公式(4)更新每一條路徑上的費洛蒙濃度

$$\tau_{ij}(t+n)=p\cdot\tau_{ij}(t)+\Delta\tau_{ij} \qquad (4)$$

p:為每條路徑上，在經過一段時間後費絡蒙蒸發速度的相關係數(時間 t 到時間 $t+n$)

F. 檢查是否找到較佳的解;若有則更新，並重複步驟 B.C.D.E。

3.2 螞蟻系統的編碼方式

在螞蟻系統解決工作排程問題中，如何有效率編碼與解碼是一個重要的議題。從文獻[11]探討編碼與解空間的對應關係可知，編號對應到解空間會有三種情況，1. 可行解，2. 不可行解及 3. 非法解。文獻中曾提及 TSP 與 JSP 的比較，因為這兩個問題有許多相似之處。TSP 是一個基本的排程問題，已經被研究多年，至今尚未被發展出一套有效的解決之道[12] 。傳統的 TSP 編碼方式與 JSP 類似，但不可以直接使用，必須透過特殊的解碼才能找到解答，接下來將介紹 JSP 的編號概念。

每一隻螞蟻所走過的路徑表示一個解，類似 TSP 的唯一編碼方式，各個工作步驟使用不同的代號，在 3x3 的問題中；螞蟻經過的路徑為[471582639]，如表 1 找出每個工作的處理順序(表 3)及工作處理時間(表 4)，以建立表 5 Decoded active schedule，詳細的選擇路徑及建表方法，在後面章節將再詳述。

3.3 Ant system for JSP 的公式轉換

由於 Equation(1)是應用於 TSP 中每一隻螞蟻選擇目的地的機率，將其應於 JSP 上，其主要差異是二點距離的求法，Colorni、Dorigo、Maniezzo 及 Trubian 針對 JSP 所提出的方法如下[2]：

SPT：表是選擇最短的作業時程(select the operation with the shortest processing time)

LPT：表示選擇最長的作業時程(select the operation with the longest processing time)

SRT：選擇這項工作剩餘之最短作業時程(select the operation belonging to the job with the shortest remaining processing time)

LRT：選擇這項工作剩餘之最長作業時程(select the operation belonging to the job with the longest remaining processing time)

LRM：除了考慮作業的條件下選擇這項工作剩餘之最長時程(select the operation belonging to the job with the longest remaining processing time excluding the operation under consideration)

Colorni、Dorigo、Maniezzo 及 Trubian

在其所提出的方法中[2]，採用了 LRT (選擇這項工作剩餘之最長作業時程)的原則，經過本研究實驗的結果發現這個部份相當重要，因為它決定了螞蟻系統(AS)求解的能力，但其效果還是讓人不是很滿意，所以本研究提出一個新的方法，假設欲求 Job i 和 Job j 的距離時，首先先求得 Job i 目前的 *makespan* 大小，為 time i，經過計算得到加入 Job j 之後的 *makespan* 大小，為 time j，則其兩點距離為 time j－time i;但是若兩點距離為 0 時，則兩點距離設為一個微小值(本研究設為 1，若此值太小，容易陷入區域解)，因為 $\eta_{ij}=1/d_{ij}$，而 d_{ij}=0 時，則 $\eta_{ij}=1/0$ 會發生無窮大的問題。

四、本研究的方法

螞蟻系統用在各個領域尋求最佳解，都已有相當不錯的效率，但由於螞蟻演算法發展迄今，算是目前智慧型演算法中較新的方法。因此，將螞蟻系統應用於工作排程上的研究較少;本研究欲嘗試將螞蟻系統應用於工作排程的領域，此演算法稱之為(Ant for JSP,AJ)，AJ 演算法之目的乃是為了改良螞蟻系統運用於工作排程問題上，以增加求解的效果。本研究所提出的演算法，首先先做初始設定，給費洛蒙表初始值，接著亂數選擇螞蟻可選擇的第一個城市，之後從第二步開始，直到全部走完，最後根據螞蟻所走的路徑更新費洛蒙表的值。

```
AJ Algorithm
Procedure Ant System algorithm for JSP
      Initialization;                    //初始設定
      For i=1 to n do   // n generation
           Start-set
           Select-next-tour
             Compare-result
             Update-pheromone-table
    End-for
End Procedure

Procedure Initialization //初始設定
     For i=1 to node (node=job num * machine num)
         do
           For  j=1  to  node  (node=job  num  *
                machine num) do
                PH(i)=c //c 為費洛蒙的初始值
           End-for
     End-for
End Procedure

Procedure Start-set
     //Place the m (m is ant num) ants random to the
       node
     For i=1 to m do
       Tour(i,1)=random permit node//亂數選擇螞
       蟻可選擇的第一個城市
     End-for
End Procedure

Procedure Select-next-tour
    //Choose the town j to move to, with probability
    p_ij^k(t) given by Eq.(1)
     For k=1 to m do
        For j=2 to node do //從第二步開始，直到
            全部走完  //利用修正後公式 1 選擇
            下一步螞蟻要走的路徑
       End-for
     End-for
End Procedure

Procedure Compare-result
     Compare and save ant so far best path//計算並
        存儲目前螞蟻累積走過最短的路徑
End Procedure

Procedure Update-pheromone-table//更新費洛蒙表
   Updating ant so far best path add 0.5 for
   pheromone
End Procedure
```

4.1 區域搜尋

本研究中提出一個新的區域搜尋法(Local search; LS)，此區域搜尋方法是將所得到的解，更進一步的精煉，避免錯失找到最佳解的機會。本研究中每 10 代使用一次此策略，在此法中將前面(螞蟻總數/10)螞蟻走過的路徑取出，並對這些路徑做區域搜尋。

此方法是對所有路徑作全面性的搜尋，由此路徑第一個位置兩兩互換到最後一個位置，如果交換後解相同時，也就是計算出來的評估(fitness)值一樣時，立即做交換的動作(swapping)。而當解較佳時，將此解的兩個位置，紀錄起來，不立即做交換的動作，待路徑位置全部兩兩互換完畢後，才做交換的動作。基本上本研究的訴求在於，使用區域搜尋的方法可以將好的解留下，更新費洛蒙，可以使效

果更加顯著。如果螞蟻系統的尋找方向為最佳解的方向時，配合區域搜尋的方法將更能將接近最佳解予與搜尋出來。詳細的演算法流程如下。

```
Procedure local search
Begin
 For i=0 to m/10 do
  Evaluate this Ant_path (i)
   For j=0 to node (node=job num * machine num)
    do
    For k=0 to node (node=job num * machine num)
      do
     If (i [j]! =i [k]) // the value of position is not the
         same;
       Exchange the position of j and k temporarily;
       Evaluate the new Ant_path (n);
       If (the new Ant_path's (n) fitness is better
           than old Ant_path's (j) fitness)
       {Keep the position of j and k in memory but
           not exchange right now;}
       If (the new Ant_path's (n) fitness is the same
           with old Ant_path's (j) fitness)
         {Exchange the position of j and k
              immediately;}
     End if
    End -for
   End-for
  Exchange the position of j and k in memory;
 End-for
End Procedure
```

本方法是從所有螞蟻所走過路徑中，選擇前面百分之十的螞蟻走過的路徑，對這些路徑作區域搜尋的動作。亦即先對此螞蟻所走過的路徑(i)做評估的動作，計算其 fitness value，使用兩個 index (j 與 k)，j 用來記錄目前的位置，k 為記錄與 j 相關的鄰近位置值，j 的搜尋值為 0 到 *node* 的大小(node=job num * machine num)。當 j 固定時，k 的搜尋位置為 0 到 *node* 的大小(node=job num * machine num)，此舉的目的為，當 j 固定時，嘗試與每一個位置作交換的動作，交換的條件必須是在交換的值要不一樣時，才有意義；交換時，即評估此新的螞蟻路徑(n)。如果新的螞蟻路徑(n)較舊的螞蟻路徑(i)佳時，不立即做交換的動作，而將其所在的位置(j 與 k)值，記錄在記憶體中，等全部掃瞄過後再做交換的動作。此動作是為了使得螞蟻不會因為目前的值較佳而作交換的動作，而馬上就落入區域解，相反的會使的搜尋的方向較易找到最佳解。此外當新的螞蟻路徑(n)的 fitness 值與舊的螞蟻路徑(i)一樣時，立即作交換的動作，此目的在於使得螞蟻路徑在區域搜尋下擁有多樣性，因為相同的 fitness value 其解碼後的 *makespan*，結果是不盡相同的。

4.2 非唯一編碼區域搜尋

螞蟻系統的編碼方式是唯一編碼，對於之前區域搜尋的效能將大打折扣，因為在工作排程問題中，每一個工作的執行有先後順序，例如前面表 2 及表 3 中 J1 的工作順序為 1 2 3、J2 的工作順序為 4 5 6 、J3 的工作順序為 7 8 9，因此在表 1 路徑表中有一定的順序，如 2 一定要等 1 排完才能排 2 之後才能再排 3，所以在做區域搜尋時，有過多的限制。當執行兩點交換時，首先要判斷第一點交換後是不是會影響到其工作的執行順序，再檢查第二點是不是會違反工作排程的規則，如此一來區域搜尋效能就被唯一編碼給限制住了，無法發揮其求解能力。有鑑於此，本論文遂將唯一編碼的編碼方式，改變成非唯一編碼的方法進行求解。

五、模擬測試

本節測試 AJ 演算法的求解能力，分為以下幾部份，第一部份是 AJ 與 Colorni、Dorigo、Maniezzo 及 Trubian AS [2]之比較，實驗結果證實我們提出的 AJ，的確能改進傳統 AS 的求解能力，第二部份是 AJ 加入區域搜尋策略的效能評估，第三部份則是 AJ 與基因演算法(Genetic Algorithm; GA)的效能比較。

5.1 AJ 與 AS 之比較

此兩者採用相同的工作排程問題進行比較，第一個是由 Muth and Thompson [13]所提出的 10*10(MT10)問題，第二、三個是由 Applegate and Cook [14] 所提出的

10*10(ORB01、ORB04)問題，而第四個則是由 Lawrence [15]所提出的 10*15(LA21)問題。每個問題各跑 3000 代，而變數設定亦因方法的不同而有所不同。其中 Colorni、Dorigo、Maniezzo 及 Trubian [2]的變數設定為 α=1、β=1、p=0.7，本研究方法的變數設定為α=1、β=3、p=0.99，其比較如表６所示。表六中在 AS 的最佳解的 AS value MIN 與本研究所提出的方法比較，可以看出，本研究所提出的方法之平均解即比 AS 的最佳解好，而且更低了 2.9%~7.8%，故可證明本研究所提出的方法是更有效率的。

5.2 AJ 加區域搜尋的效能評估

在 AS 與 AS 加 LS 兩者比較方面，採用相同的四組問題設定執行 3000 代，其餘變數設定均相同(α=1、β=3、p=0.99)，其參數比較如表 7 所示。本研究所提出的 AJ 加上 LS 後，可以把 Best 與 Average 的值降得更低，故可證明本研究所提出的 LS 方法確是有效率的。

5.3 AJ 加非唯一編碼的區域搜尋與基因演算法的比較

基因演算法(Genetic algorithm; GA)是模擬大自然的運作方式，透過父代產生子代的基因挑選，交配，突變等方法，代代更新，產生更優良的解答。在 AJ 與 GA 的比較方面，使用非唯一編碼區域搜尋的 AJ，和 Croce、Tadei 及 Volta 提出的 GA [15]做比較。JSP 應用在 GA 上的研究很多，但 Croce、Tadei 及 Volta 所發表 GA 是眾多研究之中結果較好、同時數據也較豐富的。本研究採用二組測試問題，第一組是由 Muth and Thompson [13]所提出 6*6(MT06)、10*10(MT10)及 5*20(MT20)的問題，第二組是由 Lawrence [15]所提出 5*10(LA01)、5*15(LA06)、5*20(LA11)及 10*10(LA16)的問題。在 Croce、Tadei 及 Volta 所發表的論文中其 GA 參數設定如下：交配率為 1、突變率為 0.03、染色體為 300、代數設定為 2971 代、並重覆執行 5 次(RUN)。而 AJ 的參數設定為α=1、β=3、p=0.99、代數設定為 3000 代,並重覆執行 30 次(RUN)，其中螞蟻的數量設定為 Job 數乘於 machine 數，例如在 MT06 問題中 Job 數為 6 而 machine 數為 6，則螞蟻的數量為 6*6=36。其效能比較如表 8 和表 9 所示。MT06 問題中，AJ 在 160 代時平均解已經達到最佳解；LA01 問題中，AS 在 90 代平均解已經達到最佳解；LA06 問題中，AS 在 20 代平均解已經達到最佳解；LA11 問題中，AJ 在 50 代平均解已經達到最佳解。AJ 加上非唯一編碼的 LS 和 GA 相比，不論是在最佳解(MIN)和平均解(AVG)的效能都比 GA 好，證明本研究所提出 AS 加上非唯一編碼的 LS 方法是有效率的。

六、結論

AJ 演算法是利用 AS 所延伸出的一個新的演算法，此演算法具有比 AS 計算上速度較快及更佳的效能兩項特色。本論文並提出兩個有效的區域搜尋方法，第一個是單純的區域搜尋法、第二個則是非唯一編碼的區域搜尋法。將 AJ 與 Colorni 等人所提出 AS 及 Croce 等人所提出的 GA 作比較，經由實際的電腦模擬結果發現 AJ 的求解能力的確優於 AS 及 GA。

參考文獻

[1] M. Kolonko., (1999),Some new results on simulated annealing applied to the job shop scheduling problem, *European Journal of Operational Research*, pp. 123-136.

[2] Colorni A., M. Dorigo, V. Maniezzo and M. Trubian., (1994), Ant system for Job-shop Scheduling, *JORBEL - Belgian Journal of Operations Research, Statistics and Computer Science*, 34(1):39-53.

[3] Dorigo M., V. Maniezzo & A. Colorni (1991). The Ant System: An Autocatalytic

Optimizing Process. *Technical Report* No. 91-016 Revised, Politecnico di Milano, Italy.

[4] Dorigo M., V. Maniezzo & A. Colorni (1996). The Ant System: Optimization by a Colony of Cooperating Agents. *IEEE Transactions on Systems, Man, and Cybernetics*-Part B, 26(1):29-41.

[5] M.Dorigo, V.Maniezzo, and A. Colorni. (1996), The Ant System: Optimization by a Colony of Cooperating Agent. *IEEE Transaction on System, Man, and Cybernetics* – Part B,26(1):29-42.

[6] Maniezzo V., Colorni A., Dorigo M.(1994), ALGODESK: An experimental comparison of eight evolutionary heuristics applied to the QAP problem. *To appear in European Journal of Operational Research.*

[7] Denebourg J.L., Pasteels J.M., Verhaeghe J.C.(1983), Probabilistic Behaviour in Ants: a Strategy of Errors?. *Journal of Theoretical Biology* 105, 259-271.

[8] Runwei Cheng, Mitsuo Gen and Yasuhiro Tsujimura, (1996), A tutorial Survey of job-shop scheduling problem using genetic algorithm-I. *Reprentation, Computer ind. Engng* vol. 30, no. 4, pp. 983-997.

[9] M. Kolonko, (1999), Some new results on simulated annealing applied to the job shop scheduling problem," *European Journal of Operational Research*, pp. 123-136.

[10] Graham R.L., Lawler E.L., Lenstra J.K., Rinnooy Kan A.H.G (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of Discrete Mathematics*. 5, 287-326.

[11] M. Gen and R. Cheng, (1997), Genetic Algorithms and Engineering Design, *John Wiley&Sons*, New York.

[12] Koji Morikawa, Takeshi Furuhashi, Yoshiki Uchikawa, (1992), *Single Populated Genetic Algorithm and its Application to Jobshop Scheduling*, Proc. of Industrial Electronics, Control, Instrumentation, and Automation on Power Electronics and Motion Control, pp. 1014 –1019.

[13] Ling Wang, Da-Zhong Zheng,(2001) An effective hybrid optimization strategy for job-shop scheduling problems, *Computers & Operations Research* 28, pp. 585-596.

[14] Muth J.F., Thompson G.L.*Industrial Scheduling*. Englewood Cliffs, N.J., Prentice Hall 1963.

[15] Applegate D., Cook W.(1990). A computational study of the job-shop scheduling problem. *ORSA Journal on Computing*. 3, 149-156.

[16] Croce FD, Tadei R, Volta G.,(1995), A genetic algorithm for the job shop problem, Computers & Operations Research, pp. 15-24.

表 1 路徑表

4	7	1	5	8	2	6	3	9

表 2 對照表

對照表		
1	2	3
4	5	6
7	8	9

表 3 Operation Process order on each job

Job	Machine sequence		
	Machine order		
	1	2	3
J1	M1	M2	M3
J2	M1	M3	M2
J3	M2	M1	M2

表 4 Job process time

Job	Process Time		
	Process order		
	1	2	3
J1	3	3	2
J2	1	5	3
J3	3	2	3

表 5. Decoded active schedule

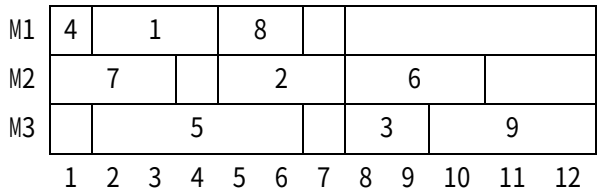

表 6　AS 與 AJ 比較表

	MT10	ORB01	ORB04	LA21
Known optimal value	930	1059	1005	1040
AJ MIN	939	1075	1045	1085
AJ AVG	972	1102	1076	1129
AS value MIN	1015	1166	1077	1177

表 7　AJ 加 LS 效能表

	MT10	ORB01	ORB04	LA21
Known optimal value	930	1059	1005	1040
LS_Best	939	1060	1030	1071
Best	939	1075	1045	1085
LS_Average	965	1099	1056	1112
Average	972	1102	1076	1129

表 8、AJ 與 GA 在第一組問題效率比較表

	MT06	MT10	MT20
Known optimal value	55	930	1165
AJ MIN	55	939	1174
AJ AVG	55	955	1184
GA_MIN	55	946	1178
GA_AVG	55	1162	1199

表 9、AJ 與 GA 在第二組問題題效率比較表

	LA01	LA06	LA11	LA16
Known optimal value	666	926	1222	945
AJ MIN	666	926	1222	957
AJ AVG	666	926	1222	973
GA_MIN	666	926	1222	979
GA_AVG	666	926	1222	989