序的最佳化於隨機工程生產排程問題之應用

洪士程
朝陽科技大學資訊工程系
e-mail：schong@cyut.edu.tw

滿冠伶
朝陽科技大學資訊工程系
e-mail：s9727627@cyut.edu.tw

摘要

隨機工作生產排程問題是等待處理的工作，操作的程序不相同，而且加工時間是隨機變動的。以實際應用的角度來看，加工時間或長或短常常超乎預期，正因為在每個機器上面工作的加工時間是隨機變動，造成在每台機器上面工作的生產順序無法事先定好，縱使順序已事先定好，然而工作的實際加工時間的改變，也會使的這些順序變成無效。為了能夠評估這項變化，我們將加工時間設為隨機變數，快速的以模擬方式求出一個足夠好的排程解。我們提出一個以序的最佳化理論為基礎的演算法，同時考量隨機工作排程問題中延遲成本與儲存成本二種因素，針對三種不同的加工時間機率分布函數，利用一個典型的基因演算法，搭配一個已事先訓練好的類神經網路作為粗略模型來評估適應力，搜尋整個決定變數空間Ω以挑選出 N (=1024) 個粗略好的解。然後利用一個更準確的代理模型從 N 個粗略足夠好的解中求出一個足夠好的解。同時將我們方法所得到的結果與現有的派工法則進行比較，發現我們方法的表現的確超越其他方法。

關鍵詞：隨機工作生產排程，序列最佳化，類神經網路，基因演算法。

Abstract

In this paper, an ordinal optimization based approach is proposed to solve for a good enough schedule that minimizes expected sum of storage expenses and tardiness penalties of stochastic classical job shop scheduling problem using limited computation time. The proposed approach consists of exploration and exploitation stage. The exploration stage uses a genetic algorithm to select a good candidate solution set, where the objective function is evaluated with an artificial neural network that is trained beforehand. The exploitation stage composes of multiple substages, which allocate the computing resource and budget by iteratively and adaptively selecting the candidate solutions. At each substage, remaining solutions are simulated and some of them are eliminated, and the solution obtained in the last substage is the good enough schedule that we seek. The proposed approach is applied to a SCJSSP with random processing time in truncated normal, uniform, and exponential distributions. The test results demonstrated that the obtaining good enough schedule is successful in the aspects of solution quality and computational efficiency.

Keywords: Stochastic classical job shop scheduling; Ordinal optimization; Artificial neural network; Genetic algorithm.

1. 前言

排程主要功能是提供工廠內製造系統在考量加工時間與交貨日期等因素下，安排未來工作的生產順序，使全工廠的工作負荷平衡，以增加產出與降低成本。工作生產(job shop)是屬於少量多樣(High-mix low-volume)的生產模式，不同的生產或操作程序必須執行以完成一件工作。而等待完成的這些工作皆須依照各自排定的先後操作限制條件，在儘早完成所有工作的情況下得到一個生產順序，便是工作生產排程的目的。

隨機工作生產排程問題是等待處理的工作，操作的程序不相同，而且加工時間是隨機變動的[1]。因為在每個機器上面工作的加工時間是隨機變動，造成在每台機器上面工作的生產順序無法事先定好，縱使順序已事先定好，然而工作的實際加工時間的改變，也會使的這些順序變成無效。大部分在工廠內的製造系統工作生產排程皆是隨機排程問題，而隨機工作排程則屬於NP-Hard的問題。

近年來因彈性製造系統的需求，使得著重於隨機加工時間的排程問題成為新的研究主題。近年來，已有研究針對隨機加工時間問題探討不同的目標模型下的排程問題。其中有研究指出最佳化可解析模型僅可應用於特定具有隨機加工時間的工作生產排程問題[2]-[3]。在[4]中利用啟發式決定策略規則提供一般具有隨機加工時間的工作生產排程，然而其中並未考慮到最佳化問題。在[5]中則利用七種派工法則針對具有gamma分佈的隨機加工時間的工作生產排程問題，進行平均總完工時間(makespan)的分析。對於一般具有隨機加工時間的工作生產排程問題，最佳化方法尚無法得到數學解

析方式的解[6]。Ginzburg與Gonik [6]提出了一個新的決策規則，在同時考量未準時完工成本、延遲成本、儲存成本等三種因素下，針對三種不同的加工時間機率分布函數，以cyclic coordinate descent method找出最早開始加工時間。然而其目的是在避免機器不必要的閒置，僅找出最早開始加工時間，並未找出最佳的排程解。此外執行2個iteration需花費兩個小時的計算時間才能找出最早開始加工時間，且並無與其他方法進行比較。

Yoshitomi利用基因演算法針對具有normal 分佈的隨機加工時間的工作生產排程問題，以平均實耗時間(elapsed time)為目標尋求近似最佳解[7]。在[8]中針對具有normal分佈的隨機加工時間的工作生產排程問題，以平均總完工時間(makespan)為目標利用基因演算法進行搜尋，基因演算法中的corssover步驟則透過Giffler and Thompson algorithm調整為可行解，最後以Monte Carlo方法挑選出近似最佳解。然而加工時間為normal隨機分佈，會產生負數或零的情形，對於6x6大小的問題需花費17分鐘的計算時間才能得到結果，同時並無與其他方法進行比較。

在[9]中則提出了一個混合的方法來求解隨機工作生產排成問題，在同時考量實際加工時間、加工成本、閒置成本等三種因素下，先以類神經網路產生初始的可行解，接著以模擬退火法改進初始解的性能，在可容許的計算時間內求得接近最佳化的最早開始加工時間與排程解，並與Lingo 6軟體進行效能比較。不過加工時間僅考慮以confidence interval訂出上下限範圍的Uniform機率分布函數，且對於6x6大小的問題需花費74分鐘的計算時間才能得到結果，當問題大小變複雜後，就需花費很長的計算時間才能求得解。

在現有的排程方法裡，通常是假設加工時間為事先準備好且為固定值，也就是為deterministic，以實際應用的角度來看，加工時間或長或短常常超乎預期，為了能夠評估這項變化，我們將加工時間設為隨機變數，快速的以模擬方式求出一個足夠好的排程解，比起花費更多的時間去尋找最佳的排程解顯得更有價值。

在本篇論文中，我們提出一個以序的最佳化理論為基礎的演算法，同時考量隨機工作排程問題中延遲成本與儲存成本二種因素，針對三種不同的加工時間機率分布函數，求出隨機工作排程問題一個足夠好的解。先從決定變數空間Ω中挑選出N個粗略足夠好的解，然後結合現存目標軟化搜尋方法來求出一個足夠好的解。採用序的最佳化理論為基礎的的演算法來求出一個足夠好的解，在隨機模擬最佳化問題上是一個新的做法，也是這篇論文的第一個貢獻。將所提出的演算法應用在隨機工作排程的最佳化問題上，則是這篇論文的另一個貢獻。

這篇論文的編排方式如下：在第二章，將針對隨機工作排程問題建立數學模式，並且詳述這個隨機模擬最佳化問題的困難點。在第三章，將描述序的最佳化理論為基礎的演算法，如何從決定變數空間Ω中挑選一個足夠好的解，並且闡述以OO理論為基礎的二層次演算法。在第四章，將OO理論為基礎的演算法應用於隨機工作排程問題，針對truncted normal, uniform, exponential等三種不同機率分布的加工時間，測試我們的方法，並且以我們的方法獲得之結果與現有的派工規則獲得的結果進行比較。另外，也將對我們提出的方法進行性能的分析。最後，在第五章做一個結論。

2. 問題陳述與數學模式

2.1 隨機工作生產排程問題

在此我們探討的隨機排程問題為一般型工作生產排程問題。在一個彈性製造系統裡包含了n個工作J_i，$1 \le i \le n$，以及m台機器M_k，$1 \le k \le m$。每個工作指定了一連串的操作加工程序，對機器M_k而言，同一時間只能有一個工作進行加工，同一時間也只有一台機器可以對工作J_i進行加工。當一個或多個工作已準備好在某一台機器進行加工時，若那台機器為空閒狀態，則一個工作會被挑選且立刻送至那台機器。工作J_i在機器M_k上所需的加工時間為p_{ik}，為一隨機變數具有期望值$\overline{p}_{ik}$與變異數v_{ik}。每個工作J_i的交貨日期d_i為是先給定。

由於每個工作在機器上的加工時間為隨機變動，工作的完成時間不能確知，因此各個工作到達機器的時間便無法事先決定。雖然透過最佳化方法無法得到數學解析的解，但是可以利用模擬的方法，在可接受的計算時間範圍內得到一個足夠好的解。欲尋找的最佳排程解必須滿足操作優先限制(precedence constraint)與互斥性限制(mutual exclusion constraint)，同時要使某個特定性能準則(例如延遲時間、總完工時間、實耗時間等)達到最佳化。然而，在有限的時間內並沒有辦法來評估所有的解，因此就發展出來一些啟發式的演算法在製造過程找出合適的排程解。我們考慮的性能準則為使每個工作的平均延遲花費與庫存花費達到最小，在有限的計算時間內找出合適的排程解。

2.2 數學模式

首先定義隨機排程問題裡的一些參數符號意義如下:

n: 工作數量

m: 機器數量

J_i: 第 i 個工作，$1 \le i \le n$

M_k: 第 k 台機器，$1 \le k \le m$

d_i: 工作 J_i 的交貨日期(須先給定)

O_{ik}: 工作 J_i 的第 k 個操作，$1 \le k \le m_k$

m_k:工作 J_i 的操作數目，$1 \le k \le m_k$

p_{ik}: 操作 O_{ik} 的隨機加工時間

$\overline{p}_{ik}$: p_{ik} 的期望值(須先給定)

v_{ik}: p_{ik} 的變異數(須先給定)

m_{ik}:工作 J_i 的操作優先限制(須先給定)，

$1 \le m_{ik} \le m$

$[\overline{p}_{ik}, v_{ik}, m_{ik}]$: Operation-job pair matrix (須先給定)

t_{ik}: 決定向量，第 i 個工作在第 k 台機器的起始工作時間

Ω:決定向量空間

在此所考慮的隨機工作生產排程問題，其目標要使每個工作的平均延遲花費與庫存花費達到最小，在有限的計算時間內從決定向量空間 Ω 裡找出合適的排程解 S。可以將這個問題描述為隨機模擬最佳化問題，如(1)式所示

$$\min_{S \in \Omega} E\{\sum_{i=1}^{n} \tau_i T_i + \eta_i A_i\} \qquad (1)$$

其中 $T_i = \max(C_i - d_i, 0)$ 為 J_i 的延遲時間 tardniess， $A_i = \max(d_i - C_i, 0)$ 為 J_i 的庫存時間 earliness， C_i 為 J_i 的完成時間， d_i 為 J_i 的交貨日期，τ_i 為 J_i 的單位時間延遲成本，η_i 為 J_i 的單位時間庫存成本，因此 $\tau_i T_i$ 為 J_i 的operational costs，$\eta_i A$ 為 J_i 的idle costs， $\tau_i T_i + \eta_i A$ 為 J_i 的sum of operational and idle costs，$E\{\sum_{i=1}^{n} \tau_i T_i + \eta_i A_i\}$ 為expected sum of operational and idle costs。單位時間延遲成本與庫存成本可依照不同的工作訂定不同的值，通常將單位時間延遲成本訂定較大的值，而將單位時間庫存成本訂定較小的值。排程解 S 為決定向量，指定了每一個操作在各台機器上面的順序。

在排程解 S 的表示方法上有很多種[10]， 例如交叉排列表示法(permutation representation)、工作順序矩陣(Job sequence matrix)、甘特圖(Gantt chart)、分離圖(disjunctive graph)等，其中的交叉排列表示法(permutation representation)可以一維空間形式來表示，適合運用於我們的方法上面，因此挑選作為我們的方法裡面排程解 S 的表示方法。

我們以一個例子來說明交叉排列表示法。以一個3個工作3台機器的工作生產排程問題為例，每個工作的操作優先限制如下

表1 3個工作3台機器的工作生產排程問題

Jobs	Precedence constraint		
1	$O_{1,1}$	$O_{1,2}$	$O_{1,3}$
2	$O_{2,1}$	$O_{2,3}$	$O_{2,2}$
3	$O_{3,2}$	$O_{3,1}$	$O_{3,3}$

其中的 $O_{i,k}$ 代表工作 J_i 於機器 M_k 的操作。我們以交叉排列表示法來代表排程解 S，在維持各工作的操作優先限制條件下，隨機進行穿叉交列編碼成為一維空間的表示法，如 S_1 所示即為一個排程解，S_1 =[$O_{1,1}$ $O_{1,2}$ $O_{3,2}$ $O_{2,1}$ $O_{2,3}$ $O_{1,3}$ $O_{3,1}$ $O_{3,3}$ $O_{2,2}$]。

一個排程解為可行解(feasible solution)的條件必需滿足操作優先限制與互斥性限制。各個工作在不同機器進行加工時須滿足指定的先後順序的條件，稱為操作優先限制。當一台機器正在加工一個工作時，別的工作不能排入使用該機器，一定要等到正在加工的工作完畢之後，才能輪由別的工作使用，這種特性我們稱為互斥性(Mutual Exclusion)。

傳統的交叉排列表示法(permutation representation)是在不考慮各工作的操作優先限制條件下進行隨機交叉排列，如此編碼完成的排程解當中包含了不可行解，使決定向量空間變的更大。因此我們採用的隨機穿叉交列是在維持各工作的操作優先限制條件下進行，如此可使編碼完成的排程解 S 皆為可行解(feasible solution)，可避開不可行解的排列方式，有效降低決定向量空間的大小。在一個具有 n 個工作與 m 台機器的一般型工作生產排程問題，在不考慮各工作的操作優先限制條件下進行隨機交叉排列，共有 $(n*m)!$ 個排法，若是在維持各工作的操作優先限制條件下進行，則共有 $(n*m)!/(m!)^n$ 個排法，因此決定向量空間 Ω 大小即為 $(n*m)!/(m!)^n$ 。

一旦隨機穿叉交列產生排程解 S 後，便可決定

分離圖(disjunctive graph) [10]中虛線箭頭的方向，配合每個工作 J_i 的所有操作加工時間 p_{ik} ，即可求出每個 J_i 的完成時間 C_i 以及目標函數值。以 S_1 排程解為例，其對應的分離圖如圖1。

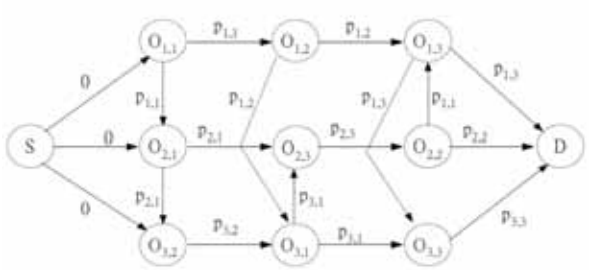

圖1. disjunctive graph of schedule S_1.

很顯然地，最佳化問題(1)是個具有很大的離散決定變數空間Ω的隨機模擬最佳化問題，若要計算每一個決定向量的真正目標函數值，需要執行足夠多次數的模擬程序，才能使目標函數值趨於穩定。然而這樣太耗費時間，顯的不切實際。因此可透過模擬次數的多寡，將原本(1)式修改如下

$$\min_{S\in\Omega} F(S)=\frac{1}{L}\sum_{l=1}^{L}E\{\sum_{i=1}^{n}\tau_i T_i^l(S)+\eta_i A_i^l(S)\} \qquad (2)$$

其中L代表模擬的次數(replications of simulation)，足夠大的 L 可使(2)式的目標函數值 $F(S)$ 趨於穩定。$T_i^l(S)$ 代表針對排程解 S 執行第 l 次模擬所得到 J_i 的延遲時間，而 $A_i^l(S)$ 表示針對排程解 S 執行第 l 次模擬所得到 J_i 的庫存時間。目標函數值 $F(S)$ 表示針對排程解 S 執行 L 次模擬後得到的每個工作在延遲與庫存花費的平均值。

3. 序的最佳化理論為基礎的演算法

3.1 二層次演算法

為了克服問題(1)的計算複雜度，我們將採用序的最佳化(Ordinal Optimization, OO)理論為基礎的目標軟化搜尋方法[22]-[23]，可以有效地找出一組具有高可靠性且足夠好的排程解，來取代傳統搜尋最好的排程解。即使隨機模擬使用的是一個粗略的模型，可以發現這個粗略的模型仍然保留著決定向量的性能順序。

透過觀察可以發現決定向量的性能縱使經由一個粗略的代理模型來評估，仍然保有性能上的順序，OO理論的基本概念是使用一個簡單粗略的代理模型來快速評估一個決定向量的估測性能，在有限的計算時間內從候選解集合中挑選一個估計足夠好的子集合。若是決定變數空間大小很巨大，可以分成許多階段來降低搜尋空間。我們提出的OO 理論為基礎的方法包含了兩個層次來求解(2)式得到一個足夠的決定向量。

第一層次是探索層次，在這層次我們利用了一個典型的基因演算法，搭配一個已事先離線訓練好的類神經網路作為粗略模型來評估適應力，搜尋整個 空間挑選出 N (=1024)個粗略好的決定向量。第二層次是開發層次，利用一個更準確的代理模型從第一層次得到的 N 個解當中找出一個足夠好的解。假設採用準確的模型來評估所有 N 個解，可以獲得最佳的解，然而卻需要花費太多的計算時間，這與我們的目標相牴觸。因此我們將第二層次分成許多的子層次。在這些子層次裡用來估計目標函數值的代理模型，是透過不同模擬次數的隨機模擬，模擬的次數可由非長短(粗略模型)至非常長(準確模型) 。在每一個子層次的候選解(或是從前一個子層次所獲得的估計足夠好的子集合)將逐漸的減少。在最後一個子層次，將針對留下的候選解使用準確的模型來評估所有的決定向量，具有最小的目標函數值的解就是我們想要的答案。因此，計算的複雜度便可以大大地減少，因為當代理模型變的更為準確時，候選解的集合大小也大大地降低。接下來我們將描述OO理論為基礎的兩層次方法的細節。

3.2 層次-1 演算法

如同在OO理論所指出[22]-[23]，排程的目標函數值即使以粗略的模型來評估，其順序性能也會被保留。因此，要從Ω中挑選 N (=1024)個粗略足夠好的排程且不花費太多的計算時間，必須建構一個粗略的模型。只要給定一組排程 S 便可以方便地估計出(2)式的目標函數值 $F(S)$ ，接著使用一個有效的方法來挑選出 N 個粗略足夠好的排程。我們建構之粗略的模型為類神經網路(Artificial Neural Network, ANN) [27]，而挑選機制為基因演算法(Genetic Algorithm, GA) [16]。

3.2.1 類神經網路為基礎的粗略模型

ANN可考慮做為一個萬用型的近似器[24]，包括應用在離散事件模擬系統[25]-[26]的輸入與輸出之間的對應關係，然而，近似器的準確性跟ANN的架構複雜性是息息相關。換句話說，在準確性與訓練時間上存在著一個互相矛盾。既然我們在乎的是相對的順序性能而不是目標函數值的絕對性能，就可以利用一個簡單的ANN當作模型。

考慮輸入為排程 S 以及輸出為相對應的目標函數值 $F(S)$，可以使用一個類神經網路ANN來完成輸入與輸出之間的對應關係[24]。首先，透過一個給定的族群決定隨機取樣的大小[27]，考慮在confidence level=99%, confidence interval=1%的情況下，可以從中均勻挑選出 M =16641個排程 S 代表Ω的子集合。然後利用一個完整的模型來評估這 M 個排程的目標函數值 $F(S)$，可以透過一個具有足夠長的模擬次數 L_s，例如10000次，的隨機模擬來完成。這些收集到的 M 個輸入-輸出對 $(s, F(S))$ 將被用來訓練ANN以調整每個類神經的權重。我們採用一個簡單的三層前饋式ANN，在第一層輸入層具有 $n*m$ 個神經元，在第二層隱藏層具有 $2*n*m$ 個神經元，在第三層輸出層有1個神經元。在隱藏層中神經元使用的轉移函數為正切雙彎曲函數(hyperbolic tangent sigmoid function)，在輸出層中神經元使用的轉移函數為線性函數。為加快倒傳遞訓練的收斂速度，以比例共軛梯度演算法進行訓練[28]，訓練的停止條件為當底下兩種情況任何一種發生時: (i)均方差的總合小於 10^{-5}，或是(ii)疊代的次數超過500次。一旦ANN訓練完畢，就可以輸入任意的排程 S 到ANN裡，經由ANN的輸出可得到相對應的目標函數值 $F(S)$；如此對於一個排程 S 可以避免花費冗長的隨機模擬時間來評估 $F(S)$。以上的部分完成建構一個粗略的模型，對於一個給定的排程 S 可以粗略且有效地來評估(2)式的目標函數值。

3.2.2 基因演算法

有了ANN這個有效率的目標函數值(在GA術語上稱做適應函數值) 評估方法，可以使用GA很有效率的來挑選出 N 個出色的排程，既然GA 可以透過上一代至下一代的演化來改進族群的適應函數值，很適合我們的需要，將做法簡短的描述如後。假設任意產生一初始族群，計算每一個基因的適應函數值，接著透過以下三個基本操作:(a)母代選擇;(b)交配;(c)突變，來進行基因的演化過程。

在排程問題裡面，我們採用的交叉排列表示法(permutation representation)是透過一維空間形式來表示，正好可直接套用於GA裡代表基因，每一個基因以工作的交叉排列進行編碼為字串，這個字串就叫做染色體。從Ω裡均勻地挑選出 l，例如5000個排程當做初始族群。每一個排程的適應函數值是透過ANN計算所得到的目標函數值的倒數(假設 $F(S)$，$\forall S \in \Omega$)。母代的選擇是個簡單的程序，在我們的方法中所使用的選擇母代方式為輪盤選擇法，將兩個染色體從母代族群中挑選出來，具有較大適應函數值的染色體將有更高的機會可以貢獻出自己繁衍的子代到新世代中。交配在GA中扮演一個非常重要的角色，不但負責了遺傳基因的重組（交配的染色體間資訊交換）也會影響到GA的收斂速度，所以交配的參數通常設定成一個高的機率值(p_c)。兩個母代的染色體挑選出來後結合形成新的染色體，同時繼承了原本母代的資訊片段。交配是主要的基因操作部分，可以探索目前世代所包含的資訊，但並未產生新資訊。

我們以單點交配法來進行交配，每個母代的染色體在位交配之前都是可行解，經過單點交配法後會造成子代染色體變成不可行解，因此必修對子代進行修正，才能確保子代都是可行解。交配的動作以底下圖2做說明，兩個母代 P_1 與 P_2 皆為可行解，隨機挑選單點交配位置為5，經過交配後得到子代 C_1 與 C_2 皆不是可行解，因此對子代 C_1 進行修正，先將重複項目 $O_{2,3}$ 刪除，並補回欠缺項目 $O_{3,1}$，$O_{3,1}$ 補回的位置可藉於 $O_{3,2}$ 與 $O_{3,3}$ 隨機挑選出一個來補回，如此修正即可得到屬於可行解的子代 $\hat{C}_1$。同樣地對子代 C_2 進行修正，先將重複項目 $O_{3,1}$ 刪除，並補回欠缺項目 $O_{2,3}$，$O_{2,3}$ 補回的位置可藉於 $O_{2,1}$ 與 $O_{2,2}$ 隨機挑選出一個來補回，如此修正即可得到屬於可行解的子代 $\hat{C}_2$。

$$
\begin{aligned}
P_1 &= [O_{1,1}\ O_{1,2}\ O_{3,2}\ O_{2,1}\ O_{2,3} \mid O_{1,3}\ O_{3,1}\ O_{3,3}\ O_{2,2}] \\
P_2 &= [O_{3,2}\ O_{3,1}\ O_{1,1}\ O_{1,2}\ O_{2,1} \mid O_{2,3}\ O_{3,3}\ O_{2,2}\ O_{1,3}] \\
C_1 &= [O_{1,1}\ O_{1,2}\ O_{3,2}\ O_{2,1}\ O_{2,3} \mid \underline{O_{2,3}}\ O_{3,3}\ O_{2,2}\ O_{1,3}] \\
C_2 &= [O_{3,2}\ O_{3,1}\ O_{1,1}\ O_{1,2}\ O_{2,1} \mid O_{1,3}\ \underline{O_{3,1}}\ O_{3,3}\ O_{2,2}] \\
\hat{C}_1 &= [O_{1,1}\ O_{1,2}\ O_{3,2}\ O_{2,1}\ O_{2,3} \mid \overleftarrow{O_{3,1}}\ O_{3,3}\ O_{2,2}\ O_{1,3}] \\
\hat{C}_2 &= [O_{3,2}\ O_{3,1}\ O_{1,1}\ O_{1,2}\ O_{2,1} \mid O_{1,3}\ O_{3,3}\ \overleftarrow{O_{2,3}}\ O_{2,2}]
\end{aligned}
$$

圖2. 基因演算法交配之範例

突變則是負責注入新資訊的操作，以一個很小的發生機率 p_m，從子代的染色體中任意挑選幾個位元進行變化，如此就能得到原本不存在母代的新特徵。我們採用的突變方法如下，對每一個 J_i 隨機產生介於0~1的數值 p_i，$1 \le i \le n$，若 $p_i < p_m$ 表示對 J_i 的所有操作 O_{ik} 進行突變，首先保留原有染色體中其餘未突變工作 J_j，$i \ne j$，的所有操作 O_{jk}，接者將 J_i 的所有操作 O_{ik} 刪除，然後依照 J_i 的操作優先限制將 O_{ik} 依照順序隨機重新排入保留的所有操作 O_{jk} 當中。突變的動作以底下圖3做說明，以 J_1

的所有操作O_{1k}進行突變，先將子代染色體C_1中的所有操作O_{1k}刪除，保留其餘O_{jk}的順序，接著依照J_1的操作優先限制將O_{1k}依照順序隨機重新排入保留的所有操作O_{jk}當中，如此即可得到屬於可行解的突變子代M_1。

$C_1 = [O_{1,1} \quad O_{1,2} \quad O_{3,2} \quad O_{2,1} \quad O_{2,3} \quad O_{1,3} \quad O_{3,1} \quad O_{3,3} \quad O_{2,2}]$

$\hat{C}_1 = [O_{3,2} \quad O_{2,1} \quad O_{2,3} \quad O_{3,1} \quad O_{3,3} \quad O_{2,2}]$

$M_1 = [O_{3,2} \quad O_{1,1} \quad O_{2,1} \quad O_{2,3} \quad O_{3,1} \quad O_{1,2} \quad O_{3,3} \quad O_{1,3} \quad O_{2,2}]$

圖3. 基因演算法突變之範例

GA有兩個準則可用來判斷是否要停止演化，第一個準則是當連續幾個世代裡面最好的基因的適應函數值已不再改進，另一個準則為已經演化了足夠多的世代。當套用GA進行演化一直到收斂或停止後，將最後一個世代的所有族群I依照適應函數值由小排列到大，挑選出前面N個基因，即是我們要找的粗略足夠好的N個排程。

3.3 層次-2 演算法

在這一個層次，利用一個比ANN更準確的代理模型從第一層次得到的N個解當中找出一個足夠好的解。這個用來估計目標函數值的代理模型，可透過不同模擬次數的隨機模擬來完成。

首先定義一個基本的模擬次數L_0=500，設定在子層次i的模擬次數L_i為$L_i = kL_{i-1}$（或$L_i = k^i L_0$），$i = 1,2,...,$其中的正整數$k(\geq 2)$代表控制模擬次數L_i的參數。設定$N_0 = N$，在子層次i裡所挑選出來估計足夠好的子集合大小為$N_i = N_{i-1}/k$（或$N_i = N_0/k^i$），$i = 1,2,...,$。以n_k表示子層次的層次數， 可經由公式(2)來決定如下:

$$n_k = \arg\{\min_{n_k}(L_0 k^{n_k - 1} \leq L_s < L_0 k^{n_k}, 1 < N_{n_k - 1} \leq 10)\} \quad (2)$$

其中$L_s = 100000$。子層次的層次數n_k是由以下兩者的最小值來決定:(i)在第n_k子層次的模擬次數$L_0 k^{n_k}$已超過L_s時, (ii)在子層次n_{k-1}裡所挑選出來估計足夠好的子集合大小已足夠小， 例如$1 < N_{n_k - 1} \leq 10$。

一旦n_k已決定好，可以設定$L_{n_k} = L_s$與$N_{n_k} = 1$，代表在最後一個子層次(即子層次n_k)，代理模型實際上就是完整的模型，具有最小值$F(S)$的決定向量即是我們要找的足夠好的決定向量。如果k的值非常大使得$L_1 = kL_0 > L_s$，將只會有一個子層次，所有N個決定向量將透過完整的模型來進行評估，這樣將會耗費太多的計算時間，縱使找到的決定向量的確是所有N個決定向量裡面是最好的。然而，將計算時間與得到足夠好的決定向量的好壞之間的矛盾情形以可解析的公式表達並不容易。事實上，要挑選出最好k的值需視問題而定，因為某些問題比較在意計算時間而其他問題則比較在乎所得到解的好壞。因此，我們將在第四章針對不同的k值展示計算時間與所得到解的好壞，然後挑選出最好的k值套用在我們的方法。

3.4 序的最佳化理論為基礎的演算法

以OO理論為基礎的二層次演算法，來求解(2)式足夠好的排程可以整理如下：

Off-line trained ANN: 從Ω中均勻挑選出M個排程S，以L_s的模擬次數對每一個排程S評估目標函數值$F(S)$。以收集到的M個輸入-輸出對$(s, F(S))$來訓練ANN以調整類神經元的權重ω。以$f(S, \varpi)$代表訓練好的ANN的輸出。

Step 1: 從Ω裡均勻地挑選出I個排程S當作初始族群。套用以下的GA步驟: 輪盤選擇法，單點交配法其交配機率為p_c，突變機率為p_m，適應函數值設定為$1/f(S, \varpi)$。當GA經過20個世代演化後，將最後一個世代的所有族群I依照適應函數值由小排列到大，挑選出最好的N個基因(排程S)。

Step 2: 由$i = 1$到$n_k - 1$，透過模擬次數為$L_i = k^i L_0$的隨機模擬來評估N/k^{i-1}個候選解的目標函數值$F(S)$，將這N/k^{i-1}候選解依照目標函數值由小排列到大，挑選出最好的N/k^i個S作為子層次$i+1$的候選解。

Step 3: 透過模擬次數為L_s的隨機模擬來評估$N/l^{n_k - 1}$個候選解的目標函數值$F(S)$，具有最小目標函數值的排程S即是我們尋找的足夠好的解。

4. 測試結果

在一個具有n個工作與m台機器的一般型工作生產排程問題，以工作的交叉排列進行編碼成為一維空間表示法，則離散決定變數空間大小為$(n*m)!/(m!)^n$，若是具有6工作與6台機器的情況下，離散決定變數空間Ω大小為$(6*6)!/(6!)^6 = 2.67*10^{24}$。很顯然地，這是個具有很大的離散決定變數空間$\Omega$的隨機模擬最佳化問

題。

因此我們選定的測試範例為一個具有6個工作與6台機器之一般型工作生產排程問題，Operation-job pair matrix [$\overline{p}_{ik}$, v_{ik} , m_{ik}]如表2設定:

表2 加工時間期望值、變異數與操作優先限制設定

	Op. #1	Op. #2	Op. #3	Op. #4	Op. #5	Op. #6
J1	50,150,1	40,120,2	80,240,3	60,180,5	70,210,4	40,120,6
J2	60,180,2	50,150,1	60,180,3	70,210,4	80,240,5	40,120,6
J3	70,210,3	80,240,2	90,270,1	50,150,6	40,120,5	60,180,4
J4	80,240,1	40,120,2	50,150,3	90,270,4	40,120,5	50,150,6
J5	60,180,2	80,240,3	90,270,1	70,210,5	50,150,6	40,120,4
J6	50,150,4	50,150,3	70,210,2	40,120,1	50,150,5	60,180,6

矩陣中每一個元素($\overline{p}_{ik}$, v_{ik} , m_{ik})裡的數值代表意義， $\overline{p}_{ik}$ 為操作 O_{ik} 的隨機加工時間期望值， v_{ik} 為操作 O_{ik} 的隨機加工時間變異數， m_{ik} 為工作 J_i 的操作優先限制。工作 J_i 的交貨日期、單位時間延遲成本，與單位時間庫存成本，設定如表3:

表3 交貨日期、單位時間延遲成本庫存成本的設定

J_i	1	2	3	4	5	6
d_i	540	620	680	600	700	500
τ_i	10	10	10	10	10	10
η_i	1	1	1	1	1	1

針對工作 J_i 在機器 m_k 上所需的加工時間，考慮三種不同機率分佈的隨機變數。第一種為機率分佈為Truncated normal distribution with parent mean $\overline{p}_{ik}$ and parent variance v_{ik} ，第二種為機率分佈為Uniform distribution in the interval [$\overline{p}_{ik} - \sqrt[3]{v_{ik}}, \overline{p}_{ik} + \sqrt[3]{v_{ik}}$]，第三種為機率分佈為Exponential distribution with mean $\overline{p}_{ik}$ 。

套用OO理論為基礎的演算法的步驟**Off-line trained ANN**:到問題(2)，必須先建構出以ANN為基礎的粗略模型，包含了兩個部分:(A)收集訓練資料，(B)訓練ANN。在此採用一個三層前饋倒傳遞類神經網路，在第一層輸入層具有36個神經元，在第二層隱藏層具有72個神經元，在第三層輸出層有1個神經元。，而隱藏層與輸出層的神經元採用的活化函數分別是雙曲線正切彎曲函數與線性函數。ANN的輸入為排程 S ，輸出為 $F(S)$ ，接著利用以下的步驟來得到ANN的訓練資料。均勻地從Ω中挑選出 M =16641個排程，使用一個適當數量的模擬次數的隨機模擬[24]來計算相對應的輸出 $F(S)$ ，也就是執行模擬程序重複 L_s ，例如100000，次模擬的隨機模擬。如此便建構出粗略模型的(A)部分。

有了上述有效率的目標函數值評估方法，便可以套用OO理論為基礎的演算法的步驟1，使用基因演算法從Ω中挑選出 N 個出色的排程。

在Ω中所有的排程所採用的編碼方式相當明確，因為組成排程 S 的每一個成份都是一個工作。先從Ω中任意的挑選出 I (=5000)個排程當作初始族群。每一個排程的適應函數值是透過ANN的輸出來計算(2)式的目標函數值，從族群池當中挑選出到交配池所使用的選擇母代方式為輪盤選擇法，70%在交配池中的數量將被挑選當作交配的母代，使用單點交配方式且假設突變機率為0.02，當疊代次數超過20便停止GA。套用GA進行演化一直到停止時，將最後一個世代的所有族群 I (=5000) 依照適應函數值由小排列到大，挑選出前面 N (=1024) 個排程。

由GA演化得到的 N (=1024)個排程 S 開始，接下來利用一個比ANN較準確的模型，來評估每一個排程 S 的(2)式目標函數值。將第二層次分成許多的子層次。在這些子層次裡用來估計目標函數值的代理模型，是透過不同模擬次數的隨機模擬，在增加的模擬次數後，相對也提高了模型的準確性，同時逐漸的減少在每一個子層次的候選解。

在第二層次中為挑選出最好的 k 值套用在我們的方法，我們將針對不同的 k 值展示計算時間與所得到解的好壞。

以Truncated normal distribution為例，對於 k =2,3,4,5,6和200，設定 L_0 =500，模擬次數為 $L_n = k^n \cdot L_0$ ，當模擬次數 L_n >100000時，則設定 L_n =100000，而候選解的數目為 $N_n = 1024/k^{n-1}$ 。

在第二層次中以 k =2，設定 L_0 =500，模擬次數為 $L_n = 2^n \cdot L_0$ ，當模擬次數 L_n >100000時，則設定 L_n =100000，而候選解的數目為 $N_n = 1024/2^{n-1}$ ，表4為層次-2中各子層次的模擬次數與候選解的數目。在最後一個子層次裡找出8個候選解當中最小的(2)式目標函數值所對應的排程 S ，就是我們要尋找的足夠好的排程 S_g 。

表4 為層次-2中各子層次的模擬次數與候選解的數目

n_l	1	2	3	4	5	6	7	8
N_n	1024	512	256	128	64	32	16	8
L_n	1000	2000	4000	8000	16000	32000	64000	100000

為分析我們的方法所得到足夠好的解其性

能，透過Determining Random Sample Size from a Given Population[28]，考慮在confidence level=99%, confidence interval=0.1%的情況下，可以從Ω中均勻挑選出|Θ|=1664100個排程 S 代表Ω的性能分析子集合Θ。由各個方法得到的解，其目標函數值在分析子集合Θ中所佔的名次(order)除以|Θ|，即可得到名次性能百分比，Performance= Order/|Θ|。表5為我們的方法所得到足夠好的解、目標函數值與性能分析。

表5 我們的方法所得到足夠好的解與性能分析

Distribution	S_g	$F(S_g)$	Order	Performance
Truncated normal	14 32 2 8 33 26 20 9 27 34 21 35 36 3 28 37 22 15 29 4 23 10 16 24 11 30 25 31 5 6 17 12 13 18 7 19	1140	251	0.01513%
Uniform	20 26 8 2 14 27 21 22 28 32 3 23 4 29 15 9 30 33 16 34 17 24 31 25 10 35 5 36 18 6 11 7 37 12 13 19	1110	113	0.0068%
Exponential	20 14 8 32 15 9 21 2 26 16 10 11 33 12 22 3 34 4 27 17 18 23 24 19 25 5 35 13 28 36 37 29 6 7 30 31	1210	582	0.035%

5. 結論

為了克服隨機模擬最佳化工作排程問題在計算上會耗費大量時間，我們提出一個以序的最佳化理論為基礎的演算法，在一個合理的計算時間內，同時考量隨機工作排程問題中延遲成本與儲存成本二種因素，針對三種不同的加工時間機率分布函數，求出隨機工作排程問題一個足夠好的解。先從決定變數空間Ω中挑選出 N 個粗略足夠好的解，然後結合現存目標軟化搜尋方法來求出一個足夠好的解。同時展示了我們的方法與現有的派工規則所獲得的結果，發現我們的方法表現最好。而高效率的計算與獲得到足夠好的解之良好品質，使我們的方法確實可以達到即時的應用。

參考文獻

[1] T.C. Lai , Y.N. Sotskov, N. Sotskova, F. Werner, "Mean flow time minimization with given bounds of processing times," *European Journal of Operational Research*, Vol.159,pp.558 − 573, 2004.

[2] M. Pinedo, "Stochastic scheduling with release dates anddue dates," *Operations Research*,Vol.31, pp.559 − 572, 1983.

[3] R.R. Weber, P. Varaiya, J. Walrand, "Scheduling jobs with stochastically ordered processing times on parallel machines to minimize expected flow time," *Journal of Applied Probability*, Vol.23, pp.841 − 847, 1986.

[4] D. G. Ginzburg, Sh. Kesler, Z. Landsman, "Industrial job-shop scheduling with random operations and different priorities, " *International Journal of Production Economics*, Vol.40, pp.185 − 195, 1995.

[5] S. R. Lawrence a, E. C. Sewell, "Heuristic, optimal, static, and dynamic schedules when processing times are uncertain " , *Journal of Operations Management*, Vol. 15, pp. 71 − 82, 1997.

[6] D.G. Ginzburg, A. Gonik, "Optimal job-shop scheduling with random operations and cost objectives," *International Journal of Production Economics*, Vol,76, pp.147 − 157, 2002.

[7] Y. Yoshitomi, "A genetic algorithm approach to solving stochastic job-shop scheduling problems " , *International Transaction in Operational Research*, Vol.9, No. 4, 479 − 495, July 2002.

[8] Y. Yoshitomi, R. Yamaguchi, "A genetic algorithm and the Monte Carlo method for stochastic job-shop scheduling " , *International Transaction in Operational Research*, Vol.10, No. 6, 577 − 596, Nov. 2003.

[9] R. Tavakkoli-Moghaddam, F. Jolai, F. Vaziri, P.K. Ahmed, A. Azaron, "A hybrid method for solving stochastic job shop scheduling problems, " *Applied Mathematics and Computation*, Vol.170, pp.185 − 206, 2005.

[10] T. Yamada and R. Nakano, Job-Shop Scheduling. Genetic Algorithms in Engineering Systems, Chapter 7 (pp. 134-160) IEE control Engineering series 55, 1997.

[11] F. Azadivar, "Simulation optimization methodologies," *Proceedings of the 1999 Winter Simulation Conference*, Phoenix, AZ, pp. 93-100, 1999.

[12] J.R. Swisher, P.D. Hyden, S.H. Jacobson, and L.W. Schruben, "A survey of simulation optimization techniques and procedures," *Proceedings of the 2000 Winter Simulation Conference*, Orlando, FL, pp. 119-128, 2000.

[13] S.M. Robinson, "Analysis of sample-path optimisation," *Mathematics of Operations Research*, Vol. 21, No. 3, pp. 513-528, 1996.

[14] A.G. Greenwood, L.P. Rees, and F.C. Siochi, "An investigation of the behavior of simulation response surfaces," *European Journal of Operational Research*, Vol. 110, pp. 282-313, 1998.

[15] Y. Carson, and A. Maria, "Simulation optimization: methods and applications," *Proceedings of the 1997 Winter Simulation Conference*, Atlanta, GA, pp. 118-126, 1997.

[16] R.L. Haupt, and S.E. Haupt, "*Practical genetic algorithms,* "2nd edition, Hoboken, NJ:JohnWiley,

2004.
[17] B. Suman, and P. Kumar, “A survey of simulated annealing as a tool for single and multiobjective optimization,” *Journal of the Operational Research Society*, Vol.57, No.10, pp.1143-1160, 2006.
[18] A.R. Hedar, and M. Fukushima, “Tabu Search directed by direct search methods for nonlinear global optimization," *European Journal of Operational Research*, Vol.170, No.2, pp.329-349, 2006.
[19] P. Winker, and M. Gilli, “Applications of optimization heuristics to estimation and modelling problems," *Computational Statistics and Data Analysis*, Vol.47, No.2, pp.211-223, 2004.
[20] M.C. Fu, F.W.Glover, and J. April, “Simulation optimization: a review, new developments, and applications,” *Proceedings of the 2005 Winter Simulation Conference*, Orlando, FL, pp.83-95, 2005.
[21] E. Tekin, and I. Sabuncuoglu, “Simulation optimization: A comprehensive review on theory and applications,” *IIE Transactions*, Vol.36, No.11, pp.1067-1081, 2004.
[22] T.W.E. Lau, and Y.C. Ho, “Universal alignment probability and subset selection for ordinal optimization,” *Journal of Optimization Theory and Applications*, Vol.93, No.3,pp.455-489, 1997.
[23] Y.C. Ho, “An explanation of ordinal optimization: Soft computing for hard problems,” *Information Sciences*, Vol.113, No.3-4, pp.169-192, 1999.
[24] K. Hornik, M. Stinchcombe, and H. White, “Multilayer fedforward networks are universal approximators,” *Neural Networks*, Vol.2, No.5, pp.359-366, 1989.
[25] D.J. Fonseca, D.O. Navaresse, and G.P. Moynihan, “Simulation metamodeling through artificial neural networks,” *Engineering Applications of Artificial Intelligence*, Vol.16, No.3, pp.177-183, 2003.
[26] F.M. Alam, K.R. McNaught, and T.J. Ringrose, “A comparison of experimental designs in the development of a neural network simulation metamodel,” *Simulation in Operational Research*, Vol.12, No.7-8, pp.559-578, 2004.
[27] Moore, D. and McCabe, G.. Introduction to the Practice of Statistics. (3rd Edition). New York:W.H. Freeman and Company**,** 1999.
[28] M.F. Moller, “A scaled conjugate gradient algorithm for fast supervised learning," *Neural Networks*, Vol.6, No.4, pp.525-533, 1993.
[29] Chen, C.H., Lin, J.W., Cesan, E.Y. and Chick, S.E., *Simulation budget allocation for further enhancing the efficiency of ordinal optimization*, Discrete Event Dynamic Systems: Theory and Applications, Vol.10, pp.251-270, 2000.
[30] Dileep R. Sule ,Production Planning and Industrial Scheduling: Examples, Case Studies and Applications, 2nd edition, PWS Publishing, Boston, 2007.