應用基因演算法於委外作業專案排程之設計
Applying a Genetic Algorithm to Project Scheduling for Outsourcing

江傳文
高雄第一科技大學電腦與通訊工程系
ccw@ccms.nkfust.edu.tw

范智欽、錢雅惠
大葉大學資訊工程學系
R9106015@mail.dyu.edu.tw

摘要

本論文中，我們探討如何導入專案管理的技術以降低委外作業的風險，進而有效控制委外作業專案執行的成本。我們提出一個演算法用以支援企業組織擇優選出適合執行專案中特定工作的承包廠商。此一方法係以遺傳演算法的基本架構為基礎。在交配程序中，我們針對問題的特性設計出一個名為交替式順序交配 (alternative order crossover，AOX) **運算子。**AOX **運算子經證明可用以產生完全符合問題限制條件的合法染色體。我們同時也將突變運算子與模擬退火方法優異的區域搜尋能力整合在一起，用以強化突變的效能，避免無效的突變運算。實驗結果顯示，在委外作業專案排程問題的求解中，我們所提出的演算法確實遠優於傳統的方法。**

關鍵詞：委外作業、專案排程、遺傳演算法、模擬退火

Abstract

In this paper, we work on an issue that helps enterprises employ the technology of project management to reduce the risk of outsourcing, and then control effectively the cost of project executing. We propose an algorithm that can provide enterpris es the best outsourcing parties to choose while they are in need of outsourcing service for some special purpose. On the basis of a general framework of genetic algorithms, this proposed algorithm is specific to its crossover operator: the alternative order crossover (AOX). We proof that the chromosomes generated by the AOX operator satisfy the precedence constraints of a project. Moreover, we also combine the mutation operator and concepts of the simulated annealing (SA) so that a useless mutation can be avoided. Experimental results show that the proposed algorithm outperforms the conventional approaches while solving the project-scheduling problem for outsourcing.

Keywords: Outsourcing, project scheduling, genetic algorithms, simulated annealing

一、簡介

近年來，為了因應經濟體系逐漸全球化的發展趨勢，於是有越來越多的企業應用「委外作業」的概念重塑其組織架構，企圖提昇企業本身的競爭力。委外作業的基本精神係指企業依照該組織的發展方針或規格書等特定之要求，將組織內部生產活動的一部份委託給其它廠商執行的一種行為 [2]**。此一概念引領企業組織更能專注於其核心競爭力的提昇，其具有節省成本、促成組織精簡化、增加組織彈性與競爭力、提昇服務品質、改善生產力等優點** [1, 3]**。然而，委外作業也具有相當程度的潛在風險。例如：額外的費用支出、服務作業與服務品質的失控現象等** [9]**。究其原因，便在於委外作業專案管理機制的缺乏。也因此，如何妥善處理委外作業專案的排程以降低委外作業在執行時的風險，進而有效地控制專案執行的時程與成本，勢將成為企業採行委外作業得以成功的重要因素。**

一般而言，專案 (project) **係指一項在特定時期之內投入預定資源以完成某一特定目標的工作；而專案管理則是一序列管理原則與控制方法之應用的過程，以使專案的目標與需求得以順利完成。是以，不同的組織或產業對專案管理的運作方式以及所著重之特性都各自不同** [7]**。儘管如此，依然還是有某些特性是所有專案管理皆必須加以考量的因素。例如：成本以及專案時程的管制。在管理專案成本以及執行時程的諸多工具之中，計劃評核技術** (program evaluation and review technique，PERT) **與要徑法** (critical path method，CPM) **則是兩種最常使用於規劃與協調大型專案的方法。藉由** PERT **或** CPM **方法的使用，管理者可以獲得以下的資訊，進而有效地管制專案的執行進程** [12]**：**

- 圖形化的專案活動表示。
- 完成專案所需花費時間的預估。
- 專案中各個活動之預定執行起始時間與結束時間。
- 專案中各個活動之間相對的重要性。

儘管 PERT 與 CPM 可以提供管理者許多有用資訊，但在對具有並行、反覆與演進式特性的軟體專案進行規劃與管理時，這兩種工具在應用上仍有明顯的不足 [4, 11]。舉例來說，當我們在考慮專案發展過程中有限資源分配的特性與限制時，CPM 便無法進一步地提供如何分配資源的相關資訊。為了解決以上的缺點，我們將在本論文中針對欲求解問題的特性設計出一個處理委外作業專案排程的演算法。

本文中所提之方法係以基因演算法的架構為基礎，其目的是協助企業在執行委外作業專案的過程中產生最佳資源分配與專案時程管制的決策結果。基因演算法是植基於達爾文「物競天擇，適者生存」演化理論的概念所發展出來的方法 [5, 6]。此一方法首先會將吾人所欲求解之問題以字串的方式編碼成為所謂的「染色體 (chromosome)」。每個染色體中包含了數個基因 (gene)，不同的基因排列表現出此一染色體的不同特徵，而這些特徵在經過評估 (evaluation) 之後便會反映出該染色體對所在環境的適存程度 (fitness)，適存度越高的染色體表示其所相對應解的品質越高。基因演算法一般具有「交配 (crossover)」、「突變 (mutation)」以及「選擇 (selection)」等三個運算子。交配運算子的主要目的在於將不同染色體的部份基因予以交換，進而使得相對應解的優良特性可以被保留至下一個世代；突變運算子則是隨機改變染色體中部份基因的內容，用以在搜尋過程中跳脫局部最佳解的陷阱；至於選擇運算子則是用以決定哪一個染色體可以存活至下一個世代。對於環境的適存程度越高的染色體而言，其在演化過程中得以存活的機率當然也就越高。如此反覆運作許多世代之後，我們預期將可以獲致高品質的問題解。

本文在後續的內容中，我們首先就所要處理之委外作業專案排程問題加以描述且定義；之後，我們將詳細說明所提出方法的設計細節並進一步探討其特性；接著我們會針對演算法的效能進行評估，然後再藉由一個協同式委外作業專案管理系統的實作以驗證設計方法的可行性。最後，我們將為本文的內容進行總結並說明未來相關的研究工作。

二、問題塑模

本論文中，我們利用一個有向非循環圖 (directed acyclic graph，DAG) 來表示一個專案。在任一特定的 DAG 中，節點係用來表示專案中的工作 (task) 或活動 (activity)；而 DAG 中的有向邊則係用來表示工作與工作之間的執行先後順序關係 (precedence relation)。若吾人在開始執行工作 t_j 之前必須先將工作 t_i 完成，則 DAG 中代表工作 t_i 與 t_j 的節點之間必定存在著一個有向邊。為了方便說明起見，我們將本文中所將探討的專案抽象化為一個具有 5 維度特性的系統 $-(T, E, l, r, p)$。系統中各個維度的定義說明如下：

- $T = \{t_1, t_2, ..., t_{|T|}\}$ 為專案中所有工作的集合，$|T|$ 則表示為此集合中所有工作個數。
- $E = \{e_{ij}\}$ 為 DAG 中所有有向邊的集合，e_{ij} 表示工作 t_i 與工作 t_j 之間具有先後順序關係之限制。在此限制條件下，我們稱 t_i 為 t_j 的立即前置工作，而 t_j 則被稱為是 t_i 的立即後繼工作。對於所有屬於工作 t_i 的立即前置工作所成的集合，我們以 $PRED_i$ 表示；而 $SUCC_i$ 則用以表示工作 t_i 的所有立即後繼工作所成的集合。若有一工作 t_i，其 $SUCC_i$ 的內容為空集合，我們稱之為起始工作。若有一工作 t_j，其 $PRED_j$ 的內容為空集合，我們稱之為結束工作。在不失一般性的情況下，我們假設每一個 DAG 中都各只有一個起始工作與結束工作，並將之分別表示為 t_{entry} 與 t_{exit}。顯而易見地，t_{exit} 的完成時間即代表整個專案的完成時間。
- l 為整個專案的完成期限。
- r 為在期限內完成專案所能獲致的收益。
- p 用以表示專案延遲時每一單位延遲時間所必須付出的成本。

一般而言，在專案起始之初，專案經理可先決定出如表 1 所示之專案中所包含的工作，以及各個工作之間執行先後順序的限制。根據表 1 所獲得之相關資訊，我們可以得到如圖 1 所示之相對應的 DAG。

由於我們必須將專案中的工作分派給最為合適的工作團隊來執行，因此除了考慮專案中各個工作彼此之間的執行順序限制條件之外，我們還必須進一步地針對工作團隊之間的關係予以抽象化的定義。吾人可將所有意欲參與專案執行的工作團隊描述成一個具有 3 維度特性的系統 $-(G, [d_{ij}], [c_{ij}])$。此一系統中各維度特性與相關定義說明如下：

- $G = \{g_1, g_2, ..., g_{|G|}\}$ 為專案中所有參與工作團隊的集合，$|G|$ 表示為此集合中所有工作團隊的個數。
- $[d_{ij}]$ 為一 $|T|\times|G|$ 大小的矩陣，d_{ij} 表示工作團隊 g_j 執行工作 t_i 所耗費的時間。
- $[c_{ij}]$ 為一 $|T|\times|G|$ 大小的矩陣，c_{ij} 表示工作團隊 g_j 執行工作 t_i 所必須支付的成本。

當 g_j 無法勝任 t_i 時，則令 c_{ij} 為無限大。

現假設 s 為某專案在排程之後所得的結果，則該專案中的工作 t_j 基於 s 所得到的完成時間 $finifh(t_j)$ 在數學上的定義可被表示為：

$$finish(t_j) = \max_{\forall t_i \in PRED_j} (finish(t_i)) + \sum_{k=1}^{|G|} d_{jk} x_{jk} \quad (1)$$

其中，x_{jk}=1 表示 s 將工作 t_j 分派給團隊 g_k 執行，反之，x_{jk}=0。基於以上所描述的符號與定義，我們可以進一步地得出求解此一問題的目標函式如下：

$$\max_{s} Z(s) = r - \sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij} x_{ij} - p[finish(t_{exit}) - l]^{+} \quad (2)$$

其中，$[a]^{+}$ 表示 $\max\{0, a\}$。本文之目的便在於根據上述的目標函式設計出適宜的演算法則，進而找出合乎要求的工作排程結果，使得專案收益在扣除各個工作執行成本以及懲戒金額之後可以達到最大值。

三、設計方法

本節在一開始將先就我們所提出的演算法做概要性的說明，其內容包括問題編碼的方式、初始解的產生、適存度的定義、交配與突變程序的特性、以及選擇機制的設計等相關議題介紹。接著，我們將詳細說明我們所提出的交替式順序交配運算子的運作方式，並就其特性進行分析。最後，我們將說明如何將模擬退火方法 (simulated annealing，SA) [8] 的精神整合於突變程序的設計之中，用以強化突變運算的效能。

(一) 設計方法綜覽

本論文中，我們以基因演算法的基本架構來解決委外作業專案管理的工作匹配與排程問題。我們藉由排程字串 (scheduling string，SS) 與匹配字串 (matching string，MS) 二者的組合來表示一個染色體 [13]。其中，排程字串係用來表示 DAG 中各個工作節點的執行先後順序；而匹配字串則是表示某一工作交由哪一個特定的工作團隊所執行。圖 2 是根據圖 1 所產生出的一個染色體實例。由圖中我們可以得知各個工作的執行順序以及分派執行的狀況。例如：工作 t_1 和 t_2 之執行係各由工作團隊 g_1 和 g_3 所負責；而工作團隊 g_2 則是以循序的方式依次執行工作 t_3、t_5、t_9 以及 t_{12}。

一旦決定了染色體的編碼方式之後，下一步便是進行初始族群的建構。在我們建構初始族群的過程中，每一個染色體中的排程字串內容是經由隨機產生一組符合 DAG 拓撲排序的工作順序所決定；而每一個染色體中的匹配字串內容則是利用隨機的方式將每一個工作指派給任意的工作團隊而產生。透過這樣的機制，我們不但可以保證初始族群中的每一個成員均為合法表示的染色體，同時也可以確保問題領域中所有的解都有被產生的可能。此一表示法具有以下的性質：

性質1. 排程字串中基因內容的排列順序必須與 DAG 中工作節點的拓撲順序相符合。

性質2. 排程字串中的每一個基因內容在該字串中僅能出現一次。

性質3. 匹配字串中的每一個基因內容允許與該字串中其他基因的內容相同。

性質4. 排程字串與匹配字串的長度與 DAG 中工作節點的數目相同。

在以上的這些特性中，性質 1 利用拓撲排序的特性讓工作的執行不會違反執行先後順序的限制條件；性質 2 則是說明了一個工作僅能夠讓一個處理單元執行，而一個處理單元在同一時間內也僅能執行一個工作；性質 3 表示每個處理單元在完成它所執行的工作後，可以再被指派處理其它工作；性質 4 則是確保在排程的過程中每一個工作只被執行一次。

我們還針對問題的特性提出一個名為交替式順序交配 (Alternative Order crossover，AOX) 運算子。AOX 運算子比傳統的單一切點順序交配 (order crossover，OX) 運算子 [10] 具備較高的有效性，可避免無效的交配運算。除此之外，為了避免染色體在突變之後因為解的品質不佳而無法存活至下一世代 (亦即，產生無效的突變運算)，我們在突變程序的設計中採用了模擬退火方法的概念，用以強化突變程序的效能。

當交配與突變程序進行完成，我們必須根據每一個染色體的適存度對所有的染色體進行評估。我們利用以下的定義決定染色體 S 的適存度：

$$fitness(S) = \frac{Z(S)_{avg}}{Z(S)} \quad (3)$$

其中，$Z(S)_{avg}$ 是族群中所有染色體相對應於問題特性所獲致利益的平均值。之後，我們採用輪盤法則 (roulette wheel selection) 來決定哪些染色體將會存活至下一個世代。我們也藉由精英策略 (elitist selection) 的原則來確保每個世代中最好的一個染色體必定會被選取而進入下一個世代。當某一個染色體已經被選取進入下一個世代之後，它將不會於本世代中再次被選取，以避免在該染色體在下一個世代中

重複出現，進而降低了下一世代族群的多樣性。

(二) 交配程序設計

在委外作業專案排程問題的求解過程中，本文所提出的 AOX 運算子主要是針對染色體中的排程字串加以操作。此一運算子必須要能確保交配之後所產生的子代能滿足工作執行先後順序的限制條件。此一運算子的運作流程如下：

步驟1. 由族群中隨機取出兩個參與交配的父染色體 P_0 與 P_1，並令交配後所產生的子代分別為 C_0 與 C_1。

步驟2. 藉由隨機的方式決定交配切點的個數 k ($2 \le k < n$)，並以 CP_1、CP_2、 ...、CP_k 分別表示依排列順序先後出現的交配切點。其中，CP_1、CP_2、 ...、CP_{k-1} 在染色體中的位置係以隨機的方式決定；而 CP_k 的位置則是固定於染色體中最後一個基因的位置。

步驟3. 令 $i = 1$，反覆執行以下步驟直到 $i > k$ 為止：

(1) 將 $P_{1\text{-}i\backslash 2}$ 與 $P_{i\backslash 2}$ 中位於 CP_i 之前尚未出現於子染色體中的基因分別複製並依序加入 C_0 與 C_1 之中。

(2) $i = i + 1$。

步驟4. 產生出新的子代染色體 C_0 與 C_1。

根據 AOX 的運作方式，我們可以得到以下的論證結果：

[定理] 在父染色體 P_1 與 P_2 皆為合法表示的前提之下，經由 AOX 運算子的操作而產生的子染色體 $C_{offspring}$ 必定符合工作執行先後順序的限制條件。

[證明] 隨機選取 $C_{offspring}$ 中的兩個基因 g_i 與 g_j，並假設 g_i 的位置在 g_j 之前，若吾人以 $pos(C_{offspring}.g_i)$ 表示基因內容 g_i 在 $C_{offspring}$ 中的位置，則可得知 $pos(C_{offspring}.g_i) < pos(C_{offspring}.g_j)$。同時，我們也假設基因 g_i 在父染色體中隸屬於交配切點 CP_m 的區段；而基因 g_j 在父染色體中則是隸屬於交配切點 CP_n 的區段。以下，我們分兩種情況探討 g_i 與 g_j 之間的關係。

Case 1. g_i 與 g_j 源自於相同的父染色體。

假設 g_i 與 g_j 源自於相同的父染色體 P_0。在 $pos(P_0.CP_m) = pos(P_0.CP_n)$ 的情況下，如果 $pos(P_0.g_i) > pos(P_0.g_j)$，則因為隸屬於相同區段的基因係由前往後依序加入子染色體中，所以可得出 $pos(C_{offspring}.g_i) > pos(C_{offspring}.g_j)$ 的結果。此與已知條件相互矛盾。因此，必定存在 $pos(P_0.g_i) < pos(P_0.g_j)$。另一方面，在 $pos(P_0.CP_m) <> pos(P_0.CP_n)$ 的情況下，如果 $pos(P_0.CP_m) > pos(P_0.CP_n)$，則因為 $pos(P_0.CP_m) \ge pos(P_0.g_i) > pos(P_0.CP_n) \ge pos(P_0.g_j)$，最後將得到 $pos(C_{offspring}.g_i) > pos(C_{offspring}.g_j)$ 的結果。此亦與已知條件相互矛盾。因此，必定存在 $pos(P_0.CP_m) < pos(P_0.CP_n)$。又因為 $pos(P_0.g_i) \le pos(P_0.CP_m) < pos(P_0.g_j) \le pos(P_0.CP_n)$，於是可推導出 $pos(P_0.g_i) < pos(P_0.g_j)$。

Case 2. g_i 與 g_j 源自於不同的父染色體。

在此一情況下，我們可以得知 $pos(P_0.CP_m) <> pos(P_0.CP_n)$。現假設 g_i 與 g_j 分別源自於不同的父染色體 P_0 與 P_1。若 $CP_m < CP_n$，則必定存在 $pos(P_0.g_i) < pos(P_0.g_j)$；否則當吾人依序將基因加入 $C_{offspring}$ 之中時，將會出現 $pos(C_{offspring}.g_i) > pos(C_{offspring}.g_j)$ 的結果，此與已知條件相矛盾。在另一方面，若 $pos(P_0.CP_m) > pos(P_0.CP_n)$，則必定存在 $pos(P_1.g_i) < pos(P_1.g_j)$，其原因如前所述。

根據以上之論述，我們可以歸納出子染色體中任意兩個基因之間的先後順序關係必存在於某一個父染色體之中。亦即，$C_{offspring}$ 最後呈現的排程結果必定符合工作執行先後順序的限制條件。 □

相較於傳統的交配法則 (例如：局部對應交配 (partially-mapped crossover，PMX) 與循環交配 (cycle crossover，CX)) [10]，AOX 的優點在於交配後所產生子代的基因內容必定符合拓樸順序，因此不需要再進一步地調整內容。故而在執行效能上，AOX 預期會有較佳的表現。再者，相較於單一切點的 OX 運算子，本文中所提出的 AOX 運算子係以多重切點的方式處理染色體的交配程序。此一處理方式的優點在於交配後所產生出來的子代與父染色體相同的機率較低，可提昇交配運算的有效性。

(三) 突變程序設計

由於 AOX 交配運算子只是針對工作的執行順序予以考量，因此在突變程序的設計部分我們採用兩種機制來加以處理。其中，第一種機制係採用模擬退火方法的概念來處理工作指派問題。假設欲進行退火處理的染色體為 S，其所具有之能量為 $Z(S)$，此一機制的運作流程如下：

步驟1. 設定初始溫度 H_0、循環次數 N，最終溫度 H_{end}，並且讓目前溫度 $H = H_0$。

步驟2. 令 $n = 0$，執行以下步驟直到 $n \ge N$ 為止：

(1) 以隨機的方式由 S 中的 MS 字串選取一個基因並改變其值。將所得之結果視為 S'。

(2) 計算 S 與 S' 二者之間的能量差 $\Delta E = Z(S') - Z(S)$。若 $\Delta E \leqq 0$，則令 $S = S'$；反之，則由機率 $exp(-\Delta E/k_B T)$ 決定是否接受 $S = S'$。其中，k_B 為

波茲曼常數。

步驟3. 令 $H' = \alpha H$ (α 為降溫速率，$0<\alpha<1$)。

步驟4. 檢查 H' 是否大於 H_{end}。若是，則令 $H = H'$ 並執行步驟 3；反之，獲得 S 為最終解。

根據以上的運作方式，我們不難發現：在模擬退火方法中所欲搜尋的解空間大小為 $|G|^{|T|}$。為了避免此一過程所花費的時間隨著處理單元或工作節點個數的增加而急遽增加，我們將初始溫度 H_0 設定為 $\sqrt{|T|\ln|G|}$。此一動態初始溫度設定的最大優點在於當 DAG 圖形愈趨複雜時，我們可以有效地避免 H_0 溫度過大，造成整個演算法中退火過程所花費的時間過長；而當 DAG 圖形較為簡單時，此一設定也能確保 H_0 的溫度不會過小，導致退火過程無法發揮功用。除此之外，為了彌補模擬退火方法只考量工作匹配情形的不足，我們則是在第二種突變機制中採用傳統的插入式突變 (insertion mutation，IM) 運算子 [13]。其具體作法是將我們在退火程序所新產生出來染色體的 SS 字串中隨機選取一個基因進行移動。在移動的過程中，我們必須判斷所得之結果是否符合拓撲排序，如果符合，即產生新的子代。

四、效能評估

我們以專案時程管控為目標設計了一個測試案例產生器，用以根據各種不同的參數來產生各種不同類型的專案，進而產生實驗所必須的測試案例以有效評估我們所提出演算法之效能。此一產生器允許使用者輸入以下相關參數用以建構出所需要的測試案例：

- $|T|$：工作節點數目。
- $|G|$：工作團隊數目。
- Deg_{avg}：工作節點的平均分支度。亦即，所有工作節點平均的子節點個數。
- Deg_{sd}：平均分支度之標準差。
- $Power_{avg}$：工作團隊之平均執行能力。
- $Power_{sd}$：工作團隊執行能力之標準差。

藉由此一測試案例產生器為工具，我們依據表 2 所列的相關參數值產生 60 種不同的參數組合。針對每一種參數組合，我們以隨機的方式產生 5 個測試案例，所有共有 300 個不同情況的工作匹配與排程案例，用以做為我們在進行演算法效能評估時的測試資料。

由於每個演算法執行一個迭代所需要耗用的時間不盡相同，若以每次執行演算法之迭代數作為演算法停止的依據，將失去其公平性。因此，我們在此以設定演算法之執行時間作為演算法停止之標準。對每一個測試案例而言，我們以 $\sqrt{|T|\ln|G|}$ 之值作為每個演算法執行該測試案例的單位時間。在後續的實驗中，我們將分別紀錄各演算法對每個測試案例執行不同單位時間後所得之結果。吾人亦可由此觀察執行時間的長短對不同演算法的影響。此外，我們針對基因演算法所設定之交配機率為 0.8，突變機率為 0.05，族群大小為 20。

本實驗中所欲評估之演算法，除了我們所提出的方法 (TOX+SA/IM) 之外，還包括了模擬退火方法 (SA) 以及兩種不同的基因演算方法。其中一種以 GA 為基礎之方法相類似於我們所提出的演算法，但其突變程序卻只包含了 IM 運算子的運作功能。此一方法主要是用來衡量使用模擬退火程序於遺傳演算法中對解決委外作業專案排程問題之影響，在此我們以 "AOX+IM" 表示之。另一種則為一般常見之遺傳演算法，其交配程序所使用的運算子為順序交配，突變程序則為插入式突變，我們以 "OX+IM" 符號表示之。在這兩種基因演算法中，其所使用的選擇與評估等其它程序均和我們提出之演算法相同。另一方面，我們所欲評估之模擬退火方法則是以一個初始解為起點開始尋找其鄰近解，產生鄰近解的作法為改變問題解中執行某一工作之工作團隊或改變其中一工作之執行順序 (此類似於遺傳演算法中之插入式突變)。

在實驗的過程中，我們會計算每一個測試案例由不同演算法執行經過單倍、十倍與二十倍單位時間後的排程結果。我們以正規排程長度 (normalized scheduling length，NSL) 作為衡量演算法優劣的基準。所謂的正規排程長度係指測試案例經排程後所得的排程長度與該案例之排程長度下限的比值；而排程長度下限則是指在 DAG 中選取一條包含起始工作與結束工作且工作量總和最大之路徑，然後將此路徑上所有工作的工作量加總後除以工作團隊中最大工作能力所得到的值。圖 2 至 圖 7 所呈現之內容係各種演算法在不同參數的影響下所評估而得的效能。

除了以上的實驗之外，我們也將所有測試案例在排程後所得之結果加以統計，以便進行不同演算法之間的全面性比較。我們在表 3 中標示出比較之後的結果。矩形中所表示的數據為矩形上方與矩形左方演算法的比較結果。">"、"=" 和 "<" 分別表示左方演算法優於、等於、劣於上方演算法的測試案例個數。以 AOX+ IM 和 OX+IM 為例，AOX+ IM 所得之結果計有 186 次優於 OX+IM、1 次與 OX+IM 所得之結果相同、其中有 113 次所得的結果較差。整體而言，我們所提出的方法遠遠優於 AOX+IM、OX+IM 和 SA。

五、系統實作

為了驗證本文所提出方法的實用性，我們建構了一個名為 CPMS 的協同式專案管理系統 (Collaborative Project Management System)。CPMS 允許使用者透過瀏覽器從不同的機器經由網路鏈結至系統所提供之一致化使用者介面。在此一介面中，企業組織之專案經理及相關人員可藉由系統所提供之協同處理功能，根據其需要建立一個圖形化表示之專案，並針對專案中之相關工作是否為委外作業進行設定。一旦設定完成，提供服務的供應商便可藉由全球資訊網瀏覽該專案所包含的委外作業工作內容，進而根據該工作團隊之專長、能力及意願，針對意欲承接之委外作業透過系統所提供的操作介面提出申請並提供完成該委外作業所需之相關資訊。最後，系統將彙整所有供應商所提供的相關資訊進行評估，針對專案發展時所必須考慮的各個因素(例如：成本、專案發展時程、人力因素等)，擇優選出合適的承包廠商，以支援企業組織決策管理之用。圖 8 與圖 9 所示分別為 CPMS 的概念模型與實作執行範例。

六、結論

本論文中，我們提出了一個在解決委外作業專案排程問題的演算法。此一方法結合了基因演算法與模擬退火方法的概念，用以確保我們在處理各種不同類型的專案時可以有效率的獲得高品質的解。

我們設計出一個名為 AOX 的交配法則。此一交配法則不論是在理論上或是在實際實驗中所得到的結果均優於傳統的 OX 交配法則。其原因在於 AOX 運算子所產生子代的多樣性較高，因此有較高的機率可避免產生與父染色體具有相同基因內容的子代。除此之外，我們還將模擬退火方法引入基因演算法的突變程序之中。藉由模擬退火程序優異的區域搜尋能力，我們可以避免染色體在經過突變程序後的解的品質變差，導致在選擇程序時未被挑選到進入下一世代，喪失突變程序原本跳脫局部最佳解的用意。根據實驗結果顯示，我們所提出的方法有高達 94.8% 的機率能獲得較其他方法為佳的排程效果。AOX+SA/IM 不似其它方法必須經過長時間的收斂程序，其在短時間內就能夠產生效果極佳之排程結果。同時，我們也利用全球資訊網為平台建構了一個協同式委外作業專案管理系統。此一系統驗證了本文所提出方法在實際應用上的可行性。

有鑑於在逐漸全球化的經濟體系中，未來執行不同專案工作的團隊其所在地點也不盡相同。因此，在專案工作執行的過程中，吾人應將工作之間所需知識移轉的依賴程度納入專案排程時的考量。此一依賴程度可由應用軟體、品質管理、開發標準、企業文化等因素來加以衡量。我們未來將針對此一議題加以研究，並將研究所得結果整合於 CPMS 中，俾使 CPMS 的功能更加完善。

七、誌謝

本文之成果承蒙國科會 (NSC 91-2213-E-212-012) 專案研究經費補助，深表感謝。

八、參考文獻

[1] 李永山, "台灣中小型企業資訊作業委外決策之探討," 台灣大學商學研究所碩士論文, 1999.

[2] 林信惠, 黃明祥, 王文良, 軟體專案管理, 智勝出版社, 2002.

[3] 蘇祐毅, "資訊委外關鍵因素之探討," 台灣大學商學研究所碩士論文, 1999.

[4] B. I. Blum, Software Engineering: A Holistic View, Oxford Univ. Press, 1992.

[5] M. Gen, and R. Cheng, Genetic Algorithms and Engineering Design, Wiley, 1997.

[6] J. H. Holland, Adaptation in Natural and Artificial Systems, Univ. of Michigan Press, 1975.

[7] H. Kerzner, *Project Management: A Systems Approach to Planning, Scheduling, and Controlling*, Van Nostrand Reinhold, 1995.

[8] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by Simulated Annealing," Science, 220, pp. 671-680, 1983.

[9] R. Klepper, and W. O. Jones, *Outsourcing Information Technology, Systems, and Services*, Prentice Hall, 1999.

[10] Y. Kwok and I. Ahmad, "Efficient Scheduling of Arbitrary Task Graphs to Multiprocessors Using a Parallel Genetic Algorithm," *J. Parallel and Distributed Computing*, vol. 47, no. 1, pp.58-77, November 1997.

[11] L. C. Liu and E. Horowitz, "A Formal Model for Software Project Management," IEEE Trans. Software Eng., vol. 15, no. 10, pp. 1280-1293, October 1989.

[12] W. J. Stevenson, *Production/Operations Management*, McGraw Hill Company, 1999.

[13] L. Wang, H. J. Siegel, V. P. Roychowdhury, and A. A. Maciejewski, "Task Matching and Scheduling in Heterogeneous Computing Environments Using a Genetic-Algorithm-Based Approach," *J. Parallel and Distributed Computing*, vol. 47, no. 1, pp.8-22, November 1997.

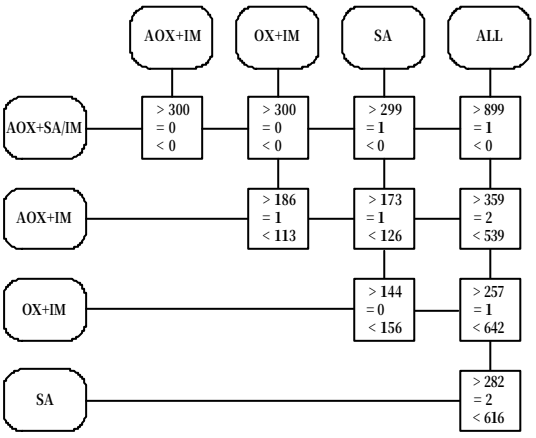
表 1. 專案工作列表

工作	立即前置工作	工作	立即前置工作
t_1	--	t_7	t_1
t_2	--	t_8	t_4
t_3	t_1	t_9	t_3, t_6
t_4	--	t_{10}	t_5, t_7
t_5	t_2, t_4	t_{11}	t_9
t_6	t_1, t_2	t_{12}	t_{11}

表 2. 測試案例相關參數設定

參數名稱	參數數值
$\|T\|$	20、40、...、200
Deg_{avg}	$\frac{\sqrt{T}}{2}$、$\sqrt{T}$、$2\sqrt{T}$
Deg_{sd}	$0.2 \times Deg_{avg}$
$\|G\|$	8、16
$Power_{avg}$	1
$Power_{sd}$	0.4

表 3. 測試案例排程結果之全面性比較



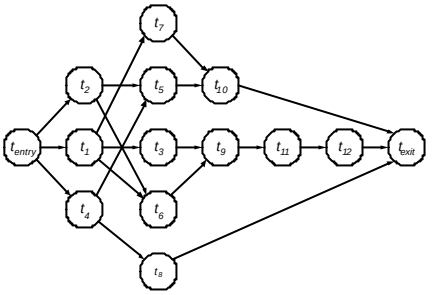


圖 1. 相對應於表 1 的 DAG

t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}	t_{12}
g_1	g_3	g_2	g_1	g_2	g_1	g_3	g_3	g_2	g_1	g_1	g_2

圖 2. 基於圖 1 的染色體編碼範例

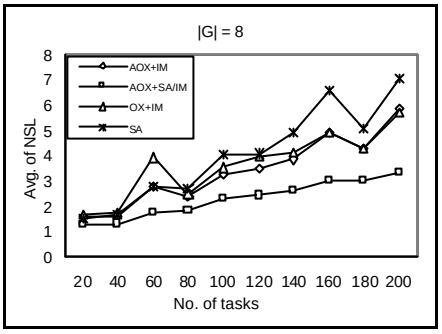


圖 2. 單倍執行時間效能評比 (|*G*|=8)

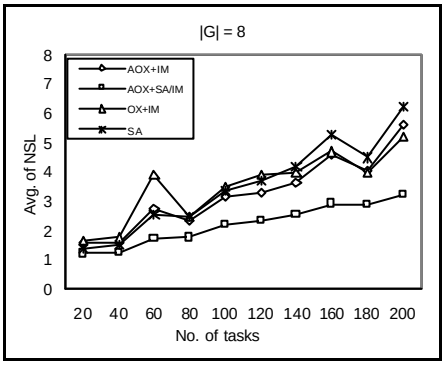


圖 3. 十倍執行時間效能評比 (|*G*|=8)

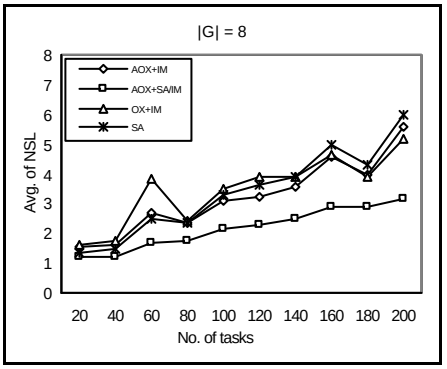


圖 4. 二十倍執行時間效能評比 (|*G*|=8)

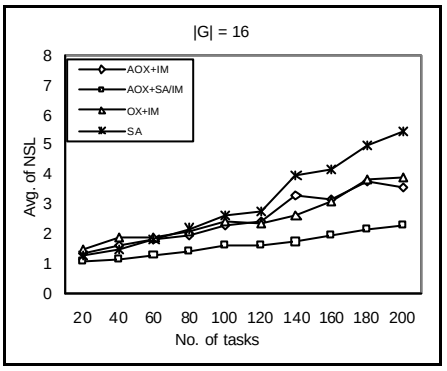


圖 5. 單倍執行時間效能評比 (|*G*|=16)

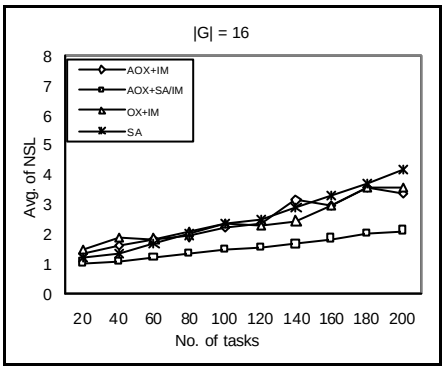


圖 6. 十倍執行時間效能評比 (|*G*|=16)

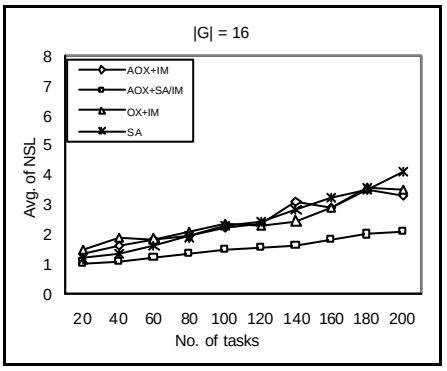


圖 7. 二十倍執行時間效能評比 (|*G*|=16)

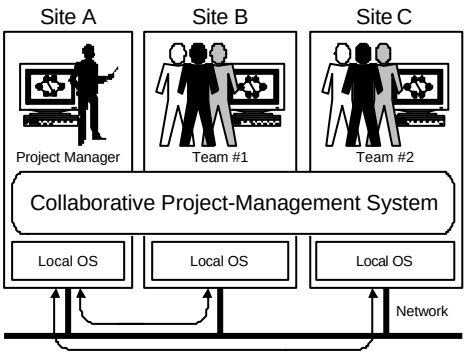


圖 8. CPMS 之概念模型

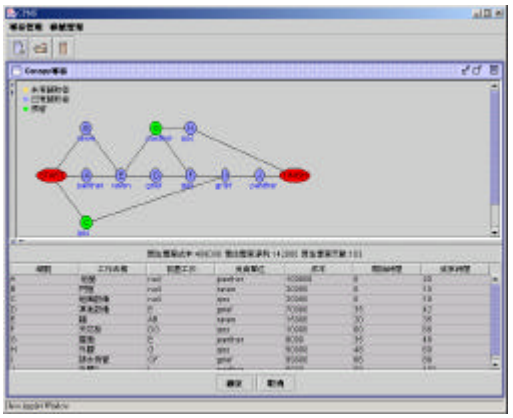


圖 9. CPMS 執行範例