輔仁管理評論

中華民國 97 年 9 月，第十五卷第三期，1-32

# 考慮資源限制與穩健性之專案排程研究

黃榮華・楊長林・劉士豪*

(收稿日期：96 年 12 月 21 日；第一次修正：97 年 1 月 21 日；
接受刊登日期：97 年 5 月 16 日)

**摘要**

在有限資源的專案排程問題 (resource constrained project scheduling problem, RCPSP) 中，普遍存在一個事實，對於已經計劃好的排程，常會受到一些不可控制的因素干擾而延工，使得專案往往無法在承諾的完工時間之內完成。本研究將穩健性 (robustness) 的觀念引入有限資源專案排程中，動態 (dynamic) 賦予平行作業先後順序，並衡量平行作業集合中各作業活動的寬裕時間 (slack time)，藉以篩選出寬裕時間獨立性較大的排程方案，以有效縮短專案延工率。

本研究針對作業活動的持續時間 (duration) 可能在專案計畫實際運作的過程中發生變動，而導致延工的問題，提出了一套求解架構，嘗試將平行作業的排工順序與非平行作業分離，採用禁忌搜尋演算法 (tabu search algorithm, tabu) 配合 ACTIM 法則選擇的起始解，在平行作業發生資源衝突時，搜尋出最適作業排程，以更高之穩健性降低專案的延工率。演算法之資料測試係取自題庫 PSPLIB (project scheduling problem library) 所提供的四種不同作業數 (30、60、90、120) 類型之問題。考慮三種不同增幅之作業時間，分別在資源限制爲 10 單位與 15 單位狀態下，共測試了 720 組樣本，所有問題的求解品質皆優於單純使用 ACTIM 法則之結果。

關鍵詞彙：有限資源專案排程問題，穩健排程，禁忌搜尋法

## 壹. 導論

在全球化的競爭壓力下，愈來愈多企業致力於快速回應顧客少量、多樣化的需求。但在缺乏大量生產 (mass production) 的規模效益下，批量生產 (batch production) 模式，將直接突顯成本控制的重要性，影響包括企業長期獲利能力與市場競爭力。然而，如何在有限時間內將所有的有形與無形資源進行最適的分配是典型的專案排程問題。於是，企業開始漸漸的重視專案管理的品質，藉由追求有效的專案規劃與控制來降低專案執行成本。

* 作者簡介：黃榮華，輔仁大學管理學研究所副教授；楊長林，輔仁大學企業管理學系副教授；劉士豪，輔仁大學管理學研究所研究生。

過去的學者普遍作法是假設資源的使用量在排程的過程之中是明確的而且沒有總資源上限,而排程的目標實現亦是建立在專案作業活動可以如期完成的假設下達成。但事實並非如此，在大多數的情況下，專案在開發階段，各作業的起迄時間會因爲非預期的因素產生而需要調整。譬如：專案經理人低估了各作業的資源使用量、非預期的設備維護……等，改變原始作業活動的參數設定，如此，將增加專案發生延工的機率。傳統的作法雖然可以根據要徑法的求解模式，衡量出各作業的寬裕時間，以寬裕時間來降低非預期因素發生所造成的延工率。

但在資源有限的前提下，當資源發生衝突時，平行作業未必能夠同時處理，導致延後排工的作業勢必消耗要徑法求解模式下的寬裕時間，降低排程解的預防能力或增加整體專案的完工時間。準此，本研究提出穩健性排程解的想法，視作業活動之實際開始時間爲一動態變化情形，探討作業活動實際開始時間之決定，並將目標作業的最晚完工時間減去其實際完工時間，以取得各作業獨立性寬裕時間 (robust)，如何最大化獨立寬裕時間使專案穩健性最佳實爲一重要課題。當各作業的持續時間非預期的增加時，藉由其獨立性的寬裕時間來吸收，有效降低非預期因素所帶來的衝擊。

有限資源專案排程問題，是一種組合最佳化問題 (combinatorial optimization problem)，在求解上可以分爲兩類：第一類爲建立數學模式求出最佳解法 (optimal procedures)；第二類爲求得近似最佳解的啓發式解法 (heuristic procedures)。雖然最佳解法有求解得最佳目標值的優點，但現行之最佳解法對於一個包含多個作業的複雜專案而言，其效率難以令人接受。於是，許多學者開始嘗試採用啓發式解法並搭配不同優先法則來處理組合最佳化問題，如基因演算法 (genetic algorithms, GA)，模擬退火法 (simulated annealing, SA) 及禁忌搜尋法，雖然有不少研究已成功應用這些演算法處理組合最佳化問題，許多文獻仍顯示其在排序問題上禁忌搜尋的效果優於其它方法，如學者 Sinclair (1993)、Marett & Wright (1996)、Dorn *et al.* (1996)、Murata & Ishibuchi (1994)、Lee (2001)。

因此，本文考慮資源限制以穩健性與整體專案完工時間爲衡量準則之專案排程爲研究主題，主要達成目標如下：

1.考量最小化閒置資源，使專案可以在最短的時間內完成。

2.建立衡量各作業獨立性寬裕時間的方法，搭配以禁忌搜尋法，動態決定各作業的實際開始時間，以求解考慮穩健性及專案完工時間雙準則之有

限資源專案排程問題。

3.利用國際題庫 PSPLIB 測試例題，驗證演算法的正確性與效率。

# 貳、 文獻探討

## 一、專程排程問題

專案排程問題開始被熱烈討論的期間可以追朔自 1950 年代末期，美國北極星飛彈計畫小組及美國杜邦公司爲了提供具有分析性資訊相繼在 1958 年與 1975 年分別提出了計畫評核術 (program evaluation and review technique, PERT) 與要徑法 (critical path method, CPM)，然而，在後續學者廣泛研究與應用下，這兩種網路分析工具已經普遍被用來進行專案的規劃與控制，有效排專案排程中各作業活動的開始時間。Davis & Patterson (1975) 曾對美國 400 大營造工程公司 (construction firms) 進行問卷調查，在回收的 235 份有效樣本之中，約有 80% 的 (約 190 家) 營造工程公司使用要徑法進行專案的規劃與控制，可見 CPM 及 PERT 受歡迎的程度，但其中隱含「 資源充分供應」 的假設與現實環境不符，例如在實際組織運作生產製造環境，從人力與物力持續投入的角度觀察，再再的顯示資源是有限的，使得原本在網路圖上看似沒有先後順序關係的平行作業，因爲資源限制而產生先後順序關係，這就是所謂「有限資源專案排程問題」。

然而，隨著 Kelley (1963) 首次提出有限資源專案排程問題，說明專案在執行過程中，資源衝突的現象會不斷發生，進而產生了資源分配上的問題，因此開始獲得學術界熱烈討論。Brand *et al.* (1964)、Patterson & Huber (1974)、Talbot & Patterson (1978) 以最小化總專案工期爲目標求解有限資源專案排程問題。Talbot (1982) 是以最小化總專案成本爲目標求解有限資源專案排程問題。Kurtulus & Davis (1982)、Kurtulus & Narula (1986)、Boctor (1990) 以最小化總專案延遲爲目標求解有限資源專案排程問題。

近年來，爲了能夠符合近實務上需要，後續國內外學者紛紛將研究的觸角從傳統的效率求解延伸到排程解的彈性及穩健度，例如 Bowers (2000) 用浮時 (float time) 來引申排程彈性的觀念，說明管理者給予較重視的作業活動較大的浮時，使該作業有較充裕的緩衝時間來回應非預期的變動；Al-Fawzan & Haouari (2005) 強調，好的排程解除了要能夠滿足基本限制之外，還要多一個

穩健 (robustness) 的衡量標準，來降低專案因爲不可控制因素發生而帶來的成本; Wang (2005) 引入 CSP (constraint satisfaction problems, CSP) 模型，將排程過程中所發生的不可控制因素干擾，直接視爲限制式的增減變動，修復可能因此中斷的排程解，增加原來排程工具的彈性。

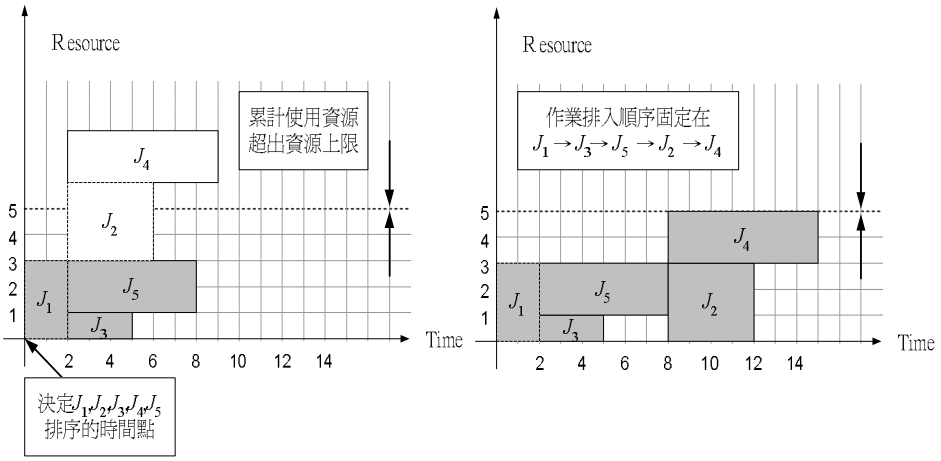
## 二、專案排程問題的求解方法

過去文獻在求解 RCPSP ，主要集中在數理模式 (mathematical models) 與啓發式 (heuristic models) 的解法。排程問題在數學規劃中屬於組合性最佳化問題。然而，大多數組合性最佳化問題具有 NP-Hard 的特性，表示，求解該類問題的解題時間會隨著問題的複雜度增加而呈現指數成長，於是，爲了要在設定的時間內滿足專案管理者多元化的需求，必須要尋求更有效率的求解方法。在過去 40 年，已經有許多學者陸續使用啓發式解法取代數理模式以求解有限資源專案排程問題，其具有時效性的次佳解在實務應用上往往較不具時效性的最佳解更有意義。

啓發式求解方法最早可以追溯至 Kelley (1963) 提出的串列法 (serial method) 與平行法 (parallel method) 兩種求解過程。然而，後進的學者大多數在求解方法上都是繼承 Kelley (1963) 的想法，再加上自己的觀點予以調整、修改。兩種方法的求解邏輯如下：

### (一)串列法：

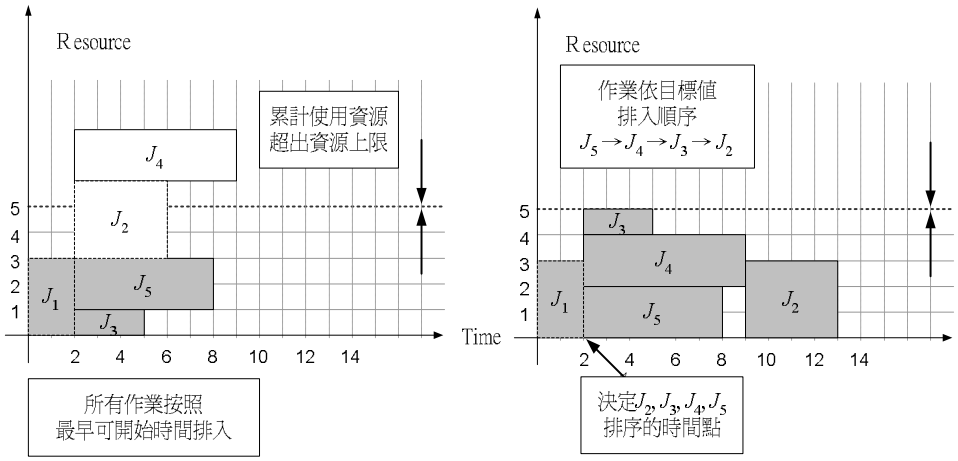
強調在排程之初，即爲專案中各項作業先按照某種優先規則 (prior rule) 建立一個排程優先次序表，然後，再按照此優先次序表進行排程，一次排入一個作業，在排入作業時，必須在不違反作業先行關係及資源限制的條件下儘早排入，如圖一所示：




圖一　串列法說明圖

## (二)平行法：

與串列法最大的差異在於，在排程之初並沒有一個排程優先次序表，來指引每一個作業的排程順序，取而代之的是，依時進展的排入作業，每次以一單位時間往完工時間移動，對於擁有相同開始時點的作業，只要不超過資源限制皆可同時排入多項作業，唯當資源不足時再依平行作業優先規則來選擇延遲作業，並往下一個時點移動，如此不斷的重覆步驟，直到所有專案內的作業皆被排入為止，如圖二所示：




圖二　平行法說明圖

Kelley (1963) 所提出的串列法其最主要缺點在於排程之初即決定好各作業的優先次序，缺乏動態性，無法及時反應各作業在各排程時點的緊要程度，所以排程績效較平行法差，無法滿足專案實際運作的需求。學者 Boctor (1990) 曾對平行法及串列法進行比較，證實了平行法的求解效率優於串列法。所以往後的啓發式解法研究大多集中在平行法。

# 三、禁忌搜尋法

禁忌搜尋法最早由 Fred Glover 於 1977 年所提出應用於求解整數規劃上，一直到 1986 年才詳述禁忌搜尋法的基本原理與使用 Pirlot (1996)，之後便有許多學者根據禁忌搜尋法理論來求解 NP-Hard 的問題，例如零工式工廠排程 (job shop scheduling problem)、流程型工廠排程問題 (flow shop scheduling problem)、路徑指派問題 (path assignment problem)、銷售員旅行問題 (traveling salesman problem) 等問題。Taillard (1990)、Dell'Amico & Trubian (1993)、Murata & Ishibuchi (1994)、Lee (2001)、Malek *et al.* (1989)。學者 Sinclair (1993)、Marett & Wright (1996)、Nonobe & Ibaraki (2002)，也都曾經以禁忌搜尋法爲基礎求解工業工程各種組合性最佳化問題，均獲得不錯的結果。於是，本研究將使用禁忌搜尋法來求考慮資源限制穩健性之專案排程問題，以下針對禁忌搜尋法的組成要素進行文獻回顧。

禁忌搜尋法可以細分成四個組成要素：(1)鄰近搜尋(2)禁忌名單(3)解禁規則(4)停止條件，以下將針對此四個要素加以介紹。

## (一)鄰近搜尋 (neighborhood search)

所謂的鄰近搜尋，是藉由目前排程解的組成原素彼此交換，即將具有先後順序的作業活動彼此進行位置上的交換，使得一個解改變到另一個解過程，而在每次的反覆中，搜尋最優的鄰近解作爲新的接受解。假如新的接受解優於目前的最佳解，則將目前最好的解更新爲新的接受解，反之則保留目前的最佳解；不斷反覆此步驟直到滿足停止條件爲止。

## (二)禁忌名單 (tabu list)

所謂的禁忌名單，是用來記錄過去搜尋中每次發生解改變時的路徑，本質爲一個提供禁忌路徑限制容器的記憶架構。一般而言，禁忌名單越大則愈有機會跳出區域最佳解並貼近全域最佳解，但所需的記憶體空間也越大，且電腦

每次所需偵測時間越長，可提供鄰近搜尋的空間越小，這些現象將降低求解的效率。禁忌搜尋法就如同多數的啓發式演算法一樣，沒有所謂的最佳參數設定，換句話說，用來記錄禁忌名單的記憶架構，沒有最適的參數輸入，必須依問題的大小和複雜性決定。一般最多學者採用的禁忌搜尋策略爲短期記憶策略，又稱「簡單禁忌搜尋法」Glover & Laguna (1997)，其主要在說明存放禁忌名單的記憶結構採取一種短期的記憶，只存放最近的數次記錄，再配合先進先出法 (FIFO)，當取得下一個新的禁忌路徑時，就把禁忌名單中最舊的路徑捨去，如此可避免重複選到之前決策產生循迴圈的情形。準此，在求解過程中，除了可以避免重複回到上一次的解之外，另一方面仍然可以避免求解時受陷於局部最佳解的困境，此點是禁忌搜尋法與其他法則最大不同處。

### (三)解禁規則 (aspiration rule)

用來釋放一個被列在禁忌名單中的移動路徑，表示，此禁忌移動路徑所可獲得比目前最佳目標值還好的值。準此，爲增加求解上的效率，雖然該路徑已被列入禁忌名單中，但仍然可以透過解禁規則解除禁忌限制。

### (四)停止條件 (stopping criterion)

用來終止禁忌搜尋的條件，較常見的大約有下列四種方式：(1)允許之最大搜尋次數，(2)目標值持續未改善次數，(3)允許最長處理時間，(4)可接受之目標函數值。一旦搜尋達到這些條件，則 Tabu 會停止搜尋，取搜尋終止時點之最佳排程解作爲最終解。其中，大部分使用禁忌搜尋法的學者又以允許最大搜尋次數爲終止條件的最多，因爲使用此種方法保證在一段搜尋後會終止搜尋，且不因所使用電腦系統不同而有所影響。

## 參· 有限資源之專案排程問題

求解有限資源專案排程的方法，根據過去學者研究顯示，爲了爭取時效性，會採取啓發式解法進行求解。雖然無法保證可以找到全域最佳解，至少可以找到近似最佳解，且整體的求解品質也不會太差。儘管，使用啓發式解法來求解有限資專案排程問題的文獻已有許多，但隨著專案經理人的目標不同，求解問題的架構也將進行調整。爲了有效達到本研究的專案目標，考慮專案作業活動時間有隨機變動的可能，首先，根據要徑法求取各作業活動之最早開始時

間，了解整個專案各作業先後加工順序的脈絡，再進一步的依實際排程情況，更新各作業活動的實際開始時間，並配合其它決策變數的求解，完成考慮穩健性的專案排程。於是，在本節的第一部份，即針對 RCPSP 之基本假設及數學模式做一說明，接著再說明禁忌演算法求解 RCPSP 的過程。

# 一、基本假設與符號說明

## (一)基本假設

本研究針對第二章文獻中所敘的此類問題之特性進行假設，如下所示：

1.本研究僅考慮單一可恢復資源 (single resource problem, SRP)－人力資源的使用。

2.當部份作業之間存在著特定的優先順序關係時，目標作業之先行作業未完成前，該目標作業與其後續作業均不得開始進行。

3.單位時間 t 的資源上限爲已知常數。

4.任何時間點 t 沒有使用完的資源，均無法遞延到下一期使用。

5.各作業的作業時間爲一非負的正整數。

6.作業執行時不得分割或中斷且不可事先預製。

## (二)符號說明

本研究主題之主要相關符號如下所示：

$J_j$ = 編號 j 之作業，$j = 1,2,\cdots, n$；

$J_0$ = 虛擬起始作業；

$J_{n+1}$ = 虛擬終止作業；

$d_j$ = $J_j$ 所需要之持續時間，$j = 1,2,\cdots, n$，($d_j \geq 0$)；

$S_j$ = $J_j$ 之實際開始時間，$j = 1,2,\cdots, n$；

$F_j$ = $J_j$ 之實際完工時間，$j = 1,2,\cdots, n$；

$r_j$ = 執行 $J_j$ 每單位時間內，人力資源的需求量，$j = 1,2,\cdots, n$；

R = 每單位時間內，人力資源的上限;

b = 已知的專案可開工時間；

t = 排程時間 (schedule time) 點，$t = 1,2,\cdots, n$；

$Robust_j$ = $J_j$ 獨立性的寬裕時間，$j = 1,2,\cdots, n$；

$Robust = \cdot\sum_{j=1}^{n} Robust_j$

$C_{\max}$ = 整體專案實際的完工時間;

$ES_0$ = 虛擬起始作業的最早開始時間；

$ES_{n+1}$ = 虛擬終止作業的最早開始時間；

$ES_j$ = $J_j$ 最早開始時間，$j = 1,2,\cdots, n$；

$LS_j$ = $J_j$ 最晚開始時間，$j = 1,2,\cdots, n$；

$LF_j$ = $J_j$ 最晚完工時間，$j = 1,2,\cdots, n$；

$LF_{n+1}$ = 虛擬終止作業的最晚完工時間；

$\text{Pred}_j$ = $J_j$ 之立即前置 (immediate predecessors) 作業集合;

$\text{Succ}_j$ = $J_j$ 之立即後置 (immediate successors) 作業集合;

$w_1$ = Robust 權重;

$w_2$ = $C_{\max}$ 權重;

Z = $w_1 \cdot \sum_{j=1}^{n} Robust_j - w_2 \cdot C_{\max}$ 。

## (三)數學模式

### 目標式

$$Max \ \mathbf{Z} = w_1 \cdot \sum_{j=1}^{n} Robust_j - w_2 \cdot C_{\max} \quad \textbf{(1)}$$

### 限制式

$$ES_0 = \mathbf{b} \quad \textbf{(2)}$$

$$ES_j = \max\{F_i\} \ ; \ J_i \in \mathbf{Pred}_j \ ; \ i \neq j \ ; \forall\, i = 1,2,\cdots, n+1$$

$$\forall\, j = 1,2,\cdots, n+1 \quad \textbf{(3)}$$

$$S_j \geq ES_j \ ; \quad \forall\, j = 1,2,\cdots, n \quad \textbf{(4)}$$

$$F_j = S_j + d_j \;;\quad \forall j = 1,2,\cdots,n \tag{5}$$

$$C_{\max} = ES_{n+1} \tag{6}$$

$$LF_{n+1} = C_{\max} \tag{7}$$

$$LF_j = \min\{S_i\} \;;\; J_i \in Succ_j \;;\; i \neq j \;;\quad \forall \mathrm{i} = 0,1,\cdots,\mathrm{n}$$
$$\forall \mathrm{j} = 0,1,\cdots,\mathrm{n}\ \ Robust_j = LF_j - F_j \;;\; \forall j = 1,2,\cdots,n \tag{8}$$

$$\sum_{j=1}^{n} r_j \cdot k_{jt} \le R \quad ; \tag{9}$$

$$k_{jt} = \begin{cases} 1 & , J_j \text{ 於時間 } t \text{ 爲執行作業中} \\ 0 & , \quad \text{否則} \end{cases}$$

$$t = 1,2,\cdots,\ ES_{n+1} \tag{10}$$

**目標式說明：**

式(1)：最大化的專案獨立性寬裕時間 (健全性) 與最小化整體專案完工時間。

**限制式說明：**

式(2)：虛擬起始作業的最早開始時間等於專案的可開工時間。

式(3)：$J_j$ 的最早開始時間等於其前置作業集合中，最大的實際完工時間。

式(4)：$J_j$ 的實際開始時間必大於等於其最早開始時間。

式(5)：$J_j$ 的實際完工時間等於其實際開始時間與持續時間之和。

式(6)：整體專案完工時間等於虛擬終止作業的最早開始時間。

式(7)：虛擬終止作業的最晚完工時間等於整體專案完工時間。

式(8)：$J_j$ 的最晚完工時間等於其後置作業集合中，最小的實際開始時間。

式(9)：$J_j$ 的獨立性寬裕時間等於其最晚完工時間減去實際完工時間。

式(10)：單位時間內，執行作業 $J_j$ 的人力資源需求量總和必小於等於。(不大於) 單位時間內人力資源上限。

# 二、演算法

以下將分別說明禁忌搜尋法的搜尋流程與作業排工流程：

## (一)禁忌搜尋法的搜尋流程

禁忌搜尋法的求解流程如圖三 所示，步驟如下：

步驟一：初始化目前搜尋所在的層級 (stage)，決定禁忌名單的記憶長度

步驟二：使用 ACTIM 法則來產生初始解，使其爲現存解

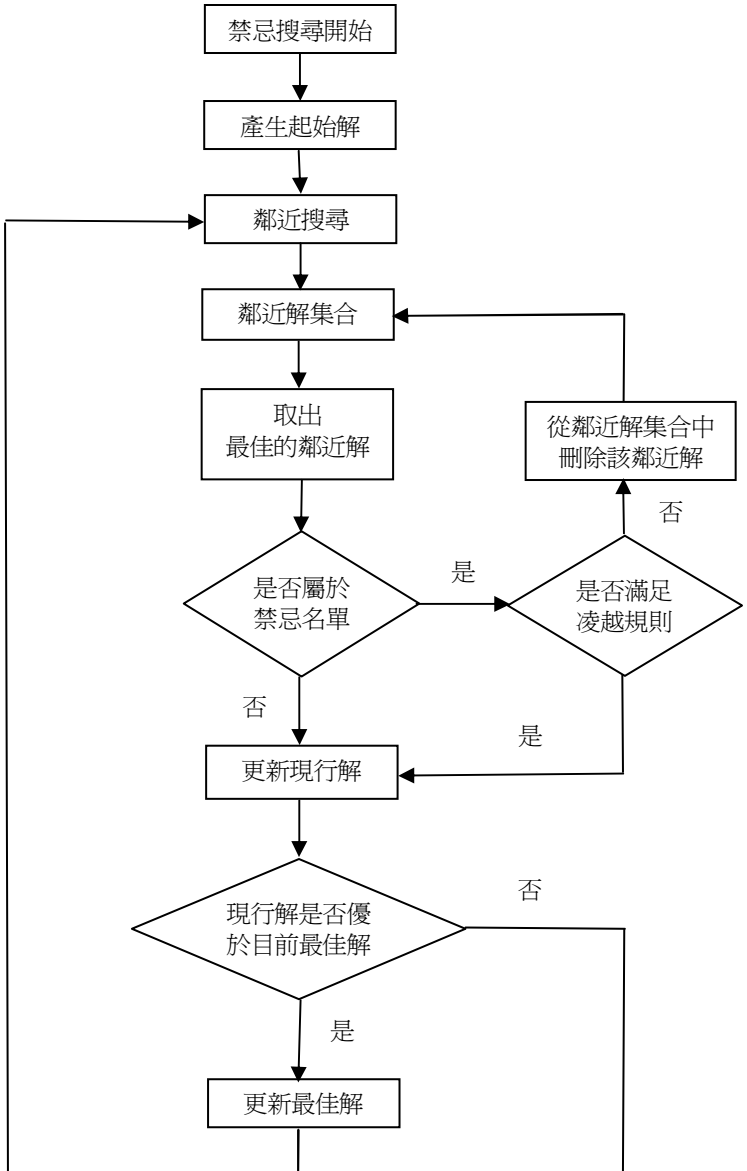
步驟三：評估現存解的所有鄰近解集合

步驟四：選擇最符合目標函數且不屬於禁忌單中的解，將其與目前最佳解進行比較

步驟五：更新禁忌名單並存入目前最佳解，進入下一個層級進行搜尋

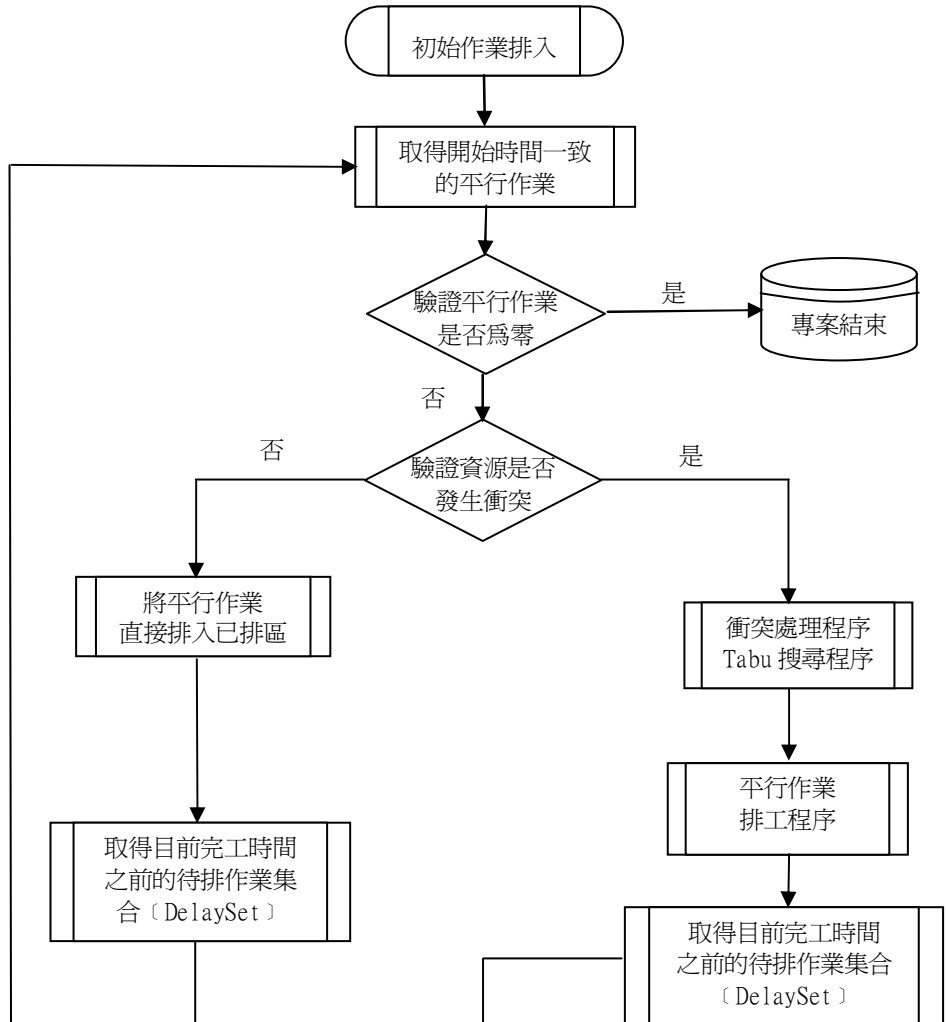
步驟六：如果新的解優於目前最佳解則更新

步驟七：如果還沒有到層級終止條件則重覆步驟三，否則，停止搜尋並回傳最佳解



圖三　禁忌搜尋流程

## (二)排工流程



圖四　排工流程

本研究將排工流程簡化爲二種狀態，第一種狀態爲資源發生衝突的平行作業排工 (parallel work)，第二種狀態爲延遲作業的排工 (delay work)。排工流程如圖四。在排工初期，我們會藉由要徑法的向前運算 (forward) 方法，取得專案各作業活動的最早開始時間，確立作業活動間的先後關係。接著，按照下列步驟，動態的更新各作業活動的實際開始時間完成專案排工：

步驟一：排入 $J_0$。

步驟二：取得已排入作業集合之直接後續作業集合，若此集合爲空集合，則到步驟六。

步驟三：驗證後續作業集合中之作業是否發生資源突，若有，則到步驟四。否則將所有開始時間一致的平行作業全部排入，到步驟五。

步驟四：以禁忌搜尋法配合 ACTIM 法則所選擇出來的起始解，決定平行作業的優先順序，依優先順序排入作業。

步驟五：以步驟二的執行結果，篩選出最早開始時間小於目前平行作業排工階段的完工時間之作業集合，作爲待排作業集合。同時，衡量

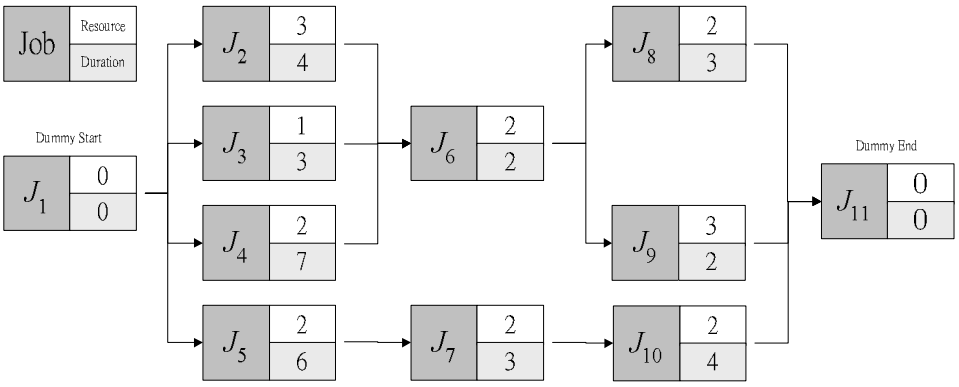
已排入的作業，在各 t 時間下的剩餘資源量，將待排作業集合中的作業按照其最早開始時間的大小，作爲實際排入優先順序，直到所有待排作業都已排入爲止，到步驟二。

步驟六：排入 $J_{n+1}$，完成排工。

## 三、平行作業的優先規則

專案排程問題在資源限制的假設下，使得原本最早開始時間一致的平行作業產生排工的優先順序，不同的加工順序將直接影後續作業的排工結果與整個專案的計劃品質。早期的研究學者都在努力的尋求可改善排程績效的優先準則。例如，Bedworth (1973) ACTIM 及 ACTRES 法、Davis & Patterson (1975) 的 MINSLK 法……等，各種排程優先規則。蔡登茂 (1996) 在有限資源多專案排程啓發法之績效評估及其應用的文獻表示，求解最小化總專案工期問題，在不同優先規則表現上有顯著不同，其比較 17 個常用的排程優先法則，分別在 7463 個問題中進行測試，並以三個評估準則進行優先法則的績效評估，發現 ACTIM 法則在求解最小化總專案工期的表現最好，於是本研究將以 ACTIM 法則配合禁忌搜尋法進行有限資源專案排程解的搜尋。

以下利用圖五的例題，在表一作業時間變動幅度的設定下，說明 ACTIM 法與本研究方法，分別在面臨作業延工前後的排工結果。




圖五　例題專案網路圖

表一　作業時間變動幅度

| 作業 | 作業時間延長幅度 (天) |
|---|---|
| $J_2$ | 2 |
| $J_3$ | 3 |
| $J_4$ | 2 |
| $J_5$ | 1 |

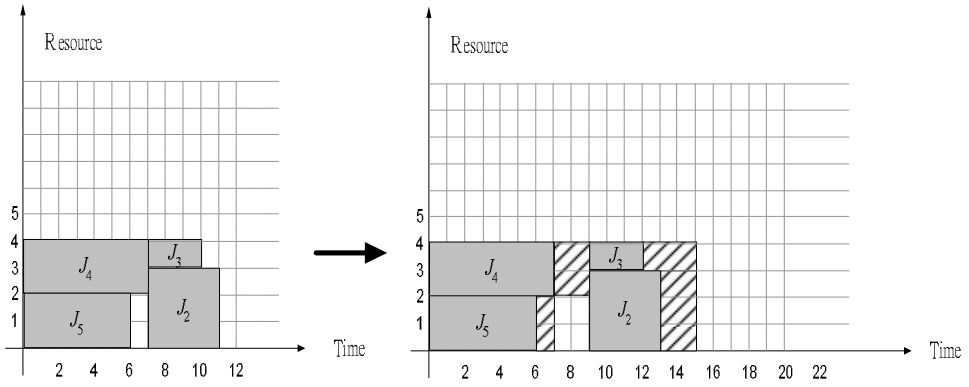
ACTIM 法則認爲，當目標作業到最終結點的距離愈長，表示此目標作業愈重要，應該要優先處理，將排程優先權指派給 ACTIM 權值大的作業。其中，ACTIM 權值爲目標作業至網路最終結點的最長路徑。根據 ACTIM 法的定義，藉由表二的衡量方式取得各作業活動的 ACTIM 權值，再依 ACTIM 權值的大小取得平行作業優先順序 $J_5 \rightarrow J_4 \rightarrow J_2 \rightarrow J_3$。

表二　各作業活動之 ACTIM 權值

| 作業 | 至最終結點的路徑組合 | ACTIM 權值 |
|---|---|---|
| $J_2$ | ( $J_2 \rightarrow J_6 \rightarrow J_8 \rightarrow J_{11}$ )=9*<br>( $J_2 \rightarrow J_6 \rightarrow J_9 \rightarrow J_{11}$ )=8 | 9/13 = 69.23% |
| $J_3$ | ( $J_3 \rightarrow J_6 \rightarrow J_8 \rightarrow J_{11}$ )=8*<br>( $J_3 \rightarrow J_6 \rightarrow J_9 \rightarrow J_{11}$ )=7 | 8/13 = 61.54% |
| $J_4$ | ( $J_4 \rightarrow J_6 \rightarrow J_8 \rightarrow J_{11}$ )=12*<br>( $J_4 \rightarrow J_6 \rightarrow J_9 \rightarrow J_{11}$ )=11 | 12/13 = 92.30% |
| $J_5$ | ( $J_5 \rightarrow J_7 \rightarrow J_{10} \rightarrow J_{11}$ )=13* | 13/13 = 100.00% |

註 1： * 表示目標作業至最終結點最長的路徑選擇
註 2： 要徑爲 ($J_1 \rightarrow J_5 \rightarrow J_7 \rightarrow J_{10} \rightarrow J_{11}$) ＝ ( 0+6+3+4+0 ) ＝ 13

圖六說明，ACTIM 法下延工前後的排程結果。

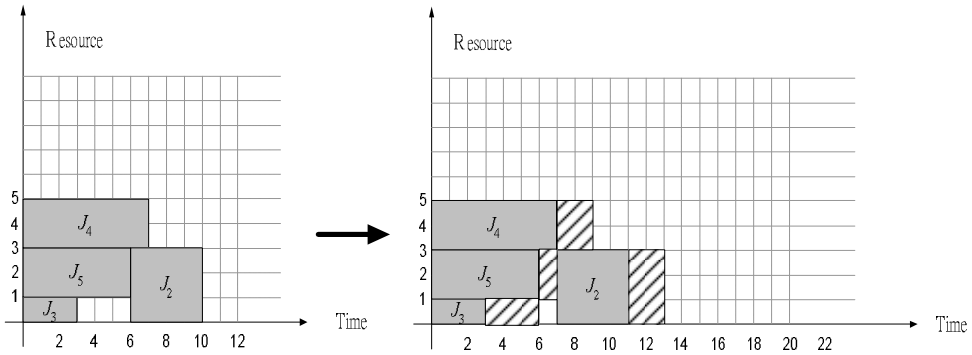



(a) 未延期的排工結果　　(b) 延期的排工結果

圖六　ACTIM 法排工延期結果 ($t \leq 15$)

由圖六可知，在表一的設計下，平行作業的完工時間由原來的 11 個單位工作天延長至 15 個工作天。

反觀本研究方法，藉由追求各作業活動之獨立性寬裕時間最大化與最小專案工期來決定平行作業優先順序。可以發現禁忌搜尋結果將使得平行作業的完工時間從原來的第 10 天延長至第 13 天。如圖七所示：




(a) 未延期的排工結果　　(b) 延期的排工結果

圖七　本研究禁忌搜尋排工延期結果 (t≤13)

# 四、排程方法

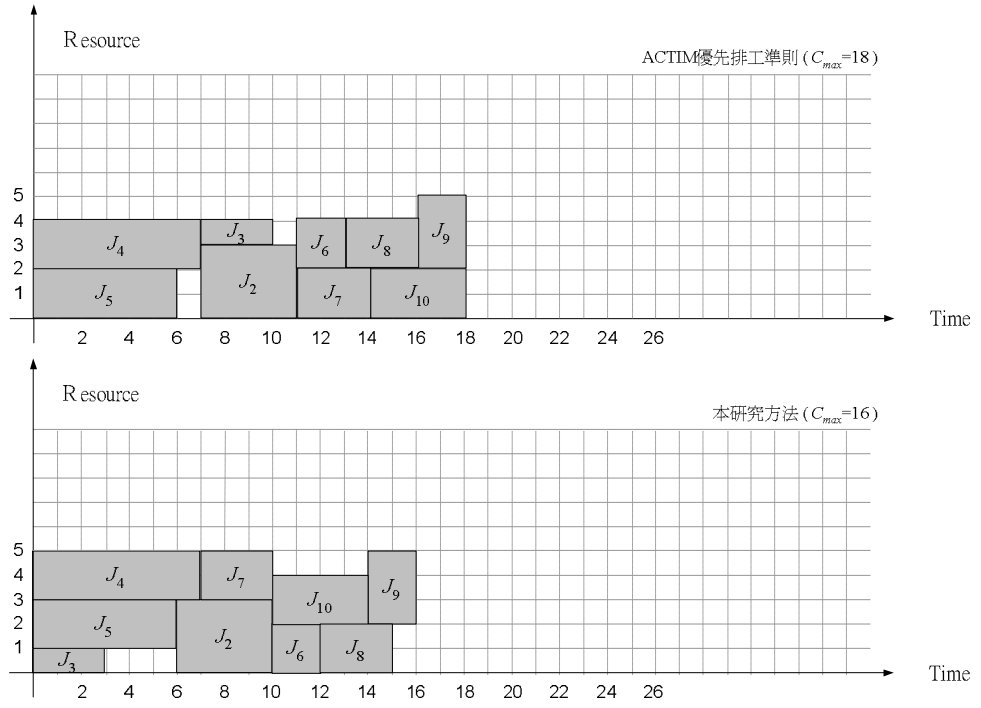
表三　不同優先排程方法的執行過程

| 排程階段 | 起始 | 平行排工 | 延遲排工 | 平行排工 | 延遲排工 | 延遲排工 | 平行排工 | 平行排工 |
|---|---|---|---|---|---|---|---|---|
| 排程時間 | - | $J_0$ | - | $J_{11}$ | $J_{13}$ | $J_{14}$ | $J_{15}$ | $J_{18}$ |
| 排入作業準則 ACTIM 權值 | start dummy | $J_5$, $J_4$, $J_2$, $J_3$ | - | $J_7$, $J_6$ | $J_8$ | $J_{10}$ | $J_9$ | End dummy |
| 完成作業 | Start dummy | - | - | $J_5$, $J_4$, $J_2$, $J_3$ | $J_5$, $J_4$, $J_2$, $J_3$, $J_7$, $J_6$ | $J_5$, $J_4$, $J_2$, $J_3$, $J_7$, $J_6$, $J_8$, $J_{10}$ | $J_5$, $J_4$, $J_2$, $J_3$, $J_7$, $J_6$, $J_8$, $J_{10}$ | $J_5$, $J_4$, $J_2$, $J_3$, $J_7$, $J_6$, $J_8$, $J_{10}$, $J_9$ |
| 待排作業 | $J_2$, $J_3$, $J_4$, $J_5$ | - | - | - | - | - | End dummy | $J_0$ |
| 延遲作業 | - | - | - | $J_8$, $J_9$ | $J_9$, $J_{10}$ | $J_9$ | - | - |
| 排程階段 | 起始 | 平行排工 | 延遲排工 | 平行排工 | 延遲排工 | 延遲排工 | 平行排工 | 平行排工 |
| 排程時間 | - | $J_0$ | $J_7$ | $J_{10}$ | $J_{12}$ | $J_{14}$ | $J_{16}$ | |
| 排入作業準則 禁忌搜尋 | Start dummy | $J_3$, $J_5$, $J_4$, $J_2$ | $J_7$ | $J_6$, $J_{10}$ | $J_8$ | $J_9$ | End dummy | |
| 完成作業 | Start dummy | - | $J_3$, $J_4$, $J_5$ | $J_2$, $J_3$, $J_4$, $J_5$, $J_7$ | $J_2$, $J_3$, $J_4$, $J_5$, $J_6$, $J_7$ | $J_2$, $J_3$, $J_4$, $J_5$, $J_6$, $J_7$, $J_8$, $J_{10}$ | $J_2$, $J_3$, $J_4$, $J_5$, $J_6$, $J_7$, $J_8$, $J_{10}$, $J_9$ | |
| 待排作業 | $J_2$, $J_3$, $J_4$, $J_5$ | - | $J_6$, $J_{10}$ | - | - | End dummy | $J_0$ | |
| 延遲作業 | - | $J_7$ | - | $J_8$, $J_9$ | $J_9$ | - | - | |

在專案排程中，後置作業的實際開始加工時間會因爲前置作業的實際排工情況而有所調整。前置作業的排工品質也會直接的影響後置作業的排工結

果。於是，當作業依時進展的過程中，若資源不足以供給所有開始時間一致的平行作業同時加工，爲了衡量平行作業排工階段之各作業活動獨立性寬裕時間，我們將排工階段區分成平行作業排工階段與延遲作業排工階段。其中平行作業排工階段指的是目前專案正在進行平行作業的排工，延遲作業排工階段指的是當平行作業排工完成之後，所有最早開始時間小於目前實際完工時間的作業排工。除此之外，每個排工階段再依作業的排入狀態，區分成完成區與待排區，藉以直接處理待排區的作業。以下沿用圖五的例子來說明排程方法的執行過程。

已知每日可恢復資源限制在 5 單位且虛擬起始作業的最早開始時間設定在 t=0 的狀態下，使用兩種不同的平行作業優先準則，進行作業排工。第一種爲 ACTIM 法下的 ACTIM 權值，第二種爲本研究方法，於是，我們可以得到表三與圖八：



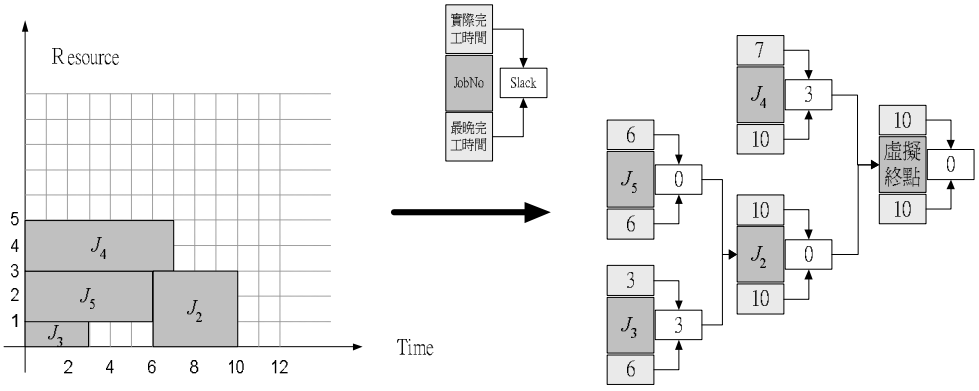

圖八　不同法則的排序

## 五、區域搜尋與寬裕時間的衡量

當開始時間一致的平行作業發生資源衝突時，便是禁忌搜尋程序的開

始。爲了盡可能在平行作業排工階段，最小化專案完工時間下，找出各作業獨立性的寬裕時間使其最大化。我們以 ACTIM 優先規則來產生起始解，作爲區域搜尋的開始。並藉由鄰近交換來產生鄰近解集合，最後，再針對每個鄰近解進行平行作業的排工與目標值的衡量。由於，本研究期望藉由排程目標值的設定，使專案工期最短之外，並嘗試找出具有預防能力的排程計畫，藉以降低專案活動工期發生變動所帶來的延工成本。

圖九將說明如何衡量平行作業排工階段之各作業獨立性寬裕時間：




圖九　平行作業排工階段之各作業獨立性寬裕時間衡量

從圖九可知，我們將平行作業排工的實際完工時間指向一個虛擬終點，並由此終點作爲往回推的起點，由後往前依序求得各作業的最晚完工時間。在此平行作業排工階段，各作業活動的實際完工時間與其最晚完工時間之差，即爲目標作業的獨立性寬裕時間。於是，我們可以準確的衡量出$J_3$與$J_4$各有 3 單位的獨立性寬裕時間。準此，除了各作業活動皆可以在其最早可排入時間的設定下盡可能的排入之外，並藉由各作業獨立性寬裕時間的搜尋來取得具有預防能力的排程解。

# 肆、資料測試與分析

本研究使用 Macromedia Flash 2004 撰寫程式，執行環境爲 Pentium(D) 2.67GHz、記憶體爲 1.5GB、作業系統：XP 之個人電腦。本節將分別說明問題結構並分析實驗結果。

在本研究中，將分二個步驟進行研究分析：(1)針對每個隨機抽取的測試例題，分別利用本研究方法與 ACTIM 法則求出在二種 (R=10;R=15) 的資源限制下，不同演算法建議的排工順序；(2)利用(1)的結果實驗在不同非預期作業參數變動幅度下，比較二種方法工期延長的幅度，進而觀察專案排工計劃的穩健度。

# 一、測試資料說明

本研究採用 PSPLIB 所提供的線上測試例題，作爲實驗測試依據。線上測試題庫依作業數可以區分爲 30、60、90 及 120 四種題型，而每種題型又分爲單一資源問題 (single resource problem, SRP) 與混合資源問題 (mix resource problem, MRP)，其中，單一資源的問題主要是強調，各作業活動僅需要一種可恢復資源，而專案排工的過程，便是將此一可恢復資源依時進展有效的分配到各作業，直到專案完工爲止。而混合資源的問題與單一資源的問題主要的差別在於，在混合資源的設定下，專案中的各作業活動所需要的資源不止一種。由於，本研究的目的在於凸顯穩健性排程所帶來的效益，準此，僅討論單一資源的問題，考慮不同資源限制變動對於獨立性寬裕時間及專案完工時間的影響，並將實驗結果帶入探討延工率之資料，將各類別中隨機抽取 30 組測試樣本進行作業活動持續時間隨機變動模擬，藉以觀察獨立性寬裕時間對延工率的影響。本研究針對 PSPLIB 所提供不同作業數的例題類別，在各個類別中皆隨機抽取 30 組測試樣本進行測試，並考慮資源限制變動與作業活動的持續時間變動等六種資料型態如表四所示：

表四　資料型態

| 作業時間 / 資源限制 | 不變 | 小幅度增額* | 大幅度增額** |
|---|---|---|---|
| 資源限制 (R=10)*** | I | II | III |
| 資源限制 (R=15) | IV | V | IV |

註 1：*以均等分配的方式，爲每個作業活動加工時間加上 0 到 2 之間的整數亂數。
註 2：**以均等分配的方式，爲每個作業活動加工時間加上 0 到 4 之間的整數亂數。
註 3：***資源限制 (R=10)，爲測試國際例題的資源下限。

禁忌搜尋法的參數說明與參數值列於下表：

表五　實驗參數表

| 參數名稱 | 參數值 | 參數說明 |
|---|---|---|

| | | |
|---|---|---|
| 禁忌名單記憶長度 n | n = $\lfloor\sqrt{n}\rfloor$ | n 的大小，爲平行作業數的根號取最小整數 |
| 停止搜尋的規則 Miter | 7 | 無法使目標值改善下運算 7 次 |

我們將每個類別的不同的資料型態分別隨機取三十個例題進行實驗，將得到的結果加以平均，觀察專案排程在各種資料測試型態的設定下，不同方法對於 *Robust* 與 $C_{max}$ 的影響。

## 二、資料測試結果分析

在平行作業排工階段，本研究方法配合 ACTIM 法則產生的起始解進行禁忌搜尋，藉由不同目標權數的設定，觀察不同作業數 (30、60、90、120) 在 *Robust* 與 $C_{max}$ 的變化。資料測試結果 (權重: $w_1 = 0.3,\ w_2 = 0.7$)，如表六所示：

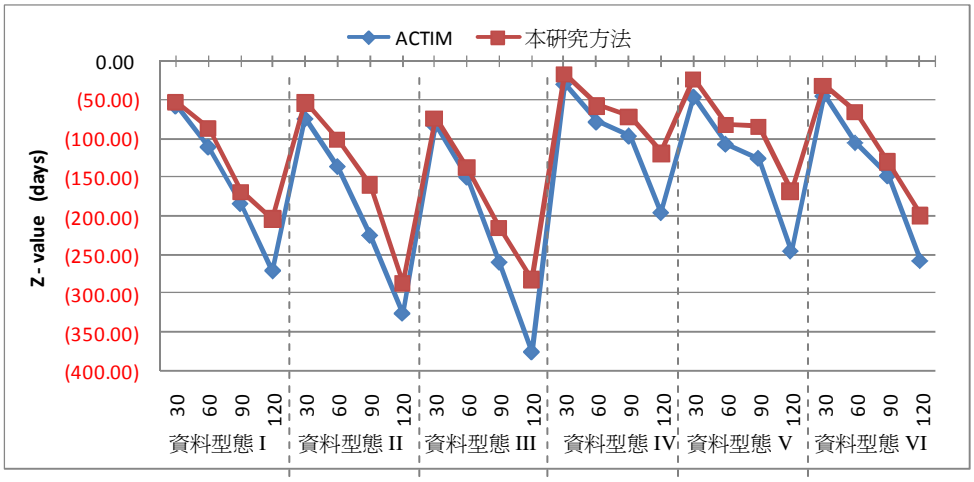
表六　資料測試結果-1

| 作業數 | 資料型態 | ACTIM | | | | 本研究方法 (權重: $w_1 = 0.3,\ w_2 = 0.7$) | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | CPU (sec.) | *Robust* | $C_{max}$ | Z | CPU (sec.) | *Robust* | $C_{max}$ | Z |
| 30 | I | 0.4173 | 19.00 | 90.33 | (57.53)* | 0.5793 | 25.69 | 85.79246 | (52.35)* |
| | II | 0.9811 | 13.00 | 111.33 | (74.03) | 1.3984 | 29.28 | 88.7391 | (53.33) |
| | III | 1.045 | 25.00 | 129.33 | (83.03) | 1.4762 | 33.56 | 118.3192 | (72.76) |
| | IV | 0.458 | 26.67 | 53.67 | (29.57) | 0.523 | 38.95 | 40.50729 | (16.67) |
| | V | 1.5993 | 15.67 | 72 | (45.70) | 1.7293 | 33.37 | 47.60835 | (23.31) |
| | VI | 1.3192 | 37.67 | 80 | (44.70) | 1.4492 | 48.72 | 64.75302 | (30.71) |
| 60 | I | 2.0017 | 46.51 | 177.67 | (110.42) | 2.746 | 68.58 | 152.3087 | (86.04) |
| | II | 3.9162 | 31.34 | 207.33 | (135.73) | 6.374 | 58.49 | 169.0981 | (100.82) |
| | III | 4.7772 | 61.67 | 240 | (149.50) | 7.5544 | 72.09 | 227.0384 | (137.30) |
| | IV | 1.8717 | 53.00 | 134.67 | (78.37) | 2.393 | 71.57 | 113.0757 | (57.68) |
| | V | 3.9864 | 30.00 | 165.67 | (106.97) | 5.029 | 49.96 | 137.5615 | (81.30) |
| | VI | 3.9276 | 76.00 | 182.67 | (105.07) | 4.9703 | 106.48 | 139.2871 | (65.56) |
| 90 | I | 5.4047 | 58.50 | 287.67 | (183.82) | 9.9727 | 83.86 | 275.952 | (168.01) |
| | II | 11.2886 | 45.00 | 340.67 | (224.97) | 20.3474 | 97.49 | 269.263 | (159.24) |
| | III | 11.2712 | 72.00 | 401.67 | (259.57) | 20.4896 | 110.71 | 355.6069 | (215.71) |
| | IV | 8.2407 | 84.50 | 173.67 | (96.22) | 9.558 | 110.93 | 150.0253 | (71.74) |
| | V | 16.9934 | 65.00 | 206.67 | (125.17) | 19.6281 | 96.33 | 161.851 | (84.40) |
| | VI | 17.172 | 104.00 | 255.67 | (147.77) | 19.8067 | 121.83 | 237.4216 | (129.65) |
| 120 | I | 13.166 | 70.67 | 417 | (270.70) | 35.471 | 129.36 | 346.2134 | (203.54) |
| | II | 26.8701 | 61.33 | 491.67 | (325.77) | 71.4801 | 92.39 | 449.2561 | (286.76) |
| | III | 26.7803 | 80.00 | 571 | (375.70) | 71.3903 | 154.88 | 468.6865 | (281.62) |
| | IV | 11.5573 | 105.00 | 324.33 | (195.53) | 24.8807 | 172.09 | 243.6934 | (118.96) |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | V | 23.4403 | 77.00 | 383.33 | (245.23) | 50.087 | 137.40 | 298.7329 | (167.89) |
| | VI | 23.8708 | 133.00 | 425.33 | (257.83) | 50.5174 | 180.31 | 361.6997 | (199.10) |

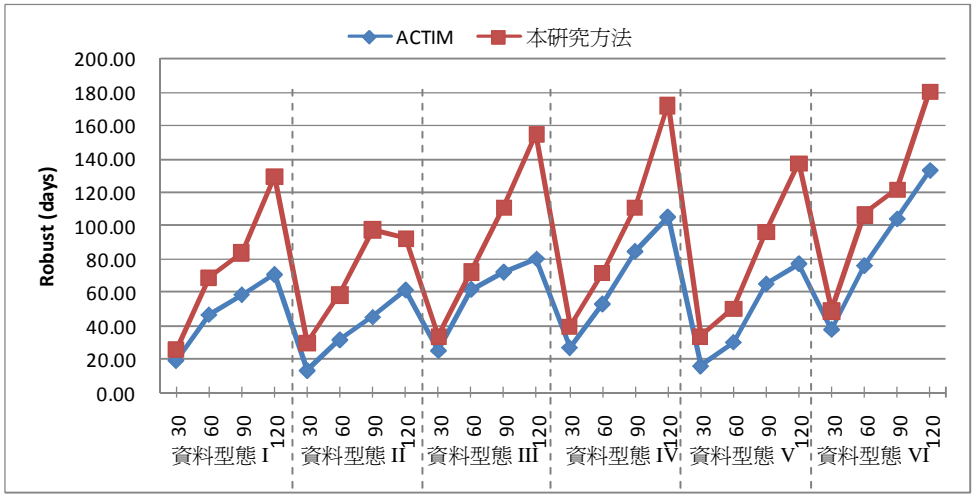
註 1：Z= $w_1 \cdot \sum_{j=1}^{n} Robust_j - w_2 \cdot C_{\max}$ 值愈大愈好，表中（ ）*代表負值。

由表六資料測試結果可知，本研究方法 (權重: $w_1 = 0.3$, $w_2 = 0.7$) 在考慮穩健性排程的前提下，所挑選出來的排程解在各種資料型態上之目標函數值的表現上均優於單純使用 ACTIM 法則所挑選出來的排程解，目標函數值之測試結果比較如圖十所示：




圖十　目標函數之測試結果比較 (權重: $w_1 = 0.3$, $w_2 = 0.7$)

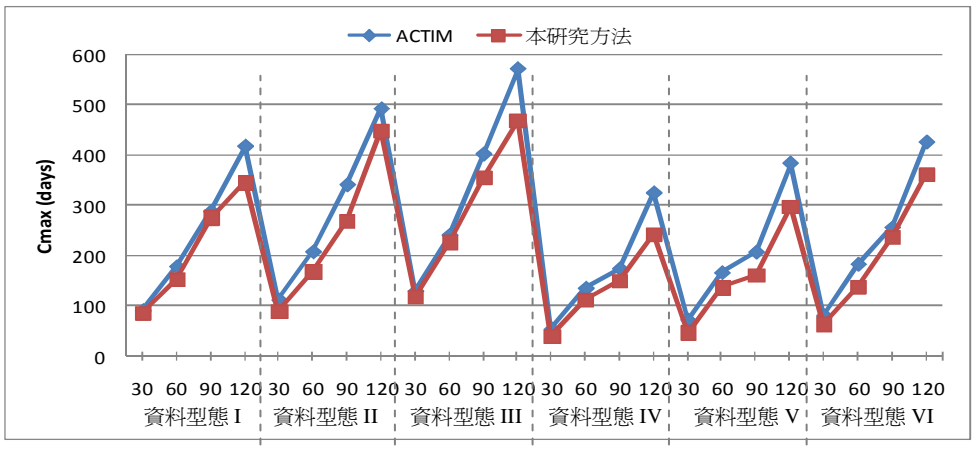
爲了更進一步的探討目標函數值背後隱含的意義，我們比較獨立性寬裕時間之測試結果，以圖十一表示，其中縱軸爲 *Robust*，橫軸爲不同資料型態下之作業數。




圖十一　獨立性寬裕時間之測試結果比較 (權重:$w_1$ = 0.3, $w_2$ = 0.7)

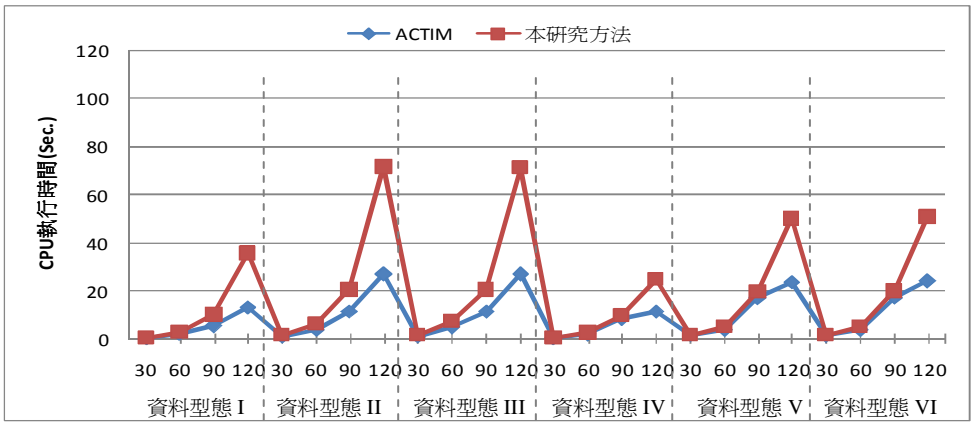
圖十一顯示，本研究方法在資源相對寬鬆 (資料型態 IV、V、IV) 的情況下，*Robust* 的表現會優於資源相對緊縮的情況；在作業時間干擾幅度較大 (資料型態 II→III 及 V→VI) 的情況下，*Robust* 的表現會優於作業時間干擾幅度較小的情況；較大作業數的測試例題，較容易搜尋出 Robust 較大的排程解。

我們再進一步的觀察圖十二專案完工時間之測試結果比較，其中縱軸爲 $C_{max}$，橫軸爲不同資料型態下之作業數。我們從資料測試結果觀察到，本研究方法在 $C_{max}$ 權重設定在 0.7 的前提下，$C_{max}$ 的表現會均優於 ACTIM 的情況；然而，在較大作業數的測試例題中，效果比較明顯，容易搜尋出 $C_{max}$ 較小的排程解。




圖十二　專案完工時間之測試結果比較 (權重:$w_1$ = 0.3, $w_2$ = 0.7)

雖然從圖十一與圖十二可以明顯的觀察到本研究方法資料測試結果（權重: $w_1 = 0.3,\ w_2 = 0.7$），均優於 ACTIM 法則，但需要以較長的 CPU 執行時間爲代價，不過也都控制在 80 秒之內搜尋完畢，如圖十三所示：




圖十三　CPU 執行時間之測試結果比較（權重: $w_1 = 0.3,\ w_2 = 0.7$）

爲了探討本研究方法的適用性，我們進一步的探討不同權重對於資料測試結果（權重: $w_1 = 0.7,\ w_2 = 0.3$）結果的顯響，如表七所示：

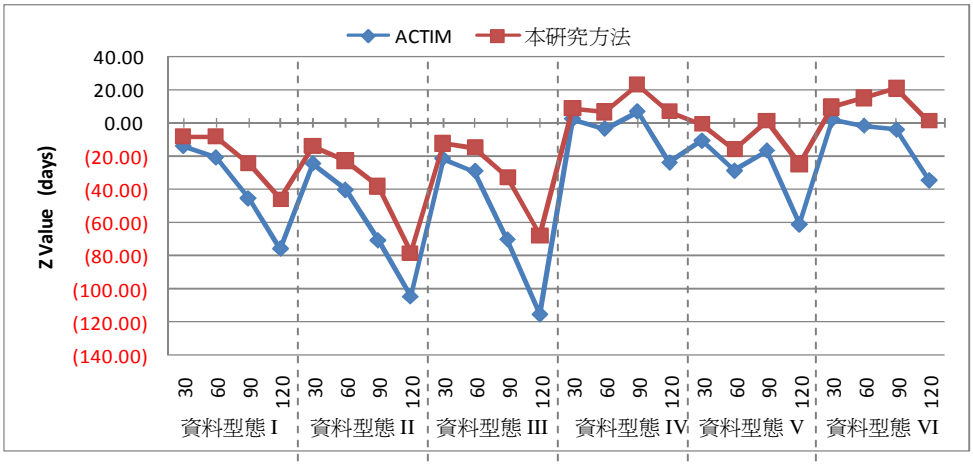
表七　資料測試結果-2

| 作業數 | 資料型態 | ACTIM | | | | 本研究方法（權重: $w_1 = 0.7,\ w_2 = 0.3$） | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | CPU (sec.) | *Robust* | $C_{max}$ | Z | CPU (sec.) | *Robust* | $C_{max}$ | Z |
| 30 | I | 0.4173 | 19.00 | 90.33 | (13.80) * | 0.7783 | 29.91 | 96.67754 | (8.07) * |
| | II | 0.9811 | 13.00 | 111.33 | (24.30) | 1.5765 | 33.32 | 124.3709 | (13.99) |
| | III | 1.045 | 25.00 | 129.33 | (21.30) | 1.6833 | 37.96 | 130.4308 | (12.56) |
| | IV | 0.458 | 26.67 | 53.67 | 2.57 | 0.5673 | 41.54 | 67.90271 | 8.70 |
| | V | 1.5993 | 15.67 | 72 | (10.63) | 1.7907 | 35.88 | 86.29165 | (0.77) |
| | VI | 1.3192 | 37.67 | 80 | 2.37 | 1.5148 | 51.39 | 88.30698 | 9.48 |
| 60 | I | 2.0017 | 46.51 | 177.67 | (20.75) | 3.8627 | 76.60 | 206.5813 | (8.36) |
| | II | 3.9162 | 31.34 | 207.33 | (40.26) | 8.3482 | 66.18 | 231.4219 | (23.10) |
| | III | 4.7772 | 61.67 | 240 | (28.83) | 8.4285 | 80.44 | 237.5016 | (14.94) |
| | IV | 1.8717 | 53.00 | 134.67 | (3.30) | 2.5147 | 77.74 | 158.9543 | 6.73 |
| | V | 3.9864 | 30.00 | 165.67 | (28.70) | 5.2565 | 55.96 | 183.1285 | (15.76) |
| | VI | 3.9276 | 76.00 | 182.67 | (1.60) | 5.0877 | 112.82 | 213.0629 | 15.05 |
| 90 | I | 5.4047 | 58.50 | 287.67 | (45.35) | 13.8748 | 97.04 | 307.798 | (24.41) |
| | II | 11.2886 | 45.00 | 340.67 | (70.70) | 25.8247 | 110.05 | 383.927 | (38.14) |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | III | 11.2712 | 72.00 | 401.67 | (70.10) | 26.0544 | 124.50 | 399.6931 | (32.76) |
| | IV | 8.2407 | 84.50 | 173.67 | 7.05 | 10.6862 | 119.03 | 200.7847 | 23.08 |
| | V | 16.9934 | 65.00 | 206.67 | (16.50) | 20.7379 | 103.93 | 238.679 | 1.14 |
| | VI | 17.172 | 104.00 | 255.67 | (3.90) | 22.6308 | 130.42 | 234.7984 | 20.86 |
| 120 | I | 13.166 | 70.67 | 417 | (75.63) | 52.8033 | 148.32 | 500.1266 | (46.21) |
| | II | 26.8701 | 61.33 | 491.67 | (104.57) | 95.9939 | 110.56 | 519.5739 | (78.48) |
| | III | 26.7803 | 80.00 | 571 | (115.30) | 103.0497 | 174.64 | 633.8935 | (67.92) |
| | IV | 11.5573 | 105.00 | 324.33 | (23.80) | 35.1658 | 186.66 | 411.4566 | 7.23 |
| | V | 23.4403 | 77.00 | 383.33 | (61.10) | 65.4049 | 151.55 | 436.2671 | (24.80) |
| | VI | 23.8708 | 133.00 | 425.33 | (34.50) | 66.7444 | 195.31 | 451.1303 | 1.38 |

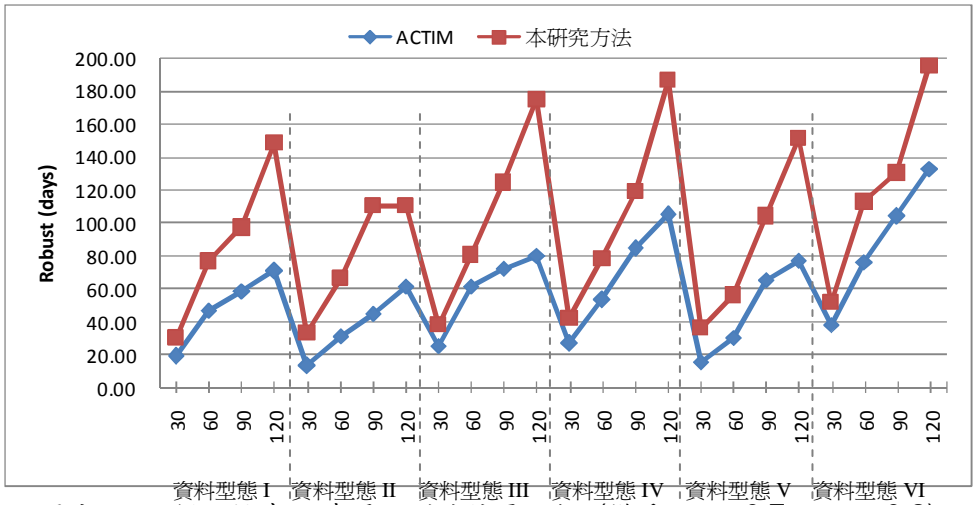
註 1：Z= $w_1 \cdot \sum_{j=1}^{n} Robust_j - w_2 \cdot C_{\max}$ 值愈大愈好，表中（ ）*代表負值。

由表七資料測試結果可知，本研究方法 (權重: $w_1 = 0.7$, $w_2 = 0.3$) 在考慮穩健性排程的前提下，所挑選出來的排程解在各種資料型態上之目標函數值的表現上均優於單純使用 ACTIM 法則所挑選出來的排程解，目標函數值之測試結果比較如圖十四所示：



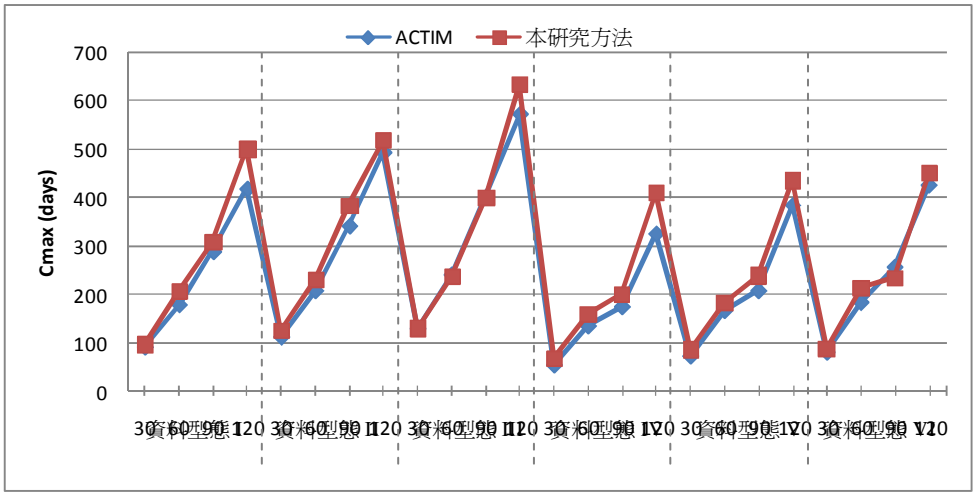

圖十四　目標函數之測試結果比較 (權重: $w_1 = 0.7$, $w_2 = 0.3$)

爲了更進一步的探討目標函數值背後隱含的意義，我們比較獨立性寬裕時間之測試結果，以圖十五表示，其中縱軸爲 *Robust*，橫軸爲不同資料型態下之作業數。




圖十五　獨立性寬裕時間之測試結果比較 (權重: $w_1 = 0.7$, $w_2 = 0.3$)

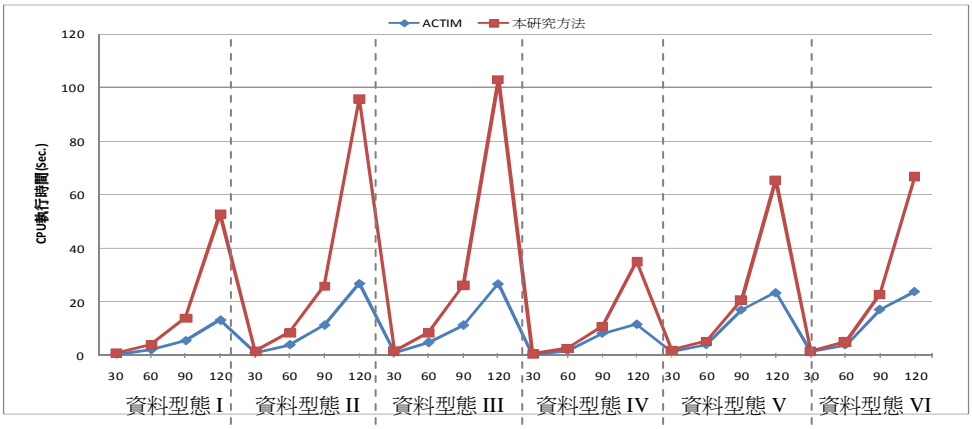
圖十五顯示，本研究方法獨立性寬裕時間之測試結果 (權重: $w_1 = 0.7$, $w_2 = 0.3$) 在各種資料型態下 *Robust* 的表現均優於 ACTIM 法則。若再進一步的觀察圖十六專案完工時間之測試結果，可以發現 *Robust* 與 $C_{max}$ 存在某種程度上的抵換關係，如圖十六所示：




圖十六　專案完工時間之測試結果比較 (權重: $w_1 = 0.7$, $w_2 = 0.3$)

我們從專案完工時間之測試結果觀察到，本研究方法在 *Robust* 權重($w_1$)設定在 0.7 時，Tabu 會搜尋 Robust 較大的排程解，甚至於比 *Robust* 權重($w_1$)設定在 0.3 的情況下還要大，以至於 $C_{max}$ 最後不減反增，大部份的排程結果均大於 ACTIM 法則，但增加的幅度仍屬於可容忍範圍。

最後從圖十七測試結果可以觀察到新的權重所需要耗費 CPU 執行時間。而結果仍需要更多的 CPU 執行時間，不過也都控制在 120 秒之內搜尋完畢。




圖十七　CPU 執行時間之測試結果比較 (權重: $w_1$ = 0.7, $w_2$ = 0.3)

## 三、探討延工率之資料測試

爲了能夠觀察獨立性寬裕時間對於專案延工率的影響，我們將本章第二節作業數 (30、60、90、120) 分別測試的例題之各作業活動以亂數的方式加上額外加工時間，作爲模擬實務上作業活動可能因爲外在因素干擾而產生延工的現象。

表八　探討延工率之測試資料型態

| 資源限制 \ 作業時間 | 小幅度增額 | 大幅度增額 |
|---|---|---|
| 資源限制 (R=10) | I | II |
| 資源限制 (R=15) | III | IV |

## 四、探討延工率之資料測試結果分析

我們藉由本章第二節的實驗結果代入模擬資料測試，觀察較大的 *Robust* 是否可以有效的降低專案的延工率，探討延工率之資料測試結果如表九所示：

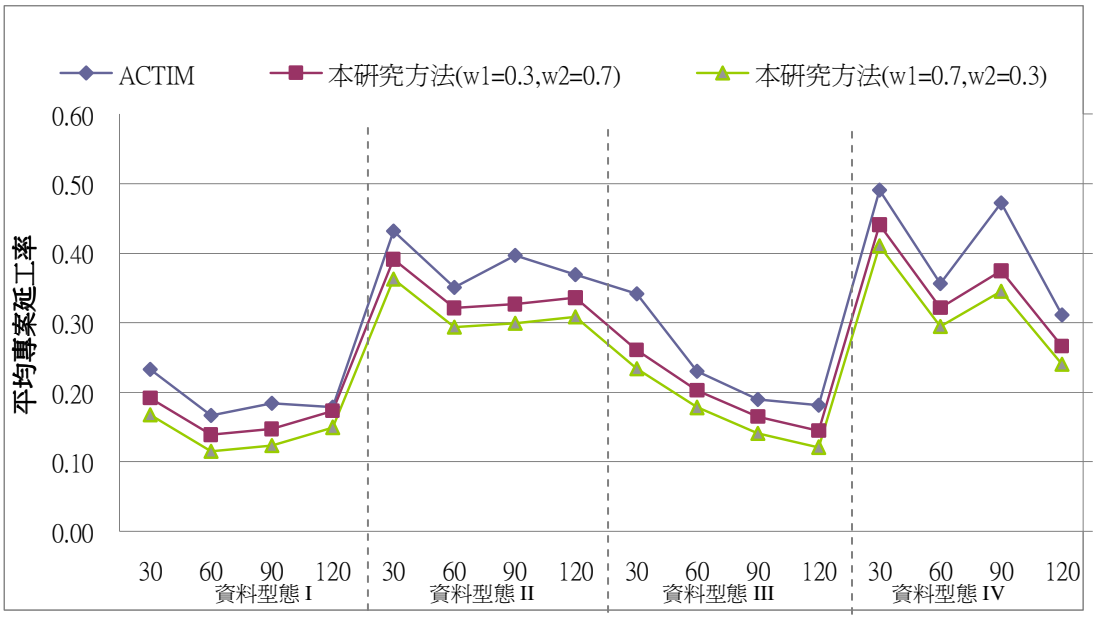
表九　探討延工率之資料測試結果

| 作業數 | 資料型態 | ACTIM | 本研究方法 | |
|---|---|---|---|---|
| | | | 權重: $w_1$ = 0.3, $w_2$ = 0.7 | 權重: $w_1$ = 0.7, $w_2$ = 0.3 |

| | | 工期 (天) | 平均延工率* | 工期 (天) | 平均延工率* | 工期 (天) | 平均延工率* |
|---|---|---|---|---|---|---|---|
| 30 | I | 111.33 | 23.25% | 107.67 | 19.20% | 105.44 | 16.73% |
| | II | 129.33 | 43.18% | 125.67 | 39.12% | 123.08 | 36.26% |
| | III | 72 | 34.15% | 67.67 | 26.09% | 66.23 | 23.40% |
| | IV | 80 | 49.06% | 77.33 | 44.08% | 75.73 | 41.10% |
| 60 | I | 207.33 | 16.69% | 202.33 | 13.88% | 198.19 | 11.55% |
| | II | 240 | 35.08% | 234.67 | 32.08% | 229.87 | 29.38% |
| | III | 165.67 | 23.02% | 162 | 20.29% | 158.69 | 17.84% |
| | IV | 182.67 | 35.64% | 178 | 32.17% | 174.35 | 29.46% |
| 90 | I | 340.67 | 18.42% | 330 | 14.71% | 323.19 | 12.35% |
| | II | 401.67 | 39.63% | 381.67 | 32.68% | 373.63 | 29.88% |
| | III | 206.67 | 19.00% | 202.33 | 16.50% | 198.2 | 14.12% |
| | IV | 255.67 | 47.22% | 238.67 | 37.43% | 233.55 | 34.48% |
| 120 | I | 491.67 | 17.91% | 489.33 | 17.35% | 479.5 | 14.99% |
| | II | 571 | 36.93% | 557 | 33.57% | 545.58 | 30.83% |
| | III | 383.33 | 18.19% | 371.33 | 14.49% | 363.67 | 12.13% |
| | IV | 425.33 | 31.14% | 410.67 | 26.62% | 402.16 | 24.00% |

註*：平均延工率係以 ACTIM 法則下之延工時間爲基準換算

我們將表四探討延工率之資料測試結果以圖十八呈現，縱軸爲平均延工率，橫軸爲不同資料型態下之作業數。




圖十八　探討延工率之資料測試比較圖

圖十八顯示本研究方法在以探討延工率之資料帶入之後，不論在資源限制相對緊縮（資料型態 I、II）或相對寬鬆（資料型態 IV、V）情況下，產生的延工率皆較單純使用 ACTIM 法則低。然而，隨著外在因素的干擾幅度擴大（從資料型態 I 到資料型態 II 及從資料型態 III 到資料型態 IV）時，本研究方法的延工率明顯低於單純使用 ACTIM 法則下的延工率。除此之外，當 *Robust* 權重（$w_1$）從 0.3 調整至 0.7 時，在沒有外在因素干擾的情境下需要以較高的 $C_{max}$ 爲代價，而在面臨作業加工時間無預警增加時，有考慮穩健性的排程解卻可表現出相對較低的延工率，正是本研究追求的目標。

# 伍、結論與建議

## 一、結論

由於過去的相關文獻主要在探討排程效率的問題，忽略專案執行過程中可能因爲外在因素干擾而中斷排程計劃。或是，藉由額外資源的投入來使專案盡可能的按照原計劃進行。如此額外資源投入的程度便存在改善的空間。於是，本研究針對作業活動的持續時間可能在專案計畫實際運作的過程中發生變動，而導致延工的問題，提出了一套求解架構。搭配禁忌搜尋法來達到最小化

整體專案之完工時間，又兼具排程目標，穩健性排程的特質，協助專案管理者有效的降低非預期因素的發生對專案計畫帶來的衝擊，減少額外資源投入的成本。

資料測試之進行過程，本研究分成二個階段處理。第一階段，在依時進展的排工過程，若資源發生衝突時，我們藉由禁忌搜尋法來決定最適延後排工的作業，直到所有的作業都被適當的排入為止，進而獲得一組建議的排程解。第二階段，我們再依照第一階段所建議的排程解代入探討延工率之模擬資料，觀察第一階段不同方法所建議的排工計劃在專案延工率上差異。

測試結果顯示，本研究所建構的演算法，不論在作業數大或小的情況，在有考慮專案排程穩健性的前提下，所求得作業活動獨立性寬裕時間均多於單純使用 ACTIM 法則。在代入模擬資料之後產生的平均延工率也均低於單純使用 ACTIM 法則。此外我們針對穩健性與總完工時間賦予不同權重組合，進行資料測試結果比較亦顯示，本研究發展之演算法可以使專案完工時間在有或無外因素干擾的情況下，均優於 ACTIM 法則的求解結果。據次，可知當面對作業加工時間無預警增加時，考慮穩健性的排程解，以本研究建構之演算法可以獲取相對較低延工率之排程解。

## 二、建議

本研究對於後續研究提出可以繼續深入做探討的建議與方向：

1.由於本研究只考慮到可恢復資源對專案排程的影響，為了可以更符合專案管理者的需求，未來可以加入不可恢復資源進行考量。

2.由於本研究為了要觀察不同方法在面對非預期因素隨機發生，對於專案延工率的影響，於是，將問題拆成二個階段處理。然而，對於後續的研究，可以嘗試建立一套新的演算法搜尋法則，降低資料處理的複雜度。

3.當資源發生衝突時，本次研究採用最小化整體專案完工時間與最大獨立性寬裕時間來決定延後排工的作業，後續研究仍可加入其它因素來強化求解品質及效率。

4.本研究針對 PSPLIB 四種作業數 30、60、90、120 進行測試。後續的學者或許可以針對其它題庫或實際資料進行測試，強化研究模型本身的適用性。

# 參考文獻

蔡登茂，「有限資源多專案排程啓發法之績效評估及應用」，*技術學刊*，第 11 卷第四期， 1996 年，頁 547-562。

Al-Fawzan, M. A. & Haouari M., "A Bi-objective Model for Robust Resource-constrained Project Scheduling", *International Journal of Production Economics,* Vol. 96, 2005, pp. 175-187.

Baar, T. P. Brucker & Knust, S., "Tabu-search Algorithms and Lower Bounds for the Resource-constrained Project Scheduling Problem, Meta-heuristics: Advances and Trends in Local Search Paradigms for Optimization", *S. Voss, S. Martello, I. Osman and C. Roucairol (Eds.), Kluwer, Boston,* 1998, pp. 1-8.

Bedworth, D. D., "Industrial System: Planning: Analysis and Control", *The Ronald Press Co., New York,* 1973.

Boctor, F. F., "Some Efficient Multi-heuristic Processdures for Resource-constrained Project Scheduling", *European Journal of Operational Research*, Vol. 49, 1990, pp. 3-13.

Bowers, J. A., "Interpreting Float in Resource Constrained Projects", *International Journal of Project Management,* Vol. 18, 2000, pp. 385-392.

Brand, J. D., Meyer, W. L. & Shaffer, L. R., "The Resource Scheduling Problem in Construction", *Civil Engineering Studies Report No.5,* Department of Civil Engineering, University of Illinois, Urbana, IL. 1964.

Davis, E. W. & Patterson, J. H., "A Comparison of Heuristic and Optimum Solutions in Resource-constrained Project Scheduling", *Management Science,* Vol. 21, 1975, pp. 944-955.

Dell'Amico, M., Trubian, M., "Applying Tabu Search to the Job-shop Scheduling Problem", *Annals of Operations Research*, Vol. 41, 1993, pp. 231-252.

Dorn, J., Girsch, M., Skele G. & Slany, W., "Comparison of Iterative Improvement Techniques for Schedule Optimization", *European Journal of Operational Research*, Vol. 94, 1998, pp. 349-361.

Glover, F., "Future Path for Integer Programming and Links to Artificial Intelligence", *Computers and Operations Research*, Vol. 13, 1986, pp. 533-549.

Glover, F. & Laguna, M., "Tabu Search", *Kluwer Academic Publishers*, 1997, Boston/ Dordrecht/ London.

Kelley, J. E., "The Critical Path Method: Resources Planning and Scheduling", *Ch. 21 in Industrial scheduling,* Prentice-Hall, Englewood Cliffs, New Jersey, 1963.

Kincaid, R. K., "Minimizing Distortion in Truss Structures: A Comparison of Simulated Annealing and Tabu Search", *Structural Optimization*, Vol. 5, 1993, pp. 217–224.

Kurtulus, I. S. & Davis, E. W., "Multi-project Scheduling: Categorization of Heuristics Rules Performance", *Management Science,* Vol. 28(2), 1982, pp. 161-172.

Kurtulus, I. S. & Narula, S. C., "Multi-project Scheduling: Analysis of Project Performance", *IIE Transactions*, Vol. 17(1), 1986, pp. 58-66.

Lee, I., "Artificial Intelligence Search Methods for Multi-machine Two-stage Scheduling with Due Date Penalty, Inventory, and Machining Costs", *Computers and Operations Research*, Vol. 28, 2001, pp. 835-852.

Lee, J. K. & Kim, Y. D., "Search Heuristics for Resource Constrained Project Scheduling", *Journal of the Operational Research Society,* Vol. 47, 1996, pp. 678-689.

Malek, M., Guruswamy, M. & Pandya, M., "Serial and Parallel Simulated Annealing and Tabu Search Algorithms for the Traveling Salesman Problem", *Annals of Operations Research*, Vol. 21, 1989, pp. 59-84.

Manoharan, S. & Shanmuganathan, S., "A Comparison of Search Mechanisms for Structural Optimization", *Computers and Structures*, Vol. 73, 1999, pp. 363–372.

Marett, R. & Wright, M., "A Comparison of Neighborhood Search Techniques for Multi-objective Combinatorial Problems", *Computers and Operations Research*, Vol. 23(5), 1996, pp. 465-483.

Murata, T. & Ishibuchi, H., "Performance Evaluation of Genetic Algorithms for Flowshop Scheduling Problems", *IEEE Transactions*, 1994.

Nonobe, K. & Ibaraki, T., "Formulation and Tabu Search Algorithm for the Resource Constrained Project Scheduling Problem", *In Essays and surveys in metaheuristics, Ribeiro, CC. and Hansen, P. (Ed.), Kluwer Academic Publishers,* 2002, pp. 557-588.

Patterson, J. H. & Huber, W. D., "A Horizon-varying Zero-one Approach to Project Scheduling", *Management Science,* Vol. 20(6), 1974, pp. 990-998.

Patterson, J. H., Slowinski, R., Talbot, F. B. & Weglarz, J., "Computational Experience with A Back-tracking Algorithm for Solving a General Class of Precedence and Resource Constrained Scheduling Problems", *European Journal of Operational Research,* Vol. 49, 1990, pp. 68-79.

Pirlot, M., "General Local Search Methods", *European Journal of Operational Research*, Vol. 92, 1996, pp. 493-511.

Sinclair, M., "Comparison of the Performance of Modern Heuristics for Combinatorial Optimization on Real Data", *Computers and Operations Research*, Vol. 20(7), 1993, pp. 687-695.

Taillard, E. D., "Some Efficient Heuristic Methods for the Flow Shop Sequencing Problem", *European Journal of Operational Research*, Vol. 47, 1990, pp.65-74.

Talbot, F. B., "Resource-constrained Project Scheduling with Time-resource Tradeoffs: the Non-preemptive Case", *Management Science,* Vol. 28, 1982, pp. 1197-1210.

Talbot, F. B. & Patterson, J. H., "An Efficient Integer Programming Algorithm for Solving Resource Constrained Scheduling Problem", *Management Science,* Vol. 24(11), 1978, pp. 1163-1174.

Wang, J., "Constraint-based Schedule Repair for Product Development Projects with Time-limited Constraints", *International Journal of Production Economics,* Vol. 95, 2005, pp. 399-414.

Widmer, M. & Hertz, A., "A New Heuristic Method for the Flow Shop Sequencing Problem", *European Journal of Operational Research*, Vol. 41, 1989, pp.186-193.

Wiest, J. D., "A Heuristic Model for Scheduling Large Projects with Limited Resources", *Management Science,* Vol. 13, 1967, pp. 359-377.

# On Project Scheduling with respect to Resources Constrained and Robust

RONG-HWA HUANG, CHANG-LIN YANG, SHIH-HAO LIU *

## ABSTRACT

A common problem in project management is that planned schedules are often disrupted by some uncontrollable factors like additional job duration. As a result, project managers are often unable to meet their promised completion dates. To aim that, we introduced the concept of schedule robustness and find the more robustness plan by finding job independent slack to less the extra cost of undesirable conditions happen like rework cost.

In this paper, we designed a bi-objective function which is not only considered the make span of the project, but also robustness to less extra cost of delay work.

We used the meta-algorithms which combined tabu-search algorithm and ACTIM to test the data of PSPLIB (project scheduling problem library) with two-factor experiment design and compared with only use ACTIM. The result of computation based on 720 schedules show that our meta-algorithms is more effective than currently published based on ACTIM rule.

Keywords: resource-constrained project scheduling problem, robustness scheduling, tabu-search algorithm

* Rong-Hwa HUANG, Associate Professor, Graduate Institute of Management, Fu Jen Catholic University. Chang-Lin YANG, Associate Professor, Department of Business Administration, Fu Jen Catholic University. Shih-Hao LIU, MBA. Student, Graduate Institute of Management, Fu Jen Catholic University.