

國立成功大學
資訊管理研究所
碩士論文

具時間限制之穩定性專案基線排程研究

On constructing stable project baseline schedules
with time constraints

研究生：楊橙坤

指導教授：王逸琳 博士

中華民國九十七年七月

國立成功大學

碩士論文

具時間限制之穩定性專案基線排程研究
On Constructing Stable Project Baseline Schedules
with Time Constraints

研究生：楊橙坤

本論文業經審查及口試合格特此證明

論文考試委員：洪一重
蔡志
王逸琳

指導教授：王逸琳

系(所)主管：

中華民國 97 年 5 月 30 日

摘要

在專案排程中，專案經理人如何掌握影響專案運行的因素來安排行程是一重要之管理議題。以現今的排程而言，可能會因為外包而必須將時間限制和不確定因素也列入考慮。時間限制係指和外包商簽約決定交貨的時間可能具有時窗範圍或時間排程等等限制；而外包商又可能會因毀約或延遲交貨等不確定因素而影響到專案活動的排程規劃，導致專案活動成本增加甚至停擺。本論文旨在探討如何在時間限制之下，對專案排程的活動開始時間加入額外的時間彈性，建立一具時間限制之穩定性專案基線排程以將排程的不確定性所引起之損失最小化。首先，我們改良文獻中的隨機專案排程網路產生器，將額外的時窗與時間排程限制同時列入考慮以設計一個新的具時間限制之隨機專案排程網路產生器。接著，本研究提出兩種多項式時間的啟發式演算法：貪婪演算法一 (Greedy I) 及貪婪演算法二 (Greedy II) 來求解。其中貪婪演算法一乃依本模式之網路圖和限制式特性為基礎發展而得；而貪婪演算法二更進一步加入成本的考量來設計排程時間該如何去調整，以得到更佳之排程結果。雖然此兩種啟發式演算法有不錯的求解效率，其求解效果卻不甚理想。最後，我們亦設計了另一個基因演算法來規劃排程，並與 CPLEX 最佳化軟體比對求解效率與效果，我們發現基因演算法在適當的終止條件設定下，可以得到不錯的解，且其求解效率在大規模的排程問題上亦遠勝 CPLEX。

關鍵字：專案排程；穩定性專案基線排程；不確定；時間限制；基因演算法

Abstract

In this thesis, we first propose a random project scheduling network generator by integrating additional time window or time schedule constraints with a popular project network generator (RanGen). We then give two polynomial heuristic algorithms (Greedy I and Greedy II) to solve the integer programming problems, where Greedy I exploits the special structure of the models so that it can suggest a feasible start time for each task in a topological order to reduce the objective values, while Greedy II further takes the objective weights associated with variables into consideration and improves the quality of the solutions obtained by Greedy I. To further get solutions of better qualities, we propose a genetic algorithm and conduct computational experiments to evaluate the effectiveness (optimality gap) and efficiency (running time) of our proposed algorithms (Greedy I, Greedy II, and GA) and a popular optimization solver, CPLEX. The results show that our greedy heuristics are very efficient but not as effective, GA can be both efficient and effective, while CPLEX usually consumes much more computational time and is especially inefficient for problems of larger scale.

keywords: project scheduling, stable baseline scheduling, uncertainty, genetic algorithm

誌謝

在研究所生涯中，承蒙指導教授王逸琳老師的細心教導，讓我在學術研究和求學精神方面能有更多的體會；且在我受挫低潮時，老師更不厭其煩的分享其經驗，並鼓勵我繼續加油，這對我的幫助真的很大。而在論文方面，更由於老師費心費力的指導與建議之下，才能使本論文得以順利完成，在此由衷的致上最高的感謝。

在論文口試期間，洪一薰教授建議學生的論文修改方向和蔡青志老師點出的論文撰寫方面的一些錯誤，皆促使學生論文能往更完善的方向去改進，在此致上最誠摯的感謝。

在我研究所求學的日子中，感謝修杰學長、正翰學長和惠娥學姊在學業上的指導及生活上的照顧；從你們身上我還學到很多學業以外的事情，感謝你們對我的教誨與照顧；還有要感謝跟我一起共度研究生生涯的同學群達和姿君，謝謝你們在課業上的幫忙與鼓勵，讓我們大家能完成研究所的學業；另外，也要感謝實驗室的其他成員家宜、正楠、姿儀和建傑，由於你們的陪伴，讓我的研究所生活多了許多的樂趣與好笑的回憶。

另外，還要非常感謝我的父親、母親以及所有關心我的家人與朋友，由於你們各方面的支持與鼓勵下，使得我能專心致力於課業上無後顧之憂，由於你們的關心，讓我在遇到挫折時都能堅強的度過，真的非常謝謝你們。

最後，將此論文獻給我最愛的家人與朋友，並且感謝曾經關心及幫助過我的人，你們的支持與鼓勵是我能繼續往前邁進的最大動力，謝謝你們。

目 錄

| | |
|-----------------------|------|
| 摘要 | i |
| Abstract | ii |
| 誌謝 | iii |
| 表目錄 | vii |
| 圖目錄 | viii |
| 符號說明 | ix |
| 第一章 緒論 | 1 |
| 1.1 研究動機 | 2 |
| 1.2 研究目的 | 2 |
| 1.3 研究流程圖 | 3 |
| 1.4 論文架構 | 3 |
| 第二章 文獻探討 | 5 |
| 2.1 專案排程問題組成之要素 | 5 |
| 2.1.1 專案排程活動 | 5 |
| 2.1.2 活動之先序關係 | 5 |
| 2.1.3 資源限制類型 | 6 |
| 2.1.4 專案排程目標 | 7 |
| 2.2 AND/OR 先序限制相關文獻探討 | 8 |
| 2.2.1 AND/OR 先序限制模式 | 8 |
| 2.2.2 AND/OR 節點之應用領域 | 9 |

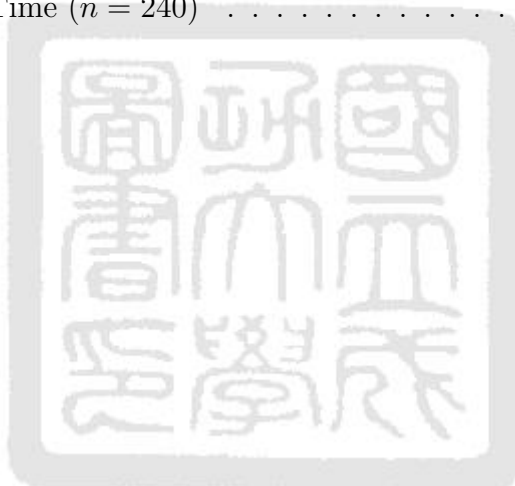
| | | |
|---------------------------|--|-----------|
| 2.3 | 具時間限制之活動網路 | 9 |
| 2.3.1 | 時間限制定義 | 9 |
| 2.3.2 | 具時間限制之排程求解方法 | 10 |
| 2.4 | 不確定性排程相關文獻探討 | 11 |
| 2.4.1 | 被動排程 (Reactive scheduling) | 12 |
| 2.4.2 | 隨機專案排程 (Stochastic project scheduling) | 12 |
| 2.4.3 | 模糊專案排程 (Fuzzy project scheduling) | 13 |
| 2.4.4 | 敏感度分析 (Sensitivity analysis) | 14 |
| 2.5 | 穩定性專案基線排程相關文獻探討 | 14 |
| 2.5.1 | LPH 排程模式 | 15 |
| 2.5.2 | ADFF 模式 | 15 |
| 2.5.3 | EWD1 排程模式 | 15 |
| 2.6 | 小結 | 16 |
| 第三章 隨機專案排程網路產生器之設計 | | 18 |
| 3.1 | 專案排程網路產生器相關文獻回顧 | 18 |
| 3.2 | RanGen 網路產生器 | 20 |
| 3.3 | 具時間限制之專案排程網路產生器 | 22 |
| 3.3.1 | 時間限制之設計概念 | 22 |
| 3.3.2 | 網路產生器應用簡例 | 24 |
| 3.4 | 小結 | 26 |
| 第四章 模式建立與演算法 | | 27 |
| 4.1 | 具時間限制之穩定性專案基線排程模式 | 27 |
| 4.1.1 | 模式建構目的與研究限制 | 27 |
| 4.1.2 | 模式特性描述與建構 | 28 |
| 4.2 | 演算法建立與說明 | 30 |
| 4.2.1 | 貪婪演算法一 (Greedy I) | 30 |

| | |
|------------------------------------|-----------|
| 4.2.2 貪婪演算法二 (Greedy II) | 35 |
| 4.2.3 基因演算法 (GA) | 37 |
| 4.3 小結 | 38 |
| 第五章 數值分析與討論 | 40 |
| 5.1 實驗環境與參數設定 | 40 |
| 5.2 數值分析與討論 | 41 |
| 5.3 小結 | 43 |
| 第六章 結論與未來研究方向 | 46 |
| 6.1 結論 | 46 |
| 6.2 未來研究方向 | 47 |
| 參 考 文 獻 | 50 |



表 目 錄

| | | |
|-----|--|----|
| 5.1 | Optimality Gap ($n = 60$) | 42 |
| 5.2 | Optimality Gap ($n = 120$) | 42 |
| 5.3 | Optimality Gap ($n = 180$) | 43 |
| 5.4 | Optimality Gap ($n = 240$) | 43 |
| 5.5 | Computational Time ($n = 60$) | 44 |
| 5.6 | Computational Time ($n = 120$) | 44 |
| 5.7 | Computational Time ($n = 180$) | 44 |
| 5.8 | Computational Time ($n = 240$) | 45 |



圖目錄

| | | |
|-----|---|----|
| 1.1 | 具時間限制之專案網路範例 | 3 |
| 1.2 | 研究流程圖 | 4 |
| 3.1 | 隨機專案排程網路圖例及其順序矩陣(來源:Demeulemeester et. al., 2003) | 20 |
| 3.2 | 具相同關係矩陣之兩相異專案排程網路圖(來源:Demeulemeester et al., 2003) | 21 |
| 3.3 | 由 remove_arcs(0.5) 產生圖 3.1 之網路圖(來源:Demeulemeester et al., 2003) | 22 |
| 3.4 | 具時間限制之隨機網路產生器運行政序 | 23 |
| 3.5 | RanGen 產生之檔案所繪成之網路圖(a)為輸出之檔案格式(b)為其網路圖 | 25 |
| 3.6 | 輸出檔案格式 | 26 |
| 4.1 | Greedy I 演算法流程圖 | 31 |
| 4.2 | 演算法實例之隨機專案排程網路圖 | 32 |
| 4.3 | 決定各活動最早開始時間之先序關係圖 | 36 |
| 6.1 | 應用 OR 節點於具時間限制之專案網路範例 | 48 |

符號說明

時間限制相關符號及變數：

- $begin(i)$ ：表示活動 i 在時窗 (time window) 限制下，最早能開始的時間。
- $end(i)$ ：表示活動 i 在時窗限制下，最晚能開始的時間。
- t_{ik} ：表示活動 i 在時間排程限制下，第 k 個能開始運行的時間。
- V ：網路圖中節點的集合。
- E ：網路圖中弧的集合。
- S ：排程中所有活動的開始時間 s_i 所組成的集合， $\forall i \in N$ 。
- T_w ：表示具時窗限制 (time window) 之節點所成的集合。
- T_s ：表示具時間排程 (time schedule) 之節點所成的集合。
- ESS_i ：表示活動 i 最早能開始運行之時間。
- LSS_i ：表示活動 i 最晚能開始運行之時間。

EWD1 模式相關符號及變數：

- PA ：表示網路圖中弧的集合。
- $P(i, j)$ ：表示網路圖中一條由節點 i 到節點 j 的路徑。
- TA (transitive closure)： TA 是弧集合 A 的傳遞閉包，也就是說 $(i, j) \in TA$ 若且唯若存在某些 $P(i, j)$ 。

- $\pi(i)$ ：表示活動 i 的前序活動之集合。
- $\sigma(i)$ ：表示活動 i 的後序活動之集合。
- s_i ：活動 i 的開始時間。
- d_i ：活動 i 運行所需時間。
- f_i ：活動 i 的完成時間。
- F_{ij} ：活動 i 和 j 之間所具有之彈性時間，定義為 $s_j(S) - f_i(S)$ 。
- ω ：專案的截止時間。
- λ_{ij} ：表示由活動 i 到活動 j 所需花費的最短運行時間，其中不包括活動 i 和活動 j 的運行時間。
- p_i ：活動 i 發生干擾的機率。
- Ψ_i ：由第 i 個活動可能會發生之干擾所組成的集合。
- l_{ik} ：活動 i 因為第 k 個干擾劇本，而產生的一段時間長度。
- c_i ：表示活動 i 超過其開始時間的每單位時間成本。
- Δ_{ijk} ：活動 i 由第 k 種劇本產生的干擾，而對活動 j 造成延遲的時間。
- $MSPF_{ij}$ (minimal sum of pairwise floats)：當圖 $G(N, A)$ 存在任意路徑 $P(i, j)$ ，則 $MSPF$ 表示此路徑上各邊的最小彈性時間 (float time) 之和。

第一章

緒論

排程的主要目的在於了解專案中各活動的資訊，以適當地安排活動開始時間。若能精確地掌握開始時間的資訊，將使得活動中合作的夥伴能更容易彼此配合，如此便能節省許多成本，也不會浪費不必要的時間。然而，現實生活中往往存在某些不確定性變因 (uncertainty) 使活動產生延遲，甚至停擺，因而造成額外的成本付出，所以管理者在安排好專案進度後，可能還需重新排程，來將不確定之變因所造成的影響減到最小。

最早用來解決專案排程問題的工具是甘特圖 (Gantt Chart)，不過此法無法考慮作業間的相依性以及分辨作業重要程度，一旦排程進行時有延誤發生，將無法得知此項延遲對專案所造成之影響。針對這些缺點，美國杜邦公司在1957年發展出要徑法 (critical path method: CPM) 和北極星飛彈計畫研究小組在1958年計劃評核術 (program evaluation and review technique: PERT) 來評估專案中各活動的最早開始時間和整個專案完成所需的最長時間，即找出關鍵路徑 (critical path)，而這兩方法至今仍被廣泛應用來處理專案排程的問題。

專案排程在各方面因素考量的問題，已經都有一定程度研究，而近年來因為網路科技的進步，虛擬組織的出現，使得競爭型態大為改變，各公司開始強調核心功能，非核心的功能則透過外包與外包商合作，而在此緊密的合作現象中，若有某個活動的排程發生延誤，將造成其後活動的嚴重損失，甚至使整個專案停擺或失敗。因此，不確定 (uncertainty) 因素方面的問題逐漸受到重視；另外，為了讓排程結果能更接近事實，時間限制和活動的特性也是需考慮的因素。

本研究之主旨即在於探討不確定性 (uncertainty) 的排程模式 (model)，並試著加入時窗 (time window) 和時間排程 (time schedule) 限制，最後設計適當的演算法求解之。

1.1 研究動機

Chen et al.(1997) 認為某些活動會被限制在某些時間內才能開始運行，因此提出時間限制 (time-constraint) 的觀念，圖 1.1 為一個 AON(activity on node) 的專案排程網路圖，該圖中有活動 0 至 9 共 10 個活動節點，各活動節點上之數字代表該活動運行所需花費的時間；此專案排程網路圖中活動節點 1、5、7 為具有時間限制的三個節點，該節點具有的時間限制類別分別標示於其下。例如活動圖中節點 1 的標示為 TS(16, 21, 28, 29)，代表活動 1 必須在單位時間 16、21、28、29 這四個時間點上才能開始運行；活動 5 和活動 7 為具時窗限制之節點，活動 5 下面標示為 TW(53, 71) 代表活動 5 必須在單位時間 53 到 71 的區間內才能開始運行；同理，活動 7 必須在單位時間 21 到 83 的時間區間內才能開始運行。上述為一個具時間限制之專案排程網路範例，而因現今企業中外包風氣之盛行，和外包商簽約決定交貨的時間可視為專案排程中時間限制方面之考量，由此可知時間限制應為專案排程中一重要的考量因素。

以不確定性方面而言，需要考慮專案運行過程中可能發生專案停止或延遲運行的情況，這種現象在如今高度分工之社會上可能會造成極大之損失；因此，如何最小化因不確定性所造成之損失，或者設計一個能吸收因不確定性所增加的額外時間之排程亦為一重之研究考量。因此，若能發展出一同時考慮時間限制和不確定性之排程模式，將更適合應用於現今專案排程問題。

1.2 研究目的

在過去的專案排程中，都是假設排程問題能有足夠且完整的資訊去求出活動的開始時間，並且設定該活動一定會依演算法所解出的開始時間去執行；但在現實中，排程活動常可能因為某些因素而延遲其開始時間甚至停止運行，我們將

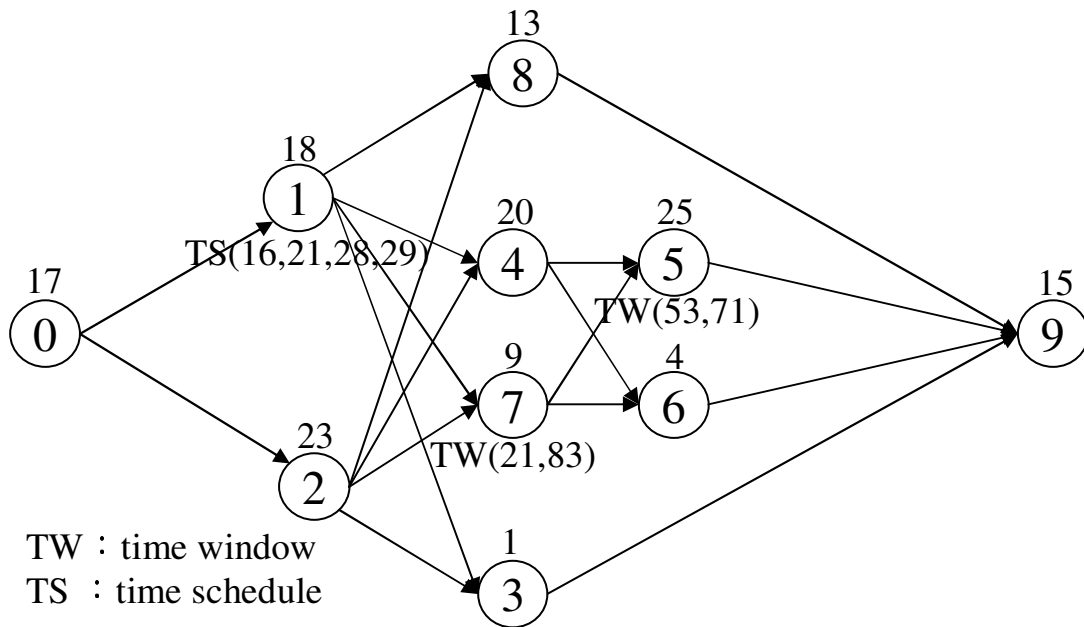


圖 1.1: 具時間限制之專案網路範例

這些因素統稱為不確定性 (uncertainty)。不確定性是專案排程必須面對之一重要問題，而要如何設計排程，以使不確定性發生時對我們的影響能減到最小，更是解決此問題的重點。

由於不確定性的存在，已經有許多學者著手去研究設計解決這類問題的方法，這些演算法的主要目的在於設計能決定排程中各活動的開始時間，使其排程在遇到不確定性時造成的損失能最小；但現實的排程問題中除了考慮不確定性外，尚須考慮排程中活動的特性，例如活動的時間限制。所以，本研究的主要目的在於發展一能考慮不確定性之排程模式，並將時間限制列入考慮，進一步發展出適合此模式之演算法來求解之。

1.3 研究流程圖

本論文之研究流程如下頁圖 1.2 所示：

1.4 論文架構

本研究的架構分為緒論、文獻探討、模式建立與演算法設計、數值分析與驗證、討論與未來研究等五大章節。在第一章中，主要介紹本論文的研究背景、

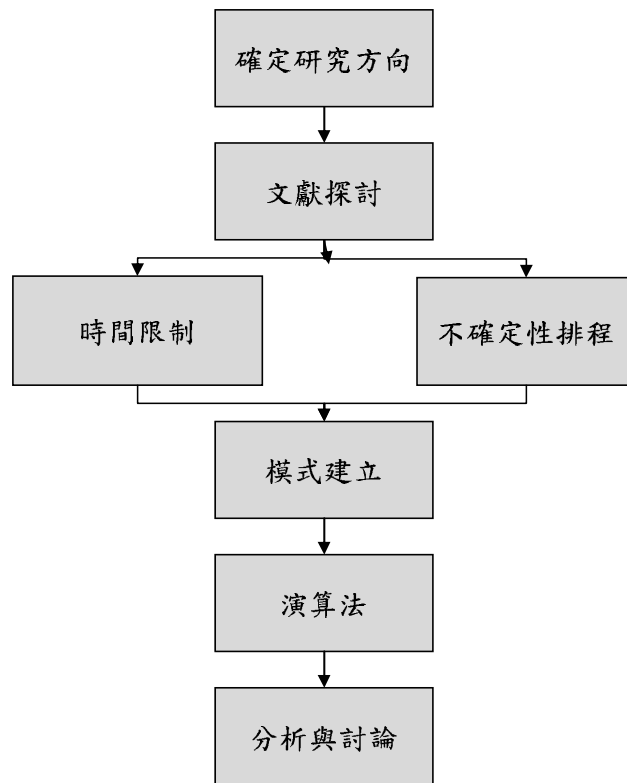


圖 1.2: 研究流程圖

研究動機、研究目的、研究流程、並在最後簡述本論文的研究架構。第二章文獻探討的部分，主要探討時間限制、AND/OR 節點先序順序限制和不確定性排程此三個領域相關之文獻。第三章設計一符合研究需求的具時間限制之隨機網路產生器。第四章將建立本研究之排程模式，並提出能求解此模式的演算法。第五章將設計適當的例子，測試本研所究提出的模式正確性並分析演算法求解的品質和效率。最後，第六章對本研究作總整理，討論本研究之結果並探討未來能研究及改善之方向。

第二章

文獻探討

本章主要回顧專案排程問題組成之要素、具 AND/OR 先序限制 (AND/OR precedence constraints) 之網路圖、具時間限制之活動網路圖 (activity network with time constraints)、穩定性專案基線排程 (stable project baseline schedule) 和基因演算法排程等相關之文獻。

2.1 專案排程問題組成之要素

Kolisch and Padman(2001) 整理出專案排成的四個組成要素，而此四要素分別為專案排程活動 (activity)、活動之先序關係 (precedence relations)、資源 (resources) 限制種類、和專案排程目標 (objectives of project scheduling)，以下分別回顧此四要素。

2.1.1 專案排程活動

在專案排程中，活動代表著為了使專案能順利完成的每個步驟，而這每個步驟在專案運行的過程中，都必須要確實的運行過一次；這些活動運行的表示方式通常為活動運行所需的時間 (duration)、需要資源的種類和數量或者現金流量的流入與流出等等，而專案排程問題大致是從以上幾種表示方式挑選一種或多種來代表其活動之意義。

2.1.2 活動之先序關係

在專案排程的活動中，有些活動是必須在某些活動運行結束才能開始運行，此為活動的先序關係。為了表示專案排程中各活動的先序關係，通常用箭頭

指向之方式於網路圖形中表示活動之先序關係，其中專案排程網路圖又分為弧上作業(activity-on-arc, AOA)和節點作業(activity-on-node, AON)兩種，因此其表示關係亦不同：

1. 弧上作業(activity-on-arc, AOA)：這類的網路圖是以弧來表示活動，圖上的節點是用來分隔活動和該活動的先序活動，而弧的順序表示活動間的先序關係。
2. 節點作業(activity-on-node, AON)：這類網路圖是以節點來表示活動，圖上的弧則是用來表示先序關係。如果某活動具有先序活動，則在表現該專案的網路圖中，表示該活動的節點必定會來自該先序活動節點的弧流入。

傳統的專案網路大都以 AOA 來表示，而 Hillier and Lieberman(2001) 指出 AON 專案網路比傳統 AOA 專案網路多了以下優點：

1. AON 專案網路比 AOA 專案網路容易建構。
2. AON 專案網路於圖形化表示比較直覺易懂。
3. 當專案排程需調整或改變時，AON 專案網路較 AOA 專案網路容易修正。

因以上的優點，AON 專案網路圖逐漸為使用者接受，是現在較普遍用來表示專案排程的網路圖。

2.1.3 資源限制類型

資源是專案排程中活動運行所需消耗之物品，Kolisch and Padman(2001) 將資源整理分為可恢復資源(renewable)、不可恢復(nonrenewable)資源、雙限(doubly constrained)資源、部分可恢復(partially renewable)資源四種，以下將簡介這四類資源類型：

1. 可恢復(renewable)資源：這類資源的特性為在一段時間後能恢復原本的數量，並供活動重複使用，如機器、設備等。

2. 不可恢復(nonrenewable)資源：這類資源在整個專案完成的時間中一旦消耗掉了就沒有了，如專案預算。
3. 雙限(doubly constrained)資源：具有總數量限制和某段時間內可用數量限制的資源，如專案預算，因為專案資本的規劃，可能會限定某期限內只能花費多少預算，而資本的總額也是有限的。
4. 部分可恢復(partially renewable)資源：指在某段時間內，其使用量被限制在一範圍內的資源。

2.1.4 專案排程目標

每個專案排程都有其想要達成之目的，Kolisch and Padman(2001)在他們所整理的文獻中將這些目的歸類為以下幾類：

1. 專案完成時間極小化(makespan minimization)：此為目標的專案排程，其目的在於使專案內的各活動能盡量在最早能開始運行的時間開始執行，使得最後一個活動的結束時間能最小，這類的專案排程問題是目前研究最常見的問題。
2. 成本極小化(cost minimization)：專案成本能分為活動成本(activity-cost)和資源成本(resource-cost)兩類，活動成本主要研究因活動而造成之成本，此一目標為最小化專案所有活動進行之總成本及專案逾期的延遲成本，Elmaghraby(1997)曾對於時間/成本抵換問題(time/cost trade-off problem)進行研究，之後De et al.(1995)和Skutella(1998)也提出許多關於成本極小化的相關的研究。
3. 淨現值極大化(net present value maximization)：在整個專案排程中，若各活動中的現金流動顯著且頻繁的話，就必須考量淨現值的問題。此乃因為在這樣的條件下，拿淨現值來當衡量專案效能的指標是比較具有意義的；這類問題的主要概念是考慮專案完成時點的整體利潤，以判斷整個專案的投資效益，Bey et al.(1981)曾提出這類問題的研究。

4. 品質極大化 (quality maximization) : Icmeli and Rom(1997) 首先明確提出「品質」做為專案排程之目標，其品質極大化的定義為專案活動重新進行 (rework) 所花的時間與成本最小。

2.2 AND/OR 先序限制相關文獻探討

在具有 AND/OR 節點的網路圖上，AND 節點是傳統在 PERT/CPM 排程模式用的節點，該節點必須等到其先序活動節點全部執行完才能開始運行，而這類節點的運用無法滿足現實中的其他需求之應用，是故有 OR 節點的產生。OR 節點之特性為只要有其某一先序活動節點完成，則該節點的活動就能開始運行，這類的問題目的通常在於找尋各活動的開始時間，和專案排程的關鍵路徑，以下將回顧 AND/OR 先序限制相關之文獻。

2.2.1 AND/OR 先序限制模式

AND/OR 網路圖之系統通常由 $\langle G, S, D \rangle$ 所組成，它通常以 AOA(activity on arc) 網路圖來表示，但在本論文中為方便之後研究的進行，一律將 AOA 的網路圖轉換為 AON(activuty on node) 的網路圖來表示；其中令 $G = (V, E)$ 表示一有向網路圖， V 為圖上節點 (node) 之集合， $|V| = n$ ，而 n 為圖上節點的總數； E 為圖上弧 (arc) 的集合； $|E| = p$ ，而 p 為圖上弧的總數； S 是活動開始時間的集合；第 i 節點的開始時間以 s_i 表示； D 為活動時間的集合， d_i 表示活動 i 運行所需時間。AND 節點通常用 N_A 表示，OR 節點用 N_O 表示，故我們可假設 $V = N_A \cup N_O$ 。這類問題主要是要去求各節點的開始時間 s_i ，而從 George and Eugene(2002) 可知對於所有的 $i \in V$ 要求解 s_i 的最佳化條件如下：

$$s_0 = t_0, \quad (2.1)$$

$$s_j \geq \max_{j \in \sigma(i)} (s_i + d_i), \quad \text{if } j \in N_A, \quad (2.2)$$

$$s_j \geq \min_{j \in \sigma(i)} (s_i + d_i), \text{ if } j \in N_O, \quad (2.3)$$

$$s_i \geq 0. \quad (2.4)$$

上列的最佳化條件意義分別說明如下；式子(2.1)表示給定整個排程第一個開始活動 s_0 一個初始時間 t_0 ；式子(2.2)表示活動 j 屬於AND節點時的最佳化條件，活動 j 是活動 i 的後序活動，即 $j \in \sigma(i)$ ，其中 $\sigma(i)$ 表示活動 i 後序活動之集合，因此活動 j 的開始時間是大於或等於其前序所有活動的開始時間加上個別的活動運行時間，亦即活動 j 必須等其所有前序活動完成後才能開始運行；式子(2.3)表示活動 j 屬於OR節點時的最佳化條件，依此式表示活動 j 的開始時間會大於或等於其所有前序活動之開始時間加上其個別活動的運行時間中最小的那個時間，亦即只要活動 j 之所有前序活動中有某一活動完成，則活動 j 即可立即開始；最後，式子(2.4)限制所有的活動開始時間必須大於或等於0。

2.2.2 AND/OR 節點之應用領域

由於AND/OR節點的特性不同，使得網路圖中計算最短路徑(shortest-path)和關鍵路徑(critical-path)之方法必須隨之調整，而這類的問題在現實生活中亦已經有許多應用實例。Crowston(1970)認為製造流程規劃已經逐漸產生具AND/OR節點特性的排程問題，Goldwasser and Motwani(1999)應用AND/OR節點特性的網路圖於組裝(assembling)和分解(disassembling)系統；Gillies and Liu(1995)也使用AND/OR網路圖於求解電腦溝通系統的及時(real-time)排程任務中。

2.3 具時間限制之活動網路

本小節將針對時間限制之定義和具時間限制之求解方法來回顧時間限制相關之文獻。

2.3.1 時間限制定義

專案排程的研究通常假設排程中各活動只要所安排的時間一到就能立即執行，然而，在實際的生活中，有些活動有其各自的時間限制，也就是說這些活動只能於其規定的時間限制內才能開始運行；若於行程安排時沒考慮這些時間限制的條件，則可能會造成排程上的錯誤。例如，某一不考慮時間限制的排程其關鍵

路徑所需的時間為 20 小時，但如果在該關鍵路徑上某一活動因時間限制的關係導致該活動的開始時間延遲，進而使得其關鍵路徑的運行時間會超過 20 小時。針對此時間限制的設定方面 Chen et al.(1997) 提出時窗限制 (time window) 和時間排程 (time schedule) 兩種時間限制，分別說明如下：

1. 時窗限制 (time window)：具有時窗限制的活動將被限制於某一時間區間內才能開始運行，令 $begin(i)$ 與 $end(i)$ 分別表示活動 i 被限制之最早及最晚開始時間，則 s_i 之範圍可用下式表示：

$$TW_i = (begin(i), end(i)) \quad (2.5)$$

2. 時間排程 (time schedule)：具時間排程限制的活動將被限制於某些時間才能開始運行，令 $t_{i1}, t_{i2}, \dots, t_{in}$ 表示該活動 i 能開始運行的 n 個時間點，則 s_i 之範圍可用下式表示。

$$TS_i = (t_{i1}, t_{i2}, \dots, t_{in}) \quad (2.6)$$

2.3.2 具時間限制之排程求解方法

Chen et al.(1997) 提出具時間限制之排程問題，並模仿二階段 (two-phase)，求解此問題，該演算法的第一階段 (first phase) 找出每個節點的最早開始時間和從起點到終點的最長路徑 (critical path)；第二階段 (second phase) 找出每個節點的最晚開始時間。

在第一階段中，該演算法首先依照拓撲順序 (topological order) 由升序來計算各節點的最早開始時間，其中和傳統二階段不同的地方在於遇到具時間限制之活動的處理方式。當處理具時間排程 (time schedule) 之活動 i 時，先從給定時間 $(t_{i1}, t_{i2}, \dots, t_{in})$ 中找出一時間點 t_{ik} ，其中 t_{ik} 要大於或等於活動 i 原本預期的開始時間，否則表示此問題無可行解；找出 t_{ik} 後，緊接於活動 i 之後的活動 j 之開始時間 s_j 就等於 t_{ik} 加上活動 i 的運行時間 d_i ；當處理具時窗限制 (time window) 之活動 i 時，若節點 i 的開始時間小於該活動受限制之開始時間 $begin(i)$ 時，則緊接於活動 i 之後的活動 j 的開始時間就等於 $begin(i)$ 加上該活動的運行時間 d_i ；若活動 i 的

開始時間 $early(i)$ 在活動所限制的開始時間內，即 $begin(i) \leq early(i) \leq end(i)$ ，則活動 j 的開始時間就等於 $early(i)$ 直接加上活動運行的時間 d_i ；但若節點 i 的開始時間 $early(i)$ 大於該活動限制時間的最後開始時間 $end(i)$ ，則表示此問題無可行解。

在第二階段中，該演算法依拓撲順序的降序，從最後的節點往回求各節點的最晚開始時間 $latest$ ，而判斷具兩類時間限制之弧的方法在 Chen et al.(1997) 的文獻中已有詳細說明，在此不贅述；但該演算法僅適用於傳統只考慮 AND 節點的網路圖。若考慮具 OR 節點之網路圖，則該方法在第二階段中判斷最晚開始時間時會出現矛盾的現象，若能修改該演算法將能使其更符合實際生活的運用。

2.4 不確定性排程相關文獻探討

過去許多關於專案排程的研究，大都事先假定該專案排程有完整的資訊來規劃排程並且假定該專案是在一穩定環境下來進行排程，即在排程的過程中不會發生其他的狀況影響排程；但在現實的排程過程的確可能存在一些會影響排程結果的不確定因素，而這些不確定因素可能來自於資源短缺、活動運行時間的增長或縮減、準備時間或截止時間的變更等任何會使活動時間更改的狀況，而這些狀況皆會使得原本預定好的計畫產生一定程度的成本損失，因此如何考慮具不確定性排程的問題是專案排程中的研究重點之一。Herroelen and Leus(2005) 將不確定性排程相關研究整理分為五種，分別為被動排程 (reactive scheduling)、隨機專案排程 (stochastic project scheduling)、模糊專案排程 (fuzzy project scheduling)、敏感度分析 (sensitivity analysis) 和穩定性專案基線排程 (robust baseline scheduling)。以下將回顧這些不確定排程之相關文獻，其中穩定性專案基線排程與本研究主題相關程度較大，故於下節中針對穩定性專案基線排程相關文獻作較為深入之探討。

2.4.1 被動排程 (Reactive scheduling)

被動排程並非在安排基線排程 (baseline schedule) 時就試著去處理不確定的狀況，而是當考慮之不確定性狀況發生時才去修正或者重新最佳化基線排程，其目的主要在於盡力調整基線排程使其能適用於具不確定因素之環境。

被動排程在規劃排程方面有許多不同的規劃策略，其一主要之規劃策略活動修復動作 (schedule repair actions) 以設計一簡單之機制使各活動能一致性調整來達成快速排程之目的，也就是說針對各活動在發生不確定性狀況時用一簡單的調整方式來規劃該活動。應用這類策略的文獻有 Sadeh et al.(1993) 所提出之右移法則 (right shift rule)，此法則為當排程因為資源狀況產生之中斷或因順序關係產生之中斷時，會及時的移動調整那些被影響而中斷之活動，以減少這些中斷造成之損失。但是這方法被指出其排程結果是不好的，因為它並沒有重新整合各活動的排程結果。

此外，另有一種規劃策略會在某個重新調整過的活動執行完畢後，重新調整整個排程的運行狀況，而通常會使用整個排程運行所花費的時間來當此排程策略的衡量指標。一般而言，排程修復是一種啟發式的排程過程，如果排程的目的在於產生一個和原本預期的排程差距盡可能最小的新排程的話，那必須在該排程活動遇到不確定性的狀況下去設定一重新排程的機制以達成其目的 (其中，特別針對在和原本預定排程差異最小且最小化總運行時間這類機制設定的相關文獻被歸為穩定性專案基線排程，將於下節回顧其相關文獻)。El Sakkout and Wallace(2000) 提出這類最小化干擾造成影響的策略 (minimum perturbation strategy)，此策略主要在於設計一準確的局部最佳化演算法來使各活動經過修正的開始時間和原本預定的開始時間差異之總和為最小。另外，Calhoun et al.(2002) 使用初始化目標式和最小化改變的活動數量之目標規劃 (goal programming) 模式來校正專案排程。

2.4.2 隨機專案排程 (Stochastic project scheduling)

在專案排程的文獻中針對隨機專案排程的相關研究相對較少，大部分文獻著重於具資源限制之隨機專案排程。在具資源限制之隨機專案排程中，其各活動

之運行時間不確定(隨機產生)，而這類排程的主要目標在於最小化各活動預期需要的運行時間並使這些活動能滿足零延遲(zero-lag)和可更新(renewable)資源的限制。而對於給定資源限制和隨機活動運行時間的隨機專案排程而言，其求解方法為排程政策(scheduling policies)或者排程策略(scheduling strategies)，而不同於以往應用基線排程(baseline schedule)之專案排程。

Igelmund and Radermacher(1983)定義了排程政策(scheduling policies)，此政策定義了一決定點(專案的開始時間)和各活動的完成時間。經由排程政策決定此兩要素後，即可整理出此排程各活動的開始運行時間和整個排程的運行時間。此類應用排程政策求解隨機專案排程問題的文獻之目的在於設定各種排程的政策以最小化專案預期花費的總時間。Radermacher(1985)提出一種使用最小禁用集合(minimal forbidden sets)的早開始政策(early start policies, ES policies)。最小禁用集合是指不具先序關係之配對活動的最小聯集合(inclusion minimal sets)不能同時進行排程，因為他們共用被限制的資源。因此，在早開始政策中，活動 j 即須等待其先序活動全部運行完畢並有足夠資源後方能開始運行，而依此政策即能逐一的決定出各活動的開始時間之列式。Igelmund and Radermacher(1983)介紹了事先選取政策(pre-selective policies, PRS)，在事先選取政策中，活動 j 若要能開始運行，只須其先序活動中有一運行完畢並有足夠資源即可開始運行。針對上述兩種求解隨機排程的排程政策，Stork(2000)實作了分支定界(branch and bound)法來求解。

2.4.3 模糊專案排程(Fuzzy project scheduling)

在模糊專案排程的文獻中，學者們質疑直接用機率分配來表示各活動的運行時間是不正確的，因為用來表示此機率分配的資料缺乏歷史參考資料。因此，模糊專案排程方面的文獻主要是使用模糊數值而非隨機變數來模式化各活動的運行時間。

在模糊專案排程中，會定義出一歸屬函式(membership function)來表示各活動運行時間可能分配的程度，而用模糊數(fuzzy number)來表示各活動的運行時間，其中模糊數為所定義歸屬函式之子集合。因此，若能更精確地定義該歸屬函

式則此模糊排程的結果將會更精確；然而 Hapke et al.(1999) 指出模糊數很難定義出來。此外 Rommelfanger(1990) 提出一個取得適合歸屬函式之模糊資料的實用方式；他建議針對歸屬函式中一些比較重要的參數依照悲觀和樂觀的資料為其設計一特殊的區間來加以衡量，如此將使採用的歸屬函式更貼近事實。

2.4.4 敏感度分析(Sensitivity analysis)

敏感度分析主要在分析「如果...將如何」此類問題的，該議題著重於調整問題中某些參數並觀察結果會如何改變。Hall and Posner(2000) 研究了多項式可解 (polynomially solvable) 和棘手機器排程問題 (intractable machine scheduling problems) 並試著來回答以下五個敏感度分析中常見之基本問題：(a) 某些參數其調整結果不影響最佳化解的調整極限在哪？(b) 給定某些特定的參數，會產生哪些新的最佳化成本？(c) 如果在某些特別的條件改變某些參數，是否會有更好的解？(d) 什麼情況下基線排程仍然維持最佳解？(e) 如何能使目標函式保持最佳解？(f) 什麼類型的敏感度分析能對穩定性最佳解有幫助。而敏感度分析的主要研究方向在於如何設定一敏感度分析之機制，以使管理人員能直接使用敏感度分析預期排程中各參數調整可能會產生之結果。

2.5 穩定性專案基線排程相關文獻探討

考慮不確定性和管理風險的專案排程是近幾年來逐漸被重視的議題，Bowers(1995) 指出在多重專案的環境下，主要的員工或設備都必須要能事先確定其可獲得性（也就是說需要時要能盡快取得該資源），因此要如何安排能吸收因干擾產生的延遲時間之事先排程 (pre-schedule) 已經變的非常重要。另外，Mehta and Uzsoy(1998) 說明這樣的事先排程有兩個重要的功能；一為能適當的分配資源來最佳化某些效能，二為能將此排程當成規劃供應鏈外部活動的基本參考；由此可知這樣的排程對於不確定性環境的重要性，接下來將回顧一些穩定性專案基線排程之相關文獻。

2.5.1 LPH 排程模式

在工廠的穩定性專案基線排程方面，Mehta and Uzsoy(1998)提出了LPH(linear programming based heuristic)的排程模式，此模式主要目的在於考慮調整排程中的額外的閒置時間(idle time)，將此時間插入排程中來吸收因故障(breakdown)或其他因素造成活動延遲之時間，並加入提早開始和延遲開始活動的懲罰(penalty)，以使活動能盡量在規劃的時間內運行。LPH模式旨在發展一個包含足夠彈性時間(slack time)之排程，並使得該排程的結果和沒包含此彈性時間之排程的差異為最小。其後，Herroelen and Leus(2004)指出該此模式的對偶(dual)為一最小成本流量問題(min cost network flow problem, MCNFP)。

2.5.2 ADFP 模式

若皆以各活動的最早開始時間來進行排程的話，雖能減少整體活動時間延遲的風險，但相對於各活動的折扣成本(discounted cost)而言卻會增加。此乃因為若將活動安排的很緊密，一旦發生活動延遲的現象時，將會造成某種程度的損失，因此在活動整體時間的安排和活動折扣成本之間要如何取捨即為一個重要的議題。Tavares et al.(1998)提出彈性係數(float factor)的概念並依此建構出ADFP模式(The activity-dependent float factor model)，此概念能用來對於考慮專案整體延遲的風險和折扣成本安排給各活動一個適當的彈性時間(float time)，以取得最佳的綜效。

令某排程之活動 i 的最早開始時間為 ESS_i 、開始運行時間為 s_i 、而最晚開始時間為 LSS_i ，則此三者的關係自然滿足 $ESS_i \leq s_i \leq LSS_i$ 。經由此關係的延伸，Tavares et al.(1998)證明可用 $s_i := ESS_i + \alpha(LSS_i - ESS_i)$ 求出一可行之排程，其中 $\alpha \in (0, 1)$ 即為彈性係數。對於 α 的設定以及考量活動彈性時間的安排和活動折扣成本兩者取捨之內容，可參考Tavares et al.(1998)發表之文獻，在此不贅述。

2.5.3 EWD1 排程模式

Leon et al.(1994)認為考慮會產生一干擾(disturbance)之專案排程模式有其重要的意涵，因此提出一目標為最小化事先排程專案之完成時間(makespan)與

最小化預期完成時間之延遲 (expected makespan delay) 之模式。Herroelen and Leus(2004) 繼續 Leon et al.(1994) 的想法，發展出一個僅考慮一次干擾之最佳化排程模式 EWD1(expected weighted deviation)。

Herroelen and Leus(2004) 用來衡量事先排程穩定性 (stable) 的指標是以活動預期開始時間和實際經排程後確定之開始時間的差異來衡量，故其目標式可視為最小化 $\sum_{i=0}^{n-1} c_j(ES_j - s_j)$ ，其中 ES_j 表示活動 j 經過排程後實際的開始時間，而後在考慮干擾發生的機率和干擾導致延遲的時間長度後，所發展出之模式如下：

$$\min \sum_{(i,j) \in TA} \sum_{k \in \Psi_i} c_j p_i g_i(l_{ik}) \Delta_{ijk} \quad (2.7)$$

$$\text{s.t. } s_i + d_i + F_{ij} = s_j \quad \forall (i, j) \in A, \quad (2.8)$$

$$s_n \leq \omega, \quad (2.9)$$

$$l_{ik} - MSPF_{ij} \leq \Delta_{ijk} \quad \forall (i, j) \in TA, \quad \forall k \in \Psi_i, \quad (2.10)$$

$$MSPF_{ii} = 0 \quad \forall i \in N, \quad (2.11)$$

$$MSPF_{ij} \leq F_{ik} + MSPF_{kj} \quad \forall (i, j) \in TA, \quad \forall k \in \sigma(i) \cap (\pi^T(j) \cup j). \quad (2.12)$$

此模式經過整理後，Herroelen and Leus(2004) 發現其對偶問題 (dual problem) 為一最小成本流問題 (MCNFP)，因此該模式能用解 MCNFP 之演算法去求解。

由於此模式為考慮各活動節點只會發生一次干擾之模式，該模式能作為以後更多問題探討之基礎，因此本研究以此模式為基本架構，延伸此模式使其將具時間限制之活動節點列入考慮以發展新的模式，並發展適當的演算法求解。

2.6 小結

在本章回顧的文獻中，我們首先檢視了專案排程活動組成之各要素；接著再討論文獻中時窗限制和時間排程限制兩種具時間限制之專案排程研究；最後回顧了不確定性排程中各種排程，其中穩定性專案基線排程 (robust baseline scheduling) 主要探討排程中如何在具不確定性的情況下考慮排程，並使其考慮不

確定性後的排程和基線排程 (baseline schedule) 的差異能最小，此與本研究之研究方向相關，因此我們特別對於穩定性專案排程之相關文獻作較為深入之回顧。

而在穩定基線排程的相關研究文獻中，Herroelen and Leus(2004) 針對各活動可能因干擾情況而發生延遲之不確定狀況而提出之 *EWD1* 模式與本研究之目的相同，因此本研究將整合上述之時間限制與此 *EWD1* 之模式，嘗試發展一具時間限制之穩定性專案基線排程模式，最後並發展適當的演算法來求解。



第三章

隨機專案排程網路產生器之設計

本論文所探討之隨機專案排程問題擬以 Herroelen and Leus(2004) 所研究之不確定性排程模式 *EWD1* 為基礎，再進一步考慮相關之時間限制，在 Herroelen and Leus(2004) 所發表的文獻中，他們使用 Demeulemeester et al.(2003) 所設計的 RanGen 隨機網路產生器以產生 AON 隨機網路。RanGen 能產生基本的專案排程網路圖形，但缺少了時間限制方面的設定，必須加以修改才能符合本論文之需求。本章將先回顧網路產生器相關文獻，說明引用 RanGen 網路產生器的原因；於第二小節將介紹 RanGen 的基本架構，而第三小節將說明我們如何以 RanGen 為基礎來設計符合本研究需求的專案排程網路圖。

3.1 專案排程網路產生器相關文獻回顧

由於專案排程問題通常使用活動網路(activity network) 來表達其各活動間的順序關係，因此網路產生器只要先設計出基本的專案活動網路圖，再加上問題所要求的額外限制，即可產生符合研究需求的隨機專案排程網路。

Elmaghraby and Herroelen(1980) 以求解專案排程網路所花費的時間多寡為基礎，設計個名為 CPU-time 的控制參數，來作為求解所產生之隨機專案排程網路之困難度衡量標準。亦即，CPU-time 被設定為較少(或較多)的專案排程網路理應更簡單(或困難)求解。如果 CPU-time 這個指標能在一開始設定，那我們就能針對專案排程的問題，找出最快速的解決方法，不用擔心我們的求解方法是因為本身沒效率，還是因為網路圖本身複雜度太高而造成求解方法的沒效率。

以網路圖形而言，有許多學者提出應該用各式的指標來衡量網路圖形的特性和效能，在接下來的文獻中將逐一介紹這些指標，並回顧針對這些指標所設計之網路產生器。Patterson(1984)最早發展出能讓許多專案排程問題使用的基本網路產生器，不論專案排程問題多困難或多複雜皆能拿該產生器產生的網路圖資料應用。不過，該網路產生器缺少控制網路圖產生的一些重要參數，無法衡量其產生之網路圖的特性更無法控制網路圖的複雜性；因此，其後之網路產生器的設計重點逐漸將能決定網路圖特性的指標設計列入考慮。

在過去的文獻中，Pascoe(1966)曾提出網路複雜度係數 (coefficient of network complexity, CNC) 來衡量 AOA 圖形的複雜程度，此衡量係數是對於弧和節點的綜合比較所設定出來的。Demeuleester et al.(1993) 考量 Pascoe(1966) 提出的 CNC 參數設計出 AOA 的網路產生器，該網路產生器可在指定弧和節點的個數之後產生出隨機性的 AOA 網路圖。之後，由於 AON 網路圖應用愈來愈廣，Davies(1973) 重新將 CNC 參數定義於 AON 的網路圖上；Kolisch et al.(1995) 將資源相關的參數列入考量設計出一個名為 ProGen 的 AON 網路產生器；Schwindt(1995) 以 ProGen 為基礎，並加入最大和最小的時間標籤 (time tag) 以產生三類不同資源限制的排程網路圖。然而他亦指出其所設計的 ProGen/MAX 網路產生器尚應考慮由 Mastor(1970) 所提出之 OS (order strength) 指標，該指標為具順序關係之弧除以該網路圖理論上最大的順序關係弧總數 $(n(n-1)/2)$ ，其中 n 表示該網路圖中節點之總個數)，OS 指標可被視為弧之總數相對於對於整個專案排程網路圖形之密度，其定義將在 3.2 節中詳述。

Agrawal(1996) 提出複雜度指標 (complexity index, CI) 的概念，並設計一個可輸入 CI 參數的 DAGEN 網路產生器，以隨機產生 AOA 專案排程網路圖。

本小節所回顧到的三個網路產生器 (ProGen, ProGen/MAX, DANGEN) 雖然有將 CI 和 OS 兩參數列入考量，卻無法保證其每次所產生之網路圖皆為唯一；也就是說，早期的專案排程產生器皆無法滿足強隨機 (strongly random) 性。直到 Demeulemeester et al.(2003) 才整合了 CI 和 OS 兩項參數，並設計出符合強隨機性之 RanGen 專案排程網路產生器。由於 RanGen 是目前考慮最完善且具強隨機性之

專案排程網路產生器，因此本論文擬以RanGen為基礎，加入時間限制以產生所需之網路圖形。

3.2 RanGen網路產生器

本小節將對RanGen之主要參數及其設計方法做介紹。在AON專案網路中，通常用 $G = (N, A)$ 來表示網路圖， G 代表網路圖， N 表示網路圖中節點的集合， A 表示網路圖中弧的集合。而節點的順序關係(precedence relations)則通常會用一個上三角的順序矩陣(precedence matrix, PM)來表示之(如圖3.1)。在順序矩陣中，當節點 j 是連結於節點 i 之後的節點時(也就是說節點 j 是直接或者間接連結在節點 i 之後)，則在關係矩陣中用 $P_{i,j} = 1$ 來表示之；反之，則 $P_{i,j} = 0$ 。圖3.1中，節點1之後連結至節點2，而節點3和節點5則是間接地連接到節點1，所以此網路圖的順序矩陣中 $P_{1,2}$ 、 $P_{1,3}$ 和 $P_{1,5}$ 皆為1。用關係矩陣來表示專案網路，主要有兩個優點：(1)在關係矩陣中絕對不會找到一活動節點其代表的數字比其前序節點(predecessor)還小；(2)OS指標將更容易自關係矩陣中計算而得，其中OS由Mastor(1970)所定義。以圖3.1為例，其OS值是由具順序關係之弧的個數(5)除以該網路圖理論上最大的順序關係之弧總數 $(5(5-1)/2 = 10)$ 而得，亦即 $OS = 5/10 = 0.5$ 。

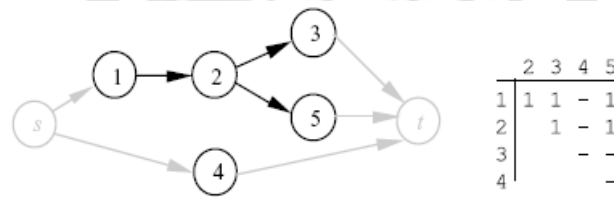


圖 3.1:隨機專案排程網路圖例及其順序矩陣(來源:Demeulemeester et. al., 2003)

然而，使用關係矩陣亦有其缺點。以圖3.2為例，可看出圖3.2(a)和圖3.2(b)之拓撲結構(topological structure)相同，但其各自的關係矩陣卻因節點4與節點5之位置不同而相異。由於一般研究通常需要產生拓撲結構不同(而非僅是關係矩陣不同)的網路，因此Demeulemeester et al.(2003)提出一演算法將具有相同拓撲結構但不同關係矩陣之關係矩陣群加以整合，最後輸出一個唯一之關係矩陣，

經過此演算法處理所產出之關係矩陣稱為標準關係矩陣 (standardized precedence martix, SPM)。由於此演算法與本研究無直接關聯，因此不於此詳述。

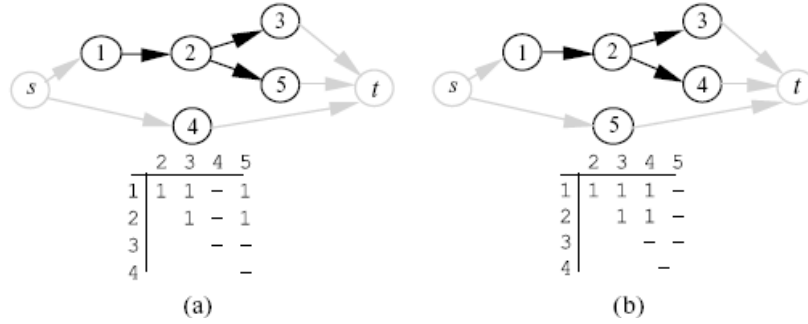


圖 3.2: 具相同關係矩陣之兩相異專案排程網路圖 (來源:Demeulemeester et al., 2003)

在RanGen中，定義了 GN 和 AN 兩種不同的集合。其中， GN 表示已經被產生出來之網路圖的集合，而 AN 表示以AON之格式所產生出的活動網路。Demeulemeester et al.(2003)提出之RanGen演算法流程簡述如下：

Algorithm 1 RanGen 演算法流程

```

procedure generate( $OS$ );
 $GN = \emptyset$ ;
Repeat
     $AN = \text{remove\_arcs}(OS)$ ;
    Unique\_representation( $AN$ );
    If ( $AN \notin GN$ ) then
        Save the network:  $GN \cup \{AN\}$ ;
        Transform  $AN$  into an AOA format;
        Compute  $CI$ ;
Until bound or time limit;
Select a number of networks with prespecified  $OS$  and  $CI$ ;
Return;

```

1. 執行 $\text{remove_arcs}(OS)$ 程序以根據所輸入之 OS 值產生一個網路 AN 。以圖 3.1 與圖 3.3 為例 ($OS = 0.5$)，一開始RanGen將 OS 值設為 1，可得到圖 3.3(a) 之關係矩陣，接著我們隨機刪除某些節點間的順序關係，譬如 (3,4)、(4,5)、(2,4)、(1,4) 及 (4,5) 等等，如此即可依序得到圖 3.3(b)-(f) 的關係矩陣，直至 $OS = 0.5$ (即圖 3.3(f)) 為止，最後將圖 3.3(f) 所對應之網路圖 AN 輸出。

| | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 |
| 2 | | 1 | 1 | 1 |
| 3 | | | 1 | 1 |
| 4 | | | | 1 |

(a)

| | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 |
| 2 | | 1 | 1 | 1 |
| 3 | | | - | 1 |
| 4 | | | | 1 |

(b)

| | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 |
| 2 | | 1 | 1 | 1 |
| 3 | | | - | - |
| 4 | | | | 1 |

(c)

| | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 |
| 2 | | 1 | - | 1 |
| 3 | | | - | - |
| 4 | | | | 1 |

(d)

| | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 1 | 1 | 1 | - | 1 |
| 2 | | 1 | - | 1 |
| 3 | | | - | - |
| 4 | | | | 1 |

(e)

| | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 1 | 1 | 1 | - | 1 |
| 2 | | 1 | - | 1 |
| 3 | | | - | - |
| 4 | | | | - |

(f)

圖 3.3: 由 `remove_arcs(0.5)` 產生圖 3.1 之網路圖 (來源: Demeulemeester et al., 2003)

2. 執行 `Unique_representation(AN)` 程序以產生 *SPM* 之演算法來檢查前步驟所產生出來之網路圖 *AN* 是否為唯一 (uniqueness)。如果所產生的網路圖 *AN* 的確唯一，則會將 *AN* 加入 *GN* 的網路圖集合。
3. 針對每次所產生的新網路圖 *AN*，使用 Bein et al.(1992) 所設計之演算法來計算 *CI* 參數。
4. 依使用者所輸入的 *OS* 和 *CI* 值隨機選取出所要求之網路圖，完成最後的輸出。

3.3 具時間限制之專案排程網路產生器

在本節中，將詳述本研究如何應用此網路產生器以產生具時間限制之隨機專案排程網路，並以一簡單的例子說明之。

3.3.1 時間限制之設計概念

本研究探討具時間限制之專案排程問題，時間限制是指某項活動在運行中可能被時間所限制，而被限制的情況又可分為時窗限制 (time window) 和時間排程限制 (time schedule) 兩種。具時窗限制的活動會被限制於某一時段才能開始執行；而具時間排程限制的活動會被限制於某些時間點方能執行。而於本節討論之網路產生器之主要目的即於 `RanGen` 產生之網路圖加上時間限制之資訊。另外，在過去具時間限制之專案排程問題中，如果該專案排程問題沒有專案截止時間 (deadline) 的限制，則各活動之時間限制可任意隨機產生，如此之專案排程皆能有其可行解。本研究探討之專案排程模式一開始即有其截止時間之限制，如此一

來，若各活動之時間限制仍隨機產生，則可能造成該專案排程模式某些活動無法在其時間限制內開始運行，甚至整導致個專案之運行時間超過其截止時間，使得此專案排程無可行解；由於無可行解之網路圖對本研究之數值分析毫無意義，因此，如何設計隨機產生專案活動的時間限制以使該網路圖恆有可行解，亦為一個重要的設計考量。

本產生器之運作過程如圖 3.4 所示，其第一步驟即讀入由 RanGen 隨機之產生網路圖檔來決定活動的運行順序，接著隨機產生各活動之運行時間；而當各活動的運行順序、活動運行所需花費之時間等參數被決定之後，即可用 Chen et al.(1997) 所提出之二階法來求得各活動的最早開始時間和最晚開始時間；接著即運用此資訊來隨機產生時窗限制和時間排程限制，最後並把這些資訊附加於 RanGen 輸出之文件檔上。

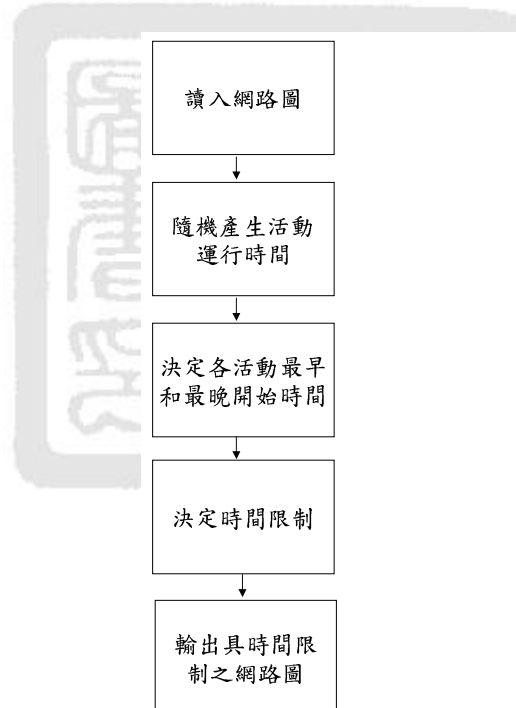


圖 3.4: 具時間限制之隨機網路產生器運程序

以下將詳述時窗限制和時間排程限制之設計過程。

1. 時窗限制：時窗限制是指該活動被限制於某一時間區間內，基於要避免隨機產生之時間限制導致專案排程無可行解，本產生器設計時窗限制之方式

如下：首先以即將產生時窗限制之活動的最早開始時間和最晚開始時間兩者之平均為中心點，隨機向左和向右產生一不對稱之時間區段，而此時間區段的長度設定為該活動之最晚開始時間和最早開始時間之差的0.5到1.5倍，此隨機產生之區間即為該活動之時窗限制。當各活動的開始運行時間位於其最早開始和最晚開始時間之區間內，表示該排程具可行解；而本產生器如此設定可確保所產生之時窗限制和活動之最早及最晚開始時間之間距必有交集，因此能避免所產生的時間限制導致無可行解之情形。

2. 時間排程限制：時間排程限制是指該活動被限制於某幾個時間點執行，因此設計方式如同時窗限制，在時窗限制之區段隨機產生完畢時，本產生器將隨機決定具時間排程限制之活動會受限於2到4幾個時間點，因此程式將於此時間區間內隨機取出適當的時間來當該活動被限制的時間點，而如此的設定方式也同樣的能避免造成無可行解的情況。

3.3.2 網路產生器應用簡例

以下將逐步簡述本網路產生器之操作：圖 3.5(a) 之表格為 RanGen 設定活動數為 10、OS 值為 0.5 和資源相關係數皆設為 0 所產生出來的網路圖輸入的文件檔，其左上第一行第一個數字 10 代表該網路圖共有 10 個節點，右邊的數字 0 代表具資源限制之節點數，因一開始在 RanGen 中設定此參數為 0 故輸出為 0。接著第二行中第一個數字 0 表示節點 1 無資源因素 (Resource factor, RF)，因為自 RanGen 中起始設定資源因素參數為 0，故此輸出為 0；其後的數字 2 代表節點 1 共連結至兩個節點，亦即其後的兩個數字 (分別為節點 2 和節點 3)；同樣地，第五行的數字分別由左而右依序代表節點 4 無資源因素限制，且節點 4 共連結到兩個節點 (分別為到節點 6 和節點 7)，此輸入檔之對應的網路圖如圖 3.5(b) 所示。

本網路產生器一開始即以上述 RanGen 輸出之檔案格式 (圖 3.5(a)) 當輸入文件檔，接著將隨機給定各活動的運行時間，令 $d[i]$ 表示活動 i 所需的運行時間，則本例中各活動運行所需之時間依序為 $d[1] = 17$ 、 $d[2] = 18$ 、 $d[3] = 10$ 、 $d[4] = 1$ 、 $d[5] = 20$ 、 $d[6] = 25$ 、 $d[7] = 4$ 、 $d[8] = 9$ 、 $d[9] = 13$ 、 $d[10] = 15$ 。決定好網路圖

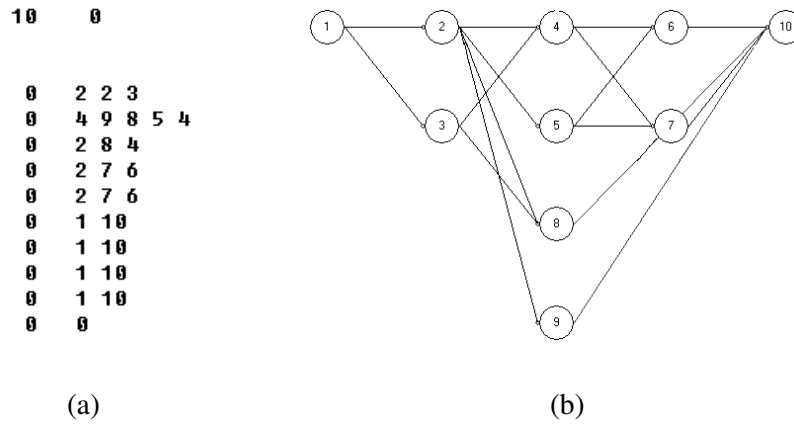


圖 3.5: RanGen 產生之檔案所繪成之網路圖 (a) 為輸出之檔案格式 (b) 為其網路圖

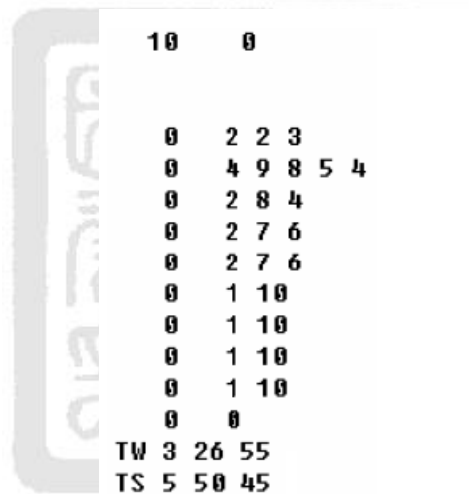
中各活動之運行優先順序和各活動之運行時間後，即執行 Chen et al.(1997) 之二階段法來求出決定各活動的最早和最晚開始運行時間。令 $ESS[i]$ 和 $LSS[i]$ 分別表示活動 i 的最早和最晚開始運行時間，則本例中各活動的最早開始時間依序為 $ESS[1] = 0$ 、 $ESS[2] = 17$ 、 $ESS[3] = 17$ 、 $ESS[4] = 35$ 、 $ESS[5] = 35$ 、 $ESS[6] = 55$ 、 $ESS[7] = 55$ 、 $ESS[8] = 35$ 、 $ESS[9] = 35$ 、 $ESS[10] = 80$ ，最晚開始時間依序為 $LSS[1] = 15$ 、 $LSS[2] = 32$ 、 $LSS[3] = 59$ 、 $LSS[4] = 69$ 、 $LSS[5] = 50$ 、 $LSS[6] = 70$ 、 $LSS[7] = 91$ 、 $LSS[8] = 86$ 、 $LSS[9] = 82$ 、 $LSS[10] = 95$ 。

決定好上述資料後，本產生器將依使用者之輸入來隨機產生時間限制，在本例中指定活動 3 將產生時窗限制，活動 5 將產生 3 個時間點之時間排程限制，其詳細設定如下：

1. 時窗限制：活動 3 的最早開始時間為 $ESS[3] = 17$ ，最晚開始時間 $LSS[3] = 59$ ，因此本產生器會依該活動最早和最晚時間之差來產生一長度，該長度為該活動的最早開始時間和最晚開始時間差的 0.5 到 1.5 倍，本例中此隨機產生之時間區段長度為 29；決定好該時間區段後，本產生器會取該活動的最早和最晚開始時間的中心點，並以此中心點和剛剛隨機產生之長度向左和向右隨機產生一不對稱的區間。在本例中，活動 3 的最早和最晚開始時間之中心點為 38，以該中心向左隨機產生之長度為 12，因為此區段長度為 29，如此以該中心向右之長度為 17，因此可產生出一 $[26, 55]$ 的時間限制。

2. 時間排程限制：活動5的最早開始時間為 $ESS[5] = 35$ ，最晚開始時間為 $LSS[5] = 50$ ，於本例中網路產生器會隨機產生一時間長度為18的區段，產生方式同時窗限制，並隨機決定此時間區段為 $[38, 56]$ ，而該活動被指定將受限於三個時間點，因此本產生器亦將於此時間區段隨機選出三時間點 $[40, 45, 52]$ 來當此活動的時間限制。

當時間限制產生完畢後，本產生器會將這些時間限制的資訊依設計好的格式附加於RanGen的輸出檔上，並以此檔案為輸出。對於具時窗限制之活動節點，本產生器以 TW 表示，因此本例中，對於活動3所產生之時窗限制的輸出為 $TW\ 3\ 26\ 55$ ；對於具時間排程限制之活動節點本產生器以 TS 來表示；於本例中，對於活動5所產生之時間排程限制之輸出為 $TS\ 5\ 40\ 45\ 52$ ，而此例子的輸出檔案格式如下圖3.6所示。



```

10    0

0     2 2 3
0     4 9 8 5 4
0     2 8 4
0     2 7 6
0     2 7 6
0     1 10
0     1 10
0     1 10
0     1 10
0     0
TW 3 26 55
TS 5 50 45

```

圖 3.6: 輸出檔案格式

3.4 小結

本章探討了過去對於網路產生器之相關文獻研究，最後因RanGen網路產生器具強隨機性、重要指標 OS 和 CI 可指定為任意輸入之參數兩優點，且其網路圖輸出符合本研究基本之需求；因此本研究針對RanGen之設計，於其中加入產生時間限制之機制來設計出符合本研究模式之具時間限制之隨機專案排程網路產生器，並預期將此產生器應用於後期數值分析上。

第四章

模式建立與演算法

本研究的目的是在於建立一能考慮時間限制和能吸收因干擾產生的額外時間使因干擾造成之時間成本最小化之排程模式，並發展適當的演算法來求解。本章首先詳細描述欲建立之模式及其特性，最後對此模式提出三個演算法 (Greedy I, Greedy II 和 GA) 來求解之。

4.1 具時間限制之穩定性專案基線排程模式

本節中首先介紹模式之建構目的與其需考慮之假設與限制，接著建構該模式並說明其特性。

4.1.1 模式建構目的與研究限制

在本研究中，模式建構的主要目的為：

1. 發展一能考量排程過程中因不確定因素致使活動運行時間延誤之專案排程模式。
2. 使此模式能考慮各活動可能被受限於某些時間範圍內方能執行之限制。
3. 依建構出來的模式設計適當的演算法來求解之。

綜合上述目的，本節將設定一個具時間限制之穩定性專案基線排程模式，此模式將考慮在專案排程中各活動可能會發生一次干擾來延誤該活動之運行時間，並依能最小化專案排程的損失為目標來決定各活動的開始時間，此模式有以下的假設和限制：

1. 假設專案排程中各活動的運行時間為已知且為固定常數。
2. 假設各活動間的優先順序關係為已知。
3. 假設各個活動運行期間只會發生一次干擾，而此干擾會隨機產生一段使活動運行時間延遲之時間。
4. 假設各活動因無法在預期時間內完開始運行而產生的成本為已知之固定常數。
5. 假設整個專案排程的截止時間為已知。

4.1.2 模式特性描述與建構

在虛擬組織的環境下，公司與外包商的合作通常是經由合約來制訂，在合約內就應會明訂外包商該完成委託者要求任務的時間，因此，若公司內部某些排程活動得透過此外包之任務才能達成，則該活動就會受到外包商完成該任務時間的影響，此可視為該活動的時間限制。時間限制又可分為時窗(time window)和時間排程(time schedule)限制等兩種限制。令 s_i 表示活動 i 的開始運行時間，則其限制式分別如下：

時窗限制 (time window)

$$TW_i = (begin(i), end(i)) \quad (4.1)$$

時間排程 (time schedule)

$$TS_i = (t_{i1}, t_{i2}, \dots, t_{in}) \quad (4.2)$$

此外，令 TW 與 TS 分別代表具時窗與時間排程限制之活動所構成的集合。

虛擬組織具有能和大公司抗衡的能力，能創造出品質不輸給大公司的產品。然而，由於虛擬組織以合作方式來完成其任務，因此，在專案排程方面要承擔比大公司多的風險；舉例來說，外包商可能會因為能力不足而無法在合約制訂的時間內完成工作；或者外包商在大客戶要求提前完工的壓力下，放棄完成某些

小客戶的工作；類似的狀況一旦發生將導致虛擬組織活動的延遲甚至停擺。因此，規劃一個能考慮因不確定性造成的延遲時間以使得組織的損失減到最小的排程是必需考慮的。本研究應用 Herroelen and Leus(2004) 發展出之穩定性專案基線排程模式 (EWD1) 為基礎：

(EWD1)

$$\min \sum_{(i,j) \in TA} \sum_{k \in \Psi_i} c_j p_i g_i(l_{ik}) \Delta_{ijk} \quad (4.3)$$

$$\text{s.t. } s_j - s_i \geq d_i, \quad \forall (i,j) \in A, \quad (4.4)$$

$$s_0 - s_n \geq -\omega, \quad (4.5)$$

$$\Delta_{ijk} + s_j - s_i \geq (l_{ik} + d_i + \lambda_{ij}), \quad \forall (i,j) \in TA, \quad \forall k \in \Psi_i, \quad (4.6)$$

$$\text{all } \Delta_{ijk} \geq 0; \quad \text{all } s_i \text{ unrestricted in sign} \quad (4.7)$$

根據之前對於模式特性的描述，我們在EWD1模式之後加入相關之時間限制如下：

(Time constraint)

$$\text{begin}(i) \leq s_i \leq \text{end}(i), \quad \forall i \in T_W, \quad (4.8)$$

$$s_i = t_{i1}y_1 + t_{i2}y_2 + \dots + t_{in}y_n, \quad \forall i \in T_S, \quad (4.9)$$

$$y_1 + y_2 + \dots + y_n = 1, \quad (4.10)$$

$$y_1, y_2, \dots, y_n \in 0, 1 \quad (4.11)$$

在上述的模式中，式子(4.3)到(4.7)為採用 Herroelen and Leus(2004) 提出的 EWD1 排程模式。其中式子(4.3)為此模式的目標式，其目的在於使此排程中各活動的實際開始時間和事先排程中各活動的開始時間之預期權重差異 (expected weighted deviation) 為最小；式子(4.4)限制節點*i*和*j*的前後關係；亦即假若節點*j*緊接在節點*i*之後的話，則節點*j*的開始時間一定會大於或等於節點*i*的開始時間加上節點*i*的運行時間*d_i*；(4.5)式是用來限制最後一個虛擬節點*n*之開始時間不能超過專案的截止時間*ω*；(4.6)式為將預期產生的干擾所造成活動的延遲時間列入考慮，以決定各活動實際的開始時間。

式子(4.8)到(4.11)代表具時間限制之限制式，其中(4.8)代表具時窗限制之節點 i 的開始時間是被限制在 $begin(i)$ 和 $end(i)$ 間的時間區間內；式子(4.9)到(4.11)應用整數規劃的技巧來限制節點 i 之最早開始時間必須是所給定的 n 個時間點其中之一。

4.2 演算法建立與說明

本節中將對貪婪演算法一(Greedy I)、貪婪演算法二(Greedy II)和基因演算法(GA)之演算流程作詳細介紹，並於各演算法中以一簡單實例說明各演算法。

4.2.1 貪婪演算法一(Greedy I)

本小節將提出一貪婪演算法來求解前一小節所建構具時間限制之穩定性專案基線排程模式。回顧前述 $EWD1$ 中之目標式我們可得知只要變數 Δ_{ijk} 愈小則目標式之結果會愈小，而影響變數 Δ_{ijk} 之值則受限制式(4.6)中之各活動開始時間 s_i 和 s_j 所影響；因此，若能發展出一個方法來適當調整各活動之開始時間 s_i ，即能使變數 Δ_{ijk} 之值減小，進而能使目標式之值往最佳解逼近。根據上述想法我們設計一貪婪演算法(Greedy I)，其運算流程如圖4.1所示。

此演算法第一步驟使用拓樸排序(Topological ordering)來決定各活動節點的調整順序，此步驟所花費之時間複雜度為 $O(m)$ (其中 m 為網路圖中弧之總數)，決定好拓樸順序後即應用Chen et al.(1997)針對具時間限制之專案排程活動網路圖所提出之二階法來求各解活動的最早及最晚開始時間。決定好各活動的最早及最晚開始時間後，由於兩相連活動間的最短間隔時間 λ_{ij} 、各活動的運行時間 d_i 、和活動因為第 k 個干擾而產生的延遲時間 l_{ik} 皆為已知。另外，由於各個活動都會有 k 組干擾的劇本，其中每組不同的干擾都會造成不同的延遲時間，而若要使求得的解能往最佳解逼近的話需盡量減少 Δ_{ijk} 之值，針對此狀況我們設計取造成干擾長度最大之干擾劇本來當代入之常數(即 l_{ik} 於 k 組干擾劇本中，取最大之值來代入演算法)，因為最大的延遲時間若能滿足上述的限制式，則次大的延遲時間也必定會滿足，如此可確定求的解為可行解。接著用二階法所求出的最早開始時

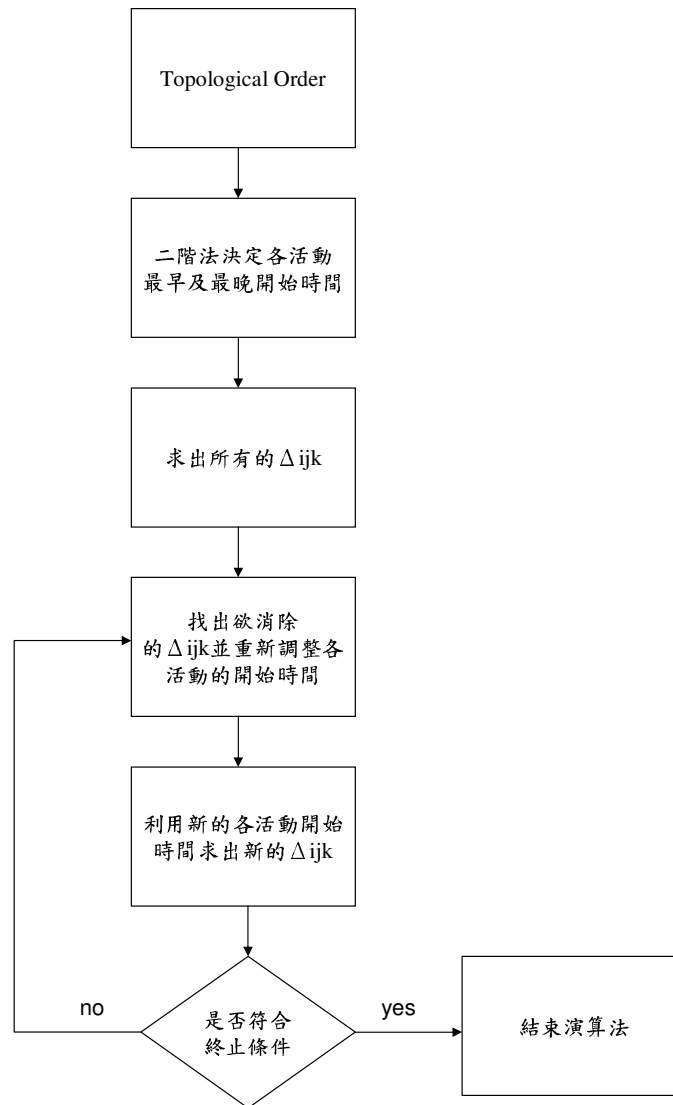


圖 4.1: Greedy I 演算法流程圖

間代入 (4.6) 式即能求出整個專案一開始時各活動之 Δ_{ijk} 最小值；另外，由 4.6 式亦可得知，如將活動 j 之開始時間往後移，則 Δ_{ijk} 之值會跟著減少，因此適當的調整 s_j 能使結果往最佳解逼近。

接下來則進入演算法之第二步驟之調整階段，此階段首先依拓撲順序來考慮活動 j 之開始時間是否能往後調整；接著再計算能調整的情況下最多能往後調整多少時間。在調整階段中，會以盡量縮減 Δ_{ijk} 之值為目的來進行調整；因為一旦活動 j 之開始時間被往後調整，則直接或間接連接於此活動後的所有活動節點之開始時間都必須跟著往後調整。因此在調整的過程中，將依序檢視這些關

聯活動節點之最晚開始時間和時間限制，以決定出 Δ_{ijk} 最大能縮減之範圍。決定的原則為以一開始求得 Δ_{ijk} 之值為最大之縮減目標值，接著逐序檢視活動 j 和其後序之活動節點之最後開始時間；在這些節點中如果某節點 l 之開始時間到其最晚開始時間之間距小於該最大之縮減目標值，則依盡量縮減之原則將最大之縮減目標值之值用節點 l 之開始時間到其最晚開始時間之間距來取代，依此原則決定出活動 j 最後能調整之值以縮減 Δ_{ijk} 之值。於此步驟中，時間調整階段依拓撲順序來決定活動能往後調整的時間，直至最後節點停止。因為此步驟依拓撲順序檢視各節點是否有能調整之時間，而在調整過程中是依 TA (transitive closure)來檢視需調整之節點，故此步驟所花費之時間複雜度可視為 $O(n * TA)$ (n 為網路圖中節點之總數)，而 $O(TA) = O(n^2)$ ，因此本步驟花費了 $O(n^3)$ 時間。

在最後步驟求出各節點 Δ_{ijk} 之值後，各自之值再與各節點之成本相乘，最後加總之值即為本演算法最後求得之解，而整合其各步驟可得知 Greedy I 所花費之時間複雜度為 $O(m) + O(n^3) = O(n^3)$ 。

以下舉一小例子實際操作說明 Greedy I 的運作流程，以圖 4.2 之隨機專案排程網路圖為例，其最佳目標函數值為 9.09。

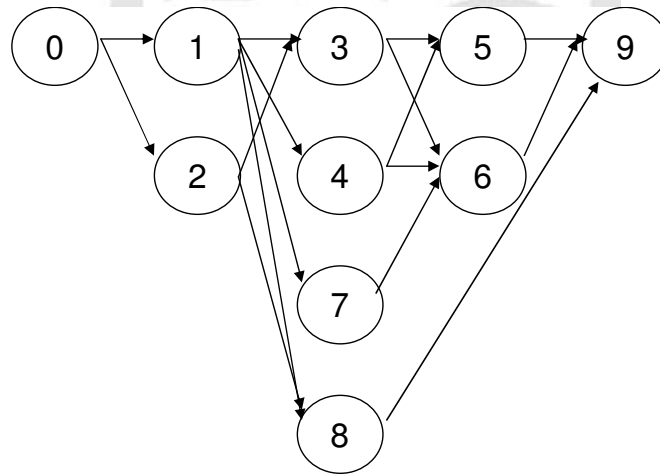


圖 4.2: 演算法實例之隨機專案排程網路圖

在此例子中，各活動運行所需之時間 d_i 為 $d_0 = 17$ 、 $d_1 = 18$ 、 $d_2 = 10$ 、 $d_3 = 1$ 、 $d_4 = 20$ 、 $d_5 = 25$ 、 $d_6 = 4$ 、 $d_7 = 9$ 、 $d_8 = 13$ 、 $d_9 = 15$ ，而透過本研究中

設計之隨機網路產生器將使以下節點隨機產生時間限制，節點1將具時間排程限制，其中該節點被限制在時間點1、21、25、27，方能開始運行；節點5和節點7為具時窗限制之活動節點，其中活動5被限制在53到71的時段間才能開始運行，節點7被限制於21到83的時段間才能開始運行。

由於各活動之運行時間限制已給定，由此能決定一常數 λ_{ij} ，其意義為活動*i*到活動*j*之最短運行時間(不包含活動*i*和活動*j*各自之運行時間)，即 $\lambda_{01}=0$ 、 $\lambda_{02}=0$ 、 $\lambda_{03}=10$ 、 $\lambda_{04}=18$ 、 $\lambda_{05}=11$ 、 $\lambda_{06}=11$ 、 $\lambda_{07}=10$ 、 $\lambda_{08}=18$ 、 $\lambda_{13}=0$ 、 $\lambda_{14}=0$ 、 $\lambda_{15}=1$ 、 $\lambda_{16}=1$ 、 $\lambda_{17}=0$ 、 $\lambda_{18}=0$ 、 $\lambda_{19}=6$ 、 $\lambda_{23}=0$ 、 $\lambda_{27}=0$ 、 $\lambda_{25}=1$ 、 $\lambda_{26}=1$ 、 $\lambda_{29}=5$ 、 $\lambda_{35}=0$ 、 $\lambda_{36}=0$ 、 $\lambda_{39}=4$ 、 $\lambda_{45}=0$ 、 $\lambda_{46}=0$ 、 $\lambda_{49}=4$ 、 $\lambda_{59}=0$ 、 $\lambda_{69}=0$ 、 $\lambda_{79}=0$ 、 $\lambda_{89}=0$ 。

另外，各活動有*k*組干擾劇本，各干擾有其發生之機率與造成之時間延遲。於本例中，活動0有1干擾 $l_{01}=5$ ；活動1有2個干擾劇本， $l_{11}=7$ 其發生機率為0.67， $l_{12}=2$ 其發生機率為0.33；活動2有4個干擾劇本， $l_{21}=18$ 其發生機率為0.4， $l_{22}=13$ 其發生機率為0.3， $l_{23}=8$ 其發生機率為0.2， $l_{24}=3$ 其發生機率為0.1；活動3有3個干擾劇本， $l_{31}=12$ 其發生機率為0.5， $l_{32}=7$ 其發生機率為0.33， $l_{33}=2$ 其發生機率為0.17；活動4有3個干擾劇本， $l_{41}=13$ 其發生機率為0.5， $l_{42}=8$ 其發生機率為0.33， $l_{43}=3$ 其發生機率為0.17；活動5有2個干擾劇本， $l_{51}=10$ 其發生機率為0.67， $l_{52}=5$ 其發生機率為0.33；活動6有5個干擾劇本， $l_{61}=21$ 其發生機率為0.33， $l_{62}=16$ 其發生機率為0.27， $l_{63}=11$ 其發生機率為0.2， $l_{64}=6$ 其發生機率為0.13， $l_{65}=1$ 其發生機率為0.07；活動7有2個干擾劇本， $l_{71}=7$ 其發生機率為0.67， $l_{72}=2$ 其發生機率為0.33；活動8有4個干擾劇本， $l_{81}=17$ 其發生機率為0.4， $l_{82}=12$ 其發生機率為0.3， $l_{83}=7$ 其發生機率為0.2， $l_{84}=2$ 其發生機率為0.1；活動9有2個干擾劇本， $l_{91}=7$ 其發生機率為0.67， $l_{92}=2$ 其發生機率為0.33。

我們將此專案排程之截止時間設定為最後節點(節點9)之最早開始時間加上其運行時間之和再乘上1.15，設定好截止時間後即可應用Chen et al.(1997)提出之二階法來求解各活動*i*的最早開始時間 ESS_i 及最晚開始時間 LSS_i ， $ESS_0=0$ 、 $ESS_1=21$ 、 $ESS_2=17$ 、 $ESS_3=39$ 、 $ESS_4=39$ 、 $ESS_5=59$ 、 $ESS_6=59$ 、

$ESS_7 = 39$ 、 $ESS_8 = 39$ 、 $ESS_9 = 84$ ；最晚開始時間依序為 $LSS_0 = 8$ 、 $LSS_1 = 26$ 、 $LSS_2 = 26$ 、 $LSS_3 = 46$ 、 $LSS_4 = 46$ 、 $LSS_5 = 71$ 、 $LSS_6 = 100.85$ 、 $LSS_7 = 83$ 、 $LSS_8 = 100.85$ 、 $LSS_9 = 113.85$ 。

在求得各活動之最早開始時間後，我們以各活動最早開始時間為各活動之第一組開始時間，即可由 (4.6) 式 $\Delta_{ijk} + s_j - s_i \geq (l_{ik} + d_i + \lambda_{ij})$ 來決定各活動之 Δ_{ijk} (其中 l_{ik} 、 d_i 、 λ_i 為已知常數)。接著演算法即進入時間調整階段，依拓撲順序逐序檢視活動 j 能調整之時間。本例中，一開始決定欲調整之 Δ_{ijk} 值為 Δ_{02k} ，其中活動 0 之開始時間為 0、活動 2 之開始時間為 17、 $\lambda_{02} = 0$ 、 $d_0 = 17$ 、 $l_{01} = 5$ ，能求出 $\Delta_{02k} = 5$ ；求出 Δ_{02k} 之值後，將該值指定為 $MaxReduce$ ，演算法即進入時間調整階段，活動 2 之最晚開始時間 $LSS_2 = 26$ ，而此時活動 2 之開始時間 $s_2 = 17$ ，表示能向後平移 5 單位時間以上，則 $MaxReduce = 5$ ；接著依序檢視活動 3、7、5、6、9 之最晚開始時間，可知這些活動皆可向後平移 5 單位時間以上，而若將這些節點往後平移 5 單位時間將使 $\Delta_{02k} = 0$ ，因此可做如此之調整，即 $s_2 = 22$ 、 $s_3 = 44$ 、 $s_7 = 44$ 、 $s_5 = 64$ 、 $s_6 = 64$ 、 $s_9 = 89$ 。

檢視完 Δ_{02k} 之後，演算法將依拓撲順序檢視 Δ_{01k} 之節點，其中活動 0 之開始時間為 0、活動 1 之開始時間為 21、 $\lambda_{01} = 0$ 、 $d_0 = 17$ 、 $l_{01} = 5$ ，能求出 $\Delta_{01k} = 1$ ；求出 Δ_{01k} 之值後，將該值指定為 $MaxReduce = 1$ ，接著依序檢視活動 1、3、4、7、8、5、6、9 之最晚開始時間，可知這些活動皆可向後平移 1 單位時間以上，因此該將這些活動都往後平移 1 單位時間以使 $\Delta_{01k} = 0$ 。然而，活動 1 被限制於時間點 16、21、25、27 才能開始執行，所以活動 1 必須往後延 4 單位時間即 $s_1 = 25$ 才合理；接著重新檢視活動 3、4、7、8、5、6、9 之最晚開始時間，可發現活動 3 無法往後調整 4 單位時間，因此 $MaxReduce$ 則將因無法進行調整而被設定為 0，導致此階段沒有任何活動之開始時間被調整。

同理，演算法持續依拓撲順序檢視 Δ_{03k} 、 Δ_{07k} 、 Δ_{04k} 、 Δ_{08k} 、 Δ_{05k} 、 Δ_{06k} 、 Δ_{09k} 、 Δ_{13k} 、 Δ_{14k} 、 Δ_{17k} 、 Δ_{18k} 、 Δ_{15k} 、 Δ_{16k} 、 Δ_{19k} 、 Δ_{27k} 、 Δ_{25k} 、 Δ_{26k} 、 Δ_{29k} 、 Δ_{36k} 、 Δ_{39k} 、 Δ_{46k} 、 Δ_{49k} 、 Δ_{59k} 、 Δ_{69k} 、 Δ_{79k} 、 Δ_{89k} ，並進行時間調整 (同時考慮時間限制) 以縮減各 Δ_{ijk} 之值，最後可得知 $\Delta_{01k} = 1$ 、 $\Delta_{14k} = 2$ 、 $\Delta_{23k} = 4$ 、

$\Delta_{45k} = 1.15$ ，其他 Δ_{ijk} 之值皆為 0，再依各自所屬之成本去做乘積與加總即可求得本演算法最後之結果，其中 $c_1 = 2$ 、 $c_4 = 3$ 、 $c_3 = 1$ 、 $c_5 = 1$ ，由此可得知所求之解為 $1 * 2 + 2 * 3 + 4 * 1 + 1.15 * 1 = 13.15$ 。

4.2.2 貪婪演算法二 (Greedy II)

在演算法 Greedy II 中，我們針對各活動延遲所產生的單位時間成本和時間調整階段兩方面來做改進。

當各活動產生延遲時會有其各自的時間成本產生，而在模式的目標式中此成本是被列入計算的；因此，演算法 Greedy II 首先會依各活動延遲會造成的時間成本的多寡，事先給定該活動一段時間，而此時間的安排方式是依 Tavares et al.(1998) 所提出之彈性係數方式來安排。演算法 Greedy II 設定此彈性係數的方式是依各活動延遲會造成的時間成本所佔總成本之比例來給定，而 Tavares et al.(1998) 在該研究中指出該方式所排程之結果必具可行解。

以時間調整階段方面而言，演算法 Greedy I 中，其調整時間之檢視過程為檢視欲調整之活動節點開始時間以及其所有直接或間接連接至此活動節點之節點的開始時間，然而如此的檢視方式過於保守，將使演算法解之品質不佳；在仔細觀察決定各活動開始時間之前序節點之後，我們發現某些節點之開始時間往後移動並不會直接影響到連接其後之節點。以圖 4.3 為例，活動 3 之開始時間是由活動 1 所決定，而活動 2 之開始時間若改變，並不會直接影響到活動 3 之開始時間，甚至活動 4、7、8 亦皆不會受到影響；但演算法 Greedy I 中，一但活動 2 之開始時間需要調整時，則仍會檢視這些活動節點來決定活動 2 是否能進行調整，這將使解的品質不佳；因此，演算法 Greedy II 將針對此點加以改進，即在 Δ_{ijk} 之值不為 0 的節點上，重新去檢視該節點之調整空間。

演算法 Greedy II 對於欲調整開始時間之節點，僅須檢視直接受其影響的活動節點，而其能調整之時間長短也直接由那些直接受其影響之節點來計算即可。圖 4.3 中以箭頭相連結之兩節點間存在有直接的影響關係；因此若欲調整活動 2 之開始時間，因該活動之後並無直接受其影響之節點，所以可逕行將其開始時間調整至其最晚開始時間以減少 Δ_{ijk} ，而不用似 Greedy I 一般還需考慮原來連接於活

動2之後之活動節點；反之，若欲調整活動4之開始時間，則需檢視活動5、6、9，來決定活動4最多能調整多少時間，而在調整完活動之後，只需調整其後直接受其影響到之活動5及活動6之開始時間即可。依此原則逐一檢視 Δ_{ijk} 不為0之活動節點，待檢視完畢後，再依剩下之 Δ_{ijk} 值乘上其對應之成本，最後再加總即可。

在分析演算法 Greedy II 之時間複雜度而方面，在重新設定檢視調整時間之原則後，依 Δ_{ijk} 之值不為0之節點重新檢視調整時間之步驟其時間複雜度為 $O(n * m) + O(n * m) = O(nm)$ ；而其前序步驟中決定各活動 Δ_{ijk} 之值的步驟為演算法 Greedy I 之步驟，故此步驟之時間複雜度為 $O(n^3)$ ；因此演算法 Greedy II 之時間複雜度為 $O(nm) + O(n^3) = O(n^3)$ 。

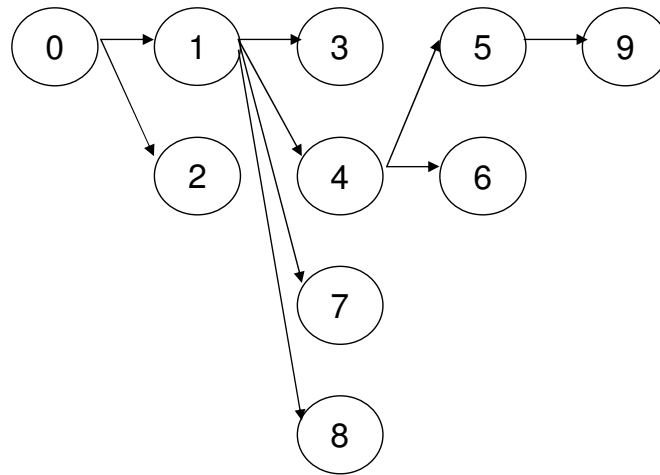


圖 4.3: 決定各活動最早開始時間之先序關係圖

以下用圖4.2之例說明演算法 Greedy II 之運作流程，於演算法 Greedy I 之最後步驟可決定出 Δ_{ijk} 不為0之值有 $\Delta_{01k} = 1$ 、 $\Delta_{14k} = 2$ 、 $\Delta_{23k} = 4$ 、 $\Delta_{45k} = 1.15$ ，演算法 Greedy II 將依這些 Δ_{ijk} 之值逐序檢視這些值是否能有縮減之空間。首先，對於 $\Delta_{01k} = 1$ 將逐一檢視活動1、4、5、6、9來決定 s_1 是否有調整之空間，但由於活動5和活動9已無調整之空間，因此 s_1 無法縮減；接著檢視 $\Delta_{14k} = 2$ ，為了檢視 s_4 是否有調整之空間必須依序檢視活動4、5、6、9；然而活動5和活動9已無調整之空間，因此此 s_4 值仍無法縮減；接著針對 $\Delta_{23k} = 4$ 檢視 s_3 是否有調整之

空間；由於圖 4.3 中活動 3 無直接受其影響之節點，因此只需檢視 s_3 本身是否能調整即可；由於 s_3 已無法調整，所以其值仍不變；最後針對 $\Delta_{45k} = 1.15$ 檢視 s_5 是否有調整空間，則依序檢視活動 5、活動 9；但由 s_5 和 s_9 已無調整之空間，所以其值仍不變。綜合上述結果可知 $\Delta_{01k} = 1$ 、 $\Delta_{14k} = 2$ 、 $\Delta_{23k} = 4$ 、 $\Delta_{45k} = 1.15$ ，其他 Δ_{ijk} 之值皆為 0，再依各自所屬之成本去乘積最後加總即可求得本演算法最後之結果。因為 $c_1 = 2$ 、 $c_4 = 3$ 、 $c_3 = 1$ 、 $c_5 = 1$ ，由此可得知此法之目標函數值為 $1 * 2 + 2 * 3 + 4 * 1 + 1.15 * 1 = 13.15$ 。

4.2.3 基因演算法 (GA)

運用基因演算法來求解最佳化的問題時，其演算法流程為：(1) 初始化階段 (Initialization)，將要搜尋的可能解編碼成為染色體 (chromosome)；(2) 求解階段 (Evaluation)，依據求解之目標及條件設計適應的函數 (fitness function)；(3) 選擇階段 (Selection)，依求解之適應函數值的大小將具有較佳適應函數值的解將被挑選至交配池 (matching pool) 中；(4) 重組階段 (Recombination)，接著再依據交配運算子與突變運算子的運算過程後，即完成一世代的基因演算法則，當完成一世代的演算法則後，即判斷是否達搜尋求解目標，如已達到終止條件則終止運算求取最佳解，如尚未達到終止條件則重覆演算流程步驟至求取最佳解為止。

以下將依上述基因演算法運算流程，逐步說明如何運用此法以求解本研究所提出之模式：

1. 初始化階段 (Initialization)：利用基因演算法求解問題前，必須先將問題的變數進行編碼，於本基因演算法中應用二元 (Binary) 編碼來進行此階段。二元編碼係指以二進位 (0,1) 方式表達染色體，多用於數值型問題。於本模式中，問題之變數為各活動之開始運行時間 s_i ，因此此階段將針對各活動之開始運行時間進行二元編碼。
2. 求解階段 (Evaluation)：經由初始化階段後將獲得初始群體 (Initial population) 係指第一代的染色體群體，代表問題的初始解。於本模式中，將透過

第一代之各活動開始時間求出各活動之 Δ_{ijk} 之值，並可得其對應之目標函

$$\text{數值} \sum_{(i,j) \in TA} \sum_{k \in \Psi_i} c_j p_i g_i(l_{ik}) \Delta_{ijk}。$$

3. 選擇階段 (Selection)：複製運算乃是自然界中「適者生存」的概念，旨在選擇族群中優質的染色體，使其得以保留並繁衍優良的後代；本研究採輪盤式選擇 (roulette wheel selection)。其中以適應函數來衡量每組染色體優劣的標準，當適應值愈大，表示生存能力佳，容易被保留下來繁衍後代；反之，則容易被環境淘汰。求解最大化問題時，可直接將問題的目標值定義為適應函數；但本研究模式所探討的最小化問題，則需將目標函數值轉換成最大化的適應函數值，因此本研究之轉換方式為取目標函數值的倒數。
4. 重組階段 (Recombination)：此步驟中最重要的兩項機制為交配 (crossover) 和突變 (mutation)，交配係隨機選取交配群中兩個父代染色體，彼此交換位元資訊，以改變基因組合，期望產生適應性更高的染色體突變是為了避免在演算過程中落入局部最佳解，它雖然會破壞演化過程中染色體的穩定性，但卻能夠增加母體內的變異程度，以擴大問題的搜尋空間，進一步逼近全域最佳解；另一方面，突變之運作機制是由一突變率 (Mutation rate) 所控制，一般而言在自然界中，基因突變率很低，約為百萬分之一，但在求解時為增加變異多會提高此一機率。
5. 終止條件 (Termination)：在演化過程中，必須事先設定終止搜尋的條件，當符合終止條件時，即結束演算。本研究設定 GA 之終止條件為，當設定進行演化之基因 (chromosome) 群組個數求出之解收斂到一定程度即終止；也就是說，當 GA 的求解因子收斂到一定程度，即認為該演算法已求得其最適解。

4.3 小結

本節中發展三種演算法，首先依研究模式限制式特性和網路圖形之特性發展出貪婪演算法一 (Greedy I)，接著針對在貪婪演算法一中考慮過於保守之部分

加以修改以發展出貪婪演算法二(Greedy II)，最後依基因演算法之演算流程去設計一符合本研究模式之基因演算法。在下章節中，將依第三章設計之具時間限制之隨機專案排程網路產生器來產生各類網路例子，並以上述三種演算法來求解之，以進行數值分析與討論。



第五章

數值分析與討論

本章擬以第三章所發展之網路產生器隨機產生幾群具時間限制之專案排程網路來測試並比較最佳化解法 CPLEX 以及第四章所提出之演算法 Greedy I、Greedy II 和 GA 之求解品質和效率。

5.1 實驗環境與參數設定

本研究進行此數值分析之環境如下：

1. 系統：Microsoft Windows XP Professional Version 2002 Service Pack2。
2. 電腦：AMD Athlon(tn) 64 Processor 3000+, 2.01 GHz, 1.50GB 的 RAM。
3. 最佳化解法軟體：ILOG CPLEX 9.0。
4. 程式設計工具：Microsoft Visual C++ 6.0。

在用來進行求解之網路圖例的設定方面，為了比較出各演算法之求解效率和求解品質，本研究使用第三章發展出之具時間排程的隨機專案排程網路產生器來產生數值分析所需之網路圖。本實驗設計將時間限制(TC)比例設定為0.6和0.9，即所產生的網路圖節點中有六成或九成之活動節點將具時間限制；其中具時窗限制之活動節點比率為0.2，具時間排程限制之比率為0.8，即在具時間限制之活動節點中有兩成會具有時窗限制，有八成會具有時間排程限制。在RanGen參數方面，將產生活動數(n)為 $n = 60$ 、 $n = 120$ 、 $n = 180$ 、 $n = 240$ 和 $OS = 0.3$ 、

$OS = 0.6$ 、 $OS = 0.9$ 之網路圖各10個。由上述設定本實驗將產生出240組網路圖來進行數值分析。

模式中各常數之設定將參考 Herroelen and Leus(2004) 實驗設計之設定；各活動之運行時間 d_i 將於 $[1, 25]$ 之區間內隨機產生；各活動之成本 c_i 將設定於 $[1, 3]$ 之區間內隨機產生；各專案網路圖之截止時間 (ω) 為 $\omega = (1 + \omega_0)S_n$ ，設定為其最後節點之最早開始時間之值加上該節點之運行時間後再乘上 $1 + \omega_0$ ；其中設定 $\omega_0 = 0.15$ 。

GA 參數方面之設定，需事先設定其突變率、交配率和其終止條件。本研究採用突變率 0.1、交配率 0.25 之參數來運行 GA 求解本研究之專案模式；而 GA 的終止條件則被設為當負責產生解的基因群收斂到一定程度之後即停止運行。

5.2 數值分析與討論

在測試例子方面，我們利用 $TC = 0.6$ 、 $TC = 0.9$ 、 $OS = 0.3$ ； $OS = 0.3$ 、 $OS = 0.6$ 、 $OS = 0.9$ 這些參數設定產生四組節點數為 $n = 60$ 、 $n = 120$ 、 $n = 180$ 、 $n = 240$ 之專案排程網路圖。其中，我們針對每組設定皆隨機產生 10 個測試例子，而其平均表現皆記錄於表 5.1 至表 5.8 中。

表 5.1 至表 5.4 中記錄各演算法之效能以 Optimality Gap 之方式呈現出來，Optimality Gap 代表演算法求得之目標函數值與最佳解之誤差百分比；其中，Optimality Gap 愈小之演算法其效能愈好。

由表 5.1 中可觀察出 Greedy I 除了在 $n = 60$ 、 $OS = 0.6$ 、 $TC = 0.9$ 和 $n = 120$ 、 $OS = 0.6$ 、 $TC = 0.9$ 兩種情況其表現優於 Greedy II 之外，其解的品質大都較 Greedy II 之求解品質為差；當固定 $TC = 0.6$ ， OS 依 0.3、0.6 和 0.9 之設定時，Greedy I 與 CPLEX 之 Optimality Gap 值分別為 0.1492、0.2025 和 0.2467，可發現 Greedy I 之求解品質隨弧之密度愈高而差；在同樣之參數設定下，Greedy II 與 CPLEX 之 Optimality Gap 值分別為 0.1275、0.1874 和 0.2044；因此，在 OS 之設定提高時，Greedy II 之求解品質也是逐漸變差。另外，依同樣方式觀察表 5.2 至表 5.4

亦可發現在這些表中，Greedy I和Greedy II之求解品質皆因 OS 之值的提高而變差。

當設定 $TC = 0.6$ 、 $OS = 0.3$ ，觀察表 5.1 至表 5.4 中 Greedy I 與 CPLEX 之 Optimality Gap 值分別為 0.1492、0.1705、0.1835、0.2253，可發現當活動數 n 逐漸增加時，Greedy I 之求解品質逐漸變差；依同樣參數之設定，以 Greedy II 與 CPLEX 之 Optimality Gap 值分別為 0.1275、0.1290、0.1465、0.1677，亦可發現當活動數 n 逐漸增加時，Greedy II 之求解品質逐漸變差。同理當固定 TC 和 OS 值，而分別觀察表 5.1 至表 5.4 中之結果，亦可得到同上之結論。

在 GA 的求解品質方面，我們發現在各參數的設定下其解和 CPLEX 之 Optimality Gap 在 0.04 到 0.08 之間；由此可觀察出 GA 求解之穩定性，且其求解品質皆比 Greedy I 和 Greedy II 好上許多。

表 5.1: Optimality Gap ($n = 60$)

| | $OS = 0.3$ | $OS = 0.6$ | $OS = 0.9$ |
|-------------------------|------------|------------|------------|
| Greedy I ($TC = 0.6$) | 0.1492 | 0.2025 | 0.2467 |
| Greedy II($TC = 0.6$) | 0.1275 | 0.1874 | 0.2044 |
| GA ($TC = 0.6$) | 0.0535 | 0.0444 | 0.0631 |
| Greedy I ($TC = 0.9$) | 0.1375 | 0.2298 | 0.2721 |
| Greedy II($TC = 0.9$) | 0.1190 | 0.2325 | 0.2551 |
| GA ($TC = 0.9$) | 0.0418 | 0.0783 | 0.0529 |

表 5.2: Optimality Gap ($n = 120$)

| | $OS = 0.3$ | $OS = 0.6$ | $OS = 0.9$ |
|-------------------------|------------|------------|------------|
| Greedy I ($TC = 0.6$) | 0.1705 | 0.2845 | 0.2711 |
| Greedy II($TC = 0.6$) | 0.1290 | 0.2106 | 0.1901 |
| GA ($TC = 0.6$) | 0.0758 | 0.0561 | 0.0657 |
| Greedy I ($TC = 0.9$) | 0.1457 | 0.195 | 0.2893 |
| Greedy II($TC = 0.9$) | 0.1313 | 0.2019 | 0.2218 |
| GA ($TC = 0.6$) | 0.0652 | 0.0415 | 0.0523 |

表 5.5 至表 5.8 記錄各演算法所花費之時間，代表其效率表現。在表 5.5 中，當固定 $TC = 0.6$ (或 $TC = 0.9$)， OS 依 0.3、0.6、0.9 之設定時，可發現 CPLEX 所

表 5.3: Optimality Gap ($n = 180$)

| | $OS = 0.3$ | $OS = 0.6$ | $OS = 0.9$ |
|-------------------------|------------|------------|------------|
| Greedy I ($TC = 0.6$) | 0.1835 | 0.3013 | 0.3164 |
| Greedy II($TC = 0.6$) | 0.1465 | 0.2019 | 0.2751 |
| GA ($TC = 0.6$) | 0.0747 | 0.0459 | 0.0644 |
| Greedy I ($TC = 0.9$) | 0.1775 | 0.2785 | 0.3303 |
| Greedy II($TC = 0.9$) | 0.1532 | 0.1911 | 0.2451 |
| GA ($TC = 0.6$) | 0.0823 | 0.0539 | 0.0572 |

表 5.4: Optimality Gap ($n = 240$)

| | $OS = 0.3$ | $OS = 0.6$ | $OS = 0.9$ |
|-------------------------|------------|------------|------------|
| Greedy I ($TC = 0.6$) | 0.2253 | 0.3332 | 0.3642 |
| Greedy II($TC = 0.6$) | 0.1172 | 0.2248 | 0.2449 |
| GA ($TC = 0.6$) | 0.0675 | 0.0467 | 0.0785 |
| Greedy I ($TC = 0.9$) | 0.212 | 0.2495 | 0.3083 |
| Greedy II($TC = 0.9$) | 0.1677 | 0.2406 | 0.2574 |
| GA ($TC = 0.6$) | 0.0542 | 0.0674 | 0.0751 |

花費之時間分別為 18.1、28.2、30.5，亦即 CPLEX 之求解時間會隨 OS 值之增加而增加。Greedy I 和 Greedy II 所花費之時間幾乎相同，同時也是四種演算法中求解時間最短的。而 GA 所花費之時間大都在 9 到 13 秒左右，其變異不大。

當設定 $TC = 0.6$ 、 $OS = 0.3$ ，觀察表 5.5 至表 5.8 中 CPLEX 所花費之時間分別為 18.1、109.9、430.9、1011.8，可發現當活動數 n 逐漸增加時，CPLEX 所花費之時間明顯的增加；依同樣參數之設定而言，以 Greedy I 和 Greedy II 所花費之時間幾乎相同，亦可發現當活動數 n 逐漸增加時，Greedy I 和 Greedy II 所花費之時間也會增加；但在一定的活動數中，其效率為最快的；而基因演算法 GA 所花費之時間皆在 9 到 14 秒左右，其變異並沒有隨活動數增加而改變。

5.3 小結

以下依求解品質和求解效率來整合數值分析所得到之結果。

以求解品質而言(表 5.1 至表 5.4)：

表 5.5: Computational Time ($n = 60$)

| | $OS = 0.3$ | $OS = 0.6$ | $OS = 0.9$ |
|-------------------------|------------|------------|------------|
| CPLEX ($TC = 0.6$) | 18.1 | 28.2 | 30.5 |
| Greedy I ($TC = 0.6$) | 0.2 | 0.1 | 0.1 |
| Greedy II($TC = 0.6$) | 0.1 | 0.2 | 0.1 |
| GA ($TC = 0.6$) | 10.1 | 11.85 | 9.3 |
| CPLEX ($TC = 0.9$) | 33.6 | 34.2 | 36.9 |
| Greedy I ($TC = 0.9$) | 0.1 | 0.1 | 0.2 |
| Greedy II($TC = 0.9$) | 0.2 | 0.1 | 0.2 |
| GA ($TC = 0.9$) | 12.6 | 10.8 | 11.9 |

表 5.6: Computational Time ($n = 120$)

| | $OS = 0.3$ | $OS = 0.6$ | $OS = 0.9$ |
|-------------------------|------------|------------|------------|
| CPLEX ($TC = 0.6$) | 109.9 | 129.1 | 142.4 |
| Greedy I ($TC = 0.6$) | 2.3 | 2.1 | 2.3 |
| Greedy II($TC = 0.6$) | 2.2 | 2.2 | 2.1 |
| GA ($TC = 0.6$) | 12.1 | 13.4 | 11.7 |
| CPLEX ($TC = 0.9$) | 158.9 | 168.5 | 220.1 |
| Greedy I ($TC = 0.9$) | 2.1 | 2.4 | 2.2 |
| Greedy II($TC = 0.9$) | 2.0 | 2.2 | 2.1 |
| GA ($TC = 0.9$) | 13.7 | 10.1 | 9.7 |

表 5.7: Computational Time ($n = 180$)

| | $OS = 0.3$ | $OS = 0.6$ | $OS = 0.9$ |
|-------------------------|------------|------------|------------|
| CPLEX ($TC = 0.6$) | 430.9 | 510.0 | 599.5 |
| Greedy I ($TC = 0.6$) | 6.1 | 6.5 | 6.2 |
| Greedy II($TC = 0.6$) | 6.2 | 6.1 | 6.6 |
| GA ($TC = 0.6$) | 12.7 | 11.1 | 13.2 |
| CPLEX ($TC = 0.9$) | 528.0 | 655.2 | 769.6 |
| Greedy I ($TC = 0.9$) | 6.4 | 6.5 | 6.1 |
| Greedy II($TC = 0.9$) | 6.5 | 6.1 | 6.4 |
| GA ($TC = 0.9$) | 13.9 | 10.1 | 12.6 |

1. 當固定活動數 n 、 OS 和 TC 三參數，可發現 Greedy II 之求解品質大都較 Greedy I 為佳。
2. 由 OS 之值來觀察各演算法解之品質可發現，當 OS 值增加時，Greedy I 和 Greedy II 之求解品質會明顯變差。

表 5.8: Computational Time ($n = 240$)

| | $OS = 0.3$ | $OS = 0.6$ | $OS = 0.9$ |
|--------------------------|------------|------------|------------|
| CPLEX ($TC = 0.6$) | 1011.8 | 1021.2 | 1059.1 |
| Greedy I ($TC = 0.6$) | 14.2 | 13.6 | 14.3 |
| Greedy II ($TC = 0.6$) | 13.8 | 13.7 | 13.1 |
| GA ($TC = 0.6$) | 11.9 | 13.2 | 11.6 |
| CPLEX ($TC = 0.9$) | 1134.7 | 1163.4 | 1188.7 |
| Greedy I ($TC = 0.9$) | 14.1 | 16.5 | 15.5 |
| Greedy II ($TC = 0.9$) | 14.6 | 14.1 | 15.1 |
| GA ($TC = 0.9$) | 9.7 | 12.1 | 11.8 |

3. 以專案活動數 n 來觀察各演算法之求解品質，當 n 增加時，Greedy I 和 Greedy II 之求解品質會明顯降低。
4. 當活動數超過一定程度時，Greedy I 和 Greedy II 的求解效率甚至會比 GA 差。

綜合上述幾點之結論，可發現影響 Greedy I 和 Greedy II 求解品質最大的參數為 OS ，亦即當此兩演算法用於求解弧密度高之網路圖形時其結果會不佳。

以求解效率而言（表 5.5 至表 5.8）：

1. 觀察 CPLEX 所花費之時間，可發現當活動 n 增加時，其花費之時間亦會隨之增加；其中當 TC 增加時，可觀察其花費之時間亦跟著明顯增加，由此可知時間限制 TC 與活動節點數 n 對 CPLEX 之效率有顯著的影響。
2. 以求解效率而言，Greedy I 和 Greedy II 在一定的活動數內，其求解效率為四種方法中最佳的；但其效能卻為最差。
3. GA 方面之時間，不論其參數設定為何，其求解之品質皆維持在一定水準之上；另外，其所花費之時間皆在 9-14 秒之間，由此可觀察出 GA 求解具有穩定性且效率也不錯。

綜合上述幾點，可得知 CPLEX 求解所花費之時間最多、而 Greedy I 和 Greedy II 之所花費之時間在一定的活動參數設定下，為四種方法中最少；另外，GA 在適當設定其交配率、突變率後，可在較短時間內求得品質較佳之解。

第六章

結論與未來發展方向

6.1 結論

本研究的主要目的在於發展一套具時間限制之不確定性排程模式，使以往之不確定性模式能加入時間限制之考量，以深入專案排程問題之應用。其中，時間限制主要參考Chen et al.(1997)所提出之時窗和時間排程限制；而穩定性專案基線排程之主要模式依據則以Herroelen and Leus(2004)所提出之*EW D1*模式為基礎。

因應數值分析之需求，本研究於第三章中探討關於專案排程網路產生器之相關文獻，並應用*RanGen*來設計一能產生時間限制之隨機專案排程網路產生器，以供本研究模式探討之應用。

在第四章中，我們考量時間限制與不確定性排程，發展出本研究欲探討之模式，即在不確定性專案排程模式中加入時窗與時間排程限制。隨後針對此模式之網路特性發展兩套貪婪演算法來求解之，最後並發展基因演算法來求解此模式，以提供各方法求解品質與求解效率之比較。

第五章乃應用第三章所發展之網路產生器產生具時間限制之隨機專案排程網路來進行數值分析，分別使用CPLEX、Greedy I、Greedy II和GA四種方法來比較其求解品質和求解效率。在求解品質方面，Greedy II之解優於Greedy I，但此兩法之求解品質相對亦較差；而基因演算法在適當調整其交配率、突變率並設定其終止條件後大都能有效逼近最佳解；在求解效率方面，Greedy I和Greedy II之求解效率在一定活動數的設定下為最快；其中GA其求解效率皆維持在一定時間以內；而CPLEX經過比較後可發現其效率在各演算法中為最差。

本研究之具體貢獻整理如下：

1. 設計一能產生具時間限制之隨機專案排程網路產生器(第三章)。
2. 建構一具時間限制之穩定性專案基線排程模式(第四章)。
3. 發展兩套貪婪演算法(Greedy I, Greedy II)和基因演算法(GA)來求解具時間限制之穩定性專案基線排程模式(第四章)。
4. 分析比較CPLEX、Greedy I、Greedy II和GA四種解法對於穩定性專案基線排程模式之求解品質和求解效率(第五章)。

6.2 未來研究方向

針對本研究提出之具時間限制之穩定性專案基線排程研究以下提出幾個方向可供未來探討與分析：

1. 演算法之設計方向：本論文所中所發展之兩套貪婪式演算法皆為著重於如何直接於網路圖形中調整各活動之開始時間，以圖縮減 Δ_{ijk} 之值，進而使求得之解往最佳解方向逼近；在進行調整之原則方面，可能考慮太過保守以至於各活動之開始時間無法彈性調整；因此，是否有其他更好或更有彈性之原則應是一個值得思考的研究方向。

另外，Herroelen and Leus(2004)所提出之 *EWD1* 模式亦已證明其對偶(dual)為一最小成本網路流問題(MCNFP)，因此若能針對時間限制之限制式發展其對偶式，並於網路圖形上深入思考其對偶式所代表之意義與網路特性，或可發展出更有效率及效能之網路相關解法。

2. 基因演算法的改良建議：基因演算法首先隨機求出一組起始解，依此起始解進行淘汰的機制以使所求之解能往最佳解逼近；因此，若一開始能給定較佳的起始解，則其收斂速度和求解品質應能更佳。Greedy I和Greedy II能在短時間內求出一可行解，雖其解的品質不佳，但若將之設定為GA之起始解，是否能使GA求得之解能更進一步改善，甚至縮短GA的運行時間，此亦為未來一可行之研究方向。

3. 考慮不同之活動節點特性：傳統之網路問題探討皆以 *AND* 節點來探討其意義，具 *AND* 特性節點必須等到其先序活動節點全部執行完才能開始運行；然而有學者指出網路節點尚需具備 *OR* 特性方能滿足現實專案排程問題之需求，而具 *OR* 特性之節點只要有其某一先序活動節點完成，則該節點的活動就能開始運行。

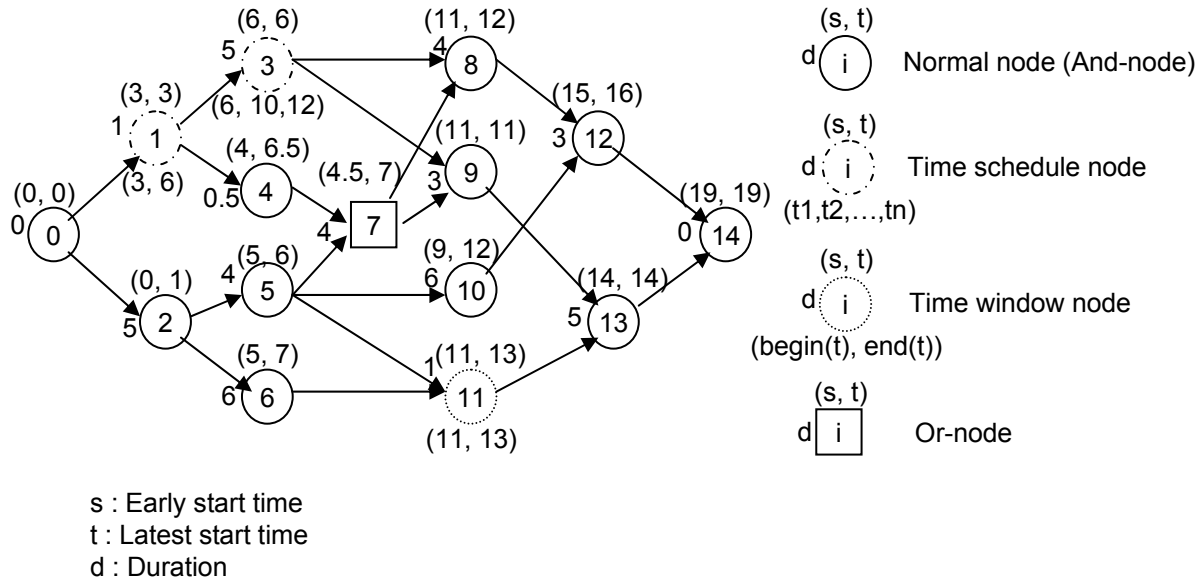


圖 6.1: 應用 *OR* 節點於具時間限制之專案網路範例

以下舉一應用 *OR* 節點可能發生之問題為例。圖 6.1 中，若將節點 7 改成 *OR* 節點，其結果將會發現依照 Chen et al.(1997) 的演算法可得節點 5 的最晚開始時間應該為 3，但是該節點的最早開始時間為 5，此與原本預期之結果互相矛盾，因此該演算法必須加以修正才能適用於具 *OR* 節點的專案網路圖。

此外，在過去純粹只有 *AND* 節點的情況下，若假設有某一活動 i 為活動 j 的前序活動，令 s_i 為活動 i 的開始時間， s_j 為活動 j 的開始時間， d_i 為活動 i 的運行時間，則 $s_i + d_i \leq s_j$ 為設計各種模型的基本限制式；但若考慮具 *OR* 節點的網路圖時，圖 6.1 中的節點 5 之開始時間已大於節點 7 的開始時間，故若依以往考慮 *AND* 節點之方式來處理具 *OR* 節點的網路圖上會出現矛盾。

由上所提出之例子可知，過去依 *AND* 節點所設計之演算法，代入具 *OR* 節點之網路圖，其結果不一定適用。由於具 *OR* 節點之網路圖的卻存在於於現

實應用中，因此未來亦可朝此方向研究如何設計可實現包含 *OR* 節點之穩定性專案基線排程演算法。



參考文獻

- Adelson-Velsky, G. M. and Leveney, E. Project scheduling in and-or graphs: A generalization of dijkstra's algorithm. *Mathematics of Operations Research*, **27**(3), 504-517, 2002.
- Agrawal, M. K., Elmaghraby, S. E. and Herroelen, W. Dagen: a generator of testsets for project activity nets. *European Journal of Operational Research*, **90**, 376-382, 1996.
- Bein, W. W., Kaburowski, J. and Stallmann, M. F. M. Optimal reduction of two-terminal directed acyclic graphs. *Society for Industrial and Applied Mathematics Journal on Computing*, **21**, 1112-1129, 1992.
- Bey, R. B., Doersch, R. and H., P. J. The net present value criterion: its impact on project scheduling. *Project Management Quarterly*, **12**(2), 35-45, 1981.
- Bowers, J. A. Criticality in resource constrained networks. *Journal of the Operational Research Society*, **46**, 80-91, 1995.
- Calhoun, K. M., Deckro, R. F., Moore, J. T., Chrissis, J. W. and Van Hove, J. C. Planning and re-planning in project and production planning. *Omega*(30), 155 – 170, 2002.
- Chen, Y. L., Rinks, D. and Tang, K. Critical path in an activity network with time constraints. *European Journal of Operational Research*, **100**, 122-133, 1997.

- Crowston, W. Decision cpm: Network reduction and solution. *Operations Research Quarterly*, **21**(40), 435-445, 1970.
- Davis, E. W. An experimental investigation of resource allocation in multiactivity projects. *Operational Research*, **24**, 587-591, 1973.
- Demeulemeester, E., Vanhoucke, M. and Herroelen, W. Rangen: A random network generator for activity-on-the-node networks. *Journal of Scheduling*, **6**, 17-38, 2003.
- Demeulemeester, E. B., Dodin and Herroelen, W. A random activity network generator. *Operations Research*, **41**, 972-980, 1993.
- Elmaghraby, S. E. *Activity network: project planning and control by network models*. New York: Wiley, 1997.
- Elmaghraby, S. E. and S., H. W. On the measurement of complexity in activity networks. *European Journal of Operational Research*, **5**, 223-234, 1980.
- El Sakkout, H. and Wallace, M. Probe backtrack search for minimal perturbation in dynamic scheduling. *Constraints* **5**, **4**, 359 – 388, 2000.
- Gillies, D. and Liu, J. Scheduling tasks with and/or precedence constraints. *Society for Industrial and Applied Mathematics Journal on Computing*, **24**(4), 787-810, 1995.
- Goldwasser, M. H. and Motwani, R. Complexity measures for assembly sequences. *International Journal of Computational Geometry and Applications*, **9**, 371-418, 1999.
- Hall, N. and Posner, M. 2000. *Sensitivity analysis for intractable scheduling problems*. Research paper, The Ohio State University.

- Hapke, M., Jaskiewicz, A. and Slowinski, R. *Fuzzy multimode resource-constrained project scheduling with multiple objectives. in: Weglarz, j. (ed.), project scheduling : Recent models, algorithms and applications.* Kluwer Academic Publishers, 1999.
- Herroelen, W. and Leus, R. The construction of stable project baseline schedules. *European Journal of Operational Research*, **156**, 550-565, 2004.
- Herroelen, W. and Leus, R. Project scheduling under uncertainty: Survey and research potentials. *European Journal of Operational Research*, **165**, 289-306, 2005.
- Hillier, F. S. and Lieberman, G. J. *Introduction to operations research.* McGraw-Hill, 2001.
- Icmeli-Tuket, O. and Rom, W. O. Ensuring quality in resource constrained project scheduling. *European Journal of Operational Research*, **103**, 483-496, 1997.
- Igelmund, G. and Radermacher, F. J. Algorithmic approaches to preselective strategies for stochastic scheduling problems. *Networks*(13), 1-28, 1983.
- Kolisch, R. and Padman, R. An integrated survey of deterministic project scheduling. *Omega*, **29**(3), 249-272, 2001.
- Kolisch, R., Sprecher, A. and Drexel, A. Characterization and generation of a general class of resource-constrained project scheduling problems. *The Institute of Management Sciences*, **41**, 1693-1703, 1995.
- Leon, V., Wu, S. and Storer, R. Robustness measures and robust scheduling for job shops. *Institute of Industrial Engineers transactions*, **26**(5), 32-43, 1994.
- Master, A. A. An experimental and comparative evaluation of production line balancing techniques. *The Institute of Management Sciences*, **16**, 728-746, 1970.

- Meththa, S. V. and Uzsoy, R. M. Predictable scheduling of job shop subject to breakdowns. *Institute of Electrical and Electronics Engineers Transactions on Robotics and Automation*, **14**(3), 365-378, 1998.
- Pascoe, T. L. Allocation of resources -cpm. *Revue francaise de Recherche Operationelle*, **38**, 31-38, 1966.
- Patterson, J. H. A comparison of exact procedures for solving the multiple constrained resource project scheduling problem. *The Institute of Management Sciences*, **20**, 767-784, 1984.
- Prabuddha, D., Dunne, E. J. and Ghosh, C., J. B. and Well. The discrete time-cost tradeoff problem for project networks. *European Journal of Operational Research*, **81**, 225-238, 1995.
- Radermacher, F. J. Scheduling of project networks. *Annals of Operations Research*(4), 227-252, 1985.
- Rommelfanger, H. *Fulpal: An interactive method for solving (multiobjective) fuzzy linear programming problems*. in: Slowinski, r., teghem, j. (eds.), *stochastic versus fuzzy approaches to multiobjective mathematical programming under uncertainty*. Kluwer Academic Publishers, 1990.
- Sadeh, S., N. and Otsuka and Schelback, R. 1993. *Predictive and reactive scheduling with the microboss production scheduling and control system*. (Tech. Rep.). Proceedings of the IJCAI-93 Workshop on Knowledge-based Production Planning, Scheduling.
- Schwindt, C. 1995. *A new problem generator for different resource-constrained project scheduling problems with minimal and maximal time lags* (Tech. Rep.). University of Karlsruhe: WIOR-Report-449, Institut fur Wirtschaftstheorie und Operations Research.

- Skutella, M. Approximation algorithms for the discrete time-cost tradeoff problem. *Mathematics of Operations Research*, **23**(4), 909-929, 1998.
- Stork, F. 2000. *Branch-and-bound algorithms for stochastic resource-constrained project scheduling*. (Research Report No. 702/2000.). Technische Universitat Berlin.
- Tavares, V., Ferreira, J. A. A. and Coelho, S. J. On the optimal management of project risk. *European Journal of Operational Research*, **107**, 451-469, 1998.

