第一章　緒論

1.1 研究背景

對於快速發展的現代企業而言，其所執行的專案(Project)多面臨內容多元化及規模大型化的趨勢之中，面對產品生命週期的縮短、嚴格的品質要求及相關新興技術的開發，導致了企業對研究及開發專案需求的增加；在這種多專案並行、高技術、新製程、快速變化及資源有限的環境下，專案往往會因外在不確定的環境因素與所規劃的目標造成落差。根據Coppendale (1995)及 Chatzoglou & Macaulay (1996)的研究指出，平均僅約10%～15%的大型專案能夠如期完成，其餘均會發生執行延遲的現象。這不僅會造成專案在時間上的拖延，更會讓專案所投入的成本負擔增加。因此對於專案管理者而言，要能不斷推出新產品、開發新技術及新製程以提升競爭力，就必須在專案中的每個執行環節的時間點上予以精確的評估。然而專案的作業數量往往非常龐大，且其執行時間並非固定，通常為隨機變數，該時間變數亦會受外在所投入之資源量而有所改變，因此專案完成時間之估算及量化分析實非易事。另專案進行的過程中，專案經理人會針對某些瓶頸的專案作業投入額外資源，以期縮短專案的完成時間。然而隨機性的專案網路在有限資源情況及專案成本的考量因素下，專案瓶頸作業之判斷及其投入資源的數量，並非一般現有的專案分析工具所能夠勝任的。因此能夠有效的估算專案時程、掌控進度並決定適當的資源投入量，在專案管理中亦為非常重要的課題。

鑑於前述專案執行的環境條件日趨複雜且多元的情況下，本研究依專案問題種類及其相關分析技術整理如圖 1.1 所示，圖中將每一種分類所較常使用的解法或代表性的文獻列出。

圖 1.1 將專案問題依作業性質分成兩類，一為確定性的作業時間，另一為隨機性的作業時間。每一分類中，又因不同的資源投入條件細分成三種: 無限資源條件、有限資源條件及時間成本互償(Time/Costs trade-off)條件，在無限資源條件下，專案的目標在於如何求得最快的完成時間；在有限資源條件下，會形成專案排程(Project scheduling)等問題型態；在時間及成本互償的條件下，則屬處理專案資源投入最佳化之問題。

圖 1.1 中，在確定性的專案作業時間下，專案相關的問題種類，已有

相當多的學者從事這方面的研究，尤以專案排程(Project Scheduling)問題最為顯著，European Journal of operation Research (EJOR)期刊在 1990~1996 年間，前後收錄約兩百多篇該專案問題之相關文獻。然而在現實的專案環境中，專案作業時間為一隨機變數，專案的完成時間亦必為一隨機變數，隨機變數間的演算，須以數理統計分析為基礎，較為複雜。因此在確定性的專案作業時間下，其所使用的相關理論分析、演算工具及流程模組均不再適用，且依本文之瞭解，針對該類問題所發表的相關文獻並不多，故本論文的研究重點，擬以隨機性的專案作業時間之問題種類為對象。

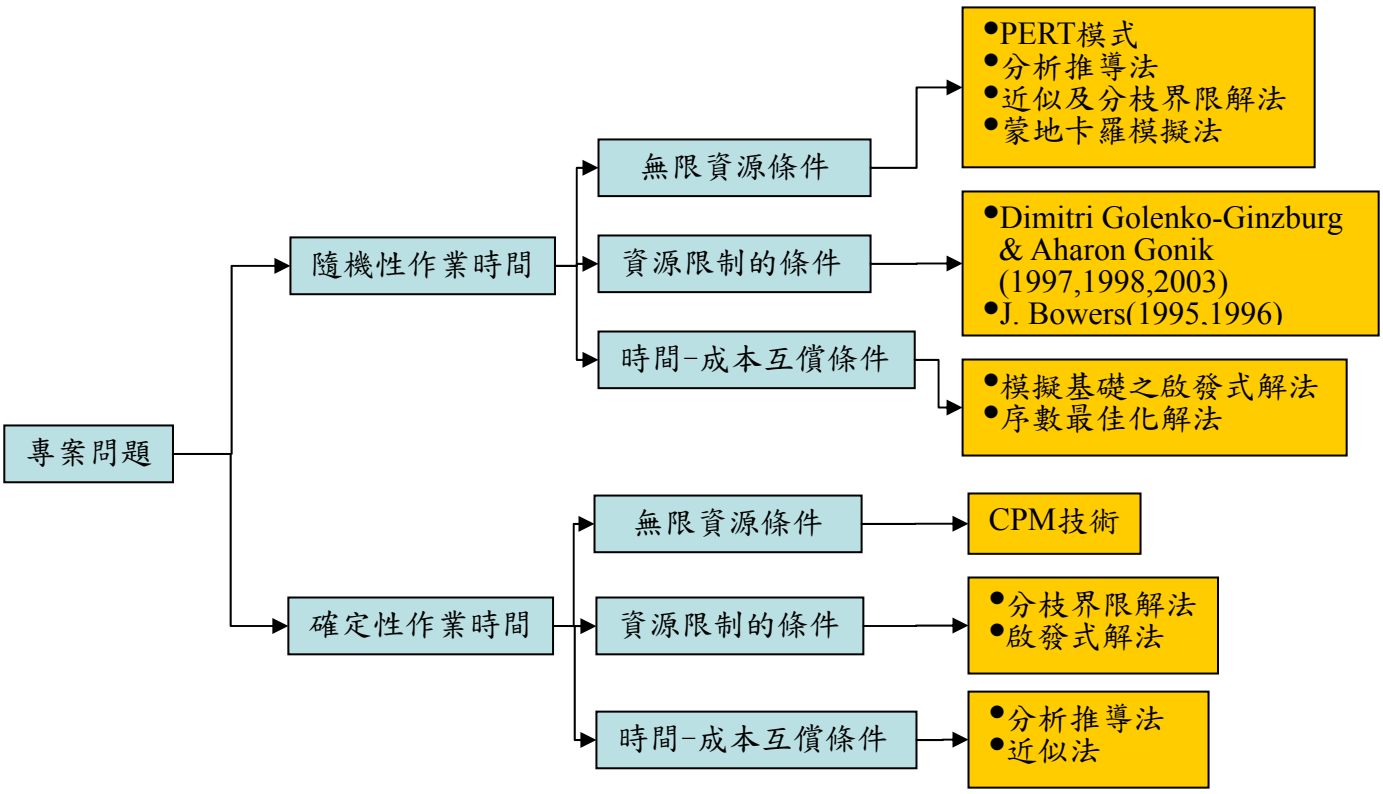


圖 1.1 專案問題種類及其分析技術

1.2 研究動機

鑒於前文所述，在隨機性專案作業時間下，處理該類專案問題的文獻並不多，因為隨機性專案作業時間均為隨機變數之故，不易以一般數理方式執行分析計算，專案分析的方法及技術會更為複雜。然而不論其複雜的程度為何，所有隨機性專案問題中，首要先解決之問題為專案網路完成時間之估算。舉凡專案排程問題、專案最佳化資源投入問題，專案成本及專

案風險之評估等，均須以專案完成時間作為參考之依據，因此專案完成時間的估算精度及其估算速度，會嚴重影響隨機專案上述相關問題求解的品質。且就專案管理者而言，準確及快速的評估出專案完成的時間亦是最重要的決策項目之一。本論文擬先針對此一問題做研究，期能發展出有效率的演算法來估算隨機性專案網路完成時間的機率分配。有效率的演算法不僅能夠執行精確的估算，同時亦須考慮到執行估算所耗的時間，因為該執行時間通常會隨專案網路作業量的增加而呈指數函數的成長。根據Fishman(1985)指出，即使一小型之隨機性專案(作業數量少於 50)，執行其數理分析都會很困難。因此就實務面而言，本論文所發展出來的演算法須有別於其它使用數理分析的演算法，要求其能實際應用於大型的專案網路。另隨機性專案網路完成時間的估算過程中，須將專案中的「路徑相依性」考慮進去，這因素會影響隨機性專案網路完成時間之估算精確度，且常為相關文獻所忽略之。專案在其各完成路徑(即專案起始節點至終端節點之路徑) 彼此間通常會存在共同之作業，這些共同之作業會同時影響這些路徑的完成時間，因此這些專案完成路徑不得視為相互獨立。Michel, Francesco & Salvatore(2000)指出，隨機性專案網路完成時間之求解為一NP-hard 問題，甚少有文獻提出解決方案，即便有之，亦僅侷限於專案作業時間在某單一之機率模式。例如 Martin(1965)將作業執行時間限定為Polynomial function 以利專案完成時間之推論；Sculli(1983), Clark(1961), Cox(1995)則限定為常態分配，Kamburowski(1985a,1985b)使用了指數分配, McBride & McCelland(1967), Premachandra(2001)則使用 PERT 網路模式中之 Beta 分配。另 Kulkarni & Adlakha (1986)及 Hardie(2001)使用了 Iterative markov process 來處理隨機性專案的估算，而本論文則允許專案作業時間機率分佈可為任何型態之情況。

除了上述隨機性專案網路完成時間之問題，Elmaghraby(1995)亦指出了近年來專案網路研究的四個重點趨向發展：

1. 發展新的數學式及模型來描述專案，並執行演算分析。
2. 在資源限制的條件下之專案排程問題。
3. 專案的最佳化之成本分析。
4. 隨機的作業時間條件下，最佳資源的投入。

在專案網路有限的資源或是完成時限下，針對上述專案網路未來發展趨向

之第 3、4 項，其目標為計算專案網路中最佳化的成本或決定各專案作業之最佳資源投入(Optimal resource allocation)量。因專案網路的成本與資源投入是息息相關的，當專案作業投入愈多的資源便可縮短愈多的作業時間，同時也會增加其資源使用成本；若投入之資源量過少，致使專案完成時間超過規定完成時限，則會產生懲罰成本。很明顯的，專案資源的投入，致使專案懲罰成本與資源使用成本之間存在著取捨(Trade-Off)的關係，且會影響到專案的完成時間亦同時影響專案的總成本(Dawson & Dawson(1998), Tavares & Ferreira(1998)及 Ward & Chapman (2003))。在專案作業資源有限制投入的條件下，欲針對所有專案作業安排最佳之資源分配量(Resource Allocation)，以求取最少的完成專案總成本，該類問題我們稱為隨機網路資源分配問題(Stochastic network Resource Allocation Problem, SRAP) 。根據 Antonella & Lorenzo(2001)稱此問題亦屬 NP-hard 問題。專案處理 SRAP 問題，尤其針對稀有資源之分配最能顯現其重要性，專案管理者所投入的每一分資源都不能浪費，務必要恰如其分，而其中相關的決定並非易事，因隨機性專案網路中，每一作業對所投入之資源量所造成的專案完成時間變化均不一樣，故專案所對投入不同型態之資源組合所所產生之總成本結果亦不一樣。且專案所投入之資源種類，對專案中作業的時間影響方式亦有所不同，因此在這複雜的專案決策環境中，不容易使用一般的數理分析方式得到解答。就實務上而言，本論文擬以前述估算隨機性專案網路完成時間之演算法為基礎，另發展出一演算法處理隨機性專案網路 SRAP 問題，除能做最佳資源分配外，同時亦能夠應用於大型的專案網路中。對於專案經理人來說，在隨機性專案網路中，能夠迅速的作出作業資源最佳分配之決定，以求得最小的專案總成本，亦是專案管理者另一重要之決策考量。

綜合上面所述，估算隨機性專案完成時間必須考慮路徑彼此間之相依性及專案作業時間亦不得為固定之機率分配，且估算大型專案完成時間又必須要求其估算效率及速度，因此不適於以數理分析的方式執行之。故鑑於隨機性專案完成時間其機率分配估算的重要性，且迄今尚無相關文獻提出有效的解法，本論文擬期發展出一近似解演算法(Approximation Algorithm)，執行大型專案完成時間之估算。

1.3 研究目的

依前文研究動機內容所述，本論文的研究目的可以分成兩部分:第一部分為估算隨機性專案網路完成時間之機率分佈，第二部分為處理隨機性專案網路 SRAP 問題。因考量隨機性專案網路完成時間估算的準確性，其估算過程必須將專案完成路徑彼此間之相依性納入考慮；另就實務上而言，專案作業時間機率分佈亦不得侷限在某固定之機率分佈型態，且估算大型專案完成時間又必須要求其估算效率及速度。因此本論文所發展之演算法，其執行的隨機性專案網路所要求的條件綜整如下:

1. 發展之新演算法不僅能估算出隨機性專案完成平均時間(Mean)及變異數(Variance)，並能夠實際估算出該完成時間之機率分佈。
2. 隨機性專案基於非回饋性(Acyclic)、連結性(Connected)及定向性(Directed)等網路模型條件下，其作業時間機率分佈要能適用於一般的機率分配，不能僅侷限在任何特定的機率分佈。
3. 依據 Kolisch(1996)，專案網路結構複雜度(Complexity)與作業之間節點連接數量及密度有關係，因此新的演算法對於複雜度較高的專案網路結構仍能保證其估算的品質。
4. 若干文獻通常以較小型的專案來做實驗，或專案作業時間侷限在某些單一之機率分配條件下；實務上專案常須處理超過上百的作業，因此演算法的複雜度及電腦計算效率必然是一個考慮的條件因素。

雖已有若干文獻針對上述兩種目的提出解法，然而就本論文所瞭解，迄今尚無任何相關文獻，針對上述隨機性專案網路之執行條件，提出有效率的演算法，多數演算法均未將專案網路完成路徑相依之因素納入考慮，或令專案作業執行時間侷限在某特定的機率分配型態。在此情況下，迄今亦尚無任何文獻所提出之演算法，能夠準確且快速的估算出大型隨機性專案網路的完成時間機率分佈，若未能準確的估算該完成時間，更遑論再進一步處理隨機性專案網路 SRAP 問題。

實務作業中估算隨機性專案網路完成時間，計畫評核術(Program Evaluation and Review Technique, PERT) 最被廣泛的使用也最受青睞，然而利用該方法所得之估算結果，已受若干文獻質疑，且該方法亦僅能求得專案完成時間之期望值而無法提供其機率分佈之情形。本論文擬於下節文獻

探討文中，詳細針對 PERT 之缺點予以說明，並將其估算專案網路完成時間所產生之誤差，以實際專案網路範例之實驗數據來表示之。經本論文參閱近二十年來相關於隨機性專案網路之文獻，其中執行隨機性專案完成時間估算，較具代表性的演算法為 Dodin(1985b)所提出，本文將該演算法稱之為 Dodin Algorithm(DA)。該演算法屬一近似解法(Approximation method)，雖能實際估算出隨機性專案完成時間之機率分佈，然而其同多數文獻一樣，未能將專案完成路徑相依性納入考慮，且未能提供大型專案網路實際演算所得之數據結果，故仍無法判斷其實務上運用之可行性。本論文擬以 DA 演算法為比較之對象，並針對其缺點做改善，接續再深入探討其它相關近似解法之文獻，藉以充分瞭解隨機性專案的特性，以期能發展出有效之隨機性專案完成時間估算之演算法。

SRAP 問題亦同隨機性專案完成時間問題一樣，雖已有若干的文獻提出相關之解法，然多數文獻亦均未能符合上述隨機性專案網路所要求之三項執行條件，近年來僅見 Tereso, et al. (2004a, 2004b),利用動態規劃法(Dynamic Programming, DP)及啟發式演算法結合蒙地卡羅模擬法來求解 SRAP 問題。很明顯的，該文獻利用動態規劃法及蒙地卡羅模擬法來求解隨機性專案網路的完成時間，對演算的效率而言毫無幫助，因此僅能處理較小型的專案網路。依該文獻所述，動態規劃法甚至無法處理作業量超過 36 之專案網路，且僅以取樣值 100 之蒙地卡羅模擬法，來執行隨機性專案網路完成時間之估算，蒙地卡羅模擬法過少的取樣值，會對估算的準確性受到相當之影響。針對該文獻之缺點，本論文擬以前述所發展之隨機性專案完成時間估算演算法為基礎，並充分理解隨機性專案網路資源投入與專案網路完成時間彼此間之關係，待隨機性專案完成時間有了較準確的估算，且將專案網路資源投入之特性納入演算之考量後，隨機性專案網路 SRAP 問題便能獲得可信度較高之演算處理。

1.4 研究方法與步驟

本論文擬先蒐整探討隨機性專案之文獻，並依圖 1.1 的分類原則分類之，文獻的分類有助於研究對象問題的歸類，並能夠有系統的瞭解隨機性專案網路相關問題的脈絡。再依據本論文研究的動機及目的，將研究的架構分成下列七個階段，前五個階段主要是針對隨機性專案網路完成時間為

研究的對象，其目的是要發展出較具效率之估算方法，本論文亦於此階段中發展出兩種不同的演算法: 標籤修訂演算法(LCTA)及區域積分演算法(MIA)。後兩階段乃處理隨機性專案網路 SRAP 問題，其中一個階段僅考量單資源種類投入，且使用改良之磁場機制演算法(EMA)作為決策之工具；最後一個階段則考量多資源種類投入情況，並利用基因演算法(GA) 來作為決策之工具。各階段的研究內容及步驟整理如圖 1.2 所示，並說明如下：

1. 第一階段：研讀隨機性專案完成時間之相關文獻，以充分瞭解各種估算專案完成時間所使用的分析方法，做分類並記錄其優缺點，同時確認本論文所發展之方法與其他文獻沒有重覆之處。本論文所發展之演算法，其理論基礎須排除使用數理分析法、界線分析法及 MCS 模擬法等相關文獻，因這些方法理論無法達成本論文之研究目的。另就現有之文獻，找尋最具代表性的演算法，作為本論文比較之對象，除針對該演算法作通盤的瞭解外，並以實際專案網路執行演算，以確實掌握其優、缺點。
2. 第二階段：為本論文的重點工作，即是發展出較有效率的演算工具來估算專案完成時間。演算法除須強調估算的準確性外，亦要求其估算的效率。首先須將專案網路實際轉換成一結構模型，該結構模型須能夠充分表達出隨機性專案作業之相關特性，例如作業執行時間及作業前後順序關係等。再利用圖形理論(Graph theorem)做為該結構之分析工具，另一方面則是利用適當的軟體將上述的演算法程式建構出來。相關的執行步驟如下：

   (1)將轉換成圖形理論中之樹形結構(Tree structure)能夠很清楚的表現其內部各節點間之前後順序關係，其階層化且具彈性的結構關係非常適合作為專案網路轉換的結構模型對象。

   (2)束形的結構可結合圖形理論之追蹤演算法(Tracing Algorithm)，可以有系統的拜訪其內各個節點位置，本論文可透過該追蹤機制，加上適當的演算便能夠有系統的將專案完成時間累算出來。

   (3)樹形結構的資料結構亦很單純，易於程式化；Matlab ver.6.5 程式語言除具有高能力之矩陣演算機制外，其中部分指令能夠很清楚的描述樹形結構之資料結構。故本論文擬使用該程式語言執行其相關之演算。
3. 第三階段：本論文使用 DA 演算法相同之離散式計算方式，即將所有作業時間機率分佈，取樣後並以離散的資料表示。在執行演算的過程中，

convolution 及 max 運算均使用離散式的計算方式。然而離散計算存在兩個主要問題，其中 max 運算會因「最長路徑偏差(LPB)」因素之故，造成專案網路完成時間估算上之誤差，PERT 即因未考慮該因素，致使其估算網路完成時間造成相當的偏差，本論文擬於後文針對該問題再作詳細之說明。離散運算的另一個問題，就是作業的離散單元，在運算的過程中會以等比級數方式擴增，致使演算效率大打折扣，因此本研究擬發展出一重複取樣方法(Resampling Method)，在不影響精確度的情況下，有效的降低離散單元的擴增。本階段相關的執行步驟如下：

(1)分別試用 Agrawal & Elmaghraby(2001)所提之三種方法離散技術，並取其最適當的方式來執行之。

(2)使用離散技術(Discrete technique)將所有作業時間機率分配予以離散化。

(3)將重複取樣技術(Resampling)納入離散技術演算中，以增進離散運算之效率。

4. 第四階段：本階段將專案網路完成路徑彼此相依的因素納入考量，路徑相依之相關證明已可以數學式來表示之，任何演算法若無考慮專案網路路徑間之相依性，其估算出來的隨機性專案完成時間必定存在著相當之誤差。為彰顯其重要性，本論文擬以一專案網路產生器，隨機設計出一大型專案網路模型(100 個節點)作為實驗的對象，並使用 DA 演算法執行完成時間之估算，所得到的結果再與 20000 次取樣之蒙地卡羅模擬法結果來比較(可視為真實值)，因 DA 演算法並未將路徑相依性納入演算法中，因此路徑相依性所造成之估算誤差，比較其結果即可證明之。本階段相關的執行步驟如下：

(1)發展出路徑相依的相關理論，並將其納入演算法中，相關之理論擬於後文詳述之。

(2)自行設計程式，隨機產生 100 個節點之大型專案模型作為實驗的對象，其中網路複雜度(Complexity)、作業數目均可以調整，另作業時間可為多種機率分佈型態。

(3)使用 DA 演算法與 20000 次取樣之蒙地卡羅模擬法分別執行(2)項之專案網路模型實驗，其估算之結果差異甚巨。

(4)結合第三、四階段之工作，發展出本論文第一個針對隨機性專案網路完成時間估算之演算法: 標籤修訂追蹤演算法(Label-Correcting Tracing Algorithm, LCTA)。

5. 第五階段：標籤修訂追蹤演算法(LCTA)執行的過程中，max 離散運算最為耗時且會產生最長路徑偏差(LPB)，即若干並行路徑匯集在某一節點時，該節點的輸出時間結果之機率分佈函數，並不等於並行路徑中最長路徑時間機率分配函數。考統計學中「順序統計理論」(Order Statistic Theory)即已對此提出相關的理論。本論文擬針 max 離散運算，設計出更有效率之離散計算技術來取代 max 運算，則會令 LCTA 估算專案網路完成時間變得更有效率。本階段即發展出區域積分演算法(Marginal Integration Algorithm, MIA)，相關的理論及說明詳如後文。
6. 第六階段：完成前五階段之工作後，即產生較為準確的演算法來估算隨機性專案網路完成時間，本論即可接續處理隨機性專案網路 SRAP 問題。為求單純化，本階段先以單種類資源投入專案為處理問題之對象。處理 SRAP 問題，最可行的方法是採啟發式演算法來作為決策的工具，找尋專案作業中最佳的資源分配組合。本論文擬採磁場機制演算法(Electromagnetism algorithm, EM)，雖有理論證明(Birbil, et al., 2004)該演算法能找到最佳解，然而其缺點是求解收斂速度過慢。經充分理解該演算法之性質，且針對隨機性專案網路資源投入對專案完成時間的特性做研究，本論文已有效解決該缺點，故將其謂之為改良式磁場機制演算法，整個求解之程序謂之為 EM_SRAP，相關之說明亦詳如後文。
7. 第七階段：本階段乃處理隨機性網路多種類資源分配問題(SMRAP)，與上階段不同之處，在於專案網路考慮多種類之資源投入，且本階段之啟發式演算法擬採用基因演算法(Genetic Algorithm, GA)。因每一種類的資源對於專案作業的時間影響均不相同，在多種類資源投入的狀況下，其資源分配相關的決策亦會變得非常複雜，故啟發式演算法仍為合適可行的決策工具，而考量使用 GA 演算法原因有二: 針對 SMRAP 問題所得之結果，擬與上階段改良式 EM 在效率上做一比較；另一原因為本階段的決策環境較複雜之故，GA 演算法非常合適應用於該問題上。

第一階段

1. 依所蒐整之文獻及研究動機所關注的問題，研讀相關之文章，以充分掌握足夠的資訊。
2. 找尋關於求解隨機專案網路完成時間代表性之文獻 Dodin(1985a)，所發展出之 Dodin Algorithm(DA)擬作為本研究之比較對象。

↓

第二階段

1. 將專案網路以樹狀網路模型表示之，並以相關圖形理論來作為專案完成時間估算之演算流程基礎。
2. 利用 Matlab 的軟體將專案樹狀網路模型建構出來，並將相關之作業資料結構架設起來。

↓

第三階段

1. 依 Agrawal & Elmaghraby(2001)所提之三種方法建構作業離散計算之機制。
2. 為解離散單元等比級數的擴張的問題，發展出重複取樣技術(Resampling)，並控制住額外失真之問題。

↓

第四階段

1. 修改原先之演算法，將路徑相依問題考慮進樹狀網路模型。
2. 設計網路產生器，製造大型專案網路實例來做實驗。
3. 將實驗結果與 Dodin Algorithm(DA)及蒙地卡羅模擬法(MCS)之結果相比較。

↓

第五階段

因離散式運算執行專案網路中路徑 max 運算較花時間，且亦是估算最大誤差來源，嘗試深入 max 運算機制，使整體估算效率增加。

↓

第六階段

1. 參考相關文獻，定義出單種類資源投入對作業時間所造成的影響。
2. 採用磁場機制演算法作為決策工具，並針對其缺點加以改良。
3. 使用 Matlab 程式語言處理 EM_SRAP 問題。

↓

第七階段

1. 參考相關文獻，定義出多種類資源投入對作業時間所造成的影響。
2. 採用基因演算法作為決策工具，並分析找出該演算法最佳執行參數。
3. 使用 Matlab 程式語言處理 GA_SMRAP 問題。

圖 1.2 研究架構之七個階段工作概述

1.5 論文架構

本論文的內容共分為六章，研究架構如圖 1.3 所示，分別簡述如下。第一章敘述本論文之研究背景、目的、研究方法、步驟及論文架構等相關內容。第二章則整理及探討與本論文研究相關題目之文獻，主要包括求解隨機性專案網路完成時間及最佳資源分配等問題。第三章介紹本論文所發展的第一個估算隨機性專案網路完成時間之演算法 LCTA，本章節會詳細介紹離散式運算技術、max 運算所造成之 LPB 偏移量、重複取樣技術、路徑相依性對估算完成時間的影響、專案網路轉樹狀結構及其相關之追蹤計算流程。第四章則介紹本論文所發展的第二個隨機性專案網路完成時間估算之演算法 MIA，基本上 MIA 所採用的相關估算理論及技術同 LCTA，唯 MIA 設計一特別的運算元來處理 LPB 偏差，用來取代原 LCTA 中之 max 運算。第五章則負責處理隨機性專案網路 SRAP 問題，其中專案採單種類資源投入採改良式磁場機制演算法來作為資源分配的決策工具，而多種類資源投入則採基因演算法來作為資源分配的決策工具。第六章說明論文未來研究發展方向，第七章則歸納本論文的結論。

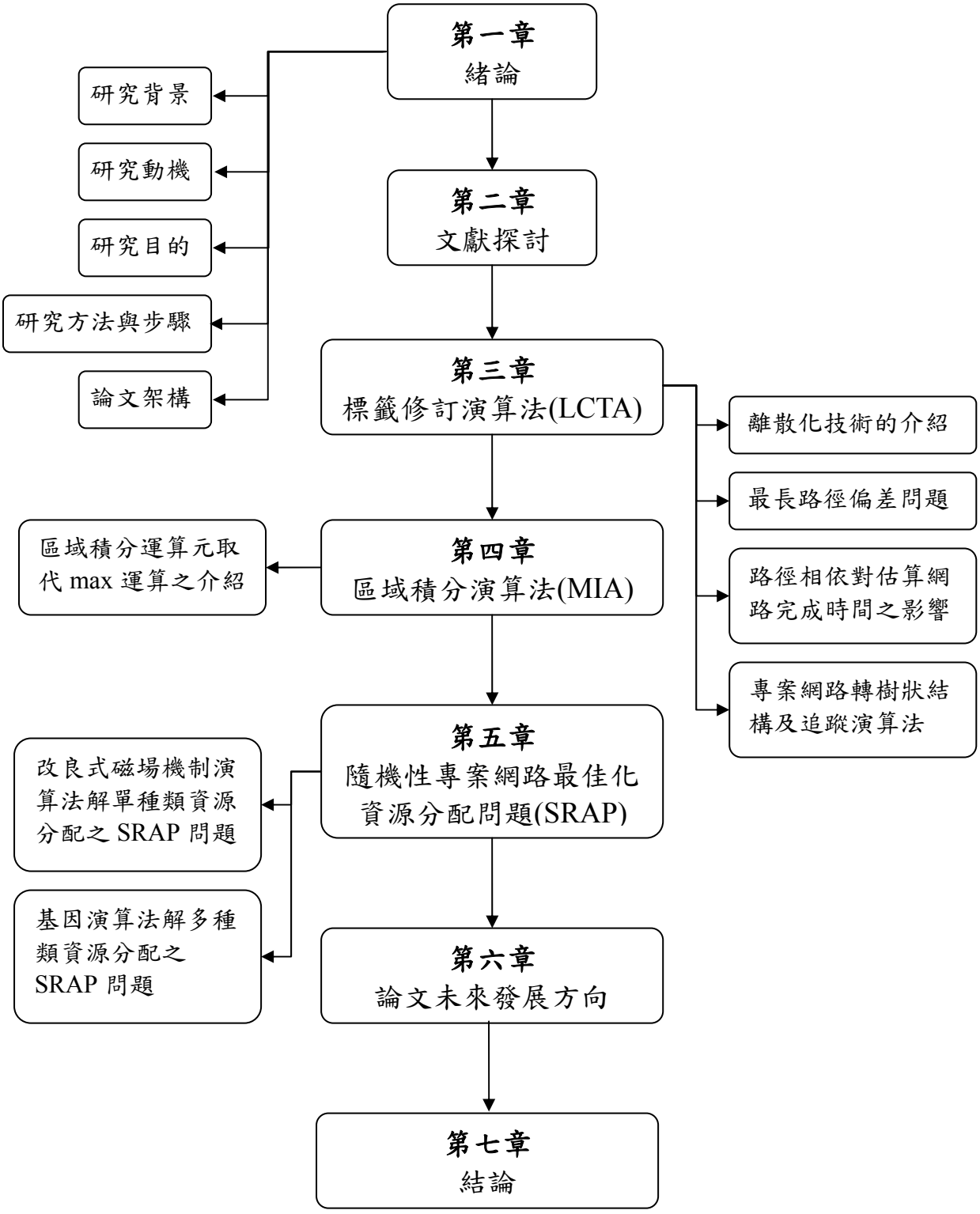

第一章
緒論
研究背景
研究動機
研究目的
研究方法與步驟
論文架構
第二章
文獻探討
第三章
標籤修訂演算法(LCTA)
離散化技術的介紹
最長路徑偏差問題
路徑相依對估算網路完成時間之影響
專案網路轉樹狀結構及追蹤演算法
第四章
區域積分演算法(MIA)
區域積分運算元取代 max 運算之介紹
第五章
隨機性專案網路最佳化資源分配問題(SRAP)
改良式磁場機制演算法解單種類資源分配之 SRAP 問題
基因演算法解多種類資源分配之 SRAP 問題
第六章
論文未來發展方向
第七章
結論

圖 1.3　論文架構

第二章　文獻探討

估算隨機性專案網路完成時間最常用且熟悉的工具為計畫評核術(Program Evaluation and Review Technique, PERT)，因該方法單純且估算容易，倍受工商企業界普遍應用。然而 PERT 理論是建立在若干假設條件下，致使該方法之應用受到相當之限制，面對大型複雜之專案無法執行精確之估算。本章節擬將 PERT 分析模式做一概略介紹，並將其使用的限制因素做詳細之說明。其它有關隨機性專案網路完成時間估算之文獻(1966~1987)，則由 Adlakha & Kulkarni(1989)做了整理，並將其歸納為三個種類：分析推導法(Exact analysis approach)、近似解法(Approximation approach)及蒙地卡羅模擬法(Monte Carlo Simulation, MCS)，擬分別敘述如下文。隨機性專案網路資源分配問題是本論文所討論的第二項主題，主要是處理隨機性專案網路單種類資源及多種類資源，如何以最佳化的分配在專案作業中以求得最少的總成本。其相關討論的文獻並不多，但仍由本論文將其分類為三類: 數理分析解法(Analytical approach)、模擬分析法(Simulation approach)及啟發式解法(Heuristic approach)，相關的介紹如下文所述。

2.1 PERT 網路模式的介紹

於1950年代末期，美國政府鑒於在研究發展方面的契約日漸增加，新武器的交貨日期延後頻繁，致使國防預算成本難以適當地規劃和控制，為解決此一問題，美國海軍部專案局(Special Project Office)研究提出計劃評核術(簡稱為PERT)，並首先運用於北極星飛彈的計劃中，且由Malcom, et al. (1959)首先發展出PERT相關之統計分析方式。PERT能針對任何大、小專案執行計畫、評價與查核的技術，特別是專案中有許多性質不同的作業，且須按一定次序進行及規定完成期限之專案。PERT的特色是以事件為導向的網路圖為其基本架構，考量不確定性的作業時間。在PERT中， 將專案以節點(Nodes)作業線(Arcs)的集合來做表示，並以Activity on Arcs(AOA)或Activity on Nodes(AON)兩種網路表示方式描述專案結構，其表達專案的方式既簡單又能夠反應出作業間前後順序關係。

Lau & Somarajan(1995)指出，典型PERT 的計算程序，主要是遵照下述3項步驟執行：

1. 界定出專案內所有作業之樂觀時間a（Optimistic time）、悲觀時間b（Pessimistic time）與最大可能時間m（Most likely time）：

(1)樂觀時間： 即在最順利的情況下，作業最短的完成時間。

(2)悲觀時間：即在最不順利的情況下，作業最長的完成時間。

(3)最大可能時間：即作業的最大可能完成時間。

2. 假設作業時間服從Beta 分配，並計算其平均值與變異數。由於Beta 分配是使用a、m、b 三種數值來描寫作業時間，所以稱為「三時估計法」。依據Beta 分配，專案中每項作業 i 之時間，其平均值$\bar{t}_i$ 與變異數σ_i^2可表示如下：

$$\bar{t}_i = \frac{a_i + 4m_i + b_i}{6} \tag{2.1}$$

$$\sigma_i^2 = \frac{(b_i - a_i)^2}{36} \tag{2.2}$$

3. 專案完工時間T為所有路徑完工時間之最大值，即為關鍵路徑完成時間：

$$\begin{aligned} T_j &= \sum_{i \in P(j)} \bar{t}_i \\ T &= \max(T_j) \end{aligned} \tag{2.3}$$

其中

$P(j)$：第j 條路徑所有作業的集合。

$\bar{t}_i$：第i 個作業的平均時間。

T_j：第 j 條路徑完工時間。

2.2 PERT 的不適性

PERT 雖廣受歡迎，但就實務的觀點，PERT 的分析模式是在下述假設條件下才成立的(Elmaghraby 1977)：

1. 專案中作業時間的機率分配為 Beta distribution。
2. 專案中每一路徑含有大量的作業，依統計學中之中央極限定理(Central Limit Theorem)，可將其完成時間機率分配視為常態分配(Normal distribution)。

3. 專案中各路徑均互相獨立(Independent)，互不影響。

從上述前三種PERT假設我們不難瞭解到這些假設條件往往在實際的情況下是無法滿足的，就第三種假設而言，專案中，若任兩路徑使用到共同的作業，則在此種情況下兩路徑便不能視為互相獨立；針對第二種假設，因PERT乃執行專案完成時間的平均值時間及變異數之估算，故其完成時間之機率分配，則基於中央極限定理，以常態分配視之，然該條件必須在大型專案網路中才能適用。第一種假設則是PERT最常用的作業時間機率分配之假設(即Beta機率分配)，最初由Malcom, et al. (1959)所提出，且使用具有對稱性質的常態分機率分配。但是考慮到作業時間不能有負值的情況下，Williams(1962)建議作業時間應使用四種具不對稱性之機率分配：Beta, Triangular, Gamma及Exponential機率分配，其中証明又以Beta機率分配為最佳。另有若干文獻質疑Beta機率分配的三種估算模式的合理性，即Gallagher(1987), Grubbs(1962)及Sasieni(1986)針對PERT之作業時間所訂出的端點限制(Absolute endpoints)，認為不合理，MacCrimmon and Rayvec (1964)則認為使用Beta分配來估計作業時間，可能發生誤差。實際上該公式的估計確實存在相當的限制，且可經由運算式推導之結果可以證明，若PERT採用(2.1)、(2.2)式之簡化Beta運算模式，其專案作業時間機率分配函數之偏態值 SK (Skewness value)，只能限制在三種情況，即 $SK=(0,-1/\sqrt{2},1/\sqrt{2})$，若在此三種情況外，將會使得隨機性專案完成時間平均值估算之信心水準降到非常低。詳細的證明過程如附錄A所列。

雖一般文獻所處理的網路作業時間均會使用四種具不對稱性之機率分配：Beta, Triangular, Gamma及Exponential機率分配，但機率性模式仍舊無法獲得學術界一致的認同，因為如果使用同一機率分配函數來表示所有作業時間，則無法滿足各個作業相異的情境條件；若使用不同機率分配函數來表示不同的作業，卻有計算上的困難。對於機率性模式的批判，Kamburowski (1987) 則提出三點看法：

1. 作業時間是由專家的主觀意見所決定的，而非服從某一機率分配。
2. 專案計畫及其中的作業，常無法重複樣本的觀察，故以機率分配來表示作業資料並不合理。
3. 機率方法的運算存有許多困難點與複雜性。

根據上述的分析說明可知 PERT 雖然使用上非常方便，能夠很快的估計出專案的完成時間，然而針對上述若干的假設條件及機率的分析中，似乎又顯得粗略了些。

2.3 隨機性專案網路完成時間之估算

對專案管理者而言最重要的決策之一就是要能夠準確的評估出專案完成時間，這在確定性(Deterministic)的專案網路而言可利用 CPM 技術便可輕而易舉的求出專案完成時間。在未討論隨機性專案網路之前，本章節先針對確定性的專案網路來做瞭解。確定性的專案網路的完成時間是從關鍵路徑(Critical Path, CP)來決定的，即要先找到網路的 CP，再決定其完成時間。CP 可由下列線性規劃(LP)模式定義求取：

$$\begin{aligned} \max \quad & \sum_{(i,j)\in \mathbf{A}} X_{ij} q_{ij} \\ s.t. \quad & \sum_{j\in A_1} q_{ij} = 1, \\ & -\sum_{i\in B_j} q_{ij} + \sum_{k\in A_j} q_{jk} = 1 \quad \forall j = 2,3,\ldots,n-1, \\ & -\sum_{i\in B_n} q_{in} = -1, \end{aligned} \tag{2.4}$$

其中 A_i 表示專案網路中節點 i 接續之節點集合，B_i 則表示專案網路中節點 i 輸入之節點集合，q_{ij} 為附加在作業(i, j)上的資源流量，可以增、減作業的時間，$i \to j$ 則為其作業時間進行的節點順序方向。X_{ij} 則是作業(i, j)的基本時間。式(2.4)的 Dual LP 模式可以表示為：

$$\begin{aligned} \min \quad & \gamma_n - \gamma_1 \\ s.t. \quad & \gamma_j - \gamma_i \geq X_{ij} \quad \forall (i,j) \in \mathbf{A}, \end{aligned} \tag{2.5}$$

其中 γ_i 為專案網路節點 i 之輸出時間。

就現實層面看，對隨機性專案網路而言，X_{ij} 作業時間通常不是固定不變的，意即為一時間隨機變數，有其個別的機率分配函數(Probability distribution function, *pdf*),因此整個專案完成時間也必定為一機率分配函數，作業間總合的時間不再能用加減的關係而須以 convolution 運算來取代之，故其專案的複雜的程度不能再以確定性網路的觀點待之，即不能再將 CPM 中之最長路徑當作是專案完成時間路徑，而是要以機率的觀點來分析之，因為最長的路徑時間是一個時間機率變數，一個專案可能同時存在多條完成路徑，當這些路徑之時間機率變數其機率分配之值域(Domain)有重

疊時，就機率的觀點而言，表示每條專案完成路徑都有機會成為專案的要徑(Critical Path)，專案完成時間必須考慮到所有可能完成路徑，若指定某特定路徑為要徑，並指定專案完成時間即為該路徑之完成時間，該計算方式並不合理，因此式(2.4)及(2.5)便為隨機規劃之問題(Stochastic Programming Problem)。

若干文獻對於隨機性專案完成時間則用採界限解法(Bounding approach)，估算專案完成時間之上、下界值。該方法代表性的文獻計有 Kleindorfer(1971), Shogan(1977), Splede(1977), Robillard, Trahan(1976), Dodin(1985a)等。另部份文獻則估算其平均值及變異數之臨界值，主要代表文獻計有 Clark(1961), Fulkerson(1962), Elmaghraby(1967), Robillard, Trahan(1976), Devroye(1979), Sculli(1983)及 Kamburowski(1985a, 1985b)。因界線解法並無法實際得到專案完成時間機率分配之資訊，這便造成實用上的限制，亦不符合本論文研究的目的。

針對隨機性專案網路完成時間機率分配之求解，雖已有若干的文獻提供相關的演算法，然均有其執行效率的問題，於實務上無法運用在大型專案中。 Adlakha & Kulkarni(1989)將這些文獻歸納為三個種類：分析推導法(Exact analysis)、近似解法(Approximation approach)及蒙地卡羅模擬法(Monte Carlo Simulation, MCS)，本文將依序說明如下文。

2.3.1 分析推導法

此類方法以數理分析方式計算，得到該專案完成時間之機率分配函數。分析推導法直接將路徑作業間變數，以 convolution 及 max 運算直接累加求得專案完成時間封閉式。然而當專案網路作業量增加，網路路徑數量亦跟著增加，convolution 及 max 運算次數重複執行的次數亦增多，則路徑的完成時間之計算會變得非常複雜而無法解求得其機率分佈函數。因此 Hartley and Wortham (1966), Ringer (1969), Martin (1965)及 Robillard & Trahan(1977)利用串聯/平行(Series/Parallel)處理技術，先將專案縮小到不能再縮小的網路結構(Irreducible network)，以降低專案的複雜度。所謂縮小網路結構就是將專案網路結構內之部份節點，經過了化簡的程序並以單一節點來取代之。Elmaghraby & Pulat(1979)對網路結構縮小，提供了相關的演算規則。Hartley and Wortham 及 Ringer 的觀點指出，整個網路為若干小單位結構節點(Wheatstone Bridge 及 Double Wheatstone Bridge) 類以拼圖方式

所構成，故 Hartley and Wortham 將網路 Wheatstone Bridge 的結構(即“Crossed network”結構，為不可化簡之最小結構)，進一步轉化成單一節點，並求出其機率分配函數。Ringer 則將網路中另一種不可縮減的 Double Wheatstone Bridge 結構(亦謂之“Criss-crossed network”)作進一步之化簡。專案之網路結構若不先執行串聯/平行演算，其 convolution 及 max 運算的次數，將會隨專案作業量之增加，以指數函數方式增加。即使如此，串聯/平行縮減計算仍會遇到兩種問題：

1. 如何發展出一些原則能夠很有效率的將專案縮小？

2. 如果存在兩個以上的 Irreducible network patterns，如何加以區別之？

Colby(1984)證明上述的兩個問題為 NP hard 問題。當然另有一些學者提供了其他的觀點，利用一些技巧將專案化簡成若干簡單的子網路，再行處理；如 Burt & Gartman(1971)使用了”Common Arcs”的概念來縮減網路作業量，Sigal, et al.(1979)則利用了”Maximal Uniformly Directed Cutest, MUDC”的概念來分解網路，然其所遇到的問題亦類如上述兩點的情況。因此「網路縮減的作業量」變成了隨機性專案中使用分析推導法的重要指標，因為它與網路複雜度(Network complexity)幾乎可以相提並論(Elmaghraby & Herroelen ,1980)。

分析推導法中尚有若干學者提出不同的演算方式，Charnes (1964)提出一”Chance Constrained”的模式概念，並利用現性規劃法來執行網路完成時間的機率分配函數的求解，該方法之實用性亦不適用於大型的網路結構。另有 Fisher(1983,1985)利用”Order-of-Precedence(OP)”方法將網路中可能的完成時間作業路徑序列 S 集合找出，並利用該序列計算出完成時間的機率分配函數。Fisher 主要的缺點在於，即使一個小的網路結構，S 序列集合的數量就非常的龐大了，故其求解並不容易。若所有的作業均為指數函數(Exponential distribution)，則 Order-of-Precedence(OP)的方法就可視為連續時間的馬可夫鏈模式(Continuous Time Markov Chain, CTMS)，Kulkarni & Adlakha (1986)即是利用此法能有效的擺脫冗長的 S 完成時間作業序列之搜尋，然仍無法擺脫因計算所產生之狀態空間(State space)數量，亦同樣會隨網路之作業量之增加，呈指數函數的增加。

從上述所有的分析推導法文獻來看，雖有若干文獻對於網路的分析卓有貢獻，但是幾乎所有文獻所提之方法，其計算的複雜度，均會隨網路作

業量增加而呈指數函數的提升，就實務層面而言並不理想。後期有關於這方面的文獻亦並不多。其中 Schmidt & Grossmann(2000)使用 PD-type 函數來取代作業的時間機率分配函數，該函數為一在固定範圍的值域中，Piecewise polynomial Segments 及 Dirac delta 函數所構成的，並利用該函數的運算來求出網路的完成時間機率分配，然該方法仍不易擺脫上述分析推導法所存在之問題，故均不為本論文所參考採納。

2.3.2 蒙地卡羅模擬法

用蒙地卡羅模擬法(Monte Carlo Simulation, MCS)的方法來模擬專案的完成時間機率分佈函數是最精確也最直接的方法，文獻中通常也常以取樣數量很大的 MCS 模擬的結果作為實驗的真實值，來與其他的演算結果作比較。唯其缺點就是模擬計算所花的時間過長，對於大型專案而言計算負荷過重。早期首位使用 MCS 來執行專案完成時間的文獻為 Van Slyke(1963)，它使用的模擬法我們稱之為"Crude MCS"，文獻中將 MCS 的整個運算過程以簡潔、清楚的數學式來表示。

MCS 的取樣數目愈多，則精確度愈大，但是相對的會犧牲其運算速度，故有若干文獻提出方法來增加該 MCS 的精確度及運算速度，Burt & Garman(1971)利用"Unique Arc"的概念，先將網路中僅存在於一個路徑上的作業(即不與其他路徑共用相同的作業)決定出來，再透過相關的理論分析，以減少須執行取樣的網路作業數量，又能夠同時達到相同的精確度要求，這方法我們亦稱之為"Conditional MCS Simulation"。 Sigal, Pritsker & Solberg (1979)提出了前文所述的 MUDC 來達到降低取樣作業量維持同樣的精確度之效果。另 Sullivan & Hayya(1980)將 Kleindorfer(1971)之界線法與 MCS 法來做一比較，證明 MCS 法比該界線法更勝一籌。本研究擬使用大數量的取樣數量(例如 20000 次)來執行 MCS，其目的是要求取接近專案的實際值來作為一標準值，並與其他估算法作比較，因此雖然還有相當多的方法被提出來以增進 MCS 的效能，這些亦不在本研究的重點範圍之內因此就不予一一列舉。

2.3.3 近似解法

鑒於上述分析解法的複雜性及蒙定卡羅模擬法負荷過重之計算問題，不難了解估算隨機性專案網路的完成時間機率分佈實際上是非常困難

的。因此近似解法(Approximation approach)就應運而生。然而多數近似解法相關之文獻，為求計算上的簡化，均對網路作了若干條件的假設。例如網路中所有完成路徑，決定其中一條關鍵路徑(Critical Path)，再由該路徑計算出專案網路完成時間，PERT 即採此方式(Malcolm, et al., 1959)，取關鍵路徑上所有作業之時間期望值及變異數之和，作為專案網路完成時間之解答。在隨機性專案網路中，從起始點到終點任何一條路徑，均有機會成為專案網路之關鍵路徑，因此透過任何指標計算來指定網路中之關鍵路徑並不合理。另 Sculli(1983)則利用 PERT 及中央極限定理(Central Limit Theorem)為依據，計算專案網路完成時間並以常態分佈函數(Normal distribution function)表示之。Clark(1961)亦將專案作業之時間以常態分佈函數表示，估算完成時間之期望值及變異數。然而專案作業時間機率分佈，若侷限某特定之型態並不合理，且在專案網路完成時間之計算過程中，當兩路徑進入同一作業節點則須執行 max 運算，兩常態分佈函數其 max 運算的結果並不必然為常態分佈函數，且 max 運算的結果絕非為最大之輸入路徑值，這與 PER 的觀點不同，max 運算會因「最長路徑偏差(LPB)」因素之故，造成專案網路完成時間估算上之誤差，LPB 現象可由統計學中之「順序統計量(Order Statistics Theorem)」解釋，另 Klingel(1966), Mehrotra, et al.(1996)及 Pontrandolfo(2000)對此問題亦有做相關的討論，並指出當兩變數之機率分配，值域重疊的部分愈多，則產生 LPB 偏差愈大，相關的解釋及說明擬後文詳述。Dodin and Sirvanci(1990)亦指出隨機性專案網路的完成時間機率分佈會呈極值分佈(Extreme value distribution)而非上述之常態分佈，因此若將專案網路完成時間以常態分佈函數視之，並不正確。另中央極限定理也必須在路徑作業數量夠大的情況下才能成立。除上述文獻較常使用隨機性專案網路的假設條件外，大部分的文獻亦未將路徑相依性考慮進去，即兩路徑間可能存在部分相同之作業，這些作業會同時影響兩路徑之執行時間，這情況會影響估算的結果。Mehrotra, et al. (1996)將專案網路分成兩部分: 一部分為路徑不相依，另一部份路徑具相依性。兩部分分開計算，最後再將結果合併。Dodin and Sirvanci(1990)亦利用極值分佈查詢表(Look-up tables of extreme value distribution)，計算專案網路完成時間機率分佈。該兩文獻雖將網路路徑相依性考慮計算，然因演算法過於複雜且累算過程中需要執行查表的程序，對於大型的專案網路而言並無效率。因此近似解法在執行專案網路完成時間估算，欲得到較佳的精確度、演算效率及實務上應

用，須考量下述四種條件：

1. 不能採用 PERT 對專案網路的假設(指定關鍵路徑, 作業期望值, 中央極限定理)，且演算法執行 max 運算時須將 LPB 偏差納入考慮。
2. 隨機性專案網路作業時機不得侷限任何特定之機率分佈型態。
3. 專案網路中，完成路徑間須考慮路徑相依性的影響。
4. 將相關的計算公式、程序能夠方便電腦輸入計算，Martin(1965)即將作業時間以多項式表示，再利用 convolution 及 multiplication 執行多項式的運算，這種多項式表示方法即非常適合電腦運算。

最具有代表性的的近似解法為 Dodin(1984), Dodin(1985b), Dodin & Elmaghraby(1985c), Agrawal & Elmaghraby(2001), Fatemi & Rabbani(2003)，其中前三個文獻，運用了離散化(Discritization)運算技術，除了執行隨機性專案網路完成時間之估算外，另亦處理隨機性專案網路前 *K* 條最長路徑、計算關鍵路徑指標(Path Critical Index, PCI)及關鍵作業指標(Activity Critical Index, ACI)值的估算。離散化運算技術即是將連續性函數的機率分配予以離散化(Discritization)處理，其相關的加、減、乘、除之計算亦以離散化的方式執行。其優點是可省略複雜的公式推導及數理分析，其缺點就是會與實際結果產生一些誤差。

Dodin(1985b)是少數文獻中，能實際將隨機性專案網路完成時間機率分佈值估算出來，該文獻先利用串列/並行(Series/Parallel)的縮減技術簡化網路結構，以降專案網路的複雜度，再利用利用前述之離散化運算技術，以疊代計算方式(Iterative operation)，執行專案網路中作業間 convolution 及 max 運算。然而該文獻仍存在若干缺點，首先是未將路徑相依性考慮進演算法中，其二是該文獻的疊代計算方式並無一系統化的執行方式，僅以一 4 個節點的網路實例說明之，這對於大型專案網路而言，並無實務上的貢獻，而亦無法驗證其執行估算的效果，該文獻相關的介紹及說明於下一章節詳述。

如前文所述，當專案作業數增加，界限解法則無法得到實際專案完成時間之機率分配資訊，分析推導法數理計算過於複雜，無法運用於實際之問題，蒙地卡羅模擬法則會遭遇計算負荷過於龐大的問題。唯上述所提之近似解法(Approximation approach)能夠實際估算出專案完成時間之機率分配資訊。然就本文所瞭解，有關近似解法之文獻(包括上述之 Dodin

Algorithm)均無考慮到路徑相依的問題，且無提供執行大型專案的實例，無法得知其估算的效率及精確性，故就實務上而言，對於隨機性專案完成時間的估算，仍尚存努力改善的空間。

2.4 隨機性專案網路最佳化資源分配之處理

如何將資源精準且有效率的分配在每個專案作業中，一向為專案管理者最重要的決策項目。因為外部作業資源的投入量對專案作業執行時間會有相當的影響，資源投入愈多，作業時間愈縮短，專案的完成時間亦會隨著縮短而能減少專案逾時的機率。雖此舉能降低專案逾時的成本，然而額外的資源投入，同時也增加專案資源的使用成本；很明顯的，專案資源的投入，會令專案逾時成本與資源使用成本間，存在著取捨(Trade-Off)之關係。Demeulemeester, et al.(1996)將該專案資源分配問題整理成三種型態：

1. 資源不可重複使用(nonrenewable resource)且投用量有限制之條件下，求專案網路最小的完成時間。
2. 在專案網路完成時間規定的時限條件下，求最少的資源投入量(不可重複使用)。
3. 資源不可重複使用且投用量有限制，專案網路完成時間亦有時限的條件下，求最少的專案網路總成本(包括資源使用成本及超過時限之懲罰成本)。

本論文擬處理專案資源分配問題屬上述第三種型態，在隨機性的專案環境中，作業資源有投入限制的條件下，針對所有專案作業安排其最佳之資源分配 (Resource Allocation)，以求得最少的完成專案總成本，該類問題我們稱為隨機性專案網路資源分配問題(Stochastic network Resource Allocation Problem, SRAP)。若參考第一章圖 1.1 之專案問題分類，SRAP 屬隨機性作業時間類別，時間-成本互償條件項目之問題。該分類之專案問題所處理之對象，即為上述專案最佳化資源分配問題。Antonella & Lorenzo(2001)視該問題為一強烈的(strongly) NP hard 之解題型態，又 Antonella & Lorenzo(2001)認為該問題求解演算法的複雜度，會隨著專案網路的增加而成指數函數的增加(Exponentially worest-case complexity)。

就本論文之瞭解，處理SRAP問題的文獻並不多，且一直未獲重大進展，其主要原因在於隨機性專案問題本身除了網路完成時間不易準確估算

外，專案網路的每一作業時機機率型態，因應不同種類之資源投入及可用資源的限制因素，使得SRAP求解過程中隱含高度的複雜性，致使早期研究SRAP的文獻嘗試著各種不同的演算法，然而都未能獲得良好的成果。由於SRAP的諸多問題無法獲得有效的解決，而且現行解法僅能用來求解較小型的專案問題，因此目前仍無任何可行之演算法提供實務界作應用。本論文針對處理SRAP文獻，所提出之演算方法概分為三類：

1. 數理分析解法(Analytical approach)：相關的文獻計有Tereso, et al. (2004a),Basso & Peccati(2001)及Elmaghraby(1992)。
2. 模擬解法(Simulation approach)：相關的文獻計有Dorp & Duffey(1999)及Chu & Yao(2003)。
3. 啟發式解法(Heuristic approach)：Boctor(1990), Boctor(1996), Kolisch & Drexl(1997), Leu & Yang(1999) 及 Feng, et al.(2000)。

上述三種方法擬分別敘述如下。

2.4.1 數理分析解法(Analytical approach)

數理分析解法一般文獻常用整數規劃法(Integer Programming, IP)、分枝界限法(Branch and Bound algorithm)及動態規劃法(Dynamic Programming, DP)。

2.4.1.1 整數規劃法(Integer programming, IP)

「0與1 整數規劃法」是專案排程問題中較早被提出的最佳解法。0與1整數規劃法的基本觀念是將整個專案的計劃期間分割為幾個區間或工期，某項作業在該時間點上「是否正在進行」、「是否開始」或「是否結束」，以0 或1 來表示。目標函數乃視公司期望達成目標而定，至於技術限制(Technological constrains)亦即條件限制式，通常須滿足下列限制：(1)作業先後順序關係限制(所有先行作業未完成前，其緊接作業不得開始)；(2)資源限制式(每期某種資源總需求不得超過該資源的可用數量)；(3)作業不得分割的限制；(4)作業工期的限制。至於其他限制式則視目標函數之不同以及各模式建構之需求再自行加入。

由於決策變數不容易定義、限制式建立且需大量運算時間，因此如何減少決策變數及條件限制式的個數以增進整數規劃的求解速率，為該方法

之桎梏。由於現階段該決策變數及條件限制式的問題點仍無突破性的進展，因此對大型專案問題求解並無太大的助益。

2.4.1.2 分枝界限法(Branch and Bound algorithm)

分枝界限法基本上是一種列舉技術，透過分枝(Branch)的方式，不斷的將原來的問題分割成好幾個子問題，並利用上限值和下限值的比較，將某些不可能的子問題去除，以避免不必要的運算時間。多半的文獻只限於應用在簡單的例子上，每項作業只能一種資源型態需求。部分文獻大多集中於發展優越的上、下限值及凌越原則(Dominance rules)，以期更能有效率獲得最佳解。但一般而言由於網路型態因專案不同而有所不同，在進行凌越原則考量時很難將所有影響因素皆列入考慮，而且在進行凌越比較時將耗費大量的計算時間及佔用大量的電腦記憶空間，於是造成求解上的困難。

2.4.1.3 動態規劃法(Dynamic Programming, DP)

甚少文獻使用此方法，僅見近年來 Tereso, et al.(2004a)及 Basso & Peccati(2001)使用動態規劃法處理 SRAP。Tereso, et al.(2004a)先將專案網路利用“Cutset”概念找出“固定作業組(fixed subset)”的作業集合，並將該組作業視為專案網路“參考條件 (Conditional upon)”，即這些作業須以列舉法(Enumeration)之方式將所有可能的資源分配型態列舉出來，並選擇出最好的分配型態。專案網路再依據此資源分配條件，進行找出下一階段之“固定作業組” 的作業集合，不斷重覆此動作完成其動態規劃法之計算工作。Basso & Peccati 亦以類似之方法，唯採不同之分析策略。動態規劃法在執行的過程中無法避免的使用了列舉法計算，若遇上了網路作業量龐大情況下，會同上述數理分析法一樣存在計算上的負擔，造成執行上的困難。

2.4.2 模擬解法(Simulation approach)

依據 Abel (1998)模擬解法的處理程序不外乎有下列步驟：

1. 建立專案網路模式(作業特性、作業前後接續關係及資源種類等)。
2. 資源對作業時間機率分佈影響的模式關係。
3. 隨機針對各作業及資源變數取樣。
4. 依確定性專案網路模式計算專案網路完成時間及其總成本。
5. 記錄每次取樣所得的計算結果，最後再做資料分佈之綜合分析。

上述的模擬步驟並不能夠針對作業決定出最好的資源投入量，因此模擬法通常必須要配合其它決策演算工具，如 Chu & Yao(2003)使用基因演算法(GA)及 Tereso, et. al.(2004b)使用磁場機制演算法(EM)。

當專案網路作業量大時，模擬解法亦同樣存在計算上的負擔且更為耗時，現今一般的文獻不再採用此方法解 SRAP 問題。

2.4.3 啟發式解法(Heuristic approach)

由於 SRAP 問題具有 NP-Hard 的特性，因而限制了最佳解法在大型而且複雜專案下的應用與發展，大部分的文獻均朝向啟發式解法的方向進行，而且不斷有新的啟發式解法在發展中。因此解決 SARP 最實際有效的方法應為啟發式解法，該類方法中較為一般文獻所採用的不外乎為模擬退火演算法(SA)、禁忌搜尋演算法(Tabo search)、基因演算法(GA)及蟻群復至最佳化法(Ant Colony Optimization)等。近年來由 Birbil & Fang(2003)所提之磁場機制演算法(EM)亦為啟發式解法之一種，亦有完整的相關的證明(Birbil & Fang 2004)。因其理論架構完整且程序簡單較易於程式化之故，非常適合用來處理專案網路 SRAP 問題，Terreso, et al.(2004b)即利用 EM 解決 SRAP 問題，然該研究遇上兩個問題須待解決，第一個問題是當專案網路結構擴增、作業數量增加之際，EM 亦同其他上述啟發式解法一樣，遭遇決策變數求解範圍過大的問題，以致於嚴重影響該方法之效率。第二個問題是該研究採用計算量龐大的蒙地卡羅模擬法來先執行專案網路的完成時間之估算，進而求出專案的總成本。使用該模擬解法，即便處理一個小規模之專案網路，亦會讓整個 EM 執行效率下降。對於處理多資源種類的 SRAP 問題，因決策變數過多，啟發式解法多傾向使用基因演算法(GA)，如 Godfrey & Michael(2001)及 Luis & Antonio(1998)所示。

SRAP 問題除使用上述三種分類的方法，另也有文獻使用其它較特別種類的演算方法，其中 Wan(1994)採用晃動分析法(Perturbation Analysis, PA)來決定最佳專案網路作業資源投入量；PA 乃源於等待理論(Queueing Theory)，最初由 Ho, et al. (1979)所發展。該方法結合少量的模擬計算及最佳化的數理分析方式，透過對問題中某一參數之微量變化，便能觀測出問題整體的結果。Wan 首度將該理論運用於專案網路 SRAP 問題中，針對專案網路中作業參數微量的變化，預測出專案網路完成時間，並藉以判斷作

業適當的資源投入量。然而 PA 仍有其使用的限制，除要求作業調整之參數對作業時間存在可微分性，另需假設專案網路作業為指數分佈函數。

第三章　標籤修訂演算法

本章節的目的乃發展一演算法，估算隨機性專案網路完成時間之機率分佈函數值，本論文將其謂之為標籤修訂演算法(Label-Correcting Tracing Algorithm, LCTA)。該演算法除要求估算的結果要準確外，針對大型網路亦要求其演算效率。LCTA 採用了 Dodin Algorithm(Dodin, 1985b)中所使用的離散化運算技術，該技術結合現代電腦運算能力，能擺脫數理分析運算之桎梏。該演算法及離散化技術擬於本章節中詳述。Dodin Algorithm 雖是少數文獻中能實際估算出隨機性專案網路完成時間之機率分佈函數值，然而它並未考慮專案網路路徑間之相依性，且其執行網路作業節點之 max 及 convolution 疊代運算(Iterative operation)時，並沒有提出有系統或規律的演算規則，其演算複雜度(Complexity)無法得知，故實際應用於大型專案網路之可行性無從評估。

估算隨機性專案完成時間，若無考慮專案路徑相依之情況(即兩路徑共擁有相同之作業)，其所估算的結果會與實際值發生偏差，該偏差與網路中路徑間執行 max 運算有關係，本論文將其稱之為「最長路徑偏差(Longest Path Bias, LPB)」。Klingel(1966)為最早指出該問題之文獻，且該問題會隨網路結構複雜度(Complexity)增加而增大。DA 及其他相關之文獻均未見將該因素納入考慮，因此 LCTA 擬針對 Dodin Algorithm 估算未納入考量的因素做研究，並將其納入估算過程中；另利用圖形理論中之樹形結構(Tree structure)，並將該結構稱之為擴張樹結構(Expanded Tree Structures, ETS)，結構中將專案作業視為節點，並從該樹狀結構之根節點(Root node)處開始，依圖形理論中後序追蹤(Postorder Tracing)方式，有系統的完成網路作業節點之 max 及 convolution 疊代運算，並記錄修正所經過之節點標籤，最後能夠估算出專案之完成時間。本章節最後以 20 個大型專案網路(100 個作業節點)為實例，實際以 PERT、Dodin Algorithm 及 LCTA 執行所得的估算結果作比較，來證明 LCTA 優於其它演算方法。

論文中所使用之專案網路相關符號及運算式，其定義及說明表列如下：

X_{ij}：為專案網路節點 i 至節點 j 之作業時間。

Y_i^j：為專案網路節點 i 至節點 j 之路徑時間。

γ_j：為專案網路起始節點 1 至節點 j 之完成時間。

$f_{ij}(\cdot)$：為 X_{ij} 之機率分配函數(*pdf*)。

$g_j(\cdot)$：為 γ_j 之機率分配函數(*pdf*)。

B_j：為連接至節點 j 之後續節點(Precedence nodes)集合。

A_j：為節點 j 連接至其它節點之前續節點(Successor nodes)集合。

3.1 Dodin Algorithm 之介紹

Dodin(1985b)提出一種具直覺又不會過於複雜的演算法，我們稱其為 Dodin Algorithm(DA)，該方法乃用來估算隨經性專案網路完成時間機率分佈之近似值。在 DA 中，假設專案中各完成路徑均不相依(Independent)，且須執行專案網路序列/平行之縮減計算(Series/Parallel reduction)。DA 在演算過程中，須從專案網路始節點 *1* 持續不斷執行 convolution 及 max 運算，直至專案網路的終端節點 N 為止。convolution 及 max 運算表示式說明如下:

令$\oplus$為 convolution 運算符號，$f_{ij}(\cdot)$，$g_j(\cdot)$ 分別為 X_{ij} 及 γ_j 之機率分配函數，則通過節點 i 至節點 j 之路徑時間 Y_i^j 其計算表示式如下：

$$Y_i^j = \gamma_i \oplus X_{ij},\ \ i \in B_j \tag{3.1}$$

$$\Pr(Y_i^j \le t) = \int_{-\infty}^{\infty} g_i(t-\tau) f_{ij}(\tau) d\tau \tag{3.2}$$

假設另存在一節點 k 至節點 j 之路徑 Y_k^j，則節點 j 之完成時間 γ_j，其計算方式須對路徑 Y_i^j 及 Y_k^j 執行 max 運算，表示式如下：

$$\gamma_j = \max\{Y_i^j, Y_k^j\} \tag{3.3}$$

$$\Pr(\gamma_j \le t) = \Pr(Y_i \le t, Y_k \le t) = \Pr(Y_i \le t) \cdot \Pr(Y_k \le t) \tag{3.4}$$

DA 在累算的過程中，對專案網路中的任一節點 j 之完成時間 γ_j 之計算，須先找出節點 j 所有的輸入節點集合 B_j(Precedence nodes)，再依據(3.2)及(3.4)式，計算所有到達節點 j 的路徑完成時間 $Y_i^j, i \in B_j$。假設所有到達節點 j 的路徑共計 q 條且彼此相互獨立，則隨機變數 γ_j 的累積機率分佈可以表示為：

$$\begin{aligned} \Pr(\gamma_j \le t) &= \Pr(Y_1^j \le t, Y_2^j \le t, ..., Y_i^j \le t) \\ &= \Pr(Y_1^j \le t) \cdot \Pr(Y_2^j \le t) \cdot ... \cdot \Pr(Y_i^j \le t),\ \ q = \left|B_j\right| \end{aligned} \tag{3.5}$$

DA 從起始點開始，針對專案網路中所有節點重複上述(3.5)式的運算，最後計算到終端節點 N 得 γ_N，該值便是專案網路完成時間機率分佈之近似值。

本節擬以圖 3.1 為例，對 DA 運算的過程進行說明，DA 從專案網路起始節點 *1* 開始累算時間，連接下一個節點 *2, 3* 及 *4*，計算過程開始依序先執行 γ_2, γ_3 及 γ_4 之完成時間。γ_2 的完成時間即等於作業時間 X_{12}，節點 3 有兩條輸入路徑 Y_1^3 及 Y_2^3，其中 Y_1^3 為作業時間 X_{12} 與 X_{23} 之和，Y_2^3 等於作業時間 X_{13}，因此 γ_3 為該兩條路徑執行 max 運算之結果。節點 *4* 為專案終端節點，γ_4 為專案網路之完成時間，該節點有三條輸入路徑 Y_1^4, Y_2^4 及 Y_3^4，依圖 3.1 所示，Y_1^4 為作業時間 X_{12} 與 X_{24} 之和，Y_2^4 等於作業時間 X_{14}，Y_3^4 則為節點 *3* 時間 γ_3 與作業時間 X_{34} 之和，γ_4 即為該三條路徑執行 max 運算之結果。相關求解程序如下：

1. $\gamma_2 = X_{12}$
2. $Y_2^3 = \gamma_2 \oplus X_{23}$
3. $\gamma_3 = \max\{X_{13}, Y_2^3\}$
4. $Y_2^4 = \gamma_2 \oplus X_{24}$
5. $Y_3^4 = \gamma_3 \oplus X_{34}$
6. $\gamma_4 = \max\{X_{14}, Y_2^4, Y_3^4\}$

γ_4 即為圖 3.1 中專案網路的完成時間，共執行六個運算步驟完成。

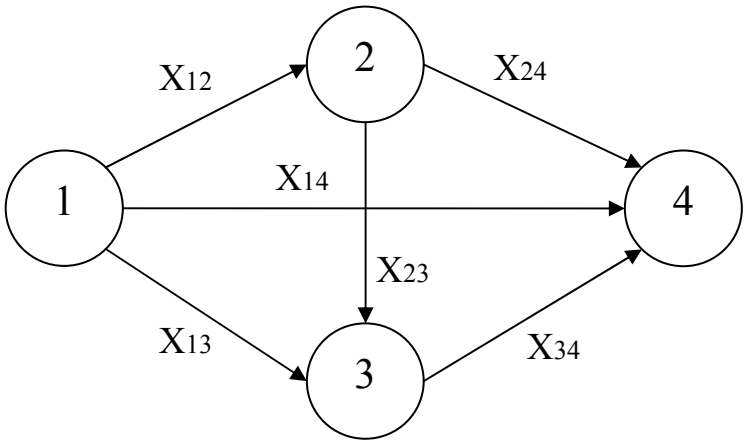


圖 3.1 四個節點之專案網路

3.2 離散化技術之介紹

DA 最重要的特色之一，就是專案網路在執行完成時間累算前，先將所有網路作業時間之機率分佈函數先予以離散化(Discretization)處理，計算中 convolution 及 max 運算均以離散化技術方式執行，使重複執行上述(3.2)及(3.4)式運算變為可行，故能有效的擺脫數理分析方式之桎梏。本節擬說明離散化技術的種類及其優缺點、離散化 convolution 及 max 運算如何執行及重新取樣技術之介紹。

3.2.1 離散化技術的種類及其優缺點

以作業 X_{ij} 為例，其機率分佈函數經離散化後，可表示為一組順序向量(Ordered vector)：

$$\{(v_k^{ij}, p_k^{ij}) \mid k = 1...m\} = \begin{pmatrix} v_1^{ij} & v_2^{ij} & \cdots & v_m^{ij} \\ p_1^{ij} & p_2^{ij} & \cdots & p_m^{ij} \end{pmatrix} \tag{3.6}$$

其中 m 表示離散取樣點數，v_k^{ij} 為 X_{ij} 第 k 個時間樣本點值，p_k^{ij} 為 X_{ij} 第 k 取樣點值，相對於 v_k^{ij} 之機率分佈函數值，其中 $\sum_{k=1}^{m} p_k^{ij} = 1$。

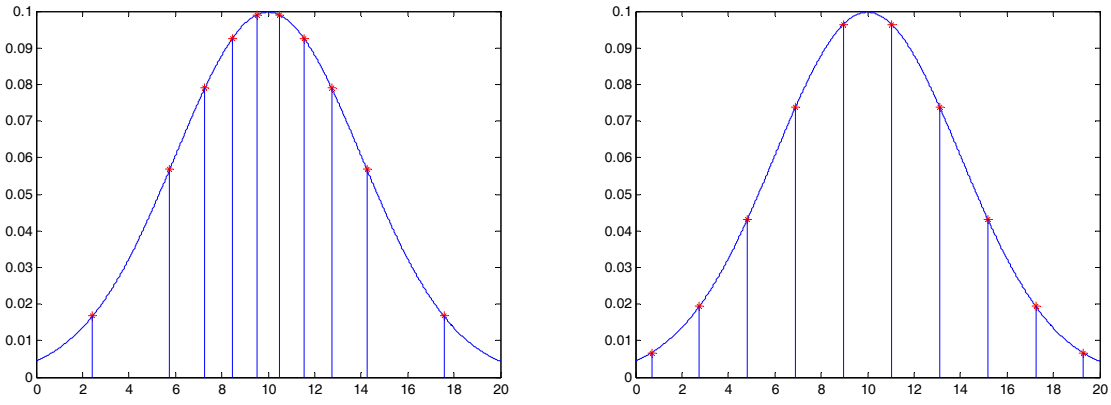
離散化技術雖有其優點，然而根據 Agrawal & Elmaghraby(2001)指出，離散化的運算會存在三種問題：

1. 記憶體需求量龐大：離散化樣本點數量在運算的過程中會持續不斷的擴大，電腦的記憶體需求量會持續跟著擴大。
2. 運算之精確度：離散化後結果是使用若干取樣點來表示連續性函數資料，取樣點數愈多則愈接近實際值，然終究其運算後的結果不能如連續函數之資料來的完整。
3. 分佈集中化(Support shrinkage)：依據中央極限定理(Central Limit Theory, CLT)，離散化資料經過若干運算過程後，其結果終究會趨向常態分佈，且其結果機率分佈函數時間軸上之分佈值，會傾向結果機率分佈函數之平均值位置靠攏集中，故其機率分佈範圍較不會如 MCS 模擬真實的分佈來得廣，即函數之變異數值會較真實值來得小。

針對上述第 1 及 2 項問題，若能配合適當的資料取樣點數，配合現代電腦的運算能力，電腦記憶體的需求量並不會構成嚴重的問題；且本論文亦透

過若干實驗驗證，若取適當數量之樣本點，對離散化運算所得結果，其機率函數平均值(Mean)並不會造成影響。唯該結果之機率函數變異數(Variance)會因上述第 3 項分佈集中化問題，會較實際值來得小。對專案管理者而言，機率分佈函數變異數變小，平均值不變的情況下，會讓專案之決策環境變得較為嚴謹，就管理實務角度來看並非缺點。然而針對專案風險評估之決策計算，其結果會趨向保守而與實際狀況失真。鑒於此，對本論文而言，離散化之相關技術存在若干之研究空間，期改善或解決上述所產生之問題。

Agrawal & Elmaghraby(2001)則介紹了三種離散化技術，且各有其優缺點，其分別為：柴比契夫取樣點法(Chebychev sample points)，等距取樣點法(Equal distance sample points method)及等機率取樣點法(Equal probability sample points method)。等距取樣點法就是將作業機率分佈函數之時間軸均分成數等份，每等份即視為一取樣點區間(Sample area)，在每一區間時間軸上取其中間值為該取樣點之時間樣本點值，其相對應之機率分佈函數即為該取樣點之取樣點值。等機率取樣點法同等距取樣點法作法一樣，唯其取樣點區間是將作業機率分佈函數均分成數等份，每一取樣點區間之累積機率值相同，及該作業時間每一取樣點之取樣點值是相同的。而每一取樣點區間相對應之時間樣本點為該區間之質量中心位置(Moment position)。該兩種技術的示意圖如圖 3.2 所示。



(a)等機率取樣點法　　(b) 等距取樣點法

圖 3.2 等機率取樣點法及等距取樣點法之執行範例

Agrawal & Elmaghraby(2001)所介紹之第三種離散技術—柴比契夫取樣點法，可以有效避免資料數值集中化問題，且為 Agrawal & Elmaghraby 所推薦。本論文將上述三種離散化技術實際以平均值 mean=0.25 之指數分佈函數為例，取 5 個樣值本值予以離散化，所得的離散資料再求算其前 4 個動差值(Moment values)，與 20000 次取樣之蒙地卡羅模擬法結果比較如表 3.1 所列。表中顯示柴比契夫取樣點法僅第一動差值略遜於等機率取樣點法(0.056 vs. 0)，其餘三個動差值均表現比另兩個方法好甚多，也較趨近於蒙地卡羅模擬法所模擬之真實值。實驗結果證明以柴比契夫取樣點法(Chebychev sample points method)為最佳。

表 3.1 三種離散化技術對 Exp(0.25)取樣後動差值之比較

		離散樣本值					動差值			
		i=1	i=2	i=3	i=4	i=5	1st	2nd	3rd	4th
CSP	x_i	0.04	5.31	13.84	22.36	27.63	0.056	18.96	122	2146
	p_i	0.41	0.49	0.09	0.01	0				
EP	x_i	0.45	1.47	2.85	5.05	10.18	0	11.93	34.92	332.5
	p_i	0.2	0.2	0.2	0.2	0.2				
ED	x_i	0.04	4.64	9.23	13.83	18.42	-0.25	17.78	79.44	1324
	p_i	0.44	0.38	0.12	0.04	0.02				
20000 次取樣之 MCS 之動差值							0	16	128.7	2296

CSP: 柴比契夫取樣點法

EP: 等機率取樣點法

ED: 等距取樣點法

3.2.2 柴比契夫取樣點法

針對 3.2.1 三種離散化技術比較之結果，本論文擬採用柴比契夫取樣點法執行專案網路離散化相關之運算，柴比契夫取樣點法與其它取樣技術最大的不同處，是其樣本點的取點位置很特殊，首先在時間軸上訂出欲離散

化之機率分佈函數的資料範圍[a, b] (參考圖 3.3)，a, b 兩點的決定可由隨機變數之機率分佈函數取其 0.05 及 0.95 累積機率之時間軸位置，並以該範圍直線距離為直徑畫一圓，其取樣點是均勻等距分佈在圓周上，再投影於時間軸上，該投影的位置就是柴比契夫取樣點法取樣點位置(若不包括兩端點，共取 5 個樣本點)。

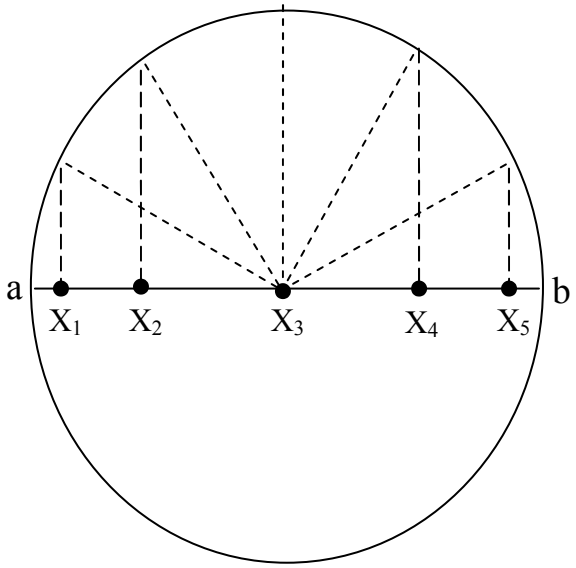


圖 3.3　柴比契夫取樣點法之取樣點圖示

柴比契夫取樣點法主要的取樣步驟表示如下:

1. [*a,b*]為作業時間資料範圍，利用下列之公式尋找 *Q* 個機率分佈樣本點:

$$x_i = a + \frac{(b-a)}{2}[1-\cos(\frac{\pi(2i-1)}{2Q})],\ \ i=1,...,Q \tag{3.7}$$

2. 依據作業時間的機率分佈型態及其所給定之參數資料(機率分佈之 moments 值 $m_k,\ k=1,...,4$)，利用下式求解相對於 x_i 離散資料之機率值 p_i。

$$\begin{aligned} &\sum_{i=1}^{5} P_i = 1 \\ &\sum_{i=1}^{5} x_i^k P_i = m_k,\ \ j=1,...,4 \end{aligned} \tag{3.8}$$

根據上式，一般作業時間機率分佈通常提供 4 個動差(Moment) m_k，故柴比契夫取樣點法所取樣的點數為 *Q*=5 個，當離散機率分佈能夠符合原機率

分配之前 4 個動差值的前提下，該離散機率分佈即與原式相去不遠。

範例 3.1:

針對平均值 mean=2 之指數分佈函數，資料範圍為[0, 20]，使用柴比契夫取樣點法取 5 個樣本點，執行該函數之離散化。

執行步驟:

1. 先計算出該函數之前四個動差值，分別為 [2.0, 8.0, 47.5, 372.77]。
2. 依據資料範圍[0, 20]及式(3.7)，計算出 5 個時間樣本點值為[0.49, 4.12, 10.0, 15.88, 19.51]。
3. 再依據式(3.8)之聯立方程式計算出相對應之 5 個取樣點值，求得離散化之樣本點為

$$\begin{bmatrix} 0.49 & 4.12 & 10.0 & 15.88 & 19.51 \\ 0.56 & 0.41 & 0 & 0.03 & 0 \end{bmatrix}$$

3.2.3 離散化 max 及 convolution 運算原理

存在兩專案時間變數 *X, Y* 之機率分配函數，其取樣點數分別為 *Q* 及 *R*,離散化後可以表示為：

$$X = \begin{bmatrix} x_1, x_2, \cdots, x_Q \\ p_1^x, p_2^x, \cdots, p_Q^x \end{bmatrix}, \qquad Y = \begin{bmatrix} y_1, y_2, \cdots, y_R \\ p_1^y, p_2^y, \cdots, p_R^y \end{bmatrix} \tag{3.9}$$

$$x_1 < x_2 < \ldots < x_Q,\ p_i^x > 0,\ i = 1,\ldots,Q,$$

$$y_1 < y_2 < \ldots < y_R,\ p_j^y > 0,\ j = 1,\ldots,R$$

X, Y 兩變數均表示為矩陣型態，第一列表示變數的函數值，第二列則表示相對應之機率值，該兩變數之離散化 max 運算結果 *Z*=max(*X,Y*)可以表示如下：

$$Z = \begin{bmatrix} z_1, z_2, \cdots, z_s \\ p_1^z, p_2^z, \cdots, p_s^z \end{bmatrix} \tag{3.10}$$

$$z_k = z_{ij} = \max(x_i, y_j),\ p_k^z = \sum_{(i,j)\in T} p_i^x p_j^y,\ T=\{(i,j) \mid z_{ij} \textit{ has same value}\}$$

max 相關的運算方法以下列程序來做說明:

1. 先取 X, Y 中每一樣本函數值來做 max 運算，可得 $z_{ij}=\max(x_i,y_j)$。所得之 Z 變數樣本函數值 $z_k=z_{ij}$ 相對應之機率值為 $p_{ij}^Z=p_i^x p_j^y$，其中 $i=1,\ldots,Q$, $j=1,\ldots,R$, 且 Z 變數最多可有 $W=\max(Q,R)$ 樣本單元。

2. 將求出之 Z 變數每一樣本函數值 (z_{ij},p_{ij}^Z), 依 z_{ij} 從小至大做排序。

3. 將 z_{ij} 值相同的樣本函數值合併為一，並表示為 z_k，且其中相對應之 $p_i^x p_j^y$ 要加總 $p_k^z=\sum_{(i,j)\in T} p_i^x p_j^y$。

從上述可知，max 運算第一步驟需執行 $Q\times R$ 次 $\max(x_i,y_j)$ 運算，第二步驟則須執行 $W\ln W$ 次排序工作，第三步驟亦同樣需要 W 次合併比較。convolution 運算亦同 max 運算方式執行，其中僅將第一步驟中之 $z_{ij}=\max(x_i,y_j)$ 改為 $z_{ij}=x_i+y_j$ 即可，且 Z 變數最多可有 $Q\times R$ 樣本單元。運算的複雜度亦同 max 運算。

未進一步說明 convolution 及 max 運算執行情況，本章節利用圖 3.1 為範例，並設定每一專案作業一離散資料，實際以 DA 演算方式執行專案網路完成時間之估算，相關的說明及計算程序參考附錄 B。

3.2.4 離散化重新取樣技術

就實務上而言，執行 X, Y 變數離散化 convolution 及 max 運算，所得之結果 $Z=\{z_{ij},\ i=1,\ldots,Q,\ j=1,\ldots,R\}$，若執行相同的 z_{ij} 樣本單元合併可以減少樣本單元數。然而 z_{ij} 樣本單元數值通常為帶小數之實數，因此 z_{ij} 具相同樣本單元之機率並不大。在此情況下，X, Y 變數間執行 convolution 及 max 運算所得到的 z_{ij} 樣本單元數目一般情況為 $Q\cdot R$，若運算持續執行若干次，其所得結果之樣本單元數目會成長得非常快速。參考 3.2.3 節 convolution 及 max 運算所需花費的執行步驟，樣本單元數過大仍會造成離散化運算之負擔，這會嚴重影響專案網路整體估算之效率。因此必須要降低該樣本單元數目，同時要維持變數相關數值(平均值、變異數)之精確度。本論文擬以上述所求之 Z 變數為例，將其所得到的 z_{ij} 樣本單元(假設共有 $Q\cdot R$ 個)重新取樣，並將重新取樣的樣本數定為 S，即將(3.10)式轉換成：

$$Z' = \begin{bmatrix} z_1', z_2', \cdots, z_S' \\ p_1^{z'}, p_2^{z'}, \cdots, p_S^{z'} \end{bmatrix}, S<QR \tag{3.11}$$

相關的執行步驟如下：

1. 將 Z 變數數值分割成 S 個區間，區間的間隔為 $\Delta = (z_w - z_1)/S$。每一區間含原 Z 變數之數值單元為

2. $$D_k = \{z_d \mid z_d \in (z_1 + (k-1)\Delta, z_1 + k\Delta], \ k = 1,...,S\} \tag{3.12}$$

3. 每一區間以新的數值單元 $(z_k', \ p_k^{z'}), \ k = 1,...,S$ 來表示，其中

4. $$z_k' = \sum_{D_k} z_d \cdot p_d^Z \Big/ \sum_{D_k} p_d^Z, \quad p_{\mathrm{k}}^{Z'} = \sum_{D_k} p_{\mathrm{d}}^{\mathrm{Z}} \tag{3.13}$$

5. 若區間沒有存在樣本點，則該區間所產生之新的單元數值，可表示為

$$(z_k', \ p_k^{z'}) = (z_1 + (k - 0.5)\Delta, \ 0) \tag{3.14}$$

當以離散化技術執行 convolution 及 max 運算，其結果之單元數值數目若超過 S 個，則執行重新取樣技術**(**Discrete Resample Technique, DRT**)**，如此就能夠有效的控制單元數值數量過大的問題。

本章節以附錄 B 實例所演算出來的結果 γ_4 為對象，取離散單元為 7，實際執行重新取樣，其結果如表 3.2 及 3.3 所列。証明重新取樣後的離散資料，平均值沒有改變，僅變異數略為減小。

表 3.2　γ_4 之離散資料

γ_4	3	4	5	6	7	8	9
$\Pr(\gamma_4)$	0.0001	0.0006	0.0003	0.0023	0.0105	0.0103	0.0388
γ_4	10	11	12	13	14	15	17
$\Pr(\gamma_4)$	0.1667	0.2052	0.0926	0.2330	0.0748	0.0765	0.0833

$\gamma_4 = \max\{X_{14}, Y_2^4, Y_3^4\}$: Mean(γ_4)=**12.20,**　Std.(γ_4)=**2.24**

表 3.3　γ_4 重新取樣後之資料

$DRT(\gamma_4)$	3.86	5.88	7.50	9.81	11.31	13.24	16.04
$\Pr(DRT(\gamma_4))$	0.0007	0.0026	0.0208	0.2055	0.2978	0.3078	0.1598

DRT(γ_4): Mean($DRT(\gamma_4)$)=**12.20,**　Std.($DRT(\gamma_4)$)=**2.17**

3.3 路徑相依對網路完成時間求解的影響

在專案分析的工具中，PERT 模式雖廣受歡迎，但 PERT 模式須在若干假設條件下才能成立，其中之一，即是假設專案完成路徑均互相獨立(Independent)，互不影響。就實務面而言，若專案其中兩路徑使用到共同的節點作業，在此種情況下兩路徑便不再視為互相獨立，故上述的假設便不能成立。Dodin(1985b)估算專案完成時間的演算法中，亦未將路徑相依性因素納入考慮。本章節擬詳細說明路徑相依對專案網路完成時間估算之影響，發生的原因探討，並提出解決之方法。

3.3.1 路徑相依性造成專案網路估算之偏差

專案網路路徑的相依性是源於路徑間使用共同之作業，以圖 3.1 所例，路徑(1-2-4)與(1-2-3-4)有重複的節點{1,2}，路徑(1-3-4)與(1-2-3-4)之重複的節點為{1,3}。這些重複的節點會造成 DA 演算法產生偏差。這偏差形成之原因乃源於節點 6 所執行之 max 運算結果，其估算隨機性專案網路完成時間往往會高估實際值。本章節另以圖 3.4 為範例:

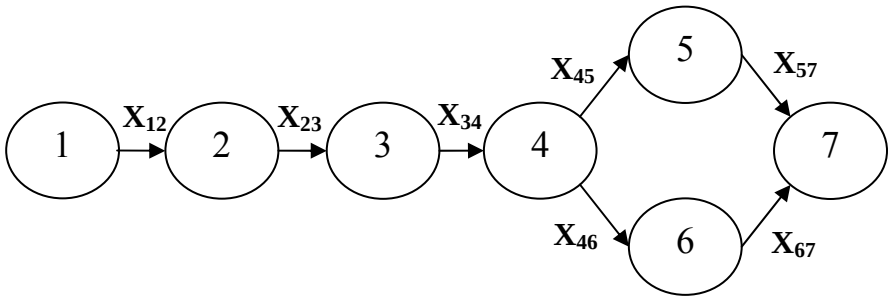


圖 3.4 . 簡單專案網路結構

使用 DA 演算法所得到的估算結果，與 20000 次取樣之蒙地卡羅模擬結果

作一比較，實際來瞭解其偏差產生之原因。

圖 3.4 的專案共計 2 條完成路徑(1-2-3-4-5-7)及(1-2-3-4-6-7)，該 2 路徑造成相依的節點集合為{1,2,3,4}，然而 DA 演算法忽略該相依節點之影響，將路徑(1-2-3-4-6)及(1-2-3-5-6)視為相互獨立，因此其計算程序如下：

$$
\begin{aligned}
&1.\ \gamma_3 = X_{12} \oplus X_{23} \\
&2.\ \gamma_4 = \gamma_3 \oplus X_{34} \\
&3.\ \gamma_5 = \gamma_4 \oplus X_{45} \\
&4.\ \gamma_6 = \gamma_4 \oplus X_{46} \\
&5.\ Y_1^7 = \gamma_5 \oplus X_{57} \\
&6.\ Y_2^7 = \gamma_6 \oplus X_{67} \\
&7.\ \gamma_7 = \max(Y_1^7,\ Y_2^7)
\end{aligned}
\tag{3.15}
$$

若令所有的作業時間均為指數分佈，其相關之作業時間平均值依序為 $\mu(X_{12})$=1，$\mu(X_{23})$=3，$\mu(X_{34})$=5，$\mu(X_{45})$=2，$\mu(X_{46})$=2，$\mu(X_{57})$=3 及 $\mu(X_{67})$=3。DA 估算所得的最後結果為 γ_7，其時間平均值為 17.69。若使用 20000 次取樣蒙地卡羅模擬所得的結果為 15.89，其間的誤差來源便是上述演算法將兩條完成路徑未考慮相依的作業節點所致。圖 3.4 中節點 6 執行 max 運算因未考慮到相依路徑的影響致使所得到的平均值會比蒙地卡羅模擬法來得大，且標準差亦會因離散技術運算執行 max 運算之故，會比蒙地卡羅模擬法來得小，相關的結果由圖 3.5 表示之。

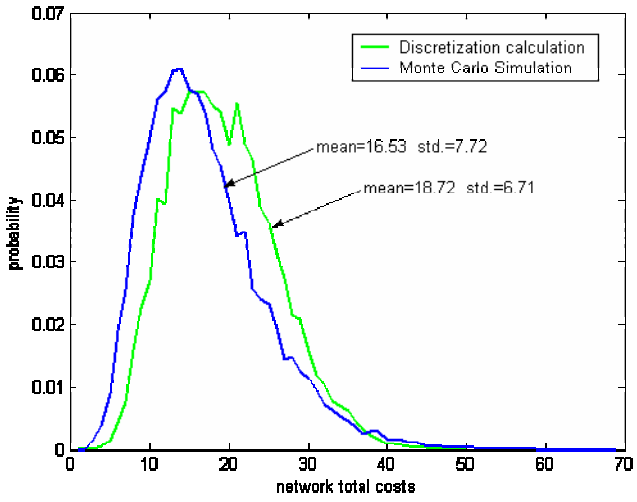


圖 3.5　未考慮到相依路徑與 MCS 結果之比較

3.3.2 最長路徑偏差(Longest Path Bias, LPB)

max 運算會受輸入路徑之時間機率分佈範圍(即變異數值)大小之影響，此即為「最長路徑偏差(Longest Path Bias, LPB)」。以圖 3.1 為例，節點 4 共有三條輸入路徑: (i) (1-2-4), $E(X_{12})+E(X_{24})$ (ii) (1-4), X_{14} 及 (1-3-4), $E(X_{12})+E(X_{23})+E(X_{34})$ 以 PERT 模式計算，三條輸入路徑之平均時間分別為 $E(X_{12})+E(X_{24})$, $E[X_{14}]$ 及 $E(X_{12})+E(X_{23})+E(X_{34})$，取其最長路徑時間，節點 4 之輸出時間平均值及變異數為 T_{PERT}, V_{PERT}:

$$T_{PERT}=\max(E(X_{12})+E(X_{24}),\ E(X_{1,4}),\ E(X_{12})+E(X_{23})+E(X_{34})) \tag{3.16}$$

$$\begin{aligned} V_{PERT}&=Var(X_{12})+Var(X_{23})+Var(X_{34}),\quad if\ T_{PERT}=E(X_{12})+E(X_{23})+E(X_{34})\\ V_{PERT}&=Var(X_{12})+Var(X_{24}),\quad if\ T_{PERT}=E(X_{12})+E(X_{24})\\ V_{PERT}&=Var(X_{14}),\quad if\ T_{PERT}=E(X_{14}) \end{aligned} \tag{3.17}$$

而真實的節點 4 輸出結果平均值及變異數為 T_{real}, V_{real}:

$$T_{real}=\mathrm{E}[\max(E(X_{12})+E(X_{24}),X_{14},E(X_{12})+E(X_{23})+E(X_{34}))] \tag{3.18}$$

$$V_{real}=\mathrm{Var}[\max(E(X_{12})+E(X_{24}),X_{14},E(X_{12})+E(X_{23})+E(X_{34}))] \tag{3.19}$$

一般而言，機率變數實際執行 max 所得的結果平均值會大於 PERT 的結果，即 $T_{real}\geq T_{PERT}$，變異數亦會有所差距。以附錄 B 的數據及圖 3.1 為例，實際執行式(3.16)~式(3.19)，其結果如下:

$$\begin{aligned} T_{PERT}&=\max(E(X_{12})+E(X_{24}),\ E(X_{1,4}),\ E(E(X_{12})+E(X_{23})+E(X_{34}))\\ &=\max(7.83,\ 7.5,\ 10.16)\\ &=10.16 \end{aligned} \tag{3.20}$$

$$V_{PERT}=\mathrm{var}(X_{12})+\mathrm{var}(X_{23})+\mathrm{var}(X_{34})=8.91$$

$$\begin{aligned} T_{real}&=E(\gamma_4)=E(\max\{X_{14},Y_2(4),Y_3(4)\})\\ &=12.20 \end{aligned} \tag{3.21}$$

$$V_{real}=5.20$$

結果證明 $T_{real}\geq T_{PERT}$，變異數亦有差距。

統計學中「順序統計理論」(Order Statistics Theorem, 1981)及 Fisher-Tippet Theorem(1928)對 LPB 的形成有理論上的說明。另外 Klingel(1966)，Dodin & Sirvanci (1990)，Mehrotra, et al.(1996)及 Pontrandolfo(2000)執行專案完成時間的估算亦考慮到 LPB 的現象，其中

Klingel 指出若當輸入兩路徑之完成時間機率分配值域相接近或其變異量較大的話，則 LPB 的偏移現象會愈明顯。

3.3.3 路徑相依性造成偏差之原因

(3.15)程序執行 DA 演算法的過程中，並分別記錄節點 2, 3, 4, 5, 6 及 7 的輸出平均值及標準差，並同時與蒙地卡羅模擬法所得到的結果做一比較，並紀錄於表 3.2 中。從表中很明顯得知，DA 演算法的差異主要源於節點 7 執行 max 運算之結果。鑒於 3.3.2 節所述之 LPB 現象，DA 估算所產生的誤差即與路徑時間之變異數有關。依據圖 3.4 所示，DA 視路徑(1-2-3-4-5-7)及(1-2-3-4-6-7)為兩獨立之路徑，即該兩路徑之完成時間之變異數為該路徑上所有作業時間變異數之加總。然而子路徑(1-2-3-4)為該兩路徑共同持有，該子路徑時間之變異數應僅能對節點 7 執行 max 運算影響一次。然而 DA 的演算方式會將該子路徑的變異數影響，重複一次的加諸在節點 7 的 max 運算上，使得 max 運算的結果多了一次子路徑(1-2-3-4)變異數之影響，由於 LPB 偏移量與輸入變數之變異數值有關，因此 DA 的估算結果往往會比實際值來得大。

3.3.4 路徑相依因素納入估算之方法

隨機性專案網路完成時間估算若考慮路徑相依的情況，則須將 3.3.3 節中，子路徑(1-2-3-4)變異數所造成多餘之 LPB 偏移量移出即可。故圖 3.4 應改以圖 3.6 的結構視之，其中 $X_{12}=X_{1'2'}, X_{23}=X_{2'3'}, X_{34}=X_{3'4'}$ 。

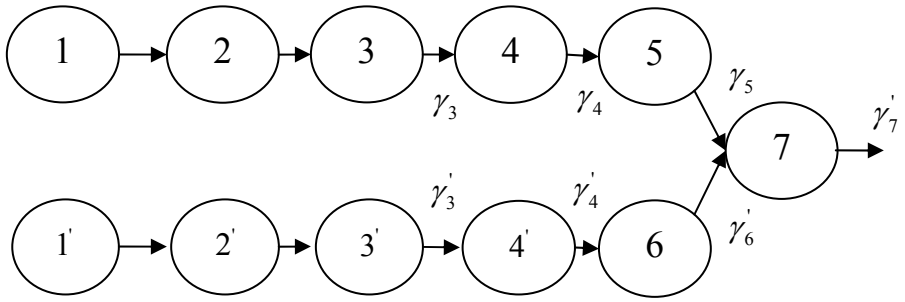


圖 3.6　圖 3.4 之節點樹狀結構

且(3.15)式之計算程序應修改如下：

$$
\begin{aligned}
&1.\ X_{12} = X_{1'2'} \\
&2.\ X_{23} = X_{2'3'} \\
&3.\ X_{34} = X_{3'4'} \\
&4.\ \gamma_{3'} = X_{1'2'} \oplus X_{2'3'} \\
&5.\ \gamma_{4'} = E[\gamma_{3'} \oplus X_{3'4}] \\
&6.\ \gamma_{6'} = \gamma_{6'} \oplus X_{46} \\
&7.\ Y_1^7 = X_{12} \oplus X_{23} \oplus X_{34} \oplus X_{45} \oplus X_{57} \\
&8.\ Y_{2'}^7 = \gamma_{6'} \oplus X_{67} \\
&7.\ \gamma_{7'} = \max(Y_1^7,\ Y_{2'}^7)
\end{aligned}
\tag{3.22}
$$

(3.15)式與原(3.22)式之計算程序的主要差別在於$\gamma_{4'} = E[\gamma_{3'} \oplus X_{3'4'}]$而非$\gamma_4 = \gamma_3 \oplus X_{34}$，因此節點 7 的 max 運算所產生的 γ_7'會與原γ_7兩結果不一樣因為γ_4是隨機變數，而$\gamma_{4'}$是γ_4的平均值變異數為 0，因此節點 7 的第二條路徑$Y_{2'}^7$之子路徑(1-2-3-4)變異數，便不會對節點 7 的 max 產生額外之 LPB 偏差。表 3.4 為執行 DA 演算法及考量路徑相依性之演算法，在圖 3.4 各節點所得的結果與 MCS 結果之比較，很明顯的，考量路徑相依性之演算法在節點 7 之輸出為 15.57，較為接近 MCS 之結果 15.89。本文另以信心水準 α=0.05 執行 K-S 檢定，DA 演算法所造成的誤差無法通過檢定，而考量路徑相依性之演算法則順利通過。

表 3.4 路徑相依性考量與否的估算比較

網路節點	MCS		DA 演算法			考量路徑相依性		
	平均值	標準差	平均值 (誤差)	標準差 (誤差)	K-S 檢定 $\alpha = 0.05$	平均值 (誤差)	標準差 (誤差)	K-S 檢定 $\alpha = 0.05$
3	3.98	3.16	4.00 (0.50%)	3.16 (0.00%)	通過	4.00 (0.50%)	3.16 (0.00%)	通過
4	9.07	5.97	9.00 (0.77%)	5.92 (0.84%)	通過	9.00 (0.77%)	5.92 (0.84%)	通過
5	11.05	6.98	11.05 (0.45%)	6.98 (0.72%)	通過	11.05 (0.45%)	6.98 (0.72%)	通過
7	15.89	7.01	**17.69 (11.3%)**	6.91 (1.43%)	沒通過	**15.57 (2.01%)**	6.89 (1.71)	通過

估算隨機性專案完成時間必須將所有的路徑相依情況納入考慮，首先將專案網路結構各完成路徑間重複的路徑節點找出。針對專案網路中執行 max 運算的節點，若其輸入路徑中存有重複路徑，則只允許其中一條路徑完成時間以估算出之機率分佈函數表示，其餘重複之路徑必須以該機率分佈函數之平均值取代之，以去除重複路徑所產生的不必要的變異量。這種作法可有效的防止執行 max 運算，產生多餘的 LPB 偏移量。再以圖 3.4 及圖 3.6 為例，圖 3.6 為圖 3.4 之擴張結構，重疊路徑為($1'$-$2'$-$3'$-$4'$)，並以 μ 值表示時間機率函數 $\mu = E[X_{1'2'} \oplus X_{2'3'} \oplus X_{3'4'}]$，節點 4 的輸出 γ_4' 離散式可表示為：

$$\gamma_4' = \begin{bmatrix} \mu \\ 1 \end{bmatrix} \tag{3.23}$$

將 γ_4' 代入(3.22)中計算可以得到修正後的 $Y_{2'}^7$ 及 γ_7' 的結果。修正後所求出的 $Y_{2'}^7$ 並不會影響原該路徑的平均值，即 $mean(Y_2^7) = mean(Y_{2'}^7)$，本文擬以下列計算式子證明之。

令路徑 $Y_2^7 = A \oplus B$，其中 A 為重複路徑節點的隨機時間變數(即圖 3.6 中之 γ_4')，B 則為不重複部份路徑節點隨機時間變數(即圖 3.6 中之 $X_{45} \oplus X_{57}$)，為求簡單起見，令 A, B 的離散表示式如下：

$$\begin{aligned} A &= \begin{bmatrix} a_1 & a_2 \\ p_{1a} & p_{2a} \end{bmatrix} \\ B &= \begin{bmatrix} b_1 & b_2 \\ p_{1b} & p_{2b} \end{bmatrix} \\ \gamma_4' &= \begin{bmatrix} \mu \\ 1 \end{bmatrix}, \quad \text{其中 } \mu = p_{1a}a_1 + p_{2a}a_2 \end{aligned} \tag{3.24}$$

修正的 $Y_{2'}^7$ 可以計算如下：

$$Y_{2'}^7 = \gamma_4' \oplus B = \begin{bmatrix} \mu \\ 1 \end{bmatrix} \oplus \begin{bmatrix} b_1 & b_2 \\ p_{1b} & p_{2b} \end{bmatrix} = \begin{bmatrix} \mu + b_1 & \mu + b_2 \\ p_{1b} & p_{2b} \end{bmatrix} \tag{3.25}$$

原 Y_2^7 可以計算如下：

$$Y_2^7 = A \oplus B = \begin{bmatrix} a_1 & a_2 \\ p_{1a} & p_{2a} \end{bmatrix} \oplus \begin{bmatrix} b_1 & b_2 \\ p_{1b} & p_{2b} \end{bmatrix} = \begin{bmatrix} a_1 + b_1 & a_1 + b_2 & a_2 + b_1 & a_2 + b_2 \\ p_{1a}p_{1b} & p_{1a}p_{2b} & p_{2a}p_{1b} & p_{2a}p_{2b} \end{bmatrix}$$

(3.26)

求出(3.26)式的平均值，再以 $\mu = p_{1a}a_1 + p_{2a}a_2$ 化簡之：

$$\begin{aligned} mean(Y_2^7) &= (a_1+b_1)p_{1a}p_{1b} + (a_1+b_2)p_{1a}p_{2b} + (a_2+b_1)p_{2a}p_{1b} + (a_2+b_2)p_{2a}p_{2b} \\ &= (p_{1a}a_1 + p_{2a}a_2)p_{1b} + (p_{1a}a_1 + p_{2a}a_2)p_{2b} + (p_{1a}+p_{2a})b_1p_{1b} + (p_{1a}+p_{2a})b_2p_{2b} \\ &= ((p_{1a}a_1 + p_{2a}a_2)+b_1)p_{1b} + ((p_{1a}a_1 + p_{2a}a_2)+b_2)p_{2b} \\ &= (\mu+b_1)p_{1b} + (\mu+b_2)p_{2b} \\ &= mean(Y_2^{'}(6)) \end{aligned}$$

(3.27)

3.4 標籤修訂追蹤演算法

本章節擬介紹本論文所發展之演算法，來估算隨機性專案網路完成時間，本論文並將其稱之為標籤修訂追蹤演算法(Label-Correcting Tracing Algorithm, LCTA)。LCTA 除運用了 3.3 節中所有與離散式技術有關之運算外，主要是依據該章節中所提，將專案網路之路徑相依性納入演算的考慮，以降低估算過程中所產生的誤差。DA 演算法之缺點，除未將路徑相依性因素納入外，其另一項為未能提出一具規則或有系統的演算程序來執行專案網路完成時間之估算。因此 DA 演算法不能保證其運用在大型專案網路上之可行性，這也是本論文所發展之 LCTA 最重要的訴求之一。LCTA 須先將專案網路轉換成一樹狀結構，該結構內之節點代表專案網路中之節點，專案網路節點與節點間之作業時間與節點間接續狀態，均會存入特別的資料結構(Data structure)中。這些資料結構內容透過本論文所設計出之追蹤演算程序，無論專案網路有多複雜、多龐大，LCTA 均能夠有系統的將專案網路完成時間累算出來。

3.4.1 擴張樹結構

DA 演算法必須先使用序列/平行縮減演算程序(Series/Parallel reduction)進行專案網路之縮減，其目的是為了要降低專案網路結構的複雜度，以減輕計算的負擔。然而根據文獻探討 2.3.1 節所述，網路縮減演算法本身即為一複雜的演算法，Colby(1984)更指出其為一 NP hard 問題型態。本論文擬擺脫該複雜的演算步驟，利用專案網路節點間接續之前後關係，直接將專案網路專換成一平面式的樹狀結構，本論文將其稱之為擴張樹結構

(Expanded Tree Structure, ETS)。ETS 結構內之節點代表專案網路中之節點，其特色除能夠顯示各節點間之接續關係，另能夠清楚的表現出專案節點間階層(Layer)的關係，這對專案網路完成時間的分析是有幫助的。

執行 ETS 轉換前，須先將專案網路每一節點予以編號，其中 1 號固定為專案網路起始節點，最後號碼 N 則定為專案網路之終端節點。ETS 對專案網路另有兩項要求：(1)網路的起始節點及終端節點只能有一個。(2) 起始節點至終端節點之編號不得有斷號，即每一號碼均須有對應之節點存在。ETS 的轉換步驟並不困難，首先從專案網路結構圖中取得了各節點間先後順序的資料，再以終節點 N 為樹狀結構之根節點(Root node)，開始將其後接續之節點以樹狀結構展開，再以展開後之節點為根節點，同上述的步驟持續執行，直到起始節點 1 出現在展開之節點為止。以圖 3.1 專案網路為例，其擴張樹結構如圖 3.7 所示。

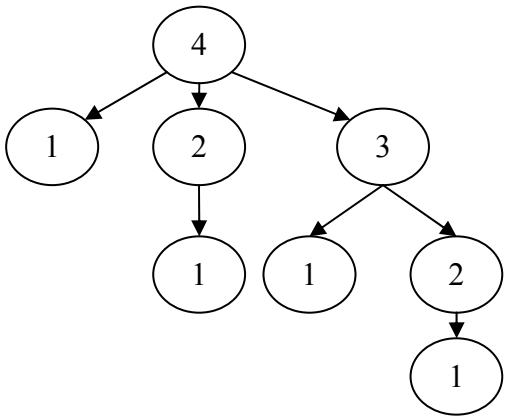


圖 3.7 在圖 3.1 之專案網路之擴張樹結構

3.4.2 擴張樹節點資料結構

ETS 在納入 LCTA 演算前，須將專案網路中所擁有的訊息紀錄在 ETS 結構中，訊息包括專案網路節點的接續關係及節點間之作業時間變數。因此 ETS 的各節點均以一特別之資料結構，紀錄、儲存及計算專案網路上述之訊息及 LCTA 演算過程中所產生之資料。在 ETS 中對任一節點 i，其資料結構的內容包括：

1. 節點 i 本身之編號。

2. 節點 i 之輸入節點集合，將其定義為 $\text{Prec\_Node}(i) = \{j\},\ j \in B_i$。

3. 完成節點運算指標 $finish\_flag_i$ 為一布林代數(Boolean variable)，初始值

$finish\_flag_i = 0$，當 LCTA 完成節點 i 之時間運算 γ_i 後，$finish\_flag_i = 1$。

4. 節點 i 之輸出時間機率分佈 $Output_i = \gamma_i$，當 LCTA 完成節點 i 之時間運算後，γ_i 即存入此處。該值為 2 維度(Dimension)之離散化資料，其中一維度是時間樣本點值，另一維度是取樣點值。
5. 節點 i 輸入路徑完成指標值集合，將其定義為 $\text{Path\_Flag}(i) = \{flag_j\},\ j \in B_i$，初始值所有 $flag_j = 0$，表示所有輸入路徑 $Y_j^i,\ j \in B_i$ 均尚未完成計算，當完成了其中輸入路徑 j 之 Y_j^i，則 $flag_j = 1$。
6. 節點 i 輸入路徑完成時間集合，將其定義為 $\text{Path\_Time}(i) = \{Y_j^i\},\ j \in B_i$，節點 i 將第 5 項所計算出之 Y_j^i 存入此處。
7. 節點 i 與輸入節點 $j,\ j \in B_i$ 間之作業時間集合，將其定義為 $\text{Path\_Act}(i) = \{X_{ij}\},\ j \in B_i$，所有作業時間 X_{ij} 為 2 維度離散化資料，專案網路作業時間均紀錄於此。

上述 ETS 資料結構 7 種內容，以節點 i 為例可表示如表 3.5 所列:

表 3.5　ETS 節點 i 資料結構

node	Prec_Node(i)	$finish\_flag_i$	$Output_i$
Path_Flag(i)	Path_Time(i)	Path_Act(i)	-

若以圖 3.7 之 ETS 結構為例，令初始的資料均依上述所言，且所有的作業時間均假設為平均值 mean=2 之指數分佈函數，並採柴比契夫離散化方法處理該作業時間機率分佈函數，ETS 資料結構可表示為表 3.6 所示:

3.4.3 擴張樹之追蹤程序

在本節擬介紹 LCTA 執行的兩個追蹤程序，該兩個程序能夠遵循一個特別的順序，拜訪 ETS 結構中的每一個節點。在拜訪的過程中配合 ETS 資料結構內容的參考及計算，便可有規律的將隨機性專案網路時間估算出來。

LCTA 追蹤程序類似在確定性網路環境(Deterministic network)中，使用標籤修訂演算法(Label-Correcting approach, LC)來尋找最短路徑問題

表 3.6　圖 3.7 之 ETS 初始的資料結構

node	1	2	3	4
$finish\_flag_i$	0	0	0	0
Prec_Node(i)	$\varnothing$	{1}	{1,2}	{1,2,3}
$Output_i$	$\begin{bmatrix}0\\1\end{bmatrix}$	$\begin{bmatrix}0\\1\end{bmatrix}$	$\begin{bmatrix}0\\1\end{bmatrix}$	$\begin{bmatrix}0\\1\end{bmatrix}$
Path_Flag(i)	{0}	{0}	{0, 0}	{0, 0, 0}
Path_Time(i)	$\{\begin{bmatrix}0\\1\end{bmatrix}\}$	$\{\begin{bmatrix}0\\1\end{bmatrix}\}$	$\{\begin{bmatrix}0\\1\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}\}$	$\{\begin{bmatrix}0\\1\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}\}$
Path_Act(i)	{A}	{A}	{A, A}	{A, A, A}

其中 $A=\begin{bmatrix}0.49 & 4.12 & 10.0 & 15.88 & 19.51\\ 0.56 & 0.41 & 0 & 0.03 & 0\end{bmatrix}$

(Shorest Path problem, SP)。LC 也是將每一個專案網路之節點以一標籤號碼來取代，這些標籤會被置入一佇列(Queue)中，LC 會掃描這佇列，取出欲處理的標籤，當這標籤被處理後，佇列會被置換另一適當的標籤，或將佇列中的標籤做適當的位置調整，LC 重複此動作直到佇列的標籤淨空，這也表示所有的標籤均已被適當的處理，最短路徑也就可以找到。佇列標籤的置換及取用的原則方式，全賴 LC 之策略而定，因此在文獻中有多種 LC 方法，其中包括 Bellman(1958)使用先進先出(First in First out)策略，Pape(1974)使用 d-queue 策略，Gallo and Pallottino(1988)使用了雙佇列策略(Two queue method)，Glover, et al.(1986)則使用了位準策略(Threhold method)，近年來 Bertsekas(1993)則使用了小標籤優先策略(Small-label-first principle)來改良 Glover, et al.及 Bertsekas 的方法。本論文所發展之 LCTA 所不同於 LC 有三處:

1. LCTA 所處理的專案網路乃針對隨機性的網路環境，不同於 LC 之確定性網路環境。
2. LCTA 對專案網路所關注的問題是最長路徑，非 LC 之最短路徑問題。
3. LCTA 使用堆疊(Stack)來作為標籤存放的工具，與 LC 所使用的佇列不同，因此 LC 所使用的策略與 LCTA 追蹤過程所使用的規則完全不同。

標籤(Label)在追蹤的過程中表示 LCTA 正在處理的節點為何，LCTA 可以透過 ETS 資料結構Prec_Node(*i*)的內容獲知該標籤之子節點，因此 LCTA 可得以接續追蹤下一個處理的標籤。如圖 3.8 為例，節點 2, 3, 4 為節點 1 之子節點，若處理中之標籤值為 1，LCTA 很容易從 ETS 資料結構中得知節點 2, 3, 4 是下一個待處理的標籤。然而處理中的標籤，卻無法透過 ETS 資料結構得知其父節點其它之子節點，從圖 3.8 中，若處理中之標籤為 2，其無從得知節點 3 及 4 的存在，必須再回到父節點 1 處才能得知。因此 LCTA 必須要透過一堆疊結構，存放處理過的標籤，當標籤 1 處理完畢移至標籤 2 後，標籤 1 被存放入堆疊中；當標籤 2 處理完畢欲移至標籤 3 處，LCTA 須先將堆疊中之標籤 1 取出，再將標籤 2 存放堆疊中，如此方能持續處理標籤 3 之節點，依此原則標籤 4 亦順利找到處理。堆疊存放的原則為先進先出(First in First out)，因此存取的順序不會錯亂，圖 3.8 之標籤處理的順序，最後會依(1→2→3→4)順序完成。

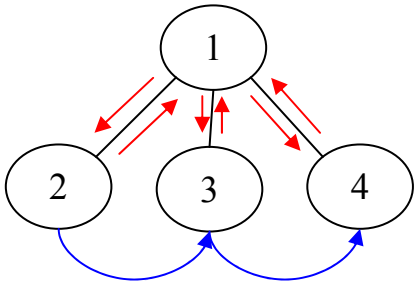


圖 3.8　一個簡單樹狀結構

從圖 3.8 之範例說明，已經很清楚的描繪出 LCTA 之追蹤原則，即樹狀結構中之後序追蹤(Post-Order Tracing)原則，該原則本文將其整理如下:

1. 從現行節點(Current node)開始執行追蹤，並將其標籤數紀錄於堆疊中。
2. 拜訪現行節點之子節點，拜訪之順序由左至右，所拜訪的子節點亦將其標籤數記錄於堆疊中。當完成一子節點拜訪後，必須先將堆疊最上層的標籤取出，回到父節點之位置，再依由左至右順序拜訪下一子節點。重複此步驟直到所有子節點完成拜訪後，回到其父節點的位置。
3. 在步驟 2 執行的過程中，若所拜訪的子節點不是起始節點(即節點 1)，則將該子節點取代為現行節點，重複執行步驟 1 及 2；若為起始節點 1，則

持續步驟 2 的程序。

4. 追蹤程序最終會回到最初始的現行節點位置，並停止追蹤程序。

從上述後序追蹤原則中可以清楚的瞭解，堆疊在 LCTA 中扮演非常重要的角色，其最上層的節點標籤即表示剛結束拜訪的節點位置。若以圖 3.7 為範例執行後序追蹤程序，並從根節點 4 開始執行之，其拜訪之順序表示於圖 3.9 中，拜訪的過程中須將所有經過的節點標籤紀錄於堆疊中，另有一堆疊指標負責指示出標籤存放之位置，本節後述之 LCTA 的兩個追蹤程序擬以 *push_stack*(*node*)及 *pull_stack*()指令表示堆疊存與取標籤的動作。圖 3.9 中會出現重覆拜訪的節點(節點 1 及 2)，這問題可透過 LCTA 追蹤程序中所執行的計算予以避免排除之，如此可以節省 LCTA 估算的時間。

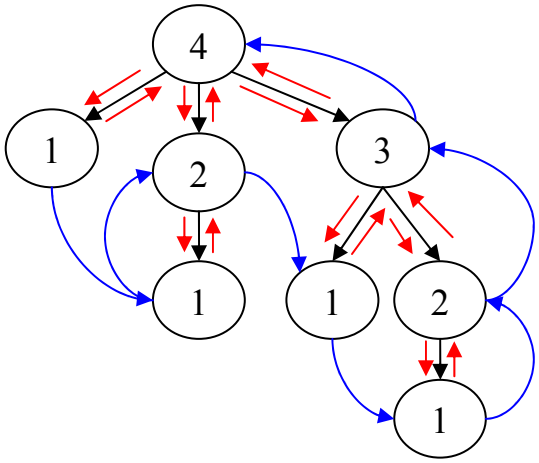


圖 3.9　圖 3.7 後序追蹤順序

LCTA 結合前述之 ETS、ETS 資料結構及後序追蹤程序三種要項，僅完成了 LCTA 之主架構，雖該主架構能讓 LCTA 之追蹤演算，從 ETS 之根節點做起使點，拜訪 ETS 中所有的節點，但是還缺乏節點資料之分析計算。若要能夠執行隨機性專案網路完成時間機率分佈之估算，還必須要賦予 LCTA 分析計算能力。因此 LCTA 必須對每個所拜訪節點，額外執行下列的判斷及計算：(以現行節點 j 為例)

1. 判斷現行節點 j 是否所有的子節點均已拜訪完畢？

(1) 全部追蹤完畢：執行 $\gamma_j = \max\{Y_i^j\},\ i \in B_j$ 的運算，並設定 $finish\_flag_j = 1$，即專案網路起始點 *1* 至節點 j 的完成時間機率分佈估

算，執行 node=*pull_stack*()，再回到節點 *j* 之父節點位置。

(2)尚未完成所有子節點的追蹤：執行 $Y_i^j = \gamma_i \oplus X_{ij}$，即計算節點 *i* 到節點 *j* 的路徑完成時間的機率分佈，若 Y_i^j 的離散樣本單元超過所規定之位準值(本論文定為 100)，則額外須執行重新取樣運算(DRT)(參考 3.2.4 節內容)，再依後序追蹤原則持續進行後序追蹤程序，尋找下一個尚未完成追蹤的子節點。

2. 判斷節點 *j* 是否為專案網路的起始節點？

(1)是起始節點：則計算 γ_1，並設定 $finish\_flag_1 = 1$，因追蹤已觸 ETS 結構之底層，故執行 node=*pull_stack*()，尋求起始節點的父節點，並依後序追蹤原則持續進行後序追蹤程序。

(2)不是起始節點：依後序追蹤原則持續進行後序追蹤程序。

3. 利用 *Shared_flag* 旗標來表示 ETS 內的節點是否重複出現，第一次拜訪的節點 *Shared_flag*=0，重複出現的節點 *Shared_flag*=1。

LCTA 將後序追蹤演算程序及上述兩項計算內容整合，發展出下列兩個演算程序，執行專案完成時間之估算。

1. Upward_ Tracing (last_node, current_node)
2. Downward_ Tracing (current_node)

current_node 是指追蹤程序正在處理的節點標籤，last_node 則是上次處理完畢的節點標籤。Downward_Procedure 在 ETS 結構中往下執行追蹤，並負責上述第二項的判斷及計算工作；Upward_Procedure 則在 ETS 結構中往上執行追蹤，所負責的是第一項的判斷及計算工作。詳細的追蹤程序表列如下:

追蹤程序：Upward_Tracing(k,j) // *k*= last_node, *j*= current_node
while (γ_N 尚未計算完成) **do**
  if (last_node *k* 是否在追蹤的過程中重複出現過?)
    $Y_k^j = \mu(\gamma_k) \oplus X_{kj}$;
    Shared_flag=0；
  Else
    $Y_k^j = \gamma_k \oplus X_{kj}$;
    Shared_flag=0；

End

if (Y_k^j 的樣本單元大於 100?)

$DRT(Y_k^j)$；　　// 執行重新取樣計算工作

End

if ($finish\_flag_j = 1$)　　// 是否 γ_j 已完成計算?

push_stack(*j*)；　　// 將 current_node *j* 至入堆疊中

找出 current_node *j* 另一個的子節點 *i*, $i \in \mathrm{B}_j$，並令 current_node *j*=*i*；

執行 **Downward_Tracing (*j*)**；

Else

$\gamma_j = \max(Y_k^j), \forall k \in B_j$ ；

$finish\_flag_j = 1$；

Shared_flag=0；　　// 節點 *j* 第一次拜訪, *Shared_flag* 要重設

k=*j*;　　// last_node *k* = current_node *j*

j= *pull_stack*()；　　// 從堆疊中取出節點標籤並令為 current_node

End

End

追蹤程序：**Downward_Tracing (*j*)**　　// *j* = current node

while (γ_N 尚未計算完成) **do**

if ($finish\_flag_j = 1$)　　// 是否 γ_j 已完成計算?

Shared_flag=1；

k=*j*;　　// last_node *k* = current_node *j*

j= *pull_stack*()；　　// 從堆疊中取出節點標籤並令為 current_node

執行 **Upward_Tracing(*k*,*j*)**；

Else

if (current_node *j*=1?)　// 追蹤是否碰觸起始節點?

$\gamma_1 = 0$ ；

$finish\_flag_j = 1$；

k=*j*;　　// last_node *k* = current_node *j*

j= *pull_stack*()；　　// 從堆疊中取出節點標籤並令為 current_node

執行 **Upward_Tracing(*k*,*j*)**；

Else

push_stack(*j*)；　　// 將 current_node *j* 至入堆疊中

找出 current_node j 下一個的子節點 i, $i \in \mathrm{B}_j$，並令 current_node $j=i$；

End

End

End

Upward_Tracing(k,j)及 Downward_Tracing(j)兩追蹤程序在執行的過程中，會有許多的資料跟著計算結果而變動，相關的變動資料亦會紀錄在 ETS 的資料結構或參數中。本節將其整理如下:

1. current_node j：追蹤程序正在處理的節點標籤。
2. last_node k：追蹤程序前一個處理完畢的節點標籤。
3. *Shared_flag*：追蹤程序界此旗標，判斷 current_node j 是否為重複出現的節點標籤。
4. Stack_pointer：只是節點標籤存放位置之堆疊指標。
5. Stack：存放節點標籤之堆疊內容。
6. ETS 資料結構內容，包括:
   (1) $finish\_flag_j$：ETS 節點完成節點路徑時間 γ_j 計算之設定旗標。
   (2) $Output_j$：紀錄 γ_j 數值之存放處。
   (3) Path_Flag(j)：ETS 節點輸入路徑時間 $Y_i^j, i \in B_j$ 計算完成設定旗標。
   (4) Path_Time(j)：ETS 節點輸入路徑時間 $Y_i^j, i \in B_j$ 數值之存放處。

LCTA 從 ETS 的根節點處(即專案網路之輸出節點)開始執行 Downward_Tracing(j)程序，因採後序追蹤原則，樹狀結構節點拜訪的順序是由左至右，因此追蹤程序開始會沿 ETS 結構左側邊一直下探，在下探的過程中，執行 *push_stack*(j)指令，將所經過的節點紀錄在堆疊中，以便紀錄追蹤之路徑。當追蹤程序碰觸起始節點 1，便開始啟動 Upward_Tracing(k, j)追蹤程序，追蹤開始沿著所紀錄的追蹤路徑往上爬升，執行 *pull_stack*()指令尋求上一層父節點之標籤，並開始執行路徑時間累加計算，計算的結果也會紀錄在上述之 ETS 資料結構中。專案網路的時間計算主要由 Upward_Tracing(k, j)負責，Downward_Tracing(j)僅執行起始節點 $\gamma_1 = 0$ 之設定。在 Upward_Tracing(k, j)中，時間計算又可分成兩部分：

1. 當追蹤由節點 k 爬至節點 j 時，逕執行 $Y_k^j = \gamma_k \oplus X_{kj}$，完成節點 j 之 k 輸入路徑之時間累加，Y_k^j 會存入 ETS 資料結構 Path_Time(j) 中，Path_Flag(j) 亦會執行相關之設定。當執行完 Y_k^j 後，程式會檢查 Y_k^j 的樣本單元是否超過 100 個，若有的話，須依據 3.2.4 節所述之方法，再執行 $DRT(Y_k^j)$ 重新取樣計算，以控制樣本單元的數量。

2. 當執行完第 1 項 Y_k^j 計算後，Upward_Tracing(k, j) 會針對節點 j 之 $finish\_flag_j$ 旗標來做判斷，判斷是否 $finish\_flag_j = 1$? 若答案為肯定，表示節點 j 所有的子節點，均已完成路徑之計算(即 $Y_k^j, k \in B_j$ 均已計算完成)，則可以執行該節點所有輸入路徑之 max 運算，即 $\gamma_j = \max(Y_k^j), \forall k \in B_j$，同時 γ_j 會紀錄於 ETS 資料結構 $Output_j$ 中，且 max 計算完成後，追蹤程序仍會持續爬升，尋找更上層之節點標籤。若 $finish\_flag_j = 0$，答案為否定，表示節點 j 尚有子節點尚未完成路徑之計算，此時 Upward_Tracing(k,j)便要將節點 j 堆入堆疊中，緊接著執行 Downward_Tracing(j)，持續下探節點 j 尚未完成路徑計算之子節點路徑。

本節再以圖 3.7 之 ETS 結構為範例，實際執行上述之追蹤程序，相關的演算步驟及數據紀錄均有詳細之說明，並詳列於附錄 C 中。

Upward_Tracing(k,j) 及 Downward_Tracing(j) 在執行的過程中，*Shared_flag* 旗標是讓追蹤程序判斷追蹤程序中之 current_node j 是否重複被拜訪過，若 *Shared_flag*=1，表示 current_node j 已經在追蹤的過程中出現過，且其相關節點時間之累算亦已經完成。若兩路徑有共用的子路徑存在，在 ETS 結構中就會出現重複的節點，這情況便是本論文在第 3.3 節中所討論之路徑相依性問題。LCTA 能夠處理解決該問題，便是在 Upward_Tracing(k,j) 及 Downward_Tracing(j)追蹤的程序中加入了 *Shared_flag* 旗標的判斷，相關的操作細節擬於下一節說明之。

3.4.4 LCTA 處理路徑相依性所造成估算之誤差

第 3.3 節中已將專案網路中路徑相依因素，對隨機性專案網路完成時間估算造成相當之影響，做了詳細的解釋及說明。其造成估算偏差的原因，主要是專案網路中，對相依的路徑執行 max 運算所造成的。所謂兩路徑相依即表示兩路徑中，含有共同的節點或子路徑；因 max 運算結果對輸入路

徑時間之變異數(Variance value)非常敏感，即便有兩組不同之輸入路徑，執行相同之 max 運算，雖彼此之間路徑完成時間之平均值相同，但變異數不同，其 max 計算之結果就會產生差異。這些共同的節點或子路徑會讓 max 運算之其中一條輸入路徑，因重複之節點或子路徑而在此路徑上產生額外之變異數，這便是誤差之來源。其解決的方式就是要將該路徑額外產生之變異數除去，因此 LCTA 追蹤程序中必須要完成兩件事：

1. 何在專案網路中找出所有的重複路徑?

2. 及如何將重複路徑中之額外變異數除去?

一般文獻之演算法對此問題並未注意，且亦不容易處理。

LCTA 則採用了 ETS 結構及 Upward_Tracing(*k,j*)及 Downward_Tracing(*j*)追蹤程序，處理上述問題並不難。LCTA 將處理的焦點放在追蹤過程中，凡執行 max 運算之節點 *j*，先檢查其所有的輸入路徑是否有重複之節點或子路徑。參考圖 3.8，LCTA 在 ETS 結構中實施後序追蹤原則，其特色就是能有系統的將專案網路之所有節點之輸入路徑拜訪一遍，在拜訪的過程中亦可透過 ETS 資料結構將相關之專案網路資料紀錄起來。因此 LCTA 可透過此機制，另額外加入 *Shared_flag* 旗標做輔助，便能輕易解決該路徑相依所造成的問題。

假設起始節點 *1* 至節點 *j* 之子路徑為兩專案路徑所共有(兩路徑相依性發生)，當 Upward_Tracing(*k,j*) 執行節點 *j* 並完成 γ_j 運算後，即表示該節點已經被第一次「正式拜訪完畢」，因此設定 $finish\_flag_j = 1$ 作為記號。節點 *j* 第二次被拜訪的時機，一定會發生在爾後追蹤過程中之 Downward_Tracing(*j*)程序中，該程序執行的第一件事就是檢查節點 *j*，是否 $finish\_flag_j = 1$？若為肯定，表示節點 *j* 已被拜訪過，因此設定 *Shared_flag*=1，隨後接續之 Upward_Tracing(*k,j*)(原節點 *j* 變為節點 *k*)程序，因得知該節點 *k* 已被拜訪過，便使用 $Y_k^j = \mu(\gamma_k) \oplus X_{kj}$ 取代 $Y_k^j = \gamma_k \oplus X_{kj}$ 之運算，將起始節點 *1* 至節點 *k* 之子路徑額外產生之變異數，自第二條路徑 Y_k^j 中除去之。LCTA 利用 γ_k 之平均值 $\mu(\gamma_k)$ 取代 γ_k 執行多餘的路徑變異數消除，相關的理論可以參考前 3.3.4 節中所述。

為進一步說明 LCTA 處理路徑相依性之問題，本小節以圖 3.4 專案為範例，其 ETS 架構表示於圖 3.6 中，相關之演算步驟詳如附錄 D 所示。該

範例共有 7 節點、兩條完成路徑，分別為路徑(1-2-3-4-5-7)及(1-2-3-4-6-7)，兩路徑存在共同子路徑(1-2-3-4)。LCTA 從圖 3.6 節點 7 開始執行 Downward_Tracing(7)，該程序會持續追蹤到起始節點 1 為止。接續由 Upward_Tracing(1,2)沿原追蹤路徑爬升至節點 7，過程中分別將節點時間累算如 $Y_1^2=\gamma_2$，$Y_1^3=\gamma_3$，$Y_1^4=\gamma_4$，$Y_1^5=\gamma_5$及Y_1^7，並紀錄在 ETS 資料結構中，當中節點 4 之 $finish\_flag_4=1$。因節點 7 尚有路徑(1-2-3-4-6-7)未完成計算，因此繼續沿該路徑執行 Downward_Tracing(7)，當執行到節點 4 時，Downward_Tracing 發現 $finish\_flag_4=1$，因此設定 *Shared_flag*=1，並接續執行 Upward_Tracing(4,6)，因節點 4 為重複出現之節點，因此 $Y_1^6=\mu(\gamma_4)\oplus X_{46}$，其中 $\mu(\gamma_4)$ 為 γ_4 的平均值，完成該計算後 *Shared_flag* 重設(*Shared_flag*=0)。Upward_Tracing 持續進行，因 *Shared_flag*=0 之故，回復 $Y_2^7=\gamma_4\oplus X_{67}$ 之計算。因節點 7 已完成所有輸入路徑之計算，因此執行 $\gamma_7=\max(Y_1^7,Y_2^7)$ 得到最號估算結果。若採用 DA 演算法，$Y_1^6=\mu(\gamma_4)\oplus X_{46}$ 會以 $Y_1^6=\gamma_4\oplus X_{46}$ 取代，路徑(1-2-3-4-6-7)時間 Y_2^7 會多出 $Var(\gamma_4)$ 的變異量，LCTA 則利用 $\mu(\gamma_4)$ 將該路徑時間多出的變異量消除之，從表 3.4 中 LCTA 與 DA 之比較結果，很明顯的 LCTA 演算程序達到非常好的效果。

3.4.5 LCTA 演算程序之綜整

綜合本章內容，LCTA 演算的整體流程可整理如下步驟所列:

1. 依據 3.4.1 節，將隨機性專案網路結構轉換成擴張樹結構(ETS).
2. 依據 3.2.2 節，將隨機性專案網路中每一作業時間機率機率分佈函數，使用柴比契夫取樣點法先予以離散化。
3. 依據 3.4.2 節，針對 ETS 結構中每一個節點建立 ETS 資料結構。

   (1)設定所有節點資料結構內之 $finish\_flag_i$=1。

   (2)設定所有節點資料結構內之 Path_Flag(i)=0。

   (3)所有節點資料結構內之 $Output_i$ 及 Path_Time(i) 均設為 0。

2. 建立追蹤程序之初始條件：

   (1)current_node j = ETS 之根節點標籤。

   (2)last_node k=current_node。

(3)stack_pointor=1 令堆疊的指標初始值為 1。

(4)push_stack(*j*) 先將 current_node *j* 堆入堆疊中。

3. 執行 Downward_Tracing(*j*).

6. Downward_Tracing(*j*)及 Upward_Tracing(*k,j*)會交互演算，最後專案網路所有的節點輸出時間 γ_i，均會存入 ETS 資料結構中之 $Output_i$，專案網路完成時間之估算即為 $Output_i=\gamma_N$。

3.4.6 LCTA 複雜度分析

LCTA 演算法透過後序追蹤程序尋訪擴張樹結構各個節點，當各節點完成其所有子節點的路徑運算後，即執行一次 max 運算。從擴張樹演算法中得知節點間執行 convolution 運算及節點之 max 運算，若採用重新取樣技術，其運算複雜度幾可視為常數，與專案網路節點數 N 無關，因此計算 LCTA 演算法運算複雜度乃以節點間執行 convolution 運算及節點之 max 運算次數多寡為考量。擴張樹結構中各節點均須執行一次 max 運算，故其運算次數為 N；而節點間執行 convolution 運算次數則視兩種情況而定：

1. 專案網路的複雜度，即專案網路各節點之子節點數多寡。

2. 專案網路各節點是否共有同一子節點的情況多寡，若兩節點共用一個子節點(意即該子節點有兩個父節點)，則屬該子節點以下之節點間 convolution 運算僅須執行一次即可，因該子節點執行所得的結果可直接供其上所有父節點使用。

由上所述可知，若設擴張樹結構中各節點之子節點數為 r，且考慮節點間 convolution 運算次數最多的情況下(即任兩節點間均無共用子節點的情況)，其專案網路的擴張樹結構如圖 3.10 所示：

圖 3.10 之專案網路共計 N 個節點，因該結構節點數由上而下呈等比級數排列，故可分成 m 個階層(layer)，其中 $m=\log_r(N)+1$。由前文所述，專案網路 max 演算次數之複雜度可表為 $O(N)$，其中 N 從圖 3.10 中亦可以表示為：

$$N=\frac{r^{m-1}-1}{r-1} \tag{3.28}$$

專案網路節點間 convolution 的計算次數 P 實際上就是圖 3.11 中所有節點間連接次數，可以表示為：

$$P = \frac{r^m - 1}{r - 1} + r^{m-1} \tag{3.29}$$

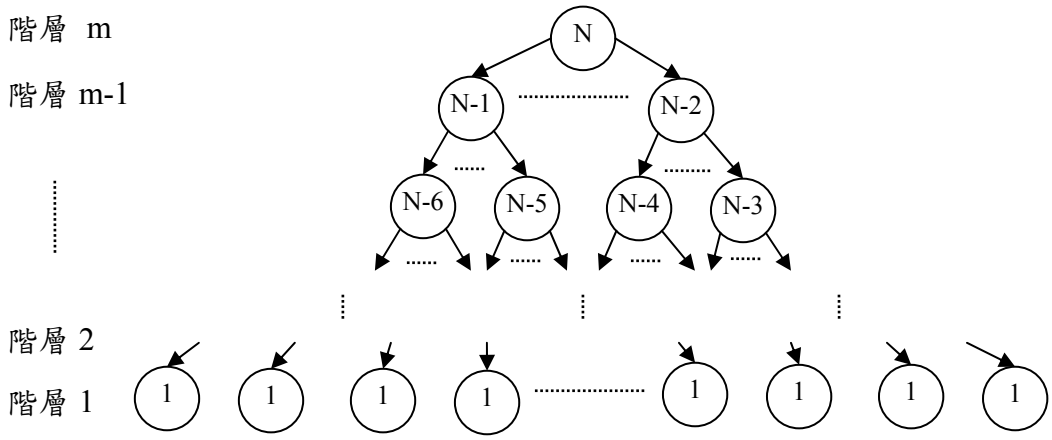


圖 3.10 專案網路最複雜之樹狀結構(worest case)

將 $m = \log_r(N) + 1$ 值代入(3.29)式，可得

$$P = \frac{N - 1}{r - 1} + \frac{N}{r} \tag{3.30}$$

從(3.30)式可以得知專案網路節點間的計算複雜度亦為 $O(N)$。

然而實際上 max 及 convolution 運算本身仍有複雜度存在且相等，固令為 $O(Oper.)$，因此 LCTA 實際上的複雜度應表示為 $O(N) \cdot O(Oper.)$

3. 5 LCTA 之實驗數據

本章節將以實驗顯示本論文所發展之 LCTA 估算隨機性專案網路完成時間的效果，證明 LCTA 優於 Dodin Algorithm 及 PERT 之演算法。依據第 1.3 章節論文目的內容，LCTA 所採用的實驗範例有下列三項要求：

1. 除要求專案作業時間機率分佈函數不得為特定之機率模式。

2. 適應各種複雜度(Complexity)之專案網路結構。

3. 另須能顯示出 LCTA 針對大型專案網路的估算的效果。

針對第 1 項要求，LCTA 所使用任何的專案網路範例，所有的作業均以常態分佈函數(Normal distribution function)、指數分佈函數(Exponential distribution function)及均勻分佈函數(Uniform distribution function)三種時間

機率分佈型態，隨機搭配給網路中之作業，且函數之平均時間(Mean value)及變異數值(Variance value)，亦以[1, 10]的數值範圍隨機指定給各函數。

針對第 2 項要求，LCTA 實驗所使用的範例則採用 Demeulemeester and Herroelen (1997)使用 Kolisch(1996)的 ProGen 軟體產生的，該文獻以 full factorial 實驗設計方式來產生專案網路結構實體(Instance)，該實驗設計所使用的專案網路作業數量均設為 32 個，所考量到的實驗因子(Factors)有三個：專案網路複雜度參數(Complexity)(水準為 1.5, 1.8, 2.0，水準係數愈大專案網路結構愈複雜)，每個網路節點之輸入節點(Predecessors)及輸出節點數(Successors)亦均隨機決定，其範圍為[1,3]之間。依上述實驗設計的方法及使用的參數取得 LCTA 實驗所需之 120 個專案網路實體(每一專案網路複雜度水準均分配 40 個網路實例)，這也是本章節 LCTA 執行的第一組實驗，實驗內容將比較 LCTA、DA 及 PERT 三種演算法，比較內容為估算執行時間、完成時間平均值及完成時間標準差值三項，每一個實驗範例，演算法均會進行兩次，第一次不採用重覆取樣技術(DRT)，第二次則採用重覆取樣技術，並與第一次結果作比較。所有實驗會以 20000 次樣本值之蒙地卡羅模擬法(MCS)所得之結果來作為演算法比較的基準值，該 MCS 模擬之結果會趨近於真實結果。本論文使用 t 分配、信心水準 0,999 的情況下，並利用第二組實驗 20 個大型專案網路為範例，驗證 20000 次樣本值之 MCS 結果，確實能夠接近真實值(與真實質平均誤差 0.36%)，相關之說明如附錄 G 所列，其它相關之文獻，如 Burt & Garman(1971), Dodin(1985), Sigal(1979)亦以大數量樣本之 MCS 結果作為演算法驗證的標準。

針對第 3 項要求， LCTA 將進行本章節的第二組實驗。LCTA 實驗所使用的範例則由本論文採用 Kolisch(1996)的 ProGen 軟體，依據上述同樣之專案網路條件，產生 20 組 100 個節點(作業數量約 130~160 之間)的大型專案網路。實驗的對像及實驗的內容同第一組實驗，同樣以 20000 次樣本值之蒙地卡羅模擬法(MCS)之所得結果比較的基準值。第二組實驗為了要進一部驗證 LCTA 所估算出來的專案網路完成時間機率分佈函數是否與真實的結果(MCS 結果)有差異，故針對每一大型專案範例，額外執行 K-S 檢定(Kolmogorov-Smirnov test)。

專案作業時間機率分佈函數在投入實驗前，均會將其實施離散化處理，因此實驗前必須先決定作業時機變數之取樣點數。取樣的點數愈大則

運算之結果愈精確，但會犧牲演算之時間。LCTA 採用柴比契夫取樣點法，因取變數前 4 個動差值(Moment value)之故，乃設定取樣點數為 5。本章節實驗所使用的個人電腦為 Intel Pentum-4 2.0 GHZ CPU、512MB 記憶體並使用 Matlab Ver.6.5 語言程式來執行。

1.5.1 LCTA 第一組實驗

複雜度愈高的專案網路，執行 max 運算的機會愈多，對於估算結果的精度影響愈大。因此 LCTA 第一組實驗的目的主要是驗證 LCTA 對於不同水準之專案網路複雜度，估算結果的比較。同時亦以 DA 演算法及 PERT 來執行相同專案網路範例的估算，並比較不同演算法間的結果，來證明 LCTA 估算隨機性專案網路完成時間的有效性。本組實驗所使用的專案網路範例是採 Demeulemeester and Herroelen (1997)使用 Kolisch(1996)的 ProGen 軟體所設計的，專案網路複雜度以三種水準設計(水準為 1.5, 1.8, 2.0)，每種水準 40 個共計 120 個，作業數量均為 32 個。另三種演算法同樣的實驗會執行兩次，一次為不採用重新取樣技術(DRT)來估算，另一次則採用該技術；目的是要進一步比較 LCTA 採用 DRT 對於估算的結果精度及演算速度之差異程度。LCTA 實驗的完整數據可參考附錄 E 所列，三種演算法綜整後的實驗結果如表 3.7、3.8 所列，演算法的結果會與 20000 次樣本之 MCS 模擬的結果作比較，比較的內容包括執行時間倍數比 $\frac{MCS}{method}$ (MCS 與演算法的執行時間比值)、平均值相對誤差 $\frac{|method - MCS|}{MCS}$ (演算法與 MCS 平均值之誤差百分比)及標準差相對誤差 $\frac{|method - MCS|}{MCS}$ (演算法與 MCS 標準差之誤差百分比)。

表 3.7、3.8 的結果很明白的顯示，LCTA 綜合的表現優於其它演算法，詳細的比較結果說明如下：

1. 雖執行的速度最快的為 PERT，然而其平均值誤差(13.75%)及標準差誤差(23.98%)值是最大的。
2. LCTA 在標準差誤差表現雖遜於 DA(23.25% vs. 8.87%)，但其平均值誤差比較好(2.5% vs. 6.26%)，其速度的表現亦比 DA 快非常多(310 vs 14.2)。
3. 演算法若採用 DRT 技術，最大的差異是演算的速度大幅度的增加，以

LCTA 為例，其執行速度比 MCS 快 1395 倍之多。

4. LCTA 採用 DRT 所估算的平均值與標準差與未採用 DRT 所得的結果沒有明顯的差異。

表 3.7　LCTA、DA 及 PERT 對不同專案網路複雜度估算結果之表較 (未使用 DRT)

專案網路複雜度	估算結果與 MCS 模擬結果之相對誤差 *method*：LCTA、DA 及 PERT								
	執行時間倍數比 $(\frac{MCS}{method})$			平均值相對誤差 $(\frac{\lvert method-MCS\rvert}{MCS})$			標準差相對誤差 $(\frac{\lvert method-MCS\rvert}{MCS})$		
method	*LCTA*	*DA*	*PERT*	*LCTA*	*DA*	*PERT*	*LCTA*	*DA*	*PERT*
1.5	197.20	20.81	3203	2.39%	6.84%	12.42%	15.53%	7.35%	25.23%
1.8	269.69	12.69	3558	2.05%	8.32%	14.93%	23.33%	4.33%	24.43%
2.0	463.08	8.96	3868	3.04%	3.64%	13.90%	30.90%	14.94%	22.29%
average	309.99	14.15	**3543**	**2.50%**	6.26%	13.75%	23.25%	**8.87%**	23.98%

表 3.8　LCTA、DA 及 PERT 對不同專案網路複雜度估算結果之表較 (使用 DRT)

專案網路複雜度	估算結果與 MCS 模擬結果之相對誤差 *method*：LCTA、DA 及 PERT								
	執行時間倍數比 $(\frac{MCS}{method})$			平均值相對誤差 $(\frac{\lvert method-MCS\rvert}{MCS})$			標準差相對誤差 $(\frac{\lvert method-MCS\rvert}{MCS})$		
method	*LCTA*	*DA*	*PERT*	*LCTA*	*DA*	*PERT*	*LCTA*	*DA*	*PERT*
1.5	1261.0	513.64	3203	2.32%	10.83%	12.42%	16.03%	5.50%	23.79%
1.8	1400.8	390.19	3558	2.07%	17.25%	14.93%	23.98%	5.95%	22.70%
2.0	1523.0	312.81	3868	3.04%	21.45%	13.90%	31.37%	11.99%	20.57%
average	1395.0	405.55	**3543**	**2.48%**	16.51%	13.75%	23.79%	**7.81%**	22.35%

1.5.2 LCTA 第二組實驗

本組實驗乃驗證 LCTA 在大型專案網路的環境中估算隨機性專案網路完成時間之成效。一般相關文獻並無提供專案作業數量大於 100 以上之專案網路範例，因此本論文同第一組實驗之範例，依據 Kolisch(1996)對專案網路之規格要求，自行寫程式隨機產生 20 個 100 個節點(作業量 130~160)之大型專案網路範例，範例的網路型態如圖 3.11 所示。

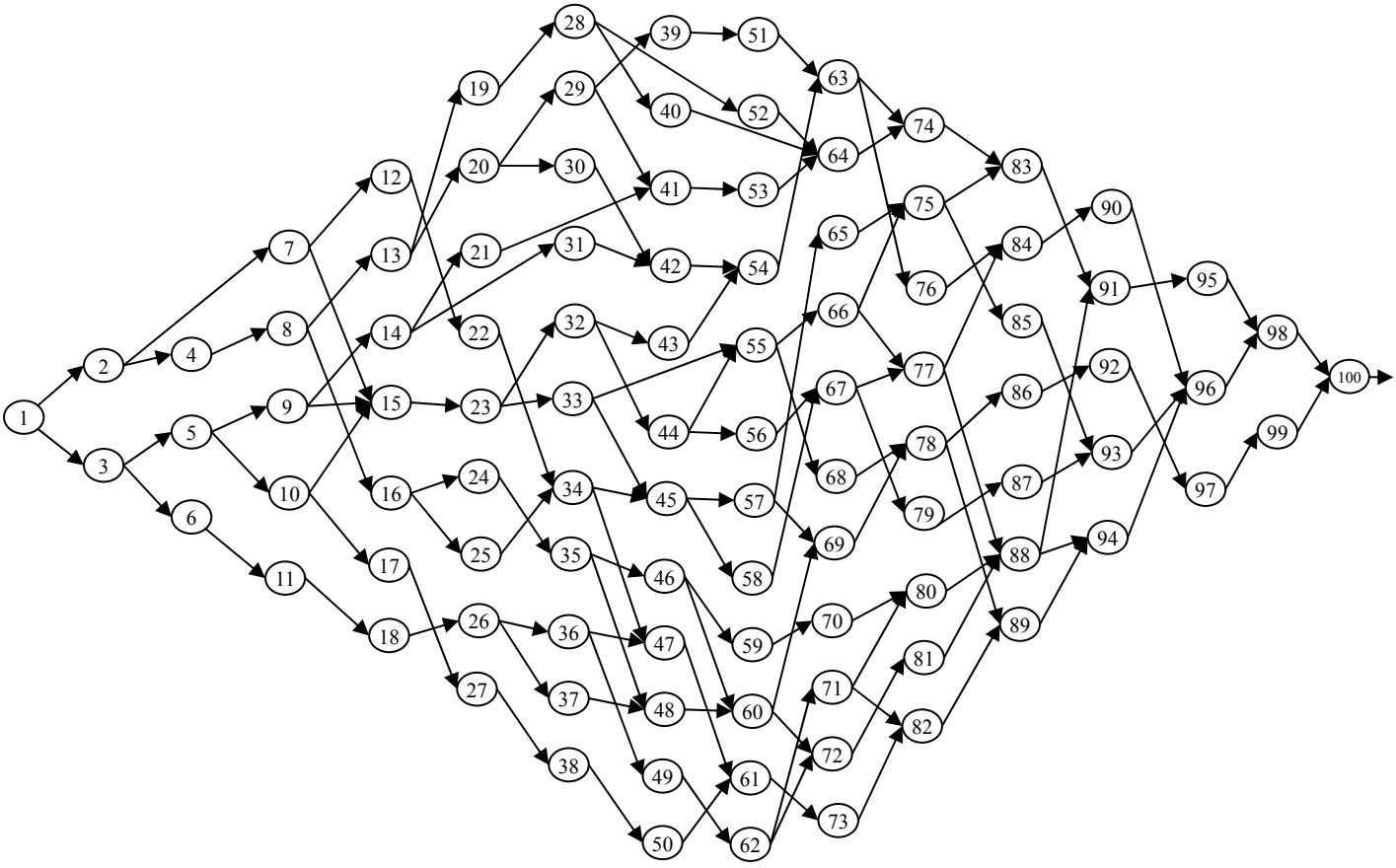


圖 3.11 大型專案網路型態範例

第二組實驗的內容及方式，同第一組實驗，針對 LCTA、DA 及 PERT 三種演算法並與 MCS 模擬之結果作比較，比較的內容同樣為估算的平均值相對誤差 $\frac{|method - MCS|}{MCS}$ 、標準差相對誤差 $\frac{|method - MCS|}{MCS}$ 及執行時間倍數比 $\frac{MCS}{method}$，相關的結果如表 3.10 所列。除上述比較的內容外，本組實驗另採用 K-S 檢定法，進一步檢驗 LCTA 所估算隨機專案網路完成時間機率分佈函數的形狀，與實際情況之差異，該檢驗亦同樣的應用於其它兩個演算法

上。K-S 檢定法與檢驗的樣本數息息相關，樣本數愈大，檢驗的標準會愈高，受檢對象愈不容易通過檢驗。本組實驗以信心水準$\alpha = 0.01$，採三種樣本數(30, 40 及 50)來執行 K-S 檢定，相關的檢驗結果亦綜整於表 3.9 中，詳細的檢驗內容如附錄 G 所列。。

表 3.9　LCTA、DA 及 PERT 對大型專案網路估算結果之比較

比較項目	MCS	LCTA w/o DRT	LCTA w/ DRT	DA w/o DRT	DA w/DRT	PERT
執行時間倍數比 $(\frac{MCS}{method})$	2103	318.2	7510.7	12.4	753.8	19118.2
平均值相對誤差 $(\frac{\lvert method - MCS\rvert}{MCS})$	-	1.89%	2.06%	29.75%	28.67%	24.40%
標準差相對誤差 $(\frac{\lvert method - MCS\rvert}{MCS})$	-	19.8%	24.8%	51.9%	25.8%	23.3%
K-S 檢驗 $\alpha = 0.01, K$=30	-	**20/20**	**20/20**	0/20	0/20	0/20
K-S 檢驗 $\alpha = 0.01, K$=40	-	18/20	19/20	0/20	0/20	0/20
K-S 檢驗 $\alpha = 0.01, K$=50	-	16/20	17/20	0/20	0/20	0/20

表 3.9 的結果亦很明白的顯示，LCTA 對於大型專案網路綜合的表現較第一組實驗更為突出，詳細的比較結果說明如下：

1. 雖執行的速度最快的仍為 PERT，然而其平均值誤差提升至 24.4%及標準差誤差仍維持 23.3%的水準。
2. DA 的整體表現在大型專案網路中最差(平均值及標準差誤差分別為 29%, 25.8%)，LCTA 正與其相反，平均值及標準差誤差均優於 DA。
3. LCTA 若採用 DRT 技術，演算的速度亦大幅度的增加，尤其是平均值誤差仍維持在 2%之水準，唯標準差誤差提升至 24.8%。
4. K-S 檢驗部份，LCTA 在樣本數 30 的情況下 20 個專案網路範例全數通過檢驗，樣本數 40 及 50 亦大部份通過檢驗。然而 DA 及 PERT 的檢驗則全部均未通過檢驗。

從上述兩組實驗非常明顯的顯示，本論文所發展之 LCTA 演算法，估算隨機性專案網路完成時間機率分佈函數之表現，優於最受青睞的工具 PERT 及文獻中之 DA 演算法。LCTA 搭配了重複取樣技術(DRT)，使演算速度大幅提升，同時對估算的品質並未造成明顯的影響，基於此，LCTA 運用於大型的專案網路之可行性大為增加。

3.6 結論

針對隨機性專案網路而言，各作業執行時間均為隨機變數，因此專案網路的完成時間亦必然為隨機變數，其型態為一機率分佈函數。若使用 PERT 模式來估算求解，並無法得到精確的結果。雖已有若干文獻提出其它估算方法，有些需要求專案網路作業時間僅能侷限於某單一機率分佈型態，有些則僅能針對專案完成時間的平均值或變異數之界線範圍作估算，另一些則因演算分析過程複雜，無法面對處理大型專案網路，而失去其實際應用的價值。就本論文之瞭解，甚少文獻能夠將專案網路完成時間機率分佈函數實際的估算出來，更遑論出現大型專案網路的演算範例。僅 Dodin(1985)採用離散化技術執行運算，利用疊代式運算的方式，執行 convolution 及 max 之累算。該文獻雖能估算專案網路完成時間機率分佈函數，唯其執行 max 運算時，未能將路徑相依性的問題考慮進去，而造成誤差。另 DA 演算法須先對專案網路執行串列/並行縮減程序(Series/Parallel reduction)，以減少專案網路之複雜度，方能執行估算工作。其原因是 DA 演算法沒有提出有系統的疊代運算程序，故仍無法處理大型專案網路。

針對上述隨機性專案網路估算完成時間所出現相關問題，均須有效的解決方能夠將估算的精度提高，另考慮對大型專案網路的實際應用，除考慮估算精度外，仍須考慮其估算的效率性，本論文基於此，乃提出 LCTA 之演算法。因允許專案網路作業時間之機率分佈可為多種型態，LCTA 仍沿用 DA 之離散化技術作為運算之基礎，唯先在離散取樣方法上做相關之實驗及評估，最後決定了柴比契夫取樣點法。另考慮大量的離散運算樣本值會擴散的問題，繼而影響演算法的效率，LCTA 則利用了重新取樣之技術，將樣本值擴散問題獲得有效的控制。針對路徑相依性的問題，LCTA 通盤分析了隨機性專案網路在 max 計算上的特性，提出相

關的解決方法，並將其納入 LCTA 演算法中。LCTA 的另一項特色，就是無論專案網路結構有多複雜、多龐大，均能利用演算法中所設計之擴張樹結構(ETS)及後序追蹤演算程序，有系統的執行專案網路完成時間之累算，無須先執行網路縮減工作。

LCTA 最主要的貢獻是面對大型之隨機性專案網路，能夠有效率的估算出其完成時間機率分佈函數，尤其是完成時間平均值之準確度，超過 DA 及 PERT 甚多。從上述 3.5 章節 LCTA 實驗數據內容得知，處理約 150 個作業數量之大型專案網路，LCTA 的估算平均值之誤差值仍可維持 2% 的準確度，相對於 DA 及 PERT 之估算誤差值，動則便超過 20%以上，實無法與 LCTA 相比較。LCTA 估算的速度若配合重複取樣技術(DRT)後，以同樣上述專案網路條件下，其執行估算的時間平均為 0.3 秒，平均為 MCS 模擬法 7510 倍之多。

實務上，就專案管理者角度來看，隨機性專案網路完成時間，其機率分佈函數(*pdf*)對專案網路是非常重要的資訊。舉凡專案網路風險管理、專案網路成本分析、專案網路排程問題及專案網路最佳化資源投入問題等，均須以該資訊為基礎來做相關之計算分析，因此隨機性專案網路完成時間估算的重要性，便由此可知。LCTA 所估算出來的完成時間 *pdf*，本論文亦於 3.5 章節中，以 K-S 檢驗法來執行檢驗，在信心水準 $\alpha = 0.01$，以樣本數為 30 的情況下，針對上述之大型專案網路，LCTA 全數通過檢驗。相對之 DA 及 PERT 演算法，在同樣的檢驗條件下，沒有一個通過檢驗。本論文採用數量甚多的專案網路實例，且經實驗結果證明，LCTA 對於隨機性專案網路完成時間之估算，確實是一有效率的估算工具。

第四章　區域積分演算法

當專案網路中有若干不同子路徑匯集於一節點上，在確定性(Deterministic)專案網路環境中，該節點的輸出時間可透過 max 運算求得，其結果即等於最長時間之子路徑。而在隨機性的專案網路環境中，因子路徑的時間均為時間變數，對該節點的輸出結果便不能以確定性專案網路的方式求取，即隨機變數間執行 max 運算的結果，其平均值會比確定性專案網路的結果來得大，本論文稱此現象為最長路徑偏移(LPB)(參考 3.3.2 章節內容)，該問題對於大型專案網路的完成時間估算會造成相當大的影響，依據第三章內容中所述，路徑相依性所造成的估算偏差亦源於 LPB 值。

LPB 是估算隨機性專案網路完成時間最主要的誤差來源，Dodin & Sirvanci(1990), Mehrotra, et al.(1996), Dodin(1985), Clark(1961) 及 Klingel(1966)所提出之估算方法，均以求取正確之網路 LPB 值為努力的目標。LPB 值主要源於專案網路中子路徑間 max 運算之結果，雖該問題能以統計學中「順序統計理論」(Order Statistics Theory)推導出數學分析模式，然而該模式過於複雜，且侷限於特定之機率模式，無法應用於實務上之專案網路。Clark(1961)亦針對 max 運算元作分析，導出 max 運算結果平均值及變異數之公式，但所處理的變數僅能侷限於常態分佈函數。Dodin & Sirvanci(1990)則採用極值理論(Extreme Value Throry, EVT)處理 max 運算，然而該理論假設所有參與 max 運算的變數，須源於同一樣本母體。Dodin(1985)及本論文第三章所提出之 LCTA 則採用離散化技術執行 max 運算，雖能擺脫變數侷限於特定機率型態的限制因素，然而其離散樣本取樣值數目，在估算的過程中，會因重複的計算而膨脹，對於大型專案網路而言，得確會造成計算上之負擔。

本章節基於上述之理由，擬提出區域積分演算法(Marginal integration Algorithm, MIA)所推導之公式，來取代 LCTA 中之 max 運算結果，期減少 max 運算的執行時間並同時將 max 運算所產生之 LPB 值分析出來，直接求得 max 運算的結果，無須執行 3.2.3 章節中 max 運算的繁複細節。本章節首先針對 max 運算之複雜度(Complexity)作分析，再介紹 MIA 的原理及公式之推導。章節內容接續使用 LCTA 之演算法結合 MIA 運算元，實際對專案網路作驗證，內容包括三種離散化技術的選擇實驗，選取適合 MIA 之離散化技術。另驗證 MIA 處理專案網路路徑相依性的執行能力，並以一簡單

的專案網路實例，驗證其估算專案網路完成時間的效果。最後本章節同第三章之數據實驗內容，同樣以 20 個大型專案網路(100 個作業節點)為實例，並與 PERT、Dodin Algorithm 及 MIA 執行所得的估算結果作比較，來證明 MIA 優於上述之演算法。

4.1 max 運算複雜度之分析

參考 3.2.3 章節第式(3.9)，令兩變數 X, Y 之離散取樣點數分別為 Q 及 R，X, Y 兩變數均表示為矩陣型態，第一列表示變數的樣本函數值，分別為 $x_i, i=1,...,Q$， $y_j, j=1,...,R$；第二列則表示相對應之樣本機率值，分別為 $p_i^X, i=1,...,Q$， $p_j^Y, j=1,...,Q$。該兩變數之離散化 max 運算結果 Z=max(X,Y) 可以表示如式(3.10)， Z 之離散取樣點數為 W，樣本函數值為 $z_k, k=1,...,S$，樣本機率值 $p_k^Z, k=1,...,S$。Max 運算的程序如下:

4. 先取 X, Y 中每一樣本函數值來做 max 運算，可得 $z_{ij}=\max(x_i, y_j)$。所得之 Z 變數樣本函數值 $z_k=z_{ij}$ 相對應之機率值為 $p_{ij}^Z=p_i^x p_j^y$，其中 $i=1,\ldots,Q$, $j=1,\ldots,R$, 且 Z 變數最多可有 $W=\max(Q,R)$ 樣本單元。
5. 將求出之 Z 變數每一樣本函數值 (z_{ij}, p_{ij}^Z), 依 z_{ij} 從小至大做排序。
6. 將 z_{ij} 值相同的樣本函數值合併為一，並表示為 z_k，且其中相對應之 $p_i^x p_j^y$ 要加總為 $p_k^z=\sum_{(i,j)\in T} p_i^x p_j^y$, $T=\{(i,j)\,|\,z_{ij}\ has\ same\ value\}$。

依據上述 max 運算步驟程序，第一步驟 max 運算要執行 $Q\times R$ 次之 $\max(x_i, y_j)$ 運算，故其複雜度為 O(N^2)；第二步驟 max 運算要執行 z_{ij} 值的排序工作，共計 $W\ln W$ 次，故其複雜度為 O($N\log N$)；第三步驟則要執行 W 次 z_{ij} 值合併工作，故其複雜度為 O(N)。Max 運算三個步驟的複雜度共計:

$$O(N^2)+O(N\log N)+O(N) \tag{4.1}$$

N 為離散變數樣本單元數，會隨著專案網路運算持續增加，因此 max 運算的計算負擔亦會愈來愈重。

4.2 區域積分計算之原理

依據 3.3.2 章節所述，LPB 值與隨機變數之變異數值有關係，原因乃是

變數之機率分佈函數會隨著變異數值之增加，機率分佈函數間發生重疊的機會亦愈高，而重疊的範圍愈多，變數間執行 max 運算所產生之 LPB 值會愈大。MIA 即是利用此現象，擬將變數間機率分佈函數之重疊範圍與 LPB 偏移量間找出關係式，藉以透過該關係式，直接計算出 LPB 偏移量，無須再執行較為費時之 max 運算。

4.2.1 MIA 計算公式推導

本章節先以圖 4.1 及 4.2 來做 MIA 之實例說明，令兩相互獨立之變數 *X, Y* 均為常態機率分佈函數，分別表示為 *N*(14,25)及 *N*(20,25)，其中 N(a,b), a 為平均值, b 為標準差值(如圖 4.1)，兩變數之機率分佈函數存在重疊之函數區間。對 *Y* 變數而言，該重疊區間範圍為$[y_{1,} y_m]$，對 *X* 變數而言則為$[x_{n,} x_Q]$。若採用 20000 次取樣之蒙地卡羅模擬法執行$Z = \max(X, Y)$運算，其 *Z* 變數所得之機率分配圖如圖 4.2 所示。若採傳統之 PERT 分析模式，其 max 運算所得之 *Z* 變數應等於 *Y* 變數；但很明顯的，max 運算因存在前文所述之最長路徑偏差(LPB)之故，其機率分配圖會較 *Y* 變數左偏(即平均值 E[*Z*]=20.82 大於 E[*Y*]=20)。這便形成本論文發展 MIA 之動機：*是否能夠以一快速的演算法，計算出兩變數執行 max 運算所產生之 LPB 偏差值，以取代較耗時之 max 運算*。圖 4.2 也證明若對 *X, Y* 變數執行 MIA 運算(取代 max 運算)，其結果的平均值為 20.78，較為接近 MCS 之結果。

為能清楚的說明MIA的原理及如何求出變數間執行max運算所產生之 LPB 偏差，本章節擬先以實例做說明，參考式(4.2)，以 *X,Y* 兩變數為例:

$$X = \begin{bmatrix} 1 & 3 & 5 & 7 & 9 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \end{bmatrix}, Y = \begin{bmatrix} 4 & 6 & 8 & 10 & 12 & 14 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \end{bmatrix} \quad (4.2)$$

本論文依據 4.1 章節中之$Z = \max(X, Y)$之演算規則，將所產生之z_{ij}值，以矩陣資料結構表示，如表 4.1 所列。該矩陣資料結構非常適合做為 MIA 原理說明之參考。執行 max 運算所產生之資料結構，所產生 *Z* 變數平均值的通式可表示為；

$$E(Z) = \sum_{j=1}^{R} \sum_{i=1}^{Q} \max(x_i, y_j) p_i^x p_j^y = \sum_{j=1}^{R} \sum_{i=1}^{Q} z_{ij} p_i^x p_j^y \quad (4.3)$$

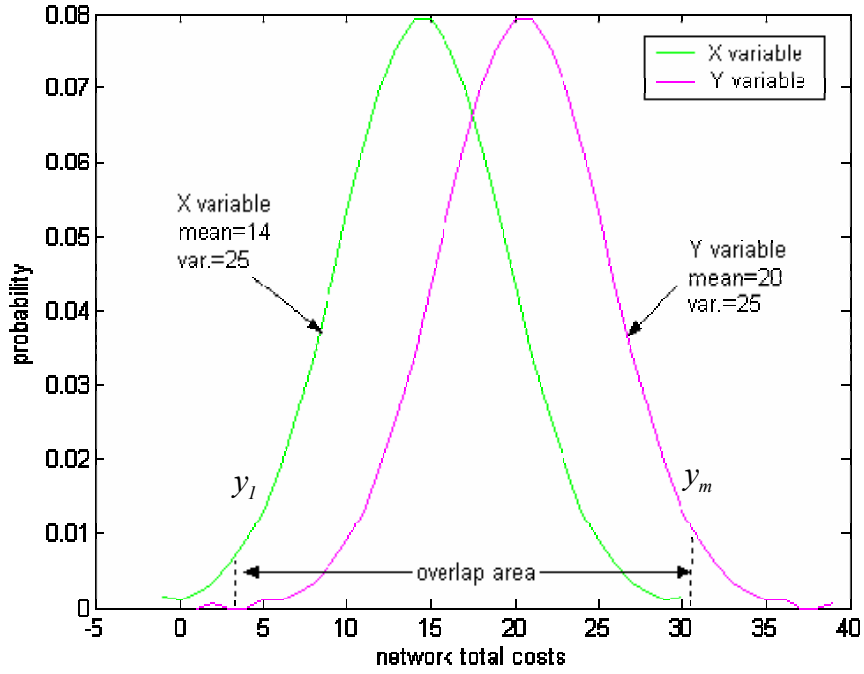


圖 4.1　X,Y 兩時間變數的機率分配函數

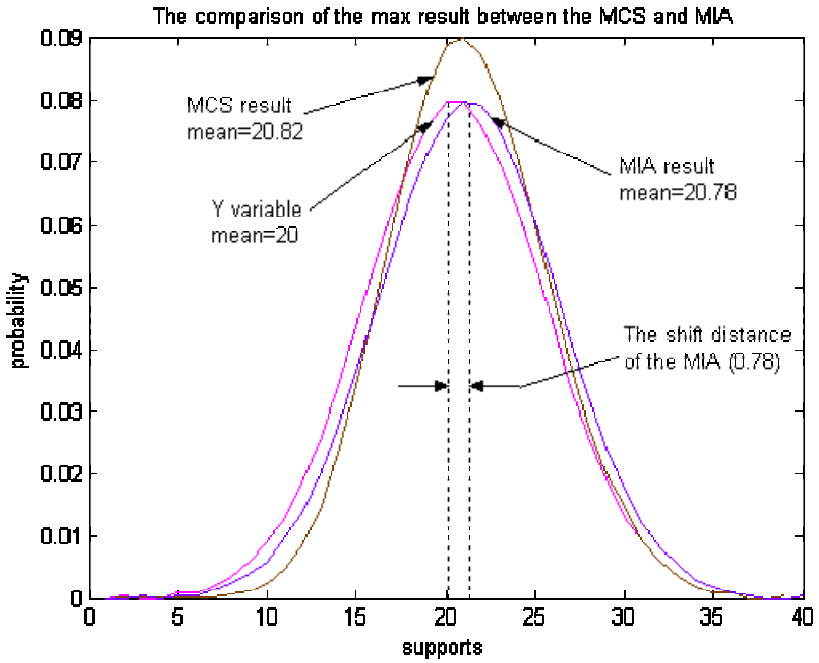


圖 4.2　MIA, PERT 及 MCS 執行圖 4.1 $Z=\max(X,Y)$ 比較

因 X,Y 兩變數有重疊函數區域，表示為$[y_{1,}y_m]$或$[x_{n,}x_Q]$。可參考表格 4.1 並針對(4.3)式，可定義兩個重要的變數 y_m 及 x_n 如下:

$$x_n \triangleq \min_{i=1,\ldots,Q}\{x_i \geq y_1\}, \quad y_m \triangleq \max_{j=1,\ldots,R}\{y_j \leq x_Q\} \tag{4.4}$$

表 4.1　$Z = \max(X,Y)$之矩陣資料結構

	X	$x_1 = 1$	$x_2 = 3$	$x_3 = 5$	$x_4 = 7$	$x_5 = 9$
Y	$\Pr(X)$ / $\Pr(Y)$	p_1^x	p_2^x	p_3^x	p_4^x	p_5^x
$y_1 = 4$	p_1^y	$\max\{y_1, x_1\} = y_1 = 4$ $p_1^y\ p_1^x$	$\max\{y_1, x_2\} = y_1 = 4$ $p_1^y\ p_2^x$	$\max\{y_1, x_3\} = x_3 = 5$ $p_1^y\ p_3^x$	$\max\{y_1, x_4\} = x_4 = 7$ $p_1^y\ p_4^x$	$\max\{y_1, x_5\} = x_5 = 9$ $p_1^y\ p_5^x$
$y_2 = 6$	p_2^y	$\max\{y_2, x_1\} = y_2 = 6$ $p_2^y\ p_1^x$	$\max\{y_2, x_2\} = y_2 = 6$ $p_2^y\ p_2^x$	$\max\{y_2, x_3\} = y_2 = 6$ $p_2^y\ p_3^x$	$\max\{y_2, x_4\} = x_4 = 7$ $p_2^y\ p_4^x$	$\max\{y_2, x_5\} = x_5 = 9$ $p_2^y\ p_5^x$
$y_3 = 8$	p_3^y	$\max\{y_3, x_1\} = y_3 = 8$ $p_3^y\ p_1^x$	$\max\{y_3, x_2\} = y_3 = 8$ $p_3^y\ p_2^x$	$\max\{y_3, x_3\} = y_3 = 8$ $p_3^y\ p_3^x$	$\max\{y_3, x_4\} = y_3 = 8$ $p_3^y\ p_4^x$	$\max\{y_3, x_5\} = x_5 = 9$ $p_3^y\ p_5^x$
$y_4 = 10$	p_4^y	$\max\{y_4, x_1\} = y_4 = 10$ $p_4^y\ p_1^x$	$\max\{y_4, x_2\} = y_4 = 10$ $p_4^y\ p_2^x$	$\max\{y_4, x_3\} = y_4 = 10$ $p_4^y\ p_3^x$	$\max\{y_4, x_4\} = y_4 = 10$ $p_4^y\ p_4^x$	$\max\{y_4, x_5\} = y_4 = 10$ $p_4^y\ p_5^x$
$y_5 = 12$	p_5^y	$\max\{y_5, x_1\} = y_5 = 12$ $p_5^y\ p_1^x$	$\max\{y_5, x_2\} = y_5 = 12$ $p_5^y\ p_2^x$	$\max\{y_5, x_3\} = y_5 = 12$ $p_5^y\ p_3^x$	$\max\{y_5, x_4\} = y_5 = 12$ $p_5^y\ p_4^x$	$\max\{y_5, x_5\} = y_5 = 12$ $p_5^y\ p_5^x$
$y_6 = 14$	p_6^y	$\max\{y_6, x_1\} = y_6 = 14$ $p_6^y\ p_1^x$	$\max\{y_6, x_2\} = y_6 = 14$ $p_6^y\ p_2^x$	$\max\{y_6, x_3\} = y_6 = 14$ $p_6^y\ p_3^x$	$\max\{y_6, x_4\} = y_6 = 14$ $p_6^y\ p_4^x$	$\max\{y_6, x_5\} = y_6 = 14$ $p_6^y\ p_5^x$

為配合表 4.1 之矩陣資料結構，將式(4.3)Z 變數的平均值分成兩部分:白色區域(White area)及灰色區域(Grey area)，可表示為:

$$\begin{aligned} E(Z) &= \sum_{j=1}^{R}\sum_{i=1}^{Q}\max(x_i, y_j)\,p_i^x\,p_j^y = \sum_{j=1}^{R}\sum_{i=1}^{Q} z_{ij}\,p_i^x\,p_j^y \\ &= (\text{White area}) + (\text{Grey area}) \end{aligned} \tag{4.5}$$

其中白色的區域的z_{ij}值為$z_{ij} = y_j = \max(x_i, y_j)$，表示 Y 變數中之樣本資料大於 X 變數之樣本資料部分。相關的表示式如下:

$$\begin{aligned}(\text{White area}) &= \sum_{j=1}^{(m-1)}\sum_{i=1}^{(k+j-2)} \max(x_i, y_j) p_i^x p_j^y + \sum_{j=k}^{R}\sum_{i=1}^{Q} \max(x_i, y_j) p_i^x p_j^y \\ &= \sum_{j=1}^{(m-1)}\sum_{i=1}^{(k+j-2)} y_j p_i^x p_j^y + \sum_{j=k}^{R}\sum_{i=1}^{Q} y_j p_i^x p_j^y\end{aligned} \tag{4.6}$$

而灰色區域的z_{ij}值為$z_{ij} = x_i = \max(x_i, y_j)$，則表示 Y 變數中之樣本資料小於 X 變數之樣本資料部分。相關的表示式如下:

$$(\text{Grey area}) = \sum_{j=1}^{m}\sum_{i=(k+j-1)}^{Q} \max(x_i, y_j) p_i^x p_j^y = \sum_{j=1}^{m}\sum_{i=(k+j-1)}^{Q} x_i p_i^x p_j^y \tag{4.7}$$

若以 PERT 的觀點執行$Z = \max(X, Y)$，因 Y 變數的平均值大於 X 變數，因此所得的結果為$Z = Y$。若同樣以表 4.1 之矩陣資料結構表示，則上述灰色區域的z_{ij}值會全數轉變成$z_{ij} = y_j = \max(x_i, y_j)$，因此灰色區域便不存在，表 4.1 之矩陣資料結構可全部表示為白色區域，表示如下:

$$(\text{White area}) = \sum_{j=1}^{R}\sum_{i=1}^{Q} \max(x_i, y_j) p_i^x p_j^y = \sum_{j=1}^{R}\sum_{i=1}^{Q} y_j p_i^x p_j^y = E[Y] \tag{4.8}$$

參考圖 4.2，PERT 的結果與實際值(MCS 的結果)有一偏差出現，該偏差即為 LPB 偏差值。從上述之分析比較，可以很明顯的確定 LPB 偏差值僅與灰色區域的數值有關係。若針對將該區域分析計算，則可將 LPB 值算出。故依據式(4.6), (4.7)及(4.8)，LPB 之計算可以表示為:

$$LPB = E(Z) - E(Y) \tag{4.9}$$

灰色區域位於表 4.1 資料結構之右上部分，這也是本論文將其稱之為「區域積分(Marginal Integration)」之故。為方便計算推導，仍將 Y 變數依據式(4.6)及(4.7)之格式，表示為:

$$\begin{aligned}Y &= (\textit{White area}) \ + \ (\textit{Grey area}) \\ &= \sum_{j=1}^{(m-1)}\sum_{i=1}^{(k+j-2)} y_j p_i^x p_j^y + \sum_{j=k}^{R}\sum_{i=1}^{Q} y_j p_i^x p_j^y + \sum_{j=1}^{m}\sum_{i=(k+j-1)}^{Q} y_j p_i^x p_j^y\end{aligned} \tag{4.10}$$

式(4.6), (4.7)及(4.10)代入(4.9)式，LPB 可以改寫成:

$$\begin{aligned}LPB &= \sum_{j=1}^{m}\sum_{i=(k+j-1)}^{Q} x_i p_i^x p_j^y \ - \ \sum_{j=1}^{m}\sum_{i=(k+j-1)}^{Q} y_j p_i^x p_j^y \\ &= \sum_{j=1}^{m}\sum_{i=(k+j-1)}^{Q} \left(x_i - y_j\right) p_j^y p_i^x \\ &= \sum_{j=1}^{m} LPB\left(y_j\right)\end{aligned} \tag{4.11}$$

其中相對於每一 y_j，$LPB(y_j)=\sum_{i=(k+j\text{-}1)}^{Q}\left(x_i-y_j\right)p_j^y p_i^x,\ j=1,...,m$。

MIA 的最後步驟，是將 Y 變數機率分佈函數右移 LPB 偏差量，所得到的機率分佈函數 Z^{MIA} 即為 MIA 運算之結果。表示如下:

$$Z^{MIA}=Y-LPB \tag{4.12}$$

4.2.2 MIA 執行效率之分析

式(4.2)X,Y 變數之數值，依式(4.11)式之公式計算，可求得 LPB=0.47。因此將 Y 變數之機率分配圖形向右移動 0.47，最後 MIA 所執行的 Z=max(X,Y)結果為:

$$Z=\begin{bmatrix}4.47 & 6.47 & 8.47 & 10.47 & 12.47 & 14.47\\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6\end{bmatrix} \tag{4.13}$$

Z 的平均值與標準差分別為 9.4667 及 3.4157。若第 3.2.3 章節中之式(3.10)，實際執行 Z=max(X,Y)的運算，可得結果:

$$Z=\begin{bmatrix}4 & 5 & 6 & 7 & 8 & 9 & 10 & 12 & 14\\ 1/15 & 1/30 & 1/10 & 1/15 & 2/15 & 1/10 & 1/6 & 1/6 & 1/6\end{bmatrix} \tag{4.14}$$

Z 的平均值與標準差分別為 9.4667 及 3.0792。

雖然 MIA 所計算之標準差與實際值存在 11.61%之誤差，然而平均值確能夠得到相當準確的估算。最重要的，MIA 僅耗 6 次離散化數值單元的 $\max(x_i,y_j)$ 計算，而實際的 max 運算，則耗 30 次的 $\max(x_i,y_j)$ 計算；相較之下，MIA 共節省 80%的運算時間，當 MIA 運用於離散數值更多的變數中，執行 max 運算所節省的時間將會更可觀。

MIA 除能夠準確的估算 max 運算之 LPB 偏差值外，更能有效的減輕執行 max 運算的計算負擔。所能減少的負擔比例，與表 4.1 中灰色重疊部分的面積成反比，即兩變數機率函數重複的區域愈少，MIA 的運算愈快。若

以 3.2.3 章節第式(3.9)為定義，X 變數共 Q 個樣本值，Y 變數共 R 個樣本值，重疊區域表示為$[y_{1,}y_m]$或$[x_{n,}x_Q]$。MIA 可以節省的時間比例為:

$$(1-\frac{n\times m}{2\times Q\times R})\% \tag{4.15}$$

4.3 運用 MIA 估算隨機性專案網路完成時間

MIA 可納入 LCTA 演算法中，執行隨機性專案網路完成時間的估算，所有 LCTA 演算機制均與第三章內容相同: (1)採用離散化技術(2)採用擴張術結構(ETS)及相關之資料結構(3)專案網路路徑相依性的考量(4)執行後序追蹤演算法。唯一不同處，是在 LCTA 執行追蹤運算過程中，使用 MIA 運算元取代 max 運算。

LCTA 採用 MIA 執行隨機性專案網路完成時間的方式，本論文擬以 LCTA_MIA 表示,本章節的目的是要驗證 LCTA_MIA 對隨機性專案網路完成時間的估算的效果。首先驗證 MIA 對專案網路路徑相依性問題的處理情況，因該問題對於專案網路完成時間的估算有相當關鍵的影響。其次擬以圖 3.11 為例，離散化技術仍延用柴比契夫取樣點法，實際執行 LCTA_MIA 對專案網路完成時間估算。

4.3.1 MIA 對專案網路路徑相依性問題的處理能力

第 3.3.3 章節中以詳述專案網路路徑相依性問題對專案網路完成時間估算會造成相當大的誤差，該章節中亦以 DA 及 LCTA 為例，證明未考慮路徑相依性問題之 DA 演算法，其估算結果與 MCS 實際結果相差甚多，LCTA 則能夠有效的解決該問題。本章節亦使用同樣網路範例及數據(圖 3.4)，令 LCTA_MIA 來執行同樣的實驗，驗證 MIA 對路徑相依性處理的能力。實驗的結果如表 4.2 所列；

表 4.2 MIA 對路徑相依性處理的能力之驗證

網路節點	MCS		DA 演算法			LCTA_MIA		
	平均值	標準差	平均值 (誤差)	標準差 (誤差)	K-S 檢定 $\alpha = 0.05$	平均值 (誤差)	標準差 (誤差)	K-S 檢定 $\alpha = 0.05$
3	3.98	3.16	4.00 (0.50%)	3.16 (0.00%)	通過	4.00 (0.50%)	3.16 (0.00%)	通過
4	9.07	5.97	9.00 (0.77%)	5.92 (0.84%)	通過	9.00 (0.77%)	5.92 (0.84%)	通過
5	11.05	6.98	11.05 (0.45%)	6.98 (0.72%)	通過	11.05 (0.45%)	6.98 (0.72%)	通過
7	15.89	7.01	**17.69 (11.3%)**	6.91 (1.43%)	沒通過	**15.57 (2.01%)**	6.89 (1.71)	通過

實驗的結果很明顯的証明，LCTA_MIA 對於專案網路路徑相依性處理的能力與原 LCTA 演算是沒有差別的，節點 3 到節點 5 的輸出，因沒有路徑相依的問題存在，固結果與 DA 的結果沒有什麼不同。但節點 7 的輸出結果，DA 則有了相當之誤差出現，MIA 與 MCS 的輸出結果接近，且信心水準為 $\alpha = 0.05$ 的情況下，執行 K-S 檢定亦順利的通過。

4.3.2 LCTA_MIA 執行專案網路完成時間估算實例

若以圖 3.11 專案網路為例，該專案網路為一 100 節點之大型專案網路，若以 AON(Active on Node)模式處理並假設所有的節點時間為指數分佈函數，平均值參數如表 4.3 所列。以 LCTA_MIA 估算該專案網路完成時間的結果，得平均值 mean=123.4, 標準差 std. value=17.04，機率分佈函數如圖 4.3 所示。與 20000 次取樣之 MCS 結果比較(mean=121.79, 標準差 std. value=18.32)，LCTA_MIA 估算誤差率為平均值 1.3%, 標準差為 7%。

表 4.3　圖 3.11 之專案網路之節點平均值

1	2	3	4	5	6	7	8	9	10
1	4	6	2	1	5	9	10	3	2
11	12	13	14	15	16	17	18	19	20
9	3	7	10	7	9	1	2	9	5
21	22	23	24	25	26	27	28	29	30
9	8	7	4	2	2	2	5	9	5
31	32	33	34	35	36	37	38	39	40
9	5	5	5	5	10	1	3	1	7
41	42	43	44	45	46	47	48	49	50
7	10	6	5	2	7	8	4	1	5
51	52	53	54	55	56	57	58	59	60
8	8	10	9	4	7	8	2	10	6
61	62	63	64	65	66	67	68	69	70
7	3	6	10	4	7	4	7	7	4
71	72	73	74	75	76	77	78	79	80
5	7	9	4	5	6	6	8	6	8
81	82	83	84	85	86	87	88	89	90
5	2	8	10	9	8	5	2	7	4
91	92	93	94	95	96	97	98	99	100
2	6	9	7	10	10	1	4	6	1

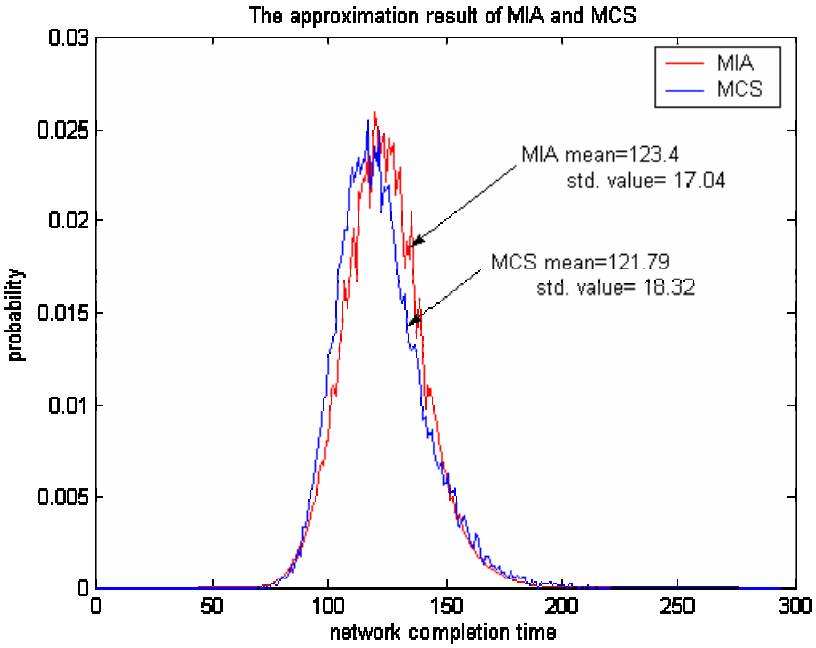


圖 4.3　LCTA_MIA 執行圖 3.11 之結果

4.4　LCTA_MIA 之專案網路實驗數據

本章節將以實驗顯示本論文所發展之 LCTA_MIA 演算法，估算隨機性專案網路完成時間的效果，證明 LCTA 優於 Dodin Algorithm(1985)及 PERT 之演算法。同第三章實驗內容之要求，LCTA 所採用的實驗範例有下列三項要求：

4. 除要求專案作業時間機率分佈函數不得為特定之機率模式。

5. 適應各種複雜度(Complexity)之專案網路結構。

6. 另須能顯示出 LCTA 針對大型專案網路的估算的效果。

針對第 1 項要求，LCTA_MIA 所使用任何的專案網路範例，所有的作業均以常態分佈函數(Normal distribution function)、指數分佈函數(Exponential distribution function)及均勻分佈函數(Uniform distribution function)三種時間機率分佈型態，隨機搭配給網路中之作業，且函數之平均時間(Mean value)及變異數值(Variance value)，亦以[1, 10]的數值範圍隨機指定給各函數。

針對第 2 項要求，LCTA 實驗所使用的範例則採用 Demeulemeester and Herroelen (1997)使用 Kolisch(1996)的 ProGen 軟體產生的，該文獻以 full factorial 實驗設計方式來產生專案網路結構實體(Instance)，該實驗設計所

使用的專案網路作業數量均設為 32 個，所考量到的實驗因子(Factors)有三個：專案網路複雜度參數(Complexity)(水準為 1.5, 1.8, 2.0，水準係數愈大專案網路結構愈複雜)，每個網路節點之輸入節點(Predecessors)及輸出節點數(Successors)亦均隨機決定，其範圍為[1,3]之間。依上述實驗設計的方法及使用的參數取得 LCTA_MIA 實驗所需之 120 個專案網路實體(每一專案網路複雜度水準均分配 40 個網路實例)，這也是本章節 LCTA_MIA 執行的第一組實驗，實驗內容將比較 LCTA_MIA、DA 及 PERT 三種演算法，比較內容為估算執行時間、完成時間平均值及完成時間標準差值三項。所有實驗會以 20000 次樣本值之蒙地卡羅模擬法(MCS)所得之結果來作為演算法比較的基準值，該 MCS 模擬之結果會趨近於真實結果。本論文使用 t 分配、信心水準 0,999 的情況下，並利用第二組實驗 20 個大型專案網路為範例，驗證 20000 次樣本值之 MCS 結果，確實能夠接近真實值(與真實質平均誤差 0.36%)，相關之說明如附錄 G 所列，其它相關之文獻，如 Burt & Garman(1971), Dodin(1985), Sigal(1979)亦以大數量樣本之 MCS 結果作為演算法驗證的標準。

針對第 3 項要求， LCTA_MIA 將進行本章節的第二組實驗。LCTA_MIA 實驗所使用的範例則由本論文採用 Kolisch(1996)的 ProGen 軟體，依據上述同樣之專案網路條件，產生 20 組 100 個節點(作業數量約 130~160 之間)的大型專案網路。實驗的對像及實驗的內容同第一組實驗，同樣以 20000 次樣本值之蒙地卡羅模擬法(MCS)之所得結果比較的基準值。第二組實驗為了要進一部驗證 LCTA_MIA 所估算出來的專案網路完成時間機率分佈函數是否與真實的結果(MCS 結果)有差異，故針對每一大型專案範例，額外執行 K-S 檢定(Kolmogorov-Smirnov test)。

專案作業時間機率分佈函數在投入實驗前，均會將其實施離散化處理，因此實驗前必須先決定作業時機變數之取樣點數。取樣的點數愈大則運算之結果愈精確，但會犧牲演算之時間。LCTA_MIA 採用柴比契夫取樣點法，因取變數前 4 個動差值(Moment value)之故，乃設定取樣點數為 5。本章節實驗所使用的個人電腦為 Intel Pentum-4 2.0 GHZ CPU、512MB 記憶體並使用 Matlab Ver.6.5 語言程式來執行。

4.4.1 LCTA_MIA 第一組實驗

複雜度愈高的專案網路，執行 max 運算的機會愈多，對於估算結果的精度影響愈大。因此 LCTA_MIA 第一組實驗的目的主要是驗證 LCTA_MIA 對於不同水準之專案網路複雜度，估算結果的比較。同時亦以 DA 演算法及 PERT 來執行相同專案網路範例的估算，並比較不同演算法間的結果，來證明 LCTA_MIA 估算隨機性專案網路完成時間的有效性。本組實驗所使用的專案網路範例是採 Demeulemeester and Herroelen (1997) 使用 Kolisch(1996)的 ProGen 軟體所設計的，專案網路複雜度以三種水準設計(水準為 1.5, 1.8, 2.0)，每種水準 40 個共計 120 個，作業數量均為 32 個。三種演算法綜整後的實驗結果如表 4.4 所列，演算法的結果會與 20000 次樣本之 MCS 模擬的結果作比較，比較的內容包括執行時間倍數比 $\frac{MCS}{method}$ (MCS 與演算法的執行時間比值)、平均值相對誤差 $\frac{|method-MCS|}{MCS}$ (演算法與 MCS 平均值之誤差百分比)及標準差相對誤差 $\frac{|method-MCS|}{MCS}$ (演算法與 MCS 標準差之誤差百分比)。

表 4.4　LCTA_MIA、DA 及 PERT 對不同專案網路複雜度估算結果之表較

專案網路複雜度	估算結果與 MCS 模擬結果之相對誤差 *method*：LCTA_MIA、DA 及 PERT								
	執行時間倍數比 $(\frac{MCS}{method})$			平均值相對誤差 $(\frac{\|method-MCS\|}{MCS})$			標準差相對誤差 $(\frac{\|method-MCS\|}{MCS})$		
method	*LCTA*	*DA*	*PERT*	*LCTA*	*DA*	*PERT*	*LCTA*	*DA*	*PERT*
1.5	797.93	20.81	3203	3.33%	6.84%	12.42%	19.6%	7.35%	25.23%
1.8	859.68	12.69	3558	3.47%	8.32%	14.93%	21.4%	4.33%	24.43%
2.0	836.43	8.96	3868	2.80%	3.64%	13.90%	22.7%	14.94%	22.29%
average	831.35	14.15	**3543**	**3.20%**	6.26%	13.75%	21.2%	**8.87%**	23.98%

表 4.4 的結果很明白的顯示，LCTA_MIA 綜合的表現優於其它演算法，詳細的比較結果說明如下：

5. 雖執行的速度最快的為 PERT，然而其平均值誤差(13.75%)及標準差誤差

(23.98%)值是最大的。

6. LCTA_MIA 在標準差誤差表現雖遜於 DA(21.2% vs. 8.87%)，但其平均值誤差比較好(3.2% vs. 6.26%)。
7. LCTA_MIA 的速度的表現亦比 DA 快非常多(831.4 vs. 14.2)，亦比第三章之 LCTA(未使用 DRT)快(831.4 vs. 310)。

4.4.2 LCTA_MIA 第二組實驗

本組實驗乃驗證LCTA_MIA在大型專案網路的環境中估算隨機性專案網路完成時間之成效。一般相關文獻並無提供專案作業數量大於 100 以上之專案網路範例，因此本論文同第一組實驗之範例，依據 Kolisch(1996)對專案網路之規格要求，自行寫程式隨機產生 20 個 100 個節點(作業量130~160)之大型專案網路範例。

第二組實驗的內容及方式，同第一組實驗，針對 LCTA_MIA、DA 及 PERT 三種演算法並與 MCS 模擬之結果作比較，比較的內容同樣為估算的平均值相對誤差 $\frac{|method - MCS|}{MCS}$ 、標準差相對誤差 $\frac{|method - MCS|}{MCS}$ 及執行時間倍數比 $\frac{MCS}{method}$ ，唯第三章 LCTA 的實驗內容，包括採納重複取樣技術(DRT)部分，故本組實驗亦將 DRT 技術納入 LCTA_MIA，來比較其結果。相關的結果如表 4.5 所列。除上述比較的內容外，本組實驗另採用 K-S 檢定法，進一步檢驗 LCTA_MIA 所估算隨機專案網路完成時間機率分佈函數的形狀，與實際情況之差異，該檢驗亦同樣的應用於其它兩個演算法上。K-S 檢定法與檢驗的樣本數息息相關，樣本數愈大，檢驗的標準會愈高，受檢對象愈不容易通過檢驗。本組實驗以信心水準 $\alpha = 0.01$，採三種樣本數(30, 40 及 50)來執行 K-S 檢定，相關的檢驗結果亦綜整於表 4.5 中，詳細的檢驗內容如附錄 I 所列。

表 4.5 的結果亦很明白的顯示，LCTA_MIA 對於大型專案網路綜合的表現較第一組實驗更為突出，詳細的比較結果說明如下：

1. 雖執行的速度最快的仍為 PERT，然而其平均值誤差提升至 24%。
2. DA 的整體表現在大型專案網路中仍最差(平均值及標準差誤差分別為 26% , 43%)，LCTA_MIA 正與其相反，平均值及標準差誤差均優於 DA。

表 4.5　LCTA_MIA、DA 及 PERT 對大型專案網路估算結果之比較

比較項目	MCS	LCTA_MIA w/o DRT	LCTA_MIA w/ DRT	DA w/o DRT	DA w/DRT	PERT
執行時間倍數比 $(\frac{MCS}{method})$	2179	1274.3	2867.1	15.9	789.5	**18158.3**
平均值相對誤差 $(\frac{\lvert method-MCS\rvert}{MCS})$	-	**2.35%**	2.42%	26.55%	25.46%	23.99%
標準差相對誤差 $(\frac{\lvert method-MCS\rvert}{MCS})$	-	**9.25%**	12.73%	43.54%	18.52%	12.19%
K-S 檢驗 $\alpha=0.01$, K=30	-	**20/20**	**20/20**	0/20	0/20	0/20
K-S 檢驗 $\alpha=0.01$, K=40	-	18/20	19/20	0/20	0/20	0/20
K-S 檢驗 $\alpha=0.01$, K=50	-	16/20	17/20	0/20	0/20	0/20

3. LCTA_MIA 的表現在大型專案網路中表現比第一組實驗來得好，執行速度非常快，尤其是平均值誤差仍維持在 2.35%之水準，標準差誤差下降至 9.25%。
4. LCTA_MIA 若採用 DRT 技術，平均值誤差仍維持在 2.35%之水準，標準差誤差略提升至 12.7%，速度也增加一倍，然而整體的表現仍未如第三章 LCTA 採用 DRT 技術的結果。
5. K-S 檢驗部份，LCTA_MIA 在樣本數 30 的情況下 20 個專案網路範例全數通過檢驗，樣本數 40 及 50 亦大部份通過檢驗。然而 DA 及 PERT 的檢驗則全部均未通過檢驗。

從上述兩組實驗非常明顯的顯示，本論文所發展之 LCTA_MIA 演算法，估算隨機性專案網路完成時間機率分佈函數之表現，優於最受青睞的工具 PERT 及文獻中之 DA 演算法。LCTA_MIA 的特色就是在未使用 DRT 技術的前提下，其整體表現(Performance)是本論文所討論的演算法中最好的，因此本章節的實驗數據證明 LCTA_MIA 運用於大型的專案網路之可行性。

4.5 結論

無論使用任何文獻之演算法，估算隨機性專案網路完成時間機率分佈函數，均須將執行專案網路節點間之 max 運算，其所產生之最長路徑偏移(LPB)納入估算的結果，否則估算的準確度會受相當的影響，且該偏差會隨專案網路規模的成長而增大。LCTA 所處理的路徑相依問題亦源於對 LPB 的考量，故專案網路完成時間之估算過程中，max 運算因 LPB 偏移所產生之誤差，會遠比離散化技術累算過程中，內部所產生之誤差量來得高也來得重要。另離散化之 max 運算，有其演算的複雜度，在估算的過程中亦較耗時，因此本章節所提出之區域積分演算法(MIA)，便是針對上述問題設計的，直接將 LPB 偏移量以公式推導出來，在不影響精確度的情況下，能夠節省 max 運算的時間，其所節省的時間則依執行 max 運算之變數變異數值而定，變異數值愈大，兩變數之機率分佈函數重疊的範圍及機率愈大，則節省的運算時間會愈多。故本章節接續以 MIA 取代離散化之 max 運算，並與 LCTA 結合成 LCTA_MIA 演算法，進行隨機性專案網路完成時間機率分佈函數之估算工作。

LCTA_MIA 在進行實驗之前，先針對專案網路路徑相依問題做了測試，驗證其能夠順利處理該問題；另採用 100 個節點的大型網路實例，實際驗證 LCTA_MIA 之估算能力，情況亦非常理想。最後本章節以第三章 LCTA 同樣規格之實驗內容，對 LCTA_MIA 作實驗，從實驗數據內容得知，LCTA_MIA 處理 150 個作業數量之大型專案網路，其平均值之誤差值仍可維持 2.4%的準確度，相對於 DA 及 PERT 之估算誤差值，動則便超過 20%以上，實無法與 LCTA_MIA 相比較。LCTA_MIA 估算標準差則為 9.3%，除低於 DA 及 PERT 之估算誤差值，更比 LCTA 還低。LCTA_MIA 最主要的貢獻仍是其快速的運算速度，未配合使用重複取樣技術(DRT)的情況下，處理 150 個作業數量之大型專案網路，平均為 MCS 模擬法 1274 倍之多，相較於 DA 之 15 及 LCTA 之 318 快了非常多。

LCTA_MIA 所估算出來的完成時間 *pdf*，本章節亦以 K-S 檢驗法來執行檢驗，在信心水準 $\alpha = 0.01$，以樣本數為 30 的情況下，針對上述之大型專案網路，LCTA_MIA 同 LCTA 演算法一樣，全數通過檢驗。相對之 DA 及 PERT 演算法，在同樣的檢驗條件下，沒有一個通過檢驗。本論文採用數量甚多的專案網路實例，且經實驗結果證明，LCTA_MIA 對於隨機性專案網

路完成時間之估算，確實是一有效率的估算工具。

第五章　隨機性專案網路最佳化資源分配問題之處理

本章節乃處理隨機性專案網路最佳化資源分配問題(Stochastic network Resource Allocation Problem, SRAP)，其中專案採單種類資源投入採改良式磁場機制演算法來作為資源分配的決策工具，而多種類資源投入則採基因演算法來作為資源分配的決策工具，擬分兩個子章節依序說明如下文。

5.1　運用磁場機制演算法解單種類資源 SRAP 問題

隨機性專案網路環境中，外部作業資源的投入量對於專案完成時間有著相當關鍵的影響，因此在一個隨機性的專案網路環境中，針對每一項作業進行有限制之資源分配以求最小化完成專案之總成本，該類問題我們稱之為隨機網路資源分配問題(SRAP)。根據 Elmaghraby(1992)、Tereso, et al. (2004a)所述，作業隨機性之來源主要有兩部分：第一為「外在因子」因素所造成，例如氣候因素、裝備不良率因素、人員到勤狀況因素等；第二部分為「內在因子」因素所造成，此部分常為專案管理人所忽略，以至未能獲得通盤之瞭解。所謂「內在因子」之影響，部分來自於作業本身之「工作內涵」（Work content）或「能力」(Effort)所致，例如研發(R & D)工作之專案人員素質、研發能力等內在因素，該因素往往是專案執行順利與否之關鍵。然而這些內在因子往往難予以量化之，但是亦不難理解其與所提供作業外部資源之種類及數量多寡有關，因此外部資源的投入雖能夠縮短專案作業的時間(質好及質精的研發人才確實可以加快研發的腳步)，但也會增加專案的成本(薪資要求較高)，從上述內容中可以清楚的瞭解到作業資源的分配與作業的執行時間存在著一種取捨關係(Trade-off relation)。在這種情況下，專案負責人員對於內在因子的掌握，可將焦點轉移到外在資源的分配上，以求得最佳專案之總成本。

就本論文所瞭解，已有若干的學者針對 SARP 提出解決方式，並分為三類型：數理規劃解法、模擬解法及啟發式解法。針對前兩類的方法，若專案網路作業數量增加，則會因龐大的數理計算而無實際的應用價值。因此解決 SARP 最實際有效的方法為上述第三類啟發式解法型態，該類方法所使用的決策工具，較為一般文獻所採用不外乎為退火消去法(SA)、禁忌搜尋法(Tabo search)、基因演法(GA)及蟻群複製最佳化法(Ant Colony Optimization)等。近年來 Birbil & Fang(2003)所提之磁場機制演算法

(ElectroMagnetism Algorithm, EM)亦為一啟發式解法，因理論架構完整且程序簡單較易於程式化之故，非常適合應用於專案網 SRAP 問題。Terreso , et al.(2004b)即利用 EM 解決 SRAP 問題，然該研究所使用的 EM，有兩個問題須待解決:第一個問題是當專案網路結構擴增、作業數量增加之際，EM 亦同其他上述啟發式解法一樣，遭遇決策變數求解範圍過大的問題，以致於嚴重影響該方法之效率。第二個問題是該研究採用計算量龐大的蒙地卡羅模擬法(MCS)執行專案網路的完成時間之估算，進而求出專案的總成本。使用 MCS 模擬法會因計算負荷過重之問題，讓 EM 解 SRAP 變得耗時沒有效率。

本章節擬針對第一個 EM 問題作改進，所提出之改進方法有兩項：一為找出適當的 EM 起始解以縮短 EM 尋求最佳解之收斂時間，二則是運用有效率的區域搜尋演算法(Local Search Algorithm)擺脫可能發生的區域最佳解問題。本文針對隨機專案網路之特性做分析，提出一關鍵路徑叢集演算法(Critical Path Cluster Algorithm, CPCA)，找出專案網路 SRAP 問題適當之起始解，讓 EM 收斂的速度增快；另亦利用關鍵路徑叢集(Critical Path Cluster)的概念執行叢集區域搜尋演算法(Cluster Local Search Algorithm, CLSA)，有效且迅速的找出 EM 各搜尋解的改良值。針對第二個問題，本章節採用本論文第三章所提之標籤修訂演算法(LCTA)來執行隨機專案網路完成時間之估算，以取代分析推導法或 MCS 模擬法，當完成專案網路完成時間之計算後，專案總成本即可求出。本章節令專案總成本包含所有作業資源使用成本(Resource usage cost)及專案完成時間因逾時之懲罰成本(Penalty cost)兩項。其中作業資源之使用成本視作業所分配之資源量而定，資源的投入同時亦會影響作業的執行時間。而懲罰成本則與專案所規定之完成期限，與實際專案完成時間之差異量有關。

本章節內容共包含五個子章節，第一子章節依序介紹 SRAP 之問題定義、投入資源與作業時間之關係及專案網路總成本之計算模式。第二子章節則介紹磁場機制演算法(EM)之原理及執行步驟，第三子章節介紹針對 EM 處理 SRAP 問題所做之改進，其中主要是採用了關鍵路徑叢集演算法(CPCA)及叢集區域搜尋演算法(CLSA)，將詳細說明其概念及其演算的方式。第四子章節會以 Terreso, et al. (2004b)所提出來之 14 個網路實體為實驗對象，該文獻採用 EM 並配合蒙地卡羅模擬法求解 SRAP 問題。本章節擬

針對第 14 個網路實體為實驗對象，利用本文所提之方法執行 SRAP 求解，再針對所有的網路實體為對象，與 Terreso, et al. (2004b)所提之方法做完整之比較。第五子章節則提出本章節之結論。

5.1.1 專案單種類資源分配問題之定義

本章節將討論於隨機性專案網路架構下，作業資源投入相關之決策變數及問題定義，本論文假設影響作業執行時間之「內在因子」為「作業內涵」(Work content)，並以「人力工時」(Man-time)為單位表示之，且先考慮單一種類內在因子，即整個作業網路僅考慮使用一種資源(即人力)，該資源的投入量多寡會影響作業執行的時間長短，本章節會以一圖例來說明此情況；而專案總成本包含作業資源使用成本及懲罰成本，故與資源使用量及專案完成時間息息相關，其間的關係亦會以方程式來說明之。

5.1.1.1 作業資源的投入

專案網路作業執行時間 Z 本身即為一變數，以指數分配(Exponential distribution)型態為例，λ 為其參數：

$$Z = \lambda r \mathrm{e}^{-\lambda r t},\ mean(Z) = \frac{1}{\lambda \cdot r} \tag{5.1}$$

故資源的投入量 r 相當於改變指數分配之參數值，相對的亦改變了作業的執行時間機率分佈及其平均時間。本文對作業資源作另一項假設就是限定其投入之上、下限範圍，即令 $r \in [0.5, 4]$。當 $r < 1$ 表示減少人力的投入，故會增加作業的時間，若 $r > 1$ 則表示增加人力會減少該作業之時間，若 $r = 1$ 則表示作業投入的人力沒有增減的意思。對於資源的投入與作業時間之關係，本節更以圖 5.1 來做更清楚之說明：

圖 5.1 分別針對作業 $Z = 0.125 r \mathrm{e}^{-0.125 r t}$ 投入 0.5 及 2 不同的資源來做一比較，發現 r=0.5 的情況下，作業欲於 8 個時間單位內完成的機率為 40%；而 r=2 的情況下，該作業於同樣的時間下完成之機率相對提升至 86%。

爲讓讀者充分瞭解本章節專案網路 SRAP 問題之定義，以一 8 個作業之專案網路圖例來做說明(參考圖 5.2)，圖中所有的作業時間 $Z_j, j = 1,...8$ 均假設為指數分配，分別表示如下：

$$Z_1 = 0.25e^{-0.25t}, \quad Z_2 = 0.25e^{-0.25t}, \quad Z_3 = 0.25e^{-0.25t}, \quad Z_4 = e^{-t}$$
$$Z_5 = 0.166e^{-0.166t}, \quad Z_6 = e^{-t}, \quad Z_7 = 0.166e^{-0.166t}, \quad Z_8 = 0.333e^{-0.333t} \tag{5.2}$$

作業 1 至 8 之時間平均值分別為{4, 4, 4, 1, 6, 1, 6, 3}，$r_1,...,r_8$ 為作業之資源投入量，且令其初始值均為 1，資源投入之上、下限範圍為 $r \in [0.5,4]$。

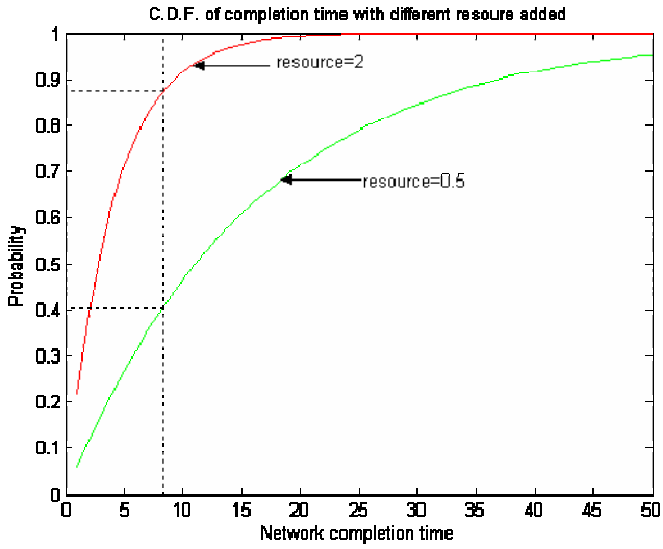


圖 5.1 資源投入對作業完成時間的影響

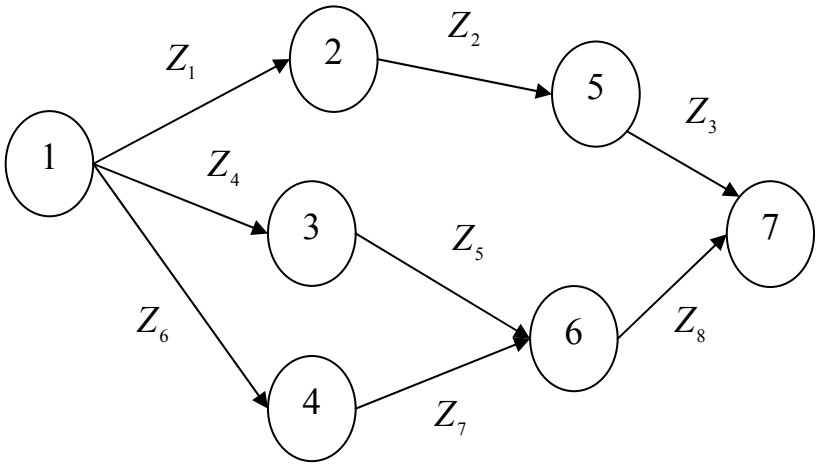


圖 5.2 8 作業之專案網路圖例

5.1.1.2 專案成本的計算

專案網路總成本包括兩成本項目：

1. 作業資源使用成本：假設每一單位資源使用成本$\$U_C$，並定義該作業成本$C_{R_i}$等於(人力工時$W_i$)・(資源投入量$r_i$)・$U_C$，表示式如下：

$$C_{R_i} = W_i \cdot r_i \cdot U_C = r_i^2 \cdot Z_i \cdot U_C \quad \text{其中} \quad Z_i = \frac{W_i}{r_i}, i = 1,\ldots,8 \tag{5.3}$$

2. 作業時間的逾時成本：當專案完成時間平均值$E[\Upsilon_N]$超過了所規定的限制時間T_z，則會產生一懲罰成本即為專案之逾時成本，本文假設該單位時間成本為$\$P_C$，故其表示式為：

$$\begin{aligned} &C_P = (E[\Upsilon_N] - T_z) \cdot P_C, \ E[\Upsilon_N] > T_z \\ &C_P = 0, \qquad\qquad\qquad otherwise \end{aligned} \tag{5.4}$$

由上述可以得知專案網路的總成本C_T為所有作業資源使用成本及作業時間的逾時成本之和，因此可表示為：

$$C_T = \sum_{j=1}^{8} C_{R_j} + C_P = C_U + C_P \tag{5.5}$$

上述Υ_N為隨機性專案網路完成時間機率分佈函數，欲解 SRAP 問題必須先將該函數估算出來，本章節是採用標籤修訂演算法(LCTA)估算之。最後 SRAP 的目的就是要決定最佳的作業資源投入值$r_1,\ldots,r_8$，得到最小的總成本C_T。

5.1.2 磁場機制演算法對 SRAP 之應用

磁場機制演算法(EM)是由 Birbil & Fang(2003)所發展出來的，亦屬隨機全域最佳化中群體基礎演算法(Population-Based Algorithms)之一種，其特色是先從可行區域中隨機抽取樣本點，根據樣本點的目標函數值，決定那些地方是具有吸引力的區域，接下來針對這些候選區域使用某種機制作更深入的搜尋。EM 利用類似基本電磁學的原理，將每個樣本點都想成是散佈在空間中的一個帶電粒子(Charged particle)利用粒子間彼此之吸引力／排斥力的機制，將點吸引到深谷中。在我們的方法中，每個點所帶的電荷相關於目標函數值，這正是我們要最佳化的目標。此電荷同時決定了該點對其他樣本點吸引力或排斥力的大小:目標函數值愈大，吸引力愈強。計算電

荷後，接著尋找每個點的移動方向：藉由計算其他點作用於(Exerted)該點之力的總和來選擇方向。EM 的機制可以應用於 SRAP 問題上，令專案網路作業數量為 N，則其資源投入組合 $\{r_1,...,r_N\}$ (Resource Allocation Pattern) 可以視為 EM 中一個帶電粒子，該粒子所帶的電荷量即為該資源投入組合 $\{r_1,...,r_N\}$ 投入專案後，專案網路之總成本。粒子亦為一向量，令專案網路作業數量為 N，該方向的每一維度(Dimension)即由 $r_i, i=1,...,N$ 來決定。因此專案網路作業數量愈多，則方向維度愈高，同時 EM 搜尋的範圍也會增大。詳細的 EM 執行步驟擬於下文敘述。

5.1.2.1 磁場機制演算法之架構

磁場機制演算法(EM)同基因演算法(Gentic Algorithm, GA)一樣，屬多點搜尋之啟發式演算；其搜尋最佳解過程的演算架構共分為四個階段(Phases)，分別為演算法的初始化(Initialization)、計算由各點施力的總和力(Total force)、根據該力之方向作移動(Movement)，和應用鄰域搜尋(Neighborhood search)尋找局部最小值，其一般架構表示如下：

EM_Algorithm：

m：樣本點的數目

MAXITER：迴圈的最多次數

LSITER：局部搜尋的最多次數

Initialize()

iteration ← 1

While iteration ＜ *MAXITER* **Do**

Local(*LSITER*, λ)

F ← CalcF()

Move(**F**)

iteration ← *iteration* + 1

End while

本章節之 EM 是擬解 SRAP 問題，因此 SRAP 所使用到之決策變數亦

會同時納入 EM 演算架構中說明。

2. 初始化的程序 Initialize()：

此程序用來從 N 維空間(Hyper-cube)的可行區域(Feasible domain)中抽取 m 個樣本點 $X^i, i=1,...,m$。根據 SRAP 之問題定義，可將上述專案網路中資源投入組合 $\{r_1,...,r_N\}$ 視為一樣本點，因專案網路的作業數量為 N，則 EM 中樣本點的維度(Dimension)即為 N，並令樣本點表示為：

$$X^i=\{x_1^i,x_2^i,...,x_n^i\},\quad i=1,...,m \tag{5.6}$$

每一個維度(Dimension)均有規定其上界至下界之空間 $[L_k^i,U_k^i],\ i=1,...,m,\ k=1,...,N$，樣本點內每一維度值 x_k^i，在 EM 演算的過程中，不得超出該範圍。EM 初始化的程序便是在 $[L_k^i,U_k^i]$ 內隨機取樣，決定每一樣本中之所有 x_k^i 值。待決定所有 m 個樣本點後，再利用函數 $f(x)$ 計算樣本點的目標函數值。SRAP 即利用 LCTA 針對每一樣本點值 X^i，估算相對之專案網路完成時間，在藉以計算出專案總成本作為目標函數值 $f(X^i),\ i=1,...,m$。EM 初始化的程序在上、下界範圍內所產生的樣本點示意圖，可參考圖 5.3 所列。

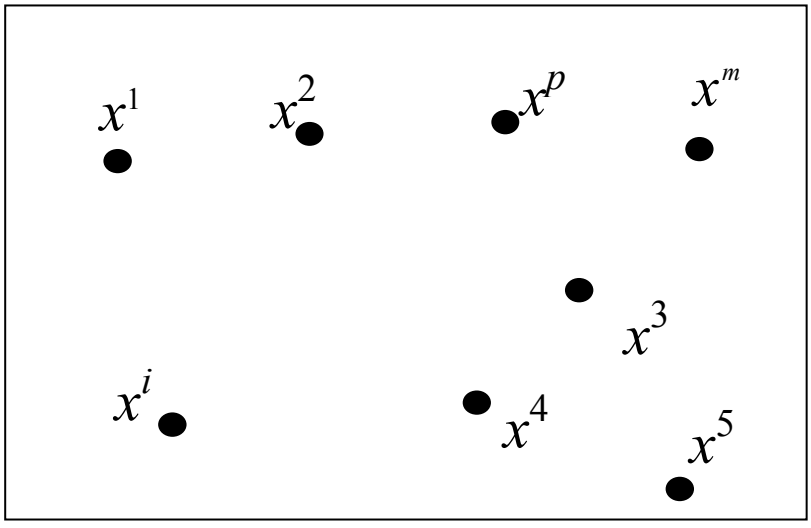


圖 5.3　EM 初始化的程序產生的樣本點

2. 力之向量總合計算 CalcF()：

樣本點在 EM 演算法中，被視為一帶電荷的粒子，電荷量之大小與其相

對應之目標函數值成正比，即樣本點是根據目標函數值來計算該點的電荷。電荷的計算方式是參考 EM 在每一代(Generation)所產生之 m 個樣本群體之目標函數值，經過正規化(Normalized)的計算而產生的，表示如下式：

$$q^i = \exp\left(-n \frac{f\left(X^i\right) - f\left(X^{best}\right)}{\sum_{k=1}^{m}\left(f\left(X^i\right) - f\left(X^{best}\right)\right)} \right),\ i = 1,...,m \tag{5.7}$$

X^{best} 為 m 個樣本群體中目標函數值最好的樣本點，即專案網路總成本最小的資源投入組合。

然不同於真正電子的電荷，EM 樣本點的電荷不加正、負號，樣本點間相互吸引或排斥(Attract/Repulse)的機制，是由兩樣本點間之目標函數值執行比較後再決定的，兩點間具有較佳目標函數值的點會吸引另外一點；相反地，具有較差目標函數值的點會排斥另一點。若以兩個樣本點 X_a, X_b 為例，其目標函數分別為 $f(X_a), f(X_b)$，若 $f(X_a) < f(X_b)$，根據(5.7)式 $q^a > q^b$，這也表示樣本點 X_a 會吸引 X_b；相對的，樣本點 X_b 會被推向樣本點 X_a。這樣本點間吸引及推向的力量(Force)之計算，乃根據電磁學理論的庫倫定律(Coulomb's law)：由其他點作用於一點的力，與兩點間的距離成反比，與兩電荷的乘積成正比(Cowan, 1968)。又每一樣本點均會與其它樣本點產生吸引及推向之作用，因此針對樣本點 i，其總和力 F^i 的計算，以下述之方程式而得：

$$F^i = \sum_{\substack{j=1 \\ j \neq i}}^{m} \left\{ \begin{array}{l} \left(X^j - X^i\right) \dfrac{q^i q^i}{\left\| X^j - X^i \right\|^2} \ \ if\ f\left(X^j\right) < f\left(X^i\right) \\ \left(X^i - X^j\right) \dfrac{q^i q^i}{\left\| X^j - X^i \right\|^2} \ \ if\ f\left(X^j\right) \geq f\left(X^i\right) \end{array} \right\} \tag{5.8}$$

(5.7)式中之樣本點 X^{best}，因為最小的目標函數值，就具有絕對吸引力的點，群體中所有的點會向傾向該點的位置，EM 即利用此機制，讓所有的樣本點最終會收斂至最佳樣本點位置。對樣本點 i 而言，產生的總和力 F^i 之示意圖如圖 5.4 所列。

3. 根據總和力移動：

力之向量總合決定了樣本點的移動方向，然而移動的距離如何決定?且樣本點移動的過程中亦要考慮不能超過 EM 可行解的範圍。以樣本點 i 為例，計算完總和力 F^i 後，該點便往總合力的方向移動一個隨機步距，如式(5.9)所示：

$$X^{i'} = X^i + \alpha \cdot RNG \cdot \frac{F^i}{\|F^i\|}, \ \alpha \in (0,1]$$
$$RNG = \min[\{s_k \mid s_k = (U_k^i - x_k^i),\ k = 1,...,n\},\ \{p_k \mid p_k = (x_k^i - L_k^i),\ k = 1,...,n\}] \tag{5.9}$$

該隨機步距由 RNG 及 α 決定。RNG 則是決定樣本點內每一維度最大的移動範圍，即每一維度 k 移往上界 U_k^i 或下界 L_k^i 的允許間距值。為了確保樣本點均有機會沿著該方向移往未曾拜訪的位置，因此 α 為 $(0,1]$ 中以隨機選取該步距。$\frac{F^i}{\|F^i\|}$ 則是一單位向量，決定了樣本點移動的方向。透過(5.9)式，樣本點 X^i 最後會移動到新位置 $X^{i'}$ 處，相關的示意圖如圖 5.5 列。

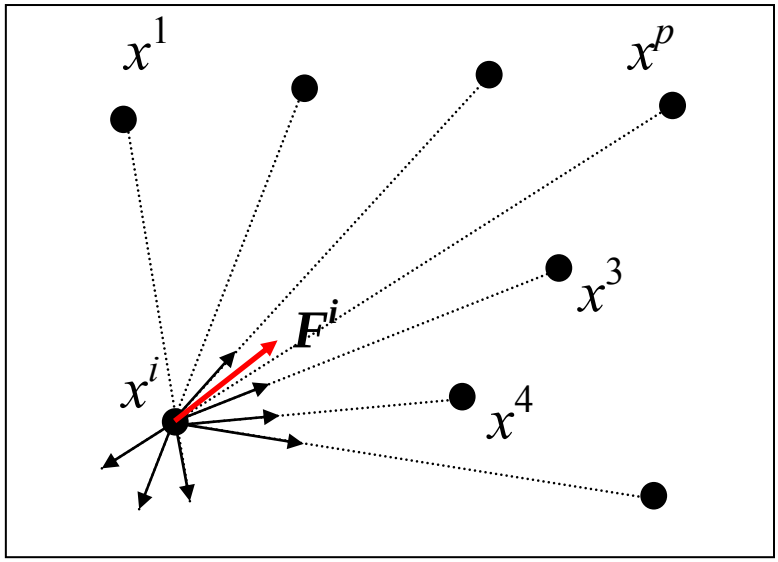


圖 5.4　EM 樣本點 i 的總和力 F^i 示意圖

4. 區域搜尋 Local(*LSITER*, λ)：

EM 在每一代(Generation)執行的過程中，若出現樣本點目標函數值 $f(X^{best})$ 比其它樣本點目標函數值有相當之差距，這會使 EM 的求解過程快速收斂，而產生區域最佳解的問題。因此 EM 就必須加入一機制，擴大增加樣本點搜尋視野，減緩樣本點朝單一方向收斂集中。EM 便在每一代的

執行過程中，針對全部或部分之樣本點執行區域搜尋(Local Search)，此程序適用來蒐集樣本點 X^i 的鄰域點，有無出現比該樣本點更好的目標函數值，若發現則以該發現的新點來取代之。參數 *LSITER* 和 λ 分別代表鄰域搜尋的迴圈(*Iteration*)次數及樣本點決定最大可行步距(*Length*)，相關的執行示意圖如圖 5.6 所列。

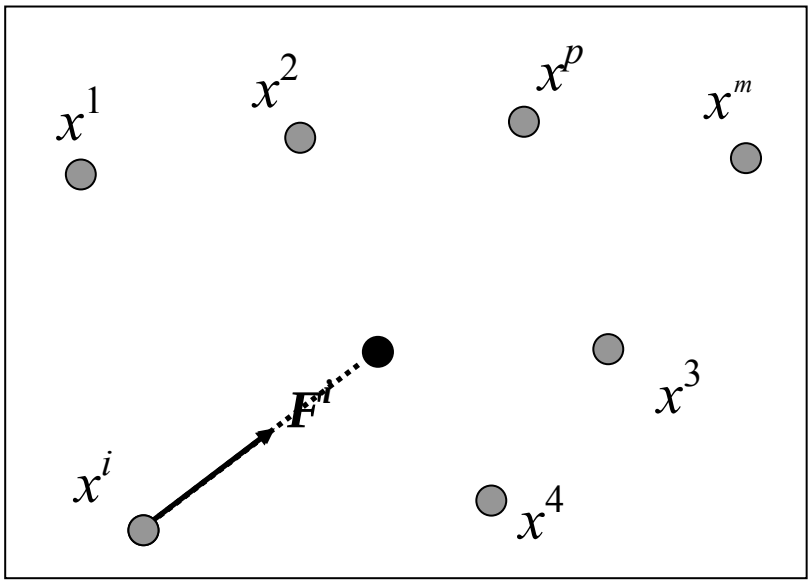


圖 5.5　EM 樣本點 *i* 移動的示意圖

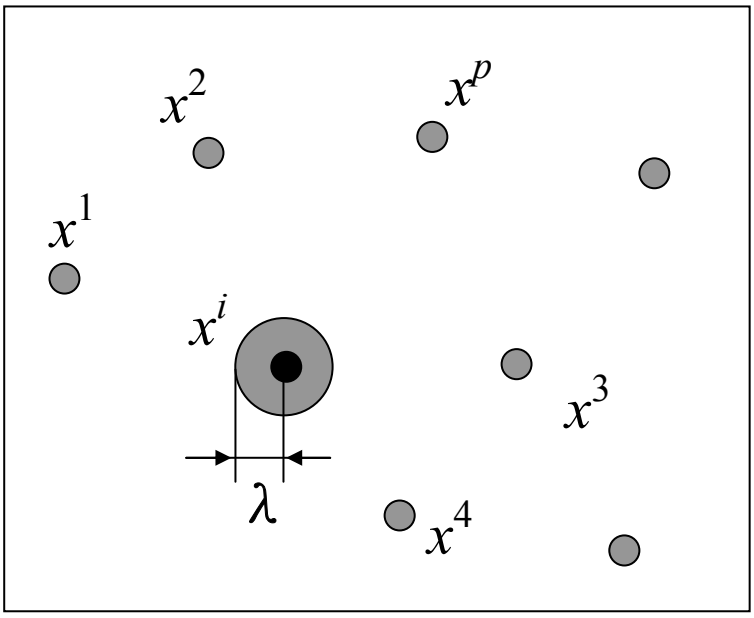


圖 5.6　EM 樣本點執行區域搜尋示意圖

區域搜尋程序有許多的方法可選用，亦會因 EM 所處理問題的對象不同

而有不同的效果，其所考慮的因素主要是當區域搜尋執行的樣本數過多或次數過度頻繁，則會影響 EM 之效率，因此合適的區域搜尋對於 EM 是有加乘之效果。

EM 同 GA 演算法一樣，當執行至 *MAXITER* 代數後或所得到最好的解已無太多進步的空間時，即停止執行，此時所有的樣本點亦會集中在最好的目標函數值之樣本點處附近。

5.1.2.2 磁場機制演算法之改進措施

Bibil (2004)證明了 EM 所產生的取樣點確實能夠向問題最佳解處收斂，然從 4.1.2.2 章節 EM 的演算流程中不難看出，EM 的初始樣本點及區域搜尋演算決定 EM 的效率(即其收斂速度)。EM 每一決策變數即代表一維度(Dimension)，每一維度值有其上、下界範圍，因此 EM 之樣本點能夠移動的範圍會非常廣，致使 EM 在搜尋最佳解的過程中產生非常大的負擔。以 SRAP 問題為例，專案網路中針對每一個作業所投入的資源量即為一個決策變數，各有其上、下界值，即便處理一作業量為 20 的小型的專案網路，樣本點移動的範圍就必須考量 20 個維度，縱使假設每個維度範圍僅分成 2 個區域，對 EM 內任一樣本點而言仍會有 2^{20} 種選擇，對 EM 搜尋最佳解而言是很大的負擔，更遑論中、大型專案網路。但若樣本點接近最佳解的位置，則透過 EM 演算法搜尋機制的特性，樣本點可以很快的移動到最佳解的位置，因此 EM 的初始樣本點位置，便成了 EM 找尋最佳解的關鍵因素。另 EM 每個樣本點移動距離的決定，必先要計算出各樣本點的「電荷量」，而電荷量的計算與問題本身的目標函數值有關，SRAP 問題的目標函數值為資源投入後之專案總成本，須先求出隨機專案網路之完成時間。就本論文第三章的內容得知，隨機性專案網路完成時間之估算並非易事且耗時，因此會使 EM 解 SRAP 問題的整個執行效率大打折扣。EM 之區域搜尋演算法在每一代的執行過程中，會持續不斷對全部或部份的樣本點執行其鄰域最佳點之搜尋，雖能加速 EM 搜尋的效率，但同時也會增加 EM 計算負荷影響其效率。

綜合上述，影響 EM 解 SRAP 問題執行效率的因素有三：初始樣本點的位置決定、目標函數值的計算及區域搜尋演算法的執行效率。本章節針對 SRAP 目標函數值計算所採用的估算方法為 LCTA，從第三章的內容得

知，LCTA 能夠快速的執行估算工作，且精準度亦非常的好。但是採用 LCTA 並非是 EM 解 SRAP 問題主要改良的對象，而是針對上述第 1 及 3 項影響 EM 效率的因素。首先針對 EM 樣本點決策變數搜尋範圍過大及如何決定初始樣本點位置之問題，本章節提出一關鍵路徑叢集演算法(Critical Path Cluster Algorithm, CPCA)，能夠有效的找出 EM 處理專案網路 SARP 問題適當之起始解(即接近最佳解的位置)，讓 EM 收斂的速度增快。另針對 SRAP 問題，本章節使用關鍵路徑叢集(Critical Path Cluster)的概念，提出叢集區域搜尋演算法(Cluster Local Search Algorithm, CLSA)及相關之執行策略，能夠有效且迅速的讓 EM 各樣本點自其鄰域找出改進值。相關的細節說明擬於下面章節敘述。

5.1.3 關鍵路徑叢集之概念

在隨機專案網路中，一般所常用的方法是 PERT 模式分析計算出路徑之關鍵路徑指標(Critical Path Index, PCI)，取其最大值找出網路中最長路徑，對專案管理者而言，將資源優先支配給該關鍵路徑上之作業，以增進專案網路之效率是順理成章的事，然本文發現事實並非一定如此，並以圖 5.7 之網路為例來做說明。

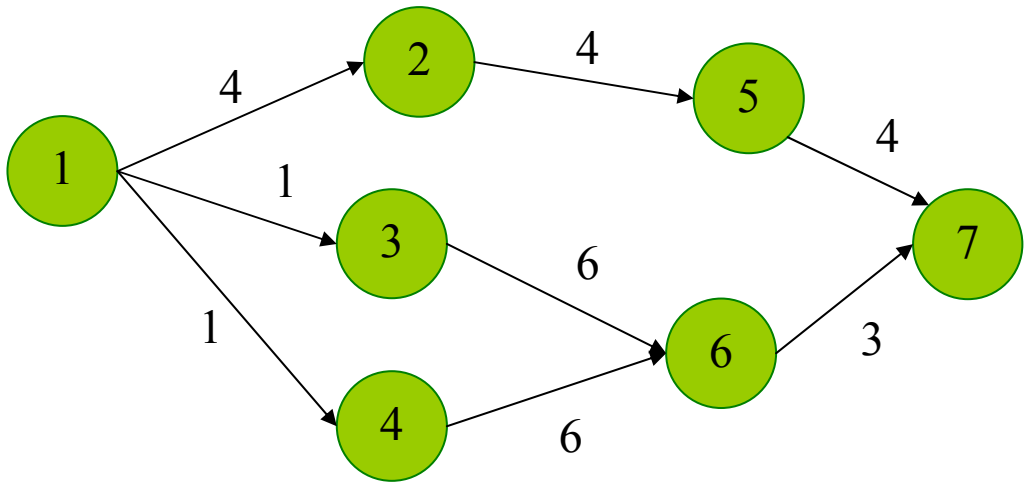


圖 5.7　7 節點之圖例

令圖 5.7 中 7 個節點間之作業時間均為指數分配，其平均值參數分別表示於圖中，其網路完成路徑共計三條：路徑(1,2,5,7), (1,3,6,7)及(1,4,6,7)。用 PERT 模式所計算得到的最長路徑為(1,2,5,7)，其完成路徑的平均值計算結

果為 12.0。但圖 5.7 之專案網路若以 MCS 模擬法取得真實的結果，則於節點 6 執行 max 運算所得到的輸出結果，其平均時間為 9.2，而於節點 7 的輸出結果其平均值為 16.62，此與 PERT 所得的完成路徑時間 12.0 有顯著的差異。另完成路徑的關鍵路徑指標值(Path Critical Index, PCI)值，亦可由蒙地卡羅模擬法所模擬出的各路徑 PCI 值如下：

$$\begin{aligned}&(1,2,5,7) \quad \text{PCI} = 0.426\\ &(1,3,6,7) \quad \text{PCI} = 0.286\\ &(1,4,6,7) \quad \text{PCI} = 0.288\end{aligned} \tag{5.10}$$

路徑 PCI 值愈大表示該路徑愈能成為該專案網路之關鍵路徑，就確定性專案網路的觀點而言，關鍵路徑是整專案網路的瓶頸，資源的投入應以關鍵路徑上之作業為優先對象，因為降低關鍵路徑時間才能讓整個專案網路時間有效的減少。依(5.10)式所計算出之 PCI 值可判定路徑(1,2,5,7)為關鍵路徑，從前文得知該路徑的完成時間為 12.0，而圖 5.7 之節點 6 的輸出時間再加上 X_{56} 作業時間共計 12.2，大於路徑(1,2,5,7)時間。經過作業 X_{56} 的兩個路徑(1,3,6,7)及(1,4,6,7)，其 PCI 值均低於路徑(1,2,5,7)。這說明了兩個現象：

1. 在隨機性專案網路中，擁有最大 PCI 值的路徑，其路徑完成時間不一定是最長的。
2. 在隨機性專案網路中，路徑作業的資源投入，其優先順序不能以 PCI 值來做決定。

對專案管理者而言，上述第二項的現象會嚴重的影響資源投入的決策；若以圖 5.7 為例，將資源優先投入路徑(1,2,5,7)的作業，對專案網路完成時間的降低並不是最好的決策，而應先將資源先投入路徑 (1,3,6,7)或(1,4,6,7)的作業中，才是最有效降低專案網路完成時間的作法，該現象可以下例來做進一步說明。

令圖 5.7 所有作業之單位資源使用成本 $U_C = \$1$，則依據(5.3)~(5.5)式可計算專案網路完成時間平均值 $E[z(t)] = 16.62$，專案總成本 $C_T = \$16.62$。若將資源 $r_{12} = 2.5$ 投入路徑(1,2,5,7)中之作業 X_{12}，該作業的時間平均值會因資源 r_{12} 的投入，而自原 4 降到 1.6，且依據(3.3)式，專案所額外耗費的資源使用成本為 $C_{R_{12}} = \$6$，這會使圖 5.7 專案網路的完成時間平均值自原 16.62 降至 15.2 且專案總成本增加至 $C_U = 16.62 + 6 = \$22.62$。若將相同的資源使用

成本\$6，平均分配給路徑(1,3,6,7)及(1,4,6,7)之作業 X_{36} 及 X_{46}，該兩作業的時間平均值會自原 6 降至 4.5，這情形會讓專案網路的完成時間自原 16.62 降至 14.85，專案總成本同樣為 $C_U = 16.62 + 6 = \$22.62$。這實例說明了將相同的資源成本投入 PCI 值較低的路徑作業中(X_{36} 及 X_{46})，專案網路完成時間的縮短量，反會比投入在關鍵路徑中之作業(X_{12})還來得多。

從上述的實驗結果得知，在隨機性專案網路環境中，PCI 值不能夠決定作業投入資源判斷的標準，而須先瞭解什麼樣的因素會直接影響專案網路完成時間?參考第三章 DA 或 LCTA 演算法之內容，在隨機性專案網路完成時間估算的過程中，max 運算是造成真實結果與 PERT 結果之間差異的最大因素，當路徑中的節點執行 max 運算後，會使路徑的完成時間，無法讓路徑作業時間以線性的關係來加總求之，因 max 運算會產生額外的 LPB 偏移量，隨著路徑時間之計算累積到專案網路輸出節點，因此專案網路輸出節點 N 的輸入路徑完成時間 $Y_i^N, i \in B_N$，才是決定專案網路完成時間之最重要因素。資源的投入應當優先給予最長輸入路徑完成時間 Y_i^N 內之作業。該最長輸入路徑是由一群路徑所構成(含有多條專案網路完成路徑)，本章節將其稱之為關鍵路徑叢集(Critical Path Cluster, CPC)。以圖 5.7 為例，整理成圖 5.8 來作進一步說明：

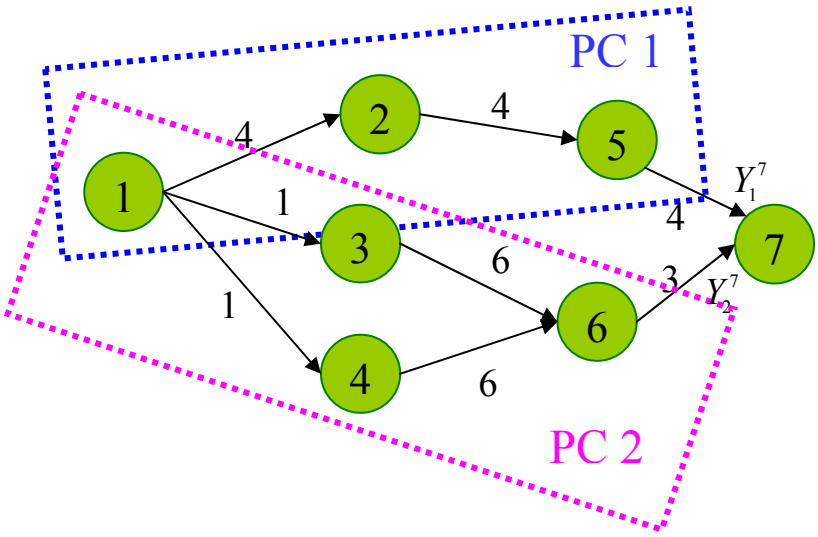


圖 5.8　關鍵路徑叢集之示意圖

圖 5.8 中路徑(1,3,6,7)及(1,4,6,7)所構成的路徑集合即為該網路之 CPC，因輸

出節點 7 之輸入路徑$Y_2^7 > Y_1^7$之故。也因此資源優先投入的判斷可以由本章節所定義之關鍵路徑叢集指標(Critical Path Cluster Index, CPCI)來取代，該指標之計算非常簡單，即該關鍵路徑叢集內所有路徑之 PCI 值之和，以圖 5.8 為例，專案網路共計兩個路徑叢集(PC_1 及 PC_2)，PC_1 內僅一條完成路徑(1,2,5,7)，PC_2 內有兩條完成路徑(1,3,6,7)及(1,4,6,7)，因此相關之 CPCI 值計算如下：

$$CPCI_1=PCI_{(1,2,5,7)}=0.426$$

$$CPCI_2=PCI_{(1,3,6,7)}+\ PCI_{(1,4,6,7)}=0.286+0.288=0.574 \qquad (5.11)$$

PC_2 的 CPCI 值大於 PC_1 之 CPCI 值(0.574 vs. 0.426)，因此 PC_2 為圖 5.7 之關鍵路徑叢集(CPC)，且專案網路資源投入應以路徑(1,3,6,7)及(1,4,6,7)內之作業為優先考慮。

5.1.4 關鍵路徑叢集演算法

第 5.1.2.2 章節指出當 EM 決策變數搜尋的空間過大時，會影響 EM 初始樣本點收斂至最佳解的速度，然而針對專案 SRAP 問題，決定適當之 EM 合理初始解(Feasible solutions)並非易事。本章節擬發展出關鍵路徑叢集演算法(Critical Path Cluster Algorithm, CPCA)，其目的就是要針對專案 SRAP 問題，建立一群 EM 合理初始值，CPCA 能令專案網路作業資源投入組合(Resource allocation pattern)，接近於 SRAP 問題的最佳解(最小的專案成本)，使得 EM 搜尋演算能夠很快的收斂至最佳解。

本章節擬先說明 CPCA 相關的符號及參數定義，接續參考 5.1.3 章節所提關鍵路徑叢集(CPC)之概念，針對專案 SRAP 問題作深入的瞭解，並對專案作業提出三種資源投入策略，CPCA 演算的程序即交互執行這三種策略，產生接近最佳解的合理作業資源投入組合，相關的說明詳如下文。

5.1.4.1 CPCA 相關的符號及參數定義

m: EM 樣本點的數目。

n: 專案網路作業數目，即決策變數之維度(dimension)數。

C_T^{ini}:專案作業在無資源投入的情況下之專案網路總成本。

C_P^i:在專案網路資源投入後，第 i^{th} 樣本點之懲罰成本。

C_U^i:在專案網路資源投入後，第 i^{th} 樣本點之資源使用成本。

C_T^i:在專案網路資源投入後，第 i^{th} 樣本點之專案網路總成本 $C_T^i=C_U^i+C_P^i$。

C_T^{best}: 在 EM 的演算過程中紀錄最低的專案網路總成本。

T^{ini}:專案作業在最低資源投入的情況下之專案網路的完成時間。

T_R^i:在專案網路資源投入後，第 i^{th} 樣本點之專案網路的完成時間。

T_Z:專案網路規定之完成時限。

a_z: 專案網路中之作業 z。

A^i:在專案網路資源投入後，關鍵路徑叢集(CPC)內之作業集合。

B^i:在專案網路資源投入後，非關鍵路徑叢集(Non-CPC)內之作業集合。

A_O^i:在專案網路資源投入後， CPC 與 No-CPC 之作業交集 $A_O^i = A^i \cap B^i$。

A_N^i:在專案網路資源投入後， CPC 與非 CPC 之作業差集合 $A_N^i = A^i - (A^i \cap B^i)$。

B_k^i:在專案網路資源投入後，第 k^{th} 個 Non-CPC 內的作業集合，其中 $k=1,...,Q$，Q 為專案網路 Non-CPC 之數目。

T_K^i:在專案網路資源投入後，第 k^{th} 個 Non-CPC 之完成時間，其中 $k=1,...,Q$，Q 為專案網路 Non-CPC 之數目。

U_j^{max}:專案網路作業 j 之資源投入量初始上限值。

U^j:EM 執行過程中，專案網路作業 j 之資源投入量上限值，其初始值設定為 U_j^{max}。

L^j: EM 執行過程中，專案網路作業 j 之資源投入量之下限值。

M^j: EM 執行過程中，調整作業 j 之資源投入量之下限值所使用的參考值。

re_al^i: EM 執行過程中，第 i^{th} 樣本點之資源投入組合。

Re_al^i: EM 初始樣本點，第 i^{th} 樣本點之資源投入組合。

UP_ratio: 專案網路資源使用成本與懲罰成本之參考比例值 $\frac{1}{2}\times(\frac{C_P}{C_U}+\frac{C_U}{C_P})$

$D\_ratio=0.9$: EM 執行過程中，調整資源投入量上限值之遞減率。

I_rate: EM 執行過程中，調整資源投入量下限值之遞減率，令$I_rate=\frac{1}{m}$。

$a_z \Leftarrow A$: 自 A 作業集合中隨機選取作業a_z之運算元。

X^i：EM 第 i^{th} 初始樣本點資源投入組合。

$f(X^i)$：EM 第 i^{th} 初始樣本點之中案總成本。

5.1.4.2 CPCA 對專案作業資源投入之策略

專案網路的總成本是由資源使用成本及懲罰成本所構成的，資源投入作業會產生資源使用成本，專案的完成時間若超過所規訂的時限，則會產生懲罰成本。該兩種成本對於專案網路中作業資源投入之反應是相反的；資源投入愈多，專案網路完成時間愈縮短，則懲罰成本愈低，反之資源使用成本增加；故總成本的增減，是要看這兩種成本隨作業資源投入量的改變，相對於總成本變化而定。SRAP 問題乃是針對隨機專案網路針對各作業執行資源投入以求得最小的成本，故針對專案網路作業資源投入，會產生若干考慮如下:

1. 對專案網路中任一作業投入的資源量愈多，會使專案成本會增加，但同時又可降低專案完成時間以減少增加懲罰成本的機率，此時專案管理者會產生兩種考慮：
   (1)當專案完成時間已超過時限，即懲罰成本已開始累計，此時資源的投入量必須斟酌考慮，以謀得最低的專案成本。
   (2)當專案完成時間未超過時限，則無須再投入資源減少專案完成時間，如此僅會增加額外的資源使用成本。假設一個專案網路所有作業均未投入額外資源的情況下，若其計算出來的完成時間未超過時限，則完全無須再投入任何資源，故專案總成本為 0。
2. 資源的投入應以專案網路中關鍵路徑叢集(CPC)內之作業為先考慮，其投入的原則亦有下列數項考慮，並以圖 5.9 來說明之。令圖 5.9 中叢集 3 為 CPC，即起始節點 1 至節點 C 之完成時間大於節點 B 及 A，從圖 5.9 中我們不難發現兩種情況:
   (1)當叢集 3 中的作業投入之資源量改變，會對專案網路完成時間所造成的影響比叢集 1 及 2 來的大。

(2)當叢集 3 中的作業資源量不變，僅投入叢集 1 及 2，其結果又有兩種狀況:

a. 當資源投入叢集 1 及 2 作業的結果，其起始節點至節點節點 B 或 A 仍未超過至節點 C 之完成時間，即叢集 3 仍為專案網路之 CPC，則該資源的投入對於專案網路的完成時間改變不大，因此該資源的投入僅會增加額外之資源使用成本，對於減少懲罰成本沒有太大的貢獻。

b. 當資源投入叢集 1 及作業的結果，其起始節點至節點節點 B 或 A 超過至節點 C 之完成時間，即專案網路之 CPC 改變了，則該資源的投入會對專案的總成本會有較大的變化。

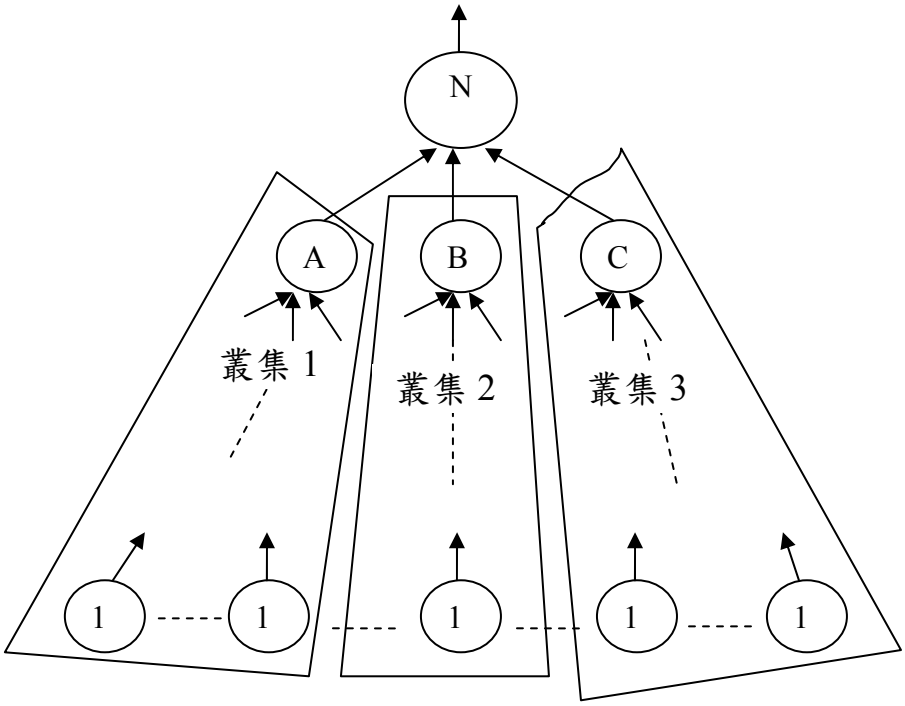


圖 5.9　專案網路路徑叢集示意圖

綜合上述專案網路資源投入作業所有考慮的因素，不難瞭解到，資源應針對專案網路中關鍵路徑叢集(CPC)內之作業作處理，因為 CPC 的完成時間對專案完成時間有最具關鍵性的影響，若該完成時間沒有太大的改變，而改變其他非關鍵路徑叢集的完成時間，對於專案網路整個完成時間影響不大，也不會增加專案懲罰成本，反而因投入額外資源於非關鍵叢集(Non-CPC)內之作業，增加專案資源使用成本，造成專案總成本的提高。

基於上述的原則，本文之 CPCA 處理 SRAP 問題主要採用了三種降低專案總成本的策略，分別敘述如下：

1. 第一種策略是傾向投入較小的作業資源使用成本，以達到降低專案總成本之目的。其作法是藉由增加 CPC 內作業資源投入量，降低專案網路完成時間，以減少專案網路懲罰成本。因資源投入要耗費專案成本，故投入資源的方式乃採漸進式遞減執行，其目的是嘗試尋找出最小的作業資源投入量，期以最小的專案網路資源使用成本達到本策略的目的。相關作法是調整作業資源投入之上界值U^j (Upper bound value)，來達到逐次降低資源投入量的效果。CPCA 執行本策略演算的過程中，會保留最小的專案總成本之作業資源投資組合紀錄，作為 CPCA 最後的演算結果。
2. 第二種策略的使用時機，是在 CPCA 使用第一種策略完成一次 EM 初始樣本點求解後實施。該策略會先檢查專案網路產生新的資源投入組合解後，專案網路之 CPC 是否有改變? 若沒改變，則會對原 CPC 之A_O^i作業做資源上界值的調整，其目的是希望改變 EM 初始樣本點資源投入組合的結構，因為後續之 EM 初始樣本點，會以前一次 EM 初始樣本點求解後之專案網路現況背景為依據，持續執行求解，故對 CPC 內之資源投入組合作微調，能避免 CPCA 落入區域最佳解的範圍。另第一種策略執行過程中，所調整的是 CPC 之A_N^i作業資源量，故本策略調整 CPC 之A_O^i作業資源量並不會對原 CPC 資源投入組合造成影響。若專案網路之 CPC 有改變，則本策略必須調整A^i，並設定A_N^i內作業資源相關之初始值，為 CPCA 進行下一次 EM 初始樣本點之求解作準備。
3. 第三種策略是當前兩種策略，對降低專案網路總成本均無效的情況下方才執行之；其執行的方式是針對 Non-CPC 內之作業，些微調降其資源投入量，以降低專案網路之資源使用成本，同時控制專案完成時間不要增加太多，而造成懲罰成本之增加。此策略要求所有參與 Non-CPC 內的作業，遵守一前提條件：就是該 Non-CPC 之路徑完成時間必須要大於專案完成時限$B_k^i > T_z, k = 1,...,Q$，若沒有超過專案完成時限就表示該叢集內的作業不會造成網路懲罰成本，因此無須再投入任何資源。

CPCA 剛開始產生 EM 的初始解時，乃先採用第一種策略，亦即對 CPC 內A_N^i之作業調整其資源投入量；因該策略會降低專案網路完成時間，減少專案的懲罰成本，因此專案網路中資源使用成本與懲罰成本間最後會達到

一個平衡狀態，而使專案網路產生較小的總成本。第二策略則是在執行 CPCA 的過程中，因應 CPC 的改變調整資源投入的對象；第三策略的功能則類似基因演算法(Gentic Algorithm, GA)中之突變運算元(Mutation)，將專案網路的資源投入組合，以隨機方式略為改變，以防 CPCA 在區域最佳解的範圍內打轉。根據本章節的數據實驗，另發現一有趣的現象，即專案網路資源使用成本與懲罰成本之比例，在 EM 收斂到最佳解的過程中會漸近收斂於 1，即當專案資源使用成本與懲罰成本之比例非常接近於 1 時，整個專案網路之作業資源分配即達一種平衡狀態，即改變任一作業資源量不容易得到更小的專案總成本，因此本章節採用上述經驗法則來作為資源投入策略重要的平衡參考點。本章節亦發現，即使專案網路中所有作業達到該平衡狀態，其資源投入作業之分配方式會有非常多種，且每一種資源分配方式所得到的總成本值亦不同；若能找到一種平衡狀態下之資源分配，使得最小的專案的總成本，則該資源分配狀態即為 SRAP 之最佳解。CPCA 所求得的作業資源投入組合並不保證為上述之最佳解，但會趨近於該最佳解的作業資源投入組合，因此 CPCA 所求得的結果非常適合做為 EM 之初始解。

5.1.4.3 關鍵路徑叢集演算法之演算程序

CPCA 會根據若干決策交替執行上述三種資源投入策略，該決策主要的考慮是當投入資源於專案網路 CPC 內之作業後，將可能造成專案網路原有專案網路 CPC 之改變，則原接受資源投入調整以降低懲罰成本之相關作業亦會變動，專案網路 CPC 的改變會同時影響專案網路的完成時間及總成本。因此在 CPCA 的演算過程中，最重要的決策內容就是要持續的判斷出需要增加資源投入之作業，或需要減少資源投入之作業，配合上述三種資源投入策略有效的調整降低專案網路的總成本。CPCA 主要的演算流程以虛擬程式表示如演算法 1(詳細的流程圖參考附錄 J):

演算法 1:

Initialization( );

If $(T^{ini} > T_Z)$

For $i = 1$ **To** m

Strategy_1( );

$X^i = \mathrm{Re}\_Al^{best}$ ；

$f(X^i) = C_T^{best}$ ；

Strategy_2( );

If $(C_T^i < C_T^{i-1})$

Strategy_3( );

End

End

Else

$X^i = L^j, \forall j$ ；

$f(X^i) = C_T^{ini}, \ \forall i$ ；

Break;

End

CPCA 的演算程序會重複不斷的執行前述三種資源投入策略，直到產生 m 個 EM 初始樣本點為止。其開始第一步驟是執行起始程序 Initialization( )，該程序會檢查$T^{ini} > T_Z$， T^{ini}表示專案網路中所有作業在最低資源投入的情況下之專案網路的完成時間，如果該時間小於專案網路規定之完成時限T_Z，則專案網路無須再投入任何資源，作業最低資源投入之組合即為專案網路 SRAP 之最佳解。相反的，如果該時間大於專案網路規定之完成時限，CPCA 就開始進行 m 個 EM 初始樣本點的計算。CPCA 在計算之初，專案網路存在懲罰成本，因此 CPCA 要先執行第一種資源投入策略(Strategy_1( ))，該策略的主要目的是讓專案網路之資源使用成本及懲罰成本間取得平衡關係，因此該策略會針對專案網路 CPC 內的作業，循序漸進的投入資源，在資源投入的過程中專案網路完成時間會逐漸降低，專案懲罰成本亦會跟著降低，計算的過程中同時也會紀錄最低的專案總成本之資源投入組合。當 Strategy_1( )在執行的過程中發現，因資源不斷的投入而造成專案網路 CPC 的改變，此時資源投入的對象作業就須改變，否則專案網路之資源使用成本及懲罰成本間的平衡關係會更惡化，因此 Strategy_1( )利用一UP_ratio作為專案網路之資源使用成本及懲罰成本間的平衡參考值，表示如下：

$$UP\_ratio=\frac{1}{2}\times(\frac{C_P}{C_U}+\frac{C_U}{C_P}) \tag{5.12}$$

當該值趨近於 1 則可結束 Strategy_1( )，此時紀錄最低的專案總成本之資源投入組合即為一個 EM 起始樣本點。當 CPCA 完成一次 EM 起始樣本點的解後，便開始進行第二資源投入策略(Strategy_2( ))。如前文所述，Strategy_2( )的目的是為 CPCA 進行下一次 EM 初始樣本點之求解作準備，因應專案網路 CPC 之改變，調整相關資源投入之作業對象。第三資源投入策略(Strategy_3( ))執行的時機，是在前兩個策略已無法再產生出更佳的 EM 初始樣本點後進行，CPCA 完成一次 EM 初始樣本點求解後，可利用判斷式 $C_T^i < C_T^{i-1}$ 查知。演算法 1 中，針對 Initialization( )及三種資源投入策略詳細流程，擬於下文說明。

CPCA 的第一個程序是 Initialization( )，參考演算法 2。它的主要工作就是將專案網路的所有作業均設定最低的資源投入量 L^j，故其專案網路第 1 個 EM 初始樣本的資源投入組合為 re_al^1，執行 CPCA 開始前，並將其設定為最佳的資源投入組合 Re_Al^{best}。將 re_al^1 投入專案網路並算專案網路時間 T^{ini} 及專案網路總成本 C_T^i，並令 $C_T^{best}=C_T^i$。接續計算求出專案網路之關鍵路徑叢集(CPC)，並將其分成兩部份：A_N^i 及 A_O^i。因為專案網路 CPC 內的作業可能部份會與 Non-CPC 之作業有重疊的情況，因此要將重疊的作業過濾出來。Strategy_1( )僅針對 CPC 內未重疊之作業 A_N^i 投入資源，其目的就是要觀察出專案網路 CPC 對專案完成時間所造成的影響。

演算法 2:

Initialization( )

$Re\_Al^{best} = re\_al^1 = L^j,\ \forall j;$

To Calculate the $C_T^{best}=C_T^i$ and T^{ini};

To calculate the A^i & B^i, $k=1,...,Q$;

$A_N^i = \{a_z \mid a_z \in A^i \ \ \& \ \ a_z \notin B_k^i,\ k=1,...,Q\}$;

$A_O^i = A^i - A_N^i,\ A^i = A_N^i \cup A_O^i$;

EM 初始樣本點主要是由 Strategy_1( )的演算流程求出，參考演算法 3。

Strategy_1( )主要是針對專案網路 CPC A_N^i 內之作業投入資源量，投入之資源量會以逐次遞減的方式執行，期以最少的資源投入量得到最小的專案網路總成本 C_T^{best}。因此 Strategy_1( )以一無窮迴圈(While operation)反複執行，每一迴圈開始便針對 A_N^i 內作業投入資源，A_N^i 作業之資源投入量最初是以該作業之上、下界的資源範圍隨機選取而得的(參考演算法 3 之(A)標籤列)，投入相關資源後，再計算專案網路完成時間 T_R^i、資源使用成本 C_U^i、懲罰成本 C_P^i 及專案網路總成本 C_T^i。由 C_U^i 及 C_P^i 可以計算出 UP_ratio 的比例值，該值是 Strategy_1( )決定是否再投入資源的重要參考，若 C_U^i 及 C_P^i 達到平衡，UP_ratio 會趨近於 1(CPCA 認定 $UP\_ratio \le 1.05$ 即為接近之意)，即表示專案網路 A_N^i 作業之資源調整已達平衡，便可離開 Strategy_1( )迴圈，求得一個 EM 初始樣本解。若 C_U^i 及 C_P^i 未符合 $UP\_ratio = 1.05$ 之要求，則降低 A_N^i 作業資源投入量之上、下界值之範圍(參考演算法 3 之(B)標籤列)，縮短該作業資源投入量隨機選取的範圍，作業所投入的資源量平均值便會降低，最後求得專案網路資源投入組合 re_al^i。將 re_al^i 投重新投入專案網路中，再計算專案網路完成時間、資源使用成本、懲罰成本及專案網路總成本，在持續執行 UP_ratio 檢查。重複降低資源範圍的目的，是要不斷嘗試找出較小的作業資源投入量，只要 A_N^i 作業所投入的資源量未符合 UP_ratio 的比例值要求，就表示作業資源投入量還容許再降低。若當 A_N^i 作業之資源投入量之上界值持續降到與下界值相接近，演算法 3 中以 $Test^i$ 測試接近的程度，若小於 ε%之比例，即表示資源範圍無法再降低，此時亦可離開 Strategy_1( )迴圈之運算。當離開 Strategy_1( )迴圈，會將最好的結果記錄起來，包括專案網路資源投入組合 Re_Al^{best} 及專案網路總成本 C_T^{best}，該結果即為 EM 初始樣本點的解 $X^i = C_T^{best}$ 及 $f(X^i) = Re\_Al^{best}$。

演算法 3:

Strategy_1( )

$C_T^{best} = C_T^{ini}$;

While $Test^i > \varepsilon\%$

$\lambda = U(0,1)$;

$re\_al^i = L^j + \lambda \cdot (U^j - L^j),\ i = 1,...,m,\ j \in A_N^i$; (A)

重新計算專案網路之T_R^i, C_U^i, C_P^i及C_T^i

$UP\_ratio = \frac{1}{2} \cdot (\frac{C_P^i}{C_U^i} + \frac{C_U^i}{C_P^i})$;

If UP_ratio>1.05

If $T_R^i > T_Z$

$U^j = L^j + D\_rate \cdot (U^j - L^j)$, $j \in A_N^i$ ； (B)

End

Else

Break;

End

If $C_T^i < C_T^{best}$

$C_T^{best} = C_T^i$;

$Re\_Al^{best} = re\_al^i$;

End

$Test^i = \frac{(U^j - L^j)}{L^j}$, $j \in A_N^i$**;**

End

EM 的 *m* 個初始樣本點是反覆執行 Strategy_1( )得到的，即 EM 初始樣本點的產生必須要參考前一個 EM 初始樣本點的結果，因此 CPCA 所產生的初始樣本點所得到的解，會一次比一次來得好。Strategy_2( )的主要目的是要為下一個 EM 初始樣本點作準備，當執行完 Strategy_1( )產生一 EM 初始樣本點後，便要接著執行 Strategy_2( )，該演算流程表示於演算法 4。Strategy_2( )首先檢查前一個 EM 樣本點資源投入組合X^i，將其代入專案網路，並檢查專案網路 CPC 是否改變$A^{i+1} = A^i$? 若原專案網路 CPC 沒變，則僅需對該 CPC 之A_O^i作業進行資源提增，該原因已於前述第 5.1.4.2 章節中第二種資源投入策略中說明。資源提增的方式是針對A_O^i內之作業 j，逐漸調升其作業資源下限值L^j與中間值M^j之範圍(參考演算法 4 之(B)標籤列)，M^j值會在 Strategy_2( )演算的過程中逐漸調升(參考演算法 4 之(A)標籤列)。若原專案網路 CPC 有改變，則對新產生之專案網路 CPC A_N^{i+1}作業進行資源提增，A^{i+1}是在前 Strategy_1()執行過程中，自 Non-CPC 叢集B^i中挑選出最大的完成時間叢集(參考演算法 4 之(C)標籤列)，A_N^{i+1}作業資源投入

的方式亦如上述方法，針對A_N^{i+1}內之作業j，逐漸調升其作業資源下限值L^j與中間值M^j之範圍，M^j值亦同樣會在 Strategy_2( )演算的過程中逐漸作調升。

演算法 4:

Strategy_2( )

投入資源組合X^i，重新計算專案網路之 CPC(A^i);

If $A^{i+1} = A^i$

$$M^j = M^j + I\_rate \cdot (U^j - L^j); \qquad \text{(A)}$$

$$\beta = U(0,1);$$

$$re\_al^{i+1} = L^j + \beta \cdot (M^j - L^j),\ i = 1,...,m-1,\ \ j \in \{A_O^i\}; \qquad \text{(B)}$$

Else

$$M^j = M^j + I\_rate \cdot (U^j - L^j);$$

$$\beta = U(0,1);$$

$$re\_al^{i+1} = L^j + \beta \cdot (M^j - L^j),\ i = 1,...,m$$

$$j \in A_N^{i+1} \in \{B_h^i |\ T_h^i = \max(T_k^i),\ h, k = 1,...,Q\}; \qquad \text{(C)}$$

End

根據演算法 1 之內容，若 Strategy_1( )及 Strategy_2( )執行後，CPCA 所求得之 EM 初始樣本點的目標函數值無法得到改善，即無法得到更小的專案網路的總成本$C_T^i < C_T^{i-1}$，此時專案網路之資源投入組合可能已進入特定之區域最佳解之範圍。為解決此問題，可依循基因演算法之突變運算元之概念，將現有的專案網路之資源投入組合隨機做一些變動，使 CPCA 有機會擺脫區域最佳解之情況。Strategy_3( )即為此功能而設計，但考量專案網路資源的變動範圍不要過廣，以至於影響 CPCA 求解的效率，故不擬改變專案網路 CPC 內之作業A^i資源投入量，因其對專案網路完成時間比較敏感；僅針對專案網路 Non-CPC 之作業B^i，以減少其資源量之方式達到擺脫區域最佳解的目的。資源減少的方式仍針對欲投入資源之作業j，逐漸調降其作業資源下限值L^j與中間值M^j之範圍(參考演算法 4 之(A)標籤列)，M^j值會在 Strategy_3( )演算的過程中逐漸調降，且執行調降資源的作業是隨機自B^i中選取(參考演算法 4 之(B)標籤列)。

演算法 5:

Strategy_3( )

$$M^j = M^j - I\_rate \cdot (U^j - L^j); \qquad \text{(A)}$$

$$\gamma = U(0,1);$$

$$re\_al^{i+1} = L^j + \gamma \cdot (M^j - L^j),\ i = 1,...,m-1 \quad ; \qquad \text{(B)}$$
$$j \in \{a_z \mid a_z \Leftarrow B^i,\ z=1,...,w\}$$

5.1.5 叢集區域搜尋演算法

EM 在搜尋最佳解的過程中，亦可以利用區域搜尋(Local search)來尋找樣本點鄰域最佳值，增進 EM 求解的效率，當樣本點愈接近最佳解的時候，可以利用區域搜尋演算得到解析度更好的結果。然區域搜尋演算會額外增加時間造成 EM 效率上的負擔，且根據 Birbel & Fang(2003)文獻所述，未採用區域搜尋演算的 EM，其求解的結果亦不遜色。因此本文權衡 EM 求解精度及其效率後，提出叢集區域搜尋演算法(Cluster Local Search Algorithm, CLSA)來執行區域搜尋演算，且考量 EM 之效率，當 EM 連續 10 代(Generations)搜尋演算，求解均未獲得改善的情況下，針對每一個樣本點執行 CLSA 一次。即當 EM 的樣本點趨近於最佳解的時候，才執行區域搜尋演算，如此讓區域搜尋演算在 EM 的執行過程發揮實質的效用。

5.1.5.1 叢集區域搜尋演算法之概念

CLSA 是專為 EM 處理專案網路 SRAP 問題所設計的，其作法是針對 EM 每一樣本點之鄰域，找尋「再好一點」的樣本點來取代現行的樣本點，並非強迫找出其周圍最好的解。從第 4.1.3 章節中瞭解專案網路對於資源投入的一些特性，專案網路的 CPC 是我們考慮的重點， CLSA 亦是基於這些特性來設計的。當專案網路投入資源後，其 CPC 內的作業資源量之變化，對於網路的完成時間有較敏感的反應，且易對網路成本造成較大的影響，故 CLSA 鑒於此，不針對 CPC 內的作業作資源量之改變，而針對專案網路 Non-CPC 內的作業來微調其資源量，主要的理由就是當改變了 Non-CPC 內的作業資源量，若未造成專案網路內之 CPC 改變的話，即表示網路的完成時間不會有太大的變化，因此專案網路在資源使用成本下降且懲罰成本又不會過度提升的情況下，便能有效降低專案網路的總成本。但若造成專案

網路內之 CPC 改變的話，本文的作法便改為逐次減少 Non-CPC 參與資源縮減之作業數目，直到專案網路內之 CPC 不產生變化為止。如果降低所有 Non-CPC 內作業數目都無法得到較少的總成本的話，就表示區域搜尋演算無法在樣本點鄰域找到更好的解。

5.1.5.2 叢集區域搜尋演算法之演算程序

CLSA 演算流程序如演算法 6 所列(詳細的流程圖參考附錄 K)，其執行步驟依序說明如下:

1. 先針對樣本點 i 其資源投入量之現值 re_al^i，計算出專案完成時間 T_R^i 及總成本 C_T^i。同時亦可以求得該資源投入的情況下之 Non-CPC 內的作業 B^i 及其 Non-CPC 各叢集完成時間 $T_k^i,\ k=1,...,Q$。
2. 針對 Non-CPC 內的作業，酌量調降其作業資源上、下界值 U^i, L^j 間之範圍(參考演算法 6 之(A)標籤列)，調降方式是以 $Ld\_rate=0.1$ 值乘上該範圍，再以亂數方式依該範圍決定所微調的資源量。最後得到專案網路資源投入組合 $re\_al^{i'}$。再將 $re\_al^{i'}$ 投入專案網路中，計算專案總成本 $C_T^{i'}$ 並與原 C_T^i 比較大小。
3. 如果微調降之資源投入量 $re\_al^{i'}$ 造成專案網路專案總成本 $C_T^{i'}$ 大於 C_T^i，或造成專案網路 CPC 的改變，表示 EM 的區域搜尋演算並沒有得到更好的解，因此 LCTA 必須減少專案網路 Non-CPC 內參與調降資源量作業之數目，以控制專案網路 Non-CPC 的完成時間不要增加太快。CLSA 每次降低的作業數目設定為 $d=\left|\frac{number(B^i)}{lsiter}\right|$，並令 $lsiter$=10 為區域搜尋演算之最大次數，$number(B^i)$ 表示專案網路 Non-CPC 之作業總數。LCTA 所減去之 Non-CPC 作業 D^i，是以亂數方式決定之(參考演算法 6 之(B)標籤列)。縮減後的 Non-CPC 作業 B^i 重複執行步驟 2、3，直到專案網路 CPC 不改變為止。
4. 如果微調降之資源投入量 $re\_al^{i'}$ 令專案網路專案總成本 $C_T^{i'}$ 小於 C_T^i，且未造成專案網路 CPC 的改變，如此就達到區域搜尋演算尋求樣本點鄰域「較佳解」的目的。

LCTA 找到鄰域較佳解後，仍可持續進行上述區域搜尋演算的步驟以

尋找鄰域「最佳解」，但同時也會消耗更多的時間，使得 EM 整體執行效率下降， 針對此問題，EM 演算法可在求解精度及效率上做一權衡考量。

演算法 6 :

CLSA 演算程序

$lsiter = 10;$

$Ld\_rate = 0.1;$

$d = \frac{number(B^i)}{lsiter};$

For *i=1* **To** *m*

以 re_al^i 投入專案網路，重新計算專案網路之 T_R^i, C_T^i 及

$B^i, B_k^i \ \& \ T_k^i, k = 1,...,Q;$

$B^{i'} = B^i;$

While $B^{i'} \neq empty$

$\lambda = U(0,1);$

$re\_al^{i'} = re\_al^i - \lambda \cdot Ld\_rate \cdot (U^j - L^j),\ i = 1,...,m,\ j \in B^i;$ (A)

以 re_al^i 投入專案網路，重新計算專案網路之 $C_T^i;$

If $C_T^{i'} \geq C_T^i$

$D^i = \{a_z \mid a_z \Leftarrow B^{i'},\ z = 1,...,d\};$ (B)

$B^{i'} = B^{i'} - D^i;$

$re\_al^{i'} = re\_al^i;$

Else

$re\_al^i = re\_al^{i'};$

$i = i + 1;$

Break;

End

End

End

5.1.6 EM 解 SRAP 之數據實驗

本章節擬採用 EM 演算法，配合本論文所發展出之 CPCA 及 CLSA，來處理專案網路 SRA P 之問題。本章節擬採用 Terreso, et al. (2004b)文獻中所提供 14 個大小不同的專案網路實體(作業數從 3 到 76 個)為實驗對象，作

業時間均假設為指數分佈函數，專案網路相關之作業時間參數、規定時限T_z、懲罰單位成本P_C及資源使用單位成本U_C，參閱附錄 L 所列。實驗所使用的個人電腦為 Intel Pentum-4 2.0 GHZ CPU、512MB 記憶體並使用 Matlab Ver.6.5 語言程式來執行。

本章節實驗內容分兩部份：第一部分實驗目的是要驗證 EM 演算法結合 CPCA 及 CLSA 處理專案網路 SRAP 問題之效果。第二部份實驗目的，乃與 Terreso, et al. (2004b)處理專案網路 SRAP 問題的結果做比較，該文獻亦使用 EM 演算法解專案網路 SRAP 問題，然其估算隨機性專案網路完成時間的方式是採 100 次取樣之蒙地卡羅模擬法(MCS)，本章節將該方法表示為"MCS_EM"。參閱本論文第三章內容，MCS 處理較中、大型之專案網路有計算負擔過重之問題，且若為解該問題而將取樣次數降低，更會影響其估算之精度及穩定度。因此第二部份實驗擬先實驗說明上述 MCS 之問題，另証明本章節所提之方法較 Terreso, et al. (2004b)有效果。

5.1.6.1 第一組實驗

本組實驗選用 Terreso, et al. (2004b)第 14th 專案網路實體為實驗對象，因該專案的作業量最大(76 個)，網路的複雜度亦較高，實驗的結果也較有說服性。EM 演算法擬執行 400 代(Generations)運算，每一代的樣本點取m=15。因 EM 演算法所處理的對象為專案網路 SRAP 問題，目標函數值為專案網路的總成本，每一樣本點均需使用第三章之 LCTA，估算專案網路完成時間，再依(5.5)式計算出專案總成本。俟所有的樣本點完成目標函數值計算後，即可依(5.7), (5.8)及(5.9)式將 EM 演算法中每一樣本點所帶之電荷量、合成力及移動步距決定出來。為彰顯 CPCA 及 CLSA 在 EM 演算法中之效果，本組實驗在相同的條件下，以下列三種不同的方法分別執行:

1. Randomly_EM:表示 EM 演算法的 15 個初始樣本點，在 EM 合理解答範圍內隨機產生的。
2. CPCA_EM: 表示 EM 演算法的 15 個初始樣本點，由 CPCA 計算提供。
3. (CPCA & CLSA)_EM : 表示 EM 演算法的 15 個初始樣本點，由 CPCA 計算提供，且當 EM 連續 10 代之搜尋演算，求解均未獲得改善的情況下，針對每一個樣本點執行 CLSA 一次。

上述 EM 三項實驗的結果與原 Terreso, et al. (2004b)之 MCS_EM 方法所得

結果作比較，並整理如表 5.1 及圖 5.10 所列。

表 5.1 EM 採用 CPCA 及 CLSA 與隨機產生初始樣本點之結果

	MCS_EM	Randomly_EM	CPCA_EM	(CPCA & CLSA) _EM
專案總成本	559.19	527.59	510.16	504.54
執行時間(秒)	22320	1672	1703	2836

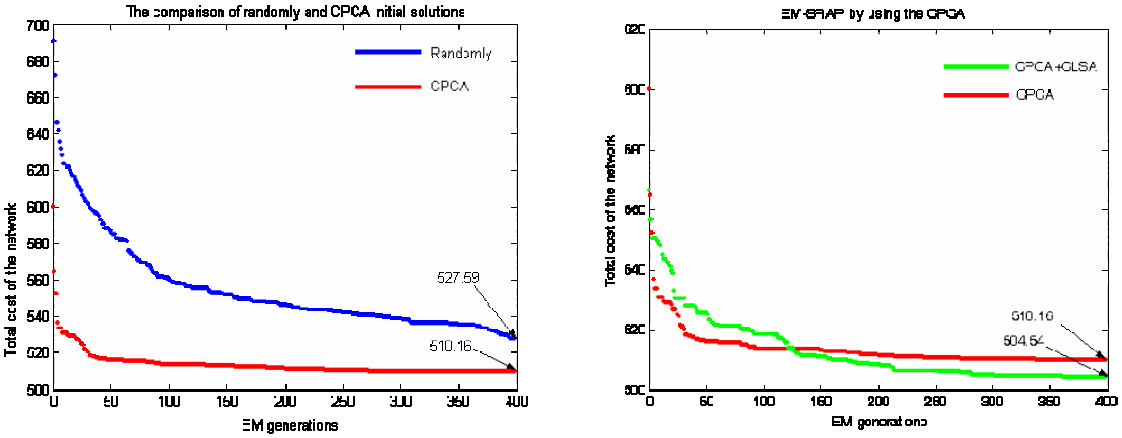


圖 5.10　EM 採用 CPCA 及 CLSA 與隨機產生初始樣本點之結果比較

從上述三項實驗結果中很清楚的顯示，Randomly_EM 的結果遜於 CPCA_EM 及(CPCA & CLSA) _EM 之結果(527.59 vs. 510.16 及 504.54)。另比較 Randomly_EM 及 CPCA_EM 的執行時間後，也顯示 CPCA 所產生的 15 個 EM 初始樣本點，並未耗費太多的時間(1672 vs. 1703 秒)，然而 EM 演算法執行 CLSA 則有較明顯的增加。Terreso, et al. (2004b)採用 100 取樣次數之 MCS，其結果明顯的優於所有的方法，執行時間亦最長(22320 秒)。另須特別強調之處，即單獨使用 CPCA 所產生出的 EM 初始解，所得到之結果(564.8)即已非常接近 MCS_EM 最後所求的結果(559.19)，且 CPCA_EM 約在 EM 演算法前 5 代即可超越該結果。如此可以證明 EM 演算法採用 CPCA 及 CLSA 解專案網路 SRAP 問題發揮效果，且能快速的將 EM 的初始樣本點朝最佳解的位置收斂。

5.1.6.2 第二組實驗

本組實驗選用 Terreso, et al. (2004b)共 14 個專案網路實體為實驗對象(相關專案網路資料可參閱附錄 L)，處理專案網路 SRAP 問題，並將該文獻

針對上述專案網路所得的結果，與本論文所提之 CPCA_EM 及(CPCA & CLSA) _EM 做比較。從第一組實驗中已知該文獻採 MCS_EM 方法，且 MCS 取樣次數僅 100。本論文質疑此作法，並以實驗顯示 MCS 取樣數過少對估算隨機性專案網路完成時間所造成的問題，同時也使用 LCTA 執行相同的實驗，作相對應的比較。實驗的方式是針對上述 14 個專案網路實體，並分別執行 20000 次取樣之 MCS 估算專案網路完成時間及總成本，所得的結果可視為真實值。再將該結果與 100 次取樣之 MCS 及 LCTA 估算結果做比較，實驗結果如表 5.2 及 5.3 所列。

表 5.2　LCTA、100 及 20000 次取樣之 MCS 結果比較
(估算專案網路完成時間)

網路編號	作業數	20000 取樣	100 取樣	誤差率	LCTA	誤差率
1	3	26.31	26.13	0.68%	26.39	0.30%
2	5	152.85	142.57	6.73%	161.07	5.38%
3	7	103.89	80.63	22.39%	105.67	1.71%
4	9	180.72	141.60	21.65%	186.18	3.02%
5	11	44.88	36.84	17.91%	45.72	1.87%
6	11	116.5	87.90	24.55%	121.27	4.09%
7	12	75.74	58.91	22.22%	78.48	3.62%
8	14	59.13	50.74	14.19%	60.27	1.93%
9	14	288.46	236.42	18.04%	299.01	3.66%
10	17	68.28	54.61	20.02%	68.22	0.09%
11	18	154.71	124.74	19.37%	159.33	2.99%
12	24	362.61	255.68	29.49%	372.91	2.84%
13	38	260.36	209.76	19.43%	250.6	3.75%
14	76	191.19	174.50	8.73%	185.28	3.09%
平均誤差率			**17.53%**		**2.74%**	

從表 5.2 及 5.3 中，可很清楚的証明，過少的取樣數會對 MCS 的估算精度造成嚴重的影響，100 次數取樣的估算結果，專案網路完成時間誤差率為 17.53%，專案網路總成本誤差率高達 32.88%，相對於 LCTA 之估算結果(2.74% 及 4.86%)有相當大的差別。另估算估算專案總成本的誤差率會較

大，其理由是因專案完成時間一旦超過規定時限T_Z，便產生懲罰成本C_P；若遇單位時間懲罰成本P_C較大時，些許的完成時間變動，便會造成較大的總成本變動，也因此造成較大的誤差。

表 5.3　LCTA、100 及 20000 次取樣之 MCS 結果比較
(估算專案總成本)

網路編號	作業數	20000 取樣	100 取樣	誤差率	LCTA	誤差率
1	3	42.01	39.58	5.78%	42.10	0.21%
2	5	414.72	330.97	20.19%	478.97	15.49%
3	7	283.43	168.02	40.72%	293.21	3.45%
4	9	475.91	320.16	32.73%	498.48	4.74%
5	11	186.81	118.78	36.42%	189.77	1.58%
6	11	389.43	238.84	38.67%	405.68	4.17%
7	12	200.36	132.12	34.06%	210.41	5.02%
8	14	118.61	91.80	22.60%	120.38	1.49%
9	14	933.46	609.55	34.70%	985.09	5.53%
10	17	204.36	105.88	48.19%	201.12	1.59%
11	18	623.60	333.63	46.50%	679.50	8.96%
12	24	2145.88	857.70	60.03%	2262.76	5.45%
13	38	952.38	693.07	27.23%	897.28	5.79%
14	76	543.45	475.14	12.57%	518.24	4.64%
平均誤差率			**32.88%**		**4.86%**	

接續的實驗乃是比較 100 取樣之 CPCA_EM 與本論文所發展之 CPCA_EM 及(CPCA & CLSA) _EM，針對上述 14 個專案網路實體，處理專案網路 SRAP 問題之結果做比較。相關的比較結果表示於表 5.4 中。

參考表 5.4，100 次取樣之 MCS_EM 所得的專案總成本結果及執行時間，與 CPCA_EM 及(CPCA & CLSA) _EM 的比較結果，相關的說明整理如下：

1. MCS_EM 於 14 個專案網路項目所得的專案總成本結果，均劣於 CPCA_EM 及(CPCA & CLSA) _EM，甚至第 12 項的結果與 (CPCA & CLSA) _EM 差近 1 倍之譜。

表 5.4　CPCA_EM、 (CPCA & CLSA) _EM 及 MCS_EM 處理 SRAP 問題結果之比較

網路編號	**MCS**_EM		**CPCA**_EM		**(CPCA & CLSA)**_EM	
	專案成本	執行時間	專案成本	執行時間	專案成本	執行時間
1	42.09	123	41.12	38	41.12	62
2	478.96	233	262.36	77	262.33	129
3	293.20	371	225.16	109	225.20	190
4	498.48	532	443.36	133	443.37	223
5	189.76	700	130.70	152	130.49	264
6	405.68	716	343.00	201	342.69	334
7	210.40	839	190.53	177	190.60	298
8	120.37	1027	117.03	183	116.89	310
9	985.08	1074	879.87	291	880.07	488
10	201.11	1473	167.98	256	123.12	429
11	679.5	1713	537.54	339	371.39	566
12	2264	2831	1570.5	701	1263.9	1175
13	897.28	7128	857.81	981	852.57	1631
14	518.23	31968	510.16	1704	504.54	2836

2. CPCA_EM 執行速度遠比 MCS_EM 快 3.2~18.7 倍，CPCA & CLSA 則為 1.9~11.2 倍左右，專案作業數愈大則差距愈大。
3. 針對 EM 演算法加入 CLSA 之效果做觀察，(CPCA & CLSA) _EM 所得的專案總成本結果，僅第 5,7,9 專案網路項目未比 CPCA_EM 來得好(差距甚微)外，其餘均小於 CPCA_EM，尤其第 10,11 及 12 項差距甚大。(CPCA & CLSA) _EM 的執行時間平均為 CPCA_EM 1.68 倍。
4. 據實驗過程中顯示，CPCA_EM 及(CPCA & CLSA) _EM 在重複執行至 50 代及 150 後，答案即已趨穩定，400 代所得的解與其相差甚小。因此 CPCA_EM 及(CPCA & CLSA) _EM 實際執行時間是可以再縮短 4 至 2.6 倍左右。
5. 單獨使用 CPCA 所產生之 EM 初始樣本點，所得到的專案網路完成時間值已趨近於 MCS_EM 最終的解答，其中 9 項專案網路甚至比 MCS_EM

最終的解答還來的好，參考表 5.5 內容所列。這證明了 CPCA 求解專案網路 SRAP 問題是非常有效果的。

表 5.5　CPCA 所得專案網路總成本值小於 MCS_EM 最終結果之比較

網路編號	1	2	3	4	5	6	7
EM+MCS	42.09	478.96	293.2	498.48	189.76	405.68	210.4
CPCA	**41.4**	**393.1**	393.1	**471.0**	**167.4**	**384.6**	214.9
網路編號	8	9	10	11	12	13	14
EM+MCS	120.37	985.08	201.11	679.5	2264	897.28	518.23
CPCA	129.7	1068.9	**174.4**	**599.4**	**1972.1**	940.4	557.9

5.1.7 結論

隨機性專案網路最佳化資源分配的問題(SRAP)，因作業為隨機變數之故，若以確定性專案網路分析工具或數理分析方式求解並不適當，根據第二章文獻探討內容，採用啟發式的解法是較為可行的方式。本論文採用近年來新發展出之磁場機制演算法(EM)，其理由是因其理論架構完整且程序簡單較易於程式化之故。然 EM 演算法有兩個因素會嚴重影響其求解收斂之效率：決定合理的起始樣本點及適當的區域搜尋演算法。關於前者，EM 演算法會因樣本點的維度(Dimension)增加，搜尋的範圍會以指數型態的擴大，若無安排適當的起始樣本點，EM 演算法會相當耗費時間收斂至最佳解處。關於後者，若無適當的區域搜尋演算法，增加樣本點搜尋周圍較佳解的機率，EM 演算法容易讓群體內表現較突出的某一樣本點，過度吸引其它的樣本點，容易導致陷入區域最佳解的泥沼。另區域搜尋演算法執行的策略也非常重要，若執行過度頻繁，會增加 EM 演算法計算負荷，反而影響其執行效率。因此要額外考慮區域搜尋演算法執行的時機，選取適當數量之樣本點為執行之對象。本論文針對上述 EM 演算法第一個個問題，提出關鍵叢集演算法(Critical Path Cluster Algorithm, CPCA)來負責找出適當的 EM 初始樣本點；CPCA 主要源於隨機性專案網路中關鍵路徑叢集(Critical Path Cluster, CPC)之概念，並以關鍵路徑叢集指標(CPCI)來取代一般專案網路所使用的關鍵路徑指標(Path Critical Index, PCI)，即在隨機性專案網路中，資源須優先投入 CPCI 值較高的專案網路叢集之作業中，而非投入較高 PCI 值的路徑作業中。這概念是由本論文首度提出，也是本論文主

要貢獻之一。基於 CPC 的概念，本論文另發展出叢集區域搜尋演算法 (Cluster Local Search Algorithm, CLSA)，供 EM 演算法處理專案網路 SRAP 問題，使能夠有效解決上述 EM 演算法第二項問題。

EM 演算法採用本論文所發展之 CPCA 及 CLSA，應用於專案網路 SRAP 問題中，無論在求解或效率方面均有非常好的表現。從第 5.1.6 章節的實驗數據中顯示，CPCA_EM 及 (CPCA & CLSA) _EM 針對 14 個專案網路實體，所得結果均優於 Terreso, et al. (2004b)所採用之 MCS_EM。若單獨執行 CPCA 求解專案網路 SRAP 問題，其解答甚至比 MCS_EM 的結果還要好。另 CLSA 的功能也在 (CPCA & CLSA) _EM 的實驗數據中彰顯出來，確實能進一步讓 EM 演算法的結果(專案網路總成本)再降低，且 EM 演算法並未出現嚴重的效率問題。上述的結果已足以證明，EM 演算法採用 CPCA 及 CLSA 確實能夠有效率的處理隨機性專案網路最佳化資源分配問題。

5.2 運用基因演算法求解多種類資源 SRAP 問題

本章節為前一章節問題的延伸，針對隨機性專案網路 SRAP 問題，在需要投入多種類資源的情況下，如何安排作業最佳化資源分配，本論文將本問題稱之為隨機性專案網路多種類資源最佳化分配問題(Stochastic network Multi-Resources Allocation Problem, SMRAP)。很顯然的，本章節所面對的資源分配問題要比只運用單種類資源的情況來得複雜；首先面對的問題是要如何定義出合理的專案網路成本計算模式，其次要決定採取何種工具執行最佳化資源分配決策。因每一種類之資源對專案網路作業時間的影響反應不同，且資源使用之單位成本亦不同，因此資源使用成本之計算，不同於單資源分配之成本計算模式。該問題並不容易運用一般數理分析模式直接求解，同前第 5.1 章節單種類資源分配問題一樣，需採用啟發式演算法。本章節多加入了多種資源種類之考慮，且運用在所有的專案作業中，因此系統的決策變數的總數會變得非常龐大，而且目標函數值無法直接以代數運算求得，故本章節擬採用基因演算法做為專案網路多種類資源分配之決策工具。

5.2.1 專案多種類資源分配問題之定義

本章節同 5.1.1 章節的所描述的問題情境一樣，針對隨機性專案網路，

執行作業最佳化資源分配之決策，唯不同處乃本章節所投入之資源為多種類，且每一種類資源對作業時間的影響亦不同。若本研究假設影響專案作業執行時間之「作業內涵」(Work content) 為「內在因子」，則除了人力資源因子外，還有「設備資源」、「材料資源」或其它有助於加快作業執行時間之資源項目。專案網路之「作業內涵」則不能再以「人力工時(Man-time)」為單位表示，應改為較通用之「資源工時(Resource-time)」表示之。因資源為多種類，故須將資源對專案網路作業時間影響之綜效(Synergy)進行定義；在決定各作業之各種資源投入量後，依綜效值決定其作業時間的機率分佈函數，後續方能決定整個專案網路完成時間的機率分佈函數。本研究的目標函數為專案總成本，其中包含所有作業之各種類資源使用成本的總合及逾規定時限之懲罰成本，相關之參數定義及關係式擬於下面章節說明之。

5.2.1.1 多種類資源的投入

本章節所處理的隨機性專案網路，計有 N 個作業，每個作業需投入多種類之資源。本研究假設共計 R 個種類之資源，且各種類之資源投入對作業時間的影響彼此沒有相依性，不會相互干擾。當多種類資源投入後，其資源對作業時間造成的綜效值設為 $X(r_1^i, r_2^i, ..., r_R^i)$，以作業 i 為例，假設作業時間為指數函數，其機率函數表示如下:

$$Z_i = \lambda_i X(r_1^i, r_2^i, ..., r_R^i) e^{-\lambda_i X(r_1^i, r_2^i, ..., r_R^i) t} \tag{5.13}$$

其作業時間平均值可表示為：

$$E[Z_i] = \frac{1}{\lambda_i \cdot X(r_1^i, r_2^i, ..., r_R^i)} \tag{5.14}$$

λ_i 仍為作業 i 之之作業時間參數， $r_k^i,\ k = 1, ..., R$ 為作業 i 第 k 種類資源投入量。Z_i 為 R 種類資源投入作業 i 後之作業時間機率分佈函數。另外本研究一假設專案網路中每一作業對各種類之資源投入量亦有上、下界範圍之限制要求，表示如下:

$$r_k^i \in [L_k^i,\ U_k^i],\ \ k = 1, ..., R,\ \ i = 1, ..., N \tag{5.15}$$

其中 L_k^i 為作業 i 第 k 種類資源的下限值，U_k^i 為作業 i 第 k 種類資源的上限值，若資源投入量 r_k^i 為負值表示減少作業 i 之 k 種類資源投入量，作業時間

因而增長。反之，若為正值，則表示增加該資源投入量，作業時間會因而縮減。故r_k^i通常會設定在一負值與正值的範圍內。

本研究假設資源綜效值$X(r_1^i, r_2^i, ..., r_R^i)$內，各種類之資源投入量$r_k^i, k=1,...,R$對作業時間彼此沒有相依性，即投入後對作業時間的影響不會彼此相互干擾。另外，不同種類之資源對作業時間的影響強度(Impact)亦會有所差別。以作業 i 為例，若所有資源種類均以等量方式投入，即$r_1^i = r_2^i, ..., = r_R^i$，不同資源種類對作業時間變化的影響大小應會有所不同，若將此情況反應在資源綜效值上，須針對不同種類之資源量加入一權重值$s_k, k=1,...,R$，以表現出該資源對作業時間的影響強度。權重值s_k的決定方式是依其資源對作業時間的影響強度在 0 與 1 的範圍內選取權值，並予以正規化(Normalize)，即$s_k \in (0, 1]$, $\sum_{k=1}^{R} s_k = 1$。另外有關於資源綜效值本研究尚有下列假設條件：

1. 當所有的資源種類投入量為 1，則表示作業沒有任何資源投入，作業時間應保持不變。
2. 資源綜效值在任何情況下不得出現負值, 否則作業時間會變成負值。

綜上所述，本研究以(5.16)式中之Δr^i方程式，反應出各資源種類對作業時間的影響強度及上述第一條假設條件之情況。另本研亦假設資源綜效值為Δr^i值的指數函數，如(5.16)式所列及圖 5.11 所示。

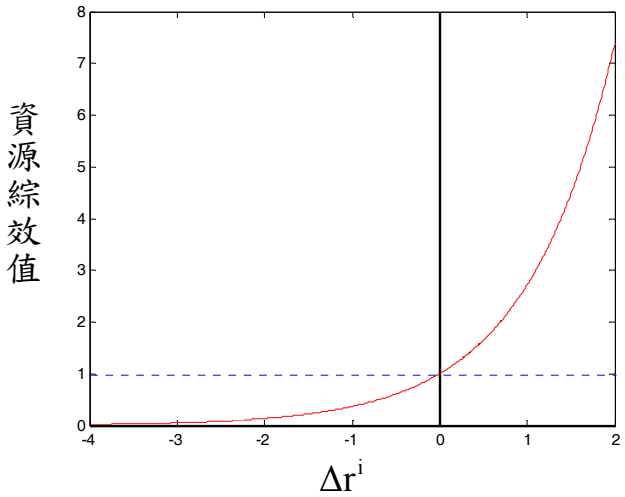


圖 5.11 資源綜效值對Δr^i之函數示意圖

$$\Delta r^i = (r_1^i - 1)\cdot s_1 + (r_2^i - 1)\cdot s_2 + \cdots + (r_R^i - 1)\cdot s_R$$
$$X(r_1^i, r_2^i, ..., r_R^i) = \exp(\Delta r^i) \tag{5.16}$$

(5.16)式除能反應上述資源綜效值兩項假設條件外，亦符合當資源投入愈多愈能縮減作業時間、愈少愈能增長作業時間之實務邏輯。

多種類資源投入的專案網路問題，其專案總成本同樣包括兩種項目：作業資源使用成本C_U及作業時間的逾時成本C_P。其中逾時成本的計算方法同第 5.1.2.2 章節(5.4)式，當專案完成平均時間$E[Y_N]$超過了所規定的限制時間T_z，則會產生一懲罰成本即為專案之逾時成本。專案網路中作業 i 資源使用成本C_{R_i}的計算型態，基本上亦類同於第 5.1.2.2 章節(5.3)式，即

$$C_{R_i} = W_i \cdot X(r_1^i, r_2^i, ..., r_R^i) \cdot U_{SC} \tag{5.17}$$

其中W_i為資源工時、$X(r_1^i, r_2^i, ..., r_R^i)$為資源綜效值及$U_{SC}$為「資源綜效單位使用成本」。$U_{SC}$為本研究作為多種類資源使用成本估算重要的計算數據，其原由及論點說明如下：

因為每一資源種類之單位使用成本$U_C^k,\ k = 1,...,R$均不同，因此每一資源種類所造成的資源使用成本$C_k^i,\ k = 1,...,R$亦不同，為了將個別的資源種類單位使用成本U_C^k反應於整體資源使用成本上，必須要將第 5.1.2.2 章節(5.3)式內之單位使用成本U_C轉變成資源綜效單位使用成本U_{SC}。因此計算多資源種類的資源使用成本，除了先透過(5.16)式求出資源綜效值$X(r_1^i, r_2^i, ..., r_R^i)$，還須決定出資源綜效單位使用成本$U_{SC}$值。在廈列的討論中，本研究將說明如何運用$U_C^k,\ k = 1,...,R$計算$U_{SC}$。

資源種類單位使用成本U_C^k可以一權重值w_k反應出其在資源綜效單位使用成本U_{SC}中的比例負擔；即比例愈大的資源種類，對整體資源使用成本所造成的成本負擔就愈重。但如何決定出各資源種類之比例負擔? 以作業 i 為例，首先定義權重值w_k與資源單位使用成本U_C^k之關係：

$$w_k = \frac{U_C^k}{\sum_{k=1}^{R} U_C^k}$$
$$\sum_{k=1}^{R} w_k = 1,\ \ k = 1,...,R \tag{5.18}$$

令U_{EC}為所有資源種類單位使用成本之期望值，

$$U_{EC}=\sum_{k=1}^{R}(U_C^k\cdot w_k) \tag{5.19}$$

當各種類資源投入量r_K^i正好符合(5.20)式比例關係，則資源綜效單位使用成本會等於資源種類單位使用成本之期望值，即$U_{SC}=U_{EC}$。

$$r_1^i\cdot w_1=r_2^i\cdot w_2=\cdots=r_R^i\cdot w_R \tag{5.20}$$

若各種類資源投入量不合乎(5.20)式比例，則資源綜效單位使用成本U_{SC}究不能再以U_{EC}視之，需利用各資源種類實際投入量$r_K^i, k=1,...,R$之比例關係，重新調整計算原本之權重值w_k得$w_k^{'}$。以作業i為例，w_k之調整計算步驟如下：

1. 先找出資源單位使用成本權重值最小的資源類別項目 g，並令其權重值為w_g：

$$g=\arg\min(w_k),\ k=1,...,R \tag{5.21}$$

2. 在符合(5.20)式比例情況下，令g項目資源類別之投入量為$\hat{r}_g^i$，並作為基準，分別計算其它資源類別之投入量$\hat{r}_g^i,\ k=1,...,R,\ k\neq g$。

$$\hat{r}_g^i=\frac{\hat{r}_g^i\cdot w_g}{w_k},\ k=1,...,R,\ k\neq g \tag{5.22}$$

3. 當資源種類投入量符合$\hat{r}_1^i$ ： $\hat{r}_2^i$ ： ...：$\hat{r}_R^i$之比例，專案網路之資源綜效單位使用成本$U_{SC}=U_{EC}$。因其他之資源投入量$r_k^i,\ k=1,...,R,\ k\neq g$並未能符合上述比例關係，因此必須透過下列之計算，調整$r_k^i,\ k=1,...,R,\ k\neq g$相對之權重值$w_k^{'}$：

$$\begin{aligned}&\frac{r_1^i}{(\frac{\hat{r}_1^i}{w_1})}:\frac{r_2^i}{(\frac{\hat{r}_2^i}{w_2})}:...:\frac{r_R^i}{(\frac{\hat{r}_R^i}{w_R})}=t_1:t_2:...:t_R\\&w_1^{'}:w_2^{'}:...:w_R^{'}=\frac{t_1}{\sum_{k=1}^{R}t_k}:\frac{t_2}{\sum_{k=1}^{R}t_k}:...:\frac{t_R}{\sum_{k=1}^{R}t_k}\end{aligned} \tag{5.23}$$

4. 資源綜效單位使用成本U_{SC}即為配合新產生之權重值$w_k^{'}$，所計算出之資源種類單位使用成本之期望值，算式如下：

$$U_{SC} = \sum_{k=1}^{R} (U_C^k \cdot w_k^{'}) \tag{5.24}$$

專案網路作業 i 之資源使用成本依據(5.17)式可以表示為：

$$\begin{aligned} C_{R_i} &= W_i \cdot X(r_1^i, r_2^i, ..., r_R^i) \cdot U_{SC} \\ &= X(r_1^i, r_2^i, ..., r_R^i)^2 \cdot Z_i \cdot U_{SC} \\ &= X(r_1^i, r_2^i, ..., r_R^i)^2 \cdot Z_i \cdot \sum_{k=1}^{R} (U_C^k \cdot w_k^{'}) \\ Z_i &= \frac{W_i}{X(r_1^i, r_2^i, ..., r_R^i)} \end{aligned} \tag{5.25}$$

最後將所有專案網路中之 N 個作業使用成本加總即可求得專案網路資源使用成本 C_U，表示如下：

$$C_U = \sum_{i=1}^{N} C_{R_i}, \ i = 1, ..., N \tag{5.26}$$

專案網路的總成本 C_T 為所有作業各種類之資源使用成本及作業時間的懲罰成本之和，即 $C_T = C_U + C_P$。上述 Y_N 為隨機性專案網路完成時間機率分佈函數，同樣由第三章之標籤修訂演算法(LCTA)來執行估算求之。多種類資源最佳化分配的目的，就是要決定專案作業中各種類資源最佳的投入量，以求得最小的總成本 C_T。

5.2.1.2 多種類資源投入之範例

以圖 5.2 及(5.2)式相關之作業參數為例，且針對多種類資源的相關假設條件如下：

1. 資源種類：R=3。

2. 所有作業之資源種類上、下界值均為 $[L_k^i, U_k^i] = [0.5, 4], \ k = 1, ..., 3$

3. 資源綜效權重值：$s_1 = 0.25, s_2 = 0.4, \ s_3 = 0.35$。

4. 各種類資源單位使用成本：$U_C^1 = \$2, U_C^2 = \$4, U_C^3 = \$3$。

5. 懲罰單位成本：$P_C = \$5$

若假設專案網路內所有作業之各種類資源投入量為 1，以作業 1 為例，即 $r_1^1 = 1, r_2^1 = 1, r_3^1 = 1$，依 (5.18) 式，計算資源權重值為 $w_1 = 0.22, w_2 = 0.45, w_3 = 0.33$，再依(5.19)式，資源單位使用成本期望值為 $U_{EC} = \sum_{k=1}^{3} (U_C^k \cdot w_k)$=\$3.23。根據(5.21)及(5.22)式，最小的資源種類權重值項目為 g=1, 則符合(5.20)式比例之資源投入量為 $\hat{r}_1^1 : \hat{r}_2^1 : \hat{r}_3^1 = 1 : 0.49 : 0.67$。接

續參考(5.23)式，將新的資源種類權重值計算出來：$w_1^{'} : w_2^{'} : w_3^{'} = 0.14 : 0.56 : 0.3$，最後得$U_{SC} = \sum_{k=1}^{3}(U_C^k \cdot w_k^{'}) = 3.42$。再根據(5.25)式，作業 1 的資源使用成本$C_{R_i} = 1^2 \cdot 8 \cdot 3.42 = \$27.36$。

5.2.2 基因演算法介紹

在1960 年代，John Von Neuman 提出一個自我複製(Self-reproducing)的理論而奠定了基因演算法(Genetic Algorithm, GA)的基礎；之後John Holland(1975) 更延續此觀念，並輔以達爾文的進化論「物競天擇、適者生存」，而在1970 年發展出簡易基因演算法，使基因演算法的架構有了初步的雛形。由於基因演算法提供一個相當簡單的系統架構、運作流程，就可以產生強大的搜尋能力，和必須要依附問題模式的傳統演算法有明顯的不同，這種彈性也是其它方法所不及的。同時基因演算法由同一代的各個染色體同時探索不同的區域，再伴隨著世代演化交替、隨機搜尋的特性，這種平行處理的能力使它不容易陷入局部最佳化(Local optimal)的困境，而向整體最佳解(Global optimal)收斂；而它的缺點則在於計算時間長，但是此缺點由於計算機的運算大幅增快而慢慢地被克服了；這些特點都讓基因演算法被廣泛地應用在各個領域。

一般在利用基因演算法來解決問題時，要先考慮到下列幾項組成要素：

1. 解決問題之染色體表示方式，即編碼及解碼方式。
2. 起始解產生及群體數目大小決定，亦即初始群組(Initial population)。
3. 如何適當定義出評估問題解的函數，亦即適應函數。
4. 使用基因運算子(Genetic operators)產生子代(Off-springs)。
5. 控制參數的設定(例如：母體大小、交配率及突變率等)。

本章節先介紹基因演算法之基本架構，在針對上述五個組成要項之內容，介紹如何運用該演算法處理隨機性專案網路多資源種類之SRAP問題。

5.2.2.1 基因演算法求解架構

GA流程圖如圖5.12所示。首先，輸入演算法參數，包括群組大小(Population size)M、交配率(Crossover rate)P_c、突變率(Mutation rate)P_m與

終止條件。輸入參數後，計算染色體位元數，進行資源投入量集合 $re\_al^j = \{x_r^i\},\ i=1,...,N,\ r=1,...,R,\ j=1,...,M$ 的編碼，產生染色體數為 M 的初始群組。計算第 j 條染色體的適應值 $F_j = T_C(re\_al^j)$，進入「輪盤選擇」。輪盤選擇的機制在挑選基因較優的染色體為父代，並根據交配率 P_c 參與「交配」運算。染色體交配後，在根據突變率 P_m 進行「突變」運算，以避免搜尋最佳解的過程中，掉入區域最佳解區域。每代染色體進行完「選擇」、「交配」、「突變」等機制後，會產生新一代之群組 $re\_al^j, j=1,...,M$，再重新針對群組內每條染色體計算其適應值 F_j。重複上述「選擇」、「交配」、「突變」等機制，直到滿足終止條件停止GA運算。GA終止條件通常以設定最大執行代數為之，或所求出的解已無再進一步的空間。

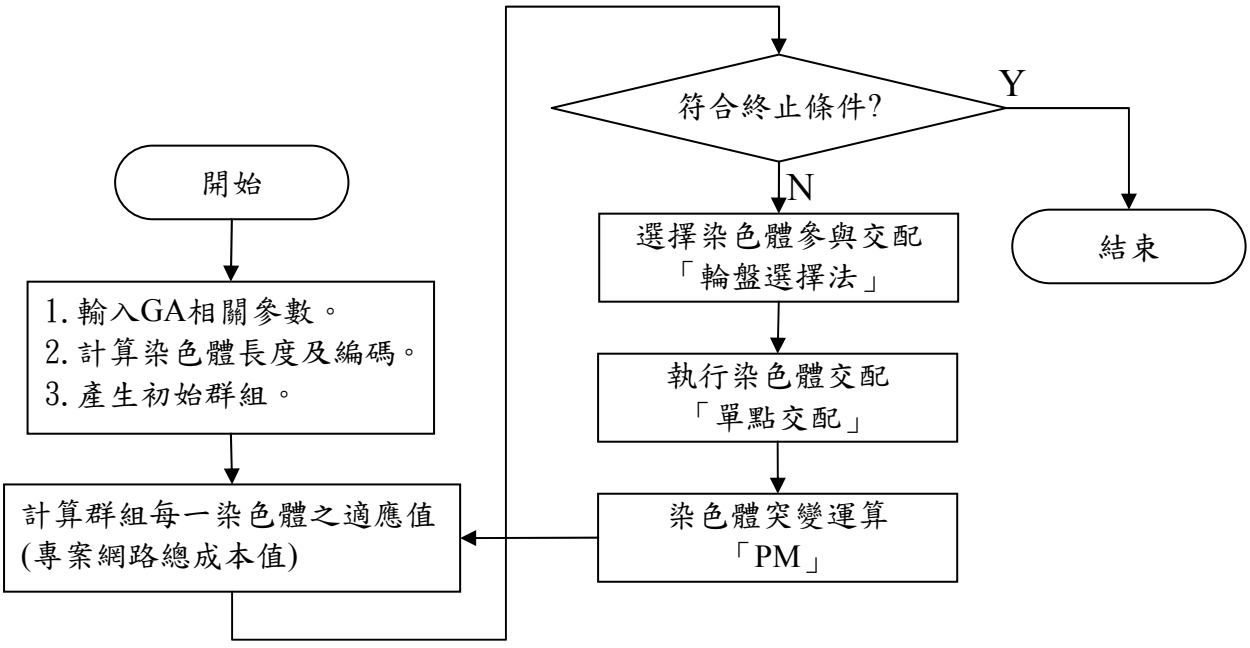


圖5.12　基因演算法流程圖

5.2.2.2 參數設定

GA 主要包括五項參數：群組大小、決策變數之位元數、交配率、突變率與終止條件。

1. 群組大小(M)：群組是由染色體所構成的，而每個染色體都分別代表想要求解的問題中的一個暫存解；一群染色體的演化過程，相當於一組暫存

解同時在解空間做平行搜尋，因此母體大小就決定了每一代所擁有的搜尋點個數。如果母體大小增加可增加母體的異樣程度，使得陷入局部解的機率減少，但同時也可增加搜尋最佳解的時間。在選擇母體大小的考量上，必須兼顧求解效率及運算時間兩者的平衡。本章節群組染色體令為 M=20 條。

2. 決策變數之位元數 b：每一決策變數以 b 個二進位數來表示，所有的決策變數集合形成一個染色體。決策變數的解析度由 2^b 位階數目決定，b 值愈大，解答的解析度會愈高，但染色體的總位元數亦會跟著增加而影響 GA 整體的運算效率。因此決策變數之位元數依問題性質取適量即可。本章節令 b=5。

3. 交配率（P_c）：交配率是指群組中的染色體需要進行交配操作的機率。較高的交配率能使得較多的染色體進行交配，而產生較多新結構的染色體，當然也較容易使得好的染色體破壞原本的結構；而太低的交配率，則會阻礙搜尋的速度，本章節令 $P_c = 0.65$。

4. 突變率（P_m）：突變率是指群體中的染色體需要進行突變操作的機率。增加突變率有助於重新引入未曾搜尋過的基因架構，但過高的突變率，也可能導致基因演算法變成隨機搜尋法，本章節令 $P_m = 0.01$。

5. 終止條件：經過連續 500 代的演化，最佳解都沒有改變時，即可視為找到最佳解，則停止搜尋。

5.2.2.3 編碼表示法及解碼

基因演算法運作的對象通常為能夠表示可行解的字串而非決策變數本身，因此在執行基因演算法之前必須透過編碼的程序將問題的決策變數轉換成字串。在求解適合度值時，再將字串解碼為實際的決策變數。GA通常採二位元的編碼，即將問題決策變數以{0, 1}字元串表示成GA染色體。因此專案網路要將所有作業投入之資源種類及數量，表現在染色體的位元上。以圖5.13為例，令資源總類為R=3，8個作業染色體內的變數共計24個。若每個變數以b=5個位元來表示，則資源投入量可以分成 $2^5 = 32$ 個位階(Level)。

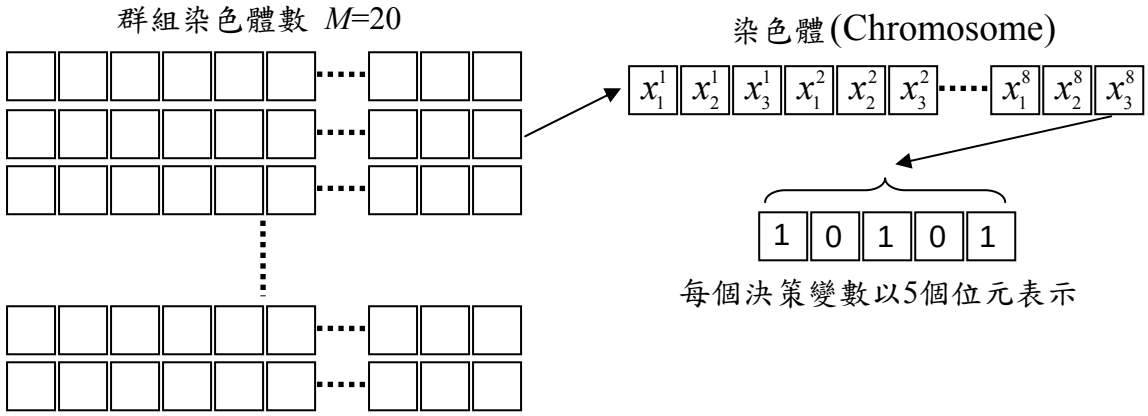


圖5.13　群組染色體編碼示意圖

編碼後的染色體是方便 GA 執行「選擇」、「交配」、「突變」等運算使用，要計算專案總成本產生染色體適應值，需將染色體的位元編碼，依決策變數的數值範圍$[L_r^i, U_r^i]$解碼回實際數值，解碼的公式如下:

$$x_r^i = L_r^i + \text{Dec}(Chromosome) \cdot \frac{U_r^i - L_r^i}{2^b - 1} \tag{5.27}$$

其中 Dec(*Chromosome*) 能將二位元數轉成十進位數。

5.2.2.4 初始群組

在執行基因演算法之前，必須先建立一組包含多個染色體的初始群組(Initial population)，每個染色體即代表一組問題的解答，接著再從每個染色體間多方向地演化，產生較佳的下一代。初始群組通常是以隨機方式產生，即其內所有染色體的位元數均隨機取0或1表示，而族群的大小M為事先給定的參數值。

5.2.2.5 適應函數

適應函數(Fitness function)為評判染色體優劣的依據，透過適應函數的運算，可得適應值。適應值越佳者越有被選取的機會；反之，適應值越差的染色體則越容易被淘汰。一般而言適應函數乃是決策變數的目標函數。在基因演算法的過程中並不處理限制條件，因此碰到不合乎限制條件的字串，通常加諸一懲罰成本，使不適合的染色體能被自然淘汰。本章節的目標函數為專案網路投入資源組合後的專案總成本，GA群組內每一條染色體均表示一種專案網路投入資源組合，將其解碼後產生$re_al^j, j = 1,...,M$，同

5.1章節EM樣本點求解，再使用LCTA演算法並配合(5.13)~(5.26)式便可求出目標函數適應值$F^j, j=1,...,M$。

5.2.2.6 輪盤選擇機制

選擇(Selection)機制是人工遺傳系統模仿自然界「適者生存」的工具，它的主要功能是用來確定某個染色體被複製的個數。適應函數和選擇機制兩者配合使用，即可實現基因演算法中的複製操作過程。本章節採用輪盤選擇機制（Roulette wheel mechanism），是透過機率原理，將染色體適應值，視為輪盤面積，因此在求解的過程中，選擇輪盤面積較大的染色體，即適應能力較高的染色體；淘汰輪盤面積較小的染色體，即適應能力差的染色體。輪盤選擇機制以圖5.14為例。先隨機產生一變數$\beta \in \left(0, \sum F_j\right)$，並判斷$\beta$屬於哪一區間。如果$R=1.724 \in (0.8, 2.0)$，則選擇第2條染色體。

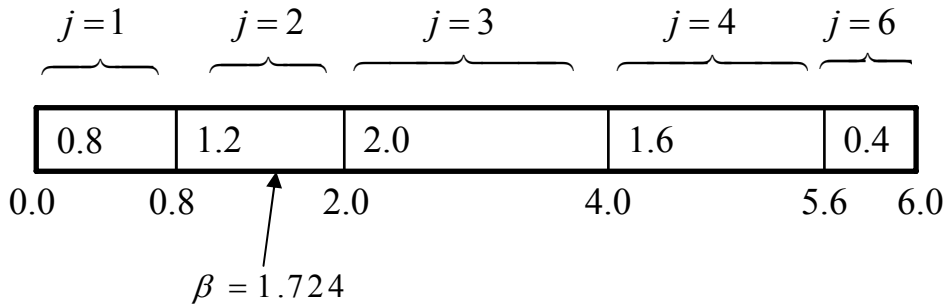


圖5.14　GA輪盤選擇機制示意圖

5.2.2.7 交配機制

交配提供個體間彼此交換訊息，基因重組以產生新個體的功能。在經過複製選取出來的新群體中，以隨機的方式兩兩配對，然後在每組成對個體的染色體上，隨機選取一個或多個交配點(Crossover point)，基因互換產生一對新的個體。交配運算有多種類型，本章節採用單點交配(Single-point crossover)，將任意2條交配候選染色體視為一組，形成交配組合，再進行單點交配機制。以隨機方式找出交配點（Crossover point），然後將交配點兩邊位元進行交換，以圖5.15說明。將染色體從交配點$pos=4$切成左右兩邊，再將交配組合內的染色體兩邊位元進行互換。

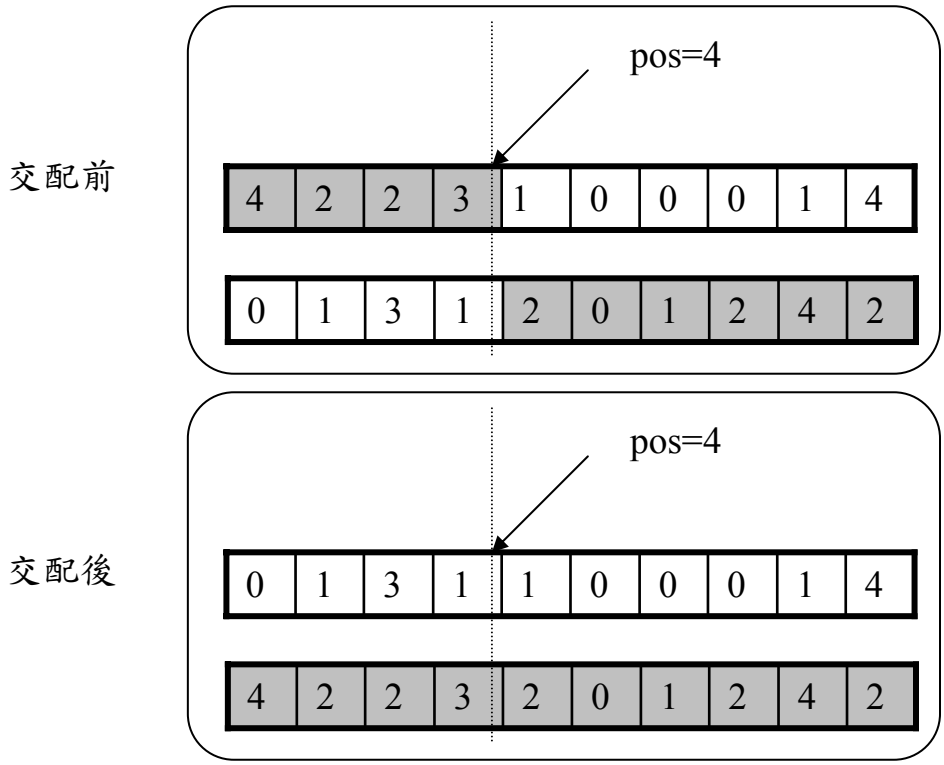


圖 5.15　GA 單點交配演算示意圖

5.2.2.8 突變機制

突變過程是將交配後所產生的子代，根據預設的突變機率進行突變，其主要目的是避免過早收斂，且可以開發新的搜尋區域，防止收斂於局部最佳解。其作法是先隨機產生機率值$\alpha \in (0,1)$，若α小於突變率P_m，則選取該條染色體的位元進行變化，以圖5.16說明。假設$K^{UB}=16$，突變點7的位元值為$p_7=2$，隨機產生機率值$\alpha \in \left(0,\log_2\left(K^{UB}\right)+1\right)$=0.724，再修正$P_7=[0.724]=0$。

5.2.3 多種類資源 SRAP 問題運用 CPCA 產生 GA 初始解

參考第 5.1.2.2 章節，指出當 EM 決策變數搜尋的空間過大時，會影響 EM 初始樣本點收斂至最佳解的速度。GA 雖無決策變數搜尋空間過大之問題，但其內之決策變數須化成二位元表示之染色體(*Chromosome*)數；當決策變數愈多且求解之精確度要求愈高時，染色體的二位元數目就擴增愈多，同樣會影響 GA 收斂至最佳解之速度，在此情況下決定適當之 GA 合理初始解(Feasible solutions)，也是非常重要的工作。

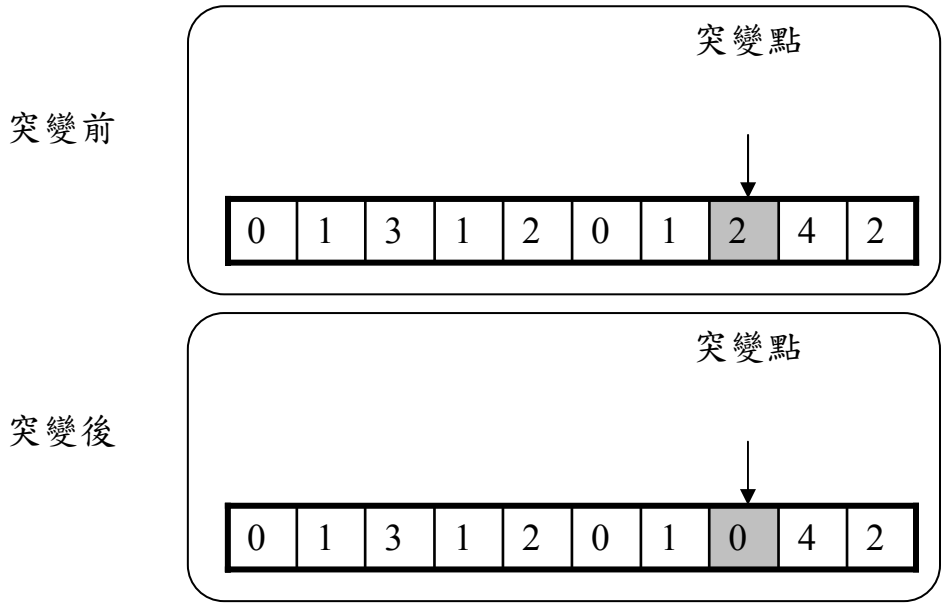


圖 5.16　GA 突變演算示意圖

第 5.1.4 章節中，針對 EM 解單種類資源 SRAP 問題，採用關鍵路徑叢集演算法(CPCA)所產生之初始解亦同樣能適用多種類資源 SRAP 問題。該章節中，演算法 1~5 所執行的流程、計算程序及所採用的三種策略均完全一樣，唯有兩處須做調整：

1. 演算法 3~5 中，執行專案網路 CPC 內作業資源量微調，乃針對 R 種類之資源做考量。以演算法 3 中之(A)標籤列為例，應改為下式:

$$\begin{aligned} &\lambda^* = (\lambda_1, \lambda_2, ..., \lambda_R),\ \lambda_k \in U(0,1),\ k = 1,...,R \\ &re\_al^i = L^j + \lambda^* \cdot (U^j - L^j),\ i = 1,...,m,\ j \in A_N^i \end{aligned} \tag{5.28}$$

參與微調之作業，隨機投入之資源種類須考慮 R 種，因此所產生之資源投入組合 re_al^i 結構與原單資源種類會不一樣。其後接續計算專案網路之 T_R^i, C_U^i, C_P^i 及 C_T^i，亦須以第 5.2.1.1 章節中之公式執行之。

2. 專案網路總成本對多種類資源投入量較為敏感，資源量些微調整便會讓專案完成時間產生較大的變化，進而影響專案成本的計算。因此 CPCA 在第 5.1.4 章節演算法 3 中執行 Strategy_1( )，針對專案網路 CPC 內作業資源量微調須較耗時間，重複計算的步驟亦較多。演算法 3 中之 UP_ratio 值，是判斷終止執行 Strategy_1( )的其中一條件，當專案網路資源使用成本接近懲罰成本時，即停止 Strategy_1( )的執行。在單種類資源的情況下，作業資源變化量，造成專案網路資源使用成本及懲罰成本變化量之關係較為單純，因此以 UP_ratio 值可作為專案網路作業資源停止調整的

判斷標準。然而在多種類資源的情況下，作業資源變化量對專案網路資源使用成本變動甚巨，因此與專案網路懲罰成本變化量間之關係較為複雜，因此不擬採UP_ratio值作為專案網路作業資源停止調整的判斷標準，而是以每執行一次 Strategy_1( )計算，所得之專案網路總成本值是否獲改善來做為考量之基準，若連續 5 次執行 Strategy_1( )計算均未獲改善則停止演算法 3。

依上述之說明，多種類資源 SMRAP 所採用的 CPCA 演算法，其中 Strategy_1( )、Strategy_2( )及 Strategy_3( )之演算程序應修改如下：

Strategy_1( )

Iter=0;

$C_T^{i'}=C_T^{ini}$;

$C_T^{best}=C_T^{ini}$;

While ($Test^i>\varepsilon\%$)& ($iter>5$)

$\lambda^*=(\lambda_1,\lambda_2,...,\lambda_R),\ \lambda_k\in U(0,1),\ k=1,...,R$

$re\_al^i=L^j+\lambda^*\cdot(U^j-L^j),\ i=1,...,m,\ j\in A_N^i$; (A)

重新計算專案網路之T_R^i, C_U^i, C_P^i及C_T^i

If $C_T^i<C_T^{i'}$

$C_T^{i'}=C_T^i$;

Iter=0;

Else

Iter= *Iter* +1;

End

If $T_R^i>T_Z$

$U^j=L^j+D\_rate\cdot(U^j-L^j)$, $j\in A_N^i$ ； (B)

End

If $C_T^i<C_T^{best}$

$C_T^{best}=C_T^i$;

$Re\_Al^{best}=re\_al^i$;

End

$Test^i=\dfrac{(U^j-L^j)}{L^j},\ \ j\in A_N^i$**;**

End

Strategy_2( )

投入資源組合X^i與專案網路中，重新計算專案網路之 CPC(A^i);

If $A^{i+1} = A^i$

$$M^j = M^j + I\_rate \cdot (U^j - L^j); \quad \text{(A)}$$

$$\beta^* = (\beta_1, \beta_2, ..., \beta_R),\ \beta_k \in U(0,1),\ k = 1,...,R;$$

$$re\_al^{i+1} = L^j + \beta^* \cdot (M^j - L^j),\ i = 1,...,m-1,\ \ j \in \{A_O^i\}; \quad \text{(B)}$$

Else

$$M^j = M^j + I\_rate \times (U^j - L^j);$$

$$\beta^* = (\beta_1, \beta_2, ..., \beta_R),\ \beta_k \in U(0,1),\ k = 1,...,R;$$

$$re\_al^{i+1} = L^j + \beta^* \cdot (M^j - L^j),\ i = 1,...,m$$

$$j \in A_N^{i+1} \in \{B_h^i |\ T_h^i = \max(T_k^i),\ h, k = 1,...,Q\}; \quad \text{(C)}$$

End

Strategy_3( )

$$M^j = M^j - I\_rate \cdot (U^j - L^j); \quad \text{(A)}$$

$$\lambda^* = (\lambda_1, \lambda_2, ..., \lambda_R),\ \lambda_k \in U(0,1),\ k = 1,...,R;$$

$$re\_al^{i+1} = L^j + \gamma^* \cdot (M^j - L^j),\ i = 1,...,m-1$$

$$j \in \{a_z \mid a_z \Leftarrow B^i, z = 1,...,w\}; \quad \text{(B)}$$

5.2.4 實驗數據

針對 SMRAP，因無其它相關文獻可提供作為本章節實驗比較對象，因此仍以第 5.1.6 章節中 Terreso, et al. (2004b)文獻所提供 14 個大小不同的專案網路實體(作業數從 3 到 76 個)為實驗對象。專案網路作業時間均假設為指數分佈函數，時間參數、網路完成時限T_z及懲罰單位成本P_C均參閱附錄 L 所列，資源使用單位成本U_C則依第 5.2.1.1 章節中(5.21)式計算之。網路作業所投入之資源種類定為 R=3，每一種類資源投入之上、下界值同附錄 L 中所規定。實驗所使用的個人電腦仍為 Intel Pentum-4 2.0 GHZ CPU, 512MB 記憶體並使用 Matlab Ver.6.5 語言程式來執行。

本章節所採用之 GA 演算法，其主要參數設定為：群組大小 M=20、決策變數(資源投入量)之位元數 b=5、交配率 P_c=0.65 並採用單點交配、突變

率 P_m=0.01、選擇機制採用輪盤法則、終止條件為執行代數 g=500。實驗的內容首先針對上述 14 個專案網路實體，以隨機取樣的方式產生 20 個 GA 初始染色體，再執行 GA 求解，並紀錄求解時間、最小專案網路資源使用成本、懲罰成本及總成本，該實驗以 Randomly_GA 表示。另在相同條件下，以 CPCA 演算法求得 20 個 GA 初始染色體，接續再執行 GA 求解(以 CPCA_GA 做表示)，同樣紀錄上述相關之結果，最後再比較上述兩種實驗的結果。實驗所得的結果分別表示於表 5.6 及表 5.7。其中 Randomly_GA 求解結果所得的專案總成本及執行時間，與 CPCA_GA 求解結果做比較，有相當一個落差，相關的說明整理如下：

1. Randomly_GA 若將 20 個初始解均以隨機方式產生，則 GA 執行 500 代所得的結果非常的差，其原因乃決策變數量過大致使 GA 不易收斂之故。故本研究自隨機產生之初始解中，選擇中一個染色體並將其內之二位元數值均設為 0，即表示專案網路所有的作業資源量均以最低限制量投入(等於 0.5)，此舉會產生較小的專案總成本，能令 GA 在執行疊代運算的過程中有機會往較小的總成本解邁進。
2. 修飾後的 Randomly_GA 於 14 個專案網路項目所得的專案總成本結果，均仍比 CPCA_GA 來得差，該差異似乎會隨專案網路作業量增大而增加。
3. CPCA_GA 因要額外執行 CPCA 演算求取 GA 20 個初始染色體，因此執行時間要比 Randomly_GA 來得多，基本上該兩實驗的執行時間相減即為 CPCA 演算法的執行時間。CPCA_GA 平均比 Randomly_GA 多出 45%的執行時間。
4. 單獨使用 CPCA 演算法所產生之 GA 初始染色體，所得到的最小專案網路總成本，除第 1 及 13 項答案一樣外，其餘均比修飾後的 Randomly_GA 執行 GA 演算 500 代後所得的結果還來得好，這情況顯示單獨使用 CPCA 即可求解 SMRAP，且可得不錯的結果。
5. 若將 CPCA_GA 執行 500 代 GA 演算所得知結果，與 CPCA 演算法所得到的最小專案網路總成本做比較，差異並不多。

表 5.6　Randomly_GA 求解結果

網路編號	作業數	總成本	資源使用成本	懲罰成本	執行時間
14	76	1184.5	671.1	513.4	1882.4
13	38	1729.0	813.1	915.8	1075.1
12	24	3487.8	1898.8	1589.0	818.2
11	18	1200.7	582.0	618.7	540.3
10	17	390.7	238.5	152.2	397.8
9	14	1764.7	882.2	882.4	424.3
8	14	238.6	135.1	103.5	321.9
7	12	393.8	168.2	225.6	269.7
6	11	684.8	328.1	356.7	280.4
5	11	315.9	231.1	84.8	274.2
4	9	916.6	470.0	446.5	230.4
3	7	530.1	232.1	298.0	183.8
2	5	812.4	604.6	207.7	127.1
1	3	80.6	39.3	41.3	68.9

表 5.7　CPCA_GA 求解結果

網路編號	CPCA 最佳初始解	總成本	資源使用成本	懲罰成本	執行時間
14	1170.2	1166.8	740.9	425.9	2101.1
13	1729.0	1729.0	813.1	915.8	1532.9
12	3472.8	3468.7	2151.5	1317.2	1241.1
11	1119.9	1123.8	662.0	461.7	701.1
10	368.4	367.5	223.3	144.1	561.1
9	1728.0	1725.7	978.7	747.0	699.2
8	236.7	236.2	138.8	97.4	559.0
7	392.0	389.2	211.1	178.1	352.4
6	679.7	677.5	320.3	357.2	482.5
5	306.1	306.4	158.8	147.6	362.6
4	883.0	883.6	484.1	399.6	391.8
3	520.7	515.3	272.3	243.1	290.3
2	829.7	794.6	582.6	212.0	176.8
1	80.6	80.5	40.1	40.4	76.9

5.2.5 結論

SRAP 已無法使用確定性專案網路分析工具或數理分析方式求解，更遑論處理 SMRAP 問題，因此根據第二章文獻探討內容，唯採用啟發式的

解法是較為可行的方式。因 SMRAP 的決策變數擴展非常的大，故本章節並不採用 EM 演算法而以 GA 演算法取代之。GA 之決策變數是以二位元數來表示，數目愈多則求解精度愈大，反之愈小。因此決策變數過大仍會使 GA 染色體的位元數目增加，同 EM 一樣，因而仍會影響其求解之收斂速度。因此本章節首要之貢獻即將前一章節所發展之 CPCA 演算法，嚐試處理多種類資源 SRAP 之問題。

每一種類之資源對專案網路作業時間變化反應不同，且資源使用之單位成本亦不同，因此多種類資源專案網路資源使用成本之計算，與單種類資源分配之成本計算模式截然不同。本文以第 5.2.1 章節之內容，以作業時間仍為指數分配的假設條件下，提出隨機性專案網路多種類資源使用成本計算之機制，並搭配本論文第三章之 LCTA 演算法估算隨機性專案網路時間，藉以計算出專案懲罰成本，最後將專案網路總成本計算出來，該計算程序即可作為 GA 求解染色體之目標適應值(Fitness values)，此亦為本章節針對 SMRAP 所提出之第二個主要貢獻。

GA 在龐大的決策變數環境下，並不容易收斂至最佳解處，或許需配合執行其他種類之選擇機制(Selection operator)、交配機制(Crossover operator)、突變機制(Mutated operator)及其它相關參數的變化，然而這並非為本章節討論的標的。本章節主要是要驗證 CPCA 演算法結合 LCTA 演算法，是否仍能適用於 SMRAP。從上述實驗數據結果顯示，CPCA 所產生之 GA 初始染色體，確實能夠大幅提升 GA 的求解品質，且亦不耗費過多的執行時間，尤其對於大型專案網路更能夠彰顯其效果。另單獨執行 CPCA 所得到的結果，即已比 GA 採隨機取樣之初始染色體，執行 500 代最後所得的結果來得好。該結果亦與 CPCA_GA 執行 500 代所得之結果差距不大，這非常明顯的証明 CPCA 確實能夠有效率的處理 SMRAP 問題。

第六章　論文未來發展方向

本論文乃針對隨機性專案網路處理專案網路完成時間及作業資源最佳化分配問題(SRAP)，且發展出新的演算法並獲得到良好的結果。尤其隨機性專案網路完成時間能獲有效的估算，許多相關專案網路應用問題(專案成本分析、風險評估等)便能藉以作進一步有效的處理，第五章專案網路作業資源最佳化分配問題亦是其應用之一。然而就專案管理者而言，專案網路關鍵路徑或關鍵作業仍會是其所關心的項目，雖本論文第五章已指出關鍵路徑並不一定是隨機性專案網路中最長(或最短)的路徑時間，其中作業也不一定是資源最優先投入的標的，對專案管理者而言參考的價值不大，但若提供專案網路最前三名關鍵路徑，則該訊息仍具有管理上的涵義。該問題在隨機性專案網路的領域中被稱為前 K 條最短路徑問題(K Shorest path Problem, KSP)，因此關鍵路徑指標(PCI)在隨機性專案網路中仍有其價值及參考的必要性。同樣對專案管理者而言，隨機性專案網路關鍵作業指標(ACI)與 PCI 相較，其參考價值較大，因其求解過程較為複雜，也較受到專案研究領域之重視。ACI 值能指出對專案網路完成時間較敏感的作業，除可藉以提供資源投入之參考，亦可執行專案網路完成時間之敏感度分析。本論文擬利用第三章所發展出來的 LCTA 為基礎，將隨機性專案網路 PCI 及 ACI 之估算列為未來後續研究之重點之一，研究該領域的文獻及其相關的發展情況擬詳述如附錄 M 所列。

本論文的第二部份為隨機性專案網路 SRAP 問題，其決策情境仍假設資源在專案執行的任何時間點，在作業可接受的資源範圍內，均能無限制的供應。然而就實務上而言，無論資源能否回收或重複使用，通常會有總量的限制，意即專案在執行的過程中，若同時有多個作業提出資源需求，在有限的資源情況下，就必須做資源投入排序之決策，這就形成了隨機性專案網路排程問題(Scheduling Problem)。就本論文之瞭解，研究該問題的文獻非常少，僅 Dimitri & Aharon(1997, 1998, 2003), Bowers (1995, 1996)。本論文同樣擬以 LCTA 為基礎，將該問題列為未來後續研究之項目。

本論文第五章所論述之關鍵路徑叢集(CPC)理論，對隨機性專案網路資源分配問題乃屬新的概念。本論文以該理論所發展之關鍵路徑叢集演算法(CPCA)，其演算的邏輯實際上屬演化演算法(Evolution Aogorithm, EA)之種類，其特性是要求現行解(Current solution)保證會比前一代來得好，且演算

過程中須有讓現行解跳脫區域最佳解的演算機制。從第五章表 5.5 及表 5.7 實驗結果顯示，針對 Terreso, et al. (2004b)中 14 個專案網路中，單種類資源部份有 8 組比 MCS_EM 的最後解答還來得好，在多種類資源實驗中則全部的結果都比 Randomly_GA 此的最後解答還來得好。此證明 CPCA 對隨機性專案網路 SRAP 及 SMRAP 問題的求解能力。若能進一步細究隨機性專案網路資源投入之特性，並針對第 5.1.4 章節內 CPCA 三種資源投入策略做調整，則 CPCA 就能單獨應用於 SRAP 及 SMRAP 問題，無須再配合任何較費時的啟發式演算法來作為其決策工具。此部分亦為本論文針對專案網路 SRAP 及 SMRAP 問題，作為未來後續研究之目標。

第七章　結論

在面臨內容多元及規模大型化之專案環境中，專案之作業會因內、外在不確定性因素影響，往往與所規劃之目標產生落差，其結果不僅造成計畫的延遲，更讓所投入之專案成本負擔增加。對專案管理者而言，在專案規劃的階段中，如何能準確的評估專案完成時間及節省專案成本考量的情況下，決定專案作業資源的投入量，是非常重要的工作。鑒於此，針對隨機性專案網路的相關研究就顯得相當重要。

面對上述問題，隨機性專案網路因作業時間為隨機變數之故，其相關之估算工作會變得非常複雜，故無法以確定性專案網路之要徑法(CPM)或直接以數理計算方式為之。計畫評核術(PERT)為最廣泛、也最受企業界青睞之專案網路分析工具，然而已有若干文獻及本論文之實驗證明，PERT 對專案網路完成時間之估算與實際值間存在非常大的誤差。另為配合實務之考量，本論文更額外進一步要求下列 4 項隨機性專案網路之執行條件：

1. 專案網路之作業時間，不能僅侷限在任何特定型態之機率分佈函數。
2. 專案網路完成時間除能估算其平均值外，亦要求其時間機率分佈結果。
3. 針對大型專案網路，除要求估算品質外，其估算的執行時間亦要在合理的範圍下完成。
4. 專案網路完成路徑間之相依性須納入考慮。

就本論文之瞭解，迄今尚無任何相關文獻，在上述執行條件要求下，提出有效率的演算法。

針對上述隨機專案網路問題及執行條件，本論文已發展估算隨機性專案網路完成時間之演算工具，並提出兩種演算法：

1. 標籤修訂追蹤演算法(LCTA)。
2. 區域積分演算法(MIA)。

上述兩種演算法均採用離散化技術(Discritized technique)，將專案網路作業時間機率分佈以柴比契夫取樣點法執行取樣，並結合重新取樣技術(Resampling technique)，以離散化之 max 及 convolution 運算執行時間疊代計算。其中 LCTA 利用圖形理論中之樹形結構，依後序追蹤方式，有系統的完成專案網路完成時間之估算。LCTA 另一項貢獻乃利用上述之演算技術，將專案網路路徑相依性納入估算。MIA 繼承 LCTA 之演算原則，並針

對疊代計算中之 max 運算做改良。該運算除較耗時外，其本身所產生之最長路徑偏差(LPB)效應，是估算隨機性專案網路完成時間最主要之誤差來源。故改良後之 max 運算，除增快演算法之運算速度，更能將 LPB 之影響充分彰顯出來。上述兩種演算法實際以 20 個大型專案網路(130~160 作業)進行實驗，參考 20000 次取樣蒙地卡羅模擬法(MCS)所得之結果，並與 PERT 及 Dodin(1985b)之結果做比較，結果顯示該兩種演算法除具快速的估算速度外(相較 MCS 快 7510 及 1274 倍)，專案網路完成時間期望值之估算誤差平均為 2.0%及 2.5%，在信心水準 $\alpha = 0.01$ 情況下，也均能通過 K-S 檢驗。上述之結果證明本論文所發展之 LCTA 及 MIA，對於隨機性專案網路完成時間之估算，確實為一有效率之估算工具。

隨機性專案網路最佳化資源分配問題(SRAP 及 SMRAP)為本論文第二部份探討的主題，分別針對單種類及多種類資源作分配，分配之結果決定專案總成本，因此亦為專案管理者最重視的問題之一。解 SRAP 及 SMRAP 問題須植基於有效率之專案網路完成時間估算，故 SRAP 及 SMRAP 可視為 LCTA 及 MIA 在專案網路之應用。資源分配須配合一決策工具，本論文採用磁場機制演算法(EM)及基因演算法(GA)分別執行 SRAP 及 SMRAP 問題。EM 及 GA 均會因決策變數增加而影響其向最佳解收斂之效率，因此本論文第二部份的主要貢獻是提出關鍵叢集演算法(CPCA)及叢集區域搜尋演算法(CLSA)，針對 SRAP 及 SMRAP 問題，有效率的計算出 EM 及 GA 合理之初始群組解(Initial population)。以 Terreso, et al. (2004b)中之 14 個網路實體為例，其實驗結果證明，CPCA 所產生之初始解已接近 SRAP 及 SMRAP 最佳解之區域範圍，故能有效縮短 EM 及 GA 尋找最佳解收斂的時間。尤以多種類資源的實驗結果，CPCA 所產生之初始解，比以隨機產生之初始解，執行 500 代 GA 演算所得之結果還來得好。

綜合上述，本論文主要貢獻是針對隨機專案網路完成時間估算，突破了相關文獻所遭遇之瓶頸，將完成時間機率分佈實際估算出來。另針對專案網路作業資源投入的優先順序，以關鍵叢集概念(CPC)取代傳統關鍵路徑(CP)概念，進而發展出 CPCA 演算法。若依循本論文所發展出來的工具及概念，針對隨機性專案網路後續其它議題之研究，應會有相當大的助益。

參考文獻

1. Abel, A.F., Robert, L.A., Julia, J.P., "Understanding simulation solutions to resource constrained Project scheduling problems with stochastic task durations", *Engineering Management Journal*, 10/4, 5-13, 1998.
2. Adlakha, V.G., "A Monte Carlo technique with quasi random points for the stochastic shortest path problem", *American Journal of Mathematic Management Society*, 7, 325-358, 1987.
3. Adlakha, V.G. & Kulkarni, V.G., "A classified bibliography of research on stochastic PERT networks: 1966-1987", *Information Systems and Operational Research*, 27/3, 272-296, 1989.
4. Agrawal, M.K., Elmaghraby, S.E., "On computing the distribution function of the sum of independent random variables", *Computers & Operations Research*, 28, 473-483, 2001.
5. Anklesaria, K.P., Drezner, Z., "A multivariate approach to estimating the completion times for PERT networks", *Journal of the Operational Research*, 37/8, 811-815, 1986.
6. Antonella, B., Lorenzo, A.P., "Optimal resource allocation with minimum activation levels and fixed costs", *European Journal of Operational Research*, 131, 536-549, 2001.
7. Basso, A., Peccati, L.A., "Optimal resource allocation with minimum activation levels and fixed costs", *European Journal of Operational Research*, 131, 536-549, 2001.
8. Bellman, R., "On a routing problem", *Quarterly of Applied Mathematics*, 16, 88-90, 1958.
9. Bertsekas, D.P., "A simple and fast label correcting algorithm for shortest paths", *Networks*, 23, 703-709, 1993.
10. Birbil, S.I., Fang, S.C., "An electromagnetism-like mechanism for global optimization", *Journal of Global Optimization*, 25, 263-282, 2003.
11. Birbil S.I., Fang S.C., Sheu R.L.. On the Convergence of a Population-Based Global Optimization Algorithm. *Journal of Global Optimization*, 30, 301~318, 2004.
12. Birge, J.R. and F. Louveaux, *Introduction to Stochastic Programming*, Springer Verlag, New York, 1997.
13. Boctor, F. F., "Some efficient multi-heuristic procedures for resource-constraint project scheduling", *European Journal of Operational Research*, 49, 3-13, 1990.
14. Boctor, F. F., "Resource-constrained project scheduling by simulated annealing", *International Journal of Production Research*, 34, 2335-2351, 1996
15. Bowers, J. A., "Criticality in resource constrained networks", *Operational Research Society*, Journal, 46, 80-91, 1995.
16. Bowers, J. A., "Identifying critical activities in stochastic resource constrained networks", Omega, *International Journal Management Science*, 24, 37-46, 1996.
17. Bowman, R.A., "Efficient estimation of arc criticalities in stochastic activity networks",

Management Science, 41/1, 1995.

18. Burt, J.M., Garman, M.B., "Conditional Monte Carlo: A simulation technique for stochastic network analysis", *Management Science*, 18/3, 1971.
19. Chatzoglou, P., Macaulay, L., "A review of existing models for project planning and estimation and the need for a new approach", *International Journal of Project Management*, 14, 173-183, 1996.
20. Cho, J.G., Yum, B.J., "Functional estimation of activity criticality indices and sensitivity analysis of expected project completion time", *Journal of the Operational Research Society*, 55, 850-859, 2004.
21. Chu, W.M., Yao, M.J., "A simulation-based genetic algorithm for the optimal resource allocation in probability activity networks", *The fifth Metaheuristics International Conference*, 2003.
22. Chu, W.M., Yao, M.J, "On Improving the Electromagnetism Algorithm for Solving the Resource Allocation Problem in Probabilistic Activity Networks'", *The 36th International Conference on Computers and Industrial Engineering*, 20~23 Jun, 2006.
23. Clark, C.E., "The greatest of a finite set of random variables", *Operations Research*, 9, 146-162, 1961.
24. Colby, A.H., Elmaghraby S.E., "On the complete reduction of directed acyclic graphs", *OR Report No. 197*, N.C. State Univ., Raleigh, NC., 1984.
25. Coppendate, J., "Manage risk in product and process development and avoid unpleasant surprises", *Engineering Management Journal*, February, 35-38, 1995.
26. Cox, A., "Simple normal approximation to the completion time distribution for a PERT network", *International Journal of Project Management*, 13/4, 265-270, 1995.
27. Dawson, R.J., Dawson, C.W., "Practical proposals for managing uncertainty and risk in project planning", *International Journal of Project Management*, 16, 299-310, 1998.
28. Demeulemeester, E.L., Herroelen, W.S. & Elmaghraby, S.E., "Optimal procedures for the discrete time/cost trade-off problem", *European Journal of Operation Research*, 88, 50-68, 1996.
29. Demeulemeester, E.L., Herroelen, W.S., "New benchmark results for the resource-constrained project scheduling problem", *Management Science*, 43/11, 1485-1492, 1997.
30. Devroye, L.P., "Inequalities for the completion times of stochastic PERT networks", *Mathematic of Operations Research*, 4/4, 441-447, 1979.
31. Dimitri, G.G., & Aharon, G., "Stochastic network project scheduling with non-consumable limited resources", *International Journal of Production Economics*, 48, 29-37, 1997.
32. Dimitri, G.G. & Aharon, G., "A heuristic for network project scheduling with random activity durations depending on the resource allocation", *International Journal of Production Economics*, 55, 149-162,1998.
33. Dimitri, G.G. & Aharon, G., Zohar, L., "Resource constrained scheduling simulation model for alternative stochastic network projects", *Mathematics and Computers in*

Simulation, 63, 105-117, 2003.

34. Dodin, B.M., "Determining the K most critical paths in PERT networks", *Operations Research*, 32/4, 859-877, 1984.
35. Dodin B.M., "Bounding the project completion time distribution in PERT networks", *Operations Research*, 33,/4, 862-881, 1985a
36. Dodin, B.M., "Approximating the distribution function in stochastic networks", *Computers and Operations Research*, 12/3, 251-264,1985b.
37. Dodin, B.M., Elmaghraby, S.E., "Approximating the criticality indices of the activities in PERT networks", *Management Science*, 31, 207-223, 1985c.
38. Dodin, B.M., Sirvanci, M., "Stochastic networks and the extreme value distribution", *Computers and Operations Research*, 17/4, 397-409, 1990.
39. Dorp J.R., Duffey M.R., "Statistical dependence in risk analysis for project networks using Monte Carlo methods", *International Journal of Production Economics*, 58, 17-29, 1999.
40. Elmaghraby, S.E., "On the expected duration of PERT type networks", *Management Science*, 13, 299-306, 1967.
41. Elmaghraby, S.E., *Activity networks: Project planning and control by network models*, Wiley, New York, 1977.
42. Elmaghraby, S.E., Pulat, P.S., "Optimal project compression with dual-dated events", *Naval Research Logistics Quarterly*, 26/2, 331-348, 1979.
43. Elmaghraby, S.E., Herroellen, W.S., "On the measurement of complexity in activity networks", *European Journal of Operational Research*, 5, 223-234, 1980.
44. Elmaghraby, S.E., "Resource allocation via dynamic programming in activity networks", *European Journal of Operational Research*, 88, 50-86, 1992.
45. Elmaghraby, S.E., "Activity nets: a guided tour through some recent developments", *European Journal of Operational Research*, 82/3, 383-408, 1995.
46. Elmaghraby, S.E., "On criticality and sensitivity in activity networks", *European Journal of Operational Research*, 127, 220-238, 2000.
47. Fatemi, G., Teimouri, E., "Path critical index and activity critical index in PERT networks", *European Journal of Operational Research*, 141/1, 147-152, 2002.
48. Fatemi, G., Rabbani, M., "A new structural mechanism for reducibility of stochastic PERT networks", *European Journal of Operational Research*, 145/2, 394-402, 2003.
49. Feng, C.W., Liu, L. and Burns, S.A., "Stochastic construction time-cost trade-off analysis", *Journal of Computing in Civil Engineering*, 14, 117-126, 2000.
50. Fisher, D.L., Saisi, D., Goldstein, W.M., "Stochastic PERT networks: OP diagrams critical paths and the project completion time", *Computers and Operations Research*, 12/5, 471-482, 1985.
51. Fishman, G.S., "Estimating network characteristics in stochastic activity networks", *Management Science*, 31/5, 579-593, 1985.

52. Fulkerson, D.R., "Expected critical path lengths in PERT networks", *Operations Research*, 10/6, 808-817, 1962.
53. Gallagher, C., "A note on PERT assumption", *Management Science*, 33, 1360-1362, 1987.
54. Gallo, G. and S. Pallottino, "Shortest path algorithms", *Annals of Operations Research*, 7, 3-79, 1988.
55. Glover, F., D. Klingman, and N. Philips, "A new polynomial bounded shortest path algorithm", *Operations Research*, 33, 65-73, 1986.
56. Godfrey, C.O., Michael, M., "Optimizing the multiple constrained resources product mix problem using genetic algorithms", *International Journal of Production Research*, 39/9, 1897-1910, 2001.
57. Hagstrom, J.N., "Computational complexity of PERT problems", *Networks*, 18, 139-147, 1988.
58. Hagstrom, J.N., "Computing the probability distribution of project duration in a PERT network", *Networks*, 20, 231-244, 1990.
59. Hardie, N., "The prediction and control of project duration: a recursive model", *International Journal of Project Management*, 19, 401-409, 2001.
60. Harley, H.O., Wortham, A.W., "A statistical theory for PERT critical path analysis", *Management Science*, 12/10, 469-481, 1966.
61. Hinddelang, T. J., Muth, J, F., "A dynamic programming algorithm for Decision CPM networks", *Operations Research*, 27, 225-241, 1979.
62. Ho, Y.C., Eyler, M.A., Chien, T.T., "A gradient technique for general buffer storage buffer storage design in a serial production line", *International Journal of Production Research*, 17/6, 557-580, 1979.
63. Holland, J. H., "Adaptation in natural and artificial systems", *Univ. of Michigan Press*, Ann Arbot, Mich., 1975.
64. Kolisch, R. and A. Sprecher, "PSPLIB-A project scheduling problem library", *European Journal of Operational Research*, 96, 205-216, 1996.
65. Kolisch, R., Drexl, A., "local search for nonpreemptive multi-mode resource-constrained project scheduling", *IIE transactions*, 29, 987-999, 1997.
66. Kamburowski, J., "An overview of the computational complexity of the PERT shortest route and Maximum flow problems in stochastic networks", *Stochastic in Combinatorial Optimization*, World Scientific Publishing, Singapore, 1987.
67. Kamburowski, J., "Upper bound on the expected completion time of PERT networks", *European Journal of Operational Research*, 21/2, 206-212, 1985a.
68. Kamburowski, J., "Normally distributed activity durations in PERT networks", *European Journal of Operational Research*, 36/11, 1051-1057, 1985b.
69. Kleindorfer, G.B., "Bounding distributions for stochastic logic", *Operations Research*,

19/7, 1586-1601, 1971.

70. Klingel, A.R., Jr., "Bias in PERT project completion time calculations for a real network", *Management Science*, 13/4, 476-489, 1966.
71. Kotiah, T.C.T., Wallace, N.D., "Another look at the PERT assumptions", *Management Science*, 20/1, 44-49, 1973.
72. Kulkarni, V.G. & Adlakha V.G., "Markov and Markov-Regenerative PERT nwtworks", *Operations Research*, 34/5, 769-781, 1986.
73. Leu, S.S., Yang, C.H., "GA-based multicriteria optimal model for construction scheduling", *Journal of Construction Engineering and Management*, 125, 420-427, 1999.
74. Luis, C, Antonio, G., "Distribution network optimization: Finding the most economic solution by using genetic algorithms", *European Journal of Operational Research*, 108, 527-537, 1998.
75. Martin, J.J., "Distribution of the time through a directed acyclic network", *Operational Research,* 13, 46-66, 1965.
76. Malcom, D.G., Roseboom, J.H., Clark C.E. & Fazar W., "Application of technique for research and development program evaluation (PERT)", *Operations Research*, 7/5, 646-669, 1959.
77. MacCrimmon, K.R., Ryacec, C.A., "An analytical study of the PERT assumptions", *Operations Research*, 12, 16-37, 1964.
78. McBride, W.J., McCelland, C.W., "PERT and the beta distribution", *IEEE Transactions on Engineering Management*, 14/4, 166-169, 1967.
79. Mehrotra, K., Chai, J. & Pillutal, S., "A study of approximating the moments of job completion time in PERT networks", *Journal of Operations Management*, 14/3, 277-289, 1996.
80. Malcolm, D.G., J.H. Roseboom, C.E. Clark and W. Fazar, "Application of a technique for research and development program evaluation", *Operations Research*, **7**, 646-669, 1959.
81. Michel, C., Francesco, L.P., Salvatore, T., "A hierarchical approach for bounding the completion time distribution of stochastic task graphs", *Performance Evaluation*, 41,1-22, 2000.
82. Molcolm, D. G., Roseboom, J.H., Clark, C.E., "Application of a technique for research and development program evaluation", *Operations Research*, 7, 646-669, 1959.
83. Pape, U., "Implementation and efficiency of Moore-algorithms for the shortest route problem", *Mathematical Programming*, 7, 212-222, 1974.
84. Premachandra, I.M., "An approximation of the activity duration distribution in PERT", *International Journal of Project Management*, 18, 215-222, 2000.
85. Pontrandolfo, P., "Project duration in stochastic networks by the PERT-path technique", *International Journal of Project Management*, 18, 215-222, 2000.

86. Ragsdale, C., “The current state of the network simulation in project management theory and practice”, *OMEGA*, 17/1, 21-25, 1989.
87. Ringer, L.J., “Numerical operators for statistical PERT critical path analysis”, *Management Science*, 16, 136-143, 1969.
88. Robillard, P., Trahan, M., “The completion time of PERT networks”, *Operations Research*. 25, 15-29, 1977.
89. Sasieni, W., “A note on PERT times”, Management Science, 32, 1652-1653.
90. Schmidt, C.W., Grossmann I.E., “The exact overall time distribution of a project with uncertain task durations”, *European Journal of Operational Research*, 126, 614-636, 2000.
91. Sculli, D., “The completion time of PERT networks”, *Journal of the Operational Research Society*, 25/1, 155-158, 1983.
92. Shogan, A.W., “Bounding distributions for a stochastic PERT network”, *Networks*, 7, 359-381, 1977.
93. Sigal, C.E., Pritsker, A.A.B., Solberg, J.J., “The use of cutsets in Monte Carlo analysis of stochastic networks”, *Mathematics and Computers in Simulation*, 21, 376-384, 1979.
94. Solis, F. J. and Wets, R. J-B., Minimization by random search techniques, *Mathematics of Operations Research*, 6, 19–30, 1981.
95. Splede, J.G., “Bounds for the distribution function of network variables”, *First Symposium of Operations Research*, 3, 113-123, 1977.
96. Sullivan, R.S., Hayya, J.C., “A comparison of the method of bounding distributions and Monte Carlo simulation for analyzing stochastic acyclic networks”, *Operations Research*, 28/3,614-617, 1980.
97. Tereso A.P., Araujo M.T., Elmaghraby S.E., “Adaptive resource allocation in multimodal activity networks”, *International Journal of Production Economics*, 92, 1~10, 2004a.
98. Tereso A.P., Araujo M.T., Elmaghraby S.E., “The optimal resource allocation in stochastic activity networks via the Electromagnetism approach”, Research Report, Universidade so Minho, Guimaraes, Portugal. Submitted for publication, 2004b.
99. Van, Slyke, R.M., “Monte Carlo methods and the PERT problem”, *Operations Research*, 11, 839-861, 1963.
100. Ward, S., Chapman, C., “Transforming project risk management into project uncertainty management”, *International Journal of Project Management*, 21/2, 97-105, 2003.
101. Wan, Y.W., “Resource allocation for a stochastic CPM-type network through perturbation analysis”, *European Journal of Operational Research*, 79, 239-248, 1994.
102. Wiest, J.D., Levy, F.K., “A Management Guide to PERT/CPM, Pretice-Hall”, Englewood Cliffs, NJ, 1977.
103. William, T.M., “Practical use of distributions in network analysis”, *Journal of the Operational Society*, 43/3, 265-270, 1962.

104. William, T.M., “Criticality in stochastic networks”, *Operational Research Society Journal*, 43, 353-357, 1992.

105. Yao M.J., Chu W.M., “A Label-Correcting Tracing Algorithm for the Approximation of the Probability Distribution Function of the Project Completion Time”, *Journal of Chinese Institute of Industrial Engineers*, Accepted, 2005.

106. Yao, M.J., Huang, J.X., ”Solving the Economic Lot Scheduling Problem with Deteriorating Items Using Genetic Algorithms,” *Journal of Food Engineering,* 2004 to appear.

附錄 A：PERT 使用限制之推導

Beta 分佈的通式可以表示為：

$$f(x)=\frac{1}{(b-a)^{k_1+k_2-1}\beta(k_1,k_2)}(x-a)^{k_1-1}(b-x)^{k_2-1};a\le x\le b \tag{A.1}$$

其中 $\beta(k_1,k_2)=\frac{\Gamma(k_1+k_2)}{\Gamma(k_1)\Gamma(k_2)}$，$[a,b]$為機率分配函數所佔的時間區域，$k_1,k_2$ 為機率分配函數的兩個參數，決定該機率分配曲線之形狀。為計算方便考量，本文將$[a,b]$轉換成[0,1]的範圍，故 Beta 機率分配函數可予以標準化為：

$$f(x)=\frac{1}{\beta(k_1,k_2)}(x)^{k_1-1}(1-x)^{k_2-1};\ \ 0\le x\le 1 \tag{A.2}$$

其中(2.1)、(2.2)式之平均值$E(x)$及變異數$Var(x)$可以求解表示為：

$$E(x)=\frac{k_1}{k_1+k_2} \tag{A.3}$$

$$Var(x)=\frac{k_1k_2}{(k_1+k_2)^2(k_1+k_2+1)} \tag{A.4}$$

且其偏態係數(skewness value)$SK(x)$可以求解為：

$$SK(x)=\frac{2(k_2-k_1)}{k_1+k_2+2}\left(\frac{k_1+k_2+1}{k_1k_2}\right)^{1/2} \tag{A.5}$$

若進行變數轉換，利用$y=a+(b-a)x$的關係式，我們亦可以得到：

$$E(y)=a+(b-a)E(x)=a+(b-a)\frac{k_1}{k_1+k_2} \tag{A.6}$$

$$Var(y)=(b-a)^2Var(x)=(b-a)^2\frac{k_1k_2}{(k_1+k_2)^2(k_1+k_2+1)} \tag{A.8}$$

Beta distribution 作業之平均時間及變異數亦可以簡式表示：

$$\bar{t}=\frac{a+4m+b}{6} \tag{A.9}$$

$$\sigma^2=\frac{(b-a)^2}{36} \tag{A.10}$$

令上述(A.9), (A.10)及(A.11), (A.12)式相等，即

$$\frac{a+4m+b}{6}=a+(b-a)\frac{k_1}{k_1+k_2} \tag{A.11}$$

$$\frac{(b-a)^2}{36}=(b-a)^2\frac{k_1k_2}{(k_1+k_2)^2(k_1+k_2+1)} \tag{A.12}$$

其中 m 值為最有可能的完成時間，其值落在專案完成時間機率分配函數之峰頂，因此令式(A.1)對 x 微分：

$$\begin{aligned}&\frac{df(x)}{dx}\\&=\frac{1}{(b-a)^{k_1+k_2-1}\beta(k_1,k_2)}[(k_1-1)(b-x)-(k_2-1)(x-a)](x-a)^{k_1-2}(b-x)^{k_2-2}\\&=0\end{aligned} \tag{A.13}$$

可以解得

$$m=\frac{a(k_2-1)+b(k_1-1)}{k_1+k_2-2} \tag{A.14}$$

將(A.14)式代入(A.11)式中便可得到兩個非線性函數，並從該兩函數中解得 k_1,k_2 兩參數，該兩參數僅有三種情況才能讓該兩非線性函數成立，即

$$k_1=3+\sqrt{2}\cong 4.4142,\ k_2=31\sqrt{2}\cong 1.5858,\ \Rightarrow SK(y)=\frac{-1}{\sqrt{2}} \tag{A.15}$$

Or $$k_2=3+\sqrt{2}\cong 4.4142,\ k_1=31\sqrt{2}\cong 1.5858,\ \Rightarrow SK(y)=\frac{1}{\sqrt{2}} \tag{A.16}$$

Or $$k_1=k_2=4,\quad \Rightarrow SK(y)=0, \tag{A.17}$$

附錄 B: 離散式 convlution 及 max 運算實例

若以圖 3.1 為例，以實際的作業離散數值進行 DA 演算法，藉以說明離散式之 convlution 及 max 運算是如何進行之，首先令圖中所有的相關作業之離散時間資料表示如下:

$$X_{12}=\begin{bmatrix}1, & 2, & 5\\ 1/6, & 1/3, & 1/2\end{bmatrix},\ X_{13}=\begin{bmatrix}2, & 6, & 8\\ 1/6, & 1/3, & 1/2\end{bmatrix},\ X_{14}=\begin{bmatrix}1, & 4, & 10\\ 1/6, & 1/6, & 2/3\end{bmatrix}$$
$$X_{23}=\begin{bmatrix}1, & 3, & 5\\ 1/3, & 1/3, & 1/3\end{bmatrix},\ X_{24}=\begin{bmatrix}1, & 6, & 12\\ 1/2, & 1/3, & 1/6\end{bmatrix},\ X_{34}=\begin{bmatrix}1, & 4, & 5\\ 1/3, & 1/6, & 1/2\end{bmatrix} \quad \text{(B.1)}$$

本文依 DA 演算法之規則，依序將各節點的輸出時間離散資料值算出，圖 3.1 節點 2 輸出時間γ_2即等於作業X_{12}時間，故表示如下:

$\gamma_2 = X_{12}$:

X	1	2	5
P(X)	1/6	1/3	1/2

節點 2 輸出時間γ_2加上作業X_{23}時間(執行 convolution 運算)，convolution 運算執行的方式，就是將兩離散變數之函數值x_i及y_j相加(參考 3.9 之定義)，得函數值$z_k = x_i + y_j$，其相對應之機率值為$p_i^x p_j^y$。因此γ_2加上作業X_{23}時間即等於節點 3 輸入路徑Y_2^3之時間，表示如下:

$Y_2^3 = \gamma_2 \oplus X_{23}$		γ_2		
		1 (1/6)	2 (1/3)	5 (1/2)
X_{23}	1 (1/3)	2 (1/18)	3 (2/18)	6 (3/18)
	3 (1/3)	4 (1/18)	5 (2/18)	8 (3/18)
	5 (1/3)	6 (1/18)	7 (2/18)	10 (3/18)

$Y_2^3 = \gamma_2 \oplus X_{23}$:

X	2	3	4	5	6	7	8	10
P(X)	1/18	2/18	1/18	2/18	4/18	3/18	3/18	3/18

由於γ_2及X_{23}均有 3 個離散值，因此它們之間所產生之 convolution 或

max 運算結果，最多的離散值數目為$(3\times3)=9$，因此Y_2^3的離散數值最多為 9 個。但上述之Y_2^3離散資料共有 8 個，即表示 convolution 運算，在這 9 次離散式資料相加的過程中有兩組所產生的z_k值是相同的，相同的z_k值是要合併為 1 個，且其產生之相對應之機率值$p_i^x p_j^y$亦必須要相加。以Y_2^3為例，γ_2之函數值 1 與X_{23}之函數值 5 相加等於 6，這數值與X_{23}之函數值 1 與γ_2之函數值 5 相加等於 6 相同，因此Y_2^3出現兩組函數值為 6 之離散資料，依上述規定Y_2^3必須要將這兩組合併為一，其中兩組所產生之機率值分別為$\frac{1}{6}\times\frac{1}{3}=\frac{1}{18}$及$\frac{1}{2}\times\frac{1}{3}=\frac{3}{18}$必須要相加，其結果為$\frac{1}{18}+\frac{3}{18}=\frac{4}{18}$，因此$Y_2^3$相對應 6 之函數值之機率值為$\frac{4}{18}$，且其離散值數目為 8 個。

接下來圖 3.1 執行節點 3 之輸出時間γ_3計算，該節點有兩條輸入路徑:$Y_1^3=X_{13}$及Y_2^3，γ_3即等於該兩路徑間，執行 max 運算之結果，表示如下：

$\gamma_3=\max\{X_{13},Y_2^3\}$	X_{13}	2	6	8
Y_2^3	$\Pr(X_{13})$ / $\Pr(Y_2^3)$	1/6	1/3	1/2
2	1/18	2 (1/108)	6 (2/108)	8 (3/108)
3	2/18	3 (2/108)	6 (4/108)	8 (6/108)
4	1/18	4 (1/108)	6 (2/108)	8 (3/108)
5	2/18	5 (2/108)	6 (4/108)	8 (6/108)
6	4/18	6 (4/108)	6 (8/108)	8 (12/108)
7	2/18	7 (2/108)	7 (4/108)	8 (6/108)
8	3/18	8 (3/108)	8 (6/108)	8 (9/108)
10	3/18	10 (3/108)	10 (6/108)	10 (9/108)

$\gamma_3 = \max\{X_{13}, Y_2^3\}$ ：

X	2	3	4	5	6	7	8	10
P(X)	1/108	2/108	1/108	2/108	24/108	6/108	54/108	18/108

max 運算的機制與 convolution 運算的機制一樣，唯不同的是 convolution 運算執行$z_k = x_i + y_j$，max 運算執行的是$z_k = \max(x_i, y_j)$，所產生的離散值z_k數目，最多為兩變數離散值數目之最大值。若產生相同的函數值亦要執行合併的工作，其相對應的率值要相加。例如上述之X_{13}有 3 個離散值，Y_2^3有 8 個離散值，因此γ_3最多會有$\max(8,3) = 8$個。上述之γ_3離散值數目亦為 8 個，因此X_{13}與Y_2^3執行 max 運算並無產生相同之離散函數值。

同上述所介紹之離散式 convolution 及 max 運算繼續執行專案網路估算，可得:

$Y_2^4 = \gamma_2 \oplus X_{24}$ ：

X	2	3	6	7	8	11	13	14	17
P(X)	3/36	6/36	9/36	2/36	4/36	6/36	1/36	2/36	3/36

$Y_3(4) = \gamma_3 \oplus X_{34}$ ：

X	3	4	5	6	7	8	9	10
P(X)	2/648	4/648	2/648	5/648	53/648	19/648	113/648	30/648
X	11	12	13	14	15			
P(X)	114/648	72/648	162/648	18/648	54/648			

$\gamma_4 = \max\{X_{14}, Y_2(4), Y_3(4)\}$ ：

X	3	4	5	6	7	8	9
P(X)	0.0001	0.0006	0.0003	0.0023	0.0105	0.0103	0.0388
X	10	11	12	13	14	15	17
P(X)	0.1667	0.2052	0.0926	0.2330	0.0748	0.0765	0.0833

圖 3.1 的最後估算結果為γ_4，共計 14 個函數值。從上數運算中不難發現，函數值數目會隨持續不斷的運算而擴大，函數值數目擴大到一定的程度會影響到整個離散式運算的效率。依此當專案網路的作業數量增加，運算的次數及機會亦會增多，函數值數目擴大所帶來的負面影響亦會增加。因此離散式運算仍須配合適當的重新取樣，調整函數值數目，且同時又不會影響估算的精度，相關的做法擬於 3.2.4 節說明。

附錄 C: LCTA 演算實例

圖 C.1 中擴張樹結構為圖 3.7 執行 LCTA 的詳細步驟圖示，其相對應之佇列資料表示如下文。除了第一步驟為初始值設定，其餘的步驟皆為 downward_procedure 或 upward_procedure 演算法相互交替執行之結果，共執行 9 個步驟後，可得到專案完成時間估算結果。每一步驟執行的結果均以 *k*=last_node, *j*=current_node, 佇列(stack)及佇列指標(stack_pointor)執行後的狀態值來表示。圖 C.1 中著色的節點 *j* 表示 γ_j 已被算出，粗框的節點表示演算法執行的現在位置，虛線表示節點間之接序關係，粗虛線表示演算法執行的路徑。若是連接 *i* 及 *j* 兩節點轉為實粗線，則表示 Y_i^j 已被算出。

圖 C.1 執行步驟及結果

步驟 1：設定追蹤初始值：

結果狀態：*k*=4, *j*=4, stack=(4), stack_pointer=1, *Shared_flag*=0.

步驟 2：執行 Downward_Tracing(4)

計算：$\gamma_1 = 0$。

設定：$finish\_flag_1 = 1$，$Output_1 = \gamma_1$

結果狀態：*k*=4, *j*=1, stack=(), stack_pointer=0.

步驟 3：執行 Upward_Tracing(1,4)

計算：$Y_1^4 = \gamma_1 \oplus X_{14}$，$X_{14}$ 從 Path_Act(4) 取得。

設定：$\text{Path\_Time}(4) = \{Y_1^4, -, -\}$，$\text{Path\_Flag}(4) = (1,0,0)$

結果狀態：*k*=2, *j*=4, stack=(4), stack_pointer=1, *Shared_flag*=0.

步驟 4：Execute the Downward_Tracing(2)

結果狀態：*k*=1, *j*=2, stack=(4), stack_pointer=1, *Shared_flag*=1.

步驟 5：執行 Upward_Tracing(1,2)

計算：$Y_1^2 = \mu(\gamma_1) \oplus X_{12}$，$\gamma_2 = Y_1^2$，$Y_2^4 = \gamma_2 \oplus X_{24}$，$X_{12}$ 及 X_{24} 分別由 Path_Act(2) 及 Path_Act(4) 取得。

設定：$finish\_flag_2 = 1$，$Output_2 = \gamma_2$, $\text{Path\_Flag}(2) = (1)$,
$\text{Path\_Time}(4) = \{Y_1^4, Y_2^4, -\}$，$\text{Path\_Flag}(4) = (1,1,0)$。

結果狀態：k=4, j=3, stack=(4), stack_pointer=1, *Shared_flag*=0.

步驟 6：執行 Downward_Tracing(3)

結果狀態：k=1, j=3, stack=(4), stack_pointer=1, *Shared_flag*=1.

步驟 7：執行 Upward_Tracing(1,3)

計算：$Y_1^3 = \mu(\gamma_1) \oplus X_{13}$，$X_{13}$ 從 Path_Act(3) 取得。

設定：$\text{Path\_Time}(3) = \{Y_1^3, -\}$，$\text{Path\_Flag}(3) = (1,0)$

結果狀態：k=1, j=2, stack=(3,4), stack_pointer=2, *Shared_flag*=1

步驟 8：執行 Downward_Tracing(2)

結果狀態：k=2, j=3, stack=(4), stack_pointer=1, *Shared_flag*=1

步驟 9：執行 Upward_Tracing(2,3)

計算：$Y_2^3 = \mu(\gamma_2) \oplus X_{23}$，$\gamma_3 = \max(Y_1^3, Y_2^3)$，$X_{23}$ 及 Y_1^3 從 Path_Act(3) 及 Path_Time(3) 取得。

設定：$finish\_flag_3 = 1$，$Output_3 = \gamma_3$，$\text{Path\_Flag}(3) = (1,1)$。

結果狀態：k=3, j=4, stack=(4), stack_pointer=1, *Shared_flag*=0

步驟 10：執行 Upward_Tracing(3,4)

計算：$Y_3^4 = \gamma_3 \oplus X_{34}$，$\gamma_4 = \max(Y_1^4, Y_2^4, Y_3^4)$。

設定：$finish\_flag_4 = 1$，$Output_4 = \gamma_4$，$\text{Path\_Flag}(4) = (1,1,1)$，$\text{Path\_Time}(4) = \{Y_1^4, Y_2^4, Y_3^4\}$。

結果狀態：k=4, j=4, stack=(), stack_pointer=0, *Shared_flag*=0.

$Output_4 = \gamma_4$ 即為專案網路完成時間估算結果。

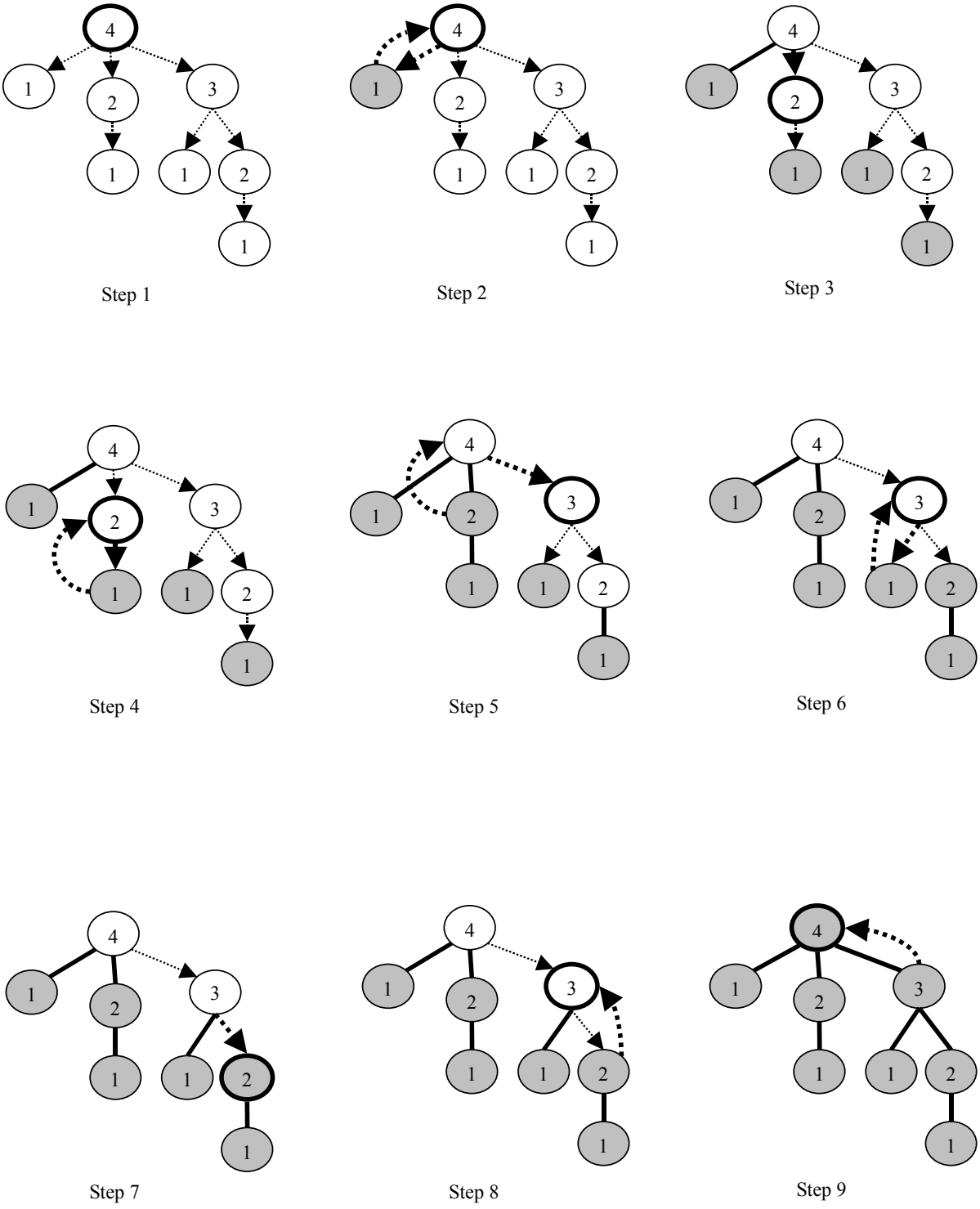


圖 C.1　圖 3.7 進行 LCTA 的執行步驟圖示

附錄 D: 路徑相依 LCTA 演算實例

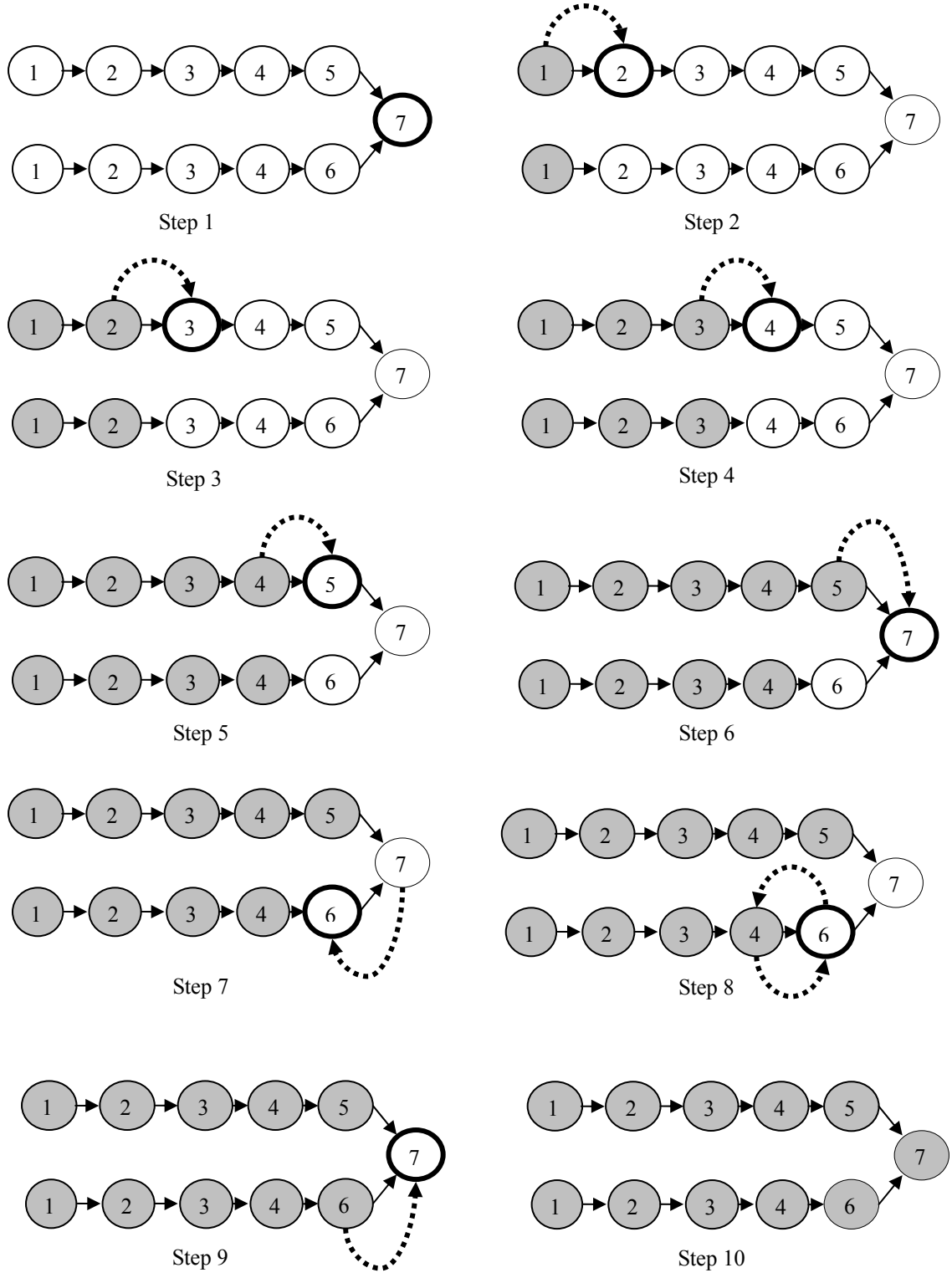


圖 D.1　圖 3.4 進行 LCTA 的執行步驟圖示

附錄 E: LCTA 執行 Kolish 範例實驗

表 E.1 LCTA 執行 120 次 Kolish 範例實驗結果
(估算專案網路完成時間 *pdf*)

項次	LCTA 不用 DRT			LCTA 使用 DRT			MCS		
	平均值	標準差	執行時間	平均值	標準差	執行時間	平均值	標準差	執行時間
1	47.99	9.92	0.85	47.94	9.87	0.18	48.78	10.20	117.55
2	54.89	13.18	0.76	54.83	13.11	0.10	55.63	15.83	113.95
3	51.87	11.04	1.17	51.78	10.96	0.11	51.18	12.56	107.67
4	63.08	9.20	0.73	63.02	9.16	0.10	62.12	12.92	113.24
5	39.60	8.87	0.47	39.54	8.85	0.09	39.37	9.41	115.22
6	50.11	9.66	0.26	50.06	9.63	0.10	48.26	10.70	129.86
7	65.65	8.30	0.47	65.61	8.29	0.09	63.75	12.32	119.80
8	60.06	17.50	1.07	59.98	17.37	0.10	57.75	16.01	114.27
9	55.37	10.21	0.98	55.31	10.12	0.09	55.08	12.07	127.80
10	46.35	6.14	0.54	46.30	6.11	0.08	45.31	7.23	131.28
11	48.44	11.36	1.08	48.38	11.22	0.10	46.14	11.45	123.88
12	53.70	8.97	0.40	53.67	8.94	0.09	52.52	8.79	119.08
13	53.37	9.07	0.67	53.31	9.04	0.09	53.64	11.17	121.70
14	54.39	10.97	0.77	54.33	10.82	0.10	53.02	12.67	109.72
15	61.41	12.49	0.69	61.36	12.46	0.10	59.46	14.73	132.86
16	52.77	8.93	0.51	52.72	8.83	0.10	52.28	11.91	127.86
17	56.42	10.69	1.44	56.36	10.60	0.11	55.65	14.37	109.99
18	62.17	11.90	0.96	62.14	11.89	0.10	60.73	16.68	115.89
19	62.30	12.87	1.72	62.19	12.68	0.10	60.29	12.41	120.38
20	57.24	10.14	0.29	57.19	10.03	0.07	54.13	14.50	129.81
21	83.17	13.00	1.09	83.15	12.95	0.11	82.46	19.92	127.59
22	49.57	9.89	0.41	49.53	9.87	0.09	49.24	10.29	123.47
23	66.78	11.94	0.26	66.76	11.92	0.08	63.96	16.24	133.33
24	101.26	18.86	1.14	101.16	18.69	0.10	99.86	21.35	143.70
25	57.44	9.23	0.79	57.38	9.07	0.09	57.35	9.72	131.58
26	65.14	10.54	0.62	65.08	10.52	0.09	65.10	14.00	122.28
27	55.48	12.00	0.81	55.44	11.96	0.11	54.01	13.45	128.03
28	68.82	14.82	0.53	68.65	14.11	0.09	65.61	16.87	117.23
29	70.76	11.67	0.45	70.69	11.63	0.09	66.86	15.53	114.08
30	67.14	14.03	0.54	67.10	14.01	0.11	64.80	15.08	120.64
31	59.67	9.05	0.77	59.60	8.85	0.09	58.99	11.87	128.61
32	71.06	13.29	0.44	70.96	13.03	0.08	66.25	15.86	115.50
33	54.38	10.42	0.87	54.35	10.41	0.11	53.64	10.81	113.55
34	70.05	10.58	0.99	69.95	10.34	0.10	68.88	15.85	161.59
35	68.04	12.67	0.62	68.02	12.57	0.11	66.36	16.57	124.72
36	57.80	13.28	1.18	57.75	13.23	0.13	55.01	13.03	112.23
37	62.30	9.38	0.69	62.27	9.37	0.11	63.30	11.44	119.36
38	63.10	11.90	0.63	63.09	11.87	0.09	63.36	13.77	159.92
39	51.30	11.94	0.75	51.23	11.90	0.12	51.56	12.83	118.55
40	57.47	11.44	1.13	57.43	11.43	0.12	56.35	12.43	120.80
41	55.76	8.14	0.50	55.73	8.14	0.15	55.88	12.13	133.02

項次	LCTA 不用 DRT			LCTA 使用 DRT			MCS		
	平均值	標準差	執行時間	平均值	標準差	執行時間	平均值	標準差	執行時間
42	71.28	11.24	0.91	71.17	10.97	0.10	69.98	15.62	139.59
43	68.24	10.43	0.54	68.19	10.23	0.08	68.20	17.36	136.94
44	50.54	9.41	0.63	50.51	9.40	0.11	53.32	13.83	144.64
45	49.45	10.10	0.48	49.38	10.00	0.10	49.93	10.89	134.74
46	71.52	8.56	0.46	71.48	8.47	0.12	70.45	15.46	138.55
47	57.87	7.64	0.37	57.85	7.61	0.10	59.57	10.95	143.16
48	64.60	13.56	0.52	64.50	13.32	0.09	65.93	15.22	133.92
49	52.85	9.89	0.35	52.82	9.88	0.09	53.65	11.62	144.80
50	76.42	15.39	0.86	76.35	15.21	0.11	77.00	18.65	130.91
51	57.41	9.73	0.43	57.37	9.71	0.10	58.20	13.68	133.05
52	65.21	8.77	0.50	65.15	8.72	0.09	64.98	12.53	155.05
53	61.12	9.91	0.38	61.07	9.84	0.10	60.56	13.90	133.02
54	77.08	12.22	0.53	77.00	11.99	0.11	78.32	16.83	132.94
55	61.17	11.81	0.67	61.11	11.60	0.10	60.03	12.88	133.67
56	62.23	11.45	0.58	62.20	11.39	0.10	64.03	13.67	131.94
57	55.44	9.41	0.30	55.41	9.39	0.10	57.48	13.29	136.42
58	57.06	7.76	0.65	57.03	7.70	0.09	59.71	10.21	142.28
59	53.10	10.41	0.29	53.07	10.39	0.09	53.91	14.44	138.81
60	51.71	8.71	0.48	51.68	8.67	0.10	52.15	10.62	141.09
61	51.42	11.14	0.85	51.35	11.01	0.10	52.07	13.06	138.61
62	62.69	10.86	0.43	62.67	10.84	0.09	63.90	14.69	137.89
63	91.25	13.00	0.91	91.15	12.79	0.10	89.38	19.14	135.49
64	50.59	10.88	0.44	50.53	10.74	0.09	51.65	13.55	129.23
65	57.86	9.78	0.77	57.80	9.71	0.09	58.76	13.18	135.03
66	56.29	11.14	0.34	56.24	11.09	0.09	55.47	13.90	142.70
67	62.16	9.84	0.34	62.15	9.84	0.09	63.35	14.20	135.67
68	65.02	12.91	0.41	64.99	12.87	0.09	63.76	13.44	141.02
69	47.62	6.47	0.27	47.60	6.45	0.09	48.34	8.31	137.69
70	55.72	9.64	0.56	55.65	9.39	0.10	56.46	12.44	133.78
71	68.97	13.45	1.24	68.92	13.28	0.09	68.88	16.44	124.97
72	80.55	10.95	0.54	80.49	10.82	0.10	78.56	16.32	138.99
73	60.50	10.77	0.61	60.47	10.75	0.11	61.14	13.47	128.05
74	60.49	11.35	0.73	60.44	11.25	0.10	59.94	13.20	132.59
75	75.50	12.16	1.32	75.37	11.73	0.09	70.66	17.44	156.27
76	63.95	9.83	0.94	63.92	9.81	0.11	66.10	14.22	157.00
77	52.73	10.59	0.45	52.70	10.58	0.10	55.25	13.79	128.00
78	54.89	8.05	0.68	54.87	8.00	0.09	58.66	9.35	139.38
79	55.20	11.71	0.80	55.17	11.69	0.11	53.57	13.46	131.16
80	48.07	8.29	0.43	48.04	8.21	0.11	48.18	10.65	136.09
81	78.36	10.12	0.47	78.35	10.07	0.14	76.02	15.56	139.09
82	60.73	8.18	0.48	60.70	8.16	0.10	64.87	12.93	137.80
83	57.56	12.48	0.57	57.51	12.38	0.10	58.59	14.62	138.77
84	86.70	14.66	0.54	86.63	14.50	0.11	84.20	19.96	145.56
85	54.35	8.19	0.36	54.30	8.13	0.09	58.51	11.85	149.73
86	62.97	8.55	0.57	62.94	8.48	0.10	63.64	12.16	147.78

項次	LCTA 不用 DRT			LCTA 使用 DRT			MCS		
	平均值	標準差	執行時間	平均值	標準差	執行時間	平均值	標準差	執行時間
87	71.19	9.51	0.30	71.17	9.44	0.09	67.50	18.28	137.22
88	59.15	9.82	0.76	59.10	9.77	0.11	60.21	13.38	141.05
89	66.25	10.68	0.35	66.22	10.68	0.11	69.23	14.41	145.45
90	61.71	7.42	0.30	61.66	7.33	0.08	60.01	9.63	144.48
91	72.45	7.15	0.15	72.40	7.08	0.08	73.39	14.32	154.47
92	52.62	8.75	0.25	52.60	8.74	0.11	55.86	12.11	146.89
93	78.23	10.52	0.22	78.15	10.19	0.09	77.55	16.93	154.25
94	75.95	7.35	0.63	75.91	7.35	0.11	79.36	15.31	161.41
95	71.08	9.15	0.56	71.05	9.05	0.11	69.71	15.68	135.98
96	62.69	10.69	0.35	62.66	10.69	0.11	62.54	14.89	137.31
97	66.27	5.77	0.20	66.27	5.76	0.09	63.54	12.05	148.14
98	62.92	9.20	0.16	62.86	9.21	0.08	61.23	11.01	149.78
99	72.98	12.94	0.23	72.94	12.85	0.10	68.95	17.64	153.08
100	57.65	7.92	0.23	57.63	7.92	0.09	59.47	10.59	146.59
101	65.93	10.49	0.47	65.90	10.47	0.11	68.36	16.16	145.41
102	65.79	11.07	0.49	65.75	11.04	0.10	65.79	16.50	139.44
87	71.19	9.51	0.30	71.17	9.44	0.09	67.50	18.28	137.22
88	59.15	9.82	0.76	59.10	9.77	0.11	60.21	13.38	141.05
89	66.25	10.68	0.35	66.22	10.68	0.11	69.23	14.41	145.45
90	61.71	7.42	0.30	61.66	7.33	0.08	60.01	9.63	144.48
91	72.45	7.15	0.15	72.40	7.08	0.08	73.39	14.32	154.47
92	52.62	8.75	0.25	52.60	8.74	0.11	55.86	12.11	146.89
93	78.23	10.52	0.22	78.15	10.19	0.09	77.55	16.93	154.25
94	75.95	7.35	0.63	75.91	7.35	0.11	79.36	15.31	161.41
95	71.08	9.15	0.56	71.05	9.05	0.11	69.71	15.68	135.98
103	68.37	9.68	0.35	68.33	9.61	0.10	66.17	13.37	149.24
104	57.26	13.04	0.30	57.23	13.00	0.09	59.78	14.77	128.81
105	69.00	13.64	0.19	68.98	13.63	0.08	67.30	16.06	148.41
106	68.13	11.28	0.35	68.09	11.12	0.08	68.04	16.14	139.99
107	67.27	11.11	0.30	67.22	10.97	0.08	70.69	15.31	159.38
108	71.06	6.91	0.25	71.01	6.88	0.09	68.42	12.99	138.52
109	72.20	13.79	0.38	72.15	13.70	0.10	72.72	17.46	140.88
110	68.11	8.28	0.46	68.08	8.27	0.10	67.45	13.98	139.08
111	70.35	9.54	0.21	70.34	9.50	0.10	72.77	15.36	141.52
112	55.40	11.61	0.24	55.33	11.51	0.10	55.95	13.40	131.78
114	72.98	11.68	0.45	72.87	11.29	0.11	68.71	17.86	141.44
115	73.99	12.74	0.30	73.95	12.71	0.08	74.48	14.59	140.92
116	57.04	8.20	0.18	57.00	8.12	0.09	56.86	11.19	145.42
117	65.18	8.21	0.41	65.15	8.17	0.09	67.50	9.87	147.05
118	75.13	14.28	0.40	75.07	14.25	0.10	74.27	18.10	141.36
119	68.23	8.48	0.37	68.20	8.48	0.09	70.96	12.86	142.61
120	62.78	5.35	0.28	62.77	5.34	0.07	65.54	11.65	146.72

附錄 F: LCTA 執行大型專網路範例實驗

表 F.1 MCS 估算大型專案網路完成時間之結果

實驗項目 / 專案項次	MCS		
	平均值	標準差	執行時間
1	108.42	13.04	1714
2	115.10	14.43	1815
3	122.17	16.10	3961
4	107.71	15.46	1573
5	106.80	15.06	2126
6	113.68	18.64	1650
7	110.69	16.06	1870
8	116.16	18.05	2461
9	100.30	14.34	1670
10	121.17	17.80	1930
11	109.22	18.57	1896
12	120.93	19.10	3271
13	121.48	19.90	2294
14	117.71	20.90	1812
15	114.54	16.29	2563
16	111.60	15.47	2139
17	129.35	25.09	1815
18	117.09	17.36	1887
19	132.15	24.45	1810
20	131.53	22.86	1817

表 F.2　LCTA 估算大型專案網路完成時間之結果-1

實驗項目 / 專案項次	LCTA 未使用 DRT					
	平均值	平均值誤差 $\frac{DA\text{-}MCS}{MCS}$	標準差	標準差誤差 $\frac{DA\text{-}MCS}{MCS}$	執行時間	時間比 $\frac{MCS}{DA}$
1	106.93	1.38%	10.07	22.78%	4.06	423
2	112.89	1.93%	11.13	22.87%	5.07	358
3	118.84	2.72%	12.36	23.22%	5.99	661
4	104.56	2.92%	12.84	16.91%	4.03	391
5	104.96	1.72%	9.84	34.63%	4.21	505
6	109.2	3.94%	14.06	24.59%	4.4	375
7	110.26	0.39%	11.09	30.99%	6.45	290
8	120.84	4.03%	16.57	8.20%	9.07	271
9	97.58	2.71%	12.49	12.88%	6.23	268
10	121.15	0.02%	14.45	18.84%	8.55	226
11	108.21	0.93%	15.35	17.35%	10.68	178
12	124.07	2.60%	16.11	15.63%	6.27	522
13	120.73	0.62%	13.28	33.26%	7.91	290
14	112.91	4.08%	16.13	22.81%	7	259
15	113.26	1.12%	13.13	19.43%	8.09	317
16	109.74	1.67%	13.54	12.47%	4.86	440
17	128.92	0.33%	20.56	18.05%	6.27	289
18	117.84	0.64%	12.62	27.28%	6.69	282
19	129.42	2.06%	21.19	13.34%	7.73	234
20	129.02	1.91%	23.03	0.76%	8.63	210
平均值	-	1.89%	-	19.81%	-	339

表 F.3　LCTA 估算大型專案網路完成時間之結果-2

實驗項目 / 專案項次	LCTA 使用 DRT					
	平均值	平均值誤差 $\frac{DA-MCS}{MCS}$	標準差	標準差誤差 $\frac{DA-MCS}{MCS}$	執行時間	時間比 $\frac{MCS}{DA}$
1	106.72	1.57%	9.75	25.27%	0.28	6120
2	112.68	2.11%	10.78	25.27%	0.31	5856
3	118.64	2.89%	11.85	26.37%	0.29	13611
4	104.32	3.14%	12.34	20.14%	0.25	6292
5	104.76	1.91%	9.49	36.98%	0.25	8503
6	109.01	4.11%	13.77	26.15%	0.27	6111
7	110.04	0.59%	10.22	36.40%	0.28	6680
8	120.53	3.76%	15.14	16.15%	0.28	8758
9	97.34	2.95%	11.94	16.73%	0.27	6183
10	120.85	0.26%	13.53	24.00%	0.29	6656
11	107.96	1.15%	14.13	23.94%	0.28	6773
12	123.82	2.39%	15.38	19.49%	0.31	10552
13	120.44	0.86%	12.63	36.54%	0.28	8193
14	112.56	4.38%	14.73	29.50%	0.3	6040
15	113.04	1.31%	12.27	24.72%	0.27	9456
16	109.56	1.83%	12.87	16.80%	0.25	8523
17	128.41	0.73%	18.25	27.25%	0.28	6460
18	117.62	0.46%	12.12	30.16%	0.28	6739
19	128.84	2.51%	18.66	23.68%	0.36	5026
20	128.37	2.40%	20.64	9.71%	0.3	6035
平均值	-	2.06%	-	24.76%	-	7428

表 F.4　DA 估算大型專案網路完成時間之結果-1

實驗項目 / 專案項次	DA 未使用 DRT					
	平均值	平均值誤差 $\frac{DA-MCS}{MCS}$	標準差	標準差誤差 $\frac{DA-MCS}{MCS}$	執行時間	時間比 $\frac{MCS}{DA}$
1	126.23	16.42%	15.39	17.98%	77.91	22
2	135.31	17.55%	16.63	15.30%	108.93	17
3	149.67	22.51%	21.64	34.40%	119.51	33
4	129.98	20.68%	19.62	26.97%	93.4	17
5	134.83	26.25%	21.79	44.71%	120.82	18
6	140.62	23.70%	23.24	24.65%	136.14	12
7	145.67	31.60%	25.69	59.89%	142.07	13
8	150.94	29.94%	29.86	65.42%	174.28	14
9	118.85	18.50%	17.88	24.67%	82.3	20
10	149.48	23.37%	25.3	42.10%	158.58	12
11	142.05	30.06%	28.57	53.83%	171.14	11
12	165.62	36.95%	26.69	39.76%	161.45	20
13	156.25	28.62%	22.37	12.40%	150.47	15
14	170.64	44.96%	49.16	135.2%	263.16	7
15	145.98	27.44%	23.39	43.57%	158.87	16
16	140.8	26.17%	18.55	19.87%	94.29	23
17	196.43	51.86%	51.62	105.8%	371.86	5
18	153.54	31.13%	23.5	35.41%	154.91	12
19	187.49	41.88%	50.91	108.2%	305.01	6
20	191.18	45.35%	51.93	127.2%	334.24	5
平均值	-	29.75%	-	51.86%	-	15

表 F.5　DA 估算大型專案網路完成時間之結果

實驗項目 / 專案項次	DA 使用 DRT					
	平均值	平均值誤差 $\frac{DA-MCS}{MCS}$	標準差	標準差誤差 $\frac{DA-MCS}{MCS}$	執行時間	時間比 $\frac{MCS}{DA}$
1	125.19	15.46%	13.1	0.49%	2.4	713
2	134.01	16.43%	13.76	4.59%	2.74	662
3	147.62	20.83%	17.43	8.31%	2.81	1408
4	128.54	19.34%	16.39	6.02%	2.36	666
5	132.77	24.32%	17.17	14.03%	2.73	778
6	139.19	22.44%	20.21	8.42%	2.63	626
7	144.56	30.60%	20.57	28.07%	2.71	689
8	158.82	36.72%	20.6	14.09%	2.97	827
9	118.92	18.56%	15.67	9.30%	2.53	659
10	148.02	22.16%	18.45	3.64%	2.75	701
11	141.49	29.55%	25.28	36.07%	2.88	658
12	164.51	36.03%	21.14	10.66%	2.63	1242
13	155.94	28.37%	19.7	1.05%	2.71	845
14	167.95	42.67%	43.41	107.7%	2.92	620
15	143.96	25.69%	18.51	13.63%	3.19	805
16	139.79	25.27%	16.56	7.00%	2.65	806
17	191.74	48.23%	41.79	66.58%	3.23	561
18	151.9	29.73%	19.66	13.29%	2.94	641
19	183.95	39.20%	42.64	74.39%	2.98	606
20	186.44	41.74%	42.93	87.80%	3.13	581
平均值	-	28.67%	-	25.76%	-	755

表 F.6　PERT 估算大型專案網路完成時間之結果

實驗項目 / 專案項次	PERT					
	平均值	平均值誤差 $\frac{DA\text{-}MCS}{MCS}$	標準差	標準差誤差 $\frac{DA\text{-}MCS}{MCS}$	執行時間	時間比 $\frac{MCS}{DA}$
1	83.03	23.42%	10.68	18.10%	0.09	19041
2	89.97	21.84%	10.81	25.06%	0.09	20170
3	95.02	22.22%	18.91	17.47%	0.23	17221
4	84	22.01%	17.47	13.03%	0.08	19661
5	82.03	23.19%	12.86	14.59%	0.12	17715
6	89.01	21.70%	22.64	21.44%	0.09	18334
7	89.99	18.70%	16.29	1.40%	0.08	23378
8	83.01	28.54%	27.35	51.50%	0.11	22372
9	76.03	24.20%	17.45	21.69%	0.1	16695
10	92.98	23.26%	16.27	8.61%	0.1	19303
11	86.04	21.22%	21.32	14.78%	0.1	18963
12	99.01	18.13%	31.07	62.67%	0.11	29736
13	95.02	21.78%	20.78	4.40%	0.13	17646
14	82.97	29.52%	24.51	17.29%	0.11	16474
15	87.03	24.02%	20	22.75%	0.11	23296
16	80.97	27.44%	26.81	73.28%	0.1	21392
17	94.97	26.58%	28.65	14.20%	0.17	10679
18	83	29.11%	22.99	32.45%	0.08	23588
19	94.04	28.84%	27.98	14.43%	0.09	20106
20	89.01	32.33%	26.82	17.34%	0.13	13974
平均值	-	24.4%	-	23.32%	-	19487

附錄 G:　MCS 取樣精確度驗證及 K-S 檢驗

G.1 蒙地卡羅模擬法(MCS)20000 次取樣精確度驗證

若使用 t 分配檢驗，並令信心水準$\alpha = 0.999$,執行蒙地卡羅模擬法 20000 次取樣精確度驗證，相關的公式如下所列：

$$\varepsilon\% = \frac{t_{0.999}\dfrac{S}{\sqrt{n}}}{\mu},\ n = 20000,\quad \mu = \sum_{i=1}^{n}\frac{x_i}{n},\quad S = \sum_{i=1}^{n}\frac{(x_i - \mu)^2}{n} \tag{G.1}$$

$\varepsilon\%$為蒙地卡羅模擬法取樣估算之誤差值，n 為取樣數量，S 為樣本之變異數值，μ則為樣本之平均值，$t_{0.999}$。針對第二組實驗 20 個專案網路範例，每個專案網路用 MCS 執行 20000(n=20000)次專案網路完成時間之估算取樣，得到$x_i,\ i = 1 \sim 20000$樣本值。計算出μ及S後在代入式(G.1)，求得$\varepsilon\%$值，詳如表 G.1 所列。

表 G.1　MCS 針對 20 組大型專案網路執行估算誤差之驗證

1	2	3	4	5	6	7	8	9	10
0.28%	0.29%	0.31%	0.33%	0.33%	0.38%	0.34%	0.36%	0.38%	0.34%
11	12	13	14	15	16	17	18	19	20
0.40%	0.37%	0.38%	0.33%	0.32%	0.34%	0.43%	0.41%	0.45%	0.40%

Average: 0.36%

表 G.1 的平均估算誤差為 0.36%，若專案網路完成時間真時平均值為 120，則 20000 次 MCS 模擬的數值會介於120 ± 0.43範圍內，與真實值的差距非常小.

G.2 對 20 組大型專案網路執行 K-S 檢驗

K-S 檢定法與檢驗的樣本數息息相關，樣本數愈大，檢驗的標準會愈高，受檢對象愈不容易通過檢驗。實驗以信心水準$\alpha = 0.01$，採三種樣本數(30, 40 及 50)來執行 K-S 檢定。K-S 檢定的公式如(G.2)式所列:

$$D(N) = \max\{x_i \mid x_i = \left|F_{LCTA}(x_i) - F_{MCS}(x_i)\right|,\ i = 1,...,50\} \tag{G.2}$$

其中 N 為 K-S 檢驗的樣本點數，$F_{LCTA}(x_i)$及$F_{MCS}(x_i)$分別為 LCTA 及 MCS

累積機率分佈函數(Cumulative distribution function, *cdf*)，x_i 為其離散化之取樣數值。因有樣本值數量的規定，LCTA 及 MCS 可以重複取樣技術(DRT)來調整配合參與檢驗的樣本數量 N，K-S 檢驗值 $D(N)$ 則是取 $F_{LCTA}(x_i)$ 及 $F_{MCS}(x_i)$ 相對於 x_i 之 *cdf* 值差，再取絕對值。求出來的 $D(N)$ 再查表與關鍵檢驗值 $d_{0.01}(N)$ 做比較，並執行下列假設檢定：

H_0 :LCTA 估算專案網路完成時間之 *pdf* 與 MCS 相同。

該假設成立的判斷式為：

$$P(D(N) > d_{0.01}(N)) = 0.01 \tag{G.3}$$

其中 $d_\alpha(N)$ 為進行 K-S 檢驗重要的參考值，本文將其整理如表 G.2 所列：

表 G.2　關鍵檢驗值 $d_\alpha(N)$

樣本數	N=30	N=40	N=50
$\alpha = 0.01$	0.29	0.25	0.23
$\alpha = 0.05$	0.24	0.21	0.19
$\alpha = 0.1$	0.22	0.19	0.17

本論文除針對 LCTA 執行 K-S 檢驗(DRT vs. Non-DRT)，另也針對 DA 及 PERT，相關的檢驗結果如表 G.3 所列。將表 G.3 數值代入式(G.3)檢驗，便能夠得知有哪些項目通過假設檢定 H_0，詳細的情況如表 G.4 所列。

從表 G.4 的結果可以得知 LCTA 在樣本 30 的情況下，LCTA_DRT 及 LCTA 均全數通過 K-S 檢驗；在樣本 40 的情況下，LCTA_DRT 及 LCTA 僅 1 及 2 個未通過檢驗；在樣本 50 的情況下，LCTA_DRT 及 LCTA 通過檢驗的數目逐見降低，有 3 及 4 個未通過檢驗。DA 及 PERT 則是全數均未通過 K-S 檢驗。

表 G.3　LCTA, DA 及 PERT 之 K-S 檢驗值 $D(N)$

比較項目	LCTA_DRT	LCTA	DA_DRT	DA	PERT
1	0.1731	0.1749	0.5926	0.607	0.6976
2	0.1335	0.1428	0.6358	0.6343	0.6654
3	0.1959	0.2063	0.6857	0.7015	0.5556
4	0.1634	0.183	0.5992	0.6308	0.5299
5	0.191	0.1974	0.7122	0.7166	0.6435
6	0.2266	0.2406	0.6073	0.6312	0.4812
7	0.2054	0.2388	0.7857	0.7876	0.4899
8	0.2796	0.3239	0.8426	0.8538	0.5498
9	0.2075	0.2288	0.5886	0.6222	0.5919
10	0.1916	0.201	0.6739	0.6715	0.5787
11	0.1689	0.206	0.7067	0.7145	0.4323
12	0.2486	0.2655	0.845	0.8581	0.3619
13	0.2499	0.2782	0.8103	0.8222	0.4561
14	0.1423	0.1709	0.7128	0.7171	0.6409
15	0.2	0.2104	0.8229	0.8309	0.4523
16	0.1887	0.2198	0.7882	0.7837	0.6128
17	0.1743	0.2169	0.7063	0.72	0.5353
18	0.2159	0.2471	0.7325	0.7186	0.581
19	0.1929	0.2122	0.7899	0.7926	0.5103
20	0.1437	0.1815	0.732	0.7493	0.6223

表 G.4　LCTA, DA 及 PERT 通過檢定假設 H_0 之結果

α=0.01	LCTA_DRT	LCTA	DA_DRT	DA	PERT
N=30	20/20	20/20	0/20	0/20	0/20
N=40	19/20	18/20	0/20	0/20	0/20
N=50	17/20	16/20	0/20	0/20	0/20

G.4 對 20 組大型專案網路執行均方根(RMS)檢驗

有別於 K-S 檢驗技術，本論文額外針對 LCTA, DA 及 PERT 演算法進行均方根(Root Mean Square test, RMS test)檢驗，該檢驗係輔助 K-S 檢驗，進一步驗證 LCTA 之估算專案網路完成時間之 *pdf* 結果與 MCS 區別。RMS 檢驗的表示式如下：

$$RMS = \sqrt{\sum_{i=1}^{N} (F_{LCTA}(x_i) - F_{MCS}(x_i))^2} \quad \textbf{(G.4)}$$

相關的檢驗結果如表 G.5 所列。

表 G.5　LCTA and DA 之 RMS 檢驗值

比較項目	LCTA_DRT	LCTA	DA_DRT	DA
1	0.5997	0.5962	2.4649	2.4766
2	0.593	0.5625	2.7211	2.7329
3	0.7906	0.7723	3.3251	3.3364
4	0.6722	0.6497	2.7811	2.801
5	0.7798	0.7823	3.3344	3.3534
6	0.8797	0.8632	3.0186	3.034
7	0.8638	0.857	4.0369	4.0549
8	0.9643	1.0127	5.0884	5.1009
9	0.6833	0.6659	2.5456	2.5564
10	0.6716	0.6604	3.3803	3.4032
11	0.6617	0.6572	3.7052	3.7166
12	0.9357	0.9441	5.1649	5.1793
13	0.6923	0.7132	4.5721	4.5844
14	0.6104	0.5793	3.4948	3.5075
15	0.5787	0.5906	4.3096	4.3215
16	0.6338	0.6444	4.3244	4.3419
17	0.8319	0.8168	4.6492	4.6669
18	1.0363	1.0281	4.5338	4.548
19	0.8737	0.8676	5.481	5.5012
20	0.7081	0.6904	4.9265	4.9467

附錄 H:　LCTA_MIA 執行大型專網路範例實驗

表 H.1　MCS 估算大型專案網路完成時間之結果

實驗項目 / 專案項次	MCS		
	平均值	標準差	執行時間
1	124.5	19.9	636.1
2	114.5	17.8	1476.8
3	103.2	14.3	3257.4
4	107.7	15.5	1572.9
5	121.1	17.1	2425.6
6	110.1	17.5	2020.7
7	113.2	15.8	1373.1
8	114.7	18.3	2375.4
9	110.8	15.8	3204.1
10	106.9	16.7	1479.4
11	109.5	14.6	1867.5
12	126.5	20.8	3042.8
13	122.0	17.5	1455.8
14	107.8	14.2	2473.9
15	120.5	17.3	3450.4
16	120.9	19.1	3271.0
17	121.7	18.1	1940.6
18	114.5	16.3	2562.6
19	117.1	17.4	1887.0
20	132.2	24.5	1809.5

表 H.2　LCTA_MIA 估算大型專案網路完成時間之結果-1

實驗項目 / 專案項次	LCTA_MIA 未使用 DRT					
	平均值	平均值誤差 $\frac{DA-MCS}{MCS}$	標準差	標準差誤差 $\frac{DA-MCS}{MCS}$	執行時間	時間比 $\frac{MCS}{DA}$
1	122.33	1.72%	19.43	2.51%	2.32	273.84
2	110.15	3.83%	19.52	9.54%	1.15	1281.97
3	104.00	0.76%	14.35	0.42%	1.75	1858.19
4	112.65	4.59%	16.22	4.92%	1.08	1453.69
5	120.47	0.54%	20.01	17.12%	1.81	1337.91
6	112.80	2.43%	20.01	14.21%	2.16	934.19
7	114.33	0.96%	18.35	15.88%	1.82	753.21
8	112.05	2.31%	17.52	4.36%	2.88	823.64
9	106.35	3.99%	16.58	5.13%	1.21	2643.68
10	108.45	1.44%	19.35	15.93%	1.65	895.51
11	110.13	0.58%	16.40	11.92%	2.05	909.63
12	124.57	1.55%	19.31	7.11%	2.08	1461.49
13	123.36	1.13%	17.18	1.92%	1.83	794.23
14	109.50	1.58%	11.83	16.84%	0.83	2976.97
15	125.32	4.03%	14.31	17.51%	2.08	1656.48
16	132.82	9.83%	18.59	2.68%	1.92	1700.97
17	119.04	2.15%	17.56	3.14%	1.38	1405.23
18	112.71	1.60%	14.62	10.24%	0.93	2752.53
19	115.06	1.73%	19.33	11.34%	1.35	1395.73
20	131.88	0.21%	27.44	12.21%	7.52	241
平均值		2.35%		9.25%		1377.48

表 H.3　LCTA 估算大型專案網路完成時間之結果-2

實驗項目 / 專案項次	LCTA_MIA 使用 DRT					
	平均值	平均值誤差 $\frac{DA-MCS}{MCS}$	標準差	標準差誤差 $\frac{DA-MCS}{MCS}$	執行時間	時間比 $\frac{MCS}{DA}$
1	122.31	1.74%	19.06	4.34%	0.84	754.60
2	110.19	3.80%	18.83	5.65%	0.73	2012.03
3	104.09	0.84%	14.26	0.24%	0.89	3660.01
4	112.77	4.70%	16.19	4.73%	0.66	2397.70
5	120.40	0.61%	19.65	14.99%	0.56	4316.06
6	113.21	2.80%	19.64	12.10%	1.05	1929.95
7	114.31	0.94%	18.11	14.35%	0.70	1953.19
8	112.20	2.18%	17.48	4.63%	0.99	2411.56
9	106.17	4.15%	16.57	5.04%	0.63	5126.62
10	108.54	1.53%	19.10	14.40%	0.72	2057.55
11	110.07	0.52%	16.27	11.04%	0.83	2252.68
12	125.14	1.10%	9.79	52.92%	0.83	3674.91
13	121.40	0.48%	15.65	10.66%	0.56	2585.84
14	109.87	1.93%	11.78	17.20%	0.55	4530.88
15	125.45	4.14%	14.29	17.61%	0.97	3564.50
16	132.49	9.56%	18.59	2.69%	0.73	4456.36
17	119.04	2.15%	17.36	4.21%	0.58	3357.48
18	114.98	0.39%	12.43	23.73%	0.61	4207.89
19	122.91	4.97%	20.99	20.94%	0.94	2011.76
20	132.28	0.10%	21.25	13.10%	0.78	2317
平均值		2.43%		12.73%		2978.92

表 H.4　DA 估算大型專案網路完成時間之結果-1

實驗項目 / 專案項次	DA 未使用 DRT					
	平均值	平均值誤差 $\frac{DA\text{-}MCS}{MCS}$	標準差	標準差誤差 $\frac{DA\text{-}MCS}{MCS}$	執行時間	時間比 $\frac{MCS}{DA}$
1	141.44	13.63%	25.36	27.27%	106.31	5.98
2	153.92	34.39%	29.40	64.98%	187.36	7.88
3	130.98	26.89%	20.85	45.88%	105.15	30.98
4	129.98	20.68%	19.62	26.97%	93.40	16.84
5	156.69	29.35%	27.63	61.70%	151.45	16.02
6	142.99	29.84%	24.09	37.45%	129.74	15.58
7	138.36	22.18%	22.13	39.71%	130.12	10.55
8	142.95	24.63%	26.53	44.81%	161.62	14.70
9	132.98	20.05%	20.40	29.37%	101.74	31.49
10	135.13	26.40%	24.03	43.96%	135.39	10.93
11	130.84	19.48%	19.14	30.69%	99.33	18.80
12	158.00	24.87%	25.09	20.69%	160.79	18.92
13	156.23	28.08%	30.06	71.66%	189.34	7.69
14	132.84	23.24%	19.46	36.81%	104.71	23.63
15	147.52	22.46%	22.17	27.80%	143.87	23.98
16	165.62	36.95%	26.69	39.76%	161.45	20.26
17	157.57	29.52%	27.46	51.45%	168.89	11.49
18	145.98	27.44%	23.39	43.57%	158.87	16.13
19	187.49	41.88%	50.91	108.2%	305.01	6
20	153.54	29.14%	23.50	18.14%	154.91	22.66
平均值		26.55%		43.54%		16.52

表 H.5　DA 估算大型專案網路完成時間之結果-2

實驗項目 / 專案項次	DA 使用 DRT					
	平均值	平均值誤差 $\frac{DA-MCS}{MCS}$	標準差	標準差誤差 $\frac{DA-MCS}{MCS}$	執行時間	時間比 $\frac{MCS}{DA}$
1	141.52	13.70%	22.34	12.12%	2.15	295
2	150.89	31.74%	22.45	25.95%	2.83	521
3	128.74	24.72%	16.22	13.49%	2.56	1270
4	128.54	19.34%	16.39	6.02%	2.36	666
5	153.96	27.10%	21.74	27.22%	2.82	859
6	140.98	28.02%	19.45	11.01%	2.65	761
7	136.22	20.29%	16.74	5.69%	2.63	521
8	141.85	23.67%	21.43	16.93%	2.72	872
9	132.31	19.45%	17.09	8.39%	2.52	1269
10	134.06	25.39%	17.96	7.57%	2.68	551
11	130.49	19.16%	15.57	6.28%	2.45	761
12	157.19	24.23%	20.75	0.21%	3.02	1006
13	155.85	27.76%	27.17	55.12%	3.04	478
14	132.73	23.14%	17.03	19.69%	2.86	864
15	147.09	22.10%	19.45	12.13%	3.17	1090
16	164.51	36.03%	21.14	10.66%	2.63	1242
17	156.70	28.81%	23.69	30.66%	2.90	668
18	143.96	25.69%	18.51	13.63%	3.19	805
19	183.95	39.20%	42.64	74.39%	2.98	606
20	151.90	29.73%	19.66	13.29%	2.94	641
平均值		25.46%		18.52%		787.44

表 H.6　PERT 估算大型專案網路完成時間之結果

實驗項目 / 專案項次	PERT					
	平均值	平均值誤差 $\frac{DA\text{-}MCS}{MCS}$	標準差	標準差誤差 $\frac{DA\text{-}MCS}{MCS}$	執行時間	時間比 $\frac{MCS}{DA}$
1	92.00	26.09%	20.06	0.66%	0.05	12723
2	93.01	18.80%	20.67	15.98%	0.08	18460
3	80.01	22.49%	13.53	5.33%	0.22	14806
4	84.00	22.01%	17.47	13.03%	0.08	19661
5	88.01	27.34%	15.18	11.16%	0.14	17326
6	79.96	27.39%	19.08	8.88%	0.11	18370
7	84.00	25.82%	18.20	14.92%	0.07	19616
8	84.02	26.75%	22.48	22.68%	0.11	21594
9	86.01	22.35%	12.78	18.96%	0.2	16021
10	79.99	25.18%	16.95	1.53%	0.07	21134
11	83.98	23.31%	15.06	2.80%	0.1	18675
12	97.01	23.33%	20.41	1.84%	0.17	17899
13	94.98	22.14%	13.66	22.00%	0.08	18198
14	84.02	22.06%	16.33	14.78%	0.14	17670
15	96.02	20.29%	16.08	7.29%	0.2	17252
16	99.01	18.13%	31.07	62.67%	0.11	29736
17	92.00	24.38%	20.75	14.46%	0.11	17642
18	87.03	24.02%	20.00	22.75%	0.11	23296
19	83.00	29.11%	22.99	32.45%	0.08	23588
20	94.04	28.84%	27.98	14.43%	0.09	20106
平均值		23.99%		15.43%		19188.6

附錄 I:　LCTA_MIA 之 K-S 檢驗

I.1 對 20 組大型專案網路執行 K-S 檢驗

K-S 檢定法與檢驗的樣本數息息相關，樣本數愈大，檢驗的標準會愈高，受檢對象愈不容易通過檢驗。實驗以信心水準 $\alpha = 0.01$，採三種樣本數(30, 40 及 50)來執行 K-S 檢定。K-S 檢定的公式如(G.2)式所列:

$$D(N) = \max\{x_i \mid x_i = \left|F_{LCTA}(x_i) - F_{MCS}(x_i)\right|,\ i = 1,...,50\} \tag{I.1}$$

其中 N 為 K-S 檢驗的樣本點數，$F_{LCTA}(x_i)$ 及 $F_{MCS}(x_i)$ 分別為 LCTA_MIA 及 MCS 累積機率分佈函數(Cumulative distribution function, *cdf*)，x_i 為其離散化之取樣數值。因有樣本值數量的規定，LCTA_MIA 及 MCS 可以重複取樣技術(DRT)來調整配合參與檢驗的樣本數量 N，K-S 檢驗值 $D(N)$ 則是取 $F_{LCTA}(x_i)$ 及 $F_{MCS}(x_i)$ 相對於 x_i 之 *cdf* 值差，再取絕對值。求出來的 $D(N)$ 再查表與關鍵檢驗值 $d_{0.01}(N)$ 做比較，並執行下列假設檢定：

H_0 :LCTA_MIA 估算專案網路完成時間之 *pdf* 與 MCS 相同。

該假設成立的判斷式為：

$$P(D(N) > d_{0.01}(N)) = 0.01 \tag{I.2}$$

其中 $d_\alpha(N)$ 為進行 K-S 檢驗重要的參考值，本文將其整理如表 I.1 所列：

表 I.1　關鍵檢驗值 $d_\alpha(N)$

樣本數	N=30	N=40	N=50
$\alpha = 0.01$	0.29	0.25	0.23
$\alpha = 0.05$	0.24	0.21	0.19
$\alpha = 0.1$	0.22	0.19	0.17

本論文除針對 LCTA_MIA 執行 K-S 檢驗(DRT vs. Non-DRT)，另也針對 DA 及 PERT，相關的檢驗結果如表 I.2 所列。將表 I.2 數值代入式(I.2)檢驗，便能夠得知有哪些項目通過假設檢定 H_0，詳細的情況如表 I.3 所列。

從表 I.3 的結果可以得知 LCTA_MIA 在樣本 30 的情況下，LCTA_MIA _DRT 及 LCTA_MIA 均全數通過 K-S 檢驗；在樣本 40 的情況下，LCTA_MIA

_DRT 及 LCTA_MIA 僅 1 及 2 個未通過檢驗；在樣本 50 的情況下，LCTA_MIA _DRT 及 LCTA_MIA 通過檢驗的數目逐見降低，有 3 及 4 個未通過檢驗。DA 及 PERT 則是全數均未通過 K-S 檢驗。

表 I.2　LCTA, DA 及 PERT 之 K-S 檢驗值 $D(N)$

比較項目	LCTA_DRT	LCTA	DA_DRT	DA	PERT
1	0.209	0.209	0.455	0.470	0.578
2	0.208	0.208	0.777	0.777	0.438
3	0.090	0.090	0.738	0.737	0.610
4	0.210	0.210	0.607	0.635	0.532
5	0.083	0.083	0.747	0.766	0.696
6	0.178	0.178	0.753	0.736	0.586
7	0.129	0.129	0.663	0.673	0.602
8	0.107	0.107	0.633	0.644	0.576
9	0.124	0.124	0.598	0.605	0.625
10	0.122	0.122	0.705	0.705	0.589
11	0.120	0.120	0.644	0.649	0.607
12	0.234	0.234	0.673	0.681	0.509
13	0.046	0.046	0.709	0.712	0.624
14	0.158	0.158	0.692	0.707	0.573
15	0.239	0.239	0.687	0.700	0.521
16	0.371	0.371	0.843	0.855	0.362
17	0.058	0.058	0.758	0.755	0.538
18	0.106	0.106	0.707	0.712	0.623
19	0.196	0.196	0.780	0.780	0.607
20	0.121	0.121	0.721	0.721	0.545

表 I.3　LCTA_MIA, DA 及 PERT 通過檢定假設 H_0 之結果

α=0.01	LCTA_MIA_DRT	LCTA_MIA	DA_DRT	DA	PERT
N=30	20/20	20/20	0/20	0/20	0/20
N=40	19/20	18/20	0/20	0/20	0/20
N=50	17/20	16/20	0/20	0/20	0/20

I.2 對 20 組大型專案網路執行均方根(RMS)檢驗

有別於 K-S 檢驗技術，本論文額外針對 LCTA_MIA, DA 及 PERT 演算法進行均方根(Root Mean Square test, RMS test)檢驗，該檢驗係輔助 K-S 檢驗，進一步驗證 LCTA_MIA 之估算專案網路完成時間之 *pdf* 結果與 MCS 區別。RMS 檢驗的表示式如下：

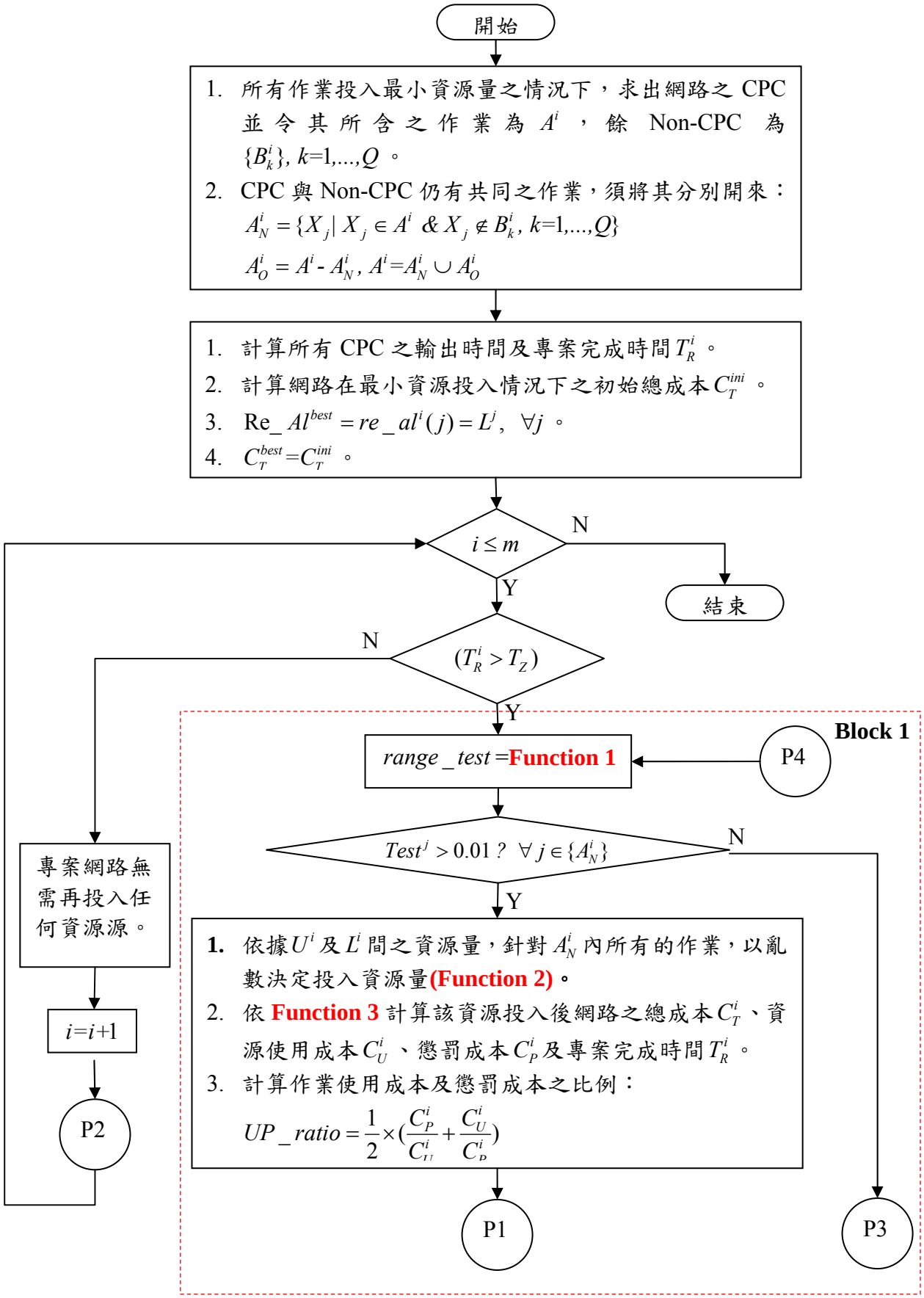
$$RMS = \sqrt{\sum_{i=1}^{N} (F_{LCTA}(x_i) - F_{MCS}(x_i))^2} \quad \textbf{(I.3)}$$

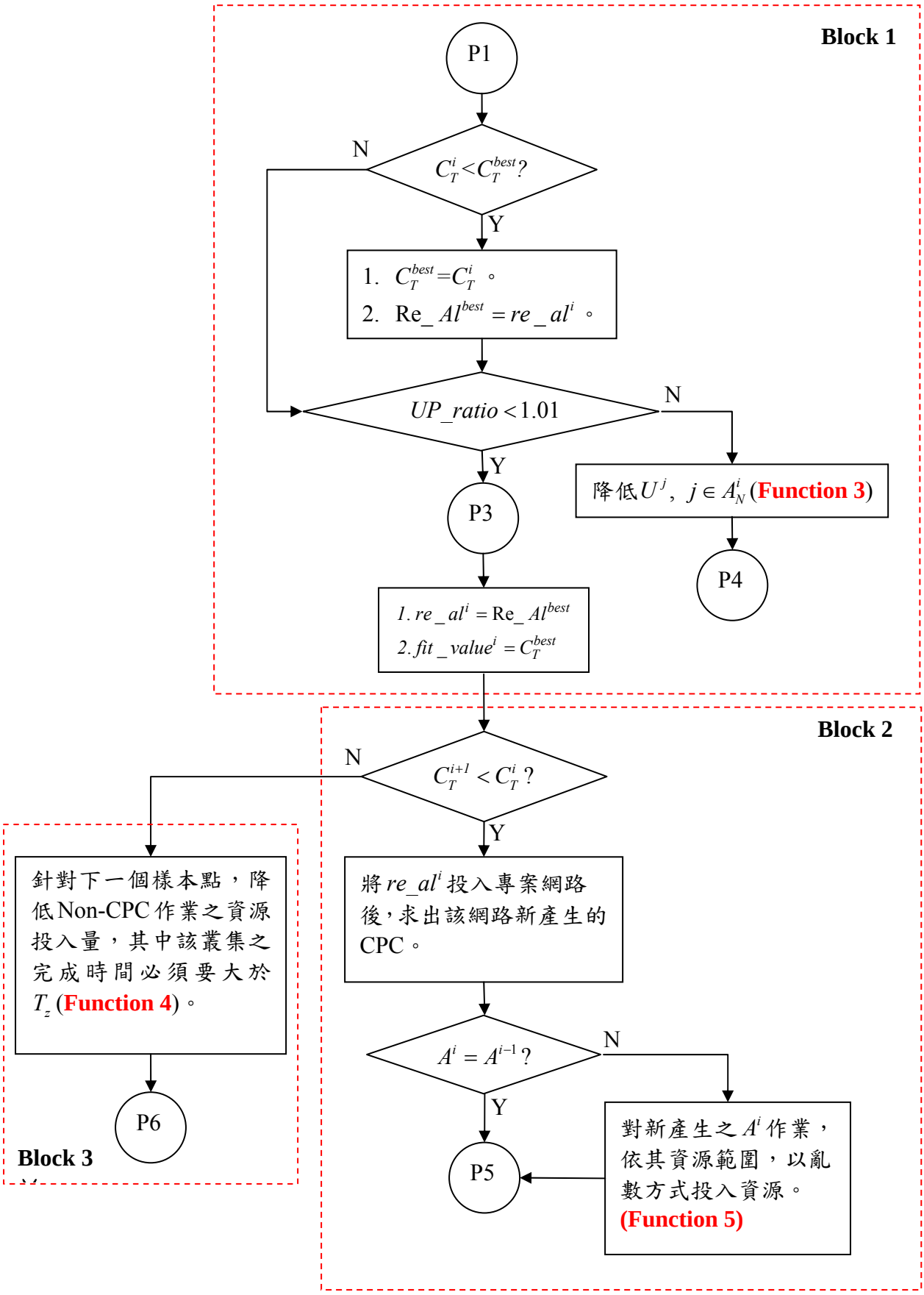
相關的檢驗結果如表 I.4 所列。

表 I.4　LCTA_MIA and DA 之 RMS 檢驗值

比較項目	LCTA_MIA _DRT	LCTA_MIA	DA_DRT	DA
1	0.578	0.578	2.166	2.173
2	0.540	0.540	4.169	4.169
3	0.264	0.264	3.405	3.440
4	0.836	0.836	2.802	2.858
5	0.350	0.350	3.971	3.959
6	0.650	0.650	3.839	3.858
7	0.475	0.475	3.216	3.194
8	0.353	0.353	3.203	3.223
9	0.557	0.557	2.795	2.792
10	0.485	0.485	3.537	3.534
11	0.393	0.393	2.983	2.961
12	1.063	1.063	3.624	3.657
13	0.227	0.227	3.681	3.665
14	0.534	0.534	3.255	3.278
15	0.996	0.996	3.393	3.418
16	1.890	1.890	5.139	5.155
17	0.206	0.206	4.115	4.159
18	0.399	0.399	3.496	3.515
19	1.025	1.025	4.353	4.353
20	0.340	0.340	4.696	4.696

附錄 J：CPCA 演算法的流程圖




Block 1
P1
N
$C_T^i < C_T^{best}$?
Y
1. $C_T^{best} = C_T^i$ 。
2. Re_$Al^{best} = re_al^i$ 。
$UP\_ratio < 1.01$
N
Y
降低U^j, $j \in A_N^i$ (Function 3)
P3
P4
1. re_al^i = Re_Al^{best}
2. $fit\_value^i = C_T^{best}$
Block 2
N
$C_T^{i+1} < C_T^i$?
Y
針對下一個樣本點，降低Non-CPC作業之資源投入量，其中該叢集之完成時間必須要大於T_z (Function 4)。
將re_al^i投入專案網路後，求出該網路新產生的CPC。
$A^i = A^{i-1}$?
N
Y
P6
Block 3
P5
對新產生之A^i作業，依其資源範圍，以亂數方式投入資源。
(Function 5)

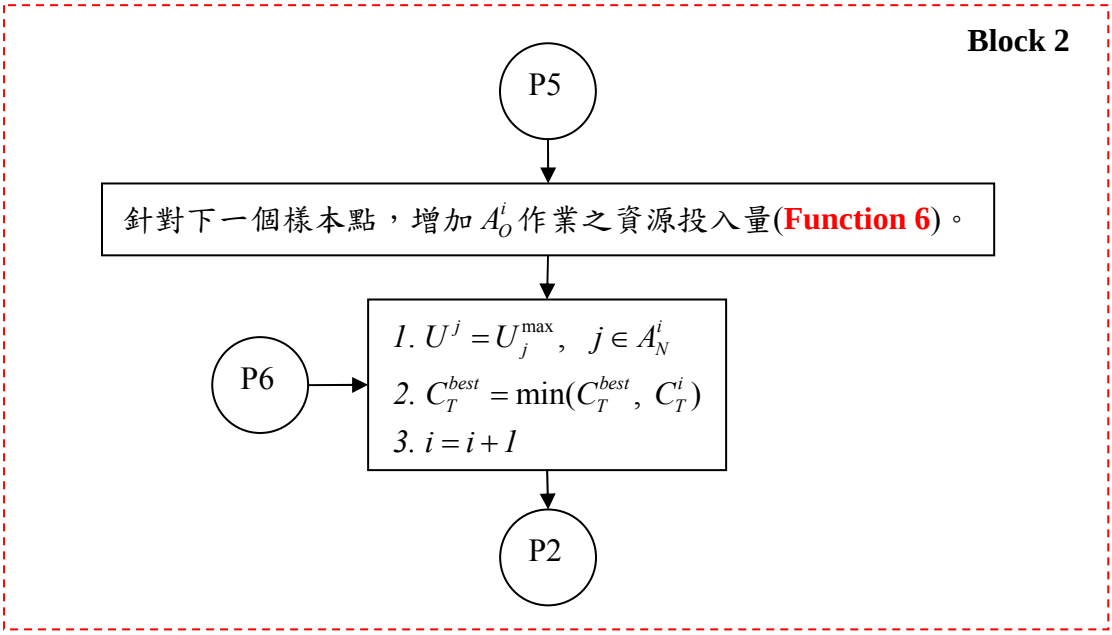


圖 J.1　CPCA 執行流程圖

Function 1：

$$Test^i = \frac{(U^j - L^j)}{L^j}, \quad j \in A_N^i$$

Function 2：

$$\lambda = U(0,1);$$

$$re\_al^i = L^j + \lambda * (U^j - L^j), \ i = 1,...,m, \ j \in A_N^i; \quad \text{(A)}$$

Function 3：

$$U^j = L^j + D\_rate \times (U^j - L^j), \ j \in A_N^i$$

Function 4：

$$\beta = U(0,1)$$

$$M^j = M^j - I\_rate \times (U^j - L^j)$$

$$re\_al^{i+1} = L^j + \beta \times (M^j - L^j), \ i = 1,...,m$$

$$j \in A_N^{i+1} \in \{B_h^i | \ T_h^i = \max(T_k^i), \ h, \ k = 1,...,Q\}$$

Function 5：

$$\beta = U(0,1)$$
$$M^j = M^j + I\_rate \times (U^j - L^j)$$
$$re\_al^{i+1} = L^j + \beta \times (M^j - L^j),\ \ i = 1,...,m$$
$$j \in A_N^{i+1} \in \{B_h^i |\ T_h^i = \max(T_k^i),\ h,\ k = 1,...,Q\}$$

Function 6：

$$\beta = U(0,1);$$
$$M^j = M^j + I\_rate \times (U^j - L^j)$$
$$re\_al^{i+1} = L^j + \beta \times (M^j - L^j),\ \ i = 1,...,m-1,\ \ \ j \in \{A_O^i\}$$

附錄 K：LCTA 演算法的流程圖

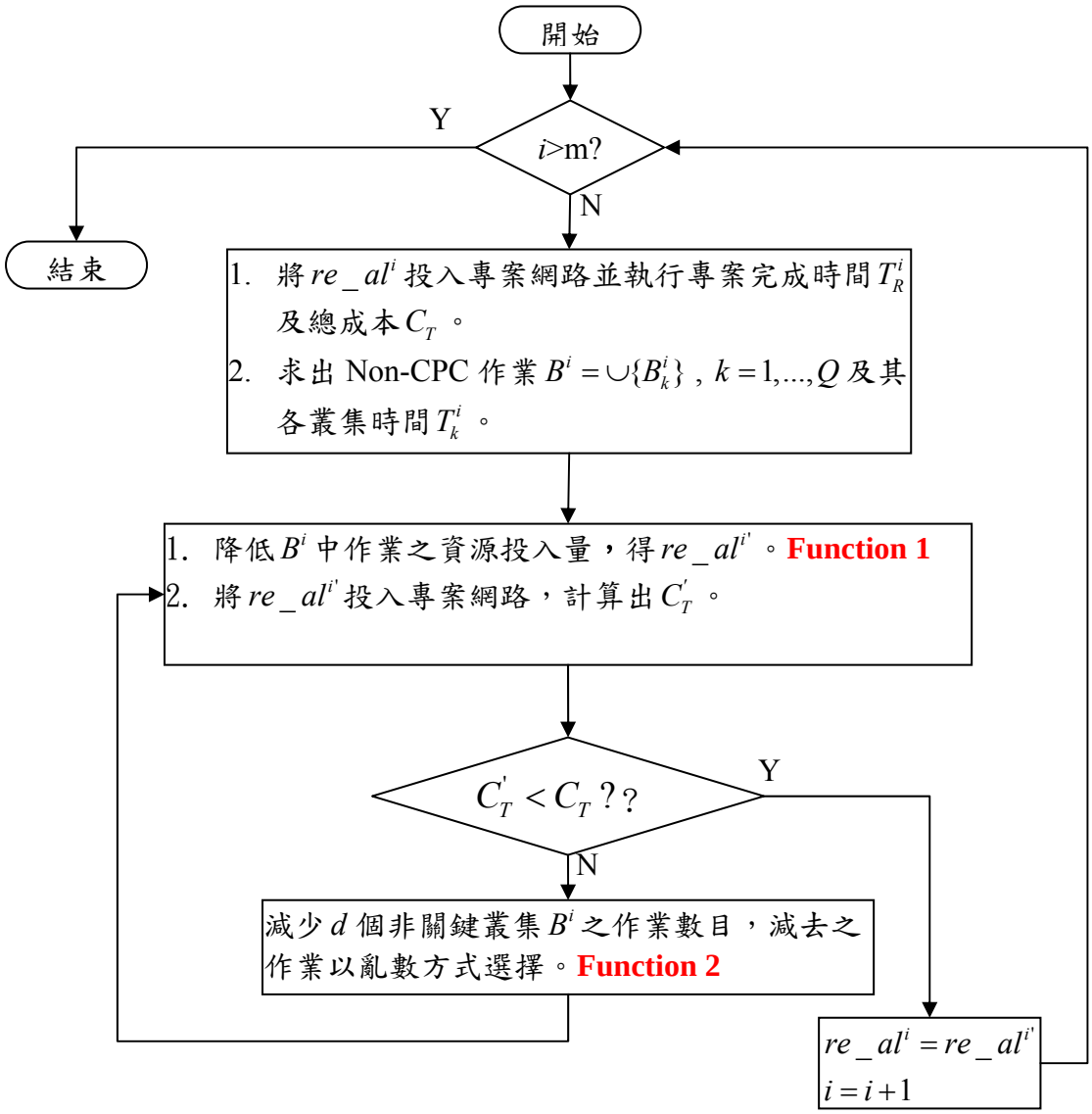


圖 K.1　CLSA 演算法流程圖

$lsiter = 10$

$Ld\_rate = 0.1$

Function 1：

$\lambda = U(0,1)$;

$re\_al^{i'} = re\_al^{i} - \lambda \times Ld\_rate \times (U^j - L^j),\ i = 1,...,m,\ j \in B^i$

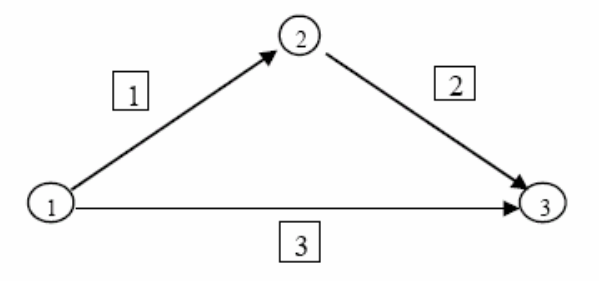
Function 2：

$$d = \frac{number(B^i)}{lsiter}$$

$$D^i = \{a_z \mid a_z \Leftarrow B^{i'},\ z = 1,...,d\}$$

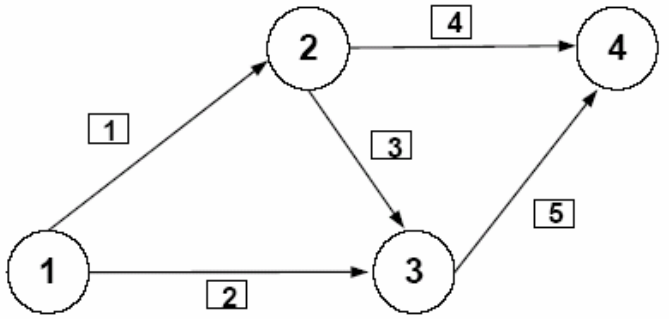
$$B^{i'} = B^{i'} - D^i$$

附錄 L：Terreso & Elmaghraby(2004) 14 個專案網路實例



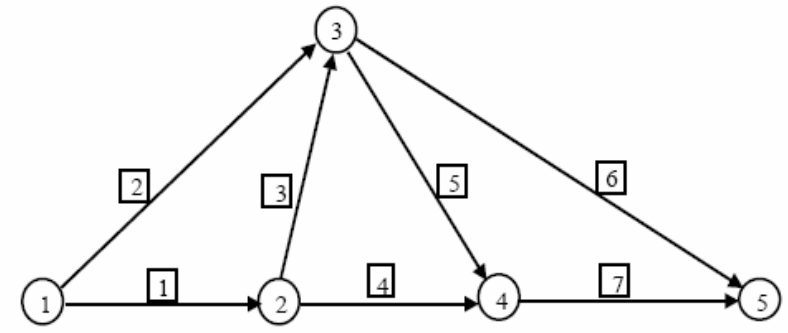
作業編號	1	2	3
平均時間	5.0	10.0	14.3

專案網路 1 $(T_Z = 16,\ P_C = 2,\ U_C = 1)$



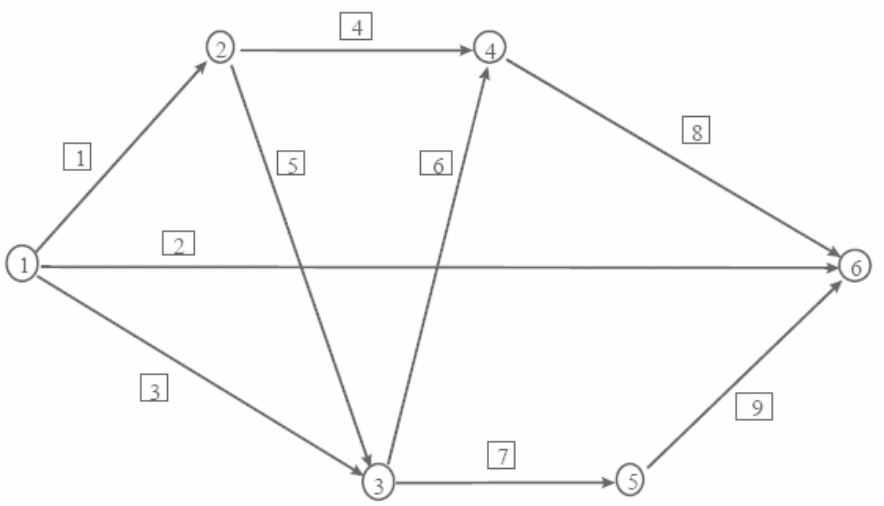
作業編號	1	2	3	4	5
平均時間	50.0	33.3	25.0	41.7	40.0

專案網路 2 $(T_Z = 120,\ P_C = 8,\ U_C = 1)$



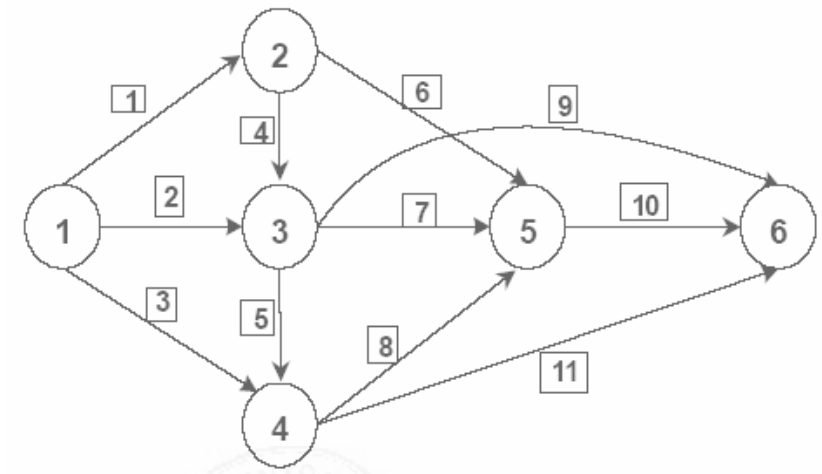
作業編號	1	2	3	4	5	6	7
平均時間	12.5	16.7	11.1	20.0	14.3	33.3	25.0

專案網路 3 $(T_Z = 66,\ P_C = 5,\ U_C = 1)$



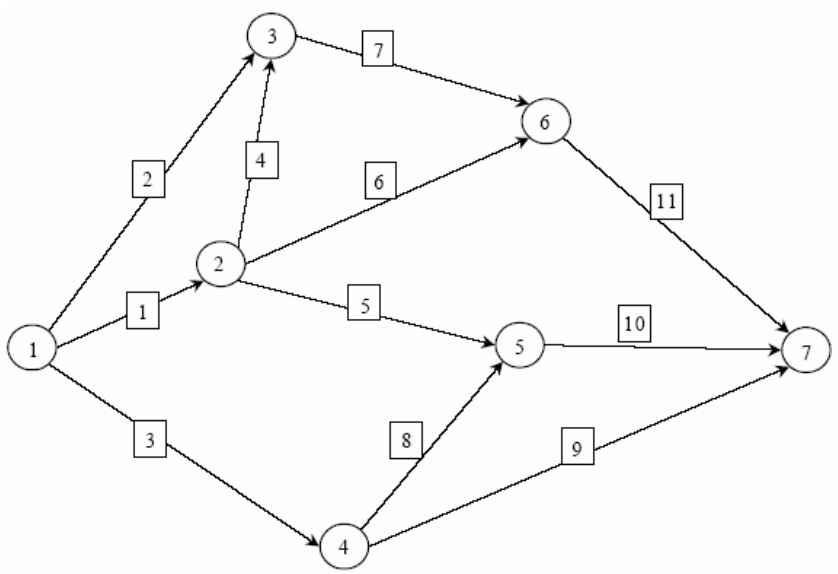
作業編號	1	2	3	4	5	6	7	8	9
平均時間	25.0	100.0	14.3	28.6	20.0	16.7	22.2	16.7	25.6

專案網路 4 $(T_Z = 105,\ P_C = 4,\ U_C = 1)$



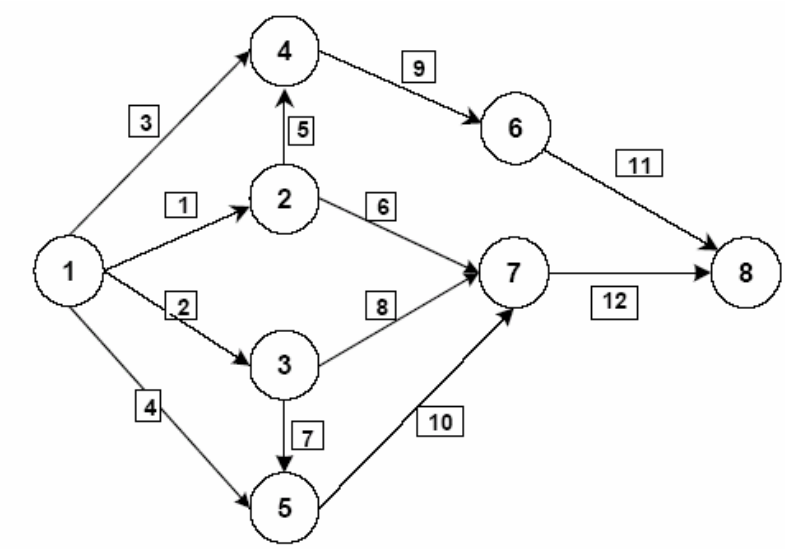
作業編號	1	2	3	4	5	6	7	8	9	10
平均時間	10.0	11.1	2.5	5.0	3.3	12.5	2.5	5.0	10.0	3.3
作業編號	11									
平均時間	3.3									

專案網路 5 $(T_Z = 28,\ P_C = 8,\ U_C = 1)$



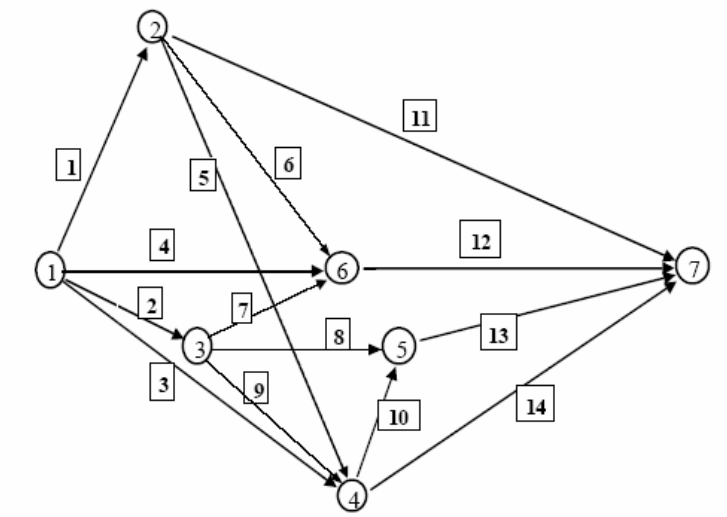
作業編號	1	2	3	4	5	6	7	8	9	10
平均時間	10.0	8.3	20.0	12.5	5.0	25.0	33.3	25.0	41.7	6.7
作業編號	11									
平均時間	6.3									

專案網路 6 ($T_Z = 65,\ P_C = 5,\ U_C = 1$)



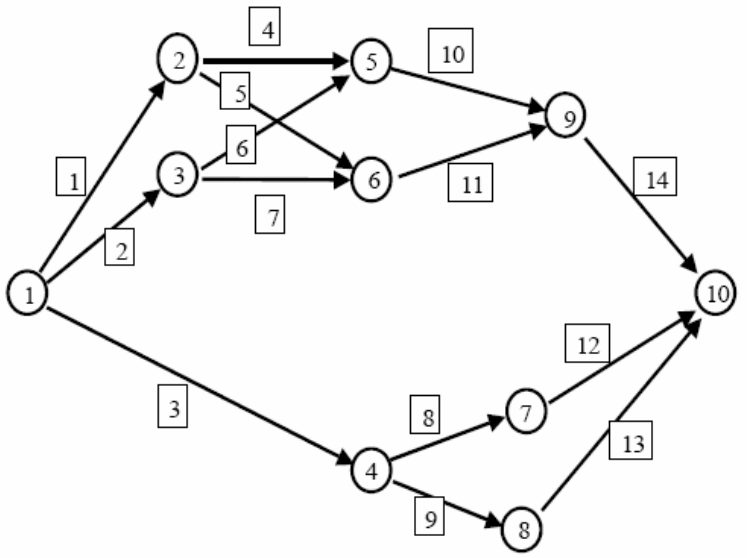
作業編號	1	2	3	4	5	6	7	8	9	10
平均時間	10.0	11.1	12.5	10.0	11.1	12.5	10.0	11.1	12.5	10.0
作業編號	11	12								
平均時間	11.1	10.0								

專案網路 7 ($T_Z = 47,\ P_C = 4,\ U_C = 1$)



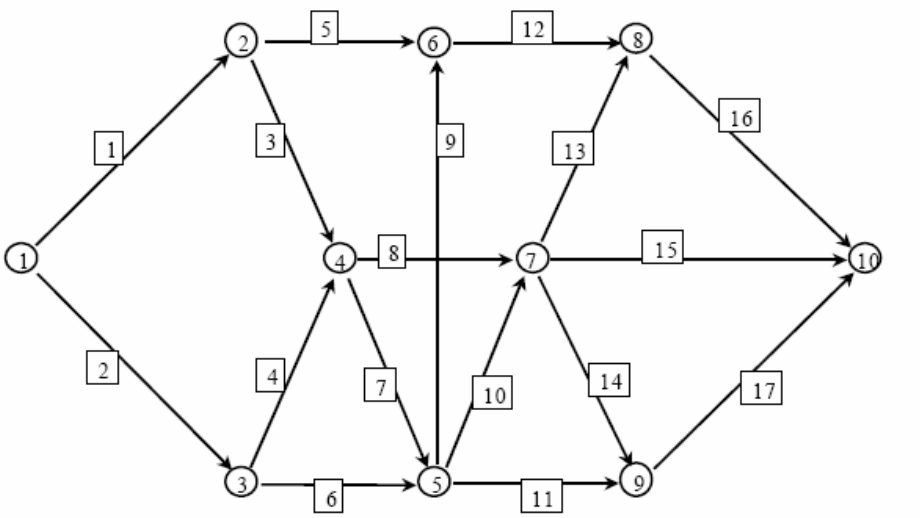
作業編號	1	2	3	4	5	6	7	8	9	10
平均時間	5.0	4.0	6.3	5.0	10.0	6.3	2.0	4.0	5.0	12.5
作業編號	11	12	13	14						
平均時間	11.1	10.0	8.0	10.0						

專案網路 8 **(**$T_Z = 37,\ P_C = 3,\ U_C = 1$**)**



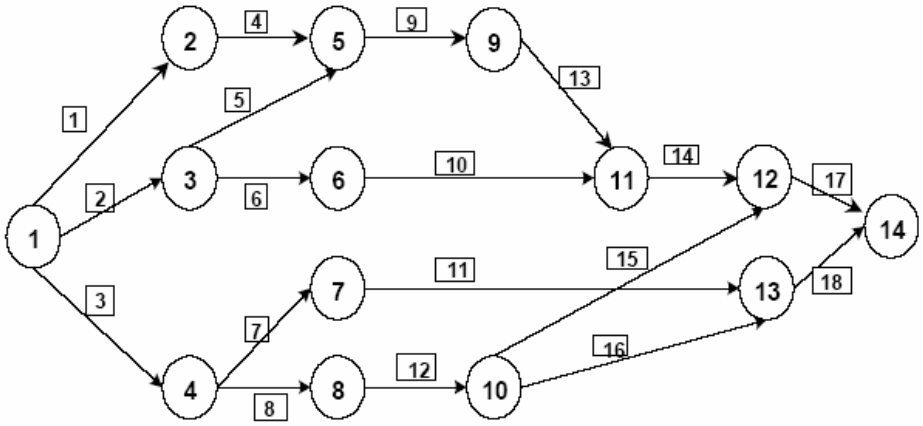
作業編號	1	2	3	4	5	6	7	8	9	10
平均時間	50.0	33.3	25.0	40.0	28.6	22.2	20.0	16.7	33.3	50.0
作業編號	11	12	13	14						
平均時間	66.7	50.0	40.0	33.3						

專案網路 9 **(**$T_Z = 188,\ P_C = 6,\ U_C = 1$**)**



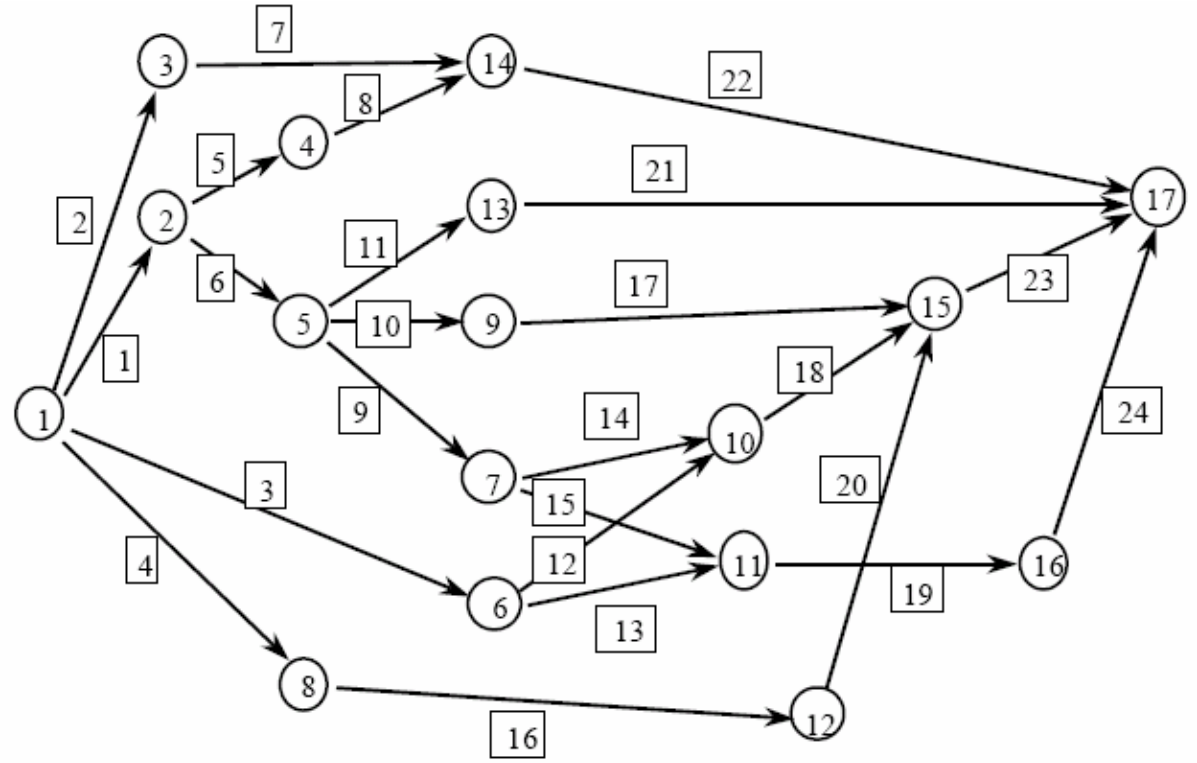
作業編號	1	2	3	4	5	6	7	8	9	10
平均時間	6.0	10.0	5.0	10.0	4.0	5.0	10.0	3.0	3.0	4.0
作業編號	11	12	13	14	15	16	17			
平均時間	2.0	6.0	7.0	2.0	8.0	6.0	9.1			

專案網路 10 ($T_Z = 49,\ P_C = 7,\ U_C = 1$)



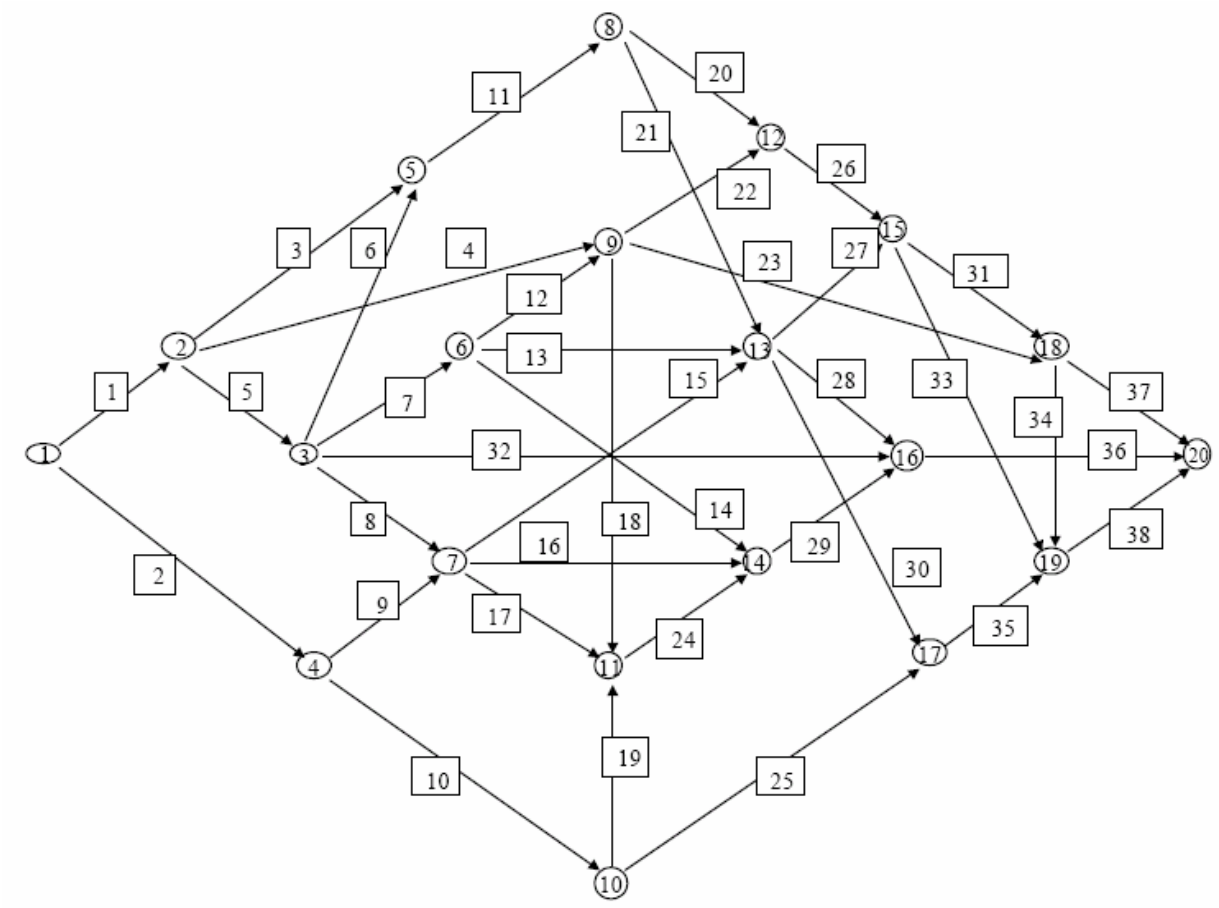
作業編號	1	2	3	4	5	6	7	8	9	10
平均時間	16.7	25.0	10.0	14.3	12.5	25.0	12.5	5.0	14.3	20.0
作業編號	11	12	13	14	15	16	17	18		
平均時間	12.5	14.3	11.1	11.1	20.0	11.1	25.0	16.7		

專案網路 11 ($T_Z = 110,\ P_C = 10,\ U_C = 1$)



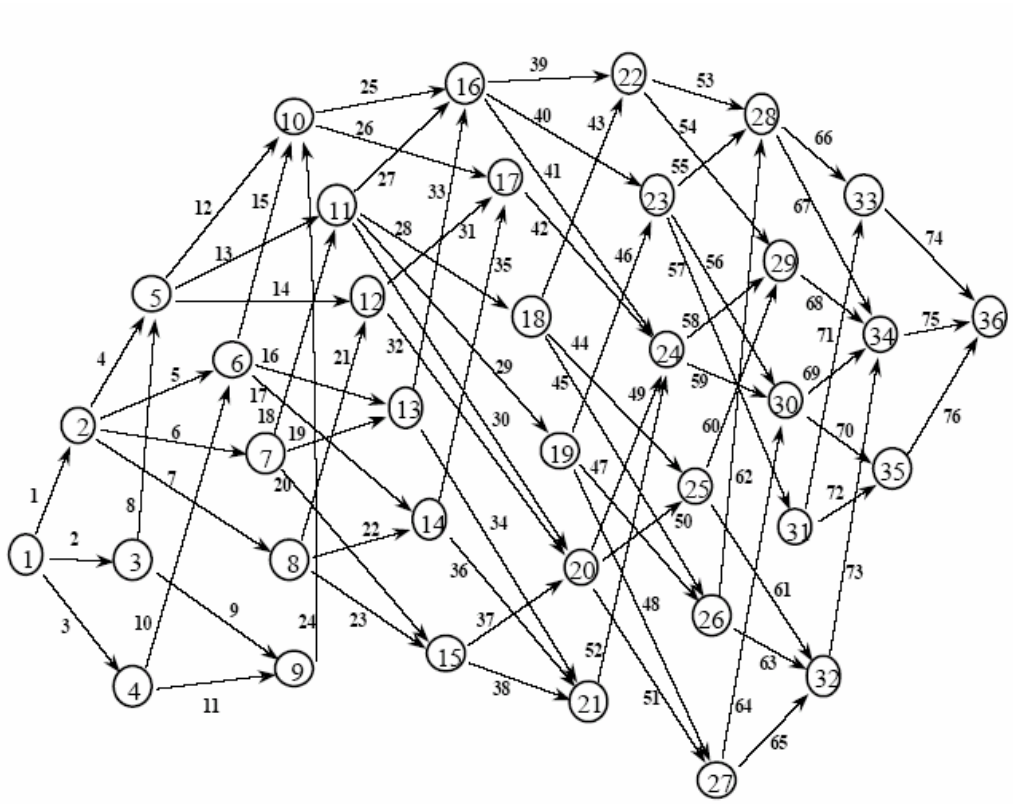
作業編號	1	2	3	4	5	6	7	8	9	10
平均時間	40.0	50.0	45.0	54.9	30.0	35.0	40.0	25.0	30.0	40.0
作業編號	11	12	13	14	15	16	17	18	19	20
平均時間	50.0	26.0	28.0	20.0	25.0	30.0	20.0	18.0	36.0	45.0
作業編號	21	22	23	24						
平均時間	15.0	59.9	25.0	46.1						

專案網路 12　($T_Z = 223,\ P_C = 12,\ U_C = 1$)



作業編號	1	2	3	4	5	6	7	8	9	10
平均時間	16.0	30.0	16.0	20.0	10.0	18.0	15.0	11.0	12.0	40.0
作業編號	11	12	13	14	15	16	17	18	19	20
平均時間	20.0	16.0	10.0	8.0	1.4	21.0	16.0	20.0	19.0	17.0
作業編號	21	22	23	24	25	26	27	28	29	30
平均時間	15.0	18.0	21.0	16.0	20.0	18.0	16.0	12.0	18.0	15.0
作業編號	31	32	33	34	35	36	37	38		
平均時間	24.0	11.0	11.0	22.0	24.0	30.0	15.0	8.0		

專案網路 13 ($T_Z = 151,\ P_C = 5,\ U_C = 1$)



作業編號	1	2	3	4	5	6	7	8	9	10
平均時間	16.7	12.5	14.3	2.0	5.0	10.0	11.1	1.7	3.3	14.3
作業編號	11	12	13	14	15	16	17	18	19	20
平均時間	12.5	2.9	3.8	2.3	11.8	40.0	1.8	13.3	10.5	6.7
作業編號	21	22	23	24	25	26	27	28	29	30
平均時間	4.0	6.7	2.9	2.5	2.7	2.1	11.8	12.5	6.7	4.0
作業編號	31	32	33	34	35	36	37	38	39	40
平均時間	4.0	2.9	2.5	14.3	1.7	6.7	4.0	6.7	15.4	13.3
作業編號	41	42	43	44	45	46	47	48	49	50
平均時間	6.7	4.5	2.9	1.9	14.3	2.9	8.3	14.3	7.7	4.0
作業編號	51	52	53	54	55	56	57	58	59	60
平均時間	2.9	2.2	4.8	11.9	5.0	2.2	1.8	11.1	8.3	5.6
作業編號	61	62	63	64	65	66	67	68	69	70
平均時間	11.1	1.8	4.0	2.9	14.3	12.5	7.1	2.9	10.5	4.8
作業編號	71	72	73	74	75	76				
平均時間	3.1	2.4	2.9	1.6	2.7	2.2				

專案網路 14　($T_Z = 121,\ P_C = 4,\ U_C = 1$)

附錄 M：隨機性專案網路 KSP, PCI 及 ACI 相關問題探討

M.1 隨機專案網路 KSP 問題探討

隨機性專案網路的路徑完成時間為時間機率函數，然而專案網路中存在多條專案網路完成路徑，當這些路徑之時間機率分佈函數值域或有重疊，表示每條專案網路完成路徑均有機會成為該專案的關鍵路徑(Critical Path)。若以確定性專案網路模式之要徑法(Critical Path Mathod,CPM)，或使用 PERT 模式分析，單獨指定其中最長的路徑時間作為專案網路完成時間，依本論文第三章之內容所述，並不正確。故於隨機性專案網路中，選取前 K 條關鍵路徑作為專案網路完成時間參考之標的，就管理的角度而言是有其意義的。根據 Eppstein(1994)所言，可引申出兩點意義：

1. 保持彈性：專案網路中的關鍵路徑可能受多種因素影響，決策者不一定會將所有的因素都能考慮到，未考慮到的因素實際上會影響已決定之關鍵路徑，因此選取 K 條關鍵路徑能夠保持適當的彈性。
2. 敏感度分析：決策者欲瞭解網路中某一因子或參數對關鍵路徑的影響，選取 K 條關鍵路徑能夠清楚知道該因子或參數對路徑變化的狀況。除此之外，如果能夠進一步準確求得前 K 條路徑之機率分佈函數(Probability distribution function, *pdf*)，則專案管理者便能掌握最充分的管理資訊。

在確定性專案網路中，計算前 K 條最短路徑問題(K-Shortest Path problem, KSP)最早由 Hoffman & Pavley(1959)所提出，隨後針對 KSP 問題，相繼的文獻較晚期的計有 Martin(1984)運用「路徑消去演算法」(*path-deletion algorithm*)，將專案網路建立 $G_1, G_2, ..., G_K$ 等 K 個有順序性之次網路，若在某次網路 G_r 中找到其最短路徑 π_1^r，則 π_1^r 即為原網路之第 r 條最短路徑。Hadjiconstantinou & Christofides(1998)使用了 Katoh, et al. (1982)方法(*FSP*)，提出找尋前 K 條最短路徑。Rink(1998)則提出一「雙重掃除演算法」(*Double-Sweep Algorithm*)來處理 KSP 的問題。近年來 Francesca, et al.(2001)則利用處理最短路徑中所使用之標籤修訂法及標籤設定法(label-setting methods)來處理 KSP 問題，並比較多種方法證明標籤修訂法中之 *Bell-Ford method* 最差，*pure threshold method* 表現最好。

本論文所處理的對象是隨機性專案網路中前 K 條關鍵路徑(最長時間路徑)，而非最短時間路徑。兩者雖然標的不同，但在演算的方法上沒有太大的差異。然而針對前 K 條關鍵路徑的相關文獻並不多，本論文僅收集

Dodin(1984)及 Tan, et al. (2001)兩篇相關之文獻。Dodin(1984)主要運用隨機優勢(Stochastic dominance)的概念，計算專案網路節點的輸出時間，並利用該概念之遞移性(Transitivity)，決定該節點輸入路徑的順序性。Dodin 在文獻中指出該遞移性存在的條件是：專案網路所有的作業時間機率分佈，必須符合對稱的要求(例如常態分佈)。Tan, et al. (2001)則是在隨機性專案網路環境中及考慮作業間依的情境下，處理汽車發車數量的 KSP 問題；利用網路擴展方法及隨機優勢(stochastic dominance)的概念，以指數分佈作為其估算分析的資料依據。該兩篇文獻對於專案網路作業時間機率分佈函數有所限制，而且運用隨機優勢的概念並未考慮本論第三章所述之最長路徑偏差(LPB)之問題，因此所求得的前 K 條關鍵路徑仍會與實際有所差異。

M.2 隨機專案網路 PCI 問題探討

針對隨機性專案網路，早期 Molcolm, et al. (1959)及 Wiest & Levy(1977)，使用 PERT 分析模式，累加每條路徑內所有的作業期望時間，並以最長時間的路徑視為關鍵路徑(Critical Path, CP)，其做法概念同確定性專案網路中所採用之關鍵路徑方法(Critical Path Method, CPM)。然依隨機性專案網路之特性，任何一條完成路徑均有機會成為最長路徑，因此隨機性網路關鍵路徑應以一機率值來表示，並將該值稱為關鍵路徑指標(Path Critical Index, PCI)(Anklesaria & Drezner(1986), Ragsdale(1989))。Van Slyke(1968)及 Sigal (1979)分別利用"Crude MCS"及"Conditional MCS"計算 PCI，Martin(1965)亦對 PCI 給予定義，然 PCI 較具代表性的表示方式為 Dodin(1984)及 Dodin & Elmaghraby(1985)所提出之隨機優勢的概念，且已有若干的文獻採用之，其表示式如下:

$$PCI = \Pr\{\pi_i > \pi_j \mid j \neq i,\ \pi_i, \pi_j \in A\} \tag{M.1}$$

π_i, π_j 表示專案網路中完成路徑 i 及 j，A 為網路中所有完成路徑之集合。

(M.1)式主要的問題是在於 π_i, π_j 兩路徑時間是否能得到準確的估算? 由本論文第三章內容所述，若以 DA 演算法執行估算，因其未考慮路徑相依之問題，所求出之 PCI 值亦不會準確。

本論文擬仍採用(M.1)式之定義，唯估算 π_i, π_j 路徑的時間採第三章所提之 LCTA 演算法，本文擬以圖 M.1 為例，實際做說明之。

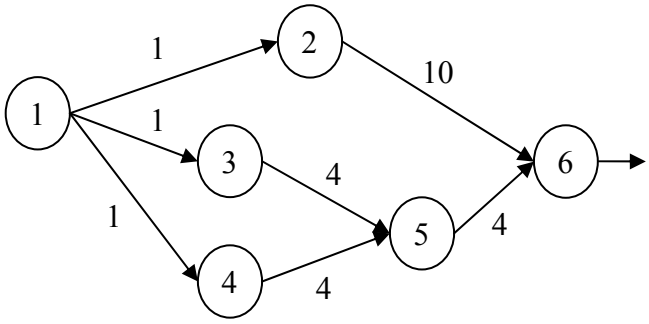


圖 M.1　說明隨機優勢演算法之圖例

圖 M.1 之專案網路的完成路徑集合共有三條，分別為路徑(1,2,6), (1,3,5,6)及(1,4,5,6)。路徑(1,2,6)的隨機優勢指標值表示為 DP_1^6：

$$DP_1^6 = DP(Y_1^6) = \Pr(Y_1^6 \succ Y_2^6) \tag{M.2}$$

其 Y_i^j 為節點 j 的第 i 條輸入路徑的完成時間，

$$Y_1^6 = X_{12} + X_{26}$$

$$Y_2^6 = X_{56} + \max(X_{13} + X_{35},\ X_{14} + X_{45}) \tag{M.3}$$

而路徑(1,3,5,6)的隨機優勢指標值則可以表示為：

$$DP_2^6 \cdot DP_1^5 = \Pr(Y_2^6 \succ Y_1^6) \cdot \Pr(Y_1^5 \succ Y_2^5) \tag{M.4}$$

其中：

$$Y_1^5 = X_{13} + X_{35},\quad Y_2^5 = X_{14} + X_{45} \tag{M.5}$$

同理，路徑(1,4,5,6)的隨機優勢指標值則可以表示為：

$$DP_2^6 \cdot DP_2^5 = \Pr(Y_2^6 \succ Y_1^6) \cdot \Pr(Y_2^5 \succ Y_1^5) \tag{M.6}$$

每一條完成路徑之隨機優勢指標值乘績的最後結果極為該路徑之 PCI 值，其值愈大則表示成為網路關鍵路徑的機會愈大。以圖 M.1 為例，所求出的路徑 PCI 值與 MCS(20000 取樣)的結果來比較如表格 1 所示：

專案網路中節點 j 之 Y_i^j 可以由 LCTA 演算法求出，再配合隨機優勢指標值之計算，各節點輸入路徑之隨機優勢指標值便可有系統的求出，最後能夠解出專案網路所有完成路徑之 PCI 值。

若專案網路節點 j 僅有兩條輸入路徑，其隨機優勢指標值可參考(M.1)式執行如下：

$$DP_1^j = \Pr(Y_1^j \succ Y_2^j),\quad DP_2^j = 1 - DP_1^j \tag{M.7}$$

表格 M.1　圖 M.1 之隨機優勢值求解

	(1,2,6)	(1,3,5,6)	(1,4,5,6)
MCS	0.28	0.28	0.44
PSDPA	0.26	0.26	0.48

當專案網路節點輸入路徑超過兩個以上，(M.1)式之路徑隨機優勢演算會有所不同。根據 Dodin(1984)，隨機優勢指標之計算不一定具備遞移性(transitivity)。令有三路徑 a, b, c，若 $a>b$ 且 $b>c$，不一定推得 $a>c$ 之結論，除非 a, b, c 三個機率函數分佈均具對稱性質。因此當節點 j 的輸入路徑超過兩個(令為 m)，其輸入路徑隨機優勢指標值的計算就會比較複雜，且其隨機優勢指標計算次數共計為 $\frac{m!}{2\cdot(m-2)!}$，本論文擬利用一矩陣資料結構來記錄變數間隨機優勢指標值，以下列 3 個路徑 a, b, c 為例，說明如下。

先執行 a，b 兩路徑優勢指標值之求解，且令 $\Pr(a \succ b)=0.3$, $\Pr(b \succ a)=0.7$，再執行 a, c 兩變數隨機優勢指標值之求解，且令 $\Pr(a \succ c)=0.4$, $\Pr(c \succ a)=0.6$，再執行 b, c 兩變數隨機優勢指標值之求解，且令 $\Pr(b \succ c)=0.2$, $\Pr(c \succ b)=0.8$，上述的結果分別記錄於表 M.2 的矩陣中，

表 M.2　隨機優勢指標值矩陣

	a	**b**	**c**	**DP**
a	-	0.3	0.4	0.23
b	0.7	-	0.2	0.3
c	0.6	0.8	-	0.47

令矩陣中之資料表為 $Matrix\_value(i, j)$，以下式計算隨機優勢指標值：

$$\sum_{j}^{m} \frac{Matrix\_value(i,j)}{(m(m-1)/2)} \tag{M.8}$$

即變數 a 的隨機優勢指標值 $DP_a = (0.3+0.4)/3 = 0.23$，變數 b 的隨機優勢指標值 $DP_b = (0.7+0.2)/3 = 0.3$，變數 c 的隨機優勢指標值 $DP_c = (0.6+0.8)/3 = 0.47$。

M.3 隨機專案網路 ACI 問題探討

早期討論隨機性專案網路 ACI 值的文獻計有 Williams(1992)所提之 “Cruciality Index, CRI” 及 “Significance Index, SI” 兩種指標計算，Elmaghraby(2000)對該兩種方法作了一些評論並也指出其實務應用上的缺點，並同時提出另一種方式，使用 Response surface approach 及 Taguchi 因子分析法，針對作業時間的變化對專案網路完成時間做敏感度之分析。另 Cho&Yum(2004)則利用回歸分析模式(Regression model)來建立作業 ACI 值與其作業時間間之關係，並藉此關係來分析作業時間變化對專案網路完成時間之影響。上述文獻所提出之各種方式，不是過時就是過於複雜，而無法得到實務上的應用。基於此也可以窺視到，求取隨機性專案網路之 ACI 值並非易事。Dodin(1985)之文獻，亦提出一演算法，能夠實際求算專案網路 ACI 值，基於本論文參考該文獻前述之隨機優勢指標值求解的延續性，針對該文獻所提之方法來進行實驗，並以圖 M.2 為例，實際算出作業 $X_{(11,12)}$ 之 ACI 值 ACAP(11, 12)。

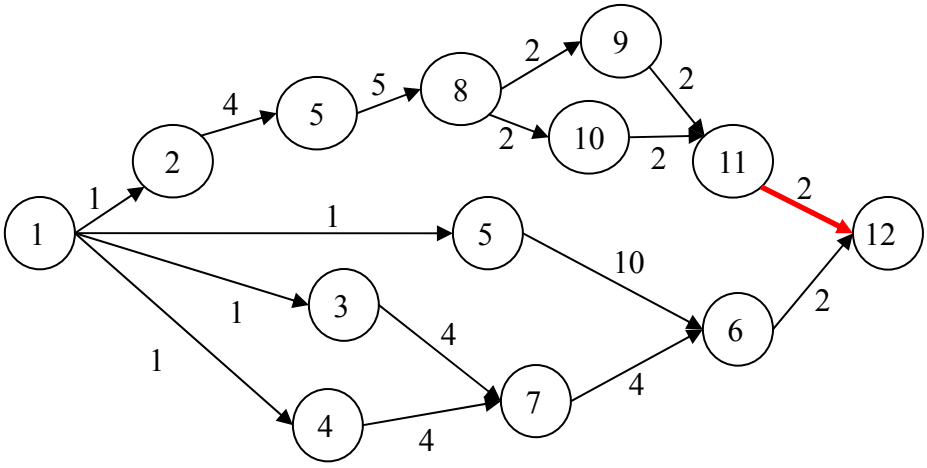


圖 M.2　計算 ACI 值之圖例

Dodin(1985)文獻中求算 ACAP(i,j)的步驟如下:

1. 從節點 12 開始計算(j=12)，並依序針對節點 11, 10, 9….進行計算。

2. 計算C_j (節點j之 cutest)。

$$C_j = \{(i,k) \in A : i < j \text{ and } k \geq j\}$$

$$C_j = C_{j+1} + \{(i,j) \in A : i < j\} - \{(j,k) \in A : k > j\} \quad \text{(M.9)}$$

3. 令$R(\pi_{ij})$為專案網路節點i到節點j所有路徑的集合，計算W_{ij}及V_{ij}時間，其定義為:

$$W_{ij} = T_i^f + Y_{ij} + T_j^b, \;\; T_i^f = \max\{R(\pi_{1i})\}, \;\; T_j^b = \max\{R(\pi_{jN})\} \quad \text{(M.10)}$$

$$V_{ij} = \max_{(r,k) \in C_j,\ (r,k) \neq (i,j)} \{W_{rk}\} \quad \text{(M.11)}$$

4. 計算:

$$ACAP(i,j) = \Pr(W_{ij} \geq V_{ij}) \;\; \textit{for all the } (i,j) \in C_j \quad \text{(M.12)}$$

$$ACN(N) = \sum_{(i,j) \in C_N} ACAP(i,j), \;\; ACN(N\text{-}1) = ACAP(N\text{-}1, N) \quad \text{(M.13)}$$

5. $j = j\text{-}1$，如果節點j的輸入路徑數僅一條，則$ACAP(i,j) = ACN(j), \; \textit{where } i < j$。如果節點$j$的輸入路徑數目大於 1 的話:

(1) 同上述步驟 2,3,4 所述之方法，計算所有連接到節點j之$ACAP(i,j), \; i < j$。

(2) 計算下式：

$$ACAP(i,j) = ACN(j) - \sum_{\substack{k \in B(j) \\ (k \neq i)}} ACAP(k,j) \quad \text{(M.14)}$$

6. 重複上述步驟 5 直到j=2 為止。

Dodin(1985)設計了很有效率的演算法來計算出專案網路節點之 cutset，在根據上述之步驟計算出節點之 ACI 值。步驟 3 中，專案網路W_{ij}及V_{ij}時間之估算是採用 DA 演算法，會產生估算上的誤差，因此 ACAP(ij)值的計算也會受到影響。

綜合上述， Dodin & Elmaghraby (1985)雖能對隨機性專案網路之 PCI 及 ACI 值的計算提出很有效率的演算流程理論，但終究須以精確之專案網

路完成時間為基礎，才能得到有效之結果。這也是本論文針對隨機性專案網路問題，先發展 LCTA 及 MIA 演算法之原因。