國　立　交　通　大　學
運 輸 科 技 與 管 理 學 系

碩 士 論 文

以改良式螞蟻演算法求解航機降落排序問題

A Modified Ant System Algorithm for the Aircraft Landing Problem

研 究 生：吳景堯
指導教授：黃寬丞　博士

中華民國九十八年七月

以改良式螞蟻演算法求解航機降落排序問題

A Modified Ant System Algorithm for the Aircraft Landing Problem

研 究 生：吳景堯　　Student: Jing-Yao, Wu

指導教授：黃寬丞　　Advisor: Dr. Kuan-Cheng, Huang

國 立 交 通 大 學

運輸科技與管理學系

碩 士 論 文

A Thesis

Submitted to Department of Transportation Technology and Management

College of Management

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of Master

in

Transportation Technology and Management

July 2009

HsinChu, Taiwan, Republic of China

中華民國九十八年七月

以改良式螞蟻演算法求解航機降落排序問題

學生：吳景堯　　指導教授：黃寬丞 博士

國 立 交 通 大 學

運輸科技與管理學系 碩士班

摘要

由於燃油等因素限制，飛機可以降落的時間為一個時窗範圍，且在此時窗範圍內，有最理想的降落時間點。此外，飛機降落會產生空氣擾動，為確保飛航安全，飛機間的降落還必須遵守隔離時間。航機降落排序問題，便是在滿足這些限制之下，分配航機的降落順序及時間。本研究將利用螞蟻演算法求解航機降落排序問題，並透過類似動態規劃的概念，計算固定某一架飛機後的成本下限值，藉此下限值限制搜尋空間，改良螞蟻演算法，更有效率的解決航機降落排序問題。研究結果顯示，問題變數較少時螞蟻演算法可求得最佳解，但隨問題變數增加，求解品質越差。而改良式螞蟻演算法能夠有效限制搜尋空間，當問題變數增加時，仍能求得最佳解，顯示改良式螞蟻演算法求解品質較佳。

關鍵詞：航機降落排序、螞蟻演算法

A Modified Ant System Algorithm for the Aircraft Landing Problem

Student: Jing-Yao, Wu Advisor: Dr. Kuan-Cheng, Huang

Department of Transportation Technology and Management
National Chiao Tung University

Abstract

In this paper, we consider the aircraft landing problem at an airport. Because of the fuel, the aircraft landing time will be a time window, in which there are the best landing time for each aircraft. Moreover, when aircraft lands, it will generate air turbulence. For safety reasons, between aircrafts landing should conform to separation time. We solve the aircraft landing problem by ant system. And similar to dynamic programming, we consider a conditional lower bound to constrain the search space, in order to solve the problem faster. The result shows that, ant system can solve the problem, but when variables change more, the solution get worse. And the modified ant system can constrain the search space effectively. When variables change more, it can also find the solution.

Keywords: Aircraft land, Ant system.

誌　謝

新竹從來就不是一個討喜的城市，交大卻擁有我許多回憶。還記得當年的懵懂無知，到現在終於脫胎換骨，學有所成。一切的一切，都要感謝曾經幫助過我，教導我的人。老師、朋友、同學、家人以及其他長輩、晚輩等，僅以本篇誌謝聊表謝意。

在交大一待就是六年，說長不長說短不短，卻是我目前人生四分之一的時間。回顧過去這些日子，雖已久遠，但又歷歷在目。論文的完成，就是這些日子的結晶，老師辛苦教導下的精華。研究得以完成，最要感謝的是我的指導老師，黃寬丞老師。雖然做研究的過程有血有淚，但就是在如此苛刻的要求下努力，才有今天論文的誕生，指導老師是最辛苦的。

在求學過程中，最要感謝的是我的同袍們，無論是在老師強大的壓迫下完成作業，或是趁著夜闌人靜之時，一起抒發對課業壓力的不滿，都要感謝這些陪在我身邊的朋友們，讓我更有動力去挑戰下一個難關。感謝我的大學同袍，達叔、邦展、新祐、建嘉、阿志、番婆、零六等人，也感謝我的大學同袍兼研究所戰友，佛諭、薈明、維中、竹軒、神父等人，以及實驗室隔壁的戰友，友維、明翰、老總、禎祥等人。

另外，在研究過程中，更要感謝水深火熱中，實驗室一起奮鬥的夥伴們，這種在同一實驗室的革命情感，更是不為人所知的，一定要身處同一個實驗室才會了解。感謝智翔、兆哲、丞博、郁英，在實驗室若沒有你們的陪伴，肯定會失去一些樂趣。

最後，我要感謝我的家人，儘管我不擅長表達對你們的謝意，但是對於你們無盡的付出，讓我可以安心在此完成學業，是我最大的後盾。也感謝其他支持我的人，雖然沒有提到名字，但是對我來說一樣的重要。要感謝的人太多，無法一一細數，對於你們的幫助，我將銘記於心。

吳景堯 謹誌

2009.7

於 新竹交大

目錄

圖目錄

表目錄

第一章 緒論

1.1 研究背景與動機

目前全世界各大型重要機場的飛航流量與日俱增，以全球最繁忙的國際機場之一香港國際機場為例，根據交通部交通統計資料(交通部運輸統計，2008)指出，香港機場在 2007 年飛航起降次數高達 29 萬 5 千架次，較 2006 年 28 萬架次多了約 5%，且呈現逐年增長的趨勢。並且，近年來大陸蓬勃發展亦帶動整體航空運輸的潛力，根據 2007 年民用機場行業分析報告(2008 年民用機場行業分析報告)指出，2007 年起降次數達到 394 萬架次，較 2007 年增長 13%，足見航空運輸佔有不可取代的重要地位。

然而，因為必須符合飛機起降的要求與限制，在尖峰時間，往往會造成班機延誤，導致成本浪費的問題。因此，有鑒於空中交通量不斷增加，飛航管制(Air Traffic Control，ATC)亦扮演重要調度的角色。當飛機要降落時，航機的位置顯示在航管單位的終端雷達，再由塔台指引飛機進場。塔台由雷達得到飛機的高度及速度等資訊，再給予指示到機場指定跑道上來完成降落。然而，以往塔台都是以先到先服務(First Come First Serve，FCFS)的方式進行指派，亦即先進入雷達範圍的飛機先給予優先降落的權力，但此種指派的方式的適當性卻值得商榷。

並且，由於在短期內，機場無法以擴建跑道的方式來滿足與日俱增的需求，在跑道容量限制下，有效率的降落就顯得非常重要。因此，需要發展出一個有效的工具來幫助塔台指引飛機快速有序的降落。本研究之重點便在於發展一種更有效率的方法，取代舊有的 FCFS，使飛機能夠在最短時間內平安降落。

1.2 研究範圍與目的

降落排序在忙碌的機場是一項非常困難的作業。在飛機降落之前，須經過塔台的層層關卡，當飛機進入雷達範圍之後，飛機的號碼、高度、速度等資訊便會傳給塔台，而塔台基於這些資訊給予指示，指示相關速度、高度及降落跑道。而在尖峰時間，塔台便需要安全且有效率的掌控所有進入雷達的飛機，並指示分配到不同的跑道進行降落。塔台便產生一項重複式的決策問題：下一架降落飛機為何，

何時降落，在哪個跑道降落。

航機降落排序問題，屬於 NP-hard 問題，問題複雜度隨著飛機架數的增加而上升，因此我們需要利用啟發式解法來求解此問題。而螞蟻演算法(Ant Colony Optimization, ACO)過去在解決旅行銷售員問題(Traveling Salesman Problem, TSP)上，有不錯的求解品質，而航機降落排序問題與旅行銷售員問題相似。因此，本研究嘗試使用螞蟻演算法求解航機降落排序問題，並利用動態規劃的概念改良螞蟻演算法。最終目的在於利用改良式螞蟻演算法求解航機降落排序問題，以求更有效率的解決降落排序問題，使總成本最小化。

第二章 問題背景

本章節介紹航機降落排序問題(Aircraft Landing Problem，ALP)之相關基礎概念研究文獻，以及過去研究此議題之成果。

2.1 航機降落排序問題概述

航機降落排序問題屬於排序問題，排序問題主要在解決實際上的問題，分配有限的資源以完成工作。解決排序問題最主要有三個目標：最有效利用資源、快速反應需求、以及在期限內解決問題(Ciesielski, 1995)。而航機降落排序問題主要在於求解，在有限制的飛機降落時窗以及飛機隔離時間內，飛機實際的降落順序以及飛機降落時間。

有關航機降落順序問題，Beasley(2000)將問題定義如下：當飛機開始進入雷達範圍時，必須考慮一個時窗(Time window)範圍，如圖 1.1。這個時窗範圍由飛機最早可降落時間(Earliest landing time)及最晚可降落時間(Latest landing time)組成，表示飛機降落不可超過這個時窗範圍。最早可降落時間表示飛機不用等待可直接以最快速度降落，而最晚可降落時間則受到飛機本身燃油的因素影響。另外飛機本身有最佳的降落時間，稱為目標降落時間(Target time)，表示飛機以最理想的速度降落所在的時間點。而塔台便依照各飛機之時窗，以及目標降落時間進行降落的排序。以下將對這些相關特性做探討：

- Earliest landing time (E_i)：最早降落時間，指飛機以最快的速率飛行，從雷達偵測到進場降落的最早時間點。
- Latest landing time (L_i)：最晚降落時間，指考量飛機油料，允許在空中盤旋的最大時間後，進場降落的時間點。
- Target time (T_i)：目標降落時間，指飛機以最經濟且最理想的速度進場所降落的時間點。

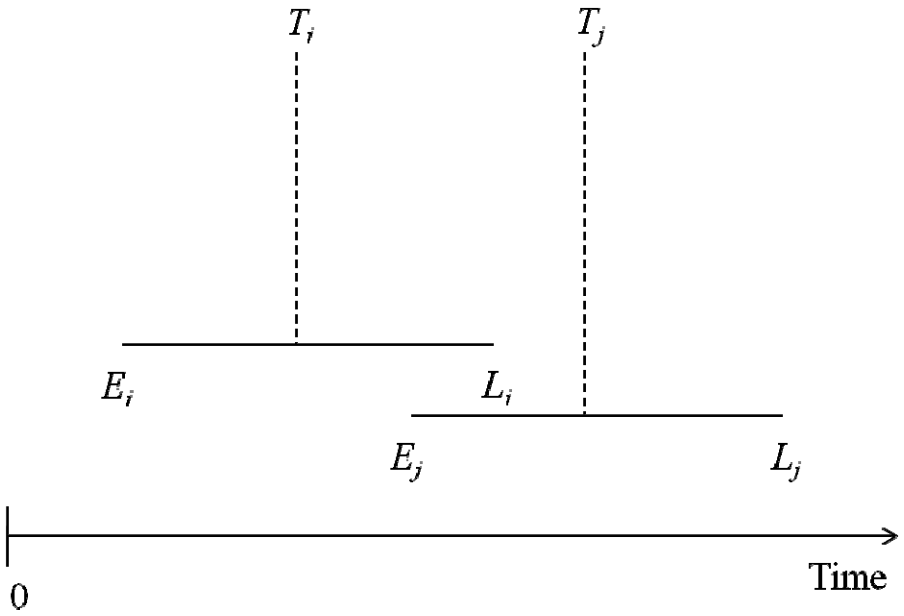


圖 2-1 時窗範圍示意圖

航機降落排序問題之主要決策變數，為安排航機降落之時間。假設共有 n 架飛機準備降落，若以 y_i 代表飛機 i 之降落時間，而其時窗限制，則必須滿足 y_i 落在 E_i 及 L_i 之間，以確保飛機能在時窗內進行降落，意即$E_i \le y_i \le L_i$。

除時窗之外，降落時間的安排還必須考慮到不同飛機降落間的隔離時間(Separation time)。對於不同等級的飛機來說，隔離時間主要考慮引擎所產生的空氣擾動，而會給予不同的隔離時間做限制，以確保飛航安全。為設定隔離時間，相關民航單位將飛機分成數個不同的等級，並且設定不同等級的隔離時間，一般相關研究都是以這些規定的數值做為基礎。考慮兩飛機組合(i,j)，若飛機 i 比飛機 j 先降落，則必須考慮其隔離時 S_{ij}，必須$y_j \ge y_i + S_{ij}$。

為便於建立數學模式，航機降落排序問題可定義一二元決策變數 δ。在考慮兩架飛機組合(i,j)降落之先後順序，代表不是飛機 i 先降落$(\delta_{ij} = 1, \delta_{ji} = 0)$，就是飛機 j 先降落$(\delta_{ij} = 0, \delta_{ji} = 1)$。因此必須滿足$\delta_{ij} + \delta_{ji} = 1$。

事實上，在某些情況下，我們可以從飛機組合(i,j)之時窗關係來判斷其降落先後順序，亦即$\delta_{ij} = 1$或$\delta_{ji} = 1$。舉例來說，若飛機 i 之時窗為[10,50]，而飛機 j 時窗為[70,100]，則飛機 i 之降落順序必在飛機 j 前面，即是確定 $\delta_{ij} = 1$，$\delta_{ji} = 0$。此外，並不是隨意的時間降落都可滿足隔離時間限制，例如當限制時間 S_{ij} 為 15 時，則無論 i、j 兩飛機在何時降落都不會造成影響，但若 S_{ij} 為 25 時，則對某些 y_i 和 y_j 的組合不一定會滿足隔離時間限制。

接下來考慮目標時間與成本間的關係(如圖 2-2)。首先定義 g_i 為飛機 i 在未到達

目標時間前降落所造成的成本，h_i 為飛機 i 在超過目標時間後才降落所造成的成本。其中 g_i、h_i 皆為正數，代表圖中線段之斜率。我們假設飛機在非目標時間降落所造成的成本為線性，若飛機降落時間在目標時間前或目標時間後，則會產生成本，若恰巧在目標時間，則其成本為零。

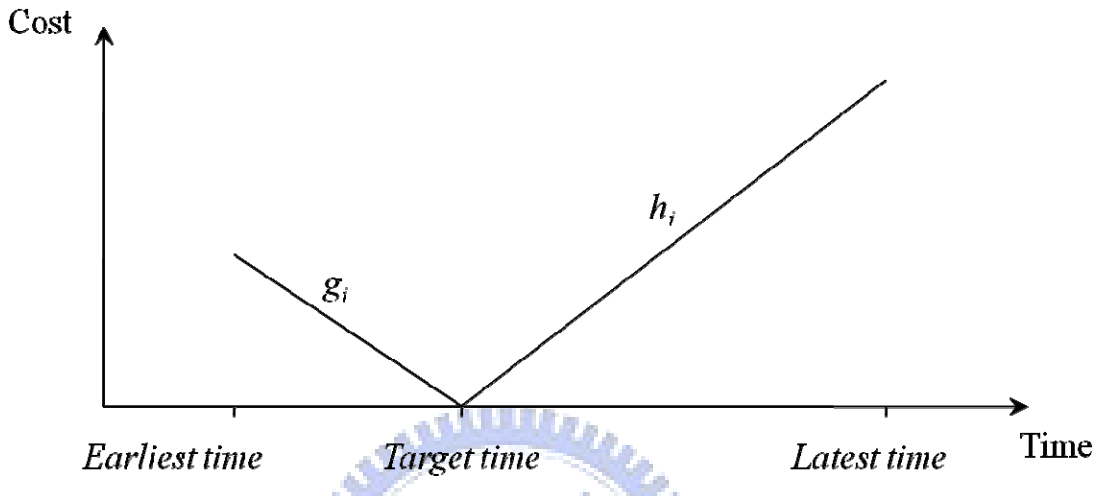


圖 2-2 飛機降落之成本函數圖

2.2 問題假設

在求解問題之前，我們會先對問題做一些假設(Beasley, 2000)，因為在求解作業研究(Operation Research, OR)問題時，是無法完全吻合實際情況的。問題假設共分為以下部分：

1. 控制問題(control problem)：在求解問題決策時，飛機的降落不考慮控制問題，也就是說，此問題假設飛機都能在塔台分配好的時間降落，不會影響整體決策。
2. 隔離時間(separation time)：本問題假設飛機間的隔離時間，只與該飛機的大小有關，而與其他因素無關。
3. 最晚降落時間(latest time)：最晚降落時間與燃料有關，本問題所設定之最晚降落時間，則是假設燃料用完的最後一刻，然而，實際情況無法等到飛機燃料用完之後才降落。
4. 跑道(runway)：隨著機場的跑道數量或是政策的不同，會將跑到分為多跑道或單跑道，而本研究求解航機降落排序問題，只考慮單飛機跑道，不考慮多飛機跑道。

5. 目標式(objective function)：本問題所設定之目標式，只有飛機降落在目標時間沒有成本，目標時間以外的時間降落都會有成本，且成本為線性。而目標時間降落的飛機沒有成本，是因為此決策問題能影響的只有降落時間，其他成本並無法影響，因此設定目標時間降落的飛機成本為零。

2.3 數學模式

接下來將問題部分以數學規劃是描述(Beasley, 2000)，包含基本符號定義以及數學規劃式。其中 y_i 為決策變數，α_i 、β_i 、δ_{ij} 是為解題方便而設計之變數。並且假設 E_i 、L_i 、S_{ij} 為整數。

符號定義：

n ：飛機數

E_i ：飛機 i 的最早降落時間($i = 1,\ldots,n$)

L_i ：飛機 i 的最晚降落時間($i = 1,\ldots,n$)

T_i ：飛機 i 的目標降落時間($i = 1,\ldots,n$)

S_{ij} ：飛機 i 與飛機 j 降落的隔離時間，其中飛機 i 比飛機 j 先降落($i = 1,\ldots,n$ ，$j = 1,\ldots,n$ ，$i \neq j$)

g_i ：飛機 i 在目標時間 T_i 前降落之單位時間懲罰成本

h_i ：飛機 i 在目標時間 T_i 後降落之單位時間懲罰成本

y_i ：飛機 i 實際降落時間

α_i ：飛機 i 在目標時間 T_i 多久前降落

β_i ：飛機 i 在目標時間 T_i 多久後降落

$$\delta_{ij} = \begin{cases} 1, \text{表示飛機} i \text{在飛機} j \text{前降落} \\ 0, otherwise \end{cases}$$

目標式與限制式：

$$min \sum_{i=1}^{n} (g_i \alpha_i + h_i \beta_i) \qquad (2\text{-}1)$$

Subject to

$$E_i \le y_i \le L_i \qquad i = 1, \dots, n \qquad (2\text{-}2)$$

$$\delta_{ij} + \delta_{ji} = 1 \qquad \forall i, j = 1, \dots, n, \quad i \ne j \qquad (2\text{-}3)$$

$$y_j \ge y_i + S_{ij}\delta_{ij} - (L_i - E_j)\delta_{ji} \qquad \forall i, j = 1, \dots, n, \quad i \ne j \qquad (2\text{-}4)$$

$$\alpha_i \ge T_i - y_i \qquad i = 1, \dots, n \qquad (2\text{-}5)$$

$$\beta_i \ge y_i - T_i \qquad i = 1, \dots, n \qquad (2\text{-}6)$$

$$y_i, \alpha_i, \beta_i \ge 0 \qquad i = 1, \dots, n \qquad (2\text{-}7)$$

$$\delta_{ij}, Binary \qquad \forall i, j = 1, \dots, n, \quad i \ne j \qquad (2\text{-}8)$$

式(2-1)為目標式，我們以線性關係討論航機之降落時間與其目標降落時間做比較，對於沒有在目標時間降落的航機給予一懲罰值，因此我們追求最小化目標式。

式(2-2)為確定飛機降落時間在時窗內，式(2-3)限制飛機 i 比飛機 j 先降落，或飛機 j 比飛機 i 先降落，兩種情況只能出現一種。式(2-4)則限制飛機 i 與飛機 j 之間必須滿足隔離時間，若飛機 i 比飛機 j 先降落，即 $\delta_{ij} = 1$, $\delta_{ji} = 0$，則限制式變成 $y_j \ge y_i + S_{ij}$，表示飛機 j 降落時必須滿足飛機 i 與飛機 j 的隔離時間(S_{ij})；若飛機 j 比飛機 i 先降落，即 $\delta_{ij} = 1$, $\delta_{ji} = 0$，則限制式變成 $y_j \ge E_j + (y_i - L_j)$，又 $(y_i - L_j) \le 0$，則限制式變成 $y_j \ge E_j +$ (某數 ≤ 0)，限制式恆成立，表示飛機 j 不須滿足飛機 i 之隔離時間(S_{ij})。而式(2-5)到式(2-6)則限制 α_i、β_i 與 y_i 間的關係，並確定變數 α_i、β_i 為線性。

此數學規劃式共包含 $3n$ 個變數，$n(n-1)$個二元變數，以及$[3n + 3n(n-1)/2]$個限制式，目前沒有找到有效率解決此問題的方法，屬於 NP-hard 問題(Beasley, 2000)。

2.4 文獻回顧

航機降落排序問題約在 1980 年左右就有學者提出用模擬的方法來解決問題(S Ricciardelli et. al, 1981)，並且在之後陸續有學者針對這個問題提出演算法進行求解。然而在早期所發表的論文，針對航機降落排序問題並沒有統一性，除了測試例題不一樣之外，最開始的假設也不一樣，因此也無法比較該演算法的優劣。

Krishnamoorthy et. al(1995)提出以基因演算法(Genetic Algorithm)求解航機降落排序問題，並且利用不同飛機大小的型態來界定隔離時間。而在基因演算法部分，則是利用飛機確切降落時間當作染色體進行交配，奠定基因演算法在航機降落排

序問題上運用的基礎。然而由於當時並沒有一個統一的測試例題，因此該論文所提出的數值測試為自行假設參數，無法與後來提出之論文作比較。並且該演算法在當時運用的時候只能算到 25 架飛機的例題，在現在只能算是初步的小例題而已。

同時，Ciesielski(1995)也提出基因演算法求解航機降落排序問題，但與 Krishnamoorthy 不一樣的是，Ciesielski 將飛機架次及跑道順序加入染色體，並將求解重點放在跑道的選擇部分。然而，Ciesielski 所使用的範例與 Krishnamoorthy 並不一樣，也因此兩位同時提出以基因演算法求解航機降落排序問題的論文，沒有比較其演算法的優劣，但後來多用 Krishnamoorthy 的論文為基因演算法發展的基礎。

之後 Krishnamoorthy et. al(1998)認為航機降落排序問題與機器工作排程有類似的地方，都是以重排序的結果影響目標示，因此，Krishnamoorthy 提出使用 Simplex Algorithm 決定降落時間，並發展出 Space Search Algorithm 來求解航機降落排序問題，並且加入多跑道的因素，最後可以求解到 44 架飛機的問題。

而後由 Beasley(2000)發表之論文將其整合，重新統一定義問題，並在 OR-Library 提供標竿範例及其當前最佳解，供後續研究參考及測試例題。測試例題包括從 10 架飛機開始到 500 架飛機，共十三題例題，其中 10 架飛機到 50 架飛機為小例題，共八題，目前已經找到最佳解(Beasley, 2000)，而從 50 架飛機以後的例題到 500 架飛機，共五題，則是目前只有當前最佳解(Beasley, 2006)，而還沒找到最佳解。

Beasley(2001)更提出使用 Population Heuristics 求解航機降落排序問題，並利用倫敦希思羅機場(London Heathrow)實際資料，求解航機降落排序問題，藉以比較求解出的結果與實際機場運作，可以節省多少成本。Beasley 在此論文中加入不適應值(unfitness)作為比較的另一項考量，顯示可以快速的求得解。並於 2006 年所提出之論文，在 Scatter Search(SS)以及 Bionomic Algorithm(BA)加入不適應值，測試 OR-Library 所提出之標竿範例，其結果顯示，SS 與 BA 大部分例題都可以求得與當前最佳解一樣的解，並且可以將時間控制在 100 分鐘以內，顯示此方法可以在短時間內求得不錯的解。然而，其所求結果並不是每個題目都可以達到當前最佳解，顯示還有改善的空間。

Beasley 在 OR-Library 所提出的標竿範例，基本參數包括每架飛機的出現時間、最早到達時間、最晚到達時間、目標到達時間以及早到與晚到的懲罰值，以及每架飛機間的隔離時間，而求解目標為最小化成本，並找出飛機確切的降落時間。而 Beasley 在 2000 所提出的論文中，已經成功將 10 架飛機到 50 架飛機的小例題求得最佳解；而在 2006 年所提出之論文中，提出 50 架飛機以上的大例題當前最

佳解，以供後續研究航機降落排序問題之演算法比較優劣，或是找到新的當前最佳解。

第三章 螞蟻演算法

本章節介紹螞蟻演算法之相關文獻及本研究使用之螞蟻演算法。3.1 介紹螞蟻演算法之特色及運算架構、參數定義、演算流程以及演算法不同的分支。3.2 介紹本論文使用螞蟻演算法求解航機降落排序問題之參數設定及流程。3.3 探討螞蟻演算法中 Heuristic Value 設定，在航機降落排序問題如何應用。

3.1 螞蟻演算法(Ant Colony Optimization, ACO)之相關文獻

螞蟻演算法是為了求解最佳化問題而設計出的啟發式解法，演算法的初始架構最早由 Dorigo et. al (1991)提出，之後陸續有學者根據此架構做改變。螞蟻演算法是由模仿自然界中，螞蟻覓食的行為而來。螞蟻藉由費洛蒙(pheromone)的遺留(laying)及跟隨(following)，達到彼此交換訊息、分享經驗的目的。而當有不同路徑可以選擇時，螞蟻會偏好有較多費洛蒙的路徑，同時在走過的路徑上留下自己的費洛蒙。因此，經過一段時間之後，大部分的螞蟻都會選擇相同的路徑行走。

螞蟻演算法最主要的特點，在於在求解過程中，會結合前面較佳的解作為參考，作為後面求得解的依據。並且，螞蟻演算法會給予較差的解一部分的權重，以避免演算法收斂到區域最佳解。控制螞蟻演算法的主要機制有兩個，分別為吸引力(attractiveness)及足跡(trail level)。吸引力表示路徑本身對螞蟻影響，通常是由不同類型的題目給予不同的權重。足跡則是前面螞蟻走過的路徑對後面螞蟻的影響，也就是藉由費洛蒙的遺留及蒸發來控制後面螞蟻選擇路徑的機率。螞蟻演算法主要包含下列部分：

1. Solution Construction：讓螞蟻選擇路徑建立出一組解，依照目前費洛蒙的分布情況以及路徑本身的影響，計算出螞蟻選擇路徑的機率，依此逐步建立完成解(complete solution)。
2. Heuristic Value：表示路徑本身對螞蟻的影響，也就是路徑本身對螞蟻的吸引力，讓螞蟻可能往較佳解的方向移動。通常路徑越長，Heuristic value 的權重越低，螞蟻越不會去選擇走該路徑。
3. Pheromone Update：藉由費洛蒙的遺留及蒸發來影響螞蟻選擇該路徑的機率，通常在螞蟻走過的路徑中，較佳的解有較多的費洛蒙，較差的解費洛蒙殘留的較少。並且在一定的時間過後，路徑上的費洛蒙會有一部分蒸發掉，避免累積

過多的費洛蒙影響求解品質。

螞蟻演算法自被提出之後，經過一連串的研究，演變出其他不同的型態，保留螞蟻演算法基本的特點，但是在計算機率部分或是費洛蒙更新的部分，與最早的螞蟻演算法有所出入，以下介紹最初的螞蟻演算法及其他演變出不同型態的螞蟻演算法。

1. 螞蟻系統(Ant System, AS, Dorigo et.al, 1991)

螞蟻系統是最早提出的螞蟻演算法(Dorigo et.al, 1991)，最開始是應用在旅行推銷員問題(Traveling Salesman Problem, TSP)(Colorni et. al, 1991)，螞蟻系統將旅行推銷員問題中，推銷員的節點選擇，視為螞蟻在尋找路徑時，所經過的節點選擇，透過螞蟻從起點到終點間的節點選擇，做出解的建構，並透過費洛蒙更新進，逐步找出最短路徑。

螞蟻演算法的演算程序，是一個迴圈式的演算方式，而解的建構部分，也是重複計算不同參數，得到的完成解。以下先定義各符號的意義，再計算各路徑機率及費洛蒙更新。

符號定義：

t：第 t 次循環(iteration)。

m：每個循環 t 所使用的螞蟻數量。

η_{ij}^{r}：Heuristic Value，表示在階段 r 時，從狀態 i 到狀態 j，路徑對螞蟻的吸引力。

$\tau_{ij}(t)$：在第 t 次循環中，從狀態 i 到狀態 j 的路徑中，費洛蒙的量。

$tabu_i^k$：表示螞蟻 k 在狀態 i 時，不能前往的目的地。

$p_{ij}^k(t)$：在第 t 次循環中，螞蟻 k 從狀態 i 選擇前往狀態 j 的機率。

$f_k(t)$：第 t 次循環後，螞蟻 k 所求得順序的目標函數值。

α：螞蟻選擇路徑時，費洛蒙影響螞蟻的權重。

β：螞蟻選擇路徑時，Heuristic Value 影響螞蟻的權重。

ρ：每次循環間，費洛蒙的蒸發比率，介於 0~1 之間，值越高表示費洛蒙蒸發速度越快。

建構解：

$$p_{ij}^{k}(t)=\begin{cases}\dfrac{[\tau_{ij}(t)]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{l\notin tabu_k^i}[\tau_{il}(t)]^{\alpha}[\eta_{il}]^{\beta}} & if\ (ij)\notin tabu_k^i \\ 0 & otherwise\end{cases} \tag{3-1}$$

式(3-1)表示螞蟻 k 在第 t 次循環時，要從狀態 i 到狀態 j 的機率，其中 j 必須不在$tabu_i^k$中，也就是前面不能走過。選擇前往 j 的機率，由狀態 i 到狀態 j 路徑上的費洛蒙量及 Heuristic Value 決定，當$[\tau_{ij}(t)]^{\alpha}[\eta_{ij}]^{\beta}$越大，前往狀態 j 的機率越高。α、β 分別表示對τ_{ij}及η_{ij}的權重，α 越大，表示偏好依τ_{ij}的大小選擇路徑，β 越大，表示偏好依η_{ij}的大小選擇路徑。

費洛蒙更新：

$$\tau_{ij}(t+1)=(1-\rho)\tau_{ij}(t)+\sum_{k=1}^{m}\Delta\tau_{ij}^{k}(t) \tag{3-2}$$

$$\Delta\tau_{ij}^{k}(t)=\begin{cases}\dfrac{Q}{f^{k}(t)} & if\ ant\ k\ uses\ arc\ (ij)in\ its\ tour \\ 0 & otherwise\end{cases} \tag{3-3}$$

式(3-2)表示費洛蒙的更新，由路徑上螞蟻遺留的費洛蒙扣掉一部分蒸發量(ρ)，加上上一次循環時，螞蟻在各路徑所留下的費洛蒙。式(3-3)表示當螞蟻走過該路徑，則會留下與該螞蟻所走路徑之目標函數值呈倒數的費洛蒙，而無螞蟻經過的路段其費洛蒙為零。其中 Q 表示一常數，用以調整費洛蒙與 Heuristic Value 之間的差異。

2. 以等級為基礎的螞蟻系統(Rank-Based Version of Ant System, AS_{rank}, Bullnheimer et. al, 1996)

相較於最初的螞蟻系統，以等級為基礎的螞蟻系統在更新費洛蒙時，先將該循環 m 隻螞蟻的求解品質$f^{k}(t)$作排序$f^{rank_k}(t)$，$f^{rank_1}(t)\le f^{rank_2}(t)\le\cdots\le$

$f^{rank_m}(t)$，只有排在最前面的 σ-1 隻螞蟻才可以增加費洛蒙，並且等級越前面的螞蟻，費洛蒙增加的量越多，其中 σ 為一開始設定之參數。此外，以等級為基礎的螞蟻系統，在費洛蒙更新時，另外加上在全部的循環中，品質最佳的解(global-best)會在該路徑上再一次增加費洛蒙，$\Delta\tau_{ij}^{gb}(t)$。因此，每次循環的費洛蒙值為：

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \sum_{rank=1}^{\sigma-1}(\sigma - rank)\Delta\tau_{ij}^{k}(t) + \sigma\Delta\tau_{ij}^{gb}(t) \tag{3-4}$$

$$\Delta\tau_{ij}^{k}(t) = \begin{cases} \dfrac{Q}{f^{rank_k}(t)} & if\ ant\ rank_k\ uses\ arc\ (ij) in\ its\ tour \\ 0 & otherwise \end{cases} \tag{3-5}$$

$$\Delta\tau_{ij}^{k}(t) = \begin{cases} \dfrac{Q}{f^{gb}(t)} & if\ arc\ (ij)\ is\ part\ of\ the\ best\ solution\ found \\ 0 & otherwise \end{cases} \tag{3-6}$$

3. 蟻群系統(Ant Colony System, ACS, Dorigo et. al, 1997)

相較於最初的螞蟻系統，蟻群系統在選擇下一個狀態時，多加入了一個規則，pseudo-random-proportional rule，此規則表示螞蟻在選擇路徑時，先給定一個機率q_0，讓螞蟻可以直接選擇機率$p_{ij}^{k}(t)$最大的路徑。這樣做的目的在於，平衡探索新空間的機會及利用過去螞蟻爬過較佳路徑的機會。另外在費洛蒙更新的部分，則分為兩個部分，全域更新(global update)及區域更新(local update)。全域更新表示，在螞蟻建構的解當中，只有在全部循環中，品質最佳的解才會留下費洛蒙，其餘的費洛蒙均為 0，有助於螞蟻能夠更快速的搜尋出最佳解。而區域更新則是當每隻螞蟻建構完一組解後，該螞蟻爬過的路徑，費洛蒙均會依照一定的比率消散，這樣可以避免在同一個循環裡，螞蟻在相同的路徑上重複搜尋。

4. 近似隨機樹狀搜尋(Approximate Nondeterministic Tree-Search, ANTS, Maniezzo, 1999)

近似隨機樹狀搜尋，其特色在於費洛蒙的蒸發並不是在一次循環之後，而是依

照螞蟻搜尋出來解的好壞，決定要增加或是減少該段路徑的費洛蒙。當螞蟻搜尋出來的解較佳時，則增加費洛蒙，較差則減少費洛蒙。判斷解的好壞，則是利用式(3-7)。

$$\Delta\tau_{ij}^{k}(t)=\begin{cases}\tau_0\left(1-\dfrac{f^{k}(t)-LB}{f_{avg}-LB}\right) & if\ ant\ k\ uses\ arc\ (ij) in\ its\ tour\\ 0 & otherwise\end{cases} \tag{3-7}$$

式(3-7)中，首先要計算該題目的下限值(Lower Bound, LB)，利用減少限制式或是分割題目求解來計算下限值，若下限值的計算較佳，近似隨機樹狀搜尋便可較快求出較佳的解。而f_{avg}是由最近 h 隻螞蟻所求得解的平均值，當新的螞蟻搜尋出來的解$f^{k}(t)$較差時，$f^{k}(t)$與 LB 的差距較大，費洛蒙增加的較少，若差距超過$f_{avg}-$LB兩倍時，則會開始減少該路徑的費洛蒙。

5. 極大極小螞蟻系統(Max-Min Ant System, MMAS, Stutzle, 2000)

極大極小螞蟻系統，與蟻群系統(ACS)較為相似，差別在於，極大極小螞蟻系統會將各路徑上的費洛蒙量控制在一個範圍$[\tau_{min},\tau_{max}]$，最小不低於τ_{min}，最大不超過τ_{max}，其中τ_{min}及τ_{max}為一開始設定之參數。解的建構則是與 ACS 一樣，費洛蒙的更新也只有在每一次循環過後，品質最佳的解才會留下費洛蒙，其餘均為 0，但是更新的費洛蒙不能超過限定的範圍$[\tau_{min},\tau_{max}]$，若超過則直接以最大值或最小值為路徑上的費洛蒙。

3.2 螞蟻演算法應用於航機降落排序問題

本研究利用螞蟻系統來求解航機降落排序問題，航機降落排序問題的航機降落排序，類似旅行推銷員問題中，推銷員選擇不同節點(node)的拜訪排序，因此，本研究將航機編號視為類似旅行推銷員問題的節點，藉由螞蟻尋找節點排出航機降落順序，進行解的建構。尋找節點的過程，首先將 n 架航機的降落順序，分為 n 個階段，每個階段螞蟻所選擇的節點 i 即為降落航機之順序，每一隻螞蟻爬完一組順序即為一組航機降落順序。航機降落排序問題屬於 NP-hard 問題，在給定降落順序的情況下，亦即確定限制式之二元變數δ_{ij}，便可以在短時間內，利用線性規劃

式(2-1)到式(2-11)求解飛機實際降落時間，以及目標式值。本研究所使用之螞蟻演算法，各參數設定如下：

1. 初始費洛蒙之設定：初始費洛蒙之設定，其目的在於讓路徑一開始就有費洛蒙存在，方便計算。費洛蒙設定需要平均設在每條路徑上，並且須存在微小的量，避免影響後來螞蟻爬過的費洛蒙不明顯。本研究設定初始費洛蒙值為 0.005，約為後來螞蟻爬過路徑之費洛蒙的 1%~10%。
2. Heuristic Value 之設定：Heuristic Value 是另一個影響螞蟻選擇路徑的因素，路徑的好壞，影響 Heuristic Value 之設定。通常路徑可能越長，Heuristic Value 應該越低，螞蟻選擇的機率越低；反之，路徑越短，Heuristic Value 應該越高，螞蟻選擇的機率越高。過去旅行推銷員問題是利用節點與節點間的距離做 Heuristic Value，但在航機降落排序問題卻沒有距離做為 Heuristic Value。因此，本研究所設定之 Heuristic Value，是考慮各飛機的目標降落時間以及單位時間懲罰成本，計算出各階段的 Heuristic Value，詳細計算方法在 3.3 說明。
3. 各項參數設定：螞蟻數量 m 設定原則上應隨著飛機數量增加而成長，設定為飛機架數的一半，$m=n/2$。其他參數的設定則是利用數值實驗而得(Leguizamon, 1999)，測試方法為先固定 β=1，測試 α、ρ 的最佳值，使演算法可以收斂，且平均求解能力較佳的參數組合。確定 α、ρ 後，做 β 參數測試，調整到費洛蒙與 Heuristic Value 的影響力大小差不多且可以達到收斂的效果。數值實驗的結果，α=0.5，β=1，ρ=0.2 為比較理想的值，之後演算法的數值測試都固定此參數做測試。
4. 費洛蒙更新：費洛蒙的更新是依照各螞蟻求解品質的好壞($f_k(t)$)，也就是螞蟻所求得順序的目標函數值，做為在路徑上留下新的費洛蒙值的分母。其中，設定分子參數 Q=10，可以讓費洛蒙的值與 Heuristic Value 的差值不會太大。
5. 機率的計算：機率為螞蟻在選擇下一個狀態時的依據，首先記錄螞蟻過去階段爬過的順序於$tabu_i^k$，避免走到重複的狀態，再將剩下可以走的順序做機率的計算。最後再根據機率的不同，選擇下一個狀態。
6. 停止規則：本研究所使用停止規則，為限制循環次數 50 次，並確保每次做完都會得到收斂的順序。並且在每個階段紀錄該階段最好的順序。演算法整體演算流程如下：

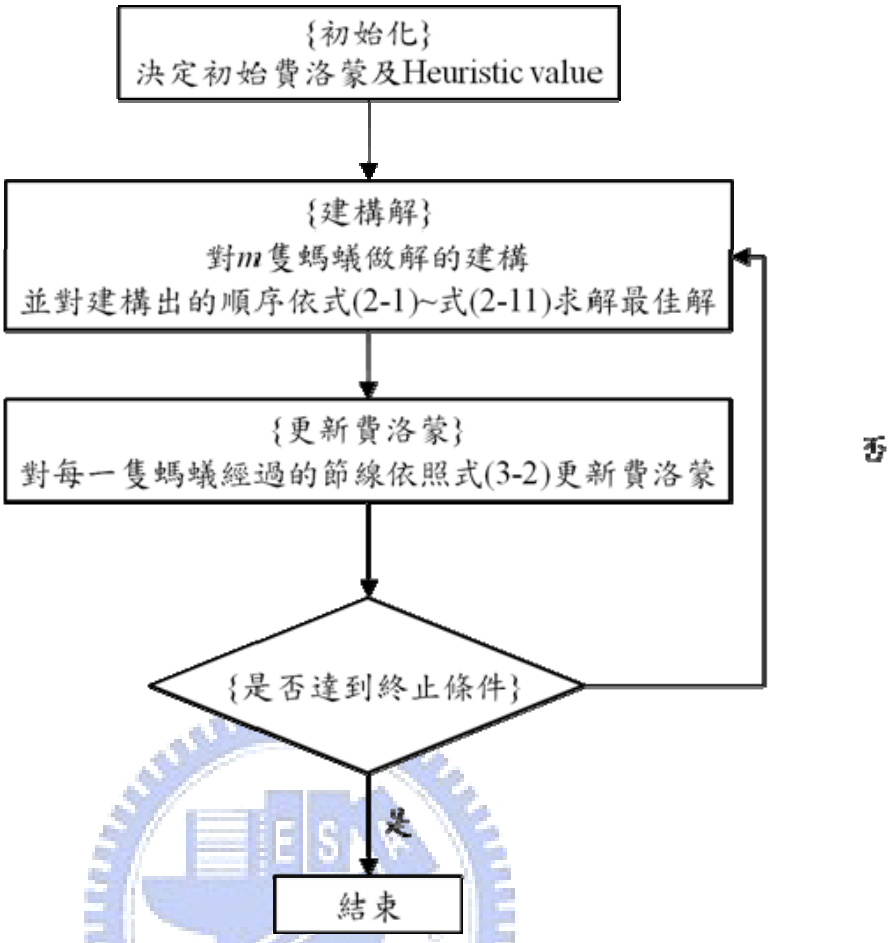


圖 3.1 螞蟻演算法流程圖

3.3 Heuristic Value之探討

本研究所探討之航機降落排序問題，雖然類似旅行推銷員問題，但是卻缺少旅行推銷員問題中，各節點之間的距離，無法用各節點之間的距離作為 Heuristic Value。因此本研究利用航機降落排序問題的特性，依照各航機目標降落時間，做出類似旅行推銷員問題中的距離，作為螞蟻選擇路徑時的 Heuristic Value。如式(3-8)所示。

$$\eta_{ij}^{k} = \frac{1}{\frac{T_j - B_i}{h_j}} \quad \forall i \neq j \tag{3-8}$$

式(3-8)中，η_{ij}^{k}表示階段 k，螞蟻在狀態 i 要前往狀態 j 時的 Heuristic Value

。T_j表示飛機 j 的目標降落時間，B_i表示飛機 i 的估計降落時間，飛機 i 原則上應以目標降落時間T_i為降落時間，但仍需考慮飛機 i 在第幾個階段降落。因為在不同階段降落，會因為前面降落的飛機，而影響飛機 i 是否能在目標時間降落。例如，飛機 i 在第 3 個階段降落，即是飛機 i 的降落順序為 3，前面已經降落了兩架飛機，因此飛機 i 降落時間的估計，必須要考慮先讓前面兩架降落的可能。因此，我們以飛機的最早降落時間為基礎，估計飛機降落時間，以式(3-9)表示。

$$B_i = Max\{T_i, E'_{r-1} + S_{ij}\} \tag{3-9}$$

式(3-9)表示飛機 i 的估計降落時間，其中將最早到達時間$E_i(i = 1..n)$由小到大排序，以數列$E'_i(i = 1..n)$表示。因此，階段 k 前一架飛機的最早降落時間，為E'_{k-1}。而飛機 i 的估計降落時間，便以E'_{k-1}加上的隔離時間(S_{ij})進行估算。

因此，Heuristic Value 表示，若飛機 j 的目標降落時間與飛機 i 的估計降落時間差距$(T_j - B_i)$較大時，即飛機 j 可以在較晚時間降落，則飛機 j 被選擇的機會較小，反之，若差距較小時，被選擇的機會則較大。

此外，h_j表示飛機 j 的單位時間懲罰值，由於測試題目中，飛機太早降落或是太晚降落，懲罰值相同，即$g_j = h_j$，因此用h_j表示單位時間懲罰值即可。加入單位時間懲罰值作為選擇下一個順序飛機時的考慮，表示若有兩架飛機的目標降落時間相同，即與前一架飛機的估計降落時間差值相同，則單位時間懲罰值較大的飛機，被選擇機會較大，單位時間懲罰值較小的飛機，被選擇機會較小。

第四章 改良式螞蟻演算法

由於螞蟻演算法在求解排序問題時，需要搜尋的搜尋空間與問題規模大小呈平方關係，因此若問題變數變多，搜尋空間亦會隨問題規模變大。導致需要更多數量的螞蟻以及更多的循環次數，來搜尋更大的搜尋空間，使得求解大型問題時的困難度上升。

螞蟻演算法在求解過程中，螞蟻在選擇飛機順序時，類似動態規劃(Dynamic Programming, DP)中各階段(stage)選擇狀態(state)的概念。因此，本研究利用動態規劃的概念，計算條件式的(conditional)下限值，也就是在限制某一架飛機降落順序的情況下，計算目標式總成本下限值，再依照該下限值組成地形(terrain)的概念，並限制螞蟻搜尋的坡度(grade)較陡的區域，也就是將坡度選擇的限制加入$tabu_i^k$中，藉此避免螞蟻搜尋到不必要的搜尋空間，減少演算法需要的螞蟻數量或是循環次數，以求解更大型的排序問題。

若有 n 架飛機，以動態規劃的概念求解飛機排序時，階段 j(j = 1,...,n)代表第 j 個降落順序，狀態以 x_j 代表在第 j 個順序降落的飛機，例如 $x_j = l$ 代表飛機 l 在 j 個順序降落。另外，以$V_j(x_j)$代表第 j 個順序降落的飛機 x_j 時，目標式總成本的下限值。

為說明條件式的成本下限值之估算方式，4.1 介紹各階段$V_j(x_j)$的基本關係及邊界值。4.2、4.3 分別介紹在排定兩架飛機順序後，加入第三架飛機於兩架飛機中間後，兩種不同的的額外成本估算法。4.4 將兩種估算法與動態規劃公式做整合，提出完整計算方式。

4.1 各階段條件式的成本下限值

定義 N 為所有飛機的集合，$\overline{N}(x_j)$表示扣除 x_j 後所有飛機的集合，也就是$N-\{x_j\}$。$\overline{N}(x_{j-1},x_j)$表示扣除 x_{j-1}、x_j 後所有飛機的集合，也就是$N-\{x_{j-1},x_j\}$。

首先界定邊界條件，$V_1(x_1)$表示當飛機 x_1 排定第 1 個順序降落會之成本下限值。飛機 x_1 後面應有$\overline{N}(x_1)$內 n-1 架飛機需要排序，在估算下限值時，假設僅考慮 x_1 與$\overline{N}(x_1)$內飛機之間排序所衍生的成本，但不考慮$\overline{N}(x_1)$內飛機之間排序的成本，以式(4-1)表示。

$$V_1(x_1) = \sum_{m \in \bar{N}(x_1)} C_{x_1,m}$$

(4-1)

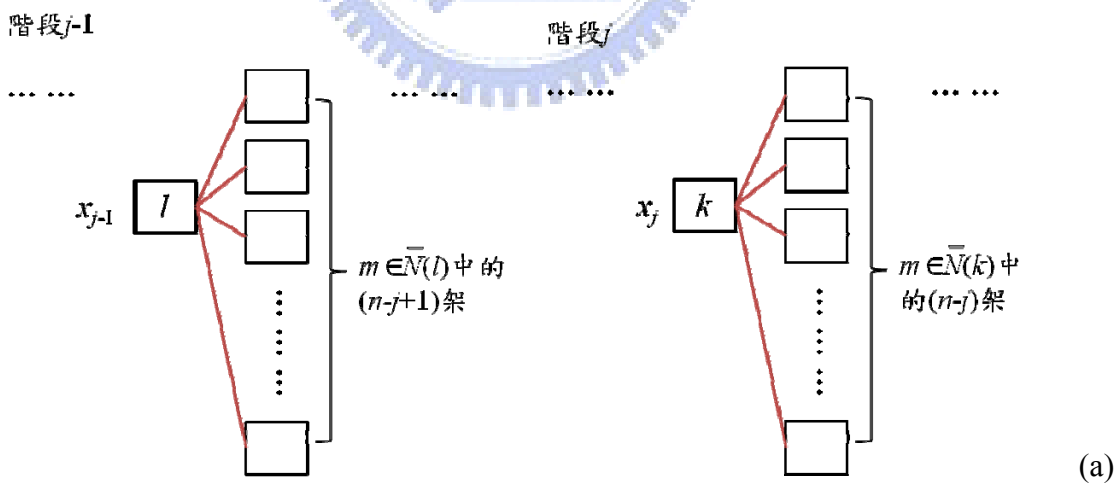
式(4-1)中，$C_{x_1,m}$表示考慮兩架飛機排定順序，x_1先降落，m 後降落，造成的成本，其詳細計算方式在 4.1.1 說明。

對任何一個階段 j，成本下限值的計算，是由上一階段(j–1)的下限值加上這一階段(j)的額外成本下限值，如式(4-2)所表示。其中f_{x_{j-1},x_j}表示從狀態x_{j-1}到狀態x_j，$V_j(x_j)$增加的額外成本。

$$V_j(x_j) = \min_{x_{j-1} \in \bar{N}(x_j)} \left\{ V_{j-1}(x_{j-1}) + f_{x_{j-1},x_j} \right\}$$

(4-2)

就$V_{j-1}(x_{j-1})$，若排定$x_{j-1} = l$，則飛機 l 後面還有(n-j+1)架飛機需要降落。下限值的估算，可以選擇$\bar{N}(l)$中，與飛機 l 成本最小的(n-j+1)架飛機，做為成本的計算，而不考慮在$\bar{N}(l)$間飛機排序的成本，如圖 4-1 所示。另外，圖 4-1 也顯示連續兩個階段(j-1)和階段 j 下限值的示意圖。



(a)

$V_{j-1}(x_{j-1} = l)$下限值　　(b) $V_j(x_j = k)$下限值

圖 4-1 條件式的成本下限值示意圖

在階段的轉換，由$x_{j-1} = l$轉換到$x_j = k$，計算其額外成本$f_{x_{j-1},x_j} = f_{l,k}$時，主要在階段 j 比階段 j-1 多排定飛機$x_j = k$，如圖 4-2。因此額外成本的計算，如同在

原本排定順序 l、m 的飛機中，多插入階段 j 所排定的飛機 k，造成原本在階段(j-1)排好的飛機，會因為飛機 k 的插入，而延遲降落產生額外成本，如圖 4-2 所示。有關f_{x_{j-1},x_j}的計算，本研究發展了兩種估算方式，分別於 4.2、4.3 詳細說明。

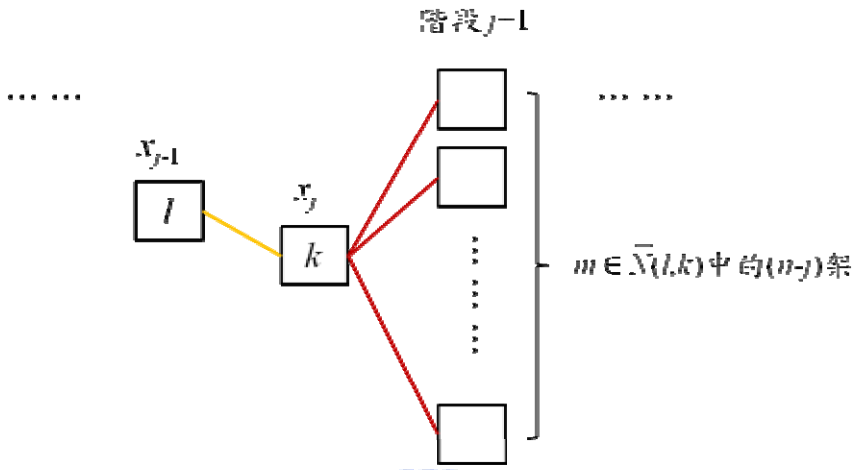


圖 4-2 插入第三架飛機額外成本示意圖

4.1.1 兩架飛機排定順序之成本

假設有兩架飛機，l 先降落，m 後降落，因為兩架飛機間必須維持隔離時間，可能導致飛機沒有辦法在目標時間降落，對目標式(2-1)產生成本。定義僅考慮兩架飛機排定順序，l 先降落，m 後降落，造成的成本為$C_{l,m}$。以a_l、a_m表示飛機 l 及飛機 m 的降落時間，其各個可能情況的成本計算如下：

1. $S_{lm} < T_m - T_l$：飛機 l 與飛機 m 的隔離時間小於兩架飛機目標時間差。因此，飛機 l 與飛機 m 皆可在目標時間降落，其成本$C_{l,m} = 0$，如圖 4-3。

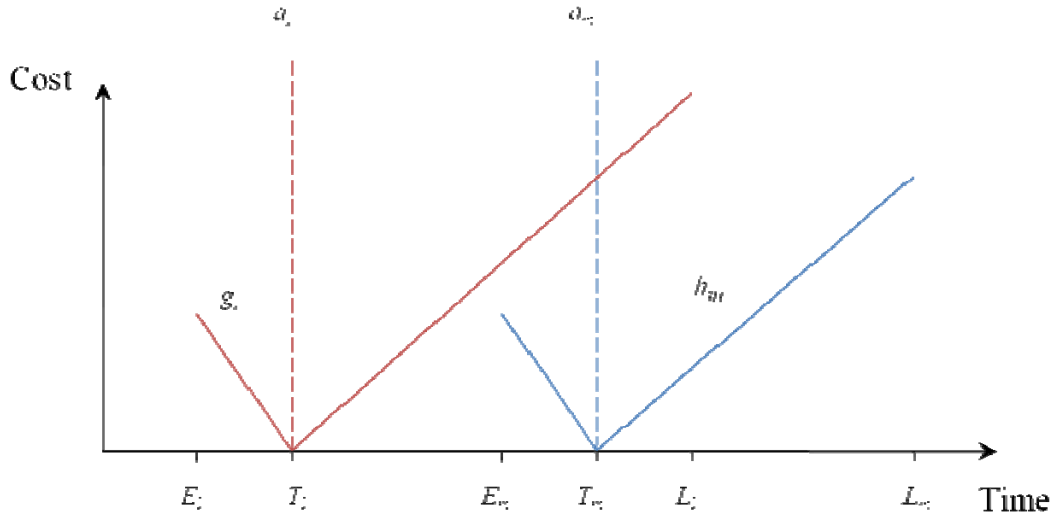


圖 4-3 兩架飛機排序圖(情況 1)

2. $S_{lm} > T_m - T_l$，$g_l > h_m$：飛機 l 與飛機 m 的隔離時間大於兩架飛機目標時間差，且由於飛機 l 提早降落成本大於飛機 m 延遲降落成本。因此，飛機 l 降落時間在目標時間，飛機 m 考慮與飛機 l 的隔離時間，必須降落在目標時間之後 $(a_m = T_l + S_{lm})$，其成本為飛機 m 降落時間超過目標時間的單位時間懲罰成本，$C_{l,m} = h_m(T_l + S_{lm} - T_m)$，如圖 4-4。

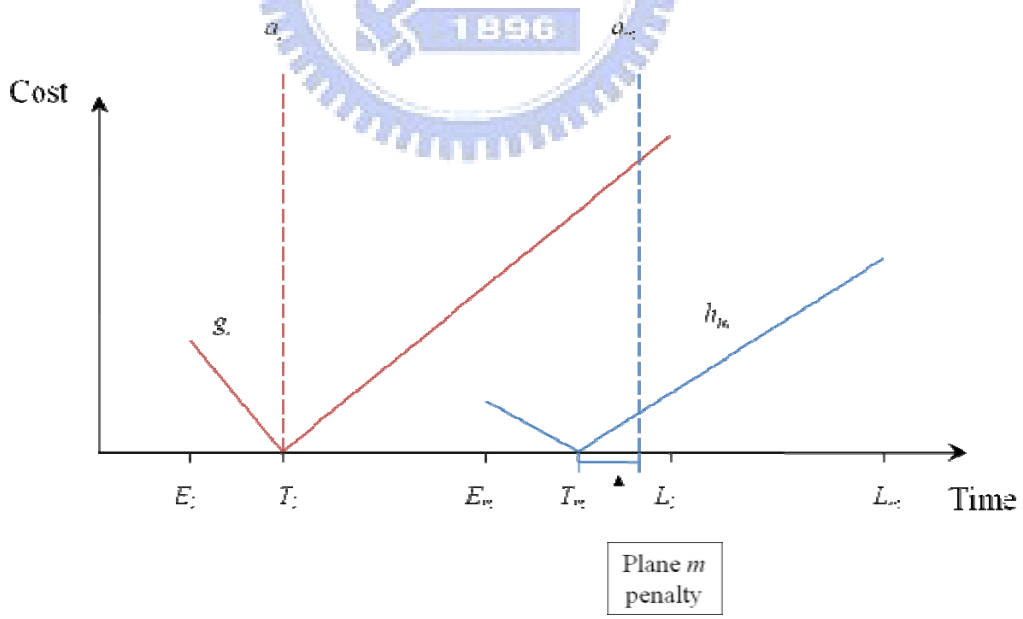


圖 4-4 兩架飛機排序圖(情況 2)

3. $S_{lm} > T_m - T_l$，$g_l < h_m$：飛機 l 與飛機 m 的隔離時間大於兩架飛機目標時間差，且由於飛機 l 提早降落成本小於飛機 m 延遲降落成本。因此，飛機 m 降落時間

在目標時間，飛機 l 考慮與飛機 m 的隔離時間，必須降落在目標時間之後 $(a_l = T_m - S_{lm})$，其成本為飛機 m 降落時間超過目標時間的單位時間懲罰成本，$C_{l,m} = g_l(T_l + S_{lm} - T_m)$，如圖 4-5。

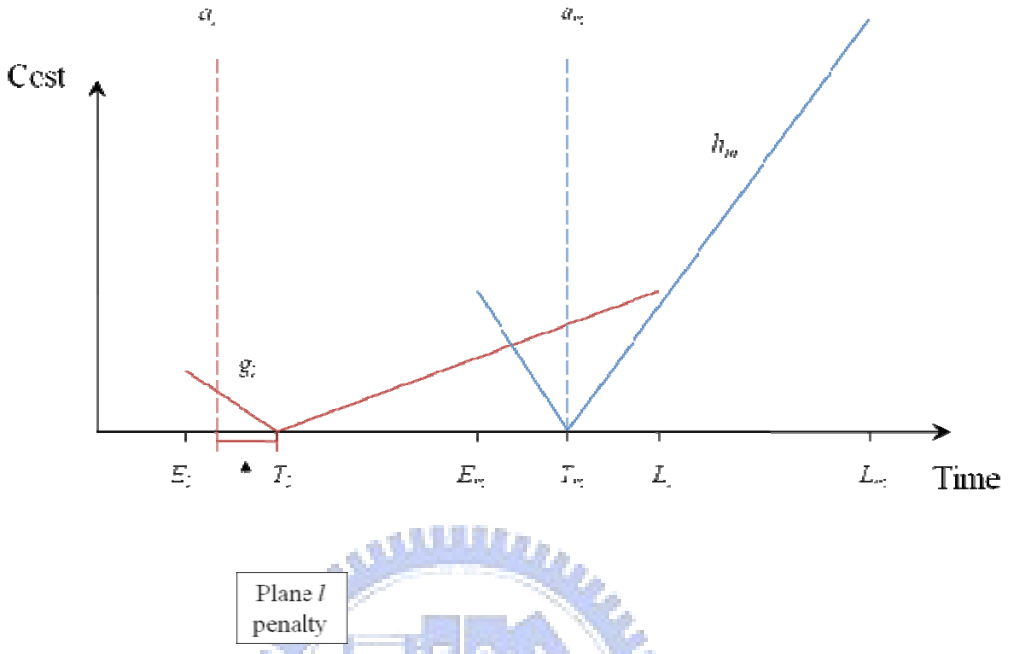


圖 4-5 兩架飛機排序圖(情況 3)

4.2 插入飛機之額外成本估算法一

為分辨插入飛機 k 前後的降落時間，我們將原本兩架飛機排定順序時的降落時間以a_l、a_m表示，後來三架飛機排定順序時的降落時間以a_l'、a_k'、a_m'表示。當飛機 l 排在飛機 m 前面，若插入飛機 k，則飛機 l 與飛機 m 之間的時間差距所增加之成本，以$W_{l,k,m}$表示。估算法一主要考慮，飛機 l、m 的隔離時間，與兩架飛機目標時間差，何者較大。當隔離時間較大時，表示飛機 l、m 的降落時間較接近，需要滿足隔離時間的條件，無法在目標時間降落。當目標時間差較大時，表示飛機 l、m 的降落時間不會受到隔離時間影響，可以在目標時間降落。另外，為確保計算為成本下限值，時間差距增加的單位時間懲罰成本，以飛機 l、k、m 中，單位時間懲罰成本最小為懲罰成本，以式(4-3)表示，其可能成本計算如下：

$$W_{l,k,m} = \left((a_m' - a_l') - (a_m - a_l)\right) min\{h_l, h_k, h_m\} \tag{4-3}$$

1. $S_{lm} > T_m - T_l$：飛機 l 與飛機 m 的隔離時間大於兩架飛機目標時間差，因此飛機 l 與飛機 m 的降落時間差距為隔離時間，$a_m - a_l = S_{lm}$。當飛機 k 插入時，飛機依 l、k、m 的順序排列，則飛機 l 與飛機 m 的降落時間差距會增加為飛機 l、k 隔離時間與飛機 k、m 隔離時間的總合，$a'_m - a'_l = S_{lk} + S_{km}$。因此，$W_{l,k,m} = \big((S_{lk} + S_{km}) - S_{lm}\big) min\{h_l, h_k, h_m\}$。

2. $S_{lm} < T_m - T_l$：飛機 l 與飛機 m 的隔離時間小於兩架飛機目標時間差，因此飛機 l 與飛機 m 的降落時間差距為目標時間差，$a_m - a_l = T_m - T_l$。當飛機 k 插入時，飛機依 l、k、m 的順序排列，則會產生兩種情況：

2a. $T_m - T_l < S_{lk} + S_{km}$：飛機 l、k 的隔離時間與飛機 k、m 的隔離時間總和大於飛機 l、m 的目標時間差值，$a'_m - a'_l = S_{lk} + S_{km}$。因此$W_{l,k,m} = \big((S_{lk} + S_{km}) - (T_m - T_l)\big) min\{h_l, h_k, h_m\}$。

2b. $T_m - T_l > S_{lk} + S_{km}$：飛機 l、k 的隔離時間與飛機 k、m 的隔離時間總和小於飛機 l、m 的目標時間差值，此情況不會因為插入飛機 k 而改變飛機 l、m 的降落時間差。但此情況必須再考慮飛機 k 與飛機 m 間的關係，若飛機 k、m 的隔離時間小於飛機 k、m 的目標時間差，$T_m - T_k > S_{km}$，表示飛機 k 及飛機 m 皆可在目標時間降落，成本為 0。若飛機 k、m 的隔離時間大於飛機 k、m 的目標時間差，$T_m - T_k < S_{km}$，表示飛機 k 或飛機 m 其中一架無法在目標時間降落，造成成本。因此成本$W_{l,k,m} = (S_{km} - (T_m - T_k))\min\{h_k, h_m\}$。

綜合以上兩架飛機 l、m 間插入第三架飛機 k 後，飛機排序為 l、k、m 後，飛機 l、m 前後時間差距所增加的成本，可以寫成式(4-4)。

$$W_{l,k,m} = Max\{Max\{(S_{lk} + S_{km} - Max\{S_{lm}, T_m - T_l\}), 0\} min\{h_l, h_k, h_m\} , Max\{(S_{km} - (T_m - T_k), 0)\} min\{h_k, h_m\}\} \quad (4\text{-}4)$$

4.3 插入飛機之額外成本估算法二

航機降落排序問題數學規劃式中，式(2-4)限制每架飛機在排序時，必須要滿足任兩架飛機間的隔離時間。然而，實務上，就已經確定的飛機排序，其實只需考慮前後相鄰飛機之隔離時間。

4.3.1 額外成本之計算

估算法二的額外成本，是指在原本已經排定好順序 l、m 的兩架飛機中間，插入第三架飛機 k，使排列變為 l、k、m，因為隔離時間的影響，單就飛機 m 的降落時間延遲，所增加的額外成本。定義插入飛機 k 後增加的額外成本之下限值為$D_{l,k,m}$，如圖 4-6 所示。

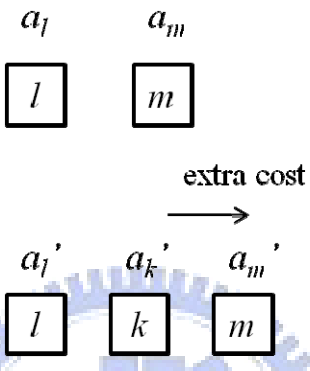


圖 4-6 額外成本示意圖

為分辨插入飛機 k 前後的降落時間，將原本兩架飛機排定順序時的降落時間以a_l、a_m表示，後來三架飛機排定順序時的降落時間以a_l'、a_k'、a_m'表示。額外成本的計算，是考慮飛機 m 前後兩個降落時間的差值，所造成的成本$h_m(a_m' - a_m)$。其中為確保飛機 m 的前後降落時間差$a_m' - a_m$為一下限值，a_m盡可能越晚越好，但不可超過最晚降落時間，a_m'盡可能越早越好，但不可超過目標時間。

為確保計算結果為額外成本之下限值，原排定順序 l、m 的兩架飛機，將飛機 l 的降落時間排在目標時間$a_l = T_l$。因此，飛機 m 降落時間a_m的計算，若可以不違反飛機 l 與飛機 m 隔離時間的情況下$(T_m - T_l > S_{lm})$，$a_m = T_m$。若違反隔離時間$(T_m - T_l < S_{lm})$，則飛機 m 降落時間$a_m = T_l + S_{lm}$。因此，綜合以上兩種情況，可以得到飛機 m 可能的最先降落時間為$a_m = Max\{T_m, T_l + S_{lm}\}$。有關$a_m'$及$D_{l,k,m}$的可能情況如下：

1. $E_l + S_{lk} < E_k$：飛機 l 在最早降落時間$a_l' = E_l$，且由於飛機 l 的最早降落時間加上飛機 l、k 的隔離時間不超過飛機 k 的最早降落時間，因此飛機 k 可以在最早降落時間降落$a_k' = E_k$。

1a. $E_k + S_{km} < T_m$：飛機 k 的降落時間加上與飛機 m 隔離時間小於飛機 m 的目標降落時間，因此，飛機 m 可以在目標時間降落$a_m{}' = T_m$。因為飛機 m 在目標時間降落，因此成本$D_{l,k,m} = 0$，如圖 4-7。

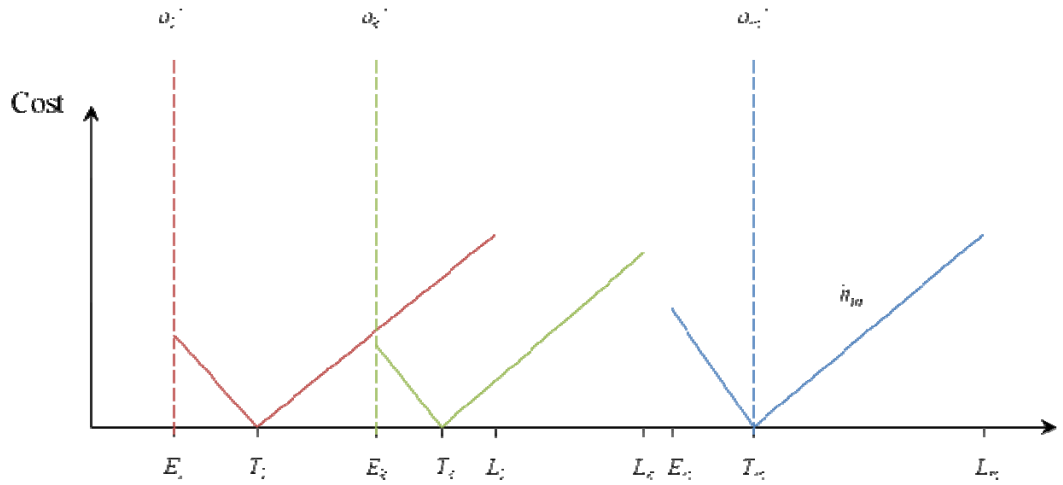


圖 4-7 三架飛機排序圖(狀況 1a)

1b. $E_k + S_{km} > T_m$：飛機 k 的降落時間加上與飛機 m 隔離時間大於飛機 m 的目標降落時間時，因此飛機 m 則必須降落在飛機 k 的降落時間加上隔離時間，$a_m{}' = E_k + S_{km}$，成本$D_{l,k,m} = h_m Max\{0, E_k + S_{km} - a_m\}$，如圖 4-8。

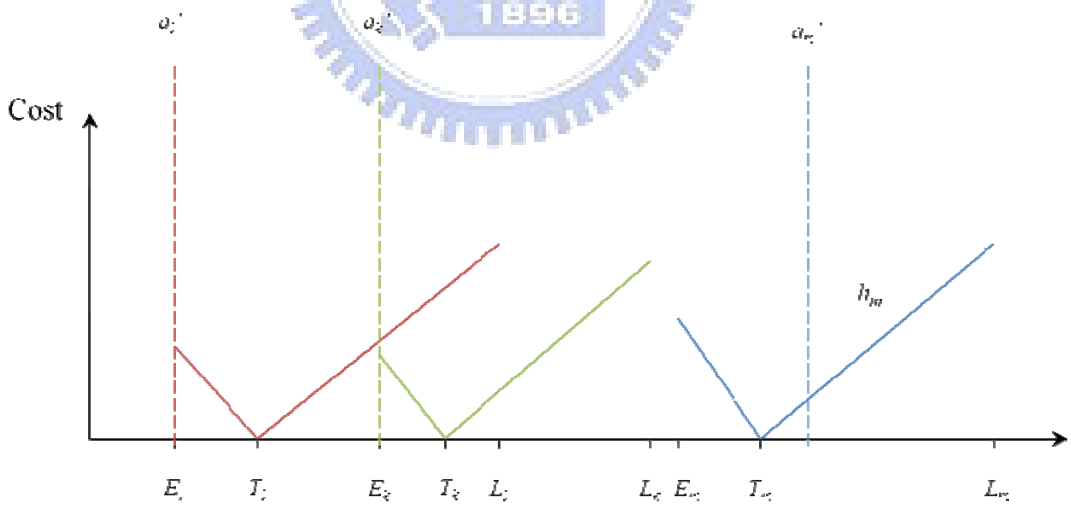


圖 4-8 三架飛機排序圖(狀況 1b)

2. $E_l + S_{lk} > E_k$：飛機 l 在最早降落時間$a_l{}' = E_l$，但由於飛機 l 的最早降落時間加上飛機 l、k 的隔離時間超過飛機 k 的最早降落時間，因此飛機 k 降落在飛機 l 的降落時間加上隔離時間$a_k{}' = E_l + S_{lk}$。

2a. $E_l + S_{lk} + S_{km} < T_m$：飛機 k 的降落時間加上與飛機 m 隔離時間小於飛機 m 的目標降落時間，因此，飛機 m 可以在目標時間降落$a_m{}' = T_m$，$D_{l,k,m} = 0$，如圖 4-9。

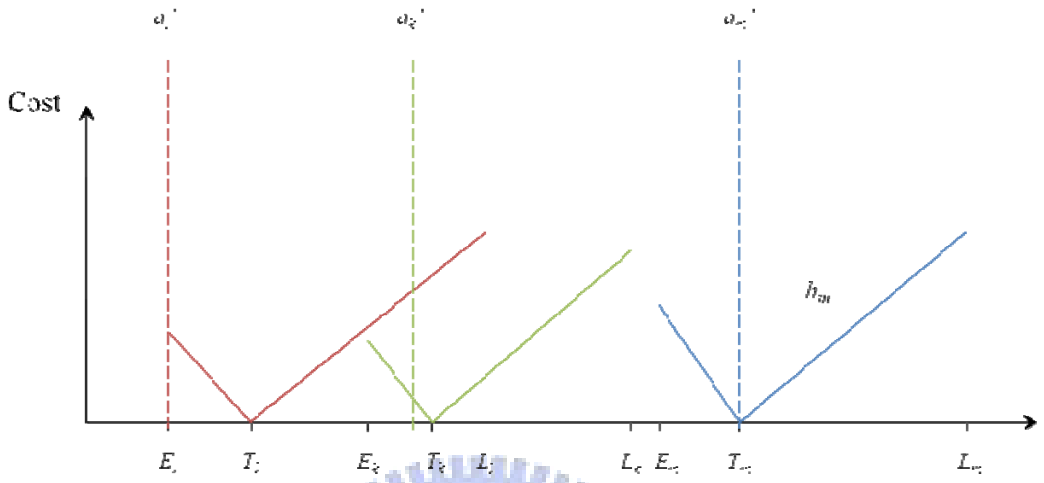


圖 4-9 三架飛機排序圖(狀況 2a)

2b. $E_l + S_{lk} + S_{km} > T_m$：由於飛機 k 的降落時間加上與飛機 m 隔離時間大於飛機 m 的目標降落時間時，因此飛機 m 則必須降落在飛機 k 降落時間加上隔離時間，$a_m{}' = E_l + S_{lk} + S_{km}$，$D_{l,k,m} = h_m Max\{0, E_l + S_{lk} + S_{km} - a_m\}$，如圖 4-10。

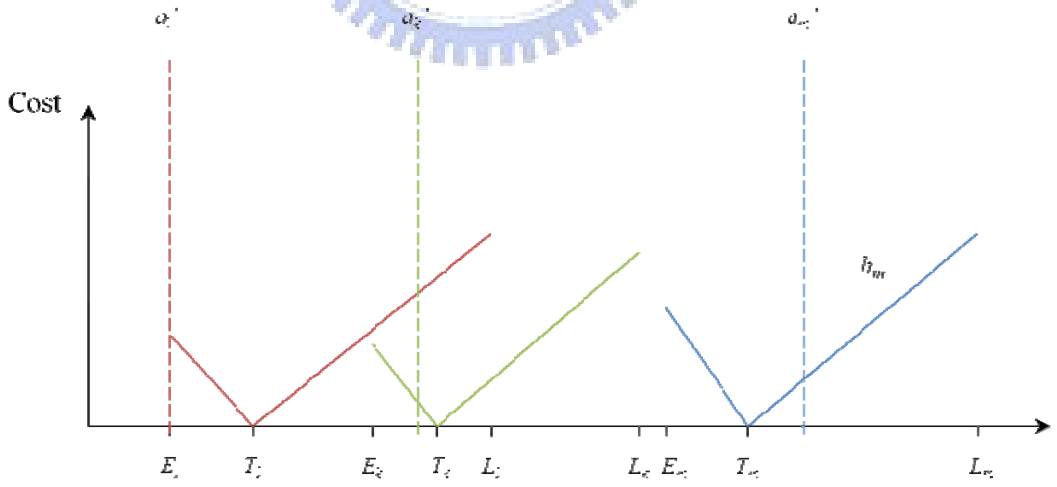


圖 4-10 三架飛機排序圖(狀況 2b)

4.3.2 額外成本下限值的改善

在計算插入第三架飛機後額外成本之下限值時，是假設飛機 l 能在最早時間降落$a_l' = E_l$，但考慮飛機 l 在第(j-1)個順序，前面已經有(j-2)架飛機先降落，因為隔離時間的影響，而可能會發生$a_l' > E_l$的情況，我們將飛機 l 的降落時間改寫成式(4-5)，以$D_{l,k,m}^{(j)}$表示加入考慮在第 j 個順序降落飛機 l 後的額外成本下限值。

$$a_l' = Max\{E_1' + (j-2)min\{S_{ij}\}, E_{j-2}' + min\{S_{ij}\}, E_l)\} \quad (4\text{-}5)$$

式(4-5)中，將最早到達時間$E_i(i = 1..n)$由小到大排序，以數列$E_i'(i = 1..n)$表示。飛機 l 的降落時間，需滿足以下兩個條件：

1. 考慮第(j-2)個最早降落時間加上隔離時間，是否會超過本身最早降落時間，飛機l的降落時間須滿足$a_l' > E_{j-2}' + min\{S_{ij}\}$，$E_{j-2}'$表示第($j$-2)小的最早降落時間，$min\{S_{ij}\}$表示隔離時間的最小值。
2. 考慮飛機 l 是第 1 個順序降落的飛機 x_1 後第(j-2)架，期間相差(j-2)個隔離時間，飛機 l 的降落時間須滿足$a_l' > E_1' + (j-2)min\{S_{ij}\}$。

4.4 總成本下限值

總額外成本下限值f_{x_{j-1},x_j}，為飛機$\overline{N}(l,k)$中，考慮估算法一及估算法二，額外成本最小的(n-j)架飛機，如式(4-6)，其中 Σ 中的 min(n-j)表示$D_{x_{j-1},x_j,m}^{(j)}$中，最小的(n-j)個相加。：

$$f_{x_{j-1},x_j} = Max\{\sum_{m\in\overline{N}(x_{j-1},x_j)}^{min(n-j)} D_{x_{j-1},x_j,m}^{(j)}, \sum_{m\in\overline{N}(x_{j-1},x_j)}^{min(n-j)} W_{x_{j-1},x_j,m}\} \quad (4\text{-}6)$$

最後，若計算出的總額外成本$f_{x_{j-1},x_j} = 0$，有可能導致成本下限值$V_j(x_j)$也為 0，多考慮加入兩架飛機排列時的成本C_{x_{j-1},x_j}做為比較。因為在計算額外成本下限值

時，是計算飛機 m 前後降落時間差，並且飛機 l 及飛機 m 的降落時間是盡可能越早越好，其中忽略了 l、m 兩架飛機排定順序後，至少會有C_{x_{j-1},x_j}的成本。因此，若計算出的成本下限值$V_j(x_j)$為 0 時，加入C_{x_{j-1},x_j}做比較，可將式(4-2)改寫為式(4-7)。

$$V_j(x_j) = \min_{x_{j-1}\in\bar{N}\{x_j\}}(Max(C_{x_{j-1},x_j}, V_{j-1}(x_{j-1}) + f_{x_{j-1},x_j})) \quad (4\text{-}7)$$

第五章 數值測試

本章節針對本研究所提出之演算法做數值測試。5.1 介紹例題數值資料，並呈現利用螞蟻演算法數值測試的結果。5.2 介紹如何將條件式的下限值導入螞蟻演算法，並呈現利用條件式的下限值導入螞蟻演算法數值測試的結果。5.3 比較螞蟻演算法與導入條件式的下限值後，數值測試的差異。5.4 探討條件式的下限值導入，在不同門檻參數限制下，螞蟻演算法的數值測試結果。

5.1 螞蟻演算法數值測試之結果

本研究所使用之數值測試，為修改Beasley(2000)在OR-Library提出的標竿範例。標竿範例中，飛機的大小分成兩種，而單位時間懲罰成本也隨飛機大小不同有所改變，並且不論早於或晚於目標降落時間降落，懲罰成本皆相同。並由於大飛機與小飛機間的隔離時間不同，分為大飛機與大飛機，大飛機與小飛機，小飛機與小飛機三種隔離時間。另外範例中，目標降落時間比較接近最早降落時間，並與最晚降落時間差距較大，顯示飛機能提早到達的時間不多，但卻能在上空盤旋較久的時間等待降落。

但是 Beasley 所提出的標竿範例，大部分例題最佳解的順序與目標降落時間的排序相同，在求解困難度方面比較沒有決策上的意義。因此，本研究藉由修改標竿範例，設計與標竿範例相似的例題，但最佳解無法依照排序目標降落時間求得。範例的修改包括，1. 將某些大飛機改為小飛機，或小飛機改為大飛機，並調整相關參數。2. 將飛機降落調整到較靠近，表示時間內需要降落的飛機增加，提高決策難度。本研究設計之例題一共 21 題，包含 10 架飛機 3 題，15 架飛機 3 題，20 架飛機 9 題，30 架飛機 3 題，40 架飛機 3 題。其中例題 1 資料如表 5-1、5-2 所示，用以呈現例題的基本特性：

表 5-1 例題 1 飛機基本參數

Plane	1	2	3	4	5	6	7	8	9	10
E_i	138	150	89	96	110	120	124	126	135	145
T_i	155	180	98	106	123	135	138	140	150	165
L_i	599	664	510	521	555	576	577	573	591	642
g_i	10	10	30	30	30	30	30	30	30	30
h_i	10	10	30	30	30	30	30	30	30	30

表5-2 例題1飛機隔離時間

S_{ij}	1	2	3	4	5	6	7	8	9	10
1	-	3	15	15	15	15	15	15	15	15
2	3	-	15	15	15	15	15	15	15	15
3	15	15	-	8	8	8	8	8	8	8
4	15	15	8	-	8	8	8	8	8	8
5	15	15	8	8	-	8	8	8	8	8
6	15	15	8	8	8	-	8	8	8	8
7	15	15	8	8	8	8	-	8	8	8
8	15	15	8	8	8	8	8	-	8	8
9	15	15	8	8	8	8	8	8	-	8
10	15	15	8	8	8	8	8	8	8	-

本研究依照第三章所設定之螞蟻演算法進行演算，每個題目依照整個流程做 10 個回合(round)，並分別記錄其中的最低值與平均值，做為演算法效能評估的指標。本研究所設計的 21 題例題，其測試結果如表(5-3)：

表5-3 螞蟻演算法數值測試結果

題號	Average of 10	Best of 10	IP-optimal
Q1	760	760 (0.00%)	760
Q2	634	540 (0.00%)	540
Q3	820	820 (0.00%)	820
Q4	1088	990 (0.00%)	990
Q5	1258	1220 (0.00%)	1220
Q6	936	920 (0.00%)	920
Q7	1257	820 (0.00%)	820
Q8	952	940 (0.00%)	940
Q9	1605	1310 (4.80%)	1250
Q10	2532	2520 (0.00%)	2520
Q11	2499	2440 (0.00%)	2440
Q12	3268	2760 (6.98%)	2580
Q13	3255	3100 (0.00%)	3100
Q14	3094	2940 (0.00%)	2940
Q15	3647	3090 (1.98%)	3030
Q16	2046	1745 (50.43%)	1160
Q17	2071	1305 (31.16%)	995
Q18	1843	980 (7.69%)	910
Q19	4315	1515 (10.99%)	1365
Q20	5418	2135 (49.82%)	1425
Q21	2745	1855 (46.06%)	1270

表 5-3 中，IP-optimal 表示該題目透過數學規劃式(2-1)到(2-11)，求得的最佳解。目前使用電腦 CPU 為 Intel Core 2 Duo 3.00GHz，2GB 記憶體，軟體為 LINGO 8.0。

然而求解 40 架飛機的例題，必須花費數十小時。要用 IP-optimal 做為與演算法比較的基礎，問題規模上限大概到 40 架飛機。Best of 10 後的百分比表示演算法求得的解與最佳解的差距，由兩個值的相減，除以最佳解計算得到。從表 5-3 可以發現，當題目飛機架數越大時，演算法所求得的解與最佳解的差距呈現越來越大的情形，可能的原因，是因為雖然螞蟻數量會隨飛機架數變大而增加，但是循環次數是固定的，因此所求的解尚未找到最佳路徑。

5.2 導入條件式的下限值螞蟻演算法數值測試之結果

導入條件式的下限值後，以地形坡度的概念，限制螞蟻搜尋的範圍。在螞蟻選擇下一階段的飛機時，若兩架飛機成本下限值之增加超過門檻參數的限制時，則將該飛機排除在選擇的可能，也就是加入$tabu_i^k$中。計算的方式，假設螞蟻在階段(j-1)選擇飛機 i，就階段 j 可選擇的飛機$\bar{N}(i)$中，當下限值的增加超過 P，則列入$tabu_i^k$，其中 P 為一個可以調整的門檻參數。如式(5-1)所示：

$$V_j(x_j) - V_{j-1}(x_{j-1} = i) > P, \forall x_j \in \bar{N}(i) \tag{5-1}$$

門檻參數的限制對變數較大的例題有較顯著的影響，因此接下來以例題 16 的 30 架飛機為例，顯示各階段之$V_j(x_j)$。例題 16 的基本參數參考附錄。利用第四章計算$V_j(x_j)$確實的數據，請參考附錄。

圖 5-1、5-2 分別以間隔為 300，計算例題 16 依照 4.2 及 4.3 所計算，估算法一及估算法二的成本下限值，做出地形劃分，以不同程度的灰階表示。將兩種估算法結合的地形圖，如圖 5-3。

其中，圖 5-1 由於估算法一在計算成本下限值時，會因為飛機 k 的目標降落時間很早，飛機 m 的目標降落時間很晚，兩架飛機不被隔離時間影響而增加額外成本，因此圖 5-1 的地形分布，右上方都是低地形的情況。而圖 5-2 則由於估算法二在計算成本下限值時，會因為飛機 l、k 的最早降落時間很晚，飛機 m 的目標降落時間很晚，因此飛機 m 不被前兩架飛機的降落時間影響，而導致額外成本不會增加，因此圖 5-1 的地形分布，右下方都是低地形的情況。但若將兩種估算法結合，則會互補缺點，形成較有效果的地形圖。

圖 5-1 例題 16 估算法一成本下限值地形劃分

圖 5-2 例題 16 估算法二成本下限值地形劃分

圖 5-3 例題 16 以 300 為間隔所繪之地形圖

加入坡度的觀念，其目的在於選擇限制螞蟻演算法在搜尋解的過程中，能夠避免搜尋到不必要的搜尋空間，減少演算法需要的循環次數。加入地形觀念後，數值測試結果如表 5-4 所示：

表 5-4 導入條件式的下限值螞蟻演算法數值測試結果

題號	Average of 10	Best of 10	IP-optimal
Q1	760	760 (0.00%)	760
Q2	584	540 (0.00%)	540
Q3	820	820 (0.00%)	820
Q4	1028	990 (0.00%)	990
Q5	1220	1220 (0.00%)	1220
Q6	920	920 (0.00%)	920
Q7	1214	820 (0.00%)	820
Q8	948	940 (0.00%)	940
Q9	1416	1250 (0.00%)	1250
Q10	2532	2520 (0.00%)	2520
Q11	2491	2440 (0.00%)	2440
Q12	3072	2630 (1.94%)	2580
Q13	3139	3100 (0.00%)	3100
Q14	3077	2940 (0.00%)	2940
Q15	3411	3030 (0.00%)	3030
Q16	1931	1160 (0.00%)	1160
Q17	1627	995 (0.00%)	995
Q18	1586	910 (0.00%)	910
Q19	3498	1455 (6.59%)	1365
Q20	3490	1515 (6.32%)	1425
Q21	2252	1330 (4.72%)	1270

表 5-4 表示加入坡度概念之後，螞蟻演算法數值測試的結果，Best of 10 後的百分比表示演算法求得的解與最佳解的差距，由兩個值的相減，除以最佳解計算得到。其中 Q12、Q19、Q20、Q21 所求的解並不是最佳解，其餘題目皆可求得最佳解，且不能求得最佳解的題目與最佳解差距甚小，有可能是除了最佳解之外第二好的解。表示導入條件式的下限值的螞蟻演算法，有不錯的求解能力。

最後，利用原始螞蟻演算法所求得的解，作為該題目的上限值 UB，當下一階段可選擇的飛機，其下限值超過該題目的上限值 UB 時，則將該飛機排除在選擇的可能，加入$tabu_i^k$中。如式(5-2)所示。加入上限值考量後，數值測試結果如表 5-5 所示。

$$V_j(x_j)\begin{cases} > UB, \text{add to } tabu_i^k \\ < UB, \text{do nothing} \end{cases} \quad \forall x_j \in \bar{N}(i) \tag{5-2}$$

表 5-5 導入條件式的下限值及考量上限值之數值測試結果

題號	Average of 10	Best of 10	IP-optimal
Q1	760	760 (0.00%)	760
Q2	584	540 (0.00%)	540
Q3	820	820 (0.00%)	820
Q4	1028	990 (0.00%)	990
Q5	1220	1220 (0.00%)	1220
Q6	920	920 (0.00%)	920
Q7	1214	820 (0.00%)	820
Q8	948	940 (0.00%)	940
Q9	1416	1250 (0.00%)	1250
Q10	2532	2520 (0.00%)	2520
Q11	2491	2440 (0.00%)	2440
Q12	3072	2580 (0.00%)	2580
Q13	3139	3100 (0.00%)	3100
Q14	3077	2940 (0.00%)	2940
Q15	3411	3030 (0.00%)	3030
Q16	1931	1160 (0.00%)	1160
Q17	1627	995 (0.00%)	995
Q18	1586	910 (0.00%)	910
Q19	3498	1455 (6.59%)	1365
Q20	3490	1515 (6.32%)	1425
Q21	2252	1330 (4.72%)	1270

5.3 比較導入條件式的下限值前後數值測試之差異

為比較條件式的下限值導入螞蟻演算法，在求解航機降落排序問題是否有幫助，本研究固定螞蟻演算法的所有參數，單純比較條件式的下限值導入前及導入後的數值測試，是否有助於螞蟻演算法找到較好的解。表 5-6 表示條件式的下限值導入前後，螞蟻演算法數值測試結果。

表 5-6 導入條件式的下限值前後螞蟻演算法數值測試結果

(a) Average of 10

Average of 10	ACO	ACO+DP	差距
Q1	760	760	0.00%
Q2	634	540	14.83%
Q3	820	820	0.00%
Q4	1088	990	9.01%
Q5	1258	1220	3.02%
Q6	936	920	1.71%
Q7	1257	1101	12.41%
Q8	952	940	1.26%
Q9	1605	1298	19.13%
Q10	2532	2520	0.47%
Q11	2499	2440	2.36%
Q12	3268	2759	15.58%
Q13	3255	3100	4.76%
Q14	3094	2940	4.98%
Q15	3647	3063	16.01%
Q16	2046	1384	32.38%
Q17	2071	1250	39.63%
Q18	1843	1026	44.36%
Q19	4315	1957	54.65%
Q20	5418	1810	66.59%
Q21	2745	1580	42.46%

(b) Best of 10

Best of 10	ACO	ACO+DP	差距
Q1	760	760	0.00%
Q2	540	540	0.00%
Q3	820	820	0.00%
Q4	990	990	0.00%
Q5	1220	1220	0.00%
Q6	920	920	0.00%
Q7	820	820	0.00%
Q8	940	940	0.00%
Q9	1310	1250	4.58%
Q10	2520	2520	0.00%
Q11	2440	2440	0.00%
Q12	2760	2630	6.52%
Q13	3100	3100	0.00%
Q14	2940	2940	0.00%
Q15	3090	3030	1.94%
Q16	1745	1160	33.52%
Q17	1305	995	23.75%
Q18	980	910	7.14%
Q19	1515	1455	3.96%
Q20	2135	1515	29.04%
Q21	1855	1330	28.30%

表 5-6 表示導入條件式的下限值前後，螞蟻演算法數值測試結果。表(a)中的差距表示最初螞蟻演算法與導入條件式的下限值螞蟻演算法間，Average of 10 的差距，計算方式由導入前後的差距除以螞蟻演算法所求得的解。表(b)中的差距表示最初螞蟻演算法與導入條件式的下限值螞蟻演算法間，Best of 10 的差距，計算方式由導入前後的差距除以螞蟻演算法所求得的解。

由差距可以發現，整體來說，導入條件式的下限值後的螞蟻演算法，其平均比原始的螞蟻演算法來的好，表示若只做一個回合，導入條件式的下限值後的螞蟻演算法可能可以得到比較好的解。相較之下，若要求得較佳的解，原始的螞蟻演算法可能要計算較多回合才能得到。而隨著題目變數變多，導入條件式的下限值前後的螞蟻演算法差距越來越大，顯示條件式的下限值地形概念，能夠有效限制螞蟻不要搜索不必要的區域，因此在循環次數相同的情況下，導入條件式的下限值後，螞蟻演算法可以求得更好的解。

5.4 門檻參數大小之比較

條件式的下限值門檻參數大小，對螞蟻的搜尋區域有不同的限制，門檻參數值越大，越接近原始沒有限制的螞蟻演算法，門檻參數值若太小，則會落於區域解。門檻參數的設定，需要可以保持螞蟻搜尋的區域，並且能夠有效限制不必要的搜尋空間。本研究嘗試使用不同的門檻參數做限制，透過執行演算法，觀察不同門檻參數影響的結果。以例題 16 為例，不同的門檻參數所造成的結果如表 5-7 所示：

表 5-7 例題 16 以不同門檻參數所計算出的解

門檻參數	round=1	2	3	4	5	6	7	8	9	10	Average	Best
200	18880	17535	17440	21115	21385	17580	17740	17755	17520	17520	18447	17440
400	1890	1680	1680	1630	1540	1680	1630	1540	1540	1600	1641	1540
600	1630	1680	1630	1745	1630	1895	1680	1890	1540	1680	1700	1540
800	1685	1745	1700	1745	1850	1745	1835	1805	1835	1745	1769	1685
1000	1835	1770	1560	1550	1770	1625	1745	1935	1745	1680	1721	1550
1200	1540	1665	1740	1735	1835	1745	1830	1680	1685	1755	1721	1540
1400	1895	2260	1755	1625	1795	1815	1745	1805	1745	1730	1817	1625
1600	1660	2490	1600	1980	1895	1850	1680	1805	2085	1835	1888	1600
1800	2045	1655	2245	2395	1630	1940	2135	1755	2230	1805	1983	1630
2000	1725	2025	1835	2935	2260	1740	1860	1835	2105	1745	2006	1725
無限制	2395	1745	1835	2060	1745	2570	1745	1805	2305	2255	2046	1745

表 5-7 設定門檻參數從 200 開始，每次增加 200 至 2000，同樣以 10 個回合做為一次求解結果，Average of 10 是指 10 個回合 best 的平均，Best of 10 是指 10 回合求解最好的值。從表 5-7 可以發現，當門檻參數設定過小時，會產生非常差的解，表示地形限制過多，螞蟻搜尋空間被限制住，無法搜尋更好的解。而門檻參數從 400 開始到 2000，求解平均值及最佳值逐漸上升，表示地形限制的效果越來越差，越來越接近沒有限制的螞蟻演算法。最後測試結果，顯示例題 16 使用門檻參數為 400，是比較好的值。而其他題目則依照相同方法測試，使用適當的門檻參數做為該題目的地形限制。

第六章 結論與建議

本研究利用改良式螞蟻演算法求解航機降落排序問題。航機降落排序問題屬於排序問題，用以解決實務上的問題，利用有效的資源，在有限時間解決問題。此問題主要在於求解，在有限制的飛機降落時窗以及飛機隔離時間內，飛機實際的降落順序以及飛機降落時間。本研究利用螞蟻演算法求解飛機降落順序，透過數學規劃式求解飛機降落時間。並加入條件式的下限值改良螞蟻演算法，求解航機降落排序，以求更有效率的解決降落排序問題，使總成本最小化。

螞蟻演算法最早是應用在旅行推銷員問題，與航機降落排序問題同屬排序問題，有諸多特性相同，因此本研究利用螞蟻演算法求解航機降落排序問題。然而，隨著航機數目變多，螞蟻演算法需要更多循環次數才能求解問題，因此，本研究提出利用動態規劃的概念，計算條件式的下限值，也就是在限制某一架飛機降落順序的情況下，計算目標式總成本下限值，再依照成本下限值繪成地形，限制螞蟻求解的範圍，使螞蟻演算法可以在循環次數不變的情況下，求得較佳的解。

本研究利用數值測試，測試螞蟻演算法是否能夠求解航機降落排序問題，並利用相同參數設計，測試導入條件式的下限值後，螞蟻演算法是否可以達到在循環次數相同的情況下，求得比初始螞蟻演算法更好的解。數值範例由本研究自行設計，與 Beasley 提出的標竿範例類似的範例。由於 Beasley 提出的標竿範例，大部分例題的最佳解，與目標降落時間順序相同，在規劃求解方面比較沒有決策意義。因此本研究自行設計一系列例題，無法依照排序目標降落時間求得最佳解，藉以測試演算法的好壞。

由數值測試結果可以得知，在相同條件下，改良式螞蟻演算法的求解品質較初始螞蟻演算法的求解品質佳。顯示加入條件式的下限值，以門檻參數限制螞蟻避免搜尋不必要的搜尋空間，對螞蟻演算法是有幫助的。在相同的循環次數下，改良式螞蟻演算法可以較快速的收斂，並且門檻參數限制，是限制不必要的搜尋空間，不會影響螞蟻搜尋到最佳解的路徑。

後續研究建議，可以探討地形門檻參數的設定，目前門檻參數是設定為一個定值，無法隨題目自動改變，因此求解效果不一定是最好。因此建議可以先設定一個定數，由改良式螞蟻演算法求得一成本上限值，再參考上限值，設定地形的門檻參數。另外，可以測試問題規模更大的問題，檢視是否在問題更大時，改良式螞蟻演算法仍然有能力限制搜尋空間，在循環次數相同的情況下，求得最佳解。並測試其他類型的排序問題，檢視改良式螞蟻演算法，是否有能力求解。

參考文獻

交通部運輸統計(2008) http://www.motc.gov.tw

2008年民用機場行業分析報告(2008) http://www.docin.com/p-689735.html

Andreussi, A., Bianco, L. and Ricciardelli, S. (1981) ” A Simulation Model for Aircraft Sequencing in the Near Terminal Area,” *European Journal of Operational Research*, *Vol.8*, 345-354.

Beasley, J.E., Krishnamoorthy, M., Sharaiha, Y. M. and Abramson, D. (2000) “Scheduling Aircraft Landings—The Static Case,” *Transportation Science, Vol.34*, 180-197.

Beasley, J.E., Sonander, J., Havelock, P. (2001) “Scheduling aircraft landings at London Heathrow using a population heuristic,” *Journal of the Operational Research Society, Vol.52,* 483-493.

Beasley, J.E., Krishnamoorthy, M., Sharaiha, Y.M. and Abramson, D. (2004) ”Displacement problem and dynamically scheduling aircraft landings,” *Journal of the Operational Research Society, Vol.55,* 54-64.

Bianco, L., Nicoletti, B. and Ricciardelli, S. (1978) ”An Algorithm for Optimal Sequencing of Aircraft in the Near Terminal Area,” *Optimization Techniques: Proceedings of the 8th IFIP Conference on Optimization Technique Wurzburg.*

Ciesielski, V. and Scerri, P. (1995) “An Approach to Scheduling Aircraft Landing Times Using Genetic Algorithms,” *honours thesis, RMIT, Department of Computer Sciences*.

Ciesielski, V. and Scerri, P. (1998) ”An Anytime Algorithm for Scheduling of Aircraft Landing Times Using Genetic Algorithms,” *Australian Journal of Intelligent Information Processing Systems, Vol.4*, 206-231

Ciesielski, V. and Scerri, P. (1998) “Real Time Genetic Scheduling of Aircraft Landing times,” *Proceedings of the 1998 IEEE International conference on Evolutionary Computation*.

Colorni, A., Dorigo, M., and Maniezzo, V.(1991) “Distributed optimization by ant colonies,” *Proceedings of ECAL'91, European Conference on Artificial Life, Elsevier Publishing, Amsterdam.*

Dorigo, M., Maniezzo, V. and Colorni, A. (1991) “The ant system: an autocatalytic

optimizing process," *Technical Report TR91-016, Politecnico di Milano.*

Krishnamoorthy, M., Abela, J. and Abramson, D. (1993) "Computing Optimal Schedules for Landing Aircraft," *The 12th National Conference of the Australia Society for Operations Research*.

Leguizamon, G., and Michalewicz, Z., (1999) "A New Version of Ant System for Subset Problem", *Evolutionary Computation, CEC 99. Proceedings of the 1999 Congress, Vol.2, 1459-1464*

Maniezzo, V., Gambardella, L.M., De Luigi F. (2004) "Ant Colony Optimization," *New Optimization Techniques in Engineering, by Onwubolu, G. C., and B. V. Babu, Springer-Verlag Berlin Heidelberg, 101-117*.

Maniezzo, V. (1999) "Exact and approximate nondeterministic tree-search procedures for the quadratic assignment problem," *INFORMS Journal of Computing 11, 358-369*.

Moser, I. and Hendtlass, T. (2007) "Solving Dynamic Single-Runway Aircraft Landing Problems with Extremal Optimisation," *Proceeding of the 2007 IEEE Symposium on Computational Intellengence in Scheduling*.

Mullen, R.J., Monekosso, D., Barman, S. and Remagnino, P. (2009) "A review of ant system," *Expert Systems with Applications, Vol36, 9608-9617*.

Pinol, H. and Beasley, J.E., (2006) "Scatter Search and Bionomic Algorithms for the aircraft landing problem," *European Journal of Operational Research*, *Vol.171*, 439-462.

Yassine, A. and Bourazza, S. (2006) "A population heuristic method used for resolving Aircraft Landing Problem," *Proceedings of the 2006 International Conference on Artificial Intelligence*, *Vol 2*.

附錄一

例題 16 飛機基本參數：

Plane	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
E_i	75	157	134	103	201	95	185	98	117	117	132	114	261	250	214	310	269	237	144	151	225	176	278	295	325	311	298	319	329	211
T_i	82	197	160	117	261	106	229	108	132	130	149	126	336	316	258	409	338	287	160	169	333	269	398	378	426	412	401	450	456	345
L_i	486	628	561	565	735	524	664	523	578	569	615	551	834	790	688	967	818	726	607	624	712	610	721	781	954	896	815	827	927	828
g_i	30	10	10	30	10	30	10	30	30	30	30	30	10	10	10	10	10	10	30	30	25	10	15	30	20	15	25	10	15	20
h_i	30	10	10	30	10	30	10	30	30	30	30	30	10	10	10	10	10	10	30	30	25	10	15	30	20	15	25	10	15	20

例題 16 飛機隔離時間：

S_{ij}	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
1	-	3	8	15	8	15	3	8	8	3	3	8	8	3	3	15	15	8	8	15	8	8	8	3	3	3	15	3	3	3
2	3	-	8	15	8	15	3	8	8	3	3	8	8	3	3	15	15	8	8	15	8	8	8	3	3	3	15	3	3	3
3	8	8	-	15	8	15	3	8	8	3	3	8	8	3	3	15	15	8	8	15	8	8	8	3	3	3	15	3	3	3
4	15	15	15	-	8	15	3	8	8	3	3	8	8	3	3	15	15	8	8	15	8	8	8	3	3	3	15	3	3	3
5	8	8	8	8	-	15	3	8	8	3	3	8	8	3	3	15	15	8	8	15	8	8	8	3	3	3	15	3	3	3
6	15	15	15	15	15	-	3	8	8	3	3	8	8	3	3	15	15	8	8	15	8	8	8	3	3	3	15	3	3	3
7	3	3	3	3	3	3	-	8	8	3	3	8	8	3	3	15	15	8	8	15	8	8	8	3	3	3	15	3	3	3
8	8	8	8	8	8	8	8	-	8	3	3	8	8	3	3	15	15	8	8	15	8	8	8	3	3	3	15	3	3	3
9	8	8	8	8	8	8	8	8	-	3	3	8	8	3	3	15	15	8	8	15	8	8	8	3	3	3	15	3	3	3
10	3	3	3	3	3	3	3	3	3	-	3	8	8	3	3	15	15	8	8	15	8	8	8	3	3	3	15	3	3	3
11	3	3	3	3	3	3	3	3	3	3	-	8	8	3	3	15	15	8	8	15	8	8	8	3	3	3	15	3	3	3
12	8	8	8	8	8	8	8	8	8	8	8	-	8	3	3	15	15	8	8	15	8	8	8	3	3	3	15	3	3	3
13	8	8	8	8	8	8	8	8	8	8	8	8	-	3	3	15	15	8	8	15	8	8	8	3	3	3	15	3	3	3
14	3	3	3	3	3	3	3	3	3	3	3	3	3	-	3	15	15	8	8	15	8	8	8	3	3	3	15	3	3	3
15	3	3	3	3	3	3	3	3	3	3	3	3	3	3	-	15	15	8	8	15	8	8	8	3	3	3	15	3	3	3
16	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	-	15	8	8	15	8	8	8	3	3	3	15	3	3	3
17	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	-	8	8	15	8	8	8	3	3	3	15	3	3	3
18	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	-	8	15	8	8	8	3	3	3	15	3	3	3
19	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	-	15	8	8	8	3	3	3	15	3	3	3
20	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	-	8	8	8	3	3	3	15	3	3	3
21	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	-	8	8	3	3	3	15	3	3	3
22	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	-	8	3	3	3	15	3	3	3
23	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	-	3	3	3	15	3	3	3
24	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	-	3	3	15	3	3	3
25	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	-	3	15	3	3	3
26	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	-	15	3	3	3
27	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	-	3	3	3
28	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	-	3	3
29	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	-	3
30	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	-

例題 16 之$V_j(x_j)$：

$V_j(x_j)$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0	2190	1020	1200	1530	1530	1530	1830	1950	2280	2740	2760	3280	3360	4000
2	15760	8160	6280	5340	4390	3600	2870	2170	1660	1210	760	330	350	350	350
3	8160	4260	2870	2300	1720	1300	940	610	470	390	330	360	450	450	690
4	2790	1290	900	330	480	480	480	780	900	1230	1890	1710	2570	2570	2950
5	32670	17030	13100	11520	10000	8570	7200	5860	4710	3620	2530	1460	740	390	330
6	1470	300	720	480	810	810	810	1110	1230	1560	2220	2040	2900	2900	3280
7	24460	14790	9310	8080	6930	5820	4770	3750	2920	2150	1400	680	330	330	330
8	1320	300	330	720	750	750	750	1050	1170	1500	2040	1980	2650	2730	3370
9	4980	3240	2520	1590	900	480	330	330	450	780	1320	1260	1930	2010	2650
10	3960	2430	1920	1200	720	360	330	330	490	830	1230	1190	1840	1920	2560
11	8040	5940	4860	3570	2520	1590	930	330	330	330	660	620	1270	1350	1990
12	3870	2310	1770	1020	510	390	360	600	760	1100	1500	1460	2110	2190	2830
13	57930	28930	24870	22540	20270	18090	15950	13830	11880	10040	8200	6380	4910	3760	2930
14	50460	28930	20900	18820	16800	14870	12980	11110	9410	7770	6150	4560	3340	2440	1830
15	33610	19810	12410	10910	9470	8120	6810	5520	4400	3340	2300	1290	650	330	330
16	86840	45920	41260	38130	35060	32080	29140	26220	23470	20830	18260	15710	13440	11490	9830
17	62150	31170	26190	23770	21410	19140	16910	14700	12660	10730	8870	7030	5470	4230	3280
18	44300	27870	17020	15180	13400	11710	10060	8430	6970	5620	4270	2940	1960	1300	930
19	12080	9500	7940	6170	4640	3380	2240	1160	590	410	480	440	1090	1170	1810
20	16620	13560	11520	9270	7260	5520	3900	2340	1290	570	330	380	1030	1110	1750
21	66120	54390	48815	43020	37420	32045	26770	21545	16745	12220	7920	6110	4670	3550	2720
22	28930	15760	14350	12690	11090	9580	8110	6660	5380	4210	3040	1960	1160	680	490
23	81980	53405	49205	44765	40430	36230	32090	27980	24125	20435	16880	14420	12330	10560	9080
24	89440	80470	72520	64360	56440	48790	41260	33790	26830	20200	13840	11630	9790	8270	7040
25	102960	76010	69850	63460	57220	51160	45180	39240	33640	28260	23060	20370	18050	16050	14130
26	89510	56600	52265	47690	43220	38885	34610	30365	26375	22550	18860	16310	14130	12270	10700
27	100975	83370	75920	68250	60775	53525	46375	39275	32600	26200	20025	17465	15275	13200	11330
28	95340	52810	49640	46220	42860	39590	36360	33150	30110	27180	24250	21410	18850	16610	14660
29	108060	71790	66795	61560	56430	51435	46500	41595	36945	32460	28110	25120	22500	20200	18190
30	61220	47290	42750	37980	33360	28920	24560	20240	16260	12500	8920	7040	5530	4340	3440

$V_j(x_j)$	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
1	4200	4260	4660	5110	5270	5610	6170	6370	6750	7050	7470	7070	7350	7590	8640
2	350	350	430	640	640	760	960	980	1200	1220	1400	1220	1340	1440	2270
3	720	720	800	1010	1010	1130	1330	1350	1570	1590	1770	1590	1710	1810	2640
4	3150	3210	3610	4060	4220	4560	5120	5320	5700	6000	6420	6020	6300	6540	7590
5	330	330	330	330	330	330	330	340	560	580	760	580	700	800	1630
6	3480	3540	3940	4390	4550	4890	5450	5650	6030	6330	6750	6350	6630	6870	7920
7	330	330	330	330	330	440	640	660	880	900	1080	900	1020	1120	1950
8	3570	3630	4030	4330	4490	4830	5390	5590	5970	6270	6690	6290	6570	6810	7860
9	2850	2910	3310	3610	3770	4110	4670	4870	5250	5550	5970	5570	5850	6090	7140
10	2760	2820	3220	3670	3830	4170	4730	4930	5310	5610	6030	5630	5910	6150	7200
11	2190	2250	2650	3100	3260	3600	4160	4360	4740	5040	5460	5060	5340	5580	6630
12	3030	3090	3490	3790	3950	4290	4850	5050	5430	5730	6150	5750	6030	6270	7320
13	2120	1370	800	570	440	330	330	330	330	330	330	330	330	330	880
14	1250	700	330	330	330	330	330	330	330	330	330	330	330	330	1080
15	330	330	330	330	330	330	350	370	590	610	790	610	730	810	1660
16	8200	6720	5420	4340	3460	2600	1760	1090	750	520	330	330	330	330	100
17	2360	1590	1000	630	460	330	330	330	330	330	330	330	330	330	860
18	590	330	330	330	330	330	330	330	370	370	540	370	490	540	1300
19	2010	2070	2470	2920	3080	3420	3830	4030	4490	4860	5280	4880	5160	5360	6090
20	1950	2010	2410	2760	3020	3240	3560	3920	4400	4800	5050	4820	5100	5135	6030
21	1920	1200	660	410	360	330	330	330	330	330	330	330	330	330	885
22	330	330	330	330	330	330	330	330	480	550	720	550	670	720	1480
23	7630	6260	5070	3975	3075	2235	1535	855	510	330	330	330	330	330	190
24	5840	4640	3520	2580	1860	1210	760	330	330	330	330	330	330	330	340
25	12420	10740	9060	7460	6040	4910	3890	2960	2050	1250	785	530	330	330	60
26	9160	7700	6420	5190	4140	3150	2360	1590	1035	645	390	330	330	330	255
27	9750	8200	6730	5440	4260	3260	2310	1510	730	460	390	330	330	330	230
28	12740	10900	9240	7870	6670	5500	4350	3270	2520	1970	1450	1010	600	330	60
29	16210	14310	12420	10700	8990	7560	6330	5115	3905	2990	2120	1415	915	420	60
30	2570	1780	1170	850	550	430	330	330	330	330	330	330	330	330	670