國 立 交 通 大 學

運輸科技與管理學系碩士班

碩 士 論 文

改良式GRASP演算法求解動態連續船席調配問題

Improved Greedy Randomized Adaptive Search Procedure for Dynamic Berth Allocation Problem

研 究 生：林岱暘

指導教授：黃明居　副教授

中 華 民 國 一 百 年 七 月

改良式 GRASP 演算法求解動態連續船席調配問題

Improved Greedy Randomized Adaptive Search Procedure for Dynamic Berth Allocation Problem

研 究 生:林岱暘　　Student：Dai-Yang -Lin

指導教授:黃明居　　Advisor：Ming-Jiu-HWang

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

June 2011

Hsinchu, Taiwan, Republic of China

中華民國一百年六月

改良式GRASP演算法求解動態連續船席調配問題

學生:林岱暘　　指導教授:黃明居

摘要

本研究探討目標為最小化等待時間加上處理時間的動態連續船席調配問題，並提出兩個構想以改良過去文獻提出之GRASP演算法，而改良式GRASP演算法和過去文獻提出的GRASP演算法在相同條件之下進行大規模和小規模問題的範例驗證後，發現改良式GRASP演算法能在更短的時間內求得目標值更低的解，且在調配兩百艘船之大規模問題時，更有高達13.65%的改善。

關鍵字:GRASP、動態連續船席調配問題、貪婪隨機適應性搜尋

Improved Greedy Randomized Adaptive Search Procedure for Dynamic Berth Allocation Problem

Student：Dai-Yang-Lin　　Advisor：Dr. Ming-Jiu-HWang

Department of Transportation Technology and Management

National Chiao Tung University

ABSTRACT

In this research, Dynamic Berth Allocation Problem of Continuous berth (BAPC) is studied to minimize the weighted processing time and waiting time. Two ideas to improve the GRASP algorithm which proposed by previous literature are addressed, and the modified GRASP algorithm could find better solution than the original in same conditions within less computing time, and when allocate the large scale problem 200 vessels, the improved algorithm could get 13.65% improvement.

Key words : GRASP, Dynamic Berth Allocation Problem, Greedy Randomized Adaptive Search Procedure

誌謝

這兩年來能走到這一步要感謝很多人，第一個要感謝的就是我爸媽，在我求學過程中一路支持我，就算我成績在高中大學研究所都是全班倒數，可是爸媽還是一直很肯定我的潛力，再來要感謝交通大學運管系的全體老師，使我在這短短兩年添加了許多的知識。其中特別要感謝我的指導教授黃明居老師，老師選擇了一個很適合我個性的方式來指導我，並在論文寫作的過程中給我很多我需要的資源。這本論文的完成，除了自己的努力之外，還有老師在背後所無法衡量的奉獻。

再來，還要感謝兩位口試委員黃寬丞老師與黃家耀老師，在他們的寶貴評論與建議之下，使我的論文能夠更佳的完善且充實，特別感謝黃寬丞老師對於這篇研究在發表潛力上的肯定。

再來感謝小豬、阿胖、NoNo、凱開、QQ、徐莎、鱷魚、阿勇、維尼、阿志太、一姐、阿尼基、小惠、黛西、阿幹、呂璇、老頭、佩慈、cool fish、陳嘉佩、kiwi、蔡季軒還有其他族繁不及備載的所上的學長姊弟妹同學們，因為有大家讓這個所總是充滿輕鬆愉快的搞笑氣氛。再來要感謝我的女朋友 ASHIRA，在碩二的時候幾乎每天陪著我，也和我分享了許多對未來的想法，這兩年的研究所生活可以說是我學生生涯中最快樂一個階段，很高興在這樣的氣氛之下畫下求學生涯的句點。

這篇論文很大的一部分都和新加坡國立大學的李德紘教授過去提出的文獻有關，也感謝李教授在我文獻回顧的過程中多次和我以 E-mail 討論我有問題地方。

我總是會受到別人直接或間接的幫助，一路走來有數不清的貴人幫助過我，要感謝的人不勝枚舉很多很多，你知道我在說你，快來對號入座吧!!

林岱暘 謹誌

2011 年 7 月於新竹交大

目錄

圖目錄

表目錄

第一章 緒論

1.1 研究動機與目的

船隻於靠港進行裝卸貨時必須面臨的第一道作業程序就是船席調配，指的是如何將進港船隻指派到其適合的位置，使整體裝卸作業效率效用最佳化。其結果將影響航商的作業成本，以及碼頭整體的營運效率。

動態連船席調配問題 (Dynamic Berth Allocation Problem of Continuous Berth，DBAPC) 船隻為先後動態到達碼頭，必須在到達之後才開始裝卸貨作業，且碼頭形式為連續形船席，較新型的港口通常採用連續形的船席，固本研究選擇求解動態連船席調配問題。

Lim *et al.*[1] 證明船席調配問題 (Berth Allocation Problem，BAP) 為 NP-HARD 問題，求解運算時間會隨著問題規模增加而呈指數性的成長，無法在合理的時間內求得最佳解，所以過去有許多學者使用不同的演算法求解 BAP，例如基因遺傳演算法(Genetic Algorithm ，GA)、貪婪隨機自適應搜尋法(Greedy Randomized Adaptive Search，GRASP)、模擬退火法 (Simulated Annealing，SA)、隨機柱型搜尋法 (Stochastic Beam Search，SBS)、禁忌搜尋法 (Tabu Search，TS)…等。其中Lee *et al.*[2]，提出一將貪心法 (Greedy) 改良並加入多重起點特性以及適應性參數的GRASP (Greedy Randomized Adaptive Search Procedure) 演算法求解連續型動態船席調配問題。其求解時間非常快速，但在小規模的求解(十艘船的分配)，和CLPEX求出的近似最佳解有大於百分之七的差距，可見在解品質和求解速度上還有很大的進步空間。

本研究將針對 Lee *et al.* [2] 提出之 GRASP 演算法發展改良構想，期望能更有效率的求解 DBAPC。

本研究目的歸納如下:

1. 探討船席調配演算法的求解概念，並提出之步驟改良構想。
2. 開發出一套動態船席調配演算法，和過去之演算法相比較，進行多個範例測試，確立本研究發展的求解機制之效能。
3. 分析改良之演算法在多種情境下之有效性
4. 提出改良式演算法之缺點及可改進的地方，以供後續學者延伸本研究。

1.2 研究內容與方法

本研究探討的船席調配問題是當船隻在到達港口時，該如何安排其最適當的停靠位置，使得所有船隻在經分配過後的等待時間加上處理時間為最小。
須達到 1.1 節所以提之研究目的，預計相關研究工作及內容如下:

1. 回顧過去相關船席調配文獻
2. 歸納船席調配問題類型
3. 了解 DBAPC 之數學模式及限制條件
4. 研習 Lee *et al.* [2]提出之 GRASP 演算法
5. 以 python 程式語言撰寫 GRASP 演算法
6. 思考並提出針對該演算法之改良步驟
7. 以 python 程式語言撰寫改良之演算法
8. 以範例驗證改良之演算法效能
9. 將改良之演算法放入不同情境測試其有效性
10. 提出結論與建議

1.3 研究範圍

Imai *et al.* [3] 將船隻依到達時間將 BAP 分為靜態船席調配問題 (Static Berth Allocation Problem) 及動態船席調配問題 (Dynamic Berth Allocation Problem) ，前者為方便學術研究討論而假設出的狀況，其假設所有船都在開始船席調配之前到達，即可由調度人員自由安排船隻的時空位置，以達到整體系統最佳化 Lee *et al.* [2]，但實務上船隻預定到達時間並不相同，所以與現實情況不符。現今的新式的港口多數屬於連續型船席，且船隻到達時間先後不同較符合現實情況，所以本研究將著重在處理動態連船席調配問題 (Dynamic Berth Allocation Problem of Continuous Berth，DBAPC)。

Imai *et al.* [4] 提出若要達成系統最佳化，不該依照船隻先到先服務原則，所以本研究發展之演算法也不依照船隻先到先服務原則。

本研究討論範圍歸納如以下:

1. 不考慮先到先服務原則
2. 船隻到達前可先得知其 ETA，即船隻為動態到達。
3. 不考慮吃水深及其他技術問題。
4. 船隻裝卸貨處理時間為每艘船之固定參數，處理時間不隨停靠位置和貨櫃堆積場間的距離而變。
5. 探討問題為連續型船席
6. 船隻到港後移動到其被指派的船席位置所需的時間將併入處理時間內
7. 相鄰船隻之間需要的合理距離，併入船的長度中。

1.4 研究流程

本研究流程分為四個階段分項描述如下:

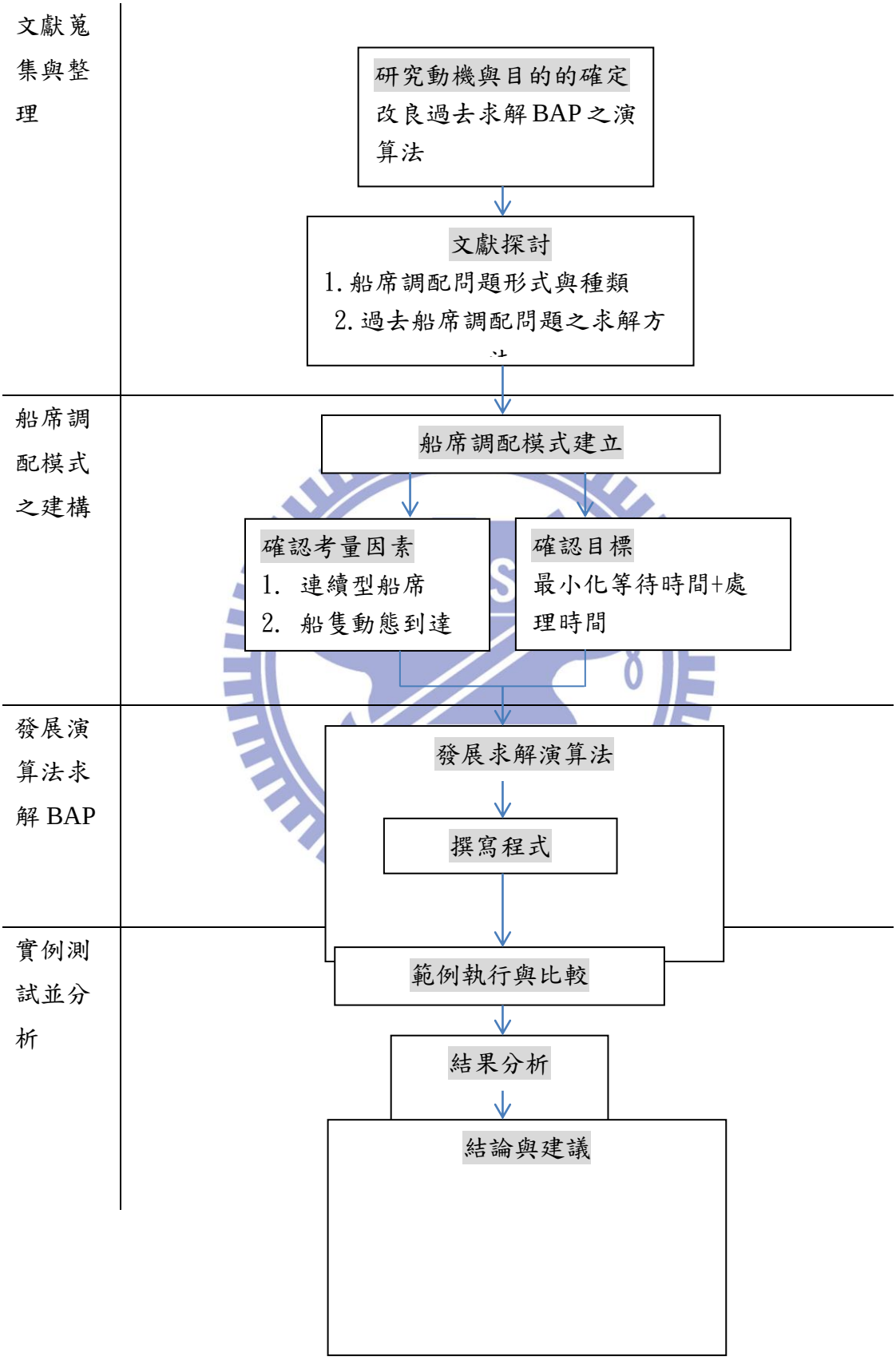
1. 文獻蒐集與整理
   (1) 確定研究動機及目的，明確定義出研究主題。
   (2) 整理船席調配問題形式並確認欲探討之問題類型，以確定研究範圍。
   (3) 整理船席調配過去求解方法，以了解各演算法之優缺點。
   (4) 選擇文獻中一較新較好之演算法。

2. 船席調配模式之建構
   (1) 確認該文獻考量因素與相關限制以及目標。

3. 發展演算法求解連續動態船席調配問題
   (1) 將從文獻研習之演算法改良，並預期求解結果。
   (2) 撰寫程式以驗證改良之效果

4. 實例測試並分析
   (1) 範例執行
   (2) 結果分析
   (3) 結論與建議

表 1-1 研究流程圖


文獻蒐集與整理
研究動機與目的的確定
改良過去求解 BAP 之演算法
文獻探討
1. 船席調配問題形式與種類
2. 過去船席調配問題之求解方
船席調配模式之建構
船席調配模式建立
確認考量因素
1. 連續型船席
2. 船隻動態到達
確認目標
最小化等待時間+處理時間
發展演算法求解 BAP
發展求解演算法
撰寫程式
實例測試並分析
範例執行與比較
結果分析
結論與建議

1.5 專有名詞解釋

1. 預計到達時間(Estimated Time of Arrival，ETA): 船隻預定到港的時間，在此時間點之後，才可靠港作業。
2. 權重(Weight): 不同的船依其與碼頭經營者的契約方式享有不同的優先權。
3. 先到先服務原則(First-Come -First-Served，FIFS): 先到達的船，先進行裝卸作業。
4. 處理時間: 船隻開始裝卸作業到離港所花費時間。
5. 等待時間: 船隻到港後到其開始作業前的閒置時間。

第二章 文獻回顧

本章整理過去對於連續型船席調配問題的相關研究，包含求解方法及特殊貢獻。本章節將分成五個小節探討，DBAPC 模式、船席調配問題相關文獻回顧、Greedy Randomized Adaptive Search (GRASP)演算法之介紹與文獻回顧小結。

2.1 船席調配問題(BAPC)

船席調配問題依空間上可分為連續型船席及離散型船席，依船隻達時間上可分為動態船席調配問題及靜態船席調配問題。其特點分述如下:

1. 連續型船席(Continuous Berth):
   碼頭邊緣供船停靠以進行裝卸作業的空間，船隻停靠在船席的任何位置以便有效率的利用船席空間。

2. 離散型船席(Discrete Berth):
   限定一個船席在同個一時間只能服務一艘船。

3. 靜態船席調配問題(Static berth Allocation Problem，SBAP):
   假設所有船隻在船席調配作業開時前就已到達，調度人員可以自由決定每艘船在何時開始裝卸貨，但此一狀況為假設情形，為方便學術研究之用，與現實狀況不符。

4. 動態船席調配問題(Dynamic Berth Allocation Problem，DBAP):
   船隻為先後動態到達，必須在到達之後才開始裝卸貨作業，此情形與現實狀況較相符。

5. 動態連船席調配問題(Dynamic Berth Allocation Problem of Continuous Berth，DBAPC):
   船隻為先後動態到達，必須在到達之後才開始裝卸貨作業，且碼頭形式為連續形船席，較新型的港口通常採用連續形的船席。

由於連續型船席為近年來新式港口的建設方式，且船隻動態到達與現實情況相符，故本研究針對動態連續型船席調配問題進行相關的文獻探討。

2.1.1 DBAPC 模式

過去相關文獻提到許多 DBAPC 模式，其中考慮狀況最單純之經典模式由 Guan and Cheung[5] 所提出，此模式提出後至 2010 年仍有學者被用來發展船席調配求解演算法。這個模式裡，船隻與船隻間所需的基本安全距離，已經被包含在船隻長度中。

以下為該模式使用之參數介紹:

S :連續船席長度

T :船席在當次船席調配開放服務的時間長度

N :進來的船的編號

p_i :船 i 的處理時間

s_i :船 i 的大小,已考慮近船與船間需要的間隔距離

a_i :船 i 的到達時間點

w_i :船 i 的加權，由碼頭經營者決定，如優先權等因素

以下為該模式之決策變數:

u_i :船 i 的開始占用船席的時間點

v_i :船 i 被配指派到港邊進行作業的位置，以船左邊邊緣代表

c_i :船 i 的離開時間

σ_{ij} :若船 i 完全在船 j 左邊記為 1，否則為 0

δ_{ij} :若船 i 完全在船 j 的下面記為 1，否則為

本 DBAPC 問題可以圖 4 之矩形時空圖描述，圖 4 已以有三艘船相繼進行裝卸作業，在這個例子中，$\sigma_{12} = \sigma_{21} = \sigma_{31} = \sigma_{32} = 0$，$\sigma_{13} = \sigma_{23} = 1$，$\delta_{13} = \delta_{21} = \delta_{23} = \delta_{31} = \delta_{32} = 0$，$\delta_{12} = 1$。

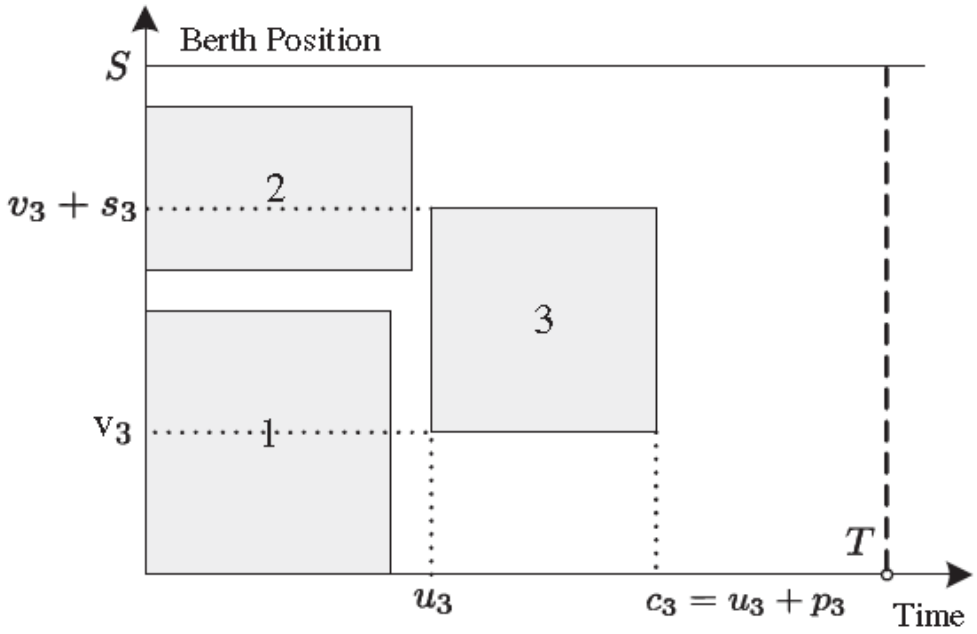


圖 2-1 矩形時空圖

以下為 DBAPC 之數學模式:

$$\min \quad \sum_{i=1}^{N} w_i(c_i - a_i) \tag{1}$$

s.t. $u_j - u_i - p_i - (\sigma_{ij} - 1) \cdot T \geq 0$， $\forall 1 \leq i$，j ≤ N，i≠j (2)

$v_j - v_i - s_i - (\delta_{ij} - 1) \cdot S \geq 0$， $\forall 1 \leq i$，j ≤ N，i≠j (3)

$\sigma_{ij} + \sigma_{ji} + \delta_{ij} + \delta_{ji} \geq 1$， $\forall 1 \leq i$，j ≤ N，i≠j (4)

$\sigma_{ij} + \sigma_{ji} \leq 1$， $\forall 1 \leq i$，j ≤ N，i≠j (5)

$\delta_{ij} + \delta_{ji} \leq 1$， $\forall 1 \leq i$，j ≤ N，i≠j (6)

$p_i + u_i = c_i$， $\forall 1 \leq i \leq$ N (7)

$a_i \leq u_i \leq (T - p_i)$，$0 \leq v_i \leq (S - s_i)$， $u_i, v_i \in \Re^+$ $\forall 1 \leq i \leq$ N (8)

$\sigma_{ij} \in \{0, 1\}$， $\delta_{ij} \in \{0, 1\}$，$\forall 1 \leq i$，j ≤ N，i≠j (9)

就(1)~(9)式，其意義如下:

(1) 最小化加權的處理時間: (離開時間點-到達時間點)加權。
(2) 為了讓船的作業時間不要在 berth 空間有交疊之下交疊。
(3) 為了讓船佔用的船席空間不要在作業時間有交疊之下交疊。
(4) i 在 j 左邊，j 在 i 左邊，i 在 j 下面，j 在 i 下面，四種情況一定要發生一種以上。
(5) i 在 j 左邊，j 在 i 左邊，兩種情況只能發生一種。
(6) i 在 j 下面，j 在 i 下面，兩種情況只能發生一種。
(7) i 開始作業的時間點+作業時間長度=離開的時間點。
(8) i 到達時間 ≦ i 開始作業的時間 ≦ 總規劃時間- i 的處理時間。
(9) 0 ≦ 船 i 的v_i ≦ 港邊的長度-船 i 的大小。

2.1.2 連續型船席調配問題(BAPC)相關文獻回顧

Lim *et al.* [1] 將連續船席調配問題轉換成二維座標系上的矩形時空圖，證明連續船席調配問題為 NP-HARD 及提出一個啟發式方法求解。其目標式為最小化船隻之間的空隙，使港口一樣長度下可以服務更多艘船。下圖是船隻在二維平面的示意圖。一個矩形的寬度代表船隻占用船席的時間，長度代表船隻占用船席的空間。

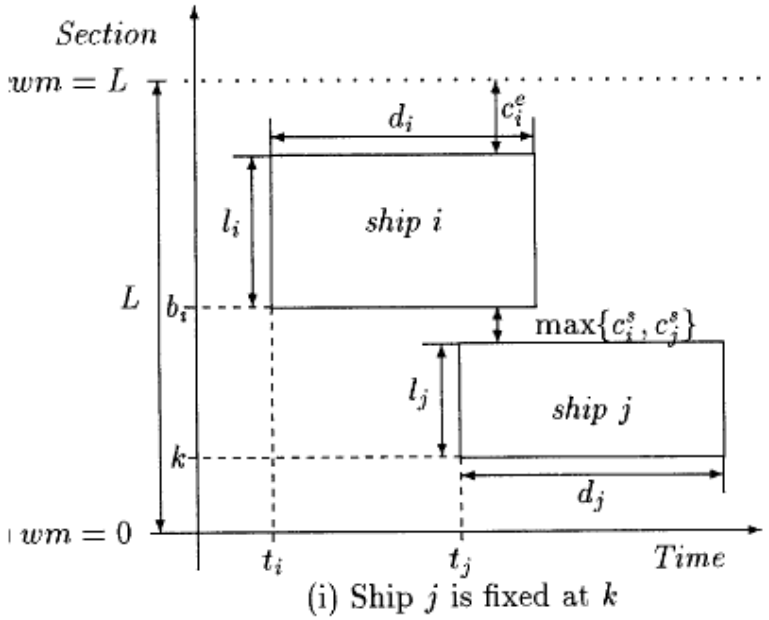


圖 2-2 連續船席調配問題時空矩型示意圖 資料來源: Lim, A. (1998)

Kim and Moon [6] 求解如何減少船隻靠岸作業時間，及盡量使船停靠在其在碼頭的最佳作業位置。目標式包含船隻無法在最佳的碼頭位置進行裝卸作業產生的成本，另一成本是船隻無法在預定時間內離開而增加的成本。採用模擬退火法(Simulated Annealing)與 Lindo 套裝軟體求解並比較。當船隻數量增加時，Lindo 的求解時間爆增。而模擬退火法則能在可接受求解時間內能求得近似最佳解。

Guan and Cheung [5] 假設船隻作業時間與船隻大小成正比，先將連續型碼頭空間與時間網格化，再利用搜尋樹 (Tree Search Procedure)搭配兩種啟發式解法求解。一是減少節點的方式減少無效的搜尋空間。另一種是利用成對交換(Pair-Wise Exchange)搜尋鄰近解以觀察解是否改善。此求解法適合小規模的問題，

若當船隻數量增加時解的品質有降低趨勢。

Lim *et al.* [3]將離散型船席調配建構初始解，並利用優先權與空間窗(window of the quay space)概念發展連續型船席調配方法以改善初始解。利用離散型求得的解，依船隻預定進港作業時間先後多寡給予遞增的作業順序。並調換空間窗內船隻的作業位置，當空間窗內船隻找到最佳作業位置則停止。若求得的解在連續空間不可行，則提前或延後該船的作業時間再重複使用上述步驟，直到獲得可行解。

Wang and Lim[7] 將船席調配問題轉換成多階段的決策程序並以隨機柱型搜尋(Stochastic Beam Search，SBS)演算法求解，並以真實新加坡港的船隻為輸入資料並與及傳統柱狀搜尋(Beam Search)進行比較，能在較短的時間內，得到較佳的解。

Lee *et al.* [2] 以貪婪隨機自適應搜尋法(Greedy Randomized Adaptive Search，GRASP)求解動態連續船席調配問題，並提出一套搜尋方式可以在矩形時空圖(圖2-1)中快速地尋找出可能停靠的位置，並以該方法建構 GRASP 的初始解，再利用區域搜尋(Local search)，尋找一群可行解，並從中挑選品質最佳的解，如此重複這兩個接，直到滿足停止條件。代入範例驗證後並和 Wang and Lim (2007) 提出的 SBS 相比，GRASP 演算法求得近似的解且運算時間極短。

Cordeau *et al.* [8] 使用兩種版本的禁忌搜尋法(Tabu Search)分別求解連續及分離的船席調配問題，並在數學模式加入時間窗的考量，以 CLPEX 軟體計算最佳解後並比較結果，求解離散型問題的 Tabu Search 只能在問題屬於小規模(25 艘船，5 個船席)時才能求得最佳解，而求解連續型船席的 Tabu Search，若在不遵守 FCFS 原則下，可以比遵照 FCFS 時的目標值有 8% 的改善。

表 2-1 文獻回顧總整理

年份	作者	目標式	求解方法
1998	Lim, A.	最小化船隻之間的縫隙	圖形概念的啟發式解法
2003	Kim, K. H. and K. C. Moon	最小化偏移最佳停靠位置的成本以及無法在預定時間離開的成本	模擬退火法
2005	Imai, A., X. Sun, et al.	最小化等待時間加上處理時間	反覆交換空間窗內的船隻直到獲得可行解
2005	Cordeau, J. F., G. Laporte, et al.	最小化等待時間加上處理時間	禁忌搜尋法
2005	Guan, Y. and R. K. Cheung	最小化等待時間加上處理時間	利用減少節點及成對交換概念的啟發式解法
2007	Wang, F. and A. Lim	最小化偏移最佳停靠位置的成本以及加上拒絕船隻停靠的成本	隨機柱型搜尋法
2010	Lee, D. H., J. H. Chen, et al.	最小化等待時間加上處理時間	貪婪隨機自適應搜尋法

2.2 貪婪隨機自適應搜尋法(GRASP)

Feo and Resende[9] 提出貪婪隨機自適應搜尋法 (Greedy Randomized Adaptive Search Procedure，GRASP)，此演算法特別適合處理組合最佳化問題，其過程主要分成兩個階段，建構階段(Construction phrase)及區域搜尋階段(Local search phrase)。

建構階段(Construction phrase)為將每個待選擇的元素排序成一個待處理的列表，並從頭開始一一加入逐漸建構成一個初始的可行解，選擇每個加入元素時時利用一個衡量函數(Greedy evaluation function)來為準則來決定選取方式。

區域搜尋階段(Local search phrase)將第一階段所產生的初始解利用一擾動(perturbation)方式，去尋找其解空間中的其他鄰居解。如此反覆這兩個階段，直到達成停止條件為止。下圖 2-3 為 GRASP 的偽代碼(pseudo code):

```
Procedure GRASP(Max_Iterations, Seed)
1.  Read_Input();
2.  for k = 1,..., Max_Iterations do
3.     Solution ← Greedy_Randomized_Construction(Seed);
4.     Solution ← Local_Search(Solution);
5.     Update_Solution(Solution,Best_Solution);
6.  end;
7.  return Best_Solution;
endGRASP.
```

圖 2-3 GRASP 的偽代碼 資料來源: Resende and Ribeiro[10]

2.2.1 GRASP 應用於船席調配問題

船席調配作業的過程就是陸續指派進港船隻到適合其停靠的船席以達到系統整體最佳化，直到全部的船都指派完畢。其中如何選擇船隻可能停靠的位置，是船席調配流程中的很重要的子問題。Lee *et al.*[2] 提出了一個尋找子問題可行解的有效方法以完成建構初始解階段，並應用 GRASP 演算法求解連續動態船席調配問題，其求解數學模式及範圍為 2.1.1 小節所示，而本小節將詳細介紹其演算法求解步驟。

2.2.1.1 搜尋子問題可行解之方法

在介紹方法前需要先了解以下四個名詞:

1. 節點(node): 在矩形時空圖的 xy 平面定義出船 n 的到達時間 x=a_n，以及整個船席調配作業的結束時間 x=T，再將目前的每個矩形的長寬邊緣延伸，所有延伸線段產生的交點，就稱為節點，如下圖 2-4 所示:

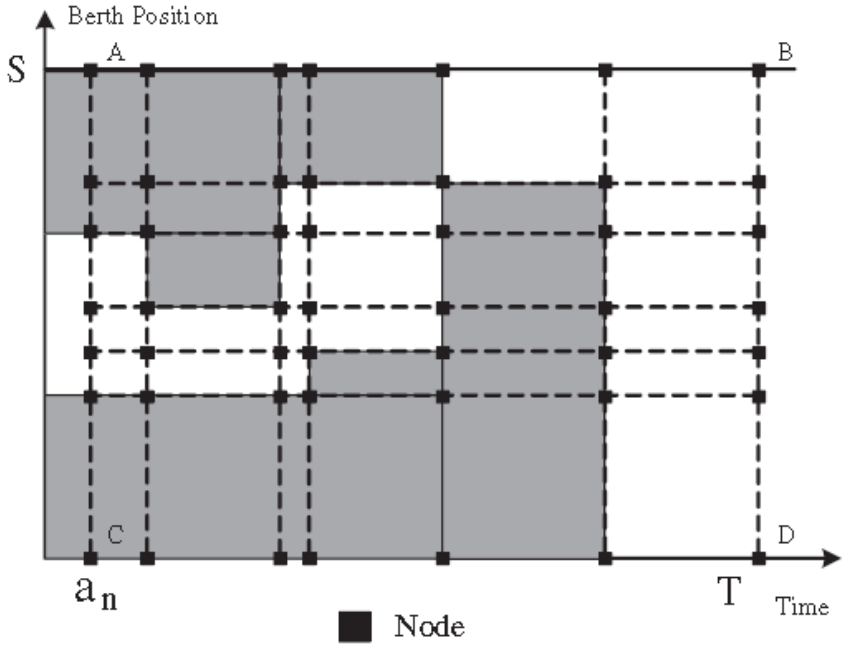


圖 2-4 節點示意圖，資料來源: Lee, D.H., Chen, J.H., & Cao, J.X. (2010)

2. 洞(Hole): 在矩形時空圖中沒有被船隻矩形佔據的空間，如下圖 6 所示。
3. 邊緣(Edge): 洞的邊緣線段
4. 頂點(Vertex): 為邊緣中的每個轉彎處，如下圖 6 所示，頂點是節點中的特殊情形。

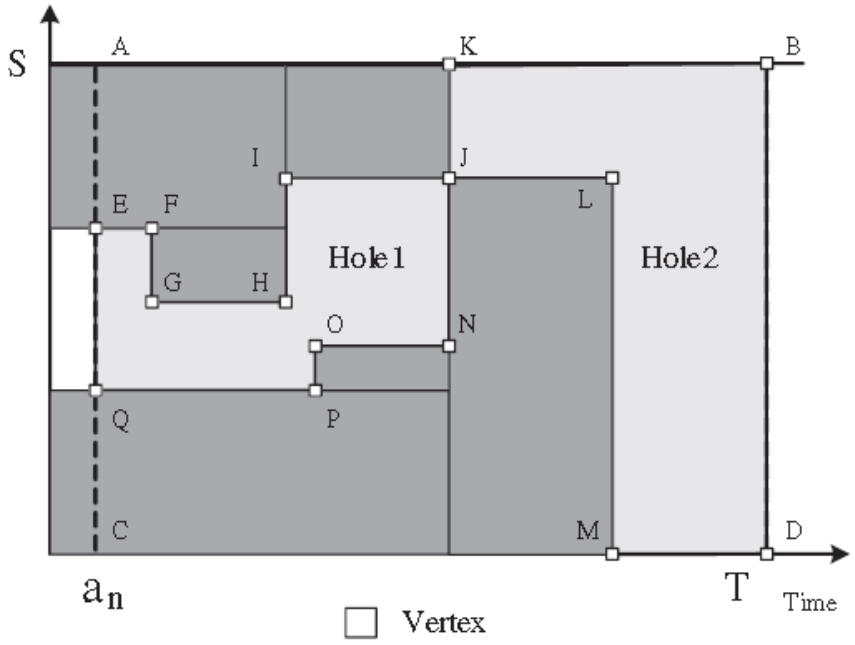


圖 2-5　頂點及洞的示意圖，資料來源: Lee, D.H., Chen, J.H., & Cao, J.X. (2010)

搜尋子問題可行解的方法流程如下:

1.輸入已知的資料

輸入欲排入的船 n 之前的 n-1 艘船的時空矩形，並定義出船 n 到港的時間a_n，以及整個作業時間的的結束點 T、船席長度 S、船 n 的大小s_i。

2.將節點轉換成向量

找出a_n和 T 之間的每個節點。並以每個節點為座標中心點並依其四個象限的性質，給予一個向量，若該象限為洞則記為 1，若不為洞則記為 0。如下圖 7，點 F 的向量依第 1、2、3、4 象限標記，可記為 [0，0，1，0]。

給予所有節點向量之後，可依其向量內 0 的數量將分成五個類別如下:

Class0: [0，0，0，0]

Class1: [1，0，0，0]、[0，1，0，0]、[0，0，1，0]、[0，0，0，1]

Class2: [1，1，0，0]、[1，0，1，0]、[1，0，0，1]、[0，1，1，0]、[0，1，0，1]、[0，0，1，1]

Class3: [1，1，1，0]、[1，1，0，1]、[1，0，1，1]、[0，1，1，1]

Class4: [1，1，1，1]

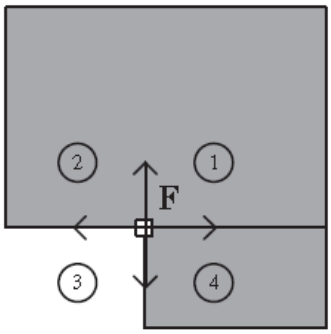


圖 2-6　點 F 的四個象限示意圖，資料來源: Lee, D.H., Chen, J.H., & Cao, J.X. (2010)

3.統合向量

接著按照以下方法把每個類別的向量都統合到 Class1，因為向量中每個 1 都表示矩形可以停靠的位置，全部轉換成 class1 可以方便判斷所有可能的停靠方式。

(1) 將 **class2** 中的[1，0，1，0]拆分成兩個向量，[1，0，0，0]及[0，0，1，0]。
(2) 將 **class2** 中的[0，1，0，1]拆分成兩個向量，[0，1，0，0]及[0，0，0，1]。
(3) 將 **class1** 中的[1，0，0，0]及[0，0，0，1]放進集合$\mathcal{C}_1$中
(4) 將所有 **Class3** 中的節點放進集合$\mathcal{C}_3$中，並$\mathcal{C}_3$中的每個節點如下表方式延伸一條有向線段。

表 2-2 $\mathcal{C}_3$中的節點之延伸線段方式

延伸方向	向量類型
左	[0，1，1，1]
下	[1，0，1，1]
上	[1，1，0，1]
左	[1，1，1，0]

為和其他$\mathcal{C}_3$中的節點之延伸線段產生交點如下圖 8 所示。點 U 及為節點之延伸線段與節點之延伸線段產生的交點，點 R、T 則為節點的延伸線段和洞的邊緣產生的交點，此類交點也是船可能停靠的位置。將所有節點的延伸線段和洞的邊緣產生的交點放入集合$\mathcal{I}_1$中。並將所有節點與節點之延伸線段產

生的交點放入集合中$\mathcal{I}_2$中。

以下表 2-3 之方式更新節點來源，更新$\mathcal{I}_1$及$\mathcal{I}_2$中的節點。

表 2 -3 更新$\mathcal{I}_1$及$\mathcal{I}_2$中的節點向量之方法，資料來源: Lee, D.H., Chen, J.H., & Cao, J.X. (2010)

集合$\mathcal{I}_1$		集合$\mathcal{I}_2$	
節點來源為下列向量與洞的邊緣的交點	更新後之向量	節點來源為下列兩個向量之延伸線段的交點	更新後之向量
[0,1,1,1]	[0,0,0,1]		
[1,0,1,1]	[1,0,0,0]	[1,0,1,1][1,1,1,0]	[1,0,0,0]
[1,1,0,1]	[0,0,0,1]	[1,1,0,1][0,1,1,1]	[0,0,0,1]
[1,1,1,0]	[1,0,0,0]		

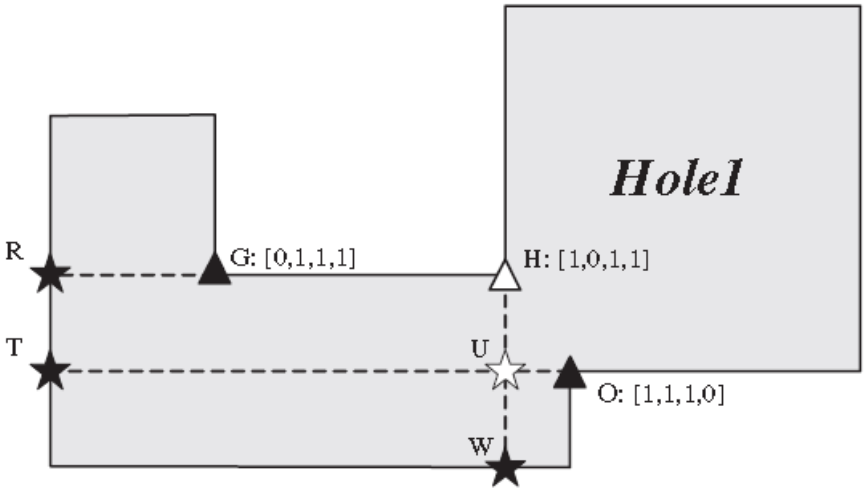


圖 2-7 $\mathcal{C}_3$ 中的節點之延伸線段產生之交點示意圖，資料來源: Lee, D.H., Chen, J.H., & Cao, J.X. (2010)

4.可行性分析

將$\mathcal{C}_1 \cup \mathcal{I}_1 \cup \mathcal{I}_2$中的節點進行可行性分析，如下圖 9 所示，i 為船 n 的時空矩形，矩形 i 左邊兩個頂點依靠在每個 Class1 中的節點，並剔除不可行的狀況，如狀況(c)和(e)，而(h)中若 H-U 長度小於矩形的高度，則矩形可以靠到最左邊。

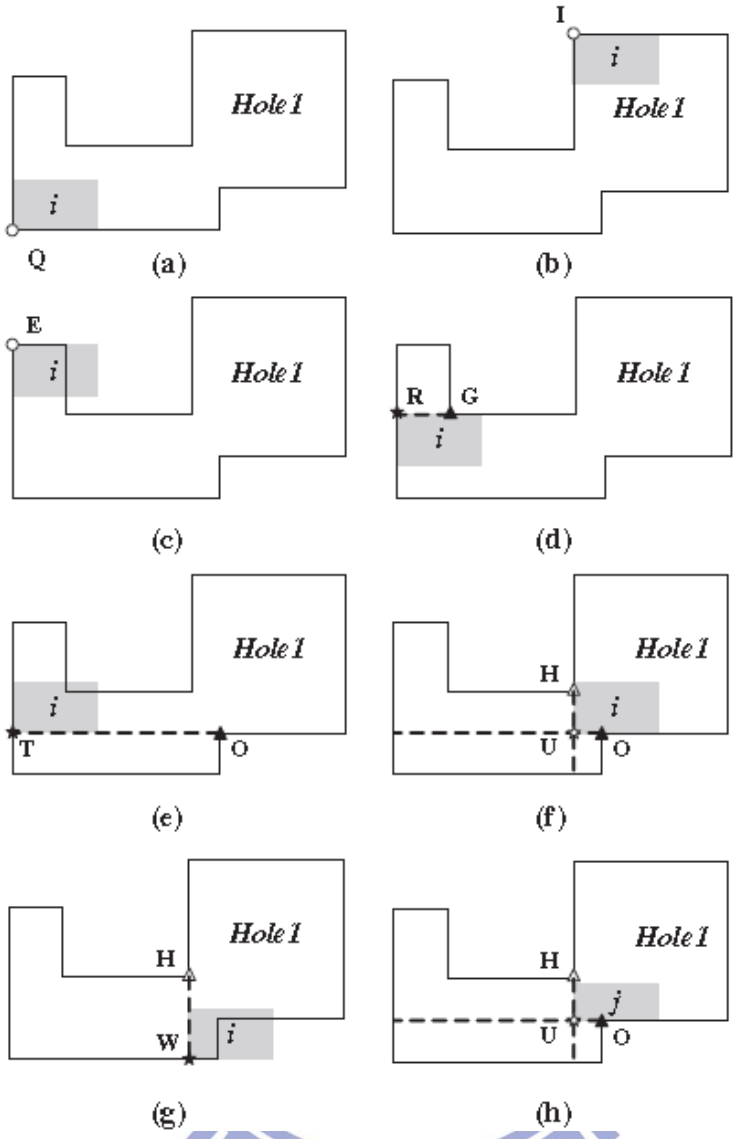


圖 2-8 可行性分析示意圖，資料來源: Lee, D.H., Chen, J.H., & Cao, J.X. (2010)

2.2.1.2 GRASP 求解流程

Lee, D.H.et al (2010) 提出之 GRASP 演算法有兩個，分別為 GRASP_1 及 GRASP_2，其求解船席調配問題的求解流程皆分為兩個階段，分述如下:

1. GRASP_1 建構階段:

(1) 將所有的船依照 ETA 排序產生一列表 Ves_seq。

(2) 從 Ves_seq 內取出即將指派的船 n，依照 2.2.1.1 介紹之方法判斷其可能

(3) 停靠的時空位置，若對於 n 來說有 K 個可能停靠的位置∈集合 K，依(10)式中的準則判斷，若滿足(10)式，則剔除該可能停靠的位置 k。

$$\frac{1}{cost_k} < r \times \max_{i=1}^{K}\left\{\frac{1}{Cost_i}\right\} \tag{10}$$

其中 r 代表對 K 中可能停靠位置相較所有位置中最少成本的位置的成本容許率

(4) 接下來對經由上一步驟篩選後的剩下的位置k^*，分別給予一個機率，以(11)式計算。

$$Pr_{k^*} = \frac{\frac{1}{cost_{k^*}}}{\sum_{i=1}^{k^*}\frac{1}{Cost_i}} \tag{11}$$

並依照這個機率對每個位置進行輪盤式選擇法(roulette wheel selection)，及機率越大越容易選中，而成本越大機率越小。

(5) 重複以上步驟直到 Ves_seq 中的船都被指派完為止。

2. GRASP_1 區域搜尋階段:

(1) 從 Ves_seq 中，挑選順序鄰近兩艘船隻交換產生新的序列，再重新計算其目標值，反覆進行l_1次之後，包含原本的初始解，從這 l_1+1 個解中挑選出最好的解。

(2) 將上一階段選出的解，對於其序列中的 B 點之後之後的每艘船重新依序再指派，而每艘船選擇停課位置時，都以成本最低的位置作為停靠位置。

3. 演算法停止條件:

反覆進行以上步驟後，如果目標值在l_2次計算內變穩定，則將 B 向前移動一個位置，且在l_3次內 B 不能在改變。B 如此往前移動但不得小於 LB。
若達成以下條件其中之一，則停止計算。

(1) 達到最大計算次數l_4
(2) 最佳解在l_5次計算後收斂

以下為 GRASP_2 演算法流程

1. GRASP_2 建構階段:

(1) 輸入所有船隻的資料和船席長度，給所有的船一個編號，放進集合 Ω，建立一個空集合叫做 Packed_Seq 用來放已經排進時空圖的船，再建立一個集合叫做 Unpacked_Seq，內容就跟 Ω 一樣，用來放尚未插入時空圖中的船。

(2) 建立一個集合 A，用來放下一艘要排進時空圖的船的所有可能停靠的位置，然後使用 2.2.1.1 介紹的方法判斷 Unpacked_Seq 內全部的船的所有可以停靠的位置，並將這些位置放入 A 中，再如 GRASP_1 建構階段 Step3 一樣利用等式

$$\frac{1}{\text{cost}_k} < r \times \max_{i=1}^{K}\left\{\frac{1}{\text{Cost}_i}\right\}$$

來過濾掉一些位置，並更新 A

(3) 如 GRASP_1 建構階段 Step4，把 A 中的所有可能停靠位置利用下式算出每個位置的機率值。

$$\text{Pr}_{k^*} = \frac{\frac{1}{\text{cost}_{k^*}}}{\sum_{i=1}^{k^*}\frac{1}{\text{Cost}_i}}$$

(4) 利用輪盤法選出最終決定的位置以及要插入的船，更新 Packed_Seq 和 Unpacked_Seq。

(5) 重複 2-4 步驟直到完整的解被建構出來。

2. GRASP_2 區域搜尋階段:

相較於 GRASP_1 區域搜尋階段 Step1 中兩艘船交換的步驟，GRASP_2 可任意取兩艘船交換，其餘步驟皆與 GRASP_1 相同，演算法停止條件也相同。

2.3 文獻回顧小結

Lee *et al*.[2]提出之以 GRASP 演算法求解動態連續船席調配問題，為求解動態連續船席調配問題中較新的研究，且注意到時空圖中若大船接著小船到達，在 hole 中可能有可停靠的位置。如下圖 2-9，若船 1、2、3 之 ETA 為 0，船 4 之 ETA 為 a4，船 5 之 ETA 為 a5，則過去之方法無法注意到圖中最左邊兩個可能停靠位置。

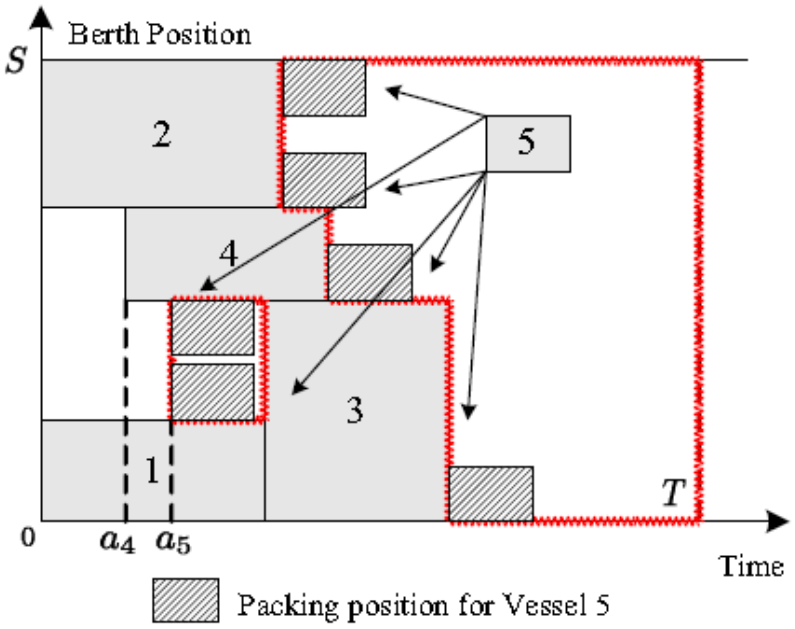


圖 2-9 產生 Hole 之可能情況，資料來源: Lee, D.H., Chen, J.H., & Cao, J.X. (2010)

就 Lee *et al*.[2]以 GRASP 求解結果顯示，由於 GRASP_2 搜尋深度較深，考慮了還沒排進時空圖裡的所有的船的可能停靠位置，所以可以求得較 GRASP_1 佳的解，但在大規模的計算，計算時間會甚至會超過 GRASP_1 的兩倍，所以 Lee, D.H. et al (2010)提出，在問題屬於小規模 5、10 艘船時，可以使用 GRASP_2 求解，但問題屬於大規模時 GRASP_1 較能同時符合時間效率及解品質的要求。

回顧以上文獻，由於船席調配資訊系統應用在較大規模的求解較有意義，故本研究針對 Lee *et al*.[2]中實用性較高的 GRASP_1 進行改良，期望改良之演算法維持相當於 GRASP_1 的解品質，縮短期運算時間，或是在相同的運算速度下，得到更好的求解結果。

第三章 改良式 GRASP 演算法求解 DBAPC

3.1 步驟改良構想

研究以上 GRASP 演算法後，3.1 節提出之改良步驟構想分別為以較好的起點開始求解以及避免增加機會成本，分述改善步驟及理由如下。

3.1.1 以較好的起點開始求解

Lee *et al.*[2]的測試題庫以每艘船之 ETA 符合 U (0，20)的均等分配方式產生，此情況在十艘船以上即為相當擁擠之情形，本研究提出假設先將船隻依大小分組，並讓每組內之船隻長度相加後盡量接近碼頭長度，再開始進行船席調配，則能將空間做最大的利用，如下圖 3-5。

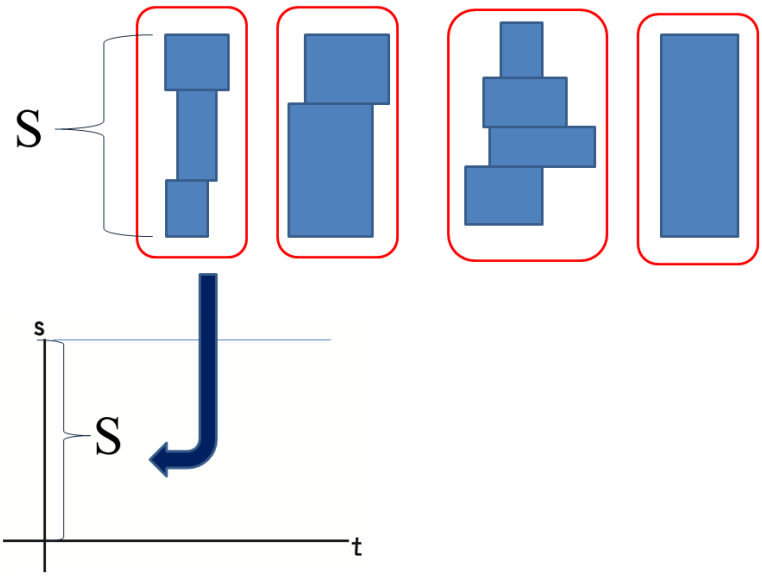


圖 3-1 船隻分組示意圖

則具體作法步驟如下:

1. 將船隻長度由大到小排序得到一序列 list。
2. 取出 list 中最大值之元素 max(list)放進集合 set 中，並從 list 中刪除 max(list)。
3. 搜尋 list 中每個元素，尋找符合長度≦10-sum(set)條件的值放入集合中，每當有元素加入 set 中，即更新 sum(set)。

4. 重複步驟 1~3 至 list 中所有元素皆被取出為止。

經由以上步驟可在 set 中將原本 list 裡的元素分成許多小組且其組內元素且總和盡量接近碼頭長度。

而 set 中的這些小組，若組內元素越多，表示組內的船隻長度較短，若組內元素越少，表示組內船隻長度較長，本研究先以 GRASP_1 之流程，設計以下三種不同的插入順序，取代 GRASP_1 原本之船隻照 ETA 排序的插入順序。

1. inference 1 組內元素少的組排前面 (盡量先插入大船)
2. inference 2 以組為單位亂序排列
3. inference 3 組內元素多的組排前面 (盡量先插入小船)

將每艘船依照 Lee, D.H. et al (2010)之測試題型設定為 ETA、船舶大小、處理時間分別以符合 U (0，20)、U (6，50)、U (20，80)的均等分配方式產生 40 艘船的規模之問題，參數設置為，L1~5 分別為 3、2、3、200、4，r 為 0.3，B 為 35，LB 為 30，求解 10 個例題，每個例題求解 3 次，將三種排列方式各 30 次之計算取平均值並進行比較，選擇求解後成本最低的方式，作為本演算法最終版本之改良插入順序，而測試結果如下表 3-1。

表 3 -1 inference 測試結果。

inference 1 cost	**16408.13**
execution time	45.56
inference 2 cost	**16404.9**
execution time	45.08
inference 3 cost	**13686.16**
execution time	40.99

由測試結果發現inference 3 不論是求解時間或是求解品質均較其他插入順序為佳，所以預計改良之演算法，以 inference 3 之插入順序開始求解。

3.1.2 避免增加機會成本

由於 3.1.1 提出的概念以及研究範圍皆不遵守先到先服務原則，所以可能產生如下圖 3-1 的情況。

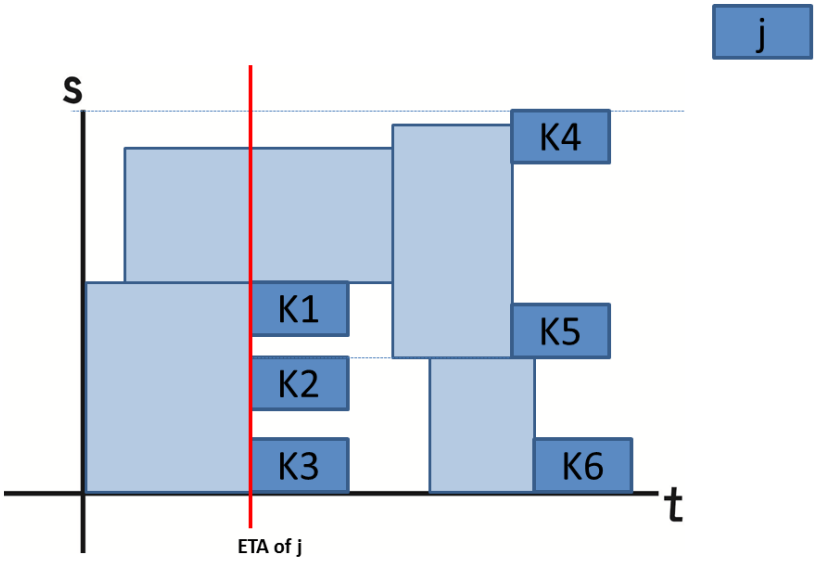


圖 3-2 船 j 之機會成本示意圖(1)

上圖 3-1 之 k1～6 為船 j 之可行的停靠位置，以 k1～3 來說，在目標式裡的成本均相同，所以在 2.2.1.1 中之以輪盤法選擇時的機率是相等的，若最後選擇為 K2，則如下圖 3-2，可能使得下一個要停靠的船失去了一個可停靠且成本相較為低的位置。

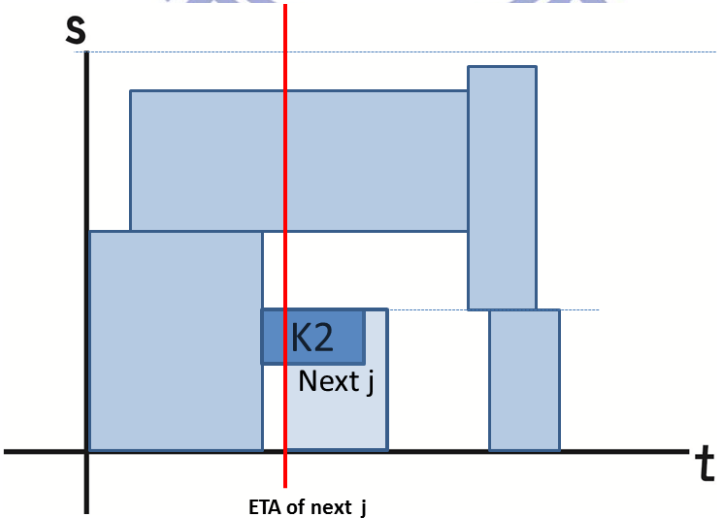


圖 3-3 船 j 之機會成本示意圖(2)

但若經由輪盤法選擇後的位置為 k1，則可留給接下來停靠進來的船較大的空間，保留最大的機會給後來停進來的船，如下圖 3-3。

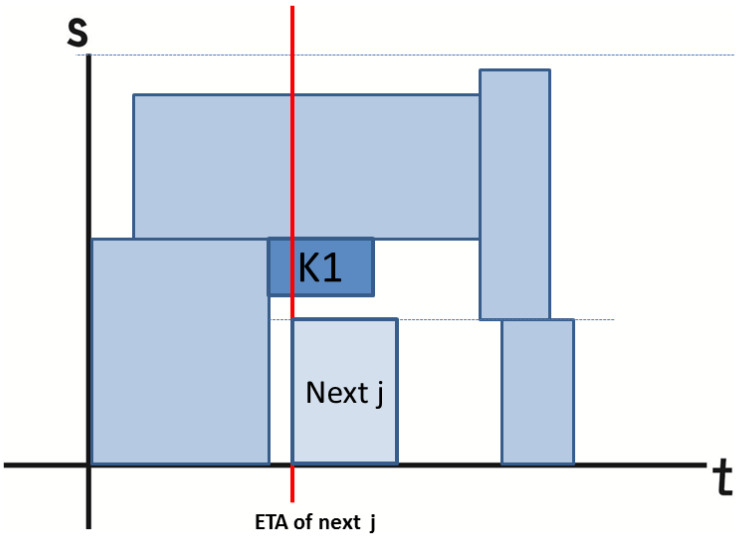


圖 3-4 船 j 之機會成本示意圖(3)

依以上推論可以得知雖然 K2～3 以目標式求得的成本相同，但選擇 K2 的機會成本卻比選擇 K1、K3 為大，所以本研究提出之改良步驟之一為將時空圖上所有可能停靠位置中，X 值相同的停靠位置為三個以上時，將 Y 值為最高或最低的兩個停靠位置保留，刪除 Y 值界於前述兩個停靠位置之前的其他停靠位置，若相同 X 值的可能停靠位置僅為一個或兩個，則直接保留，以圖 3-4 來說，需刪除掉 K2。

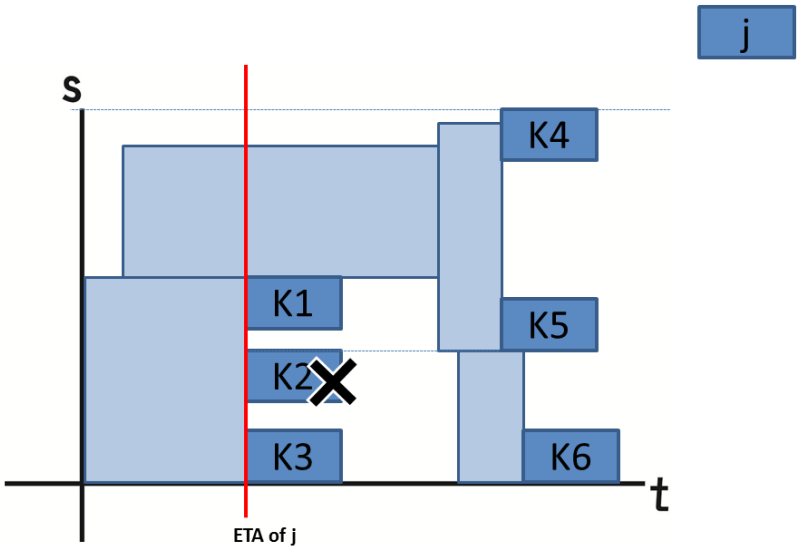


圖 3-5 船 j 之機會成本示意圖(4)

3.2 改良式 GRASP 求解流程

由以上推論及測試結果，本研究發展出兩種改良式演算法分別為 MYGRASP_1 及 MYGRASP_2，其求解船席調配問題的求解流程同樣分為兩個階段，分述如下:

1. MYGRASP_1 建構階段:

(1) 將所有的船依照 3.1.1 提出之順序排序成一列表 Ves_seq。

(2) 從 Ves_seq 內取出即將指派的船 n，依照 2.2.1.1 介紹之方法判斷其可能

(3) 停靠的時空位置，若對於 n 來說有 K 個可能停靠的位置∈集合 K，依(10)式中的準則判斷，若滿足(10)式，則剔除該可能停靠的位置 k。

$$\frac{1}{cost_k} < r \times \max_{i=1}^{K}\left\{\frac{1}{Cost_i}\right\} \tag{10}$$

其中 r 代表對 K 中可能停靠位置相較所有位置中最少成本的位置的成本容許率。

(4) 如 3.1.2 介紹之觀念，將剩下來的的可能停靠位置中，停靠時間相同的停靠位放進集合 K_t 中，下標 t 代表停靠時間。保留K_t中停靠位置最靠近碼頭邊緣兩個可能位置其他剔除，若K_t中原本只有一個 k 或兩個 k，就直接保留。

(5) 接下來對經由上一步驟篩選後的剩下的所有位置k ，分別給予一個機率，以 (11)式計算。

$$Pr_{k^*} = \frac{\frac{1}{cost_{k^*}}}{\sum_{i=1}^{k^*}\frac{1}{Cost_i}} \tag{11}$$

並依照這個機率對每個位置進行輪盤式選擇法(roulette wheel selection)，及機率越大越容易選中。

(6) 重複以上步驟直到 Ves_seq 中的船都被指派完為止。

2. MYGRASP_1 區域搜尋階段:

(1) 從 Ves_seq 中，挑選順序鄰近兩艘船隻交換產生新的序列，再重新計算其目標值，反覆進行l_1次之後，包含原本的初始解，從這 l_1+1 個解中挑選出最好的解。

(2) 將上一階段選出的解，對於其序列中的 B 點之後之後的每艘船重新依序再指派，而每艘船選擇停課位置時，都以成本最低的位置作為停靠位置。

3. 演算法停止條件:

反覆進行以上步驟後，如果目標值在l_2次計算內變穩定，則將 B 向前移動一個位置，且在l_3次內 B 不能在改變。B 如此往前移動但不得小於 LB。
若達成以下條件其中之一，則停止計算。

(1) 達到最大計算次數l_4
(2) 最佳解在l_5次計算後收斂

MYGRASP_2 的演算法流程除了區域搜尋階段 Step1 與 MYGRASP_1 不同其他步驟皆相同，演算法停止條件也相同。

1. MYGRASP_2 區域搜尋階段 Step1 敘述如下:

從 Ves_seq 中，挑選順序任意兩艘船隻交換產生新的序列，再重新計算其目標值，反覆進行l_1次之後，包含原本的初始解，從這 l_1+1 個解中挑選出最好的解。

接下來的章節，將會在和 Lee *et al.* [2] 在相同的條件下進行比較，並分析結果。

第四章 範例驗證與分析

4.1 求解品質與計算時間

本範例測試使用的電腦處理器為 AMD phenom (tm) 2 II X2 545 processer 3.0GHz，記憶體為 3.25GB。

參照 Lee *et al*. [2]的測試題庫以每艘船之 ETA、船舶大小、處理時間分別為符合 U (0，20)、U (6，50)、U (20，80)的均等分配方式產生，測試小規模範例 5 艘及 10 艘船，並分別各產生 30 組不同的例題，參數 L1~5 分別為 10,5,10,200,25，r 為 0.3，B 為(7/8×n)取最接近的整數，LB 為(3/4×n)取最接近的整數，n 為船的數量。

每個範例規模各 30 次之計算結果平均如下表:

表 4 -1 小規模範例測試結果

	目標值	std*	求解時間	std*
n=5				
GRASP_1	326.73	76.87	1.22	0.23
MYGRASP_1	326.03	75.12	1.41	0.25
MYGRASP_2	326.03	75.12	1.27	0.22
n=10				
GRASP_1	1037.03	234.02	9.38	2.80
MYGRASP_1	1016.63	227.02	8.79	3.09
MYGRASP_2	1021.40	223.48	9.98	3.00

*std 為 30 次計算的母體標準差

整理上表 4-1 可得到以 MYGRASP1、MYGRASP2 目標值和計算時間相對於 GRASP_1 的百分比，如下表 4-2。

表 4 -2　MYGRASP1、MYGRASP2 目標值和計算時間相對於 GRASP_1 的百分比

	目標值百分比	求解時間百分比
n=5		
GRASP_1	100.00%	(1.22s)100.00%
MYGRASP_1	99.79%	(1.41s)115.07%
MYGRASP_2	99.79%	(1.27s)104.08%
n=10		
GRASP_1	100.00%	(9.38s)100.00%
MYGRASP_1	98.03%	(8.79s)93.66%
MYGRASP_2	98.49%	(9.98s)106.41%

由表 4-2 可以發現不論是 5 艘或 10 艘船的計算規模，MYGRASP_1、MYGRASP_2 都可以得到較佳的解，且十艘船十比五艘船的改善更明顯。但在計算時間上則略輸 GRASP_1，但小規模的計算時間差並不會很大(約差 1 秒)，所以為了得到更好的解，在計算時間上這是可以接受的損失。

大規模範例參照 Lee *et al.*[2]的測試題庫以每艘船之 ETA、船舶大小、處理時間分別為符合 U (0，20)、U (6，50)、U (20，80)的均等分配方式產生，測試大規模範例 40 艘、80 艘、120 艘、160 艘、200 艘船，並分別各產生 30 組不同的例題每個例題算 3 次，參數 L1~5 分別為 3、2、3、200、4，r 為 0.3，B 為 7/8×n，取最接近的整數，LB 為 3/4×n，取最接近的整數，n 為船的數量。
每個範例規模各 90 次之計算結果平均如下表:

表 4-3 大規模範例測試結果

	目標值	std*	求解時間	std*
n=40				
GRASP_1	14510.33	1768.04	40.76	17.65
MYGRASP_1	13284.25	1664.58	40.57	13.16
MYGRASP_2	13577.65	1604.07	30.87	7.765
n=80				
GRASP_1	56686.57	3329.03	370.29	154.81
MYGRASP_1	50719.43	3354.87	356.13	128.30
MYGRASP_2	51659.32	3470.83	268.49	56.73
n=120				
GRASP_1	124975.12	7663.57	1465.52	570.98
MYGRASP_1	108612.33	6357.10	1318.48	432.28
MYGRASP_2	110139.20	6808.57	997.43	195.45
n=160				
GRASP_1	227705.47	11418.48	3795.87	1336.19
MYGRASP_1	196563.98	11034.35	3352.36	1039.30
MYGRASP_2	198649.73	10722.97	2530.36	511.23
n=200				
GRASP_1	352305.82	18085.35	7519.45	2485.72
MYGRASP_1	304208.67	16624.77	6873.21	2193.09
MYGRASP_2	306835.18	16829.88	5508.52	1206.65

*std 為 90 次計算的母體標準差

本研究每次計算之後可將結果呈現在矩形時空圖，以利於判斷程式是否有錯，如下圖四-1，為 40 艘船其中一次的計算的結果。

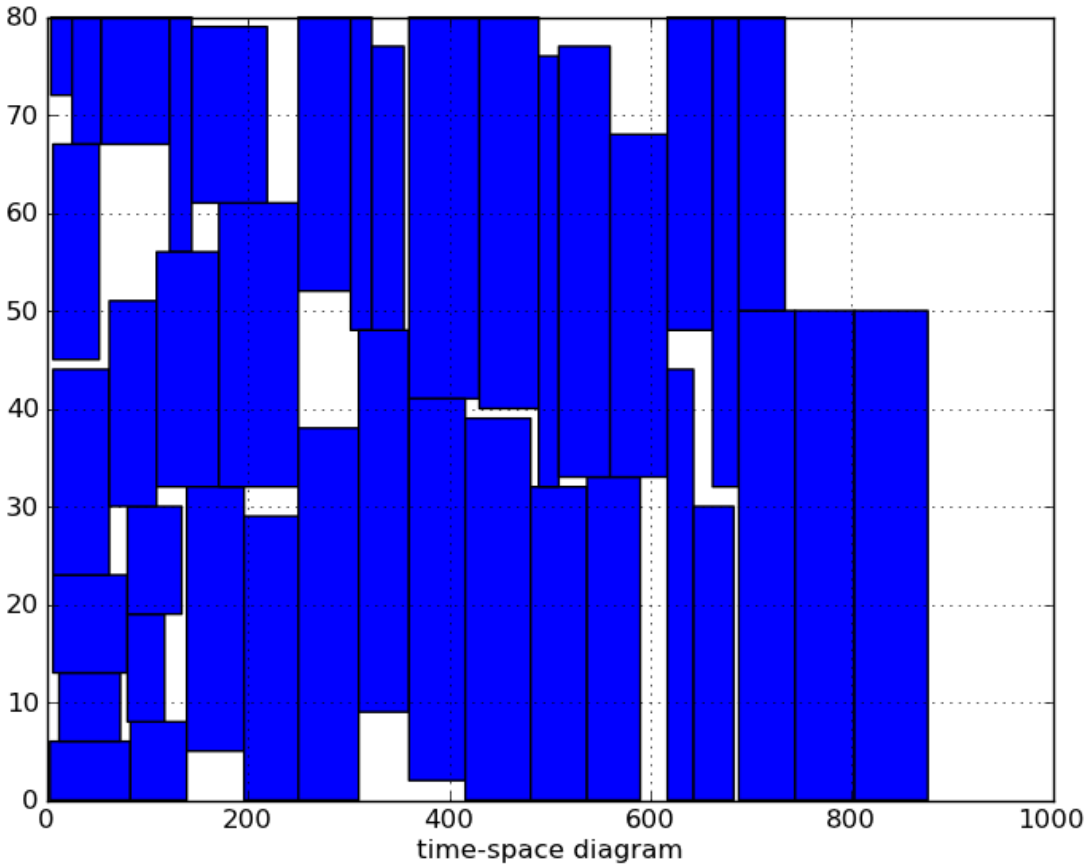


圖 4-1 本研究程式計算結果之矩形時空圖呈現方式

歸納表 4-3 可得 MYGRASP_1、MYGRASP_2 目標值相對於 GRASP_1 的百分比歸納於下表:

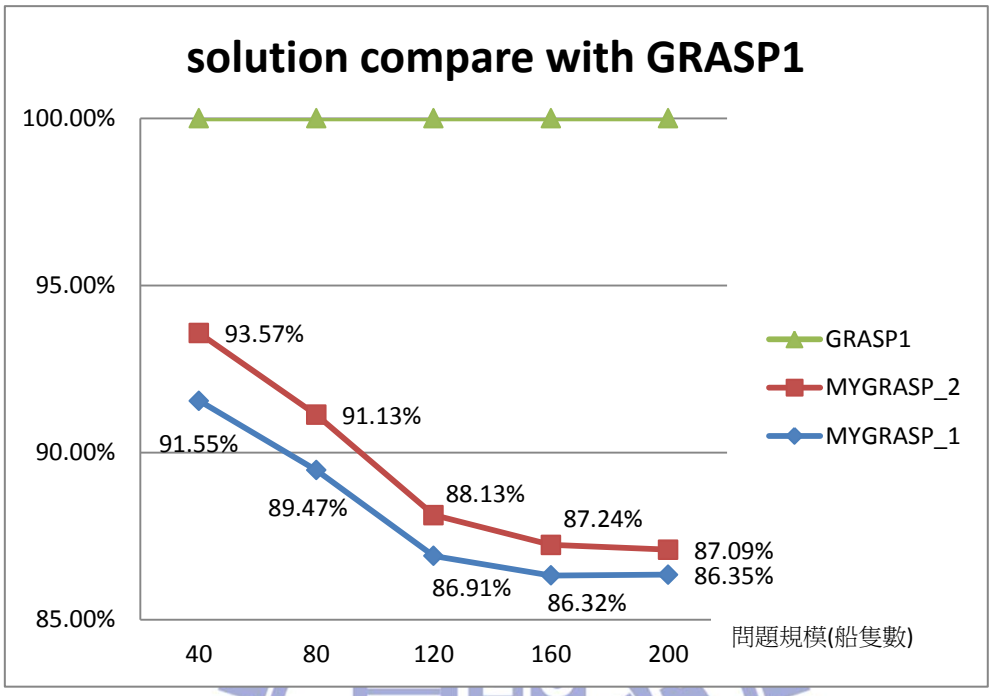


圖 4-2 MYGRASP1、MYGRASP2 目標值相對於 GRASP_1 的百分比折線圖

由上圖 4-2 中可以發現，MYGRASP_1、MYGRASP_2 在大問題規模時相較於 GRASP_1 目標值可得到 6.43%~13.65%改善。且隨著計算問題規模越來越大，MYGRASP_1、MYGRASP_2 得到的解相較於 GRASP1 的改善會越明顯，而不論在何種問題規模 MYGRASP_1 都可求得較 MYGRASP_2 好的解。

歸納表 4-3 可得 MYGRASP_1、MYGRASP_2 計算時間相對於 GRASP_1 的百分比歸納於下表:

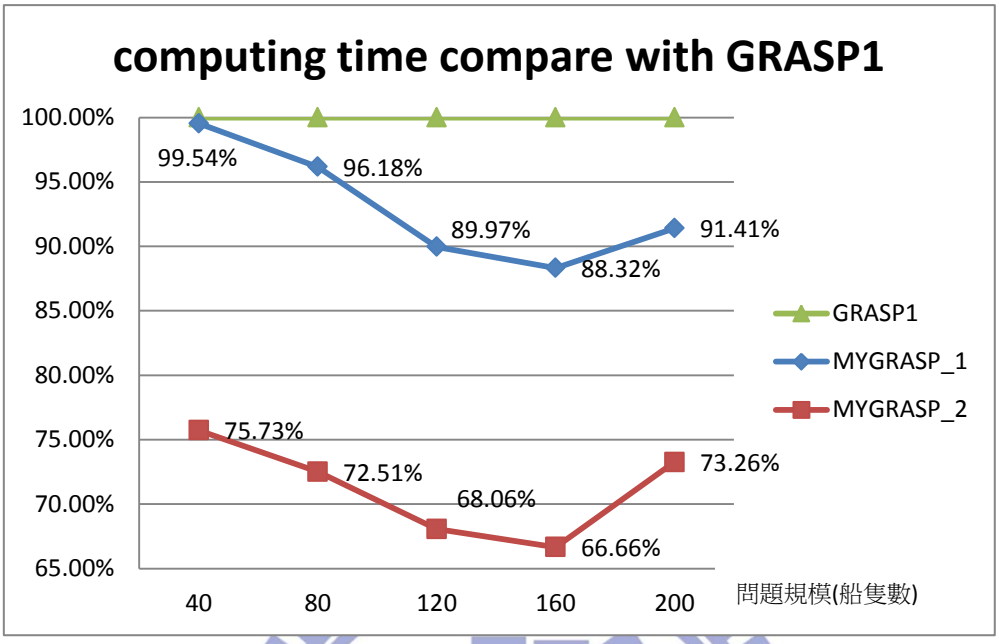


圖 4-3 MYGRASP1、MYGRASP_2 計算時間相對於 GRASP_1 的百分比折線圖

由上圖 4-3 中可以發現，MYGRASP_2 的計算時間可維持在 GRASP_1 的 75.73%~66.66%間，MYGRASP_1 在計算時間上最大也可達到 11.68%的改善，而不論在何種問題規模 MYGRASP_2 的計算速度都明顯較 MYGRASP_1、GRASP_1 快。

4.2 收斂性分析

將每艘船依照 Lee *et al.*[2]之測試題型設定為 ETA、船舶大小、處理時間分別以符合 U (0，20)、U (6，50)、U (20，80)的均等分配方式產生 40 艘船的規模之問題，參數設置為，L1~5 分別為 3、2、3、200、4，r 為 0.3，B 為 35，LB 為 30，求解 30 個例題，每個例題求解 3 次，將每次迭代的值取平均。可以得到下表 4-4 及圖 4-4。

表 4-4 三個演算法 90 次計算之平均迭代數

	平均迭代數
GRASP_1	10.5
MYGRASP_1	10.14
MYGRASP_2	7.27

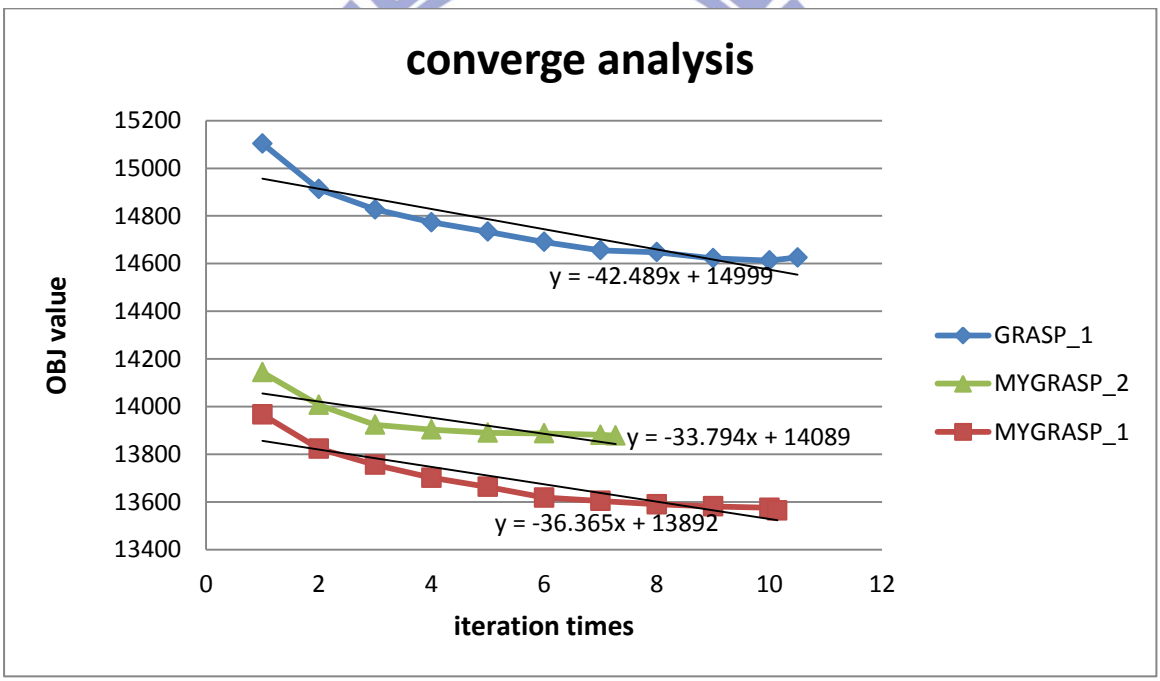


圖 4-4 三個演算法之收斂性分析

由上圖 4-4 可以發現三個演算法的線性迴歸線斜率相近，但是因為 MYGRASP_1 從較好的起點開始求解，所以相對得到更好的求解結果。而 MYGRASP_2 的線性迴歸線斜率較平緩，表示雖然和 MYGRASP_1 都是以較好的起點來求解，但其區域搜尋時任意兩艘船交換的方式，對這個起始解來說，之後會比較容易掉入區域最佳解中，所以迭代數較少，相對較早收斂，計算時間也較短。

4.3 調整大小船比例對演算法有效性之影響

如下圖 4-5 所示，由於大小船的比例，會影響同一個時間點可能停靠的位置之多寡，故本研究假設改良之方法的有效性，會受到小船和大船的數量比例影響。

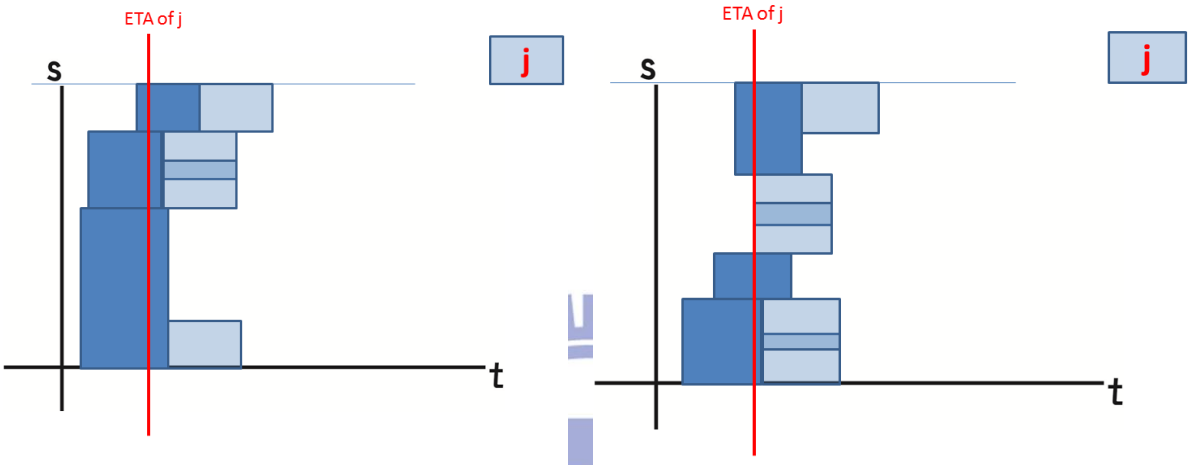


圖 4-5 小船較數量對於可停靠位置的關係

本研究設計 A、B、C、D 三組船舶大小分配不同的例題如下表 4-5，n 為船隻數量。

表 4 -5 三組船隻大小分配不同的例題

A	n
U(6,20)	5
U(20,50)	35
B	
U(6,50)	40
C	
U(6,20)	20
U(20,50)	20
D	
U(6,20)	35
U(20,50)	5

表 4-5 A、B、C、D 四組例題皆為 40 艘船，且小船佔的比例依序變多。除了船舶大小外其他條件和參數皆參照 4.2 節之方式設置，每組隨機產生 30 個例題，每個例題求解 3 次，將求得的值與 GRASP_1 之值相比，可得下表 4-6。

計算結果如下表 4-6:

表 4 -6 四種不同大小船比例的計算結果與 GRASP_1 相比

compare with GRASP_1

	Set A		Set B	
	目標值	計算時間	目標值	計算時間
GRASP_1	100.00%	100.00%	100.00%	100.00%
MYGRASP_1	90.37%	108.23%	91.55%	99.54%
MYGRASP_2	93.71%	79.35%	93.57%	75.73%
	Set C		**Set D**	
GRASP_1	100.00%	100.00%	100.00%	100.00%
MYGRASP_1	99.33%	95.37%	101.21%	112.19%
MYGRASP_2	96.02%	99.17%	99.95%	105.92%

表 4-6 可整理為下圖 4-6

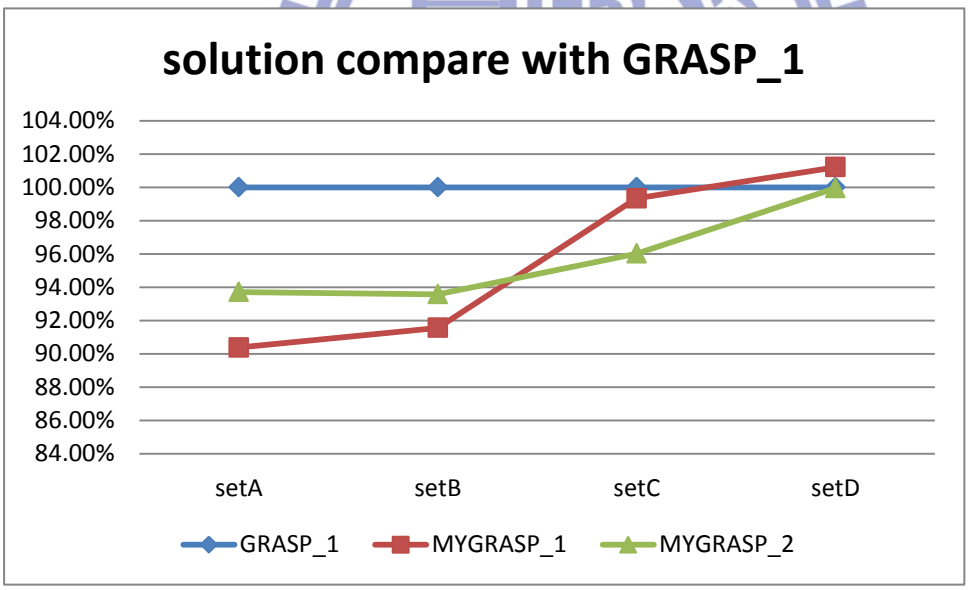


圖 4-6 三種組合之目標值相比與 GRASP_1 相比

由上圖可以發現，小船越多會使得 MYGRASP_1、MYGRASP_2 的解相較於 GRASP_1 的解變得越差，原因是由於若小船變太多，卻只保留最接近港口邊緣的位置，會忽略掉很多搜尋方向，而且小船變多互相會造成機會成本的可能性也就相對會變低，所以 3.1.2 節提出之概念在小船過多時反而使有效性變差。

表 4-6 之三種組合之求解時間與 GRASP_1 相比可整理為下圖 4-7

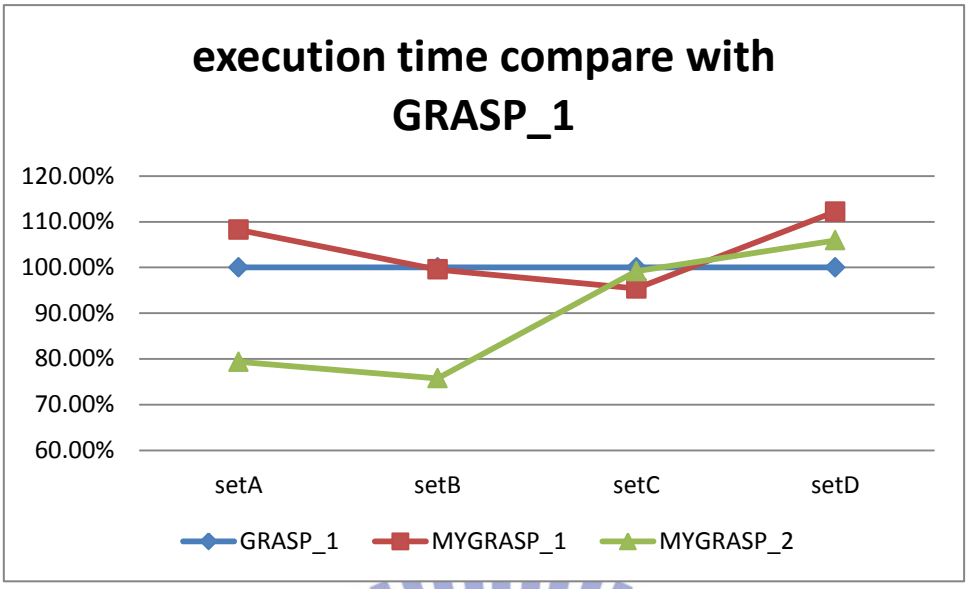


圖 4-7 四種組合之求解時間與 GRASP_1 相比

而關於計算時間方面，若是小船比例越高，會使得 MYGRASP_2、求解時間變久，但對 MYGRASP_1 來說大小船比例較平均時，如 set B、set C，求解時間較低。

4.4 調整船隻到達的離散性對演算法有效性之影響

Lee *et al.*[2]的測試例題屬於非常擁擠的情況，甚至在大規模時，擁擠到接近靜態的船席調配。由於當船隻到達時間非常鬆散時，船隻只要到了就直接靠港，符合先到先服務即為最佳的船席指派，所以本小節假設船隻插入時空圖的順序會被船隻到港的擁擠度影響，進而比較船隻到達時間分別為符合 U(0,20)、U(0,40)、U(0,60) 、U(0,80)的情況，參數設置除了 ETA 的分配其他則和 4.1 節大樣本的設置一樣，並將測試 40 艘船的結果，求解 30 個例題每個例題求解 3 遍，將 90 次的結果平均且與 GRASP_1 比較，可以得到下表 4-7。

表 4-7 到達時較分散時對結果之影響

	目標值			
	U(0,20)	**U(0,40)**	**U(0,60)**	**U(0,80)**
GRASP1	100.00%	100.00%	100.00%	100.00%
MYGRASP_1	91.55%	92.61%	92.41%	91.99%
MYGRASP_2	93.57%	94.87%	94.19%	93.68%
	求解時間			
GRASP1	100.00%	100.00%	100.00%	100.00%
MYGRASP_1	99.54%	110.45%	114.26%	108.22%
MYGRASP_2	75.73%	82.01%	84.52%	86.78%

由上表 4-7 結果發現，當到達時間變的離散時，MYGRASP_1、MYGRASP_2 的求解品質並沒有受到明顯的影響，但求解時間卻明顯增加。由於三個演算法的基本結構流程都相同，且由 4.2 節發現，迭代數會造成計算時間有明顯改變，可見 ETA 分布範圍若是變的較離散，MYGRASP_1、MYGRASP_1 的迭代數會相對 GRASP_1 增加，雖然每次迭代搜尋最佳解的能力會變弱，但卻還是可以跳出區域最佳解，所以在多幾次迭代之後，依然可以搜尋的到和原本的解品質相當的解。

第五章 結論與建議

本研究可歸納之結論如下:

1. MYGRASP_1 及 MYGRASP_2 相較於 GRASP1 不論在解品質或是時間，均有大幅度的改善，其中 MYGRASP_1 的計算時間較 MYGRASP_2 長，但解品質較好，MYGRASP_2 計算時間較短，但解品質較差。就時間與解品質的取決下，建議使用 MYGRASP_2，因為 MYGRASP_2 的求出的解只略輸 MYGRASP_1，但其計算時間卻較 MYGRASP_1 快許多(見圖四-2、圖四-3)。

2. 對於 MYGRASP_1 、MYGRASP_2 來說，小船比例較大船少的情形，題目的複雜度較低，求得的解也較佳，且 3.1.2 之方法可以先刪除去造成機會成本的位置，使求解品質變好，但若用在小船過多的情況，由於互相造成機會的情況也變少，使用 3.1.2 節之觀念，反而會喪失許多找到更好的解的機會，所以 3.1.2 節之觀念只適合用在小船比例相對少的情況。

3. 3.1.1 提出之以較好的起點開始求解的概念，是本研究改良演算法在時間上和求解品質上大幅勝出的主因，但在 ETA 分布較離散得情況，雖然相較於 GRASP_1 可以得到一樣好的解品質，但卻要花更多的計算時間。

本研究提出之建議如下:

1. 由於本研究之連續動態船席調配問題並沒有考慮到停靠位置和貨櫃堆積場的距離對成本造成的影響，故後續研究可延伸此常數為變數。

2. 本研究探討之問題情境尚無討論到加入時間窗的概念，故後續研究可延伸發展。

3. 以較好的 3.1.1 提出之以較好的起點開始求解之觀念，後續研究可將其接續在 GRASP 之外的演算法上，觀察是否能得到更好的結果。

附錄一 以 LINGO 驗證 GRASP_1 演算法

本節附錄驗證 GRASP_1 演算法在本研究以 python 語言撰寫之後，是否維持一樣的求解水準，以證實本研究程式的正確性。

以套裝軟體 LINGO 求解 4.1 節 5 艘船的 30 題，並將此 30 題代入 2.1.1 節之 DBAPC 數學模式，求解結果如下:

表 附錄-1 以 LINGO 驗證 GRASP_1 演算法

5 vessels x 30 instnces		
LINGO solution	**326.03**	**100.00%**
execution time	0.30	
GRASP1 solution	**326.73**	**100.21%**
GRASP1 iteration	30.57	
execution time	1.23	
MYGRASP1 solution	**326.03**	**100.00%**
MYGRASP1 iteration	32.53	
execution time	1.41	
MYGRASP2 solution	**326.03**	**100.00%**
MYGRASP2 iteration	29.20	
execution time	1.28	

一般演算法在求解小規模問題時，由於問題不到大規模 NP-HARD 類型，所以可以求得全域最佳解。而 GRASP_1 演算法在 30 次計算的平均之下和最佳解僅有 0.21%的誤差，此誤差是來自於 30 次計算的其中一次沒有求得最佳解而造成的，故表附錄-1 足以驗證演算法的正確性。

參考文獻

[1] Lim, A. (1998). "The Berth Planning Problem1." Operations Research Letters 22(2-3): 105-110.

[2] Lee, D. H., J. H. Chen, et al. (2010). "The Continuous Berth Allocation Problem: A Greedy Randomized Adaptive Search Solution." Transportation Research Part E: Logistics and Transportation Review 46(6): 1017-1029.

[3] Imai, A., X. Sun, et al. (2005). "Berth Allocation in a Container Port: using a Continuous Location Space aApproach." Transportation Research Part B: Methodological 39(3): 199-221.

[4] Imai, A., K. I. Nagaiwa, et al. (1997). "Efficient Planning of Berth Allocation for Container Terminals in Asia." Journal of Advanced Transportation 31(1): 75-94.

[5] Guan, Y. and R. K. Cheung (2005). "The Berth Allocation Problem: models and solution methods." Container Terminals and Automated Transport Systems: 141-158.

[6] Kim, K. H. and K. C. Moon (2003). "Berth Scheduling by Simulated Annealing." Transportation Research Part B: Methodological 37(6): 541-560.

[7] Wang, F. and A. Lim (2007). "A stochastic Beam Search for the Berth Allocation Problem." Decision Support Systems 42(4): 2186-2196.

[8] Cordeau, J.F., Laporte, G., Legato, P., & Moccia, L. (2005). " Models and tabu search heuristics for the berth-allocation problem. " Transportation science, 39(4), 526-538.

[9] Feo, T. A. and M. G. C. Resende (1995). "Greedy Randomized Adaptive search Procedures." Journal of Global Optimization 6(2): 109-133.

[10] Resende, M. and C. Ribeiro (2003). "Greedy Randomized Adaptive Search Procedures." Handbook of metaheuristics: 219-249.

簡歷

姓　　名：林岱暘
出 生 地：台北市
出生日期：民國74年1月8日
聯絡地址：北市內湖區江南街71巷65弄23號3樓
電子信箱：my.yuppieflu@gmail.com

學歷：
民國100年7月　國立交通大學運輸科技與管理學系碩士班畢業
民國97年7月　私立淡江大學運輸管理學系畢業
民國92年7月　私立東山高中畢業