國立成功大學資訊管理研究所

碩士論文

# 以網路模式求解考慮預算限制之風險最小化鋪面養護規劃問題

# A network flow model for optimal pavement rehabilitation planning with least risks under a budget constraint

研究生　　：武葉卿

指導教授　：王逸琳　博士

中華民國一百零三年六月五日

# 摘要

公路是國家發展的重要命脈，高品質的公路不僅可提升人民的生活水準，還帶動了國家的交通運輸、經濟、政治等各領域的發展。因此，如何而能不間斷地持續維護與改善公路品質，以求更安全舒適的行車環境至為重要。隨著公路的使用率增加，各路段不一之鋪面毀損程度亦將日益惡化，然而其鋪面養護預算卻經常不足以全面修復所有路段。因此，政府常必須選擇性地將有限的鋪面養護經費花在修復部分路段，以極小化公路的整體風險程度。

本研究針對鋪面毀損狀況不同的一維連續路段，在有限的養護維修預算下，探討如何將路段分組維修以達成整體風險程度最小之路段集群維修問題。此一路段集群維修問題必須遵守一些行之有年的特殊業界修復規範與習慣，譬如目前之鋪面風險程度高於門檻值的路段一定要被修復；被劃分於同一修復分組的待修路段總數必須足夠多，且這些路段必須為連續相鄰；此外，同一分組中的所有路段一定要採取其分組中修復花費最昂貴路段的修復方式來全面修復之。雖然被修復的路段其風險程度將被歸零，但不同的修復分組方式亦將引發不同的維修成本，因此求解最佳的修復分組方式為一個複雜的組合最佳化問題。

本研究首先將各路段與分組分別視為顧客與廠商，即可將本問題轉換成一個特殊的無容量限制之設施區位問題，並建構一整數規劃模式求解之。然而該整數規劃模式必須使用許多限制式來處理分組的規範與習慣，因而導致求解過程極為耗時。因此，本研究再將各路段視為節點，各種合乎風險門檻與分組路段總數限制的分組方式視為「分組節線」，並針對可以不用修復的路段建立「原狀節線」，藉以建構一個特殊的網路圖；其中每條節線代表路段的分組，分別具有其維修成本與風險等兩種權重，而該網路圖上自路段一的起始節點連通至最終路段節點的任一路徑即代表一種修復分組方式，其路徑上所有節線的兩種權重分別加總即代表該種修復方式的總維修花費與總風險，而求取滿足預算限制之最佳的修復分組方式即可對應成於該網路圖上計算一條滿足額外預算限制的最短路徑問題（Constrained Shortest Path, CSP）。

為測試解法之求解效能，我們利用最佳化軟體 Gurobi 求解本研究所發展之特殊設施區位問題以及 CSP 等兩種整數規劃模式，再依據 CSP 文獻設計拉氏鬆弛法

(Lagrangian Relaxation, LR)與第 K 短路徑演算法(KSP)等兩種常見的 CSP 求解方法。數值測試結果顯示，以 Gurobi 求解設施區位與 CSP 等兩種整數規劃模式，以及 KSP 之求解時間經常會隨著問題規模擴大而急遽增加；LR 雖可以在較短時間快速求得一個好的解，但偶爾會因對偶間隙(Duality Gap)而難以收斂。因此，本研究最後以 LR 之可行解為基礎，再運用模擬退火法(Simulated Annealing, SA)加以改善收斂效率，以更有效地求解路段集群維修問題。

**關鍵字：**鋪面養護，路段集群維修問題，無容量限制的設施區位，含額外限制式的最短路徑問題，拉氏鬆弛法，第 K 短路徑演算法，模擬退火法。

# A network flow model for optimal pavement rehabilitation planning with least risks under a budget constraint

Diep Khanh Vu

I-Lin Wang

Institute of Information Management

## SUMMARY

The regular road maintenance and rehabilitation (M&R) plan is important to keep the surface in a good shape. In practice, a road is divided into a set of contiguous segments, where each set is a pavement project treated by the same M&R plan. With limited budget, we investigate an optimal pavement rehabilitation plan that seeks an optimal combination of road segments and their rehabilitation plan such that the total risks are minimized with limited budget.

By treating each segment as a customer and a pavement project as a warehouse, this problem can be formulated as a difficult uncapacitated facility location problem (UFLP) by integer program (IP). Then, we formulate this problem as a network flow model where a segment and a project is represented by a node and an arc, respectively. Thus, solving for the optimal M&R plan equals to seeking a constrained shortest path (CSP). In particular, an optimal M&R plan corresponds to a path of minimum risks that obeys additional feasibility constraints associated with arcs and path lengths. Finally, we proposed three solution methods for solving the CSP: Lagrangian relaxation (LR), the K shortest path (KSP), and a hybrid algorithm exploiting the simulated annealing (SA) framework to select segments to be treated and then solve for their M&R plans by LR. Results of our computational experiments indicate some of our proposed heuristic solution methods are both effective and efficient.

**Key words:** Pavement rehabilitation, Integer program, Constrained shortest path, Lagrangian relaxation, K shortest path.

## INTRODUCTION

In order to have safer road of better pavement quality, the maintenance and rehabilitation (M&R) projects that treat the damaged road pavement surface are conducted regularly. Each road can be divided into a set of contiguous segments. Suppose each road segment is associated with two parameters: risk and cost, where the former represents the level of danger it may cause, and the latter denotes the budget required to treat that segment. If a road segment is treated, then it becomes riskless. Due to the limited budget, only part of the damaged pavement surfaces can be treated. In practice, all the road segments within the same M&R project have to receive the same treatment, and the treatment cost is determined by the largest one within the project. Thus more projects save more wasted treatment costs. Since each project is also associated with a fixed cost of transporting treatment equipment, this suggests fewer projects to be beneficial. Thus, how to select which part of a road for which level of treatment becomes an important and complicated issue. We investigate such an optimal pavement rehabilitation planning problem that seeks an optimal treatment plan over part of road segments with limited budget such that the total risks of a road are minimized.

Most previous pavement optimization research focus on determining the optimal treatment schedule of individual segment (Ouyang & Madanat, 2006) or the optimal treatment cost distribution corresponding spatial relationship for all segments (Tsai, Yang, & Wang, 2006; Yang, Tsai, & Wang, 2009; Wang, Tsai, & Li, 2011) with unlimited budgets. This thesis extends the work of Wang, Tsai, & Li (2011) by adding a budget constraint and the flexibility of treating some instead of all road segments. Unlike the original segment clustering problem addressed by Wang, Tsai, & Li (2011), our problem can be viewed as a

constrained shortest path problem (CSP).

## MATERIALS AND METHODS

First, we may treat each segment as a customer and a pavement project composed by a set of contiguous segments receiving the same M&R plan as a warehouse. Then our problem becomes an uncapacitated facility location problem (UFLP) by integer program (IP). In order to apply the so-called string property that guarantees each M&R project to be composed by contiguous road segments, the IP model contains many constraints and is very hard to solve.

Then, based on the work of Wang, Tsai, & Li (2011), we give a network flow model for the same problem in which a segment and a project is represented by a node and an arc, respectively. Thus, solving for the optimal M&R plan equals to seeking a constrained shortest path (CSP), where an optimal M&R plan corresponds to a path of minimum risks that obeys additional feasibility constraints associated with arcs and path lengths.

Finally, we have proposed and tested three heuristics for CSP: Lagrangian relaxation (LR), the K shortest path (KSP), and a hybrid algorithm that exploiting the simulated annealing (SA) framework to first select segments to be treated and then solve for their M&R plans by LR. LR iteratively seeks optimal Lagrangian Multiplier for the relaxed budget constraint, and solves shortest paths. On the other hand, KSP simply generates the $1^{st}$, $2^{nd}$, …, until $K^{th}$ shortest path that satisfies the budget constraint. We observe the key to speed up the solution procedures may rely on the decisions of selecting the right road segments for treatment. The hybrid algorithm tries to converge a good guess on which segments to be selected for treatment by techniques of SA, and then solves for a shortest path problem. To speed up the hybrid algorithm, denoted by LR+SA, we use the LR solution as its initial solution.

## RESULT AND DISCUSSION

We generate several simulated datasets to test performance of LR, KSP, LR+SA in solving small-, medium-, and large-scale cases, each with 10 random problem sets. LR, KSP, and LR+SA are implemented in C language, compiled by VisualC++2012. To validate the efficiency and effectiveness of proposed solution methods, we also solve the UFLP formulation (GRB1) and CSP formulation (GRB2) by Gurobi 5.6.2. All the experiments are conducted on a personal computer with Windows 7, 8GB RAM, and Intel Core Duo2 CPU of 1.86GHz.

**Table 1. The average computational time of all methods**

| Method | Small cases | Medium cases | Large cases |
|---|---|---|---|
| **GRB1** | 328.8s | - | - |
| **GRB2** | 0.3s | 7.2s | 608.8s |
| **KSP** | 751.8s | - | - |
| **LR** | 0s | 0.27s | 14.87s |
| **LR+SA** | 0s | 0.27s | 20.9s |

**Table 2. The average probability of obtaining optimal solutions**

| Method | Small cases | Medium cases | Large cases |
|---|---|---|---|
| **GRB1** | 100% | - | - |
| **GRB2** | 100% | 100% | 100% |
| **KSP** | 100% | - | - |
| **LR** | 80% | 60% | 13.33% |
| **LR+SA** | 90% | 73.33% | 30% |

Table 1 and table 2 show that GRB1 and KSP could not converge to the optimum within 10000s in medium and large cases; GRB2 always obtain the optimal solution within an acceptable time in all cases; LR+SA performs better than LR with more calculated optimal solutions within shorter time.

## CONCLUSION

Our proposed methodology can effectively spend the limited budget to obtain an optimal pavement rehabilitation plan with least risks under a budget constraint. We have proposed an UFLP and a CSP integer programming formulations, as well as 3 solution methods based on LR, KSP, and LR+SA which is a hybrid algorithm exploiting an SA framework. Our computational experiments indicate the CSP formulation can calculate exact optimal solutions with acceptable running time, and LR+SA could improve the solution obtained by LR. For future research, we suggest investigation on a temporal segment clustering problems by extending the one-dimensional segments clustering to a two dimensional segments clustering on a graph, or using time as another dimension, so that how the M&R decision made in previous time period may be taken into consideration for a long-term pavement rehabilitation planning.

.

# 誌謝

這篇論文可以順利的完成，首先要感謝我的論文指導教授王逸琳老師。感謝您不厭其煩地培訓一個資管背景的學生，讓對於最佳化領域很陌生的我一步一步地熟練，在研究遇到困難時，老師總能適時給予許多指點，使我從問題上學會更多新的知識，在您細心的指導之下，論文逐漸成型，如今可以順利完成。此外，老師也給了我很多生活瑣事方面的教導與建議，讓我處理事情時更加成熟與靈機。

接著，要感謝 ilin lab 的所有成員。感謝學長姐關於程式撰寫以及作業研究的指引，讓我很快就上手。感謝新來的學弟妹們，你們歡樂的個性總讓我可以以樂觀的心態去面對研究的壓力。

最後，也要感謝我的家人和朋友，這兩年來始終鼓勵我，在我遇到挫折時，給予許多精神上的支持，讓我可以專心完成學業與碩士論文。

# 目錄

# 表目錄

# 圖目錄

# 第一章

# 緒論

本章共分 4 節，首先描述本研究之研究背景和研究動機與目的，接著定義出研究問題與範圍和論文架構

## 1.1 研究背景

一個國家的經濟發展，貨暢其流非常重要，而其中公路的運輸則扮演舉足輕重的角色，所以如何提升高公路的鋪面品質就變成一件很重要的工作。在台灣，交通部積極地推動「路平專案」以提升道路品質，而美國也努力地推動「Fix-it-first」的政策，以優先考慮維修既有的公路、橋樑和公共交通系統等基礎設施，使得城市更安全、更符合現代化的要求。

在道路修繕方面，道路的鋪面品質會因使用量增加而變差，加上交通流量逐年增長，導致道路生命週期越來越短。但政府每年投入於道路的預算有限，無法使所需改善之道路皆獲得足夠財源來進行修復。由於地方政府財政拮据，因此地方之運輸管理單位必須借助更有效率的資源運用、縮減運輸服務範圍、增加過路費、找尋其他的財源管道等方法來改善道路的鋪面品質。

## 1.2 研究動機與目的

近年來，雖然公路養護開始受到重視，養護經費也逐漸增加，但其增額還是有限。因此，必須有良好的公路養護規劃策略才能善用拮据的經費以減少未被維修路段的總危險程度。公路養護策略在作業研究領域不是新的議題，但其相關文獻大部份只針對單一或近似的鋪面狀況之假設下，探討養護時段的選擇、養護作業對公路壽命的影響、養護作業排程規劃，並從而達到整修養護成本最小化之目的。目前為止，很少相關研究同時針對實際公路的每一個路段的破損狀況差異以及資金不足的現狀加以探討。

一個鋪面養護與修復計劃(Maintenance and Rehabilitation; M&R)通常要負責修復一系列連續的路段鋪面，而可藉由鋪面狀況年度調查所提供的資訊得知鋪面的破損型態、嚴重性、及範圍。Álvarez et al(2007)在研究中指出，一個路段可能會存在幾個破

損型態，因此必要使用單一指標衡量該路段的所有破損型態。其中， PCI(Pavement Condition Index; PCI)指標是根據路面的破損型態、嚴重性、數量等的目視調查結果所計算出來的一個數字指標，它的範圍由路面完全破損的 0 分狀況至路面完好的 100 分狀況(如表 1-1)。在路面的破損程度確認之後，M&R 計劃必須依各路段狀況不同的評分來決定其應採用之整修方法，不同的整修方法對 M&R 計劃成本而言有直接影響。

表 1-1：PCI 對鋪面破壞程度的評分表

| **PCI 值** | **等級(Rating)** |
|---|---|
| 85~100 | 最佳(Excellent) |
| 70~85 | 很好(Very good) |
| 55~70 | 好(Good) |
| 40~55 | 尚可(Fair) |
| 25~40 | 差(Poor) |
| 10~25 | 很差(Very poor) |
| 0~10 | 失敗(Failed) |

由於政府的修路預算資金不足已為常態，每年勢必會犧牲某些路段的修復機會，而實務上政府亦必須將有限的資源優先分配至全國性或地方性的重要道路。對於此取捨動作，只有規劃出適當的策略，才能夠使得損失降到最低。此外，對於被捨棄修復的路段，須做好其未來的路段品質控管，理論上越早修復越好，避免路段品質降到最低等級而被迫重建。

本研究將探討如何在維修預算有限的情況下，考慮業界行之有年的修復習慣與規範，決定各路段是否該修復及其最佳修復方式，以達到總風險程度最小之目的。若給定的預算充足，可以用來修復所有路段，則總風險為 0；反之，若給定的預算不足，則可能優先修復危險性較高的路段，以降低總風險程度。本研究將針對連續的路段，考慮不同路段有不同的風險程度與維修成本，提出適當的鋪面養護策略，此策略將有效分配各路段至各分組維修計劃，使得同組計劃內的每個路段都能夠獲得最好的修復措施。

## 1.3 研究問題與範圍

實際的案例所負責養護的道路型態非常多，本研究只針對無分叉且單一車道的高速公路進行分析，再依據各地的鋪面養護規範將所選的道路分成數個連續的路段，而路段的長度界於 0.1 至 1 英里。依據在平面上的位置將固定長度的路段一線排開而得到一維空間的路段集合。

本研究所探討的路段依據路面損壞程度而包含其風險程度及維修成本等兩個屬性。針對一維空間的路段，在維修預算有限的情形下，進行探討路段分組問題，以達到總風險程度最小之目的。其中一群組代一個M&R計劃。此路段分組問題與「設施區位問題」的概念非常近似。舉例來說，我們可將路段視為顧客, 而組別視為廠商，則某種分組方式下某路段所需的維修成本可被視為某個廠商為滿足某顧客的需求所引發的運輸成本;若某路段未被列入維修分組的話,該路段的風險程度將被累積計算，這就像某顧客未被廠商服務的話，該顧客的缺貨成本將被累積計算；路段分組問題探討的是如何在不超過維修預算下，建立總風險最小的路段分組方式，而設施區位問題探討的是如何在建置與運輸成本不超過預算下,建立總缺貨成本最小的廠商設施及顧客分配方式。由於本研究不限制一個路段分組可包含的路段個數上限，因此我們可將此路段分組問題視為一個「無容量限制之設施區位問題」，並建構整數規劃模式來求解。

道路養護作業必須遵守一些行之有年的特殊業界修復規範與習慣,譬如目前之鋪面風險程度高於門檻值的路段一定要被修復;被劃分於同一修復分組的待修路段總數必須足夠多，且這些路段必須為連續相鄰；此外，同一分組中的所有路段一定要採取其分組中修復花費最昂貴路段的修復方式來全面修復之。其中，限定維修群組只能包含相鄰的連續路段,以避免修復時經過其他包商所負責的路段使得該路段的損壞更加嚴重的特殊限制，此一連續路段分組的規範其實是一維分群問題中的「成串性質」(String Property)，此成串性質雖可減少問題的可行解數量，但會導致所建構之整數規劃模式包含許多限制式而求解耗時。雖然被修復的路段其風險程度將被歸零，但不同的修復分組方式亦將引發不同的維修成本,因此求解最佳的修復分組方式為一個複雜的組合最佳化問題。

大多數文獻指出，無容量限制的設施區位問題是個複雜的NP-Hard最佳化問題，

所以不容易獲得最佳解。過去的研究常使用啟發式演算法及近似解演算法來求解此問題，本研究則將以Wang, Tsai, & Li(2011)所發表的方法為基礎，將原路段分組問題轉換成求解一個「具額外限制式之最短路徑問題」。

## 1.4 論文架構

論文之架構如下：第一章為緒論，介紹研究背景、研究動機與目的、研究問題與範圍含論文架構；第二章為文獻探討，簡介無容量限制的設施區位問題、集群問題、鋪面養護規劃最佳化問題、含額外限制式的最短路徑問題之相關文獻、模擬退火法；在第三章中，我們首先建構無容量上限之設施區位整數規劃模式，接著解釋如何將原問題轉換成一含額外限制式的最短路徑問題，並解釋如何以拉氏鬆弛法及第 K 短路徑演算法求解該問題，最後說明如何使用模擬退火法加速求解，以及如何使用拓樸排序法求解最短路徑子問題；第四章將進行數值分析與討論；第五章則總結本研究成果以及提供未來可研究的方向和建議。

# 第二章

# 文獻探討

本章主要探討無容量限制的設施區位問題、集群問題、鋪面養護規劃最佳化問題、含額外限制式的最短路徑問題、以及模擬退火法等五個議題的相關文獻。在無容量限制的設施區位問題部分，我們將介紹無容量限制的設施區位問題的定義及應用，並說明此問題常見的求解技術;在集群問題部分,我們將介紹整數規劃在集群分析之應用、成串性質、一維的集群問題；在鋪面養護規劃最佳化問題，我們將介紹鋪面養護規劃最佳化問題的相關文獻；在含額外限制式的最短路徑問題部分，我們將介紹求解含額外限制式的最短路徑問題最常用的兩種技術，分別為拉氏鬆弛法以及第 K 短路徑演算法；最後在模擬退火法部分，則介紹其定義及應用。

## 2.1 無容量限制的設施區位問題 (Uncapacitated Facility Location Problem; UFLP )

### 2.1.1 無容量限制的設施區位問題之簡介

設施區位問題是探討在多個潛在場站中,考慮每一個場站與各需求點之間的運輸成本與設施固定成本下，如何選取總成本最小的最適區位。若所選取的場站可以服務無限多個需求點，則該問題為「無容量限制之設施區位問題」(UFLP)。

Verter (2011)在書籍中指出UFLP可視為簡單工廠設立問題(Simple plant location problem; SPLP)。研究者將簡單工廠設立問題定義如下：假設有一潛在廠址設立之集合$F$和一顧客之集合$C$；建造廠址需花費一非負之固定成本$f$；每個廠址$j(j \in F)$所服務每個顧客$i(i \in C)$皆產生一固定之運輸成本$c_{ij}$（transportation cost）。此外，假設每個選擇開啟的潛在廠址都具有無限之服務能力。此類型問題的目標在於應選擇設立哪些潛在的廠址以使總成本達到最小。

Krarup and Pruzan (1983)全面分析UFLP的過去相關研究並發現此類問題的可行解具有單一指派性質(Single assignment property)以及與集合覆蓋問題(set covering problem)、集合包裝問題(set packing problem)、集合分割問題(set partitioning problem)

之間的關係。

UFLP可視為作業研究之重要子領域，因此在作業研究中的文獻及應用相當廣泛，如：機器排程及資訊檢索(Hansen and Kaufman, 1972)、銀行帳戶設立(Cornuejols,Fisher, and Nemhauser, 1977)、經濟批量(Bilde and Krarup, 1977)、集群分析(Mulvey and Crowder, 1979)、投資組合管理(Beck and Mulvey, 1982)、網路中心(hub)之建立(Mirzaian, 1985)、物流系統設計(Klose and Drexl, 2005)等等。而隨著資訊與科技的快速演變，UFLP的概念也被運用至高科技領域，如：自配置式的無線感測器網路的管理和設計(Frank and Romer, 2007)、計算生物學 (Dueck et al., 2008)、電腦視覺 ( Lazic et al., 2009)等等。

### 2.1.2 無容量限制的設施區位問題之求解技術

UFLP 由 Cornuejols et al. (1990)證明為具有 NP-Hard 性質，且實務上問題之變數數目龐大，實為一大型整數規劃問題，故早期發展多以啟發式演算法為主，如：貪婪法(Greedy Heuristic)、Dual ascent algorithm、禁忌搜尋法(Tabu search algorithm)等。

Kuenhn and Hamburger (1963)最早提出無容量設施區位問題之求解方法，其研究將運用貪婪法(Greedy Heuristic)與交換啟發法(Interchange Heuristic)來求解大型物流配送網絡。而 Cornuejols et al. (1977)則分析貪婪法、交換啟發法及拉氏鬆弛法等三種方法的相對誤差及上下界的鬆緊程度，其測試的結果說明在最壞的情況下這三種演算法可提供較佳的界限(tight bounds)。Nemhauser et al. (1978) 將 UFLP 的理論延伸到組合最佳化問題(combinatorial optimization problem)，再進一步採用貪婪法、局部改善啟發式及線性鬆弛方法進行求解，最後將探討各演算法所求的近似解之品質。

Erlenkotter (1978)在研究中基於線性對偶問題的理論提出改善下限值的一個簡單步驟來求解 UFLP，若此步驟無法獲得最佳的下限值，則在下一個迴合須採用分枝界限法(branch and bound)求解。此研究的求解時間優於當時其它演算法，Erlenkotter (1978)也針對大型的例題進行數值測試與分析，並指出在不進行分枝界限法的特殊情況下，總運算時間小於 0.1 秒。相較當時的其它方法，此方法在解決特殊例題的求解時間方面確實較優越。針對不同的情境，更早期 Blide and Krarup (1977)也基於線性對偶問題的理論選擇分枝界限法來探討 UFLP，但沒有使用改善解的方法以及與其他方法做比較。而 Körkel (1989)進一步改善 Erlenkotter (1978)的方法使得求解效率大幅提

升。Gao and Robinson (1992)在 Erlenkotter (1978)方法的基礎之下，發展出新的演算法可解決兩階層場站的 UFLP。

Al-Sultan and Al-Fawzan (1999)和Sun (2006)皆指出禁忌搜尋法（Tabu search）對於求解UFLP小型例題的優點，如：取得正確的最佳解、高品質的解、求解時間大幅改善等。而Ghosh (2003) 運用禁忌搜尋(Tabu search)及有記憶完全區域搜尋(complete local search with memory）方式求解由 Körkel文章中所提方式產生之例題，並且得到中型問題及大型問題其最差情形分別落在0.075%倍及0.0345% 倍之中。

由於近似解演算法的優點是在可承受的範圍內，以比精確求解消耗更少的資源進行求解近似解，因此其方法被許多學者採用來簡化 UFLP 的複雜性。Barahona and Chudak (2000)藉由近似解演算法（approximation algorithm）和體積法(volume algorithm )所結合的新啟發式演算法進行求解具有 3000 個潛在場站以及 3000 個顧客的大型 UFLP，並保證所求的近似解與最佳解之間的誤差不會超過 1%。而 Shmoys et al. (1997) 指出在研究中所求出的結果不會超過最佳解的 3.16 倍。Guha and Khuller (1999)也經由集合覆蓋問題(set cover problem)之簡化，說明 UFLP 為 Max SNP-hard，也說明了近似解演算法最多只能保證至 1.463 倍。

此外，Lazic et al. (2010) 以機率理論(probability theory)與圖形理論(graph theory)所結合 Graphical model 的推論觀點建立 UFLP 模型，接著利用最大乘積型的訊息傳送演算法進行求解。

## 2.2 集群問題(Grouping/Clustering problem)

### 2.2.1 整數規劃在集群分析之應用

由於一般常用的集群分析方法，如平均連鎖法(Average Linkage Clustering)、華德氏最小變異法(Ward's Minimum Variance )、K 平均數法(K-means)、因子分析法(Factor Analysis)等，所求出來的分組結果通常只是局部的最佳解(local optimal solution)，而非真正的最佳解(global optimal solution)。若只對小型且簡單的例題做分組，則以上的方法還可以應付；但若需要求解精確的解，而又無法將分組情形一一考慮的話，則必須使用整數規劃來進行集群分析。

因此，張自強(1987)使用整數規劃，尤其是 0-1 線性整數規劃的方法來分群。其

原因有二：

1. 觀察集群分析問題，我們可以發現它基本上即為一個整數規劃的問題，因為可以假設若第 $i$ 個元素屬於第 $j$ 個集群則 $x_{ij}=1$，反之 $x_{ij}=0$。因此，此種類型的集群問題是求得一個滿足我們目標的 $x_{ij}$ 矩陣，$i=1,2,...,N$，$j=1,2,...,M$，且 $x_{ij}\in\{0,1\}$，其中 $N$ 為元素個數，$M$ 為集群個數
2. 以 0-1 整數規劃方法來解決問題的另一個好處是它不但可以用一般整數規劃方法(如：分支界限法)外，有些特殊的電腦軟體可轉用來解決 0-1 整數規劃之問題，且其在電腦上所需要的執行時間也比使用一般整數規劃程式要來得少。

早期 Fisher (1958)首次將整數規劃的概念應用在集群分析問題上。而研究者想要分析的是如何將母體分割成較小群聚，且同一群聚的元素之特質較近似。因此，為了達到同組元素的相似度最大之目的，研究者將組內變異平方總和最小化做為研究的目標式，接著以全部列舉法列出所有分組的可行解，再從中找出使得目標式最小的最佳解。此研究針對 200 個元素及 10 個集群的例題進行測試，測試完成可獲得最佳解。此外，Fisher (1958)在文獻中發現與目標式有密切相關的特質：若 $m$ 個元素按大小排列，而 G 是將 $m$ 個元素分割成 $n$ 個小集群使得組內變異總和最小的某種分割方法，且 $n<m$，則所分割出的集群將會是連續的群聚。

### 2.2.2 成串性質(String property; SP )

在人類生活中，許多事物、現象皆具有連續性的特質，而連續性特質不僅存在於一串數字，如人類的年齡、身高等，只要該事物、現象具有前後順序的特性也被稱為具有連續性，如：路段前後位置、羅馬文字等。

假設有 $m$ 個連續元素將被分配到 $n$ 個連續群組，其 $n\le m$，若 $n$ 個群組中沒有包含任何跨群組的元素，則此分群具有成串性質。反之，若 $n$ 個群組中只要有一個群組包含跨群組的元素，則此分群不具有成串性質。如 2-1 圖所示，假設有六個連續元素 $\{i_1,i_2,...,i_6\}$ 將被分配到 3 個連續群組 $\{C_1,C_2,C_3\}$，圖(a)的分群使得 $\{C_1,C_2,C_3\}$ 沒有包含任何跨群組的元素，故此分群具有成串性質，而圖(b)的分群使得 $\{C_1,C_2\}$ 包含 $\{i_2,i_3\}$ 跨群組的元素，故此分群不具有成串性質。

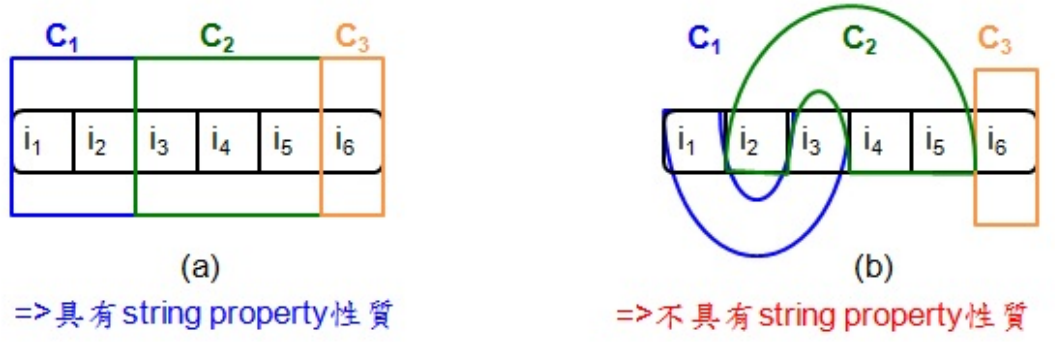



圖 2-1：成串性質的示意圖

成串性質與連續性的關係在集群分析問題是個很重要的發現。Vinod (1969)的研究中強調 Fisher (1958)所定義的集群問題是一個整數規劃問題，並認定元素的連續性是追求最小化組內離均差平方和的關鍵條件。因此， Vinod (1969)將針對 Fisher (1958)的發現提出重要的 lemma： 若分群具有成串性質，那麼任選 3 個連續元素$\{i_1, i_2, i_3\}$，其中$\{i_1, i_3\}$被分配到同一群組，則$\{i_2\}$也與$\{i_1, i_3\}$同組。

張自強(1987) 將 $m$ 個連續元素是否被分配到 $n$ 個連續群組的 0-1 變數建立 $x_{ij}$ 矩陣(n x n)，其 $n = m$，若在對角線以下(below the diagonal) 各元素滿足下列條件，則稱 $x_{ij}$ 矩陣具有成串性質。

$$x_{jj} \geq x_{j+1,j} \geq \ldots \geq x_{N,j} \qquad for\ j = 1, 2, \ldots, N$$

### 2.2.3 一維的集群問題

一維集群問題早期常以動態規劃求解。動態規劃常用於求解具有某種最佳性質的問題，而此演算法的優點是將計算過的結果記錄下來以節省時間，避免大量的重覆計算。若一維集群問題具有成串性質(string property)，則利於動態規劃的求解效率。因此，Rao (1971)在成串性質(string property)的假設之下，針對一維的集群問題建立最佳化問題的數學模型，接著運用動態規劃演算法來求解這類問題，並指出該演算法會計算出最佳的群組個數而不需要在實例中給定一個固定的群組個數值。而 Bellman (1973)在研究中假設所要分類的元素集合為自然數集合，並分析動態規劃演算法對此特殊的一維集群之求解效果，最後在測試數值部分進行一維例題與多維例題的比較。此外，Jensen (1969)認為，儘管所需分類的元素相當小，從所有可能的組合中找出最佳解的

完全列舉法（total enumeration method）對求解集群問題而言還是沒有效率的一個方法。因此 Jensen 透過動態規劃演算法的求解特點來改善完全列舉法的執行步驟，從而獲得更高效率，並保證此方法可以得到已收斂的最佳解。

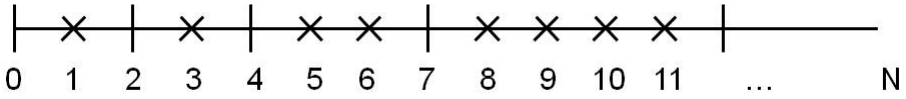



圖 2-2：一維集群的示意圖以自然數集合為例

動態規劃對於求解一般的集群問題確實很有效率，但在較複雜的情境下未必是個好方法。即使在處理一維的分類例題時，該演算法所花的時間仍是多項式時間。Hansen and Jaumard (1997)證明動態規劃在求解具有巢居性且單變量分布的元素之分類問題時其效率甚低。而單變量分布的性質將使得最佳解可能不滿足成串性質(string property)條件。

Novick (2009)以統計概念為基礎推導出全新的集群方法，並指出雖然此方法所求得之一維例題的最佳解可滿足 string property 條件，但一維的集群問題仍是 NP-Hard 問題。此外，Novick 在文獻中證明 string property 的特質不但使得可行解的數量大縮減並且可簡化尋找最佳解的流程。

## 2.3 鋪面養護規劃最佳化問題之相關研究

預算分配在公路養護規劃中十分具挑戰性，許多作業研究學者陸續探討鋪面養護規劃的議題。由於鋪面品質會隨著時間而衰退，而在衰退程度不同的情況下，其修復成本將各有不同。因此 Ouyang & Madanat (2006)在連續及有限的時間內針對鋪面隨著時間而惡化的情形，探討整修的最佳時段以及最適合的整修面層作業之等級，以達到最終修復成本最小之目的。此研究將計算出一個路況的門檻值，並假設若目前的路況評分大於門檻值（代表路況品質較差），則必須儘快修復；反之，修復作業可延遲。一旦公路面層被決定修復，修復的作業等級及修復成本會因為當時的鋪面品質而有所不同。舉例來說，假使某路段的鋪面狀況為重度破損，則其修復必須採取較高等級的作業，導致修復成本也因此增加。此外，某時段所選擇的作業等級也會影響該鋪面下一個階段的惡化函數，譬如使用等級越高的修復作業(亦即修復得更徹底)，該路段之鋪面惡化速度亦將趨緩。

大部分鋪面養護規劃之相關研究只著重於最佳化鋪面養護時間問題以及發展求

得精確解的一些技術，僅有少數研究者探討鋪面的空間性及路段評分的變異。Tsai, Yang, & Wang (2006)和 Yang, Tsai, & Wang (2009)依據路段在一維空間上的連續位置而提出全新的路段集群方法，利用 fuzzy c-means (FCM)技術尋找最佳群組個數、最佳路段分配及組內的最佳中心評分值，使得組內的路段評分至中心評分值之距離最小，因此間接地獲得最終修復成本最小。此方法的求解步驟如下：

(1) 首先根據強制路段分隔和非強制路段分隔的限制而設計數個情境。

(2) 計算每個情境的群組個數的下限上限。

(3) 初始化群組與路段之間的關係。

(4) 運用 FCM 以更新路段對各群組的歸屬程度、各組的中心評分值、目標式、目前的群組數量。

(5) 針對違反限制式而產生懲罰成本的群組進行調整路段分配。此步驟將重複執行直到各組的懲罰成本歸零。

(6) 計算並記錄目前分配方式的所產生的修復成本，接著將群組的個數加一，並重複以上的步驟直到群組個數到達上限。

(7) 選擇使修復成本最小的分配方式。

(8) 依照以上的步驟求解其他情境。

此方法雖然具有非常大的實際應用性，但無法保證所求得之解的品質，且運算時間可能耗費太多。

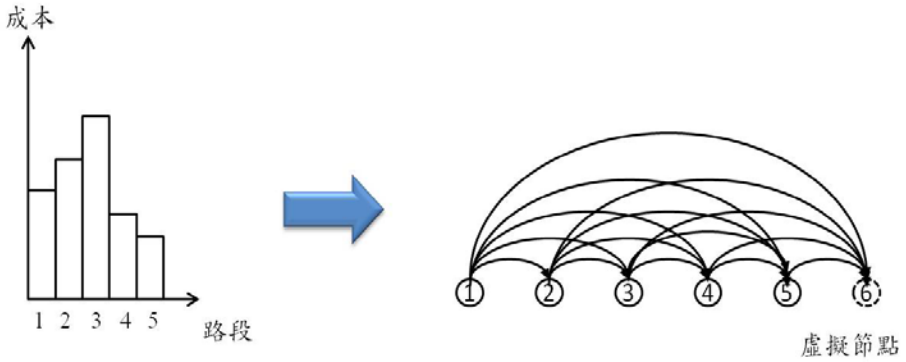



圖 2-3：一維連續路段集群問題轉換成一般網路問題

Wang, Tsai, & Li (2011)藉由 Tsai, Yang, & Wang (2006)和 Yang, Tsai, & Wang (2009)所發表的集群一維空間概念，將重點目標鎖定於路段維修成本最小化，並透過混合整數線性模型求解路段分配問題。不同的路段可能有不同的破損程度，而同組的

路段皆以組內破損程度最大的單一路段維修成本作為每段維修成本的計算參考，也就是同組的路段將獲得最好的維修措施。由於該研究所設定的問題情境將使其混整數線性規劃模型耗費太多的求解時間，即使使用目前求解最佳化問題的最先進專業軟體CPLEX，當路段個數 $m=50$ 時，還是有可能因為運算過久而被停止，因此也無法在短時間內獲得最佳解。此外，Wang, Tsai, & Li (2011)還發展出新的方法，將原問題轉換成一般的網路問題(如圖 2-2)，並運用最短路徑演算法求得問題最佳解。該解法可以大幅提升此問題的求解效率，且與 CPLEX 軟體得到相同的最佳解。舉例來說，若原問題的最佳解為路段①②③同一組，路段④⑤同一組，則對應網路圖的最短路徑為圖 2-3 的路徑 1-4-6，其中一個弧即對應一組路段分群。

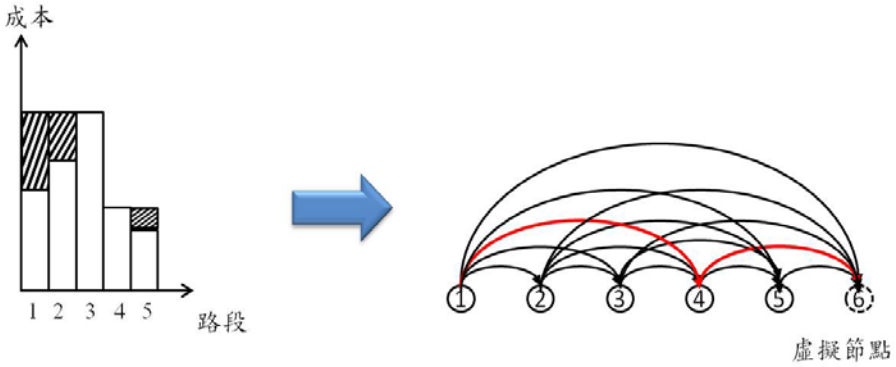



圖 2-4：路段集群的結果與網路圖的結果

## 2.4 含額外限制式的最短路徑問題(Constrained shortest path; CSP)

一般純網路問題包含最短路徑問題、最大流量問題、最小成本網路流動問題等，這些問題的研究與應用相當廣泛。但在現實生活中，物質(人或物品)在同一網路中之相關節線流動時，常會受到許多其它因素的限制，如：資源、時間等。因此， 我們在本節將探討含額外限制式的最短路徑問題，並以最短路徑問題之特例做詳細介紹。

Ahuja et al.(1993)在書中假設網路中的每一條節線 $(i, j)$ 皆包含兩個屬性，分別為節線成本 $c_{ij}$ 以及流動時間 $t_{ij}$ ，最終的目的是能找出從起點 $s$ 到訖點 $t$ 的最短路徑，且該路徑必須在總流動時間 $T=10$ 內完成任務，並以整數規劃模式描述如下：

$$\text{Minimize} \sum_{(i,j)\in A} c_{ij}x_{ij} \qquad (2.1)$$

$$\sum_{(i,j)\in A} x_{ij} - \sum_{(j,i)\in A} x_{ji} = \begin{cases} 1, & \text{for } i = s \\ 0, & \text{for } i \in N - \{s,t\} \\ -1, & \text{for } i = t \end{cases} \qquad (2.2)$$

$$\sum_{(i,j)\in A} t_{ij} x_{ij} \le T \qquad (2.3)$$

$$x_{ij} = 0 \text{ or } 1 \qquad \text{for all } (i,j) \in A \qquad (2.4)$$

其中(2.1)目標式針對被選取的節線$i$考量節線成本總合之最小化，限制式(2.2)為流量守恆限制式，限制式(2.3)為總流動時間的限制式，限制式(2.4)為$x_{ij}$是二元變數的限制式。以上所描述的情境是含「時間」限制式的最短路徑問題。含額外限制式之最短路徑問題在文獻中已被證實為一NP-hard 的問題，目前無法在多項式時間（polynomial time）內求的最佳解。

求解一般最短路徑問題時，最常使用的是Dijkstra 演算法，但若再加上額外限制式時，將無法以單純使用Dijkstra 演算法的方式去求解。求解CSP問題最常用的兩種技術為拉氏鬆弛法 (Lagrangian Relaxation; LR)及第K短路徑演算法 (K shortest path algorithm; KSP)。

### 2.4.1. 拉氏鬆弛法 (Lagrangian Relaxation; LR)

拉氏鬆弛法是數學規劃中的一個技巧，主要針對某些較困難的限制式賦予拉氏係數，針對超出限制範圍的可行解部分予以懲罰成本，並將限制式「放鬆」至目標函式中，藉由簡化原極小化問題來求得原問題的極小化下限值，接著試圖找出最佳的拉氏係數，重複這些步驟以求解原問題。拉氏鬆弛法求解的過程容易了解且直覺（可參考Ahuja et al. (1993) 書中之第十六章），配合次梯度法即能找出一條與最佳解接近的路徑，雖然產生的解可能與真正最佳解有所差異(即存在有對偶間隙)，但可提供好的上下界幫助其它演算法進一步收斂至更好的解。

Xiao et al. (2005)在研究中指出$x_{ij}$的0-1整數屬性使CSP問題求解上較困難，所以研究者放鬆二其元變數的限制式，將其變更為$x_{ij} \ge 0$，此問題則稱為釋限型的CSP問題(Relaxed CSP)。釋限型的CSP問題可藉由求解其對偶問題而得到放鬆限制CSP 問題的最佳解，而該解所對應的目標式亦為CSP問題最佳目標值的下限。Handler and Zang (1980)則將最短路徑問題加上背包問題型式(knapsack-type )的限制式，接著透過第K

短路徑演算求得此問題的最佳解。但是K值越大會導致求解時間越長，因此研究者利用拉氏鬆弛法改善K值，以提升演算法的效率。Carlyle et al. (2008)提出一種整合拉氏鬆弛法和展開接近最短路徑（near-shortest path；NSP）的LRE（Lagrangian Relaxation and Enumeration）方法，藉由展開路徑而不斷更新對偶問題之目標上限值，進而拉近對偶間隙（duality gap）。

### 2.4.2. 第K短路徑演算法 (K shortest path algorithm; KSP)

當需要求解滿足某些條件的最短路徑問題時，原始的最短路徑通常會違反條件需求，因此可能需要找尋第2短、第3短…等第K短路徑並一一檢視，直到條件滿足為止，此即為第K短路徑演算法。第K短路徑演算法最早由Hoffman在1959年所提出，而應用最廣泛的演算法則是Yen (1971)所發表的演算法。Yen在研究中針對無迴圈路徑的情況下，提出尋找起訖點間的第K條最短路徑的方法，其網路圖上節線的權重可用距離、時間、或成本等其它要素替代。Yen (1971)的方法說明第二短路徑須依據第一短路徑而求得，因此在搜尋第K短路徑之前須確認第一短至第K-1短的路徑之結果都已知。KSP 演算法可以應用在模型適合度分析、敏感度分析、求取多重可行解以及具有額外求解限制的路徑選擇問題。

雖然Yen (1971)是第一個有系統及清楚地提出第K短路徑的求解方法，但該方法的缺點是在n個節點的網路中求前K短路徑的複雜度為$O(Kn^3)$。因此許多學者推出各種演算法以改善Yen方法的複雜度。Lawler (1972)修改Yen (1971)的方法，增進了常數係數，並證明新方法在有向網路圖求解KSP問題時只花O(Kn(m+nlogn))時間。之後，Katoh and Mine (1982) 亦修改了 Yen 的方法應用於無向圖，而其求解時間為O(K(m+nlogn))。此外，Fox (1975)藉由Dijkstra演算法結合priority queue的方式在符合acyclic paths 和有向圖條件之下，求解KSP問題，並說明此方法只需要O(m+Knlogn)時間即可求出最佳解的結果。

對於具有額外限制的路徑選擇之應用，傳統的方法常採用窮舉法列出起訖點間所有可能的路徑，接著進一步檢查符合額外條件的可行解，最後對這些可行解進行遞增排序，並從中可取得前K短路徑的結果。但實際的網路龐大規模以及複雜的限制會導致傳統方法的求解時間過長，甚至面臨超時問題而被CPU中止流程。因此，Van der Zijpp and Fiorenzo Catalano (2005)透過設計節線集合的條件以減少搜索路徑數量的方

式來改善傳統方法的CPU處理時間，並保證能在合理的時間內求得最佳解。以Lawler (1976)針對第K短路徑演算法所發表的分割理論為基礎，此方法首先將找出CSP問題的最短路徑，該路徑可滿足或不滿足給定的限制式集合。接著，將最短路徑所包含的所有節線分割成數個較小的節線集合，在分割的過程中，若發現任何節線或節線集合違反預設的節線限制式，則停止繼續分割該集合，因此任何路徑經過這類節線也被歸納為不考慮搜尋的非可行解之路徑集合。此方法會根據子集合所找到的最短路徑而持續分割，直到所有可行解路徑都被搜尋，最後進行排序所有可行解路徑並獲得最佳解。

此外，針對可經過重複節線與節點的另一種KSP類型，Shier (1979)對於允許self-loop 的情況提出了一系列解法，其使用的中心概念相同但配上不同的arc processing 程序，幾種方法各有優劣及其不同的適用情況。

## 2.5 模擬退火法(Simulated Annealing; SA)

模擬退火法最早由Metropolis et al. (1953)根據統計熱力學原理描述一個系統模擬冷卻晶體的過程。其概念是物體加熱至熔化時，會從固態開始轉成液態，此時所有的分子會在液態物體中隨機的自由排列，隨著物體逐漸降溫， 這些分子逐漸趨向較低能量的狀態排列，而在結晶狀態時，系統的能量狀態最低。後來 Kirkpatrick et al. (1983)利用Metropolis所提的概念來求解最佳化問題，而所需最小化的目標值是物體能量。

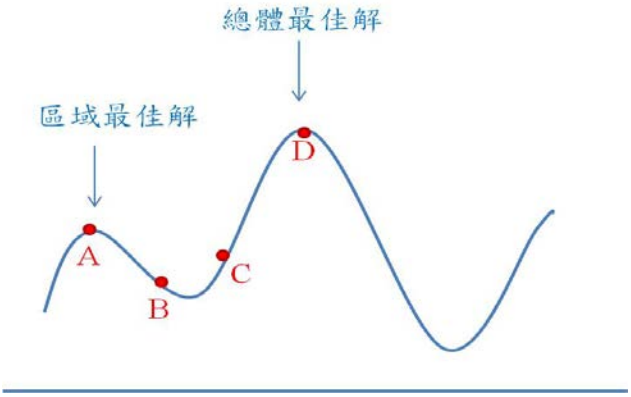



圖 2-5：模擬退火法示意圖

SA 可視為是一種機率搜索的爬山法，一般的爬山法屬於簡單的貪婪搜索法，此方法將從現行解的鄰近解進行搜尋，每次搜尋到更好的解，則將該解設為現行解，直到達到區域最佳解。而在搜索最佳解的過程中，為了跳出區域最佳解，SA 以一定的

機率來接受比目前最好的解還差的鄰近解，並最後找出總體最佳解。如圖所示，A 點為區域的最佳解，SA 的跳越機制將找出較差的 B 點和 C 點，經過 B 和 C 後將達到總體最佳解 D 點。

Maffioli (1987)指出早期 SA 是一種常用來求解具有 NP-hard 性質的最佳組合問題的隨機啟發式搜索方法，而當時最熟悉的最佳組合問題是旅行推銷員問題(Travelling Salesman Problem; TSP)。在 Allwright and Carpenter (1989)的研究中，證明 SA 在大型的 TSP 可求得相當好的解。而 Drexl (1988)也透過 SA 來求解含資源限制的背包問題，此研究利用物品被選或不選之特性來產生物品 0-1 組合，而為了每一次都能產生出可行的 0-1 組合，在鄰近解產生的步驟首先從「不被選」的物品集合隨機選出一個物品轉為「被選」狀態，若前面的選擇使得鄰近解違反限制，那麼必須從「被選」物品集合中隨機選出一個物品轉為「不被選」狀態，直到滿足資源限制為止，此研究的測試結果指出 SA 不但求解時間非常快，而且可獲得最佳解的機率也高。

## 2.6 小結

綜合上述，本研究以 Wang, Tsai, & Li (2011)的模式為主要架構基礎，針對一維連續路段加入路段風險程度以及維修預算上限之考量，探討其路段集群問題。我們亦將先提出一個無容量上限的設施區位問題，並參考 Wang, Tsai, & Li (2011)所提出的模式轉換概念將原問題轉成一個含額外限制式的最短路徑問題(CSP)，因此諸如 LR 與 KSP 等常見之 CSP 解法將被使用於求解本問題。由於整數規劃模式與 KSP 皆需要耗費許多求解時間，我們擬以 LR 之解為基礎，佐以 SA 收斂機制來設計更有效率的律法。

# 第三章

# 路段集群維修問題之數學模式及解法

本章共分成 5 節，首先第 1 節將詳細描述鋪面養護規劃問題與假設，接著第 2 節定義所有參數及變數，並說明此問題的整數規劃模式，第 3 節將以網路模型描述本問題，並認定此問題是含額外限制式的最短路徑問題，第 4 節運用拉氏鬆弛法將含額外限制式的最短路徑問題轉換成一般的最短路徑問題，接著進一步計算最佳拉氏係數，第 5 節運用第 K 短路徑演算法求含額外限制式的最短路徑問題，第 6 節將設計適當的模擬退火法來改善拉氏鬆弛法的收斂解，最後第 7 節將介紹適合求解本研究特殊最短路徑子問題的最短路徑演算法。

## 3.1 問題描述與假設

### 3.1.1. 問題描述

在本研究中，假設一條公路依據地方鋪面養護所規定的路段長度，分成 $m$ 個連續路段，而 $M$ 表示這些路段之集合。公路因為人們使用以及氣候的變化而產生破損，但實際上一個路段可能會包含許多不同破損型態，因此，必要使用單一指標衡量該路段的所有破損型態。假設該指標的數值已知，並將各路段的指標值進行分等，此等級代表路段的風險程度。

依據各路段 $i\,(i=1,2,...,m)$ 的破損程度，我們給定各路段的維修成本 $c_i$ 和風險程度 $r_i$。風險程度值較高的路段代表損壞較嚴重的路段，因此對於道路使用者比較危險。在道路安全性考量之下，我們定義了風險程度的門檻值 $\bar{r}$。此值將決定路段是否一定要修，如下表所示：

表 3-1：路段被選取修復的條件說明表

| 路段的風險程度 | 說明 |
|---|---|
| $r_i > \bar{r}$ | 第 $i$ 路段一定要修 |
| $r_i \le \bar{r}$ | 第 $i$ 路段可修可不修 |

假設有些風險程度小於門檻值的路段不被修復，則剩下的路段將都被修復，且被分到 $n$ 個維修群組，$N$ 表示維修群組之集合。變數 $x_{ij}$ 代表路段 $i$ 是否被分到第 $j$ 維修群組以及變數 $y_j$ 代表第 $j$ 維修群組( $j=1,2,...,n$ )是否被決定設立，此兩類變數即為本研究欲求出之主要決策變數。

每建立一個群組會產生一個固定成本 $f$，此成本會包含器材運送成本、通訊成本、維護中心的設置成本。固定成本會因為被選擇建立群組的數量而影響到總維修成本。若建立太多群組，則固定成本會占大部分的經費，而剩下用來修復路段的經費將減少而導致許多路段不被修，增加公路的總風險，因此必須適當選擇應設立群組的數目。此外，基於行之有年的維修規範與習慣，同一分組 $j$ 中的各路段應該使用同樣的方式來修復，且該方式必須為該分組中所有路段中花費最大（假設為 $Z_j$ ）的路段之修復方式，如此一來即可保證同分組的所有路段皆將使用同樣等級的修復方式以達到同樣的完美路況（亦即風險程度降為 0）。

為了不造成資源浪費，實際上一個養護計劃所需維修的連續路段應至少要有足夠多的路段以及花費。為了讓本研究更貼近現實，我們定義了 $m_l$ 和 $L$ 參數，其中 $m_l$ 將限制各維修群組至少包含 $m_l$ 個路段，而 $L$ 則為各維修群組的總成本下界值。

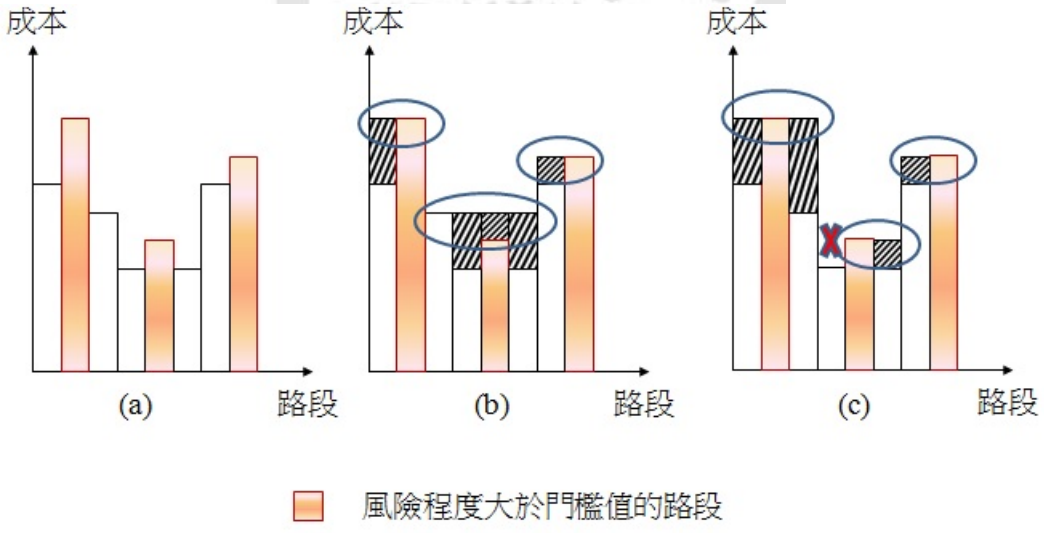



圖 3-1：一維連續路段的各種分群結果

(a)未經過分組(b)全部路段分成三組(c) 全部路段分成三組且有一個路段沒有被修。

本研究以 Wang, Tsai, & Li (2011)之模式為基礎，與 Wang, Tsai, & Li (2011)有兩處最大的差異：(1)本研究之每路段不見得必須被修復，只要其風險程度小於門檻值 $\bar{r}$ 的

路段可選擇不被修復(反之則必須被修復) (2)本研究之目標為極小化總風險，而非極小化總成本。

### 3.1.2. 問題假設

針對本問題的情境，我們列出下列假設與限制：

1. 只針對無分叉且單車道的高速公路進行分割路段
2. 被設立的群組無容量限制。
3. 避免修復時經過別維修群組所負責的路段使得該路段的損壞更加嚴重，被設立的群組將包含連續路段。
4. 路段包含兩種權重，分別為維修成本及風險程度。此兩種權重的值將設為已知。
5. 避免分出過多小規模的群組，給定群組的維修成本之下界以及群組至少包含路段個數。
6. 同組的路段皆擁有相同的養護方法、養護施工機具及基本的管理機制。
7. 當路段被選擇修復時，該路段的風險程度則歸 0。
8. 由於考量路段的連續性，本研究路段的相關變數之編號表示路段的前後位置。

## 3.2 整數規劃模式

### 3.2.1. 參數與變數定義

表 3-2：參數說明表

| 符號 | 說明 | 符號 | 說明 |
|---|---|---|---|
| $i$ | 第 $i$ 路段 | $j$ | 第 $j$ 群組 |
| $r_i$ | 第 $i$ 路段的風險程度 | $m_l$ | 任何成立的群組至少包含路段個數 |
| $c_i$ | 第 $i$ 路段的維修成本 | $L$ | 任何成立的群組的維修成本之下界 |
| $f$ | 固定成本 | $B$ | 總預算的上限值 |
| $\bar{r}$ | 風險程度的門檻值 | | |

表 3-3：集合說明表

| 符號 | 說明 | 符號 | 說明 |
|---|---|---|---|
| $M$ | 全部路段集合 | $N$ | 欲被修復路段群組集合 |

表 3-4：變數說明表

| 符號 | 說明 |
|---|---|
| $x_{ij}$ | 等於 0 表示第 $i$ 路段不屬於第 $j$ 組<br>等於 1 表示第 $i$ 路段屬於第 $j$ 組 |
| $y_j$ | 等於 0 表示第 $j$ 組被不設立<br>等於 1 表示第 $j$ 組被設立 |
| $z_{ij}$ | 第 $j$ 組的路段 $i$ 的維修成本 |
| $Z_j$ | 第 $j$ 群組的最大維修成本 |
| $B_i$ | 第 $i$ 路段的當前群組編號 |
| $C_i$ | 第 $i$ 路段的群組編號 |
| $A_i$ | 第 $i$ 路段是否被修復 |

### 3.2.2. 模式說明

本目標式會取決於路段 $i$ 是否被分到任何一組而可能產生風險程度，並考量路段風險程度總和之最小化。

$$\text{Min} \ \sum_{i=1}^{m} r_i \left(1-\sum_{j=1}^{n} x_{ij}\right)$$

限制式(3.1) 表示每一個路段 $i$ 頂多只能被分到某一組 $j$。

$$\sum_{j=1}^{n} x_{ij} \le 1 \qquad ,\forall i \in M \qquad (3.1)$$

限制式(3.2)表示若 $y_j$ 存在，對應該組 $j$ 的 $x_{ij}$ 才可存在。雖然我們可以用 $\sum_{i=1}^{m} x_{ij} \le m y_j$ 代替限制式(3.2)，但是限制式(3.2)的效果會比較好。

$$x_{ij} \le y_j \qquad ,\forall i \in M, j \in N \qquad (3.2)$$

限制式(3.3)表示若第 $j$ 組存在，則第 $j$ 組至少必須包含 $m_l$ 個路段。

$$\sum_{i=1}^{m} x_{ij} \geq m_l y_j \qquad ,\forall j \in N \ \ j \qquad (3.3)$$

限制式(3.4)表示任何一種分組方法都必須遵守其分組路段的連續性之「成串特性」。換句話說，將路段和群組依據索引值而遞增排序，接著進行路段與群組的分配。若路段$i$被分到組$j$，那下一個路段$i+1$只能被分到組$j$或組$j+1$。同樣的，前一個路段$i-1$只能被分到組$j-1$或是組$j$。總之，同一組的路段一定要連續，並且不允許任何一個跳組的路段。

$$\sum_{j' \leq j} x_{ij'} + \sum_{j' > j} x_{i'j'} \leq 1 \qquad ,\forall i, i' \in M, i' < i; j \in N \qquad (3.4)$$

限制式(3.5)表示空的維修群組之編號應被安排在其最後一個含路段的群組編號之後。因此，一旦組$j$為「空」群組的話，其後面只能接續「空」群組。

$$y_j \geq y_{j+1} \qquad ,\forall j = 1,...,n-1 \qquad (3.5)$$

限制式(3.6)透過設定很大的$Q$以及兩個條件來指定各路段的維修成本。第一條件：$x_{ij}$等於0，則$z_{ij}$等於0。第二條件：$x_{ij}$等於1，則$z_{ij} = Z_j \geq c_i$，而$Z_j$是第$j$組中某具有最大維修成本路段之維修成本。

$$\left.\begin{aligned} &-x_{ij}Q \leq z_{ij} \leq x_{ij}Q && ,\forall i \in M, j \in N \\ &Z_j - (1 - x_{ij})Q \leq z_{ij} \leq Z_j + (1 - x_{ij})Q && ,\forall i \in M, j \in N \\ &Z_j - c_i \geq (x_{ij} - 1)Q && ,\forall i \in M, j \in N \end{aligned}\right\} \qquad (3.6)$$

限制式(3.7)表示維修成本必須足夠大（至少花費$L$）才能成一組。

$$\sum_{i=1}^{m} c_i x_{ij} \geq L y_j \qquad ,\forall j \in N \qquad (3.7)$$

限制式(3.8)表示若路段$i$的風險程度$r_i > \bar{r}$，則該路段將被強迫修復$\sum_{j=1}^{n} x_{ij} = 1$。

$$r_i - \bar{r} \leq Q\left(\sum_{j=1}^{n} x_{ij}\right) \qquad ,\forall i \in M \qquad (3.8)$$

限制式(3.9)表示整體的成本不能超過給定的預算$B$。

$$\sum_{i=1}^{m} \sum_{j=1}^{n} z_{ij} + \sum_{j=1}^{n} f y_j \leq B \qquad (3.9)$$

為了避免養護施工機具對不需要維修的路段造成額外的損壞(如：大型工程車或機具經過時壓壞路面)，故任何要修路段位於不修路段的前後位置，不能共用一組養護施工機具，因此其前後路段必須屬於不同的群組，我們用限制式(3.10)來滿足此假設，其意義簡述如下: 若第$i$路段將不修(即$A_i = 0$)，則$B_i = B_{i-1} + A_{i-1}$；反之，若第$i$路

段將被修(即 $A_i=1$)，則 $B_i=C_i$。

我們先假設 $A_0=B_0=B_1=0$，利用 $A_i=\sum_{j=1}^{n}x_{ij}=1$(或 0)來代表路段 $i$ 被決定要修(或不修)。為了保證分組之編號在跨過不被維修的路段之後必須累進 1，我們從左至右依序檢查每一路段，一旦從某將被維修的路段 $i-1$ (即 $A_{i-1}=1$)遇到不被維修的路段 $i$ (即 $A_i=0$)時，先用 $B_i=B_{i-1}+1$ 將路段 $i$ 的暫時分組編號先累進 1。如果下一路段 $i+1$ 將被維修的話(即 $A_{i+1}=1$)，那 $B_{i+1}=B_i$，而藉由 $B_{i+1}=C_{i+1}=\sum_{j=1}^{n}jx_{i+1,j}$，我們可以強迫路段 $i+1$ 被分配到分組 $C_{i+1}=C_{i-1}+1$(即 $x_{i+1,C_{i-1}+1}=1$)，如此一來即可成功地達成跨過不被維修的路段之後其分組編號必須累進 1 的目的。然而，如果下一路段 $i+1$ 也將不被維修的話(即 $A_{i+1}=0$)，那 $B_{i+1}=B_i$(此時 $C_{i+1}=0$)。假使路段 $i-1$ 與 $i$ 皆將被維修的話，它們可能同組也可能不同組，此時就看 $\sum_{j=1}^{n}jx_{i-1,j}$ 與 $\sum_{j=1}^{n}jx_{ij}$ 的求解結果而定。

我們將以上數種情況整理如下:

1. If $[A_{i-1},A_i]=[1,0]$, then $B_i=B_{i-1}+1$, $C_i=0$
2. If $[A_{i-1},A_i]=[0,1]$, then $B_i=B_{i-1}$, $C_i=B_i$
3. If $[A_{i-1},A_i]=[0,0]$, then $B_i=B_{i-1}$, $C_i=0$
4. If $[A_{i-1},A_i]=[1,1]$, then $B_i=C_i=\sum_{j=1}^{n}jx_{ij}$

此四種情況又可依 $A_i$ 的數值簡化為以下兩種情況:

1. 若第 $i$ 路段將不修(即 $A_i=0$)，則 $B_i=B_{i-1}+A_{i-1}$；
2. 反之，若第 $i$ 路段將被修(即 $A_i=1$)，則 $B_i=C_i$。

此外，為了保證分組編號為遞增，我們加上 $B_i\geq B_{i-1}$ 的限制式。

$$\left.\begin{array}{ll} -QA_i+B_{i-1}+A_{i-1}\leq B_i\leq B_{i-1}+A_{i-1}+QA_i & ,\forall i\in M \\ -Q(1-A_i)+C_i\leq B_i\leq C_i+Q(1-A_i) & ,\forall i\in M \\ B_i\geq B_{i-1} & ,\forall i\in M \end{array}\right\} \qquad (3.10)$$

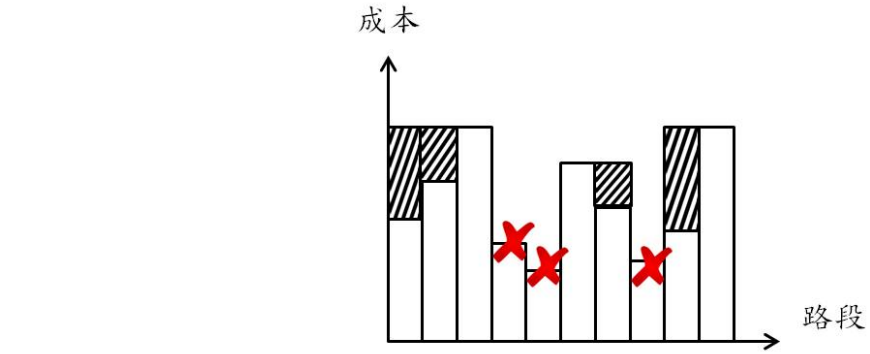
其中

$$A_i=\sum_{j=1}^{n}x_{ij} \qquad ,\forall i\in M$$

$$C_i=\sum_{j=1}^{n}jx_{ij} \qquad ,\forall i\in M$$

以圖 3-2 說明限制式(3.10)，圖 3-2 顯示最佳修復路段的方案為在 10 個路段中，

第 4，5 和 8 路段被決定不修復，而剩下的路段都要修復。當第 4 路段不修，$B_4$ 會因為 $B_3$ 和 $A_3$ 的結果而增加 $B_4=B_3+A_3=1+1=2$ ，而第 6 路段因為與不修的第 5 路段相鄰，故當前的群組編號 $B_6=C_6=2$ 。




| $i$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 第$i$路段當前組別編號 $B_i$ | 0 | 1 | 1 | 2 | 2 | 2 | 2 | 3 | 3 | 3 |
| 第$i$路段的組別編號 $C_i$ | 1 | 1 | 1 | 0 | 0 | 2 | 2 | 0 | 3 | 3 |
| 第$i$路段是否被維修 $A_i$ | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 |

圖 3-2：取得當前群組編號的例子

限制式(3.11)表示 $x_{ij}$ 和 $y_j$ 是二元變數，而 $z_{ij}$ 、$Z_j$ 、$B_i$ 是非負整數。

$$\begin{aligned} &A_0=B_0=B_1=0 \\ &x_{ij},y_j\in\{0,1\};\ z_{ij}\geq 0;\ Z_j\geq 0;\ B_i\geq 0; \qquad ,\forall i\in M,\ j\in N \end{aligned} \tag{3.11}$$

## 3.3 網路模式

### 3.3.1. 建立網路圖

假設各路段之間的分隔點為網路的節點，而一個路段由前後的兩個節點所表示，若該公路有 $m$ 個路段須要修復，那麼該網路總共有 $m+1$ 個節點。而最後的節點稱為虛擬節點(dummy sink node)，虛擬節點代表網路的終點。

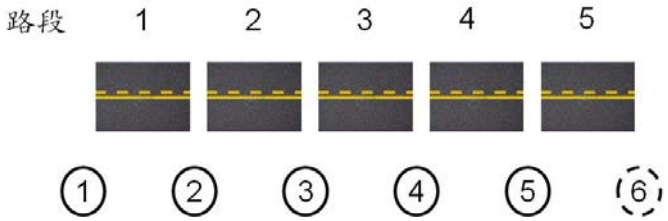



圖 3-3：節點建立圖以 $m=5$ 為例

由於鋪面養護規劃問題僅限制相鄰的路段才能屬於同一組，並不允許任何跨組的路段，如：第 1 組所負責修復的路段為第 1、2 路段，這樣的組合符合以上的限制；反之，當第 1 組所負責的路段組合為第 1、2、4，此組即不符合所列出的限制。因此屬於同一組的路段可以用一條具方向性的節線來表示，而節線是由組內的開始路段的前節點以及最後路段的後節點來界定的。例如：節線(1,3)代表一個群組包含第 1、2 路段，節線(1,2)代表一個群組只包含第 1 路段。節線的長度亦代表該組的總成本，此時可形成一個網路。

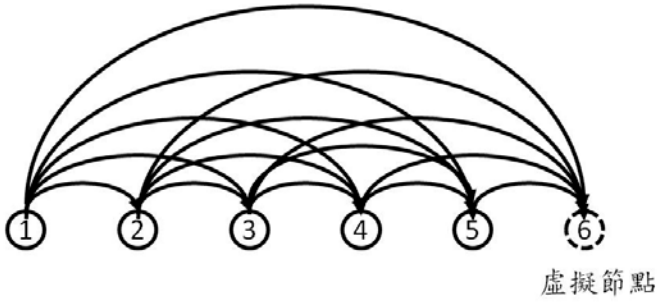



圖 3-4：分組節線建立圖以 $m=5$ 及 $m_l=1$ 為例

「分組節線」是指被修復的路段組合，節線的方向從較小編號節點到較大編號的節點，並以實線來呈現。此步驟將藉由程式碼來產生所有可能的「分組節線」，等同於原來問題的所有可能的養護群組。每一個「分組節線」皆包括兩種屬性，分別為維修成本以及風險程度。當「分組節線」被選定時，這些路段會被修好，所以這類節線的風險程度會在修復後被重設為 0。

假設 $s$ 和 $t$ 依序代表起始路段和最終路段，其中 $1 \le s \le t \le m$。因此任何路段介於 $s$ 和 $t$ 間的路段 $i$，都屬於群組 $[s,t]$，等同於網路中的節線 $(s,t+1)$。因此節線 $(s,t+1)$ 的長度，也就是群組 $(s,t)$ 的維修成本 $\hat{c}_{st}$ 的計算方式如下：

$$\hat{c}_{st} = (t-s+1)*\hat{Z}(s,t)+f \qquad (3.12)$$

公式(3.12)將從群組 $[s,t]$ 中取得該分組內維修成本最大之某路段的維修成本值 $\hat{Z}(s,t)$，並將同組內其它路段的維修成本一律設定為 $\hat{Z}(s,t)$。最後累加該分組內所有路段的維修成本，並加上固定成本。即可得到群組 $[s,t]$ 的維修成本 $\hat{c}_{st}$。

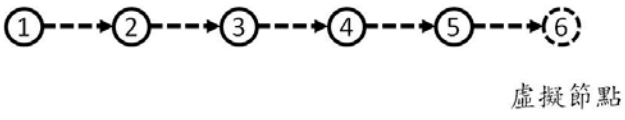



圖 3-5：原狀節線建立圖以 $m=5$ 為例

另外，「原狀節線」或「不修節線」是指因為受到侷限的預算所影響而導致可能存在不被修復的路段，該節線的方向從較小編號節點到較大編號的節點，並以虛線來呈現。由於每一個風險程度小於門檻值 $\bar{r}$ 的路段都有可能不被修復，因此在建立網路時，這類節線總個數不會大於路段的總個數。每一個「原狀節線」也包括兩種屬性，分別為維修成本 $\hat{c}_{st}$ 以及風險程度 $r$。當「原狀節線」被選取時，代表該路段將不被修復而產生 0 維修成本以及不變的風險程度值 $r_i$。由於風險程度大於門檻值 $\bar{r}$ 的任一路段必定會被修復，因此該路段將不會存在「原狀節線」。例如：若第 2 路段一定要修，那麼原狀節線(2,3)將不會存在。

對 $m_l$ 這個參數，在建立「分組節線」與「不修節線」過程中，我們會刪除經過路段個數小於 $m_l$ 的所有節線，只保留合格（亦即其路段個數至少為 $m_l$ ）的節線。如圖 3-5 所示，給定參數 $m_l$ 時，該網路圖不會包含分組節線(1,2)、(2,3)、(3,4)、(4,5)、(5,6)。同樣原理，對於 $L$ 此參數，任何花費 $\hat{c}_{st}$ 小於 $L$ 的分組節線，將被認定為不合格的分組節線，因此一開始即不存在於網路圖。此步驟可減少可行節線個數，因此助於之後求解節省不少時間。

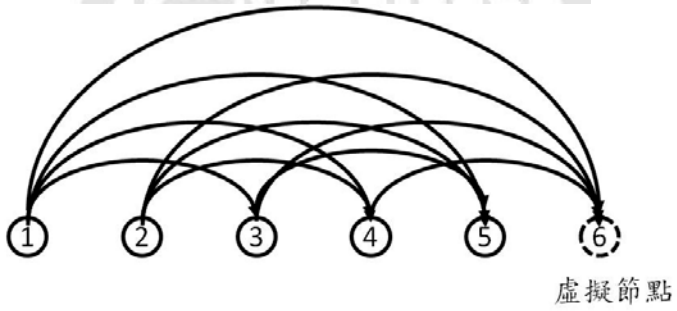



圖 3-6： $m_l$ 對網路圖的影響以 $m=5$ 及 $m_l=2$ 為例

假設從節點 1 到虛擬節點(dummy sink node)的任一個路徑代表某種分組方法。那些路徑會依據預算多寡而決定有或沒有包含「原狀節線」。路徑的成本和風險程度是該分組方法所產生的成本和風險程度。於是，我們可以利用最短路徑方法來尋找使得風險值最小的最佳路徑，不過還必須檢驗該路徑是否滿足「路徑成本的上限值」的額外限制式，此即為含額外限制式的最短路徑問題(constrained shortest path；CSP)。

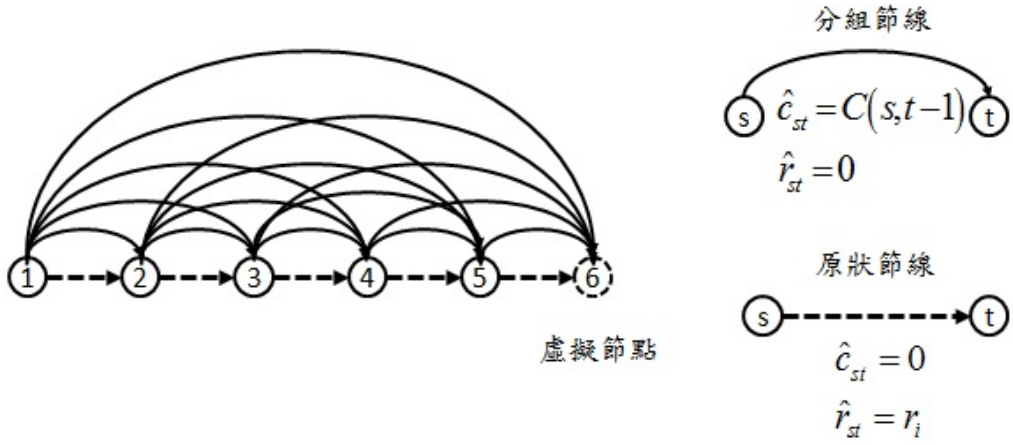



圖 3-7： $m=5$ 及 $m_l=1$ 的完整網路圖

### 3.3.2. 含額外限制式的最短路徑(CSP)模型

假設一網路 $G=(N,A)$，其中 $N$ 為所有路段分隔點之節點(node)集合，$A$ 為所有節線(arc)集合，而 $A$ 包含 $A_C$ 和 $A_B$，依序為「分組節線」之節線集合和「原狀節線」之節線集合，$B$ 為總預算的上限值，$r_{ij}$ 為節線 $(i,j)$ 的風險程度，依據公式(3.13)計算方式估算出節線 $(i,j)$ 的維修成本 $c_{ij}$，接著用二元變數 $x_{ij}$ 表示是否選取節線 $(i,j)$ 為養護規劃路徑之部分節線($x_{ij}=1$ 代表選取，$x_{ij}=0$ 代表不選取)。

藉由網路建立之後，我們可對新的模式寫出其數學規劃模型如下：

$$Z=\min\sum_{(i,j)\in A_B} r_{ij}x_{ij} \tag{3.13}$$

$$\sum_{(i,j)\in A} x_{ij}-\sum_{(j,i)\in A} x_{ji}=\begin{cases}1, & i=s\\ 0, & i\in N-\{s,t\}\\ -1, & i=t\end{cases} \tag{3.14}$$

$$\sum_{(i,j)\in A_C} c_{ij}x_{ij}\le B \tag{3.15}$$

$$x_{ij}\in\{0,1\}\ ,\ \forall i,j\in N \tag{3.16}$$

目標式(3.13)考量路徑所經過的原狀節線，並求使總風險程度最小的最佳路徑。

限制式(3.14)表示當 $i$ 等於 $s$ 時，從 $s$ 流出的節線個數總和要等於 1，意味著在 $s$ 這個開始節點只允許選一條節線做為流出節線。反之，當 $i$ 等於 $t$ 時，流進 $t$ 的節線個數

總和要等於 1，意味著在 $t$ 這個末端節點只允許一條節線做為流進節線。而當 $i$ 不等於 $s$ 或 $t$ 時，從 $i$ 所流進的節線個數要等於從 $i$ 所流出的節線個數。

限制式(3.15)表示最佳路徑所經過「分組節線」的維修成本總合不能超過預算上限。

限制式(3.16)說明 $x_{ij}$ 是二元變數，且 $i$ 、 $j$ 屬於自然數。

根據新模式得知，式(3.13)(3.14)(3.16)為最小化風險程度問題之數學規劃模型式子，而式(3.15)為額外的限制式，為「路徑成本的上限值」，使得原來問題變成「具有額外限制式之最小化風險程度問題。針對此問題，本節將說明利用 LR 配合次梯度法來求解之詳細過程。

## 3.4 以拉氏鬆弛法求解 CSP 問題

### 3.4.1 模式轉換

LR 將導入拉氏係數 $\mu$ (Lagrangian Multiplier)來鬆弛額外限制式。此步驟將複雜的 CSP 轉換成簡單最短路徑問題，亦即將額外限制式(3.15)乘上拉氏係數 $\mu$ 加回原來的目標式，則原 CSP 數學規劃模型將被轉換如下：

$$L(\mu) = \min \sum_{(i,j)\in A_B} r_{ij}x_{ij} + \mu\left(\sum_{(i,j)\in A_C} c_{ij}x_{ij} - B\right) \tag{3.17}$$

$$\sum_{(i,j)\in A} x_{ij} - \sum_{(j,i)\in A} x_{ji} = \begin{cases} 1, & i = s \\ 0, & i \in N - \{s,t\} \\ -1, & i = t \end{cases} \tag{3.18}$$

$$x_{ij} \in \{0,1\} \ , \ \forall i, j \in N \tag{3.19}$$

假設 $G(x) = \sum_{(i,j)\in A_C} c_{ij}x_{ij} - B$，在 $\mu$ 已知且是非負值的情況下，由於拉氏函數式(3.17) $L(\mu)$ 為式(3.13) $Z$ 加上 $\mu G(x)$ ，且 $\mu G(x) \le 0$ ，因此 $L(\mu) \le Z$ ，即 $L(\mu)$ 是原來問題的目標函數之下界。而上界可用在 k 個迭代運算中所計算出最好的 $Z$ 值來代表，使得 $Z^* \le$ Z ，其中 $Z^*$ 為問題的最佳解。

根據弱對偶性質， $L(\mu) \le L^*$ ，其中 $L^*$ 為拉氏函數的最大值，如以下所示；

$$L^* = \max(L(\mu) : \mu \ge 0)$$

而 $L(\mu)$ 可藉由調整 $\mu$ 來求得對偶函數的最大值。

因 $L(\mu) \le Z$ 、 $Z^* \le$ Z 和 $L(\mu) \le L^*$ ，故 $L(\mu) \le L^* \le Z^* \le Z$ 。於是，LR 方法藉由不斷地提升下界以及修正上界的方式，將原問題的現行解逼近最佳解範圍。此步驟會反覆執行直到求得最佳解為止。

在 LR 方法中，所運用的最短路徑演算法將以 $L(\mu)$ 設為路徑的長度。在 $\mu$ 已知的情況下，我們展開 $L(\mu)$ ，並去除不會影響到最短路徑結果的常數 $\mu B$ ，最後得到計算節線長度的公式如以下表格：

表 3-5：LR 所運用最短路徑演算法之節線長度計算公式

| 節線類別 | 計算公式 | 範例 |
|---|---|---|
| 分組節線 | $\hat{c}_{ij} = r_{ij} + \mu c_{ij}$ | $\hat{c}_{13} = r_{13} + \mu c_{13}$ |
| 原狀節線 | $\hat{c}_{ij} = r_{ij}$ | $\hat{c}_{12} = r_{12}$ |

### 3.4.2 次梯度法之介紹

LR 鬆弛額外限制式之後，將藉由迭代方式不斷地重覆計算 $\mu$ 值，直到本問題獲得最佳的 $\mu$ 使得拉氏函數 $L(\mu)$ 接近最高的一個點，此點即為拉氏函數的最大值。此方法稱為「次梯度法」 (subgradient method )。

「次梯度法」首先將給定拉氏係數的初始值，此值通常為 0。由於 $\mu$ 所對應的是不等式的限制式使得本問題的拉氏係數具有方向的限制，因此，下一個 $\mu$ 值為上一個 $\mu$ 值加上方向移動的幅度，如下列的式子所示：

$$\mu^{k+1} = \left[ \mu^k + \theta_k \left( cx - B \right) \right]^+ \qquad k = 1, 2, ....$$

其中，$\mu^k$ 為第 k 次疊代所計算的拉氏係數，$\theta_k$ 為第 k 次疊代所求出的梯度，$cx$ 為第 k 次疊代求出的最短路徑的維修成本，$(cx - B)$ 為控制下一個 $\mu$ 值移動的方向。

在 $\mu^k$ 已知的情況下，將其值代入節線長度計算公式並求出一個最短路徑使得 $L(\mu)$ 達到最小值，接著進一步計算該路徑的總成本 $cx$，由 $(cx - B)$ 的正負號來決定 $\mu$ 的調整方式如下: 若 $(cx - B) > 0$ 時，表示違反限制式，則下一個 $\mu$ 因為加了一個非負值而增加。若 $(cx - B) < 0$ 時，表示符合限制式，則下一個 $\mu$ 因為加了一個負值而減少。

另外，$\theta$ 大小亦會影響求解 $\mu$ 值的效率，若其值太大會使得現行解超過最佳解，

反之太小也會使得現行解沒辦法移動至更好的點。為了保證「次梯度法」會收斂且得到適當的$\mu$值，$\theta$必須滿足下列的條件：

$$\theta_k \to 0 \quad \sum_{j=1}^{k} \theta_j \to \infty$$

$$\theta_k = \frac{\lambda_k [UB - L(\mu^k)]}{\left\| cx - B \right\|^2}$$

以上兩個式子為$\theta_k$的條件以及梯度值$\theta_k$的計算式。其中，$\lambda_k$為給定的幅度參數(step length)，通常在 0 和 2 之間。若在特定回合下$L(\mu)$沒有任何改善，則下一回所使用的$\lambda$值必須減半。而$UB - L(\mu^k)$為第 k 疊代的上下界所夾擠之區間，也就是最佳解的區間範圍。當上下界值很接近時，表示所夾擠之區間較小，縮減幅度可較快獲得收斂的解。反之，當上下界的距離越遠，表示夾擠之區間越大，進而造成移動的幅度越大。$\left\| cx - B \right\|^2$表示$(cx - B)$的長度平方，由於該向量長度平方值越大時，表示欲前進的方向誤差越大，因此可控制調整的幅度不要過大。

接著，我們將次梯度法的求解流程整理成以下三個步驟：

表 3-6：拉氏鬆弛法求解 CSP 的執行步驟

| 步驟 | 執行項目 |
|---|---|
| **1** | 設定$\mu$的初始值<br>設定$\lambda$的初始值<br>設定最大迭代次數$k_{\max}$<br>搜尋一條路徑使得$Z$最大，且該路徑必須滿足$Z$的所有限制式，並將 UB 初始值設為$Z$。 |
| **2** | 將現行迭代$\mu^k$代入進節線長度計算公式<br>尋找最短路徑使得$L(\mu)$最小<br>計算$\theta$<br>將以上的最短路徑的解來計算$Z$<br>● 若$Z < UB$，則更新$UB = Z$。<br>● 若$Z \geq UB$，則 UB 不變 |
| **3** | 在執行每一次迭代的過程中，將紀錄$L(\mu)$的改善幅度，若發現$L(\mu)$沒 |

有任何改善，則 $\lambda = \lambda / 2$。

若 $k < k_{\max}$，則 $k = k+1$ 且計算 $\mu^{k+1} = \left[\mu^k + \theta^k (cx - B)\right]^+$，並重複步驟 2

若 $k = k_{\max}$，則停止運算，並獲得最佳解

---

## 3.5 以第 K 短路徑求解 CSP 問題

本研究主要藉由 Lawler (1976)的第 K 短路徑演算法來求含額外限制式的最短路徑問題。假設 $D^k_{\text{In},\text{Ex}}$ 由第 K 短路徑所分割且滿足 In 和 Ex 的子集合，$E^k$ 為第 K 短路徑所經過的節線集合，其中 In 表示一定經過的節線集合，Ex 表示不能經過的節線集合。首先， Lawler (1976)的方法在第一回合將從所有路徑 $D_{\varnothing,\varnothing}$ 找出第 1 短路徑，其 $E^1 = \{e_1, e_2, ..., e_q\}$ 為第 1 短路徑所經過的節線集合。接著，依據第 1 短路徑所經過的節線將 $D_{\varnothing,\varnothing}$ 分割成 $q$ 個子集合，如以下所示：

表 3-7：第 1 短路徑所分割的子集合

| $q$ 個子集合 | 說明 |
|---|---|
| $D^1_{\varnothing,\{e_1\}}$ | 從 $s$ 到 $t$ 不經過 $\{e_1\}$ 的路徑集合 |
| $D^1_{\{e_1\},\{e_2\}}$ | 從 $s$ 到 $t$ 必須經過 $\{e_1\}$ 但不經過 $\{e_2\}$ 的路徑集合 |
| $D^1_{\{e_1,e_2\},\{e_3\}}$ | 從 $s$ 到 $t$ 必須經過 $\{e_1,e_2\}$ 但不經過 $\{e_3\}$ 的路徑集合 |
| ⋮ | ⋮ |
| $D^1_{\{e_1,e_2,...,e_{q-1}\},\{e_q\}}$ | 從 $s$ 到 $t$ 必須經過 $\{e_1,e_2,...,e_{q-1}\}$ 但不經過 $\{e_q\}$ 的路徑集合 |

這樣的分割方式使得子集合之間沒有交集，且一定不包含第 1 短路徑。集合分割後，將進行計算並比較各子集合的最短路徑，其中最短的便是第 2 短路徑。假設包含第 2 短路徑的路徑集合為 $D^1_{\{e_1\},\{e_2\}}$，其 $E^2 = \{f_1, f_2, ..., f_p\}$ 為第 2 短路徑所經過的節線集合，則該路徑集合將被分割成 $p$ 個子集合，如以下所示：

$$D^2_{\{e_1\},\{e_2,f_1\}}, D^2_{\{e_1,f_1\},\{e_2,f_2\}}, D^2_{\{e_1,f_1,f_2\},\{e_2,f_3\}}, \cdots, D^2_{\{e_1,f_1,f_2,...,f_{p-1}\},\{e_2,f_p\}}$$

接著，此演算法將針對新分割的子集合以及第 1 短路徑所分割但未確認會包含第幾短路徑的路徑集合($D^1_{\{e_1\},\{e_2\}}$ 除外)進行計算各子集合的最短路徑並比較，其中最短的

便是第 3 短路徑。以此類推，Lawler (1976)的演算法將透過前 K-1 短的搜尋結果以及前 K-1 短的分割結果來獲得第 K 短路徑。

每找出第 K 短路徑時，我們將檢查該路徑的總維修成本是否違反預算上限。若違反，此演算法會繼續搜尋第 K+1 短路徑。若該路徑滿足額外限制式，那麼該路徑即為本研究的最佳解。

為了快速獲得最佳路徑，本研究將運用 Van der Zijpp and Fiorenzo Catalano (2005)的概念在子集合中增加條件來排除非可行的子集合，此步驟可減少每回合所須檢查的路徑集合個數，進一步改善 Lawler 演算法的求解時間。

本研究的 KSP 所運用的最短路徑演算法將以風險程度設為所需計算的長度。這樣的設計使得 KSP 將從風險程度最小的路徑進行搜尋，直到找出適合路徑為止。由於前提假設若某路段被選擇修復時，一定會被修到好，因此在KSP所使用的網路中，分組節線的長度為 0，而原狀節線的長度則為 $r_{ij}$ ，如以下表格所示：

表 3-8：KSP 所使用網路的節線長度之計算公式

| 節線類別 | 計算公式 | 範例 |
|---|---|---|
| **分組節線** | $\hat{c}_{ij}=0$ | $\hat{c}_{13}=0$ |
| **原狀節線** | $\hat{c}_{ij}=r_{ij}$ | $\hat{c}_{12}=r_{12}$ |

以下是 KSP 求解的五個步驟：

表 3-9：第 K 短路徑演算法求解 CSP 的執行步驟

| 步驟 | 執行項目 |
|---|---|
| **1** | 初始化最大迭代次數 $K_{max}$ |
| **2** | 計算並比較各子集合的最短路徑，並獲得第 K 短路徑。<br>若任何最短路徑的風險程度大於 UB，刪除該集合。 |
| **3** | 檢查第 K 短路徑的維修成本是否違反預算上限<br>◆ 若維修成本 > $B$ ，進行下一個步驟。<br>◆ 若維修成本 ≤ $B$ ，第 K 短路徑則是最佳解，並停止運算。 |
| **4** | 進行分割所包含第 K 短路徑的路徑集合<br>◆ 若子集合必經過節線集合的維修成本 > $B$ ，刪除該集合。 |

- 若子集合必經過節線集合的維修成本≤$B$，保留該子集合

**5** 檢查 K 是否到達 $K_{max}$

- 若 $K< K_{max}$ ，但未找到最佳解，則重複以上步驟 2 至 4
- 若 $K= K_{max}$ ，但未找到最佳解，則 $K_{max}=K+1000$，並重複以上步驟 2 至 4

---

## 3.6 以模擬退火法求解修復路段最佳組合

由於 LR 所求的結果有時會存在對偶間隙，此間隙將會影響解的品質，為了提升 LR 的求解效能，本研究將透過模擬退火法(Simulated Annealing; SA)搜尋修復路段最佳組合的方式來改善解的品質。

在原模式中，利用路段修復之選擇會影響總風險的特性，我們根據路段的維修情形而定義了 0-1 組合，其 0 表示該路段不被修，而 1 表示該路段被選擇修復。而與總風險程度最有直接影響的是可能不被修的路段，因此鄰近解產生時，我們只針對可能不被修的路段變動 0-1 值，而一定要修復的路段固定為 1。如圖 3-8 所示，假設有 5 個路段須修復，其$i$表示路段的順序位置，$r_i$表示各路段的風險程度，給定風險程度的門檻值$\bar{r}=5$，故可能不被修的路段為路段 2、路段 3、路段 5，而剩下的路段是一定要修的路段。因此，在產生 0-1 組合時，SA 只針對路段 2、路段 3、路段 5 變動 0-1 值。

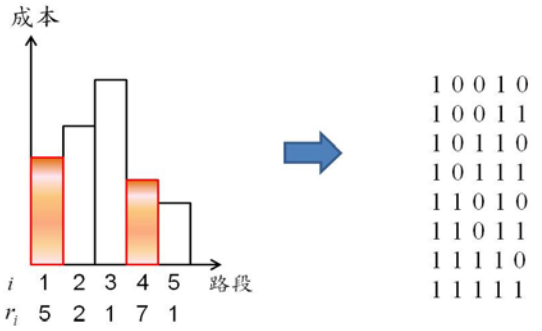



圖 3-8：修復路段 0-1 組合以$m=5$為例

在產生鄰近解的步驟，我們將檢查目前現行解的維修成本是否違反預算上限。若違反，演算法將從現行解的 0-1 組合選出一個 1 並將它轉為 0，此轉換會讓鄰近解的維修成本降低，同時也增加了總風險程度。若不違反，此現行解在原模式中是個可行解，演算法將從現行解的 0-1 組合選出一個 0 並將它轉為 1，此轉換會讓鄰近解的風

險程度減少，同時也提升總維修成本。這樣產生鄰近解的設計是為了逼鄰近解接近可行解範圍，並在可行範圍內期望可以產生出較低風險的修復路段組合。

但是這樣的 0-1 值轉換有時會產生非可行的鄰近解，此時必須重新執行此步驟直到找出可行解為止。這樣的鄰近解產生方式可能造成 SA 的求解時間大增，甚至最後的解無法收斂。因此，為了讓 SA 每一次皆可產生可行的鄰近解，在轉換 0-1 值之前，我們針對以下兩種情形建立可行的 1 值之集合或可行 0 值之集合：

**情形 1：當現行解的維修成本 > $B$ 時，SA 必須選出一個 1 並將它改為 0**

當 SA 將 1 值變為 0 時，可能會導致在兩個不修路段中間要修的路段總數未達到 $m_l$，且因為(3.10)的限制而將成立一個群組，該群組只能包含這些路段，此時該群組已違反限制式(3.3)。為了避免產生出非可行解，必須建立可行的 1 值之集合。首先，給定兩個不修的虛擬路段，分別為第 0 路段及第 $m+1$ 路段，並對於現行解進行搜尋路段修復範圍，此範圍界於兩個不修的路段之間。接著，將檢查每一個範圍所包含要修路段的總數。若路段總數 $\leq m_l$，該範圍沒有可行的 1 值。若 $m_l <$ 路段總數 $\leq 2m_l$，在該範圍中只有在緊鄰不修路段的 1 可被改為 0 值(如圖 3-9(a))。若路段總數 $> 2m_l$，在該範圍中除了緊鄰不修路段的 1 值可被改為 0 外，界於前面不修路段位置 $+m_l$ 與後面不修路段位置 $-m_l$ 之間的 1 值亦皆可被改為 0(如圖 3-9(b))。這些可被改為 0 的 1 值路段若是可不修的路段，則該路段的位置將被記錄於可行的 1 值之集合。


(a) (b)

圖 3-9：可行的 1 值之位置

(a) $m_l <$ 修復路段總數 $\leq 2m_l$ (b)修復路段總數 $> 2m_l$

**情形 2：當現行解的維修成本 $\leq B$ 時，SA 必須選出一個 0 並將它改為 1**

此情形必須建立可行的 0 值之集合。首先，給定兩個不修的虛擬路段，分別為第 0 路段及第 $m+1$ 路段。接著，檢查每一個不修路段 $i$ 的前後路段 $i-1$、$i+1$

之修復情形，其中$0<i<m+1$。若路段$i-1$、$i$、$i+1$的修復情形依序為「要修-不修-不修」(1-0-0)或是「不修-不修-要修」(0-0-1)或是「要修-不修-要修」(1-0-1)，則該不修路段$i$的 0 值皆可被改為 1(即為可行的 0 值)。此外，若路段$i-1$、$i$、$i+1$的修復情形依序為「不修-不修-不修」(0-0-0)時，除非$m_l=1$，不修路段$i$的 0 值才是可行的 0 值，否則此轉換之選擇會因為發生在兩個不修路段中間要修的路段總數未達到$m_l$而違反限制式(3.3)。最後，這些可行的 0 值之路段位置將被記錄於可行的 0 值之集合。

由於本研究的最短路徑演算法執行上很有效率，因此，為了方便求解，我們將原本修復路段 0-1 組合轉換成原狀節線的 0-1 組合，並使用最短路徑演算法求解，其 0 表示路徑會經過該原狀節線，而 1 表示路徑經過分組節線而非原狀節線。由於最短路徑必須經過被選定的原狀節線，因此在運算時此演算法不能選擇更新跨過原狀節線的分組節線，此時才能求出正確解。為了簡化求解最短路徑的流程，我們將該原狀節線定義為網路中的斷點，該斷點將網路分成數個較小的子網路。此時，SA 會針對所有子網路進行求解最短路徑，其中整體最短路徑的維修成本為所有子網路的最短路徑之維修成本總和，整體最短路徑的風險程度為被選定的原狀節線之風險程度總和。如圖 3-10 所示，當原狀節線的 0-1 組合為 11011011 時，有兩個原狀節線③→④和⑥→⑦被選定為一定要經過的原狀節線，故③→④和⑥→⑦為該網路的斷點，並且只需要針對①到③、④到⑥、⑦到⑨三個區域進行搜尋最短路徑。最後將各區域的路徑結果與固定的原狀節線依節點的順序連結起來，即可獲得整體的最短路徑。

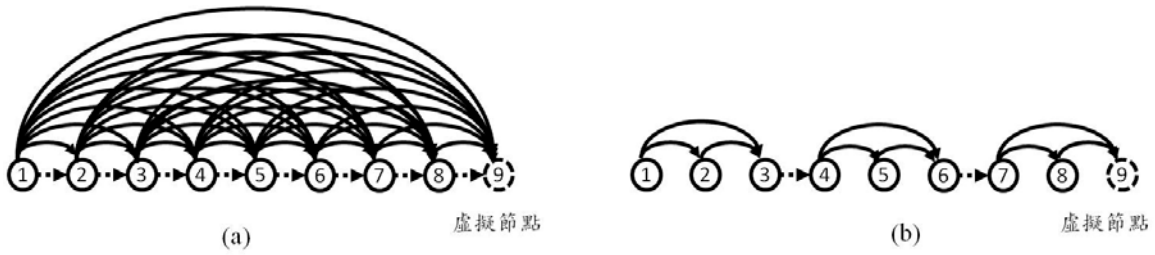



圖 3-10： $m=8$及$m_l=1$的網路圖

(a) $m=8$及$m_l=1$的完整網路圖(b) $m=8$及$m_l=1$的部分網路圖

因為 LR 的求解時間非常快，故以 LR 最後收斂的解設為 SA 的初始解。

此外，若透過任何組合所求的最短路徑其總維修成本超過預算的上限，則目標值

將被給予懲罰成本，反之，目標值只等於總風險程度。

在 SA 演算法中，本研究以波茲曼函數(Boltzmann Function)來計算鄰近解被接受的機率(probability of acceptance)，計算公式如下：

$$\text{PR(A)}=\min\left\{1, e^{-\left(\frac{\Delta f}{T_k}\right)}\right\}$$

$PR(A)$ ：鄰近解被接受的機率

$T_k$ ：在第 k 迭代的溫度

$\Delta f$ ：現行解的目標值與鄰近解的目標值之差距

當機率值 PR(A) 計算出來之後，再利用亂數產生器，產生一個介於[0,1) 之間的亂數 $R$，以判斷是否接受該鄰近解。若 $R > PR(A)$，則該解不被接受。反之若 $R \le PR(A)$，則該解被接受，並取代現行解。因此機率值越高，鄰近解被接受的機率也會越高。

在波茲曼函數中，$T_k$ 用以決定接受機率的高低，此值的設計應會隨著搜尋過程而降低。而 $T_k$ 降溫的函數通常乘於冷卻速率值(cooling rate)，此值介於0與1之間的小數。因此，一個適當的降溫機制可使SA能有效地跳離局部解，又能在有限時間內收斂。

以下是SA求解的六個步驟：

表 3-10：模擬退火法的執行步驟

| 步驟 | 執行項目 |
|---|---|
| **1** | 產生初始解 (LR步驟) |
| **2** | 初始化參數。<br>設定現行解的目標值=最佳解的目標值=初始解的風險程度 |
| **3** | 產生鄰近解 |
| **4** | 檢查鄰近解是否違反預算上限<br>◆ 若維修成本 > $B$，目標值=風險程度+10000*(維修成本 $-B$)<br>◆ 若維修成本 ≥ $B$，目標值=風險程度 |
| **5** | -計算鄰近解的接受機率，並與 $R$ 進行比較。<br>◆ 若 $R > PR(A)$，則該解不被接受<br>◆ 若 $R \le PR(A)$，則該解被接受，並取代現行解。新的現行解 |

| | |
|---|---|
| | 再進一步與最佳解比較，若新現行解的目標值比最佳解低，則取代最佳解。<br>-此步驟執行的總次數到達一定的次數後，溫度$T$會降低。<br>$$T_k = T_{k-1} * cooling \text{ rate}$$<br>-回到步驟3重新產生新的鄰近解。 |
| **6** | 反覆執行步驟四，直到達到最大迭代次數則結束運算，並獲得修復路段最佳組合。 |

## 3.7 結合拓樸排序之最短路徑演算法

由於 LR、KSP 及 SA 的求解過程皆透過最短路徑獲得最佳解或次佳解，故最短路徑演算法的求解效率十分重要。而 Wang, Tsai, & Li (2011)所提出的結合拓樸排序之最短路徑演算法方法在大型的網路規模也可以快速求出最短路徑。因此，本小節將會介紹 Wang, Tsai, & Li (2011)的最短路徑的方法及搜尋步驟。

根據(3.3.1)網路的假設，本研究的網路之所有節線具有從$i$到$j$的特定方向，其$i<j$，則形成一個無有向迴圈的網路。此假設使得沒有任何節線流進節點 1，因為沒有任何節點的編號比 1 還小，故 indegree(1)=0。當在節點 1 完成橫向搜尋並刪除節點 1 時，因為沒有任何節點的編號比 2 還小，節點 2 將成為下一個被搜尋的節點。於是，同樣原理，直到節點$m+1$被檢查完畢。由於拓樸排序適用於具有單一方向無迴圈的網路圖，且由 indegree 為 0 的節點開始進行橫向搜尋，因此依據本模型的網路特性，拓樸排序所求的結果剛好是節點編號由小排到大 1、2、…、$m+1$。

因為節點$m+1$沒有下一個節點，所以我們的演算法將不考慮節點$m+1$，僅進行檢查節點從 1 到$m$。本研究的最短路徑的搜尋流程如下：

表 3-11：最短路徑演算法的執行步驟

| 步驟 | 執行項目 |
|---|---|
| **1** | 初始化從起點至所有節點$i$的長度$D(i)=\infty$<br>初始化所有節點$i$的前置節點$P(i)=NULL$ |
| **2** | 設定節點 1 的長度$D(1)=0$<br>設定節點 1 的前置節點$P(1)=NULL$ |
| **3** | 當$i<m+1$時 |

對每一個從 $i$ 流出的節線 $(i, j)$ 進行檢查

若 $D(j) > D(i) + \hat{c}_{ij}$ ，則 $D(j) = D(i) + \hat{c}_{ij}$ 和 $P(j) = i$

**4** 若 $D(m+1) = \infty$ ，則此問題沒有最佳解。

反之，依據 $P(m+1)$ 所記錄的前置節點，我們將進行追蹤最短路徑所經過的所有節點。

---

## 3.8 小結

本章以 Wang, Tsai, & Li (2011)之模式為基礎，在已知路段維修成本和路段的風險程度的情況下，同時考慮道路的安全性以及維修經費的限制，並假設最佳的修路方案可能存在某些路段不修的情形發生，本研究藉由建構整數規劃模式來規劃適當的選擇性修路方案。但由於整數規劃模式的限制式過多，在實際龐大的資料量，此模式很難在有效的時間求得最佳解。因此，本研究將原模式轉換成含額外限制式的最短路徑模式，並以拉氏鬆弛法以及第 K 短路徑演算法來求解。此外，本研究還設計了適當的模擬退火法來改善拉氏鬆弛法的求解效能。

# 第四章

# 數值分析

## 4.1 參數設定

本研究進行數值測試的相關參數設定如表 4-1 所示，路段的風險程度 $r_i$ 是在[1,10]之間隨機產生的整數，路段的維修成本 $c_i$ 則是在[30,100] (千單位)之間隨機產生的整數，固定成本 $f$ 為 30，風險程度的門檻值 $\bar{r}$ 是[1,10]的中間數值，為了符合公路養護工程的實際規模，群組維修成本的下限值 $L$ 大約 250(千單位) 。

由於各國各地方對於路段的風險程度、路段的維修成本、固定成本、風險程度的門檻值、群組維修成本的下限值都有不同的評估和定義方式，因此以上的參數設定將不考慮數值的實際性、風險程度與維修成本的關聯性，只單純設計一個假想情境來驗證我們的方法。

表 4-1：相關參數之設定

| 參數 | 數值 |
|---|---|
| $r_i$ | 1~10 |
| $c_i$ | 30~100(千) |
| $f$ | 30 |
| $\bar{r}$ | 5 |
| $L$ | 250(千) |

UFLP 整數規劃模式(GRB1)將以 Gurobi 最佳化軟體來求解第三章提及的式子(3.1)-(3.11)，由於此模式的限制式較多，因此預估隨著路段數量變大，而限制式個數亦將快速成長。由於 GRB1 需耗費太多運算時間才能收斂至最佳解，我們最多僅記錄其執行 10000 秒內可得到的最好的解。而 GRB1 的 gap 是該解的目標值與最佳解的目標值之間的 gap，gap 的公式如下：

$$gap = \frac{obj_{GRB1} - obj^*}{obj^*}$$

其 *obj** 為 GRB2 所求出的最佳目標值。

而原始問題轉換成 CSP 模式後將以三種方法來求解，分別為：以 Gurobi 針對 3.3.2 所提的模式(式子(3.13)-(3.16))進行求解(GRB2)、拉氏鬆弛法(LR)、第 K 短路徑演算法(KSP)。此外，本研究還透過模擬退火法(SA)結合拉氏鬆弛法的方式來尋找修復路段的更佳組合，以提升拉氏鬆弛法解的品質。

拉氏鬆弛法(LR)之執行效率良好且可較快獲得最佳解或次佳解，我們定義了 LR 的 gap 來衡量其解的品質，gap 越大代表該解離最佳解越遠，反之，該解越接近最佳解。由於 UB 在迭代的運算過程中將不斷被更新，其更新的原則為若存在任何路徑具有較小的風險程度又滿足所有額外限制式的話，則 UB 值會被更新成該路徑的風險程度。因此，UB 是目前最好可行解的目標值，在無法預估最佳解的情況下，我們將 UB 設為 LR 可得到最佳解的目標值。LR 演算法的 gap 公式如下：

$$LRgap = \frac{UB - obj^*}{obj^*}$$

其 *obj** 為 GRB1 或 GRB2 所求出的最佳目標值。

在 LR 的演算法中，設定一個好 UB 初始值非常重要。好的 UB 可以讓最佳解在較少迭代次數收斂。因此，本研究設計了以下三個步驟：

步驟一：在網路圖中搜尋所有原狀節線，並計算其 $ratio_i = c_i / r_i$

步驟二：搜尋維修成本最小的路徑，並將預算與該路徑的維修成本相減取得剩下的預算。

步驟三：若剩下的預算還能夠再維修一個路段，則移除 $ratio_i$ 最小的原狀節線（亦即強迫修復該路段），並重複步驟二-三。反之，若剩下的預算不能夠再多修任何路段，便記錄該路徑的風險程度，也就是我們需找的 UB。

以上的步驟，一開始為了搜尋的維修成本最小之路徑，將會選擇經過所有原狀節線的路徑，此時該路徑的風險程度是最壞的 UB。而後來經過反覆移除原狀節線並搜尋新路徑，則會得到更好的 UB。

本研究已針對 LR 相關參數的初始值設定試圖調整，並發現 Ahuja et al. (1993)在書中範例所給定的設定使得本研究的 LR 最佳解收斂的程度相當好。因此，LR 相關參數的初始值設定以 Ahuja et al.(1993)為主，如下表所示：

表 4-2：LR 相關參數的初始值設定

| 參數 | 數值 |
|---|---|
| $\mu$ | 0 |
| $\lambda$ | 0.8 |
| $\lambda$ 減半的迭代次數 | 3 |

第 K 短路徑演算法(KSP)可以說是一種暴力法的求解方式。此演算法將透過第 K-1 短路徑所展開的分支來搜尋第 K 短路徑。由於在 $m$ 值越大時，網路的節線數量也成長不少，使得第 K-1 短路徑所展開的分支變非常多。此時，下一個回合所需計算並儲存可行路徑的數量會導致記憶體不足的發生。因此，我們設計了 $m=30$ 來測試 KSP 效率。

在模擬退火法(SA)中，由於鄰近解之選擇將影響最後最佳解收斂的程度，因此，本研究試圖透過以下幾種選擇鄰近解的方式來驗證 SA：

- 當現行解的維修成本不違反預算上限時，SA 必須選出一個「不被修」狀態的路段轉換成「被修」的狀態，此時鄰近解的維修成本將會提升，而風險程度也因此減少。本研究希望每一次都可以選出一個既不會讓鄰近解總成本超過上限又可以讓風險程度減少的路段，因此本研究考慮以維修成本最小為選擇路段的原則。
- 當現行解的維修成本違反預算上限時，SA 必須選出一個「被修」狀態的路段將其轉成「不被修」的狀態，此時鄰近解的維修成本將會降低，而風險程度也因為增加一個不被修的路段而提升。本研究希望每一次都可以選出一個可以讓風險程度緩緩增加的路段，因此本研究考慮以風險程度最小為選擇路段的原則。

但在測試的過程中，我們發現以隨機來選擇「不被修」路段，再結合以風險程度最小來選擇「被修」路段的方式意外能獲得比較好的解。因此，本研究將以收斂最好的方式來設定鄰近解之選擇。

由於SA以LR最後收斂的解設為演算法的初始解，故LR+SA的求解時間將包含LR的求解時間。本研究還試圖針對不同迭代次數來測試SA，並發現在三種不同迭代次數100、1000、10000，所收斂的最佳組合仍是一樣，因此為了省時間，SA的最大迭

代次數設為100。為了讓SA在初期接受機率較高，以便擴大搜尋範圍，增加找到最佳解的機會，我們額外設計了相同溫度的執行次數，也就是說在某溫度須執行到達一定次數後才會降溫。因此，SA的相關參數設定如下表所示：

表 4-3：SA 相關參數設定

| 參數 | 數值 |
|---|---|
| 溫度初始值 | 10000 |
| 冷卻速率 | 0.999 |
| 迭代次數 | 100 |
| 相同溫度的執行次數 | 3 |

SA 演算法的 gap 公式如下：

$$gap = \frac{SA\ best\ obj - obj^*}{obj^*}$$

其 *obj**為 GRB1 或 GRB2 所求出的最佳目標值。

本研究將以路段集合的大小來進行測試資料，在此分別以測試小型資料集、測試中型資料集及測試大型資料集代表之。其中，小型資料集將所需分析的路段集合為 $m=30$，中型資料集將所需分析的路段集合為 $m=100$、$m=200$、$m=300$，大型資料集將所需分析的路段集合為 $m=1000$、$m=2000$、$m=3000$。除了一些相關參數如同表 4-1 所設定，每資料集的預算上限值 $B$、群組的路段個數下限 $m_l$ 為由使用者輸入。對於參數 $n$ 的設定方式，若能找出很接近最佳群組個數 $n^*$ 的設定值，則其值會有效地縮短 GRB1 的求解時間。但 $n^*$ 沒辦法先前估算，因此，我們將群組個數設定為 $n=\lceil m/m_l \rceil$。針對每筆測試資料，本研究將使用具有 corei7 處理器及 8GB RAM 之個人電腦，並在 Window 7 作業系統下進行驗證 LR、KSP、LR+SA 及使用 Gurobi5.6.2 版的最佳化軟體來驗證 GRB1、GRB2。

## 4.2 數值分析與結果

### 4.2.1 小型的資料集

本小節針對 $m=30$ 的資料集產生整數規劃模式的 10 個 case。每一個 case 都包含 30 個路段，而路段的維修成本和風險程度因為在特定的區間(如表 4-1)隨機產生數值

而各 case 有所不同。接著，藉由以上模式的測試 case 結合 $m_l$ 、 $f$ 、 $\bar{r}$ 、 $L$ 等四個參數，來建立網路模式的測試檔。由於建立整數規劃模式及網路模式測試檔的時間非常快，故此時間的大小不考慮在求解時間內，並設為 0。以下表 4-4 是條列各方法求解的相關結果：

表 4-4： $m=30$ 時，各方法的求解結果

| $m=30 \quad m_l=2 \quad n=15 \quad B=2300$ | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Case** | **GRB1** | | **GRB2** | | **KSP** | | **LR** | | | | **LR+SA** | | |
| | **Obj** | **t (s)** | **Obj** | **t (s)** | **Obj** | **t (s)** | $L^*(\mu)$ | **UB** | **Gap (%)** | **t (s)** | **Obj** | **Gap (%)** | **t (s)** |
| **1** | 15 | 439 | 15 | 1 | 15* | 212 | 14 | 16 | 6.67 | 0 | 15* | 0 | 0 |
| **2** | 10 | 530 | 10 | 0 | 10* | 2810 | 9.72 | 10* | 0 | 0 | 10* | 0 | 0 |
| **3** | 2 | 225 | 2 | 0 | 2* | 49 | 1.71 | 2* | 0 | 0 | 2* | 0 | 0 |
| **4** | 10 | 314 | 10 | 0 | 10* | 1400 | 8.62 | 10* | 0 | 0 | 10* | 0 | 0 |
| **5** | 1 | 354 | 1 | 0 | 1* | 263 | 0.43 | 1* | 0 | 0 | 1* | 0 | 0 |
| **6** | 1 | 75 | 1 | 1 | 1* | 46 | 0.81 | 1* | 0 | 0 | 1* | 0 | 0 |
| **7** | 7 | 763 | 7 | 0 | 7* | 1254 | 6.76 | 9 | 28.57 | 0 | 8 | 14.28 | 0 |
| **8** | 3 | 77 | 3 | 0 | 3* | 286 | 2.65 | 3* | 0 | 0 | 3* | 0 | 0 |
| **9** | 2 | 317 | 2 | 1 | 2* | 184 | 1.44 | 2* | 0 | 0 | 2* | 0 | 0 |
| **10** | 6 | 194 | 6 | 0 | 6* | 1014 | 4.86 | 6* | 0 | 0 | 6* | 0 | 0 |
| **平均** | | 328.8 | | 0.3 | | 751.8 | | | | 0 | | | 0 |

根據表 4-4 所示，GRB1、GRB2 及 KSP 在每個 case 皆求得最佳解，而根據 LR gap 欄位，LR 只求得 80%個 case 的最佳解，其中最大的 gap 為 28.57%。KSP 的解雖然與 GRB1、GRB2 相同，但平均運算時間卻是最久 751.8 秒(s)，遠大於 GRB2 及 LR 的求解時間。KSP 演算法的求解時間之所以這麼長，主要原因是其為一種暴力求解方式，在最壞的情況下有可能必須搜尋所有可行的路徑才能找到最佳解。此演算法雖然可求出最佳解，但無法預知最佳解會是第幾短路徑。而 LR+SA 不但求解時間快而且改善效果還不錯，尤其是在 case 1，已將 LR 的次佳解改善到最好。

由於本研究的目標為最小化風險程度，而有可能會發生不同組合，但有相同的風險程度。因此，本研究欲求得的最佳解很可能不唯一。根據附錄表 A-1 和表 B-1，以 case 4 來說明，GRB1 的維修成本和最佳群組個數依序為 2293 和 9，而 GRB2 的結果卻 2247 和 6。雖然兩者是完全不相同的組合，但其解的維修成本都不會超過預算，

故兩解都是最佳解。

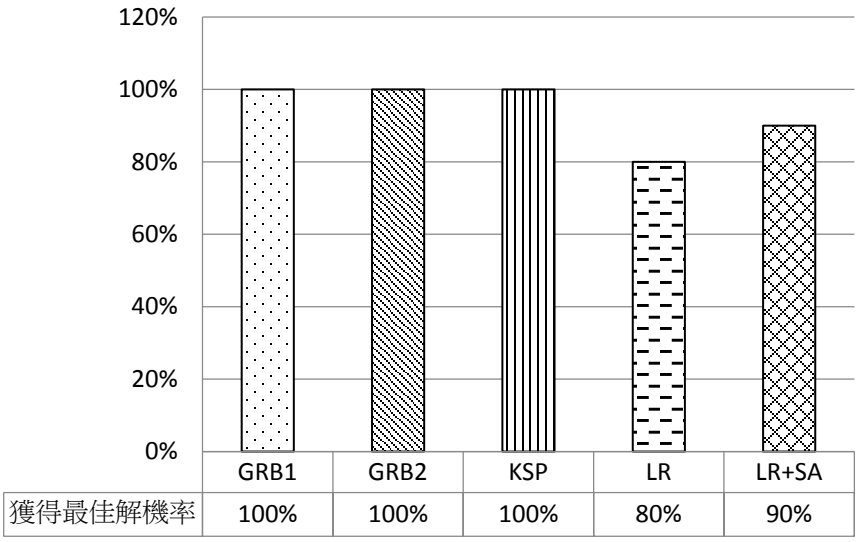



圖 4-1：小型資料集的各方法獲得最佳解情形

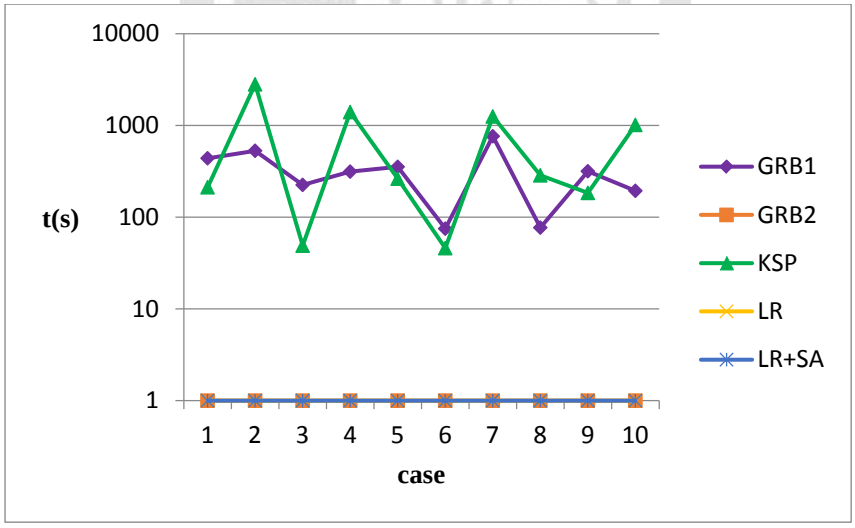



圖 4-2：小型資料集的各方法求解時間

### 4.2.2 中型資料集

本小節針對 $m=100$ 、 $m=200$ 、 $m=300$ 的中型資料集隨機產生整數規劃模式的 10 個 case。接著，根據整數規劃的測試檔，將進行建立網路模式的測試檔。此測試結果也一樣不考慮測試檔的建立時間。以下表 4-5、4-6 及 4-7 是條列各方法求解的相關結果：

表 4-5：$m=100$ 時，各方法的求解結果

| $m=100 \quad m_l=10 \quad n=10 \quad B=8500$ | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Case** | **GRB1** | | **GRB2** | | **LR** | | | | **LR+SA** | | |
| | **Obj** | **t (s)** | **Obj** | **t (s)** | $L^*(\mu)$ | **UB** | **Gap (%)** | **t (s)** | **Obj** | **Gap (%)** | **t (s)** |
| **1** | 13 | 4884 | 13 | 1 | 11.86 | 13* | 0 | 0 | 13* | 0 | 0 |
| **2** | 25 | 3225 | 25 | 1 | 23.46 | 30 | 20 | 0 | 25* | 0 | 0 |
| **3** | 19 | 1855 | 19 | 1 | 18.34 | 19* | 0 | 0 | 19* | 0 | 0 |
| **4** | 20 | 2988 | 20 | 1 | 19.06 | 20* | 0 | 0 | 20* | 0 | 0 |
| **5** | 36 | 1636 | 36 | 1 | 34.82 | 39 | 8.33 | 1 | 39 | 8.33 | 2 |
| **6** | 30 | 1364 | 30 | 1 | 27.6 | 32 | 6.67 | 0 | 30* | 0 | 0 |
| **7** | 13 | 4491 | 13 | 1 | 10.98 | 13* | 0 | 0 | 13* | 0 | 0 |
| **8** | 29 | 5091 | 29 | 2 | 25.86 | 32 | 10.34 | 0 | 32 | 10.34 | 0 |
| **9** | 23 | 6353 | 23 | 1 | 20.91 | 25 | 8.70 | 0 | 24 | 4.34 | 0 |
| **10** | 21 | 4128 | 21 | 1 | 19.35 | 27 | 28.57 | 0 | 23 | 9.52 | 0 |
| **平均** | | 3601.5 | | 1.1 | | | | 0.1 | | | 0.2 |

表 4-6：$m=200$ 時，各方法的求解結果

| $m=200 \quad m_l=10 \quad n=20 \quad B=18000$ | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Case** | **GRB2** | | **LR** | | | | **LR+SA** | | |
| | **Obj** | **t (s)** | $L^*(\mu)$ | **UB** | **Gap (%)** | **t (s)** | **Obj** | **Gap (%)** | **t (s)** |
| **1** | 17 | 2 | 16.14 | 18 | 5.88 | 0 | 17* | 0 | 0 |
| **2** | 13 | 3 | 12.2 | 13* | 0 | 0 | 13* | 0 | 0 |
| **3** | 7 | 3 | 6.56 | 7* | 0 | 1 | 7* | 0 | 1 |
| **4** | 23 | 2 | 22.28 | 27 | 17.39 | 0 | 26 | 13.04 | 0 |
| **5** | 19 | 2 | 18.78 | 19* | 0 | 0 | 19* | 0 | 0 |
| **6** | 23 | 2 | 22.01 | 23* | 0 | 0 | 23* | 0 | 0 |
| **7** | 15 | 4 | 14.99 | 15* | 0 | 1 | 15* | 0 | 1 |
| **8** | 7 | 2 | 6.52 | 7* | 0 | 1 | 7* | 0 | 1 |
| **9** | 11 | 3 | 9.98 | 11* | 0 | 0 | 11* | 0 | 0 |
| **10** | 18 | 5 | 17.37 | 18* | 0 | 0 | 18* | 0 | 0 |
| **平均** | | 2.8 | | | | 0.3 | | | 0.3 |

表 4-7： $m=300$ 時，各方法的求解結果

| $m=300$ $m_l=10$ $n=30$ $B=27000$ | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Case** | **GRB2** | | **LR** | | | | **LR+SA** | | |
| | **Obj** | **t (s)** | $L^*(\mu)$ | **UB** | **Gap (%)** | **t (s)** | **Obj** | **Gap (%)** | **t (s)** |
| **1** | 13 | 15 | 12.63 | 13* | 0 | 1 | 13* | 0 | 1 |
| **2** | 15 | 13 | 14.48 | 15* | 0 | 0 | 15* | 0 | 0 |
| **3** | 20 | 8 | 19.1 | 22 | 10 | 1 | 22 | 10 | 1 |
| **4** | 31 | 26 | 30.06 | 34 | 9.68 | 0 | 34 | 9.68 | 0 |
| **5** | 33 | 19 | 32.87 | 33* | 0 | 1 | 33* | 0 | 1 |
| **6** | 13 | 20 | 12.68 | 15 | 15.38 | 0 | 15 | 15.38 | 0 |
| **7** | 26 | 22 | 24.97 | 26* | 0 | 0 | 26* | 0 | 0 |
| **8** | 27 | 19 | 26.07 | 27* | 0 | 0 | 27* | 0 | 0 |
| **9** | 31 | 26 | 30.61 | 32 | 3.23 | 1 | 31* | 0 | 1 |
| **10** | 31 | 9 | 29.64 | 31* | 0 | 0 | 31* | 0 | 0 |
| **平均** | | 17.7 | | | | 0.4 | | | 0.4 |

如 4.2.1 所敘述，KSP 雖然求解時間較長，但對於規模不大的 CSP 問題還是一個可行的方法。但隨著資料集增大，網路的節線也會增加，使得可行路徑也因此成長了不少。在最壞的情況下，$m_l=1$時，網路將會產生 n!的分組節線，而從 n!的分組節線以及原狀節線中可以搭配出非常多種組合。因此，在$m=100$時，KSP 的分割原理使得此方法在計算過程中面臨記憶體不足的問題，並無法求得最佳解。因此，從$m=100$以上，本研究不進行探討 KSP 的求解結果。

在$m$和$n$已知的情況下，整數規劃模式包含 O($n^3m^2$)條限制式，而隨著$m$和$n$擴增，此模式的限制式總量會變得非常多，可知原問題是個複雜問題。因此，根據平均求解時間的表 4-8 所示，GRB1 計算較大的資料集時，效率顯得不好。在$m=100$時，平均求解時間已高達 3601.5 秒(s)。在求解時間上，與 GRB2 及 LR 的相較下， GRB1 明顯較落後。而在$m=200$、$m=300$時，GRB1 計算時間已長達 10000 秒但未算出任何可行解，可知此方法不適用於資料量比較大的情況。

根據測試結果可見，LR 所求出解的品質並不是很穩定，在 30 cases 中只有 18 cases 的解是最佳解，占 60%總個數，其餘 40%為次佳解。此外，到目前為止，GRB2 在所有的方法中具有領先性，因為 GRB2 皆可以在 30 秒內求出正確的最佳解。

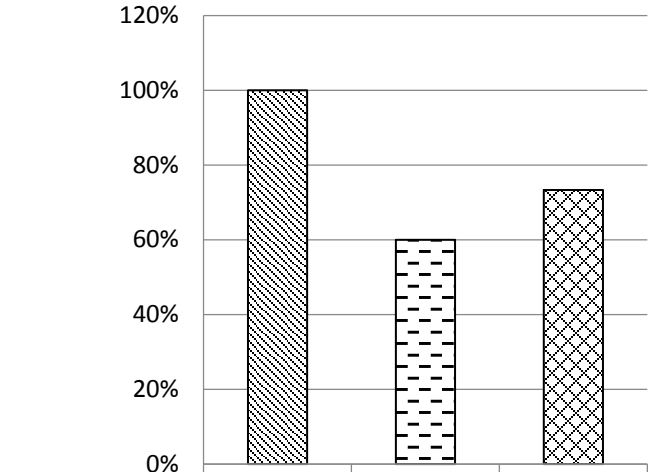



圖 4-3：中型資料集的各方法獲得最佳解情形

在中型資料集，SA 一樣進一步改善 LR 未達到最佳解的 12 個 cases，結果是獲得最佳解的次數為 4，而此資料集的最大 gap 28.57%也被改善至 9.52%，且求解時間一樣快。由於初始解設定以及選擇鄰近解的機制與最後所獲得的最佳組合有一定的影響，因此，此方法在某些情況可能無法求得更好的解，如： $m=300$ 的 case 3，case 4，case 6 已透過 SA 運算，但沒有任何改善。

表 4-8：中型資料集的平均求解時間

| | **GRB1** | **GRB2** | **KSP** | **LR** | **LR+SA** |
|---|---|---|---|---|---|
| $m=100$ | 3601.5s | 1.1s | 記憶體不足 | 0.1s | 0.2s |
| $m=200$ | >10000s | 2.8s | 記憶體不足 | 0.3s | 0.3s |
| $m=300$ | >10000s | 17.7s | 記憶體不足 | 0.4s | 0.4s |

### 4.2.3 大型資料集

在大型資料集 $m=1000$、$m=2000$、$m=3000$，在驗證之前也一樣得進行建立測式檔的步驟。此步驟如同 4.2.1 所敘述，且也不考慮建立測試檔所花的時間。此外，由於 GRB1 在越大的資料集執行效率越差，即便時間長達 10000 秒，GRB1 仍未算出結果，因此，從 $m=1000$ 起，本研究只驗證 GRB2、LR 和 SA 等三種方法。以下表 4-9、4-10 及 4-11 是條列各方法求解的相關結果：

表 4-9：$m=1000$ 時，各方法的求解結果

| $m=1000$ $m_l=50$ $n=20$ $B=95000$ | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Case** | **GRB2** | | **LR** | | | | **LR+SA** | | |
| | **Obj** | **t (s)** | $L^*(\mu)$ | **UB** | **Gap (%)** | **t (s)** | **Obj** | **Gap (%)** | **t (s)** |
| **1** | 71 | 41 | 69.57 | 77 | 8.45 | 2 | 71* | 0 | 6 |
| **2** | 89 | 39 | 87.78 | 90 | 1.12 | 2 | 90 | 1.12 | 6 |
| **3** | 89 | 37 | 87.22 | 99 | 11.24 | 3 | 91 | 2.24 | 7 |
| **4** | 92 | 37 | 91.24 | 101 | 9.78 | 2 | 95 | 3.26 | 6 |
| **5** | 74 | 31 | 72.98 | 85 | 14.86 | 2 | 75 | 1.35 | 6 |
| **6** | 93 | 42 | 92.56 | 96 | 3.23 | 2 | 93* | 0 | 4 |
| **7** | 93 | 46 | 90.81 | 93* | 0 | 2 | 93* | 0 | 2 |
| **8** | 95 | 42 | 93.34 | 119 | 25.26 | 2 | 113 | 18.94 | 6 |
| **9** | 95 | 47 | 93.91 | 96 | 1.05 | 2 | 96 | 1.05 | 6 |
| **10** | 106 | 42 | 105.08 | 110 | 3.77 | 2 | 106* | 0 | 5 |
| **平均** | | 40.4 | | | | 2.1 | | | 5.4 |

表 4-10：$m=2000$ 時，各方法的求解結果

| $m=2000$ $m_l=100$ $n=20$ $B=195000$ | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Case** | **GRB2** | | **LR** | | | | **LR+SA** | | |
| | **Obj** | **t (s)** | $L^*(\mu)$ | **UB** | **Gap (%)** | **t (s)** | **Obj** | **Gap (%)** | **t (s)** |
| **1** | 66 | 258 | 65.45 | 77 | 16.67 | 10 | 67 | 1.51 | 16 |
| **2** | 79 | 345 | 77.8 | 96 | 21.52 | 10 | 92 | 16.45 | 16 |
| **3** | 69 | 318 | 68.2 | 73 | 5.80 | 11 | 69* | 0 | 17 |
| **4** | 75 | 338 | 73.54 | 76 | 1.33 | 11 | 76 | 1.33 | 17 |
| **5** | 81 | 328 | 79.06 | 108 | 33.33 | 10 | 102 | 25.92 | 15 |
| **6** | 73 | 239 | 71.91 | 81 | 10.96 | 11 | 74 | 1.36 | 17 |
| **7** | 91 | 208 | 89.09 | 97 | 6.59 | 12 | 91* | 0 | 18 |
| **8** | 85 | 221 | 83.01 | 95 | 11.76 | 11 | 87 | 2.35 | 17 |
| **9** | 70 | 241 | 68.9 | 71 | 1.43 | 11 | 71 | 1.43 | 16 |
| **10** | 86 | 287 | 85.15 | 88 | 2.33 | 10 | 87 | 1.16 | 16 |
| **平均** | | 278.3 | | | | 10.7 | | | 16.5 |

表 4-11： $m=3000$ 時，各方法的求解結果

| $m=3000$ $m_l=100$ $n=30$ $B=295000$ | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Case** | **GRB2** | | **LR** | | | | **LR+SA** | | |
| | **Obj** | **t (s)** | $L^*(\mu)$ | **UB** | **Gap (%)** | **t (s)** | **Obj** | **Gap (%)** | **t (s)** |
| **1** | 56 | 3820 | 54.94 | 56* | 0 | 32 | 56* | 0 | 32 |
| **2** | 61 | 890 | 60.49 | 69 | 13.11 | 30 | 64 | 4.91 | 43 |
| **3** | 51 | 1044 | 50.26 | 52 | 1.96 | 33 | 52 | 1.96 | 46 |
| **4** | 44 | 964 | 43.14 | 47 | 6.82 | 33 | 45 | 2.27 | 46 |
| **5** | 49 | 1011 | 48.07 | 50 | 2.04 | 31 | 50 | 2.04 | 43 |
| **6** | 52 | 801 | 50.8 | 59 | 13.46 | 32 | 53 | 1.92 | 45 |
| **7** | 50 | 3715 | 49.22 | 50* | 0 | 31 | 50* | 0 | 31 |
| **8** | 56 | 937 | 55.39 | 58 | 3.57 | 32 | 57 | 1.78 | 45 |
| **9** | 55 | 913 | 54.95 | 62 | 12.73 | 32 | 58 | 5.45 | 45 |
| **10** | 51 | 983 | 50.43 | 51* | 0 | 32 | 51* | 0 | 32 |
| **平均** | | 1507.8 | | | | 31.8 | | | 40.8 |

根據表 4-9、4-10 及 4-11 所示，GRB2 在各 case 皆可求出最佳解。但隨著測試資料變大，GRB2 的求解時間會逐漸提升，但此提升在 $m=3000$ 時變得求解時間長度不穩定，在 $m=3000$ 時，本測試所記錄 GRB2 最差的時間長達 3820 秒，而最好的記錄只需要 801 秒即可求出最佳解。

同時，LR 方法也因為測試資料的大小而有所影響。首先，最佳解的機率明顯降低，只占 13.33%總個數，甚至在 $m=2000$ 時，此機率為 0。第二，此方法出現較差的解，使得 gap 增大，如表 4-10 的 case 5 所示，本測試所記錄的最佳解為 81，可是最好的 UB 只更新到 108。雖然 LR 求出最佳解的機率不高，但執行速度卻比 GRB2 快很多，如表 4-12 所示，在 $m=3000$ 時，GRB2 求解時間平均為 1507.8 秒，LR 平均只需花 31.8 秒(s)即可獲得最佳解或次佳解。

根據表 4-9、4-10 及 4-11 可知，SA 改善的效果不是很穩定，有時可獲得最解，有時改善後 gap 還是很大，如： $m=2000$ 的 case 5 改善後的 gap 是 25.92%，甚至有時 SA 無法找出更好的解。但總言而之，以 LR 的 UB 設為初始解的 SA 大部分可以在有效的時間內求得比 LR 更好的解，甚至最佳解。

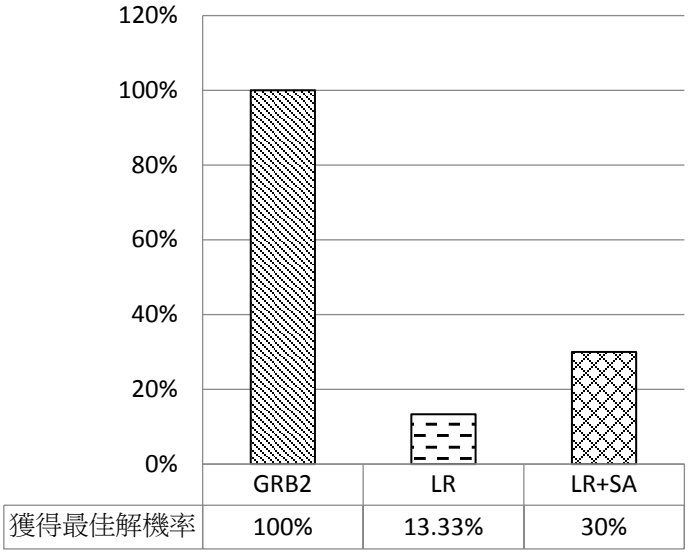



圖 4-4：大型資料集的各方法獲得最佳解情形

表 4-12：大型資料集的平均求解時間

| | **GRB1** | **GRB2** | **KSP** | **LR** | **LR+SA** |
|---|---|---|---|---|---|
| $m=1000$ | >10000s | 40.4s | 記憶體不足 | 2.1s | 5.4s |
| $m=2000$ | >10000s | 278.3s | 記憶體不足 | 10.7s | 16.5s |
| $m=3000$ | >10000s | 1507.8s | 記憶體不足 | 31.8s | 41.8s |

## 4.3 小結

本章首先介紹基本參數設定以及各演算法的相關參數設定，接著針對較小的資料集、中型的資料集、大型的資料集進行驗證兩個整數規劃模式: GRB1、GRB2，以及 3 個演算法: LR、KSP、LR+KSP 對本問題的求解效率與效能。在數值測試過程中發現，若同時考慮解的品質以及運算時間，則 GRB2 是最好的選擇；若只考慮求解時間，且該解的 gap 落在可接受的範圍內，則 LR+SA 也是一個不錯的方法。對於 GRB1 和 KSP，前者由於限制式過多導致在較大的資料集，求解效率和品質甚差，而後者因為隨著測試資料變大，而需考慮的可行解十分多，使得無法在有限的資源內求的最佳解。

# 第五章

# 結論與未來研究方向

## 5.1 結論與貢獻

公路是國家經濟發展的重要命脈。而公路鋪面常會因管線挖埋、天候與交通等因素受到破損，因此鋪面養護規劃對於人們的交通安全十分重要。但面臨路段損壞的多樣性以及資金短缺的情況下，如何在有限的經費下規劃出適當的鋪面養護策略使得道路的風險性降到最低是本研究所關心的課題。

本研究針對具有風險程度不同的一維連續路段，同時考量道路的整體安全性以及有限的維修經費，建構整數規劃模式來求解路段分群問題。本模式以 Wang, Tsai, & Li (2011)所發展的模式為主要架構，並加上最佳的修路方案可能存在某些路段不修的假設，但以追求最小化路段的風險程度為主要的目標，而非最小化維修成本。

本研究的路段分群問題與無容量限制的設施區位問題(UFLP)非常相似，因此亦為一困難問題，不容易在短時間內獲得最佳解。我們將原模式轉換為網路模式，並發現該問題可被視為一種含額外限制式的最短路徑問題(CSP)。本研究先發展拉氏鬆弛法(LR)以及第 K 短路徑演算法(KSP)來求解該 CSP。LR 放鬆額外限制而求解最短路徑子問題，接著透過次梯度法不斷地提升下限和更新上限，最後將原問題的現行解逼近最佳解範圍。而第 KSP 則將搜尋起訖點間的第一短、第二短…第 K 短路徑，並一一檢查最佳路徑是否符合預算上限的限制式。

根據第四章的數值測試，本研究發現 KSP 只能在特別小的資料集顯著可行，但由於無法預知在第幾短路徑將會找到最佳路徑導致其求解時間不甚穩定。若以 Gurobi 最佳化軟體求解 UFLP 的整數規劃模式 GRB1 求解，則將因該模式的限制式過多，隨著測試資料增大時，此方法無論是求解時間還是解的品質都明顯較差。若是以 Gurobi 最佳化軟體直接求解 CSP 的整數規劃模式 GRB2 求解，其求解效率和品質十分優異，但在處理更大的資料集時（譬如包含 10000 個以上的路段），預估其在求解過程可能會發生記憶體不足而無法獲得最佳解。以 LR 求解經驗來看，由於其藉由夾緊上下限的方式來尋找最佳解，若上下界夾緊程度不高，則最後結果會存在 duality gap。若我

們以 LR 所得的最好的解當成模擬退火法(SA)的初始解，經由模擬退火法的求解機制可再進一步改善其解之品質，測試結果發現此種 LR+SA 作法對解的改善程度尚可。若只考慮求解時間，且該解的 gap 落在可接受的範圍內，則 LR+SA 也是一個不錯的方法。

## 5.2 未來研究方向

至目前為止，本研究雖對鋪面養護規劃做整體探討，但仍有未臻完善之處，以下列舉幾個未來可延伸的議題：

1. **探討在平面上的道路**：本研究為了簡化模式之建構，假設所需養護的道路為少彎曲的高速公路，較不符合實際現況。未來可以考慮道路在平面地圖上的實際位置，此時所需分群的路段將成為二維空間的路段集合。
2. **加入時間的考量**：本研究的路段集群問題尚未考量鋪面養護的長期規劃之因素，假設以三年做為鋪面養護工程所需規劃的時間，且此規劃將分成三期，每期以一年為單位。在交通流量對於路段破損的影響程度已知的情況下，假設前一期所選擇的修復方式會影響下一期的修復方式，那麼在第一年、第二年、第三年如何進行養護規劃才能使得總三年的風險程度最小。
3. **求解 SA 最短路徑的問題**：為了簡化求解經過給定節線的最短路徑之問題，本研究藉由求解局部最短路徑來獲得整體的最短路徑。但由於 SA 演算法每一回合將從現行解的 0-1 組合只選出一個 1 或一個 0 來進行轉換，使得只有少數的部分網路有所變動。因此，我們推測若只針對變動的網路範圍重新計算最短路徑，而保留其它部分網路無變動的運算結果，或可更快速獲得新的最短路徑。假設 A 和 B 皆為不修路段的位置，且 A>B，從 A 至 B 的範圍為受影響的範圍。以下將針對現行解的維修成本是否大於預算上限的情形進行描述不同 0 或 1 的選擇會產生不同的變動：

   **情形 1：當現行解的維修成本大於預算上限時，SA 必須選出一個 1 並將它轉為 0**: 假設被選定的某一路段位置為 C，當該路段從 1 值轉換成 0 值時，某一 A 至 B 的範圍會依據不同 C 的位置而發生三種的變化(1)若 C 位於 A 與 B 之間，並且非與 A 相鄰或 B 相鄰，轉換前 A 至 B 的範圍則被分割成 A 至 C 和 C 至 B 的範圍(如圖 5-1(a))；(2)若 C 位於 A 與 B 之間，並且與 A 相

鄰，轉換前 A 至 B 的範圍則變成 C 至 B 的範圍(如圖 5-1(b))；(3) 若 C 位於 A 與 B 之間，並且與 B 相鄰，轉換前 A 至 B 的範圍則變成 A 至 C 的範圍(如圖 5-1(c))。


圖 5-1：當 SA 選出一個 1 並將它轉為 0 時，各種可能變動的情況

**情形 2：當現行解的維修成本小於或等於預算上限時，必須選出一個 0 並將它轉為 1**。假設被選定的某一路段位置為 C，當該路段從 0 值轉換成 1 值時，某一 A 至 B 的範圍會依據不同 C 的位置而發生三種的變化(1) 若 C 位於 A 與 B 之間，轉換前 A 至 C 與 C 至 B 的範圍則變成 A 至 B 的範圍(如圖 5-2(a))；(2) 若 C 位於 A 的位置，並且有一個不修路段 D 與 A 相鄰，且 D<A，則轉換前 A 至 B 的範圍則變成 D 至 B 的範圍(如圖 5-2(b))；(3) 若 C 位於 B 的位置，並且有一個不修路段 D 與 B 相鄰，且 B<D，則轉換前 A 至 B 的範圍則變成 A 至 D 的範圍(如圖 5-2(c))。

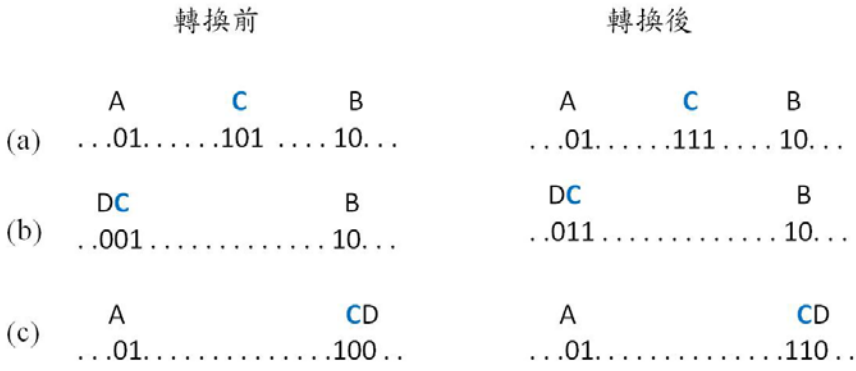



圖 5-2：當 SA 選出一個 0 並將它轉為 1 時，各種可能變動的情況

# 參考文獻

張志強(1987)，「整數規劃在集群分析上之應用研究」，國立政治大學統計學研究所碩士論文

Ahuja, R.K., Magnanti, T.L., & Orlin, J.B. (1993). Network flows: Theory, algorithms, and applications.

Al-Sultan, K., & Al-Fawzan, M. (1999). A tabu search approach to the uncapacitated facility location problem. Annals of Operations Research, 86, 91-103.

Allwright, J. R., & Carpenter, D. B. (1989). A distributed implementation of simulated annealing for the travelling salesman problem. Parallel Computing, 10(3), 335-338.

Álvarez, P., López-Rodríguez, F., Canito, J.L., Moral, F.J., & Camacho, A. (2007). Development of a measure model for optimal planning of maintenance and improvement of roads. Computers & Industrial Engineering, 52(3), 327-335.

Bellman, R. (1973). A note on cluster analysis and dynamic programming. Mathematical Biosciences, 18(3), 311-312.

Bilde, O., & Krarup, J. (1977). Sharp lower bounds and efficient algorithms for the simple plant location problem. Annals of Discrete Mathematics. v1, 79-88.

Carlyle, W.M., Royset, J.O., & Kevin Wood, R. (2008). Lagrangian relaxation and enumeration for solving constrained shortest-path problems. Networks, 52(4), 256-270.

Cornuéjols, G., Fisher, M., & Nemhauser, G.L. (1977). On the uncapacitated location problem. Annals of Discrete Mathematics, 1, 163-177.

Cornuejols, G., Nemhauser, G. L., & Wolsey, L. A. (1990). The uncapacitated facility location problem. In R. L. Francis & P. B. Mirchandani (Eds.), Discrete location theory (pp. 119–171). New York: Wiley

Drexl, A. (1988). A simulated annealing approach to the multiconstraint zero-one knapsack problem. Computing, 40(1), 1-8.

Erlenkotter, D. (1978). A dual-based procedure for uncapacitated facility location.

Operations Research, 26(6), 992-1009.

Fisher, W.D. (1958). On grouping for maximum homogeneity. Journal of the American Statistical Association, 53(284), 789-798.

Fox, B. L.(1975). $K^{th}$ shortest paths and applications to the probabilistic networks, in ORSA/TIMS Joint National Mtg., Vol. 23, p. B263

Frank, C., & Römer, K. (2007). Distributed facility location algorithms for flexible configuration of wireless sensor networks Distributed computing in sensor systems (pp. 124-141): Springer.

Gao, L.L., & Robinson, E.P. (1992). A dual-based optimization procedure for the two-echelon uncapacitated facility location problem. Naval Research Logistics (NRL), 39(2), 191-212.

Ghosh, D. (2003). Neighborhood search heuristics for the uncapacitated facility location problem. European Journal of Operational Research, 150(1), 150-162.

Guha, S., & Khuller, S. (1998). Greedy strikes back: Improved facility location algorithms. Paper presented at the Proceedings of the ninth annual ACM-SIAM symposium on Discrete algorithms.

Handler, G.Y., & Zang, I. (1980). A dual algorithm for the constrained shortest path problem. Networks, 10(4), 293-309.

Hansen, P., & Jaumard, B. (1997). Cluster analysis and mathematical programming. Mathematical programming, 79(1-3), 191-215.

Jensen, R.E. (1969). A dynamic programming algorithm for cluster analysis. Operations Research, 1034-1057.

Klose, A., & Drexl, A. (2005). Facility location models for distribution system design. European Journal of Operational Research, 162(1), 4-29.

Körkel, M. (1989). On the exact solution of large-scale simple plant location problems. European Journal of Operational Research, 39(2), 157-173.

Lawler, E.L. Combinatorial optimization: Networks and matroids. 1976. Holt, Rinehart and Winston, New York, 3.

Lawler, E.L. (1972). A procedure for computing the k best solutions to discrete

optimization problems and its application to the shortest path problem. Management Science, 18(7), 401-405.

Lazic, N., Frey, B.J., & Aarabi, P. (2010). Solving the uncapacitated facility location problem using message passing algorithms. Paper presented at the International Conference on Artificial Intelligence and Statistics.

Lazic, N., Givoni, I., Frey, B., & Aarabi, P. (2009). Floss: Facility location for subspace segmentation. Paper presented at the Computer Vision, 2009 IEEE 12th International Conference on.

Maffioli, F. (1987). Randomized heuristics for NP-hard problems. Advanced School on Stochastics in Combinatorial Optimization, World Scientific, Dordrecht, 76-93.

Mirzaian, A. (1985). Lagrangian relaxation for the star-star concentrator location problem: Approximation algorithm and bounds. Networks, 15(1), 1-20.

Mulvey, J.M., & Crowder, H.P. (1979). Cluster analysis: An application of lagrangian relaxation. Management Science, 25(4), 329-340.

Nemhauser, G.L., Wolsey, L.A., & Fisher, M.L. (1978). An analysis of approximations for maximizing submodular set functions—I. Mathematical Programming, 14(1), 265-294.

Novick, B. (2009). Norm statistics and the complexity of clustering problems. Discrete Applied Mathematics, 157(8), 1831-1839.

Ouyang, Y., & Madanat, S. (2006). An analytical solution for the finite-horizon pavement resurfacing planning problem. Transportation Research Part B: Methodological, 40(9), 767-778. doi: 10.1016/j.trb.2005.11.001

Rao, M. (1971). Cluster analysis and mathematical programming. Journal of the American statistical association, 66(335), 622-626.

Shier, D.R. (1979). On algorithms for finding the k shortest paths in a network. Networks, 9(3), 195-214.

Shmoys, D.B., Tardos, É., & Aardal, K. (1997). Approximation algorithms for facility location problems. Paper presented at the Proceedings of the twenty-ninth annual ACM symposium on Theory of computing.

Sun, M. (2006). Solving the uncapacitated facility location problem using tabu search. Computers & Operations Research, 33(9), 2563-2589.

Tsai, Y., Yang, C., & Wang, Z. (2006). Spatial clustering for determining economical highway pavement let projects. Proceedings of Geo Congress, 1-6.

Van der Zijpp, N.J., & Fiorenzo Catalano, S. (2005). Path enumeration by finding the constrained k-shortest paths. Transportation Research Part B: Methodological, 39(6), 545-563. doi: 10.1016/j.trb.2004.07.004

Verter, V. (2011). Uncapacitated and capacitated facility location problems Foundations of location analysis (pp. 25-37): Springer.

Vinod, H.D. (1969). Integer programming and the theory of grouping. Journal of the American Statistical Association, 64(326), 506-519.

Wang, I.L., Tsai, Y.-C.J., & Li, F. (2011). A network flow model for clustering segments and minimizing total maintenance and rehabilitation cost. Computers & Industrial Engineering, 60(4), 593-601.

Xiao, Y., Thulasiraman, K., Xue, G., & Jüttner, A. (2005). The constrained shortest path problem: Algorithmic approaches and an algebraic study with generalization. submitted for publication.

Yang, C., Tsai, Y.J., & Wang, Z. (2009). Algorithm for spatial clustering of pavement segments. Computer-Aided Civil and Infrastructure Engineering, 24(2), 93-108.

Yen, J.Y. (1971). Finding the k shortest loopless paths in a network. management Science, 17(11), 712-716.

# 附錄 A、GRB1 求解數值

表 A-1：小型資料集之 GRB1 求解數值表

| $m=30$ | | | | |
|---|---|---|---|---|
| **Case** | **目標值** | **總維修成本** | **設立群組** | **總求解時間** |
| **1** | 15 | 2298 | 9 | 439 |
| **2** | 10 | 2297 | 8 | 530 |
| **3** | 2 | 2298 | 12 | 225 |
| **4** | 10 | 2293 | 9 | 314 |
| **5** | 1 | 2283 | 11 | 354 |
| **6** | 1 | 2292 | 9 | 75 |
| **7** | 7 | 2293 | 10 | 763 |
| **8** | 3 | 2284 | 11 | 77 |
| **9** | 2 | 2298 | 13 | 317 |
| **10** | 6 | 2274 | 12 | 194 |

表 A-2：中型資料集之 GRB1 求解數值表

| $m=100$ | | | | |
|---|---|---|---|---|
| **Case** | **目標值** | **總維修成本** | **設立群組** | **總求解時間** |
| **1** | 13 | 8462 | 7 | 4884 |
| **2** | 25 | 8480 | 7 | 3225 |
| **3** | 19 | 8493 | 8 | 1855 |
| **4** | 20 | 8500 | 8 | 2988 |
| **5** | 36 | 8478 | 7 | 1636 |
| **6** | 30 | 8472 | 8 | 1364 |
| **7** | 13 | 8441 | 8 | 4491 |
| **8** | 29 | 8488 | 7 | 5091 |
| **9** | 23 | 8471 | 7 | 6353 |
| **10** | 21 | 8485 | 8 | 4128 |

# 附錄 B、GRB2 求解數值

表 B-1：小型資料集之 GRB2 求解數值表

| $m=30$ | | | | |
|---|---|---|---|---|
| **Case** | **目標值** | **總維修成本** | **設立群組** | **總求解時間** |
| **1** | 15 | 2298 | 9 | 1 |
| **2** | 10 | 2291 | 8 | 0 |
| **3** | 2 | 2280 | 10 | 0 |
| **4** | 10 | 2247 | 6 | 0 |
| **5** | 1 | 2241 | 8 | 0 |
| **6** | 1 | 2282 | 8 | 1 |
| **7** | 7 | 2294 | 10 | 0 |
| **8** | 3 | 2284 | 11 | 0 |
| **9** | 2 | 2256 | 8 | 1 |
| **10** | 6 | 2248 | 10 | 0 |

表 B-2：中型資料集之 GRB2 求解數值表

| | **Case** | **目標值** | **總維修成本** | **設立群組** | **總求解時間** |
|---|---|---|---|---|---|
| $m=100$ | **1** | 13 | 8462 | 8 | 1 |
| | **2** | 25 | 8480 | 8 | 1 |
| | **3** | 19 | 8481 | 8 | 1 |
| | **4** | 20 | 8500 | 8 | 1 |
| | **5** | 36 | 8479 | 6 | 1 |
| | **6** | 30 | 8500 | 8 | 1 |
| | **7** | 13 | 8441 | 8 | 1 |
| | **8** | 29 | 8491 | 7 | 2 |
| | **9** | 23 | 8491 | 7 | 1 |
| | **10** | 21 | 8485 | 8 | 1 |

| | | | | | |
|---|---|---|---|---|---|
| $m = 200$ | **1** | 17 | 17994 | 15 | 2 |
| | **2** | 13 | 17965 | 16 | 3 |
| | **3** | 7 | 17966 | 16 | 3 |
| | **4** | 23 | 17998 | 15 | 2 |
| | **5** | 19 | 17992 | 15 | 2 |
| | **6** | 23 | 17963 | 12 | 2 |
| | **7** | 15 | 18000 | 15 | 4 |
| | **8** | 7 | 17989 | 14 | 2 |
| | **9** | 11 | 17944 | 14 | 3 |
| | **10** | 18 | 17972 | 15 | 5 |
| $m = 300$ | **1** | 13 | 26974 | 21 | 15 |
| | **2** | 15 | 26963 | 22 | 13 |
| | **3** | 20 | 26976 | 23 | 8 |
| | **4** | 31 | 26984 | 21 | 26 |
| | **5** | 33 | 26995 | 21 | 19 |
| | **6** | 13 | 26992 | 22 | 20 |
| | **7** | 26 | 26959 | 20 | 22 |
| | **8** | 27 | 26959 | 20 | 19 |
| | **9** | 31 | 26999 | 21 | 26 |
| | **10** | 31 | 26948 | 21 | 9 |

表 B-3：大型資料集之 GRB2 求解數值表

| | **Case** | **目標值** | **總維修成本** | **設立群組** | **總求解時間** |
|---|---|---|---|---|---|
| $m = 1000$ | **1** | 71 | 94968 | 17 | 41 |
| | **2** | 89 | 94983 | 17 | 39 |
| | **3** | 89 | 94999 | 16 | 37 |
| | **4** | 92 | 94987 | 18 | 37 |
| | **5** | 74 | 94968 | 16 | 31 |
| | **6** | 93 | 94995 | 15 | 42 |
| | **7** | 93 | 94967 | 15 | 46 |

| | | | | | |
|---|---|---|---|---|---|
| | **8** | 95 | 94998 | 15 | 42 |
| | **9** | 95 | 94992 | 15 | 47 |
| | **10** | 106 | 94977 | 15 | 42 |
| $m = 2000$ | **1** | 66 | 194996 | 18 | 258 |
| | **2** | 79 | 195000 | 16 | 345 |
| | **3** | 69 | 194988 | 16 | 318 |
| | **4** | 75 | 194958 | 16 | 338 |
| | **5** | 81 | 194998 | 16 | 328 |
| | **6** | 73 | 194973 | 16 | 239 |
| | **7** | 91 | 194943 | 15 | 208 |
| | **8** | 85 | 194963 | 18 | 221 |
| | **9** | 70 | 194992 | 16 | 241 |
| | **10** | 86 | 194998 | 15 | 287 |
| $m = 3000$ | **1** | 56 | 294955 | 25 | 3820 |
| | **2** | 61 | 294999 | 22 | 890 |
| | **3** | 51 | 294989 | 23 | 1044 |
| | **4** | 44 | 294994 | 23 | 964 |
| | **5** | 49 | 294982 | 23 | 1011 |
| | **6** | 52 | 294947 | 24 | 801 |
| | **7** | 50 | 294965 | 23 | 3715 |
| | **8** | 56 | 294973 | 24 | 937 |
| | **9** | 55 | 295000 | 24 | 913 |
| | **10** | 51 | 294973 | 22 | 983 |

# 附錄 C、KSP 求解數值

表 C-1：小型資料集之 KSP 求解數值表

| $m = 30$ | | | | |
|---|---|---|---|---|
| **Case** | **目標值** | **總維修成本** | **設立群組** | **總求解時間** |
| **1** | 15* | 2298 | 9 | 212 |
| **2** | 10* | 2297 | 8 | 2810 |
| **3** | 2* | 2299 | 10 | 49 |
| **4** | 10* | 2300 | 8 | 1400 |
| **5** | 1* | 2289 | 7 | 263 |
| **6** | 1* | 2297 | 9 | 46 |
| **7** | 7* | 2295 | 9 | 1254 |
| **8** | 3* | 2284 | 11 | 286 |
| **9** | 2* | 2293 | 11 | 184 |
| **10** | 6* | 2300 | 11 | 1014 |

# 附錄 D、LR 求解數值

表 D-1：小型資料集之 LR 求解數值表

| $m=30$ | | | | | | |
|---|---|---|---|---|---|---|
| **Case** | $L^*(\mu)$ | **UB** | **Gap(%)** | **總維修成本** | **設立群組** | **總求解時間** |
| **1** | 14 | 16 | 6.67 | 2291 | 10 | 0 |
| **2** | 9.72 | 10* | 0 | 2262 | 7 | 0 |
| **3** | 1.71 | 2* | 0 | 2280 | 10 | 0 |
| **4** | 8.62 | 10* | 0 | 2247 | 6 | 0 |
| **5** | 0.43 | 1* | 0 | 2241 | 8 | 0 |
| **6** | 0.81 | 1* | 0 | 2282 | 8 | 0 |
| **7** | 6.76 | 9 | 28.57 | 2245 | 6 | 0 |
| **8** | 2.65 | 3* | 0 | 2284 | 11 | 0 |
| **9** | 1.44 | 2* | 0 | 2256 | 8 | 0 |
| **10** | 4.86 | 6* | 0 | 2248 | 10 | 0 |

表 D-2：中型資料集之 LR 求解數值表

| | **Case** | $L^*(\mu)$ | **UB** | **Gap(%)** | **總維修成本** | **設立群組** | **總求解時間** |
|---|---|---|---|---|---|---|---|
| | **1** | 11.86 | 13* | 0 | 8462 | 7 | 0 |
| | **2** | 23.46 | 30 | 20 | 8390 | 7 | 0 |
| | **3** | 18.34 | 19* | 0 | 8481 | 7 | 0 |
| | **4** | 19.06 | 20* | 0 | 8500 | 8 | 0 |
| $m=100$ | **5** | 34.82 | 39 | 8.33 | 8423 | 7 | 1 |
| | **6** | 27.6 | 32 | 6.67 | 8458 | 8 | 0 |
| | **7** | 10.98 | 13* | 0 | 8441 | 8 | 0 |
| | **8** | 25.86 | 32 | 10.34 | 8500 | 8 | 0 |
| | **9** | 20.91 | 25 | 8.70 | 8485 | 7 | 0 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | **10** | 19.35 | 27 | 28.57 | 8309 | 8 | 0 |
| $m = 200$ | **1** | 16.14 | 18 | 5.88 | 17908 | 15 | 0 |
| | **2** | 12.2 | 13* | 0 | 17965 | 16 | 0 |
| | **3** | 6.56 | 7* | 0 | 17966 | 16 | 1 |
| | **4** | 22.28 | 27 | 17.39 | 17914 | 15 | 0 |
| | **5** | 18.78 | 19* | 0 | 17992 | 15 | 0 |
| | **6** | 22.01 | 23* | 0 | 17963 | 12 | 0 |
| | **7** | 14.99 | 15* | 0 | 18000 | 15 | 1 |
| | **8** | 6.52 | 7* | 0 | 17973 | 15 | 1 |
| | **9** | 9.98 | 11* | 0 | 17944 | 14 | 0 |
| | **10** | 17.37 | 18* | 0 | 17972 | 15 | 0 |
| $m = 300$ | **1** | 12.63 | 13* | 0 | 27974 | 21 | 1 |
| | **2** | 14.48 | 15* | 0 | 27963 | 22 | 0 |
| | **3** | 19.1 | 22 | 10 | 27984 | 23 | 1 |
| | **4** | 30.06 | 34 | 9.68 | 27964 | 22 | 0 |
| | **5** | 32.87 | 33* | 0 | 27995 | 20 | 1 |
| | **6** | 12.68 | 15 | 15.38 | 27970 | 22 | 0 |
| | **7** | 24.97 | 26* | 0 | 27959 | 20 | 0 |
| | **8** | 26.07 | 27* | 0 | 27959 | 19 | 0 |
| | **9** | 30.61 | 32 | 3.23 | 27935 | 22 | 1 |
| | **10** | 29.64 | 31* | 0 | 27948 | 21 | 0 |

表 D-3：大型資料集 LR 求解數值表

| | **Case** | $L^*(\mu)$ | **UB** | **Gap(%)** | **總維修成本** | **設立群組** | **總求解時間** |
|---|---|---|---|---|---|---|---|
| $m = 1000$ | **1** | 69.57 | 77 | 8.45 | 94798 | 16 | 2 |
| | **2** | 87.78 | 90 | 1.12 | 94941 | 17 | 2 |
| | **3** | 87.22 | 99 | 11.24 | 94712 | 15 | 3 |
| | **4** | 91.24 | 101 | 9.78 | 94730 | 17 | 2 |
| | **5** | 72.98 | 85 | 14.86 | 94614 | 16 | 2 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | **6** | 92.56 | 96 | 3.23 | 94897 | 15 | 2 |
| | **7** | 90.81 | 93* | 0 | 94946 | 16 | 2 |
| | **8** | 93.34 | 119 | 25.26 | 94993 | 16 | 2 |
| | **9** | 93.91 | 96 | 1.05 | 94940 | 15 | 2 |
| | **10** | 105.08 | 110 | 3.77 | 94877 | 15 | 2 |
| $m = 2000$ | **1** | 65.45 | 77 | 16.67 | 194619 | 17 | 10 |
| | **2** | 77.8 | 96 | 21.52 | 194965 | 16 | 10 |
| | **3** | 68.2 | 73 | 5.80 | 194815 | 15 | 11 |
| | **4** | 73.54 | 76 | 1.33 | 194918 | 15 | 11 |
| | **5** | 79.06 | 108 | 33.33 | 194922 | 17 | 10 |
| | **6** | 71.91 | 81 | 10.96 | 194725 | 17 | 11 |
| | **7** | 89.09 | 97 | 6.59 | 194745 | 15 | 12 |
| | **8** | 83.01 | 95 | 11.76 | 194654 | 17 | 11 |
| | **9** | 68.9 | 71 | 1.43 | 194934 | 16 | 11 |
| | **10** | 85.15 | 88 | 2.33 | 194906 | 15 | 10 |
| $m = 3000$ | **1** | 54.94 | 56* | 0 | 294955 | 24 | 32 |
| | **2** | 60.49 | 69 | 13.11 | 294660 | 22 | 30 |
| | **3** | 50.26 | 52 | 1.96 | 294915 | 23 | 33 |
| | **4** | 43.14 | 47 | 6.82 | 294823 | 23 | 33 |
| | **5** | 48.07 | 50 | 2.04 | 294914 | 23 | 31 |
| | **6** | 50.8 | 59 | 13.46 | 294614 | 24 | 32 |
| | **7** | 49.22 | 50* | 0 | 294965 | 23 | 31 |
| | **8** | 55.39 | 58 | 3.57 | 294884 | 24 | 32 |
| | **9** | 54.95 | 62 | 12.73 | 294648 | 24 | 32 |
| | **10** | 50.43 | 51* | 0 | 294973 | 22 | 32 |

# 附錄 E、SA 求解數值

表 E-1：小型資料集之 SA 求解數值表

| $m=30$ | | | | | | |
|---|---|---|---|---|---|---|
| **Case** | **初始解目標值** | **目標值** | **Gap(%)** | **總維修成本** | **設立群組** | **總求解時間** |
| 1 | 16 | 15* | 0 | 2298 | 9 | 0 |
| 7 | 9 | 8 | 14.28 | 2297 | 8 | 0 |

表 E-2：中型資料集之 SA 求解數值表

| | **Case** | **初始解目標值** | **目標值** | **Gap(%)** | **總維修成本** | **設立群組** | **總求解時間** |
|---|---|---|---|---|---|---|---|
| $m=100$ | 2 | 30 | 25* | 0 | 8480 | 7 | 0 |
| | 5 | 39 | 39 | 8.33 | 8423 | 7 | 1 |
| | 6 | 32 | 30* | 0 | 8496 | 7 | 0 |
| | 8 | 32 | 32 | 10.34 | 8500 | 8 | 0 |
| | 9 | 25 | 24 | 4.34 | 8485 | 7 | 0 |
| | 10 | 27 | 23 | 9.52 | 8433 | 7 | 0 |
| $m=200$ | 1 | 18 | 17* | 0 | 17998 | 15 | 0 |
| | 4 | 27 | 26 | 13.04 | 17989 | 15 | 0 |
| $m=300$ | 3 | 22 | 22 | 10 | 26984 | 23 | 1 |
| | 4 | 34 | 34 | 9.68 | 26964 | 23 | 0 |
| | 6 | 15 | 15 | 15.38 | 26970 | 23 | 0 |
| | 9 | 32 | 31* | 0 | 26999 | 22 | 1 |

表 E-3：大型資料集之 SA 求解數值表

| | Case | 初始解目標值 | 目標值 | Gap(%) | 總維修成本 | 設立群組 | 總求解時間 |
|---|---|---|---|---|---|---|---|
| | 1 | 77 | 71* | 0 | 94998 | 16 | 6 |
| | 2 | 90 | 90 | 1.12 | 94941 | 17 | 6 |
| | 3 | 99 | 91 | 2.24 | 94954 | 15 | 7 |
| | 4 | 101 | 95 | 3.26 | 94998 | 17 | 6 |
| $m=1000$ | 5 | 85 | 75 | 1.35 | 94976 | 16 | 6 |
| | 6 | 96 | 93* | 0 | 94995 | 15 | 4 |
| | 8 | 119 | 113 | 18.94 | 94988 | 16 | 6 |
| | 9 | 96 | 96 | 1.05 | 94940 | 15 | 6 |
| | 10 | 110 | 106* | 0 | 94977 | 15 | 5 |
| | 1 | 77 | 67 | 1.51 | 194989 | 16 | 16 |
| | 2 | 96 | 92 | 16.45 | 194965 | 16 | 16 |
| | 3 | 73 | 69* | 0 | 194999 | 15 | 17 |
| | 4 | 76 | 76 | 1.33 | 194918 | 15 | 17 |
| $m=2000$ | 5 | 108 | 102 | 25.92 | 194990 | 16 | 15 |
| | 6 | 81 | 74 | 1.36 | 194993 | 16 | 17 |
| | 7 | 97 | 91* | 0 | 194944 | 15 | 18 |
| | 8 | 95 | 87 | 2.35 | 194954 | 17 | 17 |
| | 9 | 71 | 71 | 1.43 | 194934 | 16 | 16 |
| | 10 | 88 | 87 | 1.16 | 194997 | 15 | 16 |
| | 2 | 69 | 64 | 4.91 | 294944 | 22 | 43 |
| | 3 | 52 | 52 | 1.96 | 294915 | 23 | 46 |
| | 4 | 47 | 45 | 2.27 | 294979 | 23 | 46 |
| $m=3000$ | 5 | 50 | 50 | 2.04 | 294914 | 23 | 43 |
| | 6 | 59 | 53 | 1.92 | 294968 | 24 | 45 |
| | 8 | 58 | 57 | 1.78 | 294983 | 24 | 45 |
| | 9 | 62 | 58 | 5.45 | 294993 | 22 | 45 |