國 立 成 功 大 學

工 業 與 資 訊 管 理 學 系 碩 士 班

碩 士 論 文

考慮依時變化的節點或起訖組合需求量之網路節點修復排程

A node restoration scheduling problem considering time-dependent demands on nodes or origin-destination pairs

研 究 生：方思涵

指導教授：王逸琳

中華民國一百零七年五月

國立成功大學

碩士論文

考慮依時變化的節點或起訖組合需求量之網路節點修復排程

A node restoration scheduling problem considering time-dependent demands on nodes or origin-destination pairs

研究生：方思涵

本論文業經審查及口試合格特此證明

論文考試委員：

許瑞麟　王逸琳

李宇欣

指導教授：王逸琳

系(所)主管：

中　華　民　國　107　年　5　月　26　日

摘要

日常生活中有許多服務必須在特定的地點與設施上進行，譬如自動櫃員機（ATM）、通信基地台、交通場站等皆可視為某些服務系統網路之節點。而各節點上的服務需求量除可能依時而變外，可再分為「單一節點」與「起訖組合」兩類，前者之服務需求僅發生在該節點上（譬如銀行、ATM 等）；而後者則有賴起訖兩節點皆無故障時，方能滿足其起訖需求（譬如機場、YouBike 租借站等）。假設各節點或起訖組合之依時變化需求量皆為已知或可被預測，但部分（或全部）的節點卻因天災人禍而毀損故障時，在故障期間於這些節點所發生之需求量將被捨棄（譬如某 ATM 當機導致無法領錢；或 YouBike 場站故障導致無法租還車等）。由於修復系統之人力或機具資源相對有限（假設共有 K 組維修團隊），本研究探討之節點修復排程問題旨在探討如何有效地指揮調度 K 組維修團隊去修復毀損之節點，以儘快讓更多人能更早享受原有之服務，極大化修復過程中可提供的服務效益。

由於我們考量 K 組維修團隊之各隊將逐一修復所有之故障節點，此與文獻中的 K 組旅行維修員問題（k-Traveling Repairmen Problem，k-TRP）相似，但本研究的目標函式還多考慮了節點需求的因時變動與即時性，較已為 NP-hard 之 k-TRP 更符合實務需求，因此其挑戰性與困難度亦更高。在可得理論精確解之數學規劃模式設計方面，我們提出三種整數規劃模式：(1)含排除子迴圈限制式(subtour elimination)之模式；(2)以時空網路(Time-Space Network)為基礎建構之模式，而後又可再依是否考慮旅行時間而再細分成「特例模式」及「一般模式」兩類；(3)以層空網路(Level-Space Network)為基礎建構之模式，其中(1)(3)可由 k-TRP 之數學模式為基礎來建構而得。

隨著網路規模擴大，數學規劃模式之求解過程極為耗時，因此本研究又提出結合貪婪演算法(Greedy Algorithm)與模式求解的數學啟發式演算法（Matheuristics），以優先修復最小損失需求的節點為選取機制，同時考量節點維修時間、旅行時間及依時而變之需求量，以演算法求得之數組層空網路路徑為基礎，反向建構出其對應之限縮模

式網路，再進一步提出該網路以節線或路徑為變數等兩種整數規劃模式求解。

最後進行數學模式及演算法測試，若忽略節點間之旅行時間，則「單一節點」與「起訖組合」兩類問題之測試結果顯示，以時空網路為基礎所建構之模式有最好的求解表現；而在限縮模式網路下，則以路徑為變數所建構之模式求解效率較佳，針對單一節點需求類型問題，可在短時間內求得近似最佳解(Gap<1%)，但仍無法處理較大規模問題。至於起訖組合需求類型，大概只能靠貪婪演算法之解再以模擬退火法進一步改善之。因為我們建構限縮模式的數學啟發式演算法十分創新，希望未來能再利用類似想法處理其它類似的路徑規劃或排程問題。

關鍵字：節點修復排程、*K* 組旅行維修員問題、依時變化需求量、整數規劃、數學啟發式演算法

A node restoration scheduling problem considering time-dependent demands on nodes or origin-destination pairs

Sih-Han Fang
I-Lin Wang
Department of Industrial and Information Management

SUMMARY

We investigate node restoration scheduling problems for damaged service networks. Suppose all or some nodes in a service network are damaged by man-made or natural disasters, and the forecasted time-dependent demands on single nodes or between node pairs are known beforehand. We seek mathematical models and solution methods to schedule *k* teams to restore all damaged nodes, so that the demands after restoration can be maximized. Two demand types are considered: (1) node-only type, and (2) Origin-Destination (OD) pair type. The former focuses on demands at single working node only, whereas the latter considers demands between OD node pairs. Our problems are NP-hard, since the NP-hard *k*-Traveling Repairmen Problem corresponds to a special case of ours. Three integer programming models based on Subtour Elimination, Time-Space network, and Level-Space network are proposed, but very time consuming. We give an effective greedy heuristics, and use it to design matheheuristics that construct reduced formulations in both arc and path forms. Finally we give a Simulated Annealing heuristic (SA) to deal with larger problems. Computational experiments indicate the time-space models are the best to calculate the exact optimal solutions for node-only demand type problems of small to medium sizes. Our matheuristics, especially the path-form ones, can improve the solution quality of greedy heuristic for larger problems. For OD pair demand type problems, only the greedy heuristic, further improvable by SA, can deal with problems of medium to large sizes.

Key words: Node Restoration Scheduling, *k*-Traveling Repairmen Problem, Time-dependent demand, Integer Programming, Matheuristics

INTRODUCTION

A successful service system that aims at providing convenient access to the public usually deploy many service facilities in the service area. If we treat a deployed service facility such as an Automated Teller Machine (ATM), vending machine, convenient store, cell site, or a vehicle sharing station, as a service node, possible connection between service nodes as service arcs, then we can view a service system as a service network. There are two kinds of service demands. The first type is the node-only service where customer demands take place at working nodes alone. For example, an ATM, vending machine or convenient store provides this service type. For this type of service, the customer demands can be satisfied as long as they take place on functional nodes. The second type of service, called as an origin-destination (OD) pair service, requires both end nodes (origin and destination) to be functional at the same time. For example, a wireless communication between two points requires their connected cell sites to be functional at the same time. Similarly, a successful shared vehicle rental from an origin rental site to a destination site can only be satisfied while both origin and destination nodes are functional, too.

Note that these node demands are usually time dependent, which means the amount of demands on the same node may vary over time. Take a vending machine nearby a school for example, one may expect more demands in the periods near the school opening and closing times. Similarly, in a bike sharing system, usually 7-9am, 11am-1pm, 5-7pm, and 8-10pm have more rental demands than other periods.

We investigate an optimal schedule to restore these damaged nodes using limited restoration resources (e.g., k restoration teams) so that more node demands, whether for single nodes or for node pairs, can be satisfied during the entire restoration planning horizon T. This is a very difficult scheduling problem, based on the following observations: (1) assuming each node can only be processed by a single restoration team, and the scheculding is non-preemptive (i.e., once the restoration starts, it cannot be interrupted and has to be finished by the same time who starts the restoration), then this problem is similar to calculate the routings for these k teams, which is very related with the NP-hard k-Traveling Salesmen Problem (k-TSP) or k-Traveling Repairmen Problem (k-TRP). In particular, both k-TSP and k-TRP look for optimal routings for k vehicles (teams) that cover all nodes, whereas k-TSP considers only the travel time between nodes for minimum makespan but k-TRP further considers the staying time (i.e., the restoration time) on each node for minimum total waiting time. So, our problem is at least as hard as k-TSP or k-TRP, which is NP-hard. (2) the time-dependent demands further complicate the problem much more, since k-TSP and k-TRP only considers some constant one-time node demands, where the node demands are satisfied as long as that node has been visited. Yet now we consider the demands change over time,

which means the demands to be satisfied is a nonlinear decreasing function with possible breakpoints on every unit of time. To model this time-dependent demands, one must keep track of the exact arrival time for each restoration team on each node. (3) the problem of the OD pair demands will be much more difficult than the problem of node-only demands, since the satisfied demands cannot be estimated until both nodes are restored, which makes the estimation on the impact of scheduling much more complicated. To the best of our knowledge, we might be arguably the first to consider time-dependent demands over both node-only and OD pair demands to schedule node restoration tasks by multiple teams.

MATERIALS AND METHODS

For the problem of node-only demands, we first give a network reduction procedure to shrink the original network into an equivalent but smaller one where only damaged nodes are considered. Then, we propose three integer programming models for calculating exact solutions. The model M_S^N eliminates subtours by recording the finish time of restoring a node, but it pays the price of generating many ineffective constraints. The model $M_T^{N_0}$ and M_T^N are based on time-space networks, where $M_T^{N_0}$ is a special case of M_T^N without considering the travel time between nodes in cases where the restoration times are much larger than the travel times. These two models are better than M_S^N since they implicitly handle the timing constraints in constructing a time-space network. However, they are also suffered by the curse of dimensionality, because the number of nodes and arcs makes the formulations to have many variables and constraints, especially for longer planning horizon (i.e., larger T). The model M_L^N is based on level-space networks that have much fewer variables and constraints than time-space networks, but this also requires to record the time for each node on a route. In our computational tests, this disadvantage makes this model still time consuming, unless we construct a smaller level-space network. To this end, we try to build a level-space network of reduced size that may hopefully catch good routings.

In particular, we propose a matheuristic that first solves the problem by a greedy algorithm named A_{LD} to obtain a very good feasible solution. Algorithm A_{LD} tends to give a schedule that minimizes the unsatisfied demands, which means to maximize the satisfied demands. Then, we construct a reduced level-space network that includes all the arcs in the routes by A_{LD}, as well as some more arcs obtained by another feasible solutions (e.g., feasible solutions obtained in the process of solving the full level-space network model M_L^N). We also give two different formulations: (1) an arc-based formulation $\bar{M}_{L_A}^N$ or (2) a path-based formulation $\bar{M}_{L_p}^N$. The former formulation is more compact (i.e., with smaller size) but has a bad dual convergence rate. On ther other hand, the latter formulation requires efficient enumeration of all paths in the reduced level-space network by the Depth-First-

Search algorithm, but will have a very fast dual convergence rate. Finally, we also design a simulated annealing (SA) algorithm to help converge to a better solution, based on a good feasible solution by the greedy algorithm. Note that all the models and algorithms designed for the node-only demands can be further modified for dealing with the OD pair demands. In particular, integer programming models M_S^A, $M_T^{A_0}$, M_T^A, M_L^A, $\bar{M}_{L_A}^A$, and $\bar{M}_{L_p}^A$ are directly modified from their node-only variants, which further checks the availability of both end nodes for each OD pair. As for the OD pair version of the greedy algorithm A_{LD}, it does the same steps as its node-only version, after we split the OD demands to both end nodes.

RESULT AND DISCUSSION

Our testing was performed on a personal computer with Windows 10, Intel Core i7-6770 3.40 GHz*8 Processors, and 16GB RAM. All the solution methods are implemented in C++ language, compiled by Visual C++ 2015. Gurobi 8.0.1 version is used for solving integer programs. Based on our computational experiments, we have the following observations: (1) the time-space model $M_T^{N_0}$ or M_T^N has much better performance than M_S^N and M_L^N for small to medium-scale problems, (2) $M_T^{N_0}$ can solve problem (without travel time) even on large scale of networks in very short time, (3) all the integer programming models are not suggested to solve the OD pair demand problems which are usually time-consuming, (4) our propsed matheuristic is efficient because the greedy algorithm usually gives a very good feasible solution. To further improve the solution by reduced level-space networks, the path-based formulations are much faster than their arc-based variants, and (5) our proposed SA can further improve feasible solutions, obtained by our matheuristic algorithm. The effectiveness of SA for solving the OD pair demand problems is better.

CONCLUSION

The node restoration scheduling problems for rehabilitating service networks are very important in the recovery phase of post-disaster humanitarian logistics management. Our work is the first to tackle the cases of commonly seen time-dependent node demands in practice. We have developed several mathematical programming models from different theoretical points of view. Unfortuantely, the time-dependent property we considered apprantely makes the problem too challenging, so that almost all the integer programming models are useless, except $\bar{M}_{L_p}^N$ and $\bar{M}_{L_p}^A$, the model constructed from a reduced level-space network obtained by routings calculated by the greedy algorithm A_{LD}. In short, our matheuristic that integerates a greedy algorithm into a path-based integer programming model from a reduced level-space network does help to further improve the greedy solutions.

Moreover, our SA algorithm can improve the solution by matheuristic. We hope these new findings can intrigue future researchers to further develop more effective method to deal with these challenging node restoration scheduling problems.

誌謝

時光飛逝，兩年多碩班的生活終將進入了尾聲，自大四下進入實驗室，轉瞬間卻到了要離開的時候，雖然這過程非常辛苦但也十分充實，看到自己有所成長進步而感到開心。

首先，由衷感謝指導教授王逸琳老師的教導，從 RAS 火車比賽、學術研討會、論文研究等經歷，不管是在研究上還是在報告上老師都能提出切實的建議並指正出我的缺點，老師對研究嚴謹的態度、思考的邏輯及方法，均讓我獲益良多。此外，也要感謝口試委員許瑞麟老師和李宇欣老師以及系上 Proposal 審查委員李賢得老師，願意抽空審閱論文並提出諸多寶貴的建議，讓論文得以更加完善。

另外，感謝 Ilin Lab 的大家，謝謝小秉、貞泰、采緹、偉德、富元、筱昀、BK，在我剛進 Lab 時受到學長姊們諸多的照顧，帶我開拓不少台南美食地點，特別感謝小秉在程式上指導；謝謝同屆的夥伴冠緯陪我一起經歷兩年碩班的時光，無論是課堂還是論文研究上給予我許多意見和幫助，在論文進度壓力下，可以彼此互相加油打氣；還有雪湄、昀軒、嘉豪、Meidy、Jib，在論文忙得焦頭爛額時，經常在 Lab 一起吃飯聊天、推薦我好看的影片，排解寫論文時的煩悶與壓力，有你們這些學弟妹才讓實驗室更加歡樂，也祝福你們未來研究順利。當然還要感謝我周遭的朋友們，時常關心我的近況，尤其謝謝孟穎、兆敏你們的陪伴，可以互訴苦談心和分享喜悅，亦在我情緒低落時給予鼓勵與支持。真的非常感謝的大家，讓我在碩班的生活留下美好的回憶。

最後，我要感謝我的父母及家人，因為你們一直在背後默默支持，不論面對多大的難關，都能成為我最堅強的後盾，遠在台南求學六年時間裡，感謝有你們栽培與付出，平日對我的噓寒問暖及處處體諒包容，這些日子以來你們真的辛苦了。

目錄

表目錄

圖目錄

第一章 緒論

1.1 研究背景與動機

在 ATM、電信或交通運輸等服務系統網路中，其服務皆由該網路「節點」上的基礎設施所提供。於網路節點上接受服務之需求量，通常會「依時變化」，亦即不同的節點在不同時刻可能會有不同的需求量。倘若節點受到地震、颱風等天災或人為蓄意破壞而故障，將會造成當下仰賴於節點提供服務的需求（譬如去 ATM 領錢的人、藉由基地台收送的封包量、租借 YouBike 的人等）無法被滿足而流失。台灣近年來常有影響服務中斷之事件，譬如 2016 年 8 月底 YouBike 在更新系統時遭惡意程式植入，導致 YouBike 系統大當機，停車柱無法租還車，造成全台六縣市 778 租借站、2.2 萬輛公共自行車停止使用，影響通勤人數約 35 萬人次；中華郵政在 2018 年 5 月及 8 月皆發生系統大當機，造成全台超過 3000 台 ATM 停擺；此外，國內外亦曾因颱風、強風豪雨、土石流等天災，造成交通、電力、電信等設施毀損，嚴重影響居民行動、生活、及通訊等服務需求。

為了盡量減少因大規模的節點故障而受影響的服務需求量，這些故障的節點必須儘早修復。然而修復系統所需的人力或機具等資源有限，服務系統決策者在因應危機的決策管理上，必須有效地指揮調度有限的人力與機具資源，安排最佳之修復排程，以使修復期間讓受到不便所影響之總需求量愈小愈好，此即為本論文主要探討的議題。本研究依需求特性的差別，將探討兩類節點需求：(1)發生於單一「節點」的服務需求，只要該節點沒有故障即可被滿足，譬如去郵局或 ATM 存領錢、在自動販賣機買飲料等需求；(2)來自一對「起訖組合」節點的服務需求，必須當起訖節點雙方皆無故障，方可滿足其需求，譬如 YouBike 租借、無線通訊等服務需求。本研究探討在已知依時變化之需求量及各節點之維修時間下，如何調度指派多組維修資源，以儘快修復所有故障節點，滿足愈多的單一節點或起訖組合需求量之網路節點修復排程問題。

1.2 研究目的

本研究將探討修復資源的最佳排程問題，考慮服務需求發生於「單一節點」或「起訖組合」等兩大類，前者之需求僅發生於單一節點上，因此一旦該節點被修復後，其後所產生的需求即皆可被滿足，如 ATM 網路系統；而後者之需求則來自一對起訖節點組合，必須當起訖節點皆被修復之後，其後產生的兩節點間之起訖需求才可被滿足，像是電信網路、YouBike 自行車共享系統等，以 YouBike 為例，只要任一起訖租借站還未修復完工，即無法同時滿足欲從起站借車到訖站還車的需求人數。此外，我們考慮現實生活中這些節點需求會依時而變化的特性，節點修復完後可滿足的需求量將依其修復完工時刻不同而有所不同。為使排程能達到最大服務效益的目的，本研究之目標著重在極大化所有節點修復之後因而可以滿足的需求總量。

文獻中，與本研究較有關聯的議題為「K 組旅行維修員問題」（k-Traveling Repairmen Problem，k-TRP）。該議題亦探討如何派遣多組維修員以修復所有的網路節點，但其最佳化目標在於極小化所有節點修復完工時刻之總和，可被視為本研究單一節點需求模式的特例，將各節點在任一時刻之需求設定為固定常數。相較而言，本研究考量的需求會依時變化、以及來自起訖節點組合等特點，更能符合現實的考量。

以下舉例說明本研究問題之決策困難性，圖 1.1 為依時變化的節點需求量之資訊，假設單一維修隊欲修復故障節點 A、B、C，節點所需維修時間皆為 1 單位，在維修隊移動時間可忽略情況下，修復順序該如何決定才能滿足最多需求量，通常第一直覺會考慮節點修復完工後可滿足其後時刻需求量總和較大者優先之修復順序。首先自 $t=0$ 開始，假設選擇節點 A 修復，在 $t=1$ 時可修復完成而可滿足節點 A 在 $t=1,2,3$ 之需求量總和 $15+15+5=35$；以此方式再計算 $t=0$ 時開始修復 B、C 的情況，可滿足需求量總和分別為 $5+30+15=50$、$10+5+5=20$（如圖 1.1(b)所示），因此第一個點選擇可獲得需求量總和最大 $\max\{35,50,20\}=50$ 之節點 B 來修復。再來測試 $t=1$ 時可開始修復（將於 $t=2$ 時修復完成）節點 A 或 C 的情形，計算這些節點在 $t=2,3$ 可被滿

足之需求量可得 A 點15+5=20 、C 點5+5=10 ，因而選擇最大可滿足之需求量 max{20,10}=20 的節點 A 當第二個被修復的節點；最後在 $t=2$ 時開始修復（將於 $t=3$ 時修復完成）節點 C 可得需求量 5。以此選擇機制得到修復節點順序為 B-A-C，可滿足節點需求總量為50+20+5=75。以上機制是利用貪婪法則來選取當下的排程，雖符合直覺但無法保證獲得最佳解。以此例而言，若選擇 A-B-C 順序可滿足需求量 35+45+5=85 ，將比使用貪婪法則的 B-A-C 排程之結果更好。

由上例可知，一般排程常用的貪婪法則仍有改善空間，因此如何有系統地分析問題、建立嚴謹的數學模式，並設計出具求解效率之修復排程演算法為本研究之目標。

t	1	2	3
A	15	15	5
B	5	30	15
C	10	5	5

(a)每時刻之節點需求量

t	1	2	3
A	35	20	5
B	50	45	15
C	20	10	5

(b)每時刻之節點往後累積需求量

圖 1.1 節點依時變化需求資訊

1.3 論文架構

本論文之架構如下：第二章回顧 *k*-TRP、災後緊急搶修作業等相關文獻作；本研究問題依需求型態可分為兩類問題，分別在第三章討論單一節點需求型態之網路節點修復問題，而在第四章討論起訖組合需求型態之網路節點修復問題，這兩章中將介紹各主題之數種數學模式及演算法(數學啟發式演算法、模擬退火法)；第五章為數值分析，測試本研究提出的數學模式及演算法，在不同網路大小下的求解表現；第六章總結目前本研究的成果及討論未來可研究方向和建議。

第二章 文獻回顧

網路節點修復問題之相關文獻涉及工業工程、作業研究、災後管理等領域，因本研究問題可視為一特殊的 *k*-TRP，本章首先探討 *k*-TRP 的相關文獻並與本研究問題作比較，接著更深入介紹此問題的兩種常用整數規劃模式，最後回顧災後緊急搶修作業相關文獻。

2.1 *K* 組旅行維修員問題（*k*-TRP）

本研究欲指派 k 組維修人員修復全部 n 個毀損節點，因此每組維修人員所花的時間可分成在節點 i 上的修復時間 p_i，以及在兩節點 i、j 間的移動時間 d_{ij} 兩大類，此與求解 k 個旅行維修員必須經過並修復網路上所有節點的 *k*-TRP 本質相同（但目標不同）。*k*-TRP 為單一（k=1）旅行維修員問題(Traveling Repairmen Problem，TRP)的廣義版本，通常假設在一完全圖(Complete Graph)的網路 $G=(N,E)$。其中，N 為節點集合，可視為需被服務的顧客，E 為所有無向節線的集合，已知每個節點所需維修時間 p_i 及每個無向節線 (i,j) 經過所需旅行時間 d_{ij}，目標為從一起點出發找到 k 條互不相交的路徑且每個節點恰好被經過一次，而使節點總體等待時間最小，其中等待時間為維修員從起點出發的時刻直至節點修復完工的時刻 (或者是節點開始修復的時刻)。

TRP 是以顧客為導向的路徑規劃問題，與另一經典的「旅行推銷員問題」(Traveling Salesman Problem，TSP)十分相似，兩者都在求解如何規劃人員拜訪各節點的路徑，且都是希望能越快拜訪完所有節點越好。然而，TRP 與 TSP 在本質上有兩處相異：(1)TSP 不考慮各節點的停留時間而僅考慮節點間的移動時間；(2)TSP 著重於最小化各維修員的完工時刻(makespan)加總，亦即極小化各維修員最後一個節點的拜訪時刻。若以維修人員的路線規劃問題來舉例，TSP 較像是求解能儘快讓維修人員完工休息的時刻，亦即對資源供給方的成本極小化；反之 TRP 則著重於求解極小化各節點被修好時刻的加總，亦即對顧客方等待服務的成本極小化。由於服務系統的修

復問題應該還是要以極大化對顧客的服務品質為目標，因此本研究著重的最佳化目標應與 TRP 較雷同。

文獻中與 TRP 類似設定的還有以下三類問題：(1)「最小延遲問題」(Minimum Latency Problem，MLP)、(2)「旅行送貨員問題」(Traveling Deliveryman Problem，TDP)、(3)「具容量限制及累積時間下之車輛途程問題」(Cumulative Capacitated Vehicle Routing Problem，CCVRP)，若放鬆其容量限制時，亦可將其視為 TRP。TRP 的應用層面涉及災後緊急救援物資運送、易腐敗商品的運送、人員接送、具設置時間平行機台排程問題等。Sahni and Gonzalez (1976) 已證明 TRP ($k=1$)為 NP-hard 問題，所以更複雜的 k-TRP 也會是 NP-hard。

Fakcharoenphol et al. (2003)為目前已知最早的 k-TRP 文獻，該論文並未考慮節點的維修時間，發展出一套 8.497α 近似演算法(Approximation Algorithm)，其中 α 表示為找尋 k 個節點之最小生成樹問題(k-Mimimum Spanning Tree，k-MST)之最佳可達到近似比值。而 Jothi and Raghavachari (2007) 探討的 k-TRP 則有考量維修時間，提出在固定常數 k 下，可以找到($\beta+2$)近似演算法；對於任意 k 下，則可以得到((3/2)β+(1/2))近似演算法，其中 β 表示為求解維修時間為 0 之 k-TRP 最佳可達到近似比值。此外，Jothi and Raghavachari (2007)還提出「有限延遲問題」(Bounded-Latency Problem，BLP)為 k-TRP 的互補問題，求解最少需要的維修員組數(k)，以在已知有限時間內服務完所有顧客，且證明 BLP 為 strongly NP-hard。

Luo et al. (2014) 考量維修員旅行時間有一上限值，可適用於員工有上班時數的限制。譬如倘若員工一天最多工作 8 小時，則必須規劃在旅行時間小於 8 小時內的旅行路線；或者是易腐敗食品的送貨問題，為了避免在運送時間過長情況下導致食物腐敗，而需在規定保鮮期內送至顧客手中，該論文提出以「分支-價格-切割演算法」(Branch-and-Price-and-Cut Algorithm)求解具距離限制下的 k-TRP。一般求解車輛途程問題(Vehicle Routing Problem，VRP)時，常需求解一個具資源約束之基本最短路徑問題(Elementary Shortest Path Problem with Resource-Constrainted，ESPPRC)，而 Luo et

al. (2014)的子問題和一般 ESPPRC 差別在於前者考慮每個顧客到達的累計成本，且證明其為 NP-complete，並提出多種加速求解子問題的演算法。

Dewilde et al. (2013)以災後救援為例，因地震、颱風等天災造成大量傷患，有多處村落急需醫藥的物資援助，現有一台裝載藥物的卡車到各個村落提供醫藥，已知每個村落的醫藥需求量，假設每過一單位時間即會有一個人死亡，因此醫藥的需求量將隨時間而遞減（即「依時變化」）。該論文之目標為求解最佳的卡車路線規劃，以滿足最多的依時變化之醫藥需求量。舉例來說，倘若有一村落的醫藥需求量為 50 人，卡車若在時刻 30 時抵達該村落提供藥物，則將已有 30 人因來不及等到藥物而死亡，導致可滿足醫藥需求量為 20 人。特別注意的是此問題之路線僅需經過部分節點，若等太久以至於村落傷患無人能倖存的情況下，則不需提供藥物給該村落。該研究提出一數學模式，並發展「多鄰近點之禁忌搜尋演算法」(Tabu Search Algorithm with Multiple Neighborhoods)來求解。

Baez et al. (2016) 探討設置時間具相依關係之平行機台排程問題，為分派 n 個獨立工件(job)至 m 個平行機台上進行作業，每個工件在機台上的作業時間為機台的設置時間加上工件的處理時間，然而該設置時間和該機台所處理之前一工件相關（具相依關係），目標為最小化總體完工時間。以 TRP 角度來看，可將每個機台視為維修員、工件視為一網路上的節點、機台設置時間為該維修員於兩節點間的旅行時間，而處理時間即為節點的維修時間，目標為最小化工件 makespan（亦即節點之等待時間或修復完工之時刻）的總合。因此，TRP 與平行機台排程問題具有類似的問題架構。

Angel-Bello et al. (2013)探討兩種最小延遲問題(MLP)整數規劃模式，Nucamendi-Guillen et al. (2016)提出一整數規劃模式，並發展一「反覆貪婪演算法」(Iterated Greedy Algorithm)以求解 k-TRP。Baez et al. (2016)提出兩種整數規劃模式求解具相依設置時間下之平行機台排程問題，該問題亦可視為 k-TRP 問題。以上三篇文獻皆以層空(Level-Space)為架構建立數學模式，我們將在 2.1.2 小節做更詳細的介紹。

其中 Angel-Bello et al. (2013)探討考慮維修時間之 MLP，此篇等待時間的定義為

節點開始修復的時刻，可藉由將節點 i 的維修時間轉嫁至無向節線(i, j)的旅行時間上，而重新定義節線(i, j)旅行時間$c_{ij} := p_i + d_{ij}$；同樣地，若等待時間定義為節點修復完工的時刻，則將節點 j 的維修時間轉嫁至無向節線(i, j)的旅行時間上，定義節線(i, j)旅行時間$c_{ij} := d_{ij} + p_j$，可將問題轉變成不考慮維修時間下之 MLP，由於各節點維修時間不一樣，旅行時間矩陣 C 為非對稱矩陣。假設網路圖上有 n 個節點，MLP 之可行路徑可表達 P={[0],[1],[2],…[n]}，其中[i]表示路徑 P 第 i 位置的節點，[0]為出發起點，目標值為所有路徑上節點等待時間之加總可計算如式(2.1)。

$$W(P) = c_{[0][1]} + (c_{[0][1]} + c_{[1][2]}) + \cdots + (c_{[0][1]} + c_{[1][2]} + \cdots + c_{[n-1][n]}) = \sum_{i=0}^{n-1} (n-i) c_{[i][i+1]} \tag{2.1}$$

此外，Nucamendi-Guillen et al. (2016)把兩節點 i, j 上各一半的維修時間加在節線(i, j)的旅行時間上，重新定義節線旅行時間為$d_{ij} + 0.5(p_i + p_j)$，與上述轉換方式不同。其旅行時間矩陣 C 為對稱矩陣，在單一維修員的情況下證明此種方式仍然可以求出相同的最佳解，且最佳值只會差一固定常數，該常數為整體節點維修時間加總的一半，此篇經由數值分析結果發現對稱矩陣 C 比非對稱矩陣 C 之數學模式有更好的求解效率，然而此方法在多組維修員下並不適用。

表 2.1 TRP 相關文獻與本研究之比較

文獻	多組維修員	維修時間	需求 依時變化
Fakcharoenphol et al. (2003)	V		
Jothi and Raghavachari (2007)	V	V	
Luo et al. (2014)	V		
Dewilde et al. (2013)			V
Angel-Bello et al. (2013)		V	
Nucamendi-Guillen et al. (2016)	V	V	
Baez et al. (2016)	V	V	
本研究	V	V	V

我們將 TRP 相關文獻與本研究探討問題作比較整理如表 2.1，目前少有文獻探討需求依時變化之問題，而較早之前 TRP 或 k-TRP 文獻探討問題並不考慮節點的維修

時間，節點僅需被拜訪經過即可，節點的需求量常視為單一需求。然而本研究則考慮多組維修員的派遣、節點有維修時間，且最重要的與過往文獻的差別在於考慮需求依時變化。

根據 Angel-Bello et al. (2017)比較過往 *k*-TRP 常用之五種數學模式，其中前三種數學模式 Model I, II, III 以傳統多旅行推銷員問題(*k*-TSP)的建模方式，後兩種 Model IV, V 則是以層空(Level-Space)為架構建模。在數值分析結果來說，分別以路徑規劃問題及排程問題的例子資料作測試下，後者數學模式皆優於前者；前三種模式中以 Model III 表現最佳，但即便如此 Model III 也只能求解 15 個節點的路徑規劃問題及 12 個節點的排程問題例子，且對於網路節線為非對稱矩陣 C 的例子表現比對稱矩陣 C 來的差；而後兩種模式在小例子中求解時間相似，但隨著節點數的增加，可發現 Model V 表現較佳，求解節點數可達 30 個。以下我們將針對多旅行推銷員問題(*k*-TSP)及層空網路此兩種數學建模方式做更詳細的介紹。

表 2.2 *K* 組旅行推銷員/ *K* 組旅行維修員/本研究問題之比較

比較	***K* 組旅行推銷員 (*k*-TSP)**	***K* 組旅行維修員 (*k*-TRP)**	**本研究**
目的	派遣 K 組人員經過 n 節點		
節點	節點不需停留	節點需停留	節點需停留
需求	一次性/固定	一次性/固定	即時性/依時變動
目標	最小完工時刻 (makespan)	最小節點總等待時間 (waiting time)	最大節點可被滿足總需求量

2.1.1 以傳統多旅行推銷員(*k*-TSP)問題為基礎建構之數學模式

由於本研究與 *k*-TSP 或 *k*-TRP 相似，故分析三者問題的相異之處如表 2.2 所示，相同之處在於都需派遣 K 組人員經過 n 節點。然而 K 組旅行推銷員的節點是不需停留，也就是節點不需要一個處理時間(processing time)，且需求為一次性且不會依時變動，亦即不管何時經過節點能滿足的需求量一樣，因此其目標為最小完工時刻，是以

提供服務者的角度，希望能花越短的時間服務所有顧客。反觀 K 組旅行維修員則考慮每個節點皆有其處理時間，是站在接受服務的顧客角度，因此目標為最小節點等待時間。本研究與 k-TRP 相似，最大不同處在於考慮需求依時變動且具即時性，若當下需求未被滿足則將流失，目標為最大節點可被滿足總需求量。

由於 k-TRP 節點需停留，藉由將維修時間轉嫁至節線的旅行時間上，而可忽略節點維修時間，將問題轉為 k-TSP，只差目標式不一樣，因此可用 Bektas (2006) k-TSP 之數學模式為基礎建模。首先我們增設一虛擬起點 o(若有規定的出發節點 i 則 o=i)及虛擬訖點 d，使完全圖 $G=(N,E)$ 轉換為有向網路圖 $G'=(N,A)$，其中定義 $A:=\{(i,j): i\in N\cup\{o\}; j\in N\cup\{d\}\}$ 為有向節線的集合，每條節線 (i,j) 對應的旅行時間為 $c_{ij}:=d_{ij}+p_j$，其中 $d_{id}=0$、$p_d=0$。將每組維修員 k 修復節點的行走路徑表示成 $P_k=\{o,[1]_k,[2]_k,...,[n_k]_k,d\}$，其中 $[m]_k$ 表示維修員 k 第 m 次移動節點位置，n_k 表示維修員 k 總修復節點的個數，所以可將 k-TRP 轉換成找尋 K 條從虛擬起點至虛擬訖點的 o-d 路徑。以下定義符號如下：

集合

N	所有節點構成之集合(不包含虛擬起訖點 o、d)
A	所有節線構成之集合
A_i^+	所有連出節點 i 的節線構成之集合(outgoning arc)，$A_i^+\subset A$
A_i^-	所有連進節點 i 的節線構成之集合(ingoning arc)，$A_i^-\subset A$

參數

K	可派遣維修人員的組數
T	修復完工時刻上限
$c_{i,j}$	節線 (i,j) 的旅行時間

決策變數

$x_{i,j} \in \{0,1\}$　　若節線(i, j)有被經過則為 1;反之為 0

$w_i \geq 0$　　節點 i 修復完工的時刻

目標式(2.2)為最小化所有節點的等待時間。

$$Minimize \quad \sum_{i \in N} w_i \tag{2.2}$$

限制式(2.3)表示最多只派遣 K 組維修人員從起點 o 出發。

$$\sum_{j \in N} x_{o,j} \leq K \tag{2.3}$$

限制式(2.4)及(2.5)限制路徑的連通性，需確保每個節點除起訖點外皆須符合流量守恆，流進的節線數需等於流出的節線數，且每個節點只會有一組維修人員經過。

$$\sum_{(j,i) \in A_i^-} x_{j,i} = 1 \quad \forall i \in N \tag{2.4}$$

$$\sum_{(i,j) \in A_i^+} x_{i,j} = 1 \quad \forall i \in N \tag{2.5}$$

式(2.6)為排除子迴圈的限制式(Subtour Elimination Constraints)，限制當 $x_{i,j} = 1$ 時則 $w_j \geq w_i + c_{i,j}$。舉例來說，若有子迴圈的發生如圖 2.1 所示，則限制 $w_2 \geq w_1 + c_{1,2}$、$w_3 \geq w_2 + c_{2,3}$ 及 $w_1 \geq w_3 + c_{3,1}$，經由以上 3 個限制式可推論 $0 \geq c_{1,3} + c_{2,3} + c_{3,1}$ 導致矛盾，故可有效排除維修員的修復路徑為子迴圈。

$$w_i - w_j + (T + c_{i,j}) x_{i,j} + (T - c_{j,i}) x_{j,i} \leq T \quad \forall (i, j) \in A; i, j \in N \tag{2.6}$$

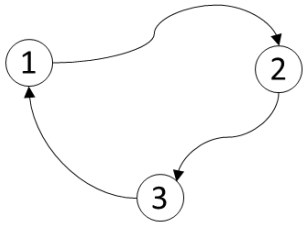

圖 2.1 子迴圈示意圖

限制式(2.7)為限制每組維修人員第一個經過的節點的等待時間。

$$w_i \geq c_{oi} \quad \forall i \in N \tag{2.7}$$

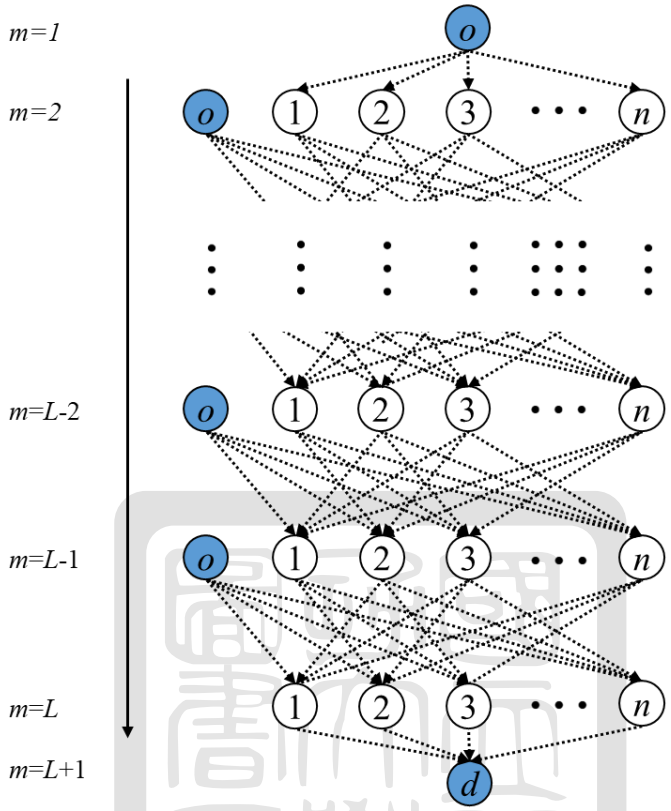

圖 2.2 k-TRP 問題之層空網路圖

2.1.2 以層空為基礎建構之數學模式

參考 Angel-Bello et al. (2017) 之 Model V 模式，層空網路圖(Level-Space network) $G_L = (\tilde{V}, \tilde{A})$ 如圖 2.2 所示，橫向表示空間狀態，縱向表示為階層狀態。為了和網路 $G' = (N, A)$ 的節點與節線作區別，我們稱多階層網路圖上的節點與節線為「階層節點」與「階層節線」，$\tilde{V}$ 表示階層網路圖上所有階層節點的集合，$\tilde{A}$ 表示階層網路圖上所有階層節線集合。此網路圖 G_L 需設置一參數 L 表示每組維修員最多修復節點數上限，使用者可自行決定 L，其大小需介於 $\left[\lceil |N|/K \rceil, |N|\right]$ 之整數才會有可行解。因最差的

情況是由一組維修員修完所有的節點，故 L 的上界為 $|N|$；L 最小的情況是平均所有節點使每組維修員修復一樣節點數，然而 $|N|/K$ 可能不能整除，故某些維修員會多修復一個節點，因此層空網路圖是由 L 及節點數 $|N|$ 決定網路的大小。

網路圖中第 m 階層的節點 i 以 $v_{i,m} \in \tilde{V}$ 表示之，而 $(v_{i,m}, v_{j,m+1}) \in \tilde{A}$ 代表由階層 m 的節點 i 連至階層 $m+1$ 的節點 j 的階層節線，因此可用 $(v_{o,L-n_k}, v_{[1]_k,L-n_k+1}) \rightarrow (v_{[1]_k,L-n_k+1}, v_{[2]_k,L-n_k}) \rightarrow \cdots \rightarrow (v_{[n_k-1]_k,L-1}, v_{[n_k]_k,L}) \rightarrow (v_{[n_k]_k,L}, v_{d,L+1})$ 之階層節線連結的路徑來代表每個維修員 k 之修復路徑 $P_k = \{o,[1]_k,[2]_k,...,[n_k]_k,d\}$。為找尋 K 條不相交之 o-d 路徑使節點等待時間最小，我們為層空網路圖中的每條階層節線設定一個二元決策變數，用以決策 K 組維修員的修復路徑，且不會有子迴圈發生。以下定義符號如下：

集合

N	所有節點構成之集合，$i, j \in N$
$\tilde{V}$	所有階層節點構成之集合，$v_{i,m}, v_{j,m+1} \in \tilde{V}$
$\tilde{A}$	所有階層節線構成之集合，$(v_{i,m}, v_{j,m+1}) \in \tilde{A}$
$\tilde{A}^+_{v_{i,m}}$	所有連出階層節點 $v_{i,m}$ 的階層節線構成之集合，$\tilde{A}^+_{v_{i,m}} \subset \tilde{A}$
$\tilde{A}^-_{v_{i,m}}$	所有連進階層節點 $v_{i,m}$ 的階層節線構成之集合，$\tilde{A}^-_{v_{i,m}} \subset \tilde{A}$

參數

K	可派遣維修人員的組數
L	維修員最多修復節點數上限
T	修復完工時刻上限
c_{ij}	節線 (i, j) 的旅行時間

決策變數

$y_{(v_{i,m},v_{j,m+1})} \in \{0,1\}$　若階層節線$(v_{i,m},v_{j,m+1})$有被經過則為 1;反之為 0

$w_i \geq 0$　節點 i 修復完工的時刻，$i \in N$

目標式如式(2.8)所式為最小化節點等待時間。

$$Minimize \quad \sum_{m=0}^{L-1} \sum_{(v_{i,m},v_{j,m+1})\in \tilde{A}} (L-m) c_{i,j} y_{(v_{i,m},v_{j,m+1})} \tag{2.8}$$

限制式(2.9)表示最多只會派遣 K 組維修人員去修復節點，因此從虛擬起點 o 連出的階層節線至多 K 條被經過。

$$\sum_{m=0}^{L-1} \sum_{(v_{o,m},v_{j,m+1})\in \tilde{A}^{+}_{v_{o,m}}} y_{(v_{o,m},v_{j,m+1})} \leq K \tag{2.9}$$

限制式(2.10)限制每個節點必須被某組維修員修復一次。

$$\sum_{m=1}^{L} \sum_{(v_{j,m-1},v_{i,m})\in \tilde{A}^{-}_{v_{i,m}}} y_{(v_{j,m-1},v_{i,m})} = 1 \quad \forall i \in N \tag{2.10}$$

限制式(2.11)為確保維修人員路徑連通性的限制，每個階層節點除虛擬起訖點外要滿足進流量等於出流量。

$$\sum_{(v_{i,m},v_{j,m+1})\in \tilde{A}^{+}_{v_{i,m}}} y_{(v_{i,m},v_{j,m+1})} - \sum_{(v_{j,m-1},v_{i,m})\in \tilde{A}^{-}_{v_{i,m}}} y_{(v_{j,m-1},v_{i,m})} = 0 \quad \forall v_{i,m} \in \tilde{V}; i \in N \tag{2.11}$$

2.2 災後緊急搶修作業

災害物流管理工作可分為災前及災後作業兩部分：

(1)災前作業：災害發生前的準備工作，以預防災後可能發生之影響，透過各種因應措施，防止災害之發生或降低災難所帶來的衝擊，如: 緊急避難處選址和支援物資預先配置。

(2)災後作業：在災難發生後，搶救攸關人民生命與財產之緊急救援，通常災後導致大規模需求發生且遠大於能供應的資源，因此，如何短時間的動員現有的資源有效進行緊急救援工作以及讓社會與經濟恢復正常運作極為重要，如:救援物資緊急運送、傷患緊急醫療救護、道路緊急搶修、基礎設施重建。

本研究探討的問題係屬災後作業，為大規模服務系統網路的節點(設施)故障之修復問題，因節點故障造成建立在節點或起訖組合上的依時需求無法被滿足，因此在現有人力下希望盡快能在短時間內修復節點以讓服務系統恢復正常，為具即時且緊急性的修復排程規劃，與大多災後緊急搶修作業性質雷同，通常假設搶修時程下的每一時間點，所能使用的修復資源如人力、機具為有限的，而以本研究問題來說則是每一個時間點所能使用的人力資源有限且固定。以下將針對災後緊急搶修作業相關文獻探討介紹：

Matisziw et al. (2010) 以電信設施維修為例，將路由器視為節點、連接路由器的骨幹線路視為節線，假設網路節點即節線皆可能損壞且皆有一已知修復成本，在每一期皆有以固定節點和節線的修復預算上限，來限制每一期可以修復的節線數與節點數，且認為在此緊急修復作業中最重要的目標是確保特定節點間的路徑之連通性。該論文提出一個多準則整數規劃模式，在不違反每期修復成本預算下，決定每一期要修復的節線與節點，以最小化總修復成本下滿足最多組起訖點間的流量。

Averbakha (2012) 討論道路搶修問題，目標極小化節線打通時刻，假設已知各個工作隊修復速度，並允許各個工作隊的能力不同，並假設工作隊合作的能力是具相加性的，亦即若工作隊 A 可每小時修復 0.6 公里，工作隊 B 可每小時修復 0.4 公里，則兩工作隊合作可每小時修復 1 公里。此研究不考慮工作隊的移動時間，並以動態規劃來求解此問題。

Xu et al. (2007)考慮地震過後電力系統的修復問題，提出一整數隨機規劃模式，根據隨機發生不同強度地震，造成設施損壞程度有所不同，假設設施的修復時間為已知機率分佈的隨機變數，在有限人力資源下決定電力設施的檢查、災害的評估以及修

復的排程，以極小化每個使用者無法使用電的平均時間。由於其數學模式的變數具隨機性，可將其可能性變成各種情境而必須求解大量的多階層隨機規劃問題，致使模式求解耗時過久，故此篇使用基因演算法(Genetic Algorithm)來求解問題。

我們比較上述文獻與本研究之後，可發現 Matisziw et al. (2010)著重在設施間的連結與否，與本研究起訖組合需求型態類似，以求能盡量滿足系統的需求；該論文與本研究不同之處，在於其還牽涉節線修復，且以每期預算成本來限制每期可修受損設施數量，反之本研究則以現有可調度人力資源限來制每個時刻至多 K 組人員進行修復。Averbakha (2012) 以修復節線為主，並盡量使節點能與供給點相連以使節點需求能夠被滿足，而本研究則關注於起訖節點之兩節點是否相通，由於本研究考慮完全圖網路，故只需起訖節點皆已修復即可相通。Xu et al. (2007)亦處理節點修復的排程，但以最小化總等待時間為目標，且考慮隨機的損壞與修復時間，這些都與本研究不同。

2.3 小結

本章節回顧過往 k-TRP 相關文獻，在早期某些文獻中不考慮具維修時間之旅行維修員問題，且大部分文獻都將節點需求視為固定常數，然而本篇著重於考量需求依時變動之情況。在數學模式的文獻探討中多以傳統旅行推銷員及以層空網路兩種來建構模式，而後者求解的表現比前者來得好。接著介紹災後緊急作業討論其問題的形式，並與本研究考慮之情境作比較，然而大多文獻皆未考量需求依時變化之情境。

第三章 考慮依時變化的節點需求量之網路節點修復排程

本章首先介紹節點需求型態問題與其假設，為求得理論精確解，我們提出三種整數規劃模式: (1)含排除子迴圈限制式之模式 M_S^N ；(2)以時空網路為基礎建構之模式，再依其旅行時間可否被忽略而再細分成「特例模式」及「一般模式」兩類: $M_T^{N_0}$ 及 M_T^N；(3) 以層空網路為基礎建構之模式 M_L^N。其中(1)(3)是以 *k*-TRP 之模式為基礎建構。由於這些整數規劃模式無法有效求解具較大規模的網路，因此再設計更有效率的數學啟發式演算法（Matheuristics），先以貪婪演算法 A_{LD} 來產生限縮層空網路，再分別以節線或路徑為變數建構該網路所對應之兩種數學模式 $\bar{M}_{L_A}^N$ 、$\bar{M}_{Lp}^N$，最後再設計模擬退火演算法。本章提出之數學模式以符號 M_α^β 整理於圖 3.1，其上標 $\beta=N$ 表示節點需求型態問題，$\beta=N_0$ 表示旅行時間可被忽略下之特例問題，而下標 α 則代表各模式建模的架構（共有 $\alpha\in\{S,T,L,L_A,L_P\}$ 等 5 種）；限縮模式網路下之數學模式則以 $\bar{M}_\alpha^\beta$ 表示之。

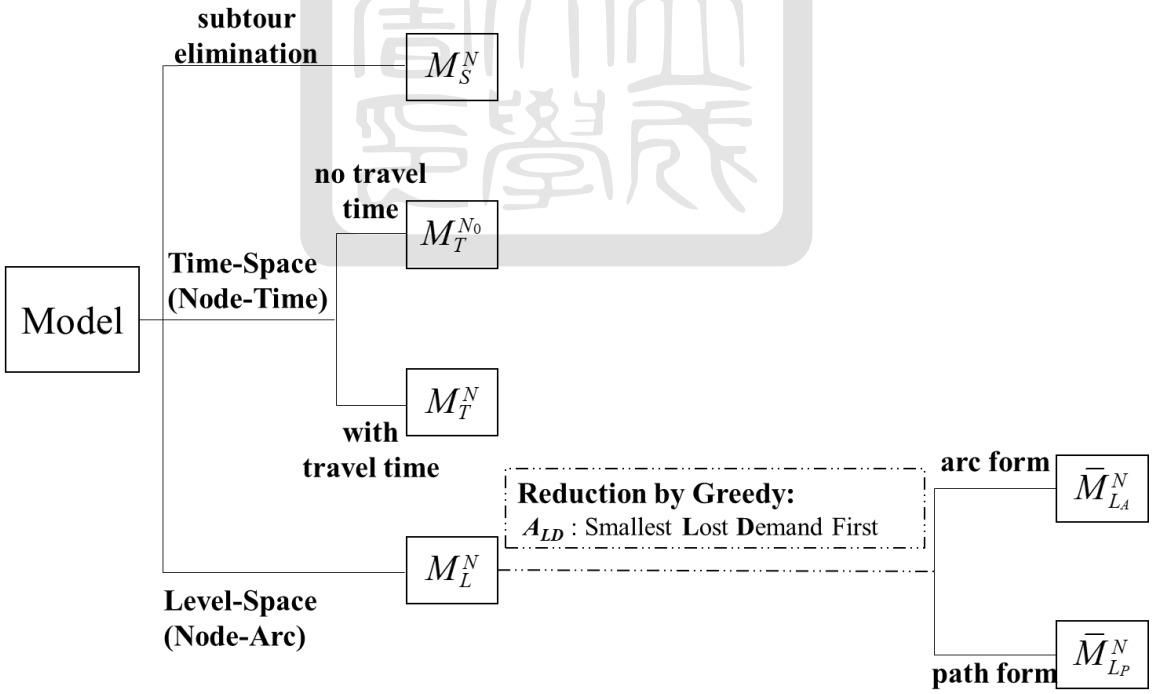

圖 3.1 節點需求型態問題之數學模式示意圖

3.1 問題描述與假設

本章主要探討節點需求型態的問題，假設有一完全圖網路$G^N=(N,E)$，N 為所有節點的集合，E 為所有無向節線的集合，d_{ij} 為每條節線(i,j)的旅行時間。若部分節點 $N_r \subseteq N$ 發生故障，決策單位需決定何時派哪組維修隊去修復哪個故障節點。假設已知可派遣維修隊數 K、規劃時間長度 T、每個故障節點 i 的修復時間 p_i 以及在每個時刻 t 需求量 $D_{i,t}$。若在時刻 t 完成修復節點 i，則在此節點可滿足從時刻 t 至時刻 T 的需求量，將其需求量加總定義為（往後）累積需求量 $AD_{i,t}$ (Accumulated Demands)，以式子表示為 $AD_{i,t} := D_{i,t} + D_{i,t+1} + \cdots + D_{i,T}$。舉例來說，若某組維修隊在時刻 t' 時抵達節點 i 並開始修復，在時刻 $t'+p_i$ 時即修復完工並滿足其累積需求量 $AD_{i,t'+p_i}$，特別注意的是累積需求量會隨著時刻增加而單調遞減，亦即對任一故障節點愈早修復節點能滿足的需求量愈多。本研究之目標為派遣 K 組維修隊在規劃時程 T 內修復所有故障節點使可滿足節點累積需求量加總最大化。

若將本研究每個節點每個時刻之需求量設定為 1，在一固定規劃時程 T 內，欲最大化每個節點可滿足的需求量，等同於最小化該節點損失之需求量。在上述設定下，節點損失之需求量等於其等待時間，所以 k-TRP 亦可被視為本研究之特例。

為了適當地簡化數學模式以及模式的使用限制，本研究有下列基本假設：

1. 假設經過系統偵測或人員檢查，可知所有發生故障的地點及損壞程度，且可估算因發生故障所影響往後每個時間區段節點需求量，以及掌握可派遣維修隊數、修復能力等，使決策單位可以在已知這些資訊之下，作出以整體為考量的決策。
2. 假設每組維修人員之工作能力與效率完全相同且完美（即 identical machine），且一旦開始修復某節點，則一定會不中斷地完成該修復任務（即 non-preemptive）。
3. 假設所有的節點開始時皆毀損，且皆可在修復時限 T 內被完成修復。
4. 假設故障節點間無特定的修復先後順序，即各節點修復優先權視為平等。
5. 不考慮多組維修人員的合作，也就是每個故障節點只會被一組維修隊所修復。

6. 毀損節點被修復完成後，在規劃時程內不會再次發生故障。

3.2 網路縮減

本研究旨在修復所有故障節點 N_f 以滿足最多的節點需求量，由於未發生故障之節點皆可滿足所有時刻下的需求量，其值為一已知常數，因此我們可進行「網路縮減」，直接忽略未發生故障的節點及與其相連之節線，如圖 3.2 網路縮減至如圖 3.3 網路，目標著重在滿足故障節點的需求量即可。

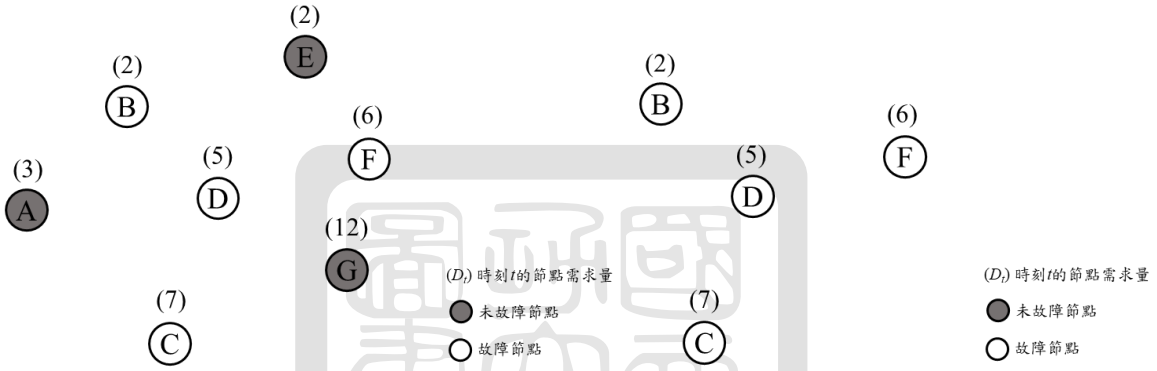

圖 3.2 縮減前之網路圖 G^N　　圖 3.3 縮減後之網路圖 G^N

3.3 含排除子迴圈限制式之數學模式 M_S^N

此模式藉由修改在 2.1.1 小節的 k-TRP 之數學模式而得，保留 2.1.1 小節的限制式(2.3)~(2.7)為限制 K 條維修隊的修復路徑及節點的等待時間，並修改目標式及新增新的變數和限制式。以下新增新的符號定義:

參數

$AD_{i,t}$　　節點 i 在時刻 t 至 T 的累積需求量，$i \in N$

決策變數

$z_{i,t} \in \{0,1\}$　　若節點 i 在時刻 t 修復完工則為 1;反之為 0

由於本研究節點需求型態之問題考量需求依時而變動，因此將目標式改為最大化節點滿足的累積需求量如式(3.1)所示。

$$Maximize \quad \sum_{i \in N_r} \sum_{t=p_i}^{T} AD_{i,t} z_{i,t} \tag{3.1}$$

新增限制式(3.2)及限制式(3.3)表示變數 w_i 和變數 $z_{i,t}$ 的關係式，將各故障節點修復完工的時刻變數轉換為二元變數表示，限制當 $t = w_i$ 時，則 $z_{i,t} = 1$；當 $t \neq w_i$ 時，則 $z_{i,t} = 0$。

$$\sum_{t=p_i}^{T} t z_{i,t} = w_i \qquad \forall i \in N_r \tag{3.2}$$

$$\sum_{t=p_i}^{T} z_{i,t} = 1 \quad \forall i \in N_r \tag{3.3}$$

在 *k*-TRP 文獻已提及此含排除子迴圈限制式的模式之求解效率不佳，在小的網路規模下進行求解即耗時甚久，此處還需將節點的修復完工時間轉換為二元變數 $z_{i,t}$ 表示，才得以寫出目標式。因此本模式必須產生許多二元變數 $z_{i,t}$，導致求解效率不佳，故不建議採用此數學模式進行求解，後續第五章數值分析會再驗證上述說法。

3.4 以時空網路為基礎建構之數學模式

本節提出以時空網路為基礎建構模式，在 3.4.1 小節介紹旅行時間可被忽略下之特例模式，而 3.4.2 小節將介紹一般模式。

3.4.1. 特例模式 $M_T^{N_0}$

本小節數學模式適用於服務系統節點間的旅行時間遠小於節點的修復時間的情

境，因此可忽略旅行時間，將之視為 0。時空網路圖 $\hat{G}^0=(\hat{V})$ 如圖 3.7 所示，其中 $\hat{V}$ 表示網路圖上所有時空節點的集合。為了和原網路 $G^N=(N,E)$ 的節點區別，在時空網路圖上的節點我們稱「時空節點」，以 $T+1$ 個水平層（即 T 規劃時間週期）和 n 個垂直層(即 n 個故障節點)來建構，其中每個水平層 t=1,2,…,T 都包含 n 個節點，而每個垂直層 i=1,2,…,n 也都包含 t=0,1,2,…,T 個單位時刻，以 $v_{i,t}$ 表示在時空網路圖的每個時空節點。若節點 i 在時刻 t 開始被修復，則路徑將經過 $v_{i,t}$ 這個時空節點。

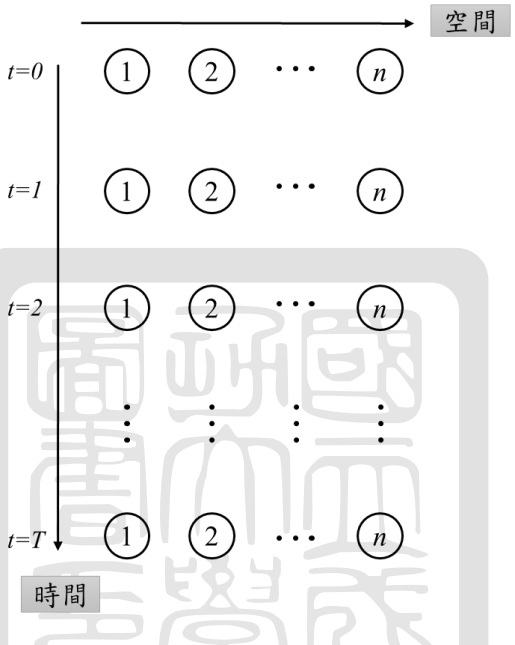

圖 3.4 時空網路圖示意圖

以下介紹符號定義：

集合

N_r	所有故障節點構成之集合
$\hat{V}$	所有時空節點構成之集合，$v_{i,t}\in\hat{V}$

參數

K	可派遣維修隊的組數
T	修復完工時刻上限

p_i　　故障節點 i 所需的修復時間，$i \in N_r$

$AD_{i,t}$　　節點 i 在時刻 t 至 T 的累積需求量，$i \in N_r$

決策變數

$x_{v_{i,t}} \in \{0,1\}$　　若節點 i 在時刻 t 開始修復(亦即 $v_{i,t}$ 被經過)則為 1;反之為 0

目標式如式(3.4)所示，旨在於修復完工上限時刻 T 內最大化可滿足的故障節點累積需求量，由於故障節點一旦被修復完成後，該節點當下及往後每單位時刻的需求皆可被滿足，其加總即為累積需求量。

$$Maximize \quad \sum_{i \in N_r} \sum_{t=0}^{T-p_i} AD_{i,t+p_i} x_{v_{i,t}} \tag{3.4}$$

限制式(3.5)說明每個故障節點只會被修復一次，且為了在時刻 T 以前完成修復所有的故障節點，維修隊必須在 $T - p_i$ 時刻之前開始修復故障節點 i。

$$\sum_{t=0}^{T-p_i} x_{v_{i,t}} = 1 \quad \forall i \in N_r \tag{3.5}$$

限制式(3.6)代表任何時刻 t 皆不能同時有超過 K 組維修隊正在修復的資源限制。

$$\sum_{i \in N_r} \sum_{t'=\max\{t-p_i+1,0\}}^{t} x_{v_{i,t}} \le K \quad \forall t = 0,1,...,T \tag{3.6}$$

3.4.2. 一般模式 M_T^N

一般模式可適用所有可能的節點間旅行時間值，首先介紹時空網路圖 $\hat{G} = (\hat{V}, \hat{A})$，接續上小節 3.4.1 時空網路圖的時空節點 $\hat{V}$，再來定義「時空節線」$\hat{A}$ 集合的組成，為了模式的建構方便，我們增設虛擬起訖時空節點 $v_{o,0}$ 及 $v_{d,T}$，共有三類時空節線：

(1) $(v_{i,t}, v_{j,t'})$,　$i, j \in N_r;\ i \ne j;\ t' = t + c_{ij};\ t = c_{oi}, c_{oi}+1,...,T;\ t' = c_{oj}, c_{oj}+1,...,T$;

(2) $(v_{o,0}, v_{i,t}),\quad i \in N_r; t = c_{oi}$;

(3) $(v_{i,t}, v_{d,T}),\quad i \in N_r;\ t = c_{oi}, c_{oi}+1, ..., T$;

其中(1)為維修隊從節點 i 至節點 j 的可行路徑如圖 3.5 藍色虛線所示，定義時空節線的旅行時間為 $c_{ij} := d_{ij} + p_j$，本質上可拆成兩條時空節線 $(v_{i,t}, v_{j,t+d_{ij}})$、$(v_{j,t+d_{ij}}, v_{j,t'})$ 如圖 3.5 的黑色虛線所示。前者表示維修隊在時刻 t 從節點 i 出發並在時刻 $t+d_{ij}$ 抵達節點 j，後者為維修隊在時刻 $t+d_{ij}$ 開始修復節點 j 並在時刻 $t+d_{ij}+p_j$ 修復完工，而可滿足需求量 $AD_{j,t+d_{ij}+p_j}$。為簡化問題將首節點(head node)維修時間計入節點間的旅行時間，因此將兩條結合成一條時空節線來看。

(2)為維修隊從虛擬起點 o(若有規定之起點 i 則 i=o)在時刻 0 出發至第一個修復節點 i 的可行路徑。其中時空節線的旅行時間為 $c_{oi} := d_{oi} + p_i$，亦即在時刻 c_{oi} 時節點 i 修復完工而可滿足需求量 $AD_{i,c_{oi}}$。

(3)則表示除虛擬起訖時空節點外的任一時空節點都與虛擬時空訖點 $v_{d,T}$ 相連，時空節線的旅行時間為 0。值得注意的是(1)(2)時空節線的旅行時間包含修復時間，故可對應一個因修復完工時而得的節點累積需求量，我們將所有包含故障節點 i 修復時間的時空節線集合定義為 RA_i。

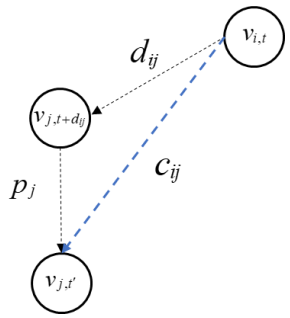

圖 3.5 時空節線示意圖

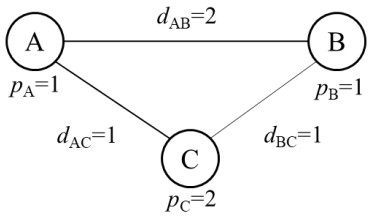

圖 3.6 待修復之網路圖

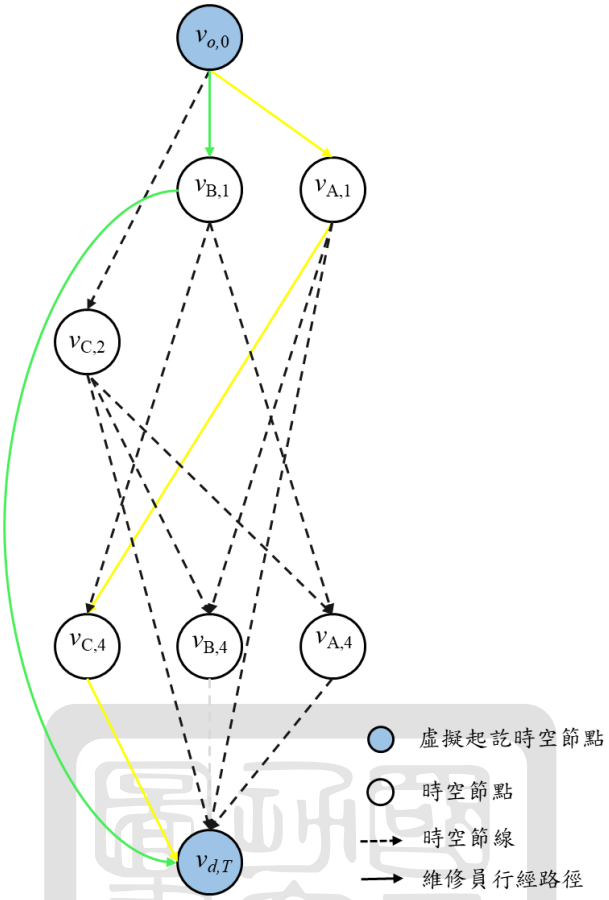

圖 3.7 時空網路圖以節線建構之示意圖

綜上所述，每條時空節線都是一個決策變數，用以決策 K 組維修隊的修復路徑。圖 3.6 的網路圖例說明了如何以節線建構時空網路圖(圖 3.7)來表示維修隊的修復路徑順序，假設 K=2、T=4，若維修隊 1 的修復順序為 A→C 及維修隊 2 的修復順序為 B，則可分別以 $(v_{o,0}, v_{A,1}) \rightarrow (v_{A,1}, v_{C,4}) \rightarrow (v_{C,4}, v_{d,T})$ 及 $(v_{o,0}, v_{B,1}) \rightarrow (v_{B,1}, v_{d,T})$ 來表達。此外，因每個故障節點只會被修復一次，故需限制連進(出)各節點的所有修復時空節線中，只能有一條被經過，舉例來說 $RA_A = \{(v_{o,0}, v_{A,1}), (v_{B,1}, v_{A,4}), (v_{C,2}, v_{A,4})\}$ 只會有一條被選到，因此 K 組維修隊的可行修復排程路徑即是在時空網路圖中找尋符合限制下之 K 條 $v_{o,0} - v_{d,T}$ 路徑。以下說明模式符號定義：

集合

N_r	所有故障節點構成之集合，$i, j \in N_r$
$\hat{V}$	所有時空節點構成之集合(不含$v_{o,0}$及$v_{d,T}$)，$v_{i,t}, v_{j,t'} \in \hat{V}$
$\hat{A}$	所有時空節線構成之集合，$(v_{i,t}, v_{j,t'}) \in \hat{A}$
$\hat{A}^+_{v_{i,t}}$	所有連出時空節點$v_{i,t}$的時空節線構成之集合，$\hat{A}^+_{v_{i,t}} \subset \hat{A}$
$\hat{A}^-_{v_{i,t}}$	所有連進時空節點$v_{i,t}$的時空節線構成之集合，$\hat{A}^-_{v_{i,t}} \subset \hat{A}$
$R\hat{A}_i$	所有節點i的修復時空節線之集合，$R\hat{A}_i \subset \hat{A}$

參數

K	可派遣維修隊的組數
T	修復完工時刻上限
$AD_{i,t}$	節點i在時刻t的累積需求量，$i \in N$

決策變數

$x_{(v_{i,t}, v_{j,t'})} \in \{0,1\}$	若時空節線$(v_{i,t}, v_{j,t'})$有被經過則為1;反之為0

目標式如式(3.7)表示最大化所有故障節點可滿足的累積需求量，若修復時空節線有被經過，則將其對應的累積需求量計入目標式中；反之，則不予計入。

$$Maximize \quad \sum_{j \in N_r} \sum_{(v_{i,t}, v_{j,t'}) \in R\hat{A}_j} AD_{j,t'} x_{(v_{i,t}, v_{j,t'})} \tag{3.7}$$

限制式(3.8)為限制每個故障節點i只會被修復一次，因此任何故障節點i的所有修復時空節線中只會選一條經過。

$$\sum_{(v_{j,t'}, v_{i,t}) \in R\hat{A}_i} x_{(v_{j,t'}, v_{i,t})} = 1 \quad \forall i \in N_r \tag{3.8}$$

限制式(3.9)限制路徑的流通性，需確保每個時空節點除虛擬時空起訖點皆須符合流量守恆，流進的節線數需等於流出的節線數。

$$\sum_{(v_{i,t},v_{j,t'})\in\hat{A}^{+}_{v_{i,t}}} x_{(v_{i,t},v_{j,t'})} - \sum_{(v_{j,t'},v_{i,t})\in\hat{A}^{-}_{v_{i,t}}} x_{(v_{j,t'},v_{i,t})} = 0 \quad \forall v_{i,t} \in \hat{V} \setminus \{v_{o,0}, v_{d,T}\} \tag{3.9}$$

限制式(3.10)表示最多只派遣K組維修隊，故虛擬時空起點流出的時空節線至多K條。

$$\sum_{(v_{o,0},v_{i,t})\in\hat{A}^{+}_{v_{o,0}}} x_{(v_{o,0},v_{i,t})} \le K \tag{3.10}$$

在以時空網路為基礎建構之整數規劃模式中，不管是以時空節點為變數的特例模式，還是以時空節線為變數的一般模式，其變數本身已包含時間的戳記，因此不需再以額外的變數來記錄每節點之修復完工時刻。然而時空網路的大小受整體修復完工的時間上限T值之影響甚大，T值會根據修復網路的規模大小以及派遣的維修隊數不同而有所變化。若T值設定太大則使模式變數及限制式的個數大幅增加，使模式求解時間變長，尤其是一般模式來得更為明顯；反之，若T值設定太小則可能導致數學模式無可行解，因此決定合適大小的T值是改善模式求解效率的重要關鍵。

3.5 以層空網路為基礎建構之數學模式M_L^N

在3.4節的整數規劃模式受T值影響甚大，然而T值會隨著故障節點數變大或維修隊數變小而驟增，進而影響模式求解的效率。因層空網路大小不受T值影響且其節點、節線個數較時空網路小，本節提出以層空網路為建構基礎之整數規劃模式。

本研究以文獻 Angel-Bello et al. (2017) 所提出之 Model V 為基礎，建構以下數學模式：延續及修改部分 2.1.2 小節模式，由於每個維修隊 k 修復路徑可以用 $P_k = \{o,[1]_k,[2]_k,...,[n_k]_k,d\}$ 來表示，我們改以另一種階層節線來表示修復路徑 $(v_{o,0}, v_{[1]_k,1}) \to (v_{[1]_k,1}, v_{[2]_k,2}) \to \cdots \to (v_{[n_k-1]_k,n_k-1}, v_{[n_k]_k,n_k})$。其中，階層節點$v_{i,m}$代表維修隊第$m$個修復的節點位置為節點$i$；階層節線$(v_{i,m}, v_{j,m+1})$若有被經過，代表某一組維修隊$k$

的修復路徑的$[m]_k = i, [m+1]_k = j$。因此種表示方式更為直覺，所以設計一新的層空網路圖$G_L = (\tilde{V}, \tilde{A})$如圖 3.8 所示。

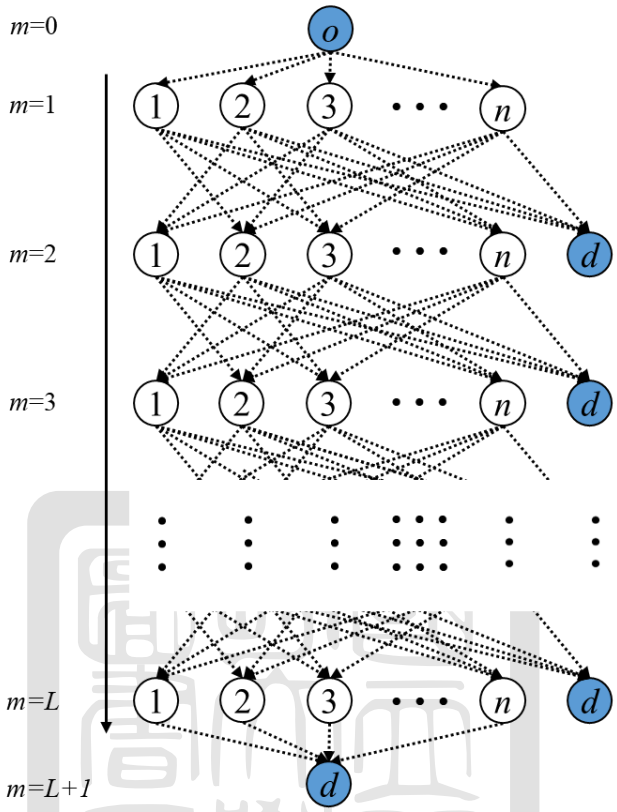

圖 3.8 本研究之層空網路圖

延續 2.1.2 小節模式的符號定義及限制式(2.9)~(2.11)，以下定義新增的數學符號：

參數

$AD_{i,t}$　　節點 i 在時刻 t 的累積需求量，$i \in N_r$

決策變數

$z_{i,t} \in \{0,1\}$　　若節點 i 在時刻 t 修復完工則為 1;反之為 0

目標式(3.11)改成在修復完工上限時刻 T 內，最大化所有點可被滿足的總累積需求量。

$$Maximize \quad \sum_{i\in N_r}\sum_{t=P_i}^{T} AD_{i,t} z_{i,t} \tag{3.11}$$

限制式(3.12)表示某組維修隊先修節點 i 再修節點 j，則節點 j 修復完成的時刻必大於等於節點 i 修復完成的時刻加上節點 j 修復所需時間；反之，則不限制。

$$T(1-\sum_{m=1}^{L-1}\sum_{(v_{i,m},v_{j,m+1})\in\tilde{A}} y_{(v_{i,m},v_{j,m+1})}) + w_j \geq w_i + c_{ij} \quad \forall i, j \in N_r; j \neq i \tag{3.12}$$

限制式(3.13)限制第一個修復節點的完工時刻大於等於 c_{oi}。

$$w_i \geq c_{oi} \quad \forall i \in N_r \tag{3.13}$$

限制式(3.14)、(3.15)將各故障節點修復完工的時刻變數轉換為二元變數表示。

$$\sum_{t=p_i}^{T} t z_{i,t} = w_i \quad \forall i \in N_r \tag{3.14}$$

$$\sum_{t=p_i}^{T} z_{i,t} = 1 \quad \forall i \in N_r \tag{3.15}$$

層空網路大小雖不直接被 T 值影響，但和 3.3 節的模式相同，需特別以變數記錄每個節點的修復完工時刻，再將其轉換為二元變數表示，致使模式求解時其對偶差值（Duality Gap）收斂緩慢，而導致求解效率不佳。

3.6 數學啟發式演算法（Matheuristics）

數學啟發式演算法為結合演算法和數學模式來求解問題之方法，首先在 3.6.1 小節介紹以優先修復最小損失需求的節點為選取機制的貪婪演算法 A_{LD}，接著在 3.6.2 小節藉由修正演算法 A_{LD} 的機制來求得數組層空網路路徑，再以這些路徑為基礎反向建構出其對應之限縮模式網路，最後在 3.6.3 小節提出針對該限縮模式網路所設計之以節線或路徑為變數的兩種整數規劃模式。

3.6.1 貪婪演算法 A_{LD}

貪婪演算法主要精神是每一步都選擇當前狀態下最有利的決策，逐步找到一組可行解。貪婪演算法雖不能保證找到最佳解，但由於其求解時間極短，不失為一種尋找近似解的方法。以下提出以優先修復最小損失需求的節點之演算法 A_{LD}，其損失需求量(Lost Demand，LD)定義為維修隊選擇修復節點的期間而導致其他節點未能滿足需求量的總值，此機制同時考量節點維修時間、旅行時間及依時變化需求。我們可將此演算法分兩階段介紹，在「建構階段」以貪婪機制產生一組解，再藉由「改善階段」持續改善此組解，最後得到的解即為此演算法的結果。

建構階段(Constructive Phase)

表 3.1 列出建構階段流程之各步驟，其中，已知之資料包含尚未修復故障節點集合 N_r、可派遣的維修隊數 K、以 $c_{i,j} := d_{i,j} + p_j$ 定義的節線旅行時間矩陣 C 及每個節點 i 的累積需求量 $AD_{i,t}$，N_s 表示尚未修復節點集合，對於每組維修隊 k 都有 $currentN_k$ 記錄維修隊最新修復的節點，$currentT_k$ 記錄維修隊最新修復節點的修復完工時刻，也就是維修隊可移動至下一個故障節點的時刻。

表 3.1 貪婪演算法 A_{LD} 的建構階段流程

Greedy Algorithm A_{LD} (Constructive Phase)

Data: N_r, K, C, $AD_{i,t}$

Initialization: $currentT_k = 0, currentN_k = o$ for each repairman $k \in \{1,...,K\}$, $N_s = N_r$

1: **While** $N_s \neq \varnothing$ **do**

2: Select a repairman $k \in \{1,2,...,K\}$ with the smallest $currentT_k$

3: Compute LD_i for each unrepaired node $i \in N_s$

4: Select node i with the smallest LD_i to repair

5: Update $N_s, currentT_k, currentN_k, S$

6: **End while**

Result: S

首先初始化參數 $N_s = N_r$，且對於每個維修隊 k 的 $currentT_k$ 初始化為 0，表示在時刻 0 時即可開始修復節點；$currentN_k$ 初始化為虛擬起點 o，代表所有維修隊皆從虛

擬起點 o 出發。再來先選擇最小 $currentT_k$ 的維修隊 k，計算若當下若先去修復節點 i 再去修復節點 j，比直接去修復節點 j 之作法所導致的整體需求損失量 $LD_i := \sum_{j \in N_s, j \neq i} (AD_{j,t} - AD_{j,t'})$，其中 $t = currentT_k + c_{currentN_{k,i}}$ 代表維修隊 k 若在當下直接去修復節點 j，其修復完節點 j 的時刻；而 $t' = currentT_k + c_{currentN_{k,i}} + c_{i,j}$ 則代表當下若先去修復節點 i 再去修復節點 j，其修復完節點 j 的時刻。因此，$AD_{j,t} - AD_{j,t'}$ 代表因為先選節點 i 致使損失的節點 j 需求量，而 LD_i 則為該損失量對所有當下仍待修的其它節點 $j \in N_s$ 加總而得。直覺上，我們認為應該優先選擇具最小損失需求量 LD_i 的節點 i 來修復，然後再更新待修節點集合 N_s 及變數 $currentT_k$ 、$currentN_k$ ，並以 S 記錄目前 K 組維修隊的修復排程。重複以上步驟直到所有節點皆已被修復完為止（即 $N_s = \varnothing$），如此可算出一套 K 組維修隊修復網路節點的排程 S。

改善階段(Improvement Phase)

表 3.2 列出了改善階段流程之各步驟，其中，已知之資料包含節線旅行時間矩陣 C 及每個節點 i 的累積需求量 $AD_{i,t}$，S 表示一組 K 組維修隊的修復排程。

表 3.2 貪婪演算法 A_{LD} 的改善階段流程(節點)

Greedy Algorithm A_{LD} (Improvement phase)
Data: *S, C, $AD_{i,t}$*
1: $O \leftarrow$ Compute objective value of solution S
2: **repeat**
3: (*S,O*) ←*exchange* (*S,O,C, $AD_{i,t}$*)
4: (*S,O*) ←*insertion* (*S,O,C, $AD_{i,t}$*)
5:**Until** (both procedures no longer improve its input solution)
Result: (*S,O*)

一開始先計算 S 的目標值並以 O 表示，之後皆以(*S,O*)表示目前解的狀態。以常見區域搜尋(Local Search)的方式來改善解：交換(exchange)及插入(insertion)，先進行交換搜尋檢查，再進行插入搜尋檢查。交換為任兩節點的位置順序交換，插入則為任選一節點插入至任何可能的位置，需檢查一遍所有可能組合，若有使其得到更好的解，則選擇使解改善最多的組合，再進行交換(插入)，並更新改變後解的狀態(*S,O*)。重複

以上步驟直到交換及插入都無法再改善解，然後結束。

3.6.2 產生限縮模式網路之方法設計

表 3.1 提出的演算法，因每一步皆選擇最小 $currentT_k$ （即最早有空）的維修隊 k 來修復最小 LD_i 的節點 i，若沒有同樣好的選項時，必將僅存一種可能的排程。為增加選擇性，我們可使用貪婪隨機選取機制，使每次選取在貪婪原則下具有變動的可能性，以產生不同的修復排程。假設迭代 R 次貪婪隨機選取機制，產生 R 組 K 條修復排程路徑，依此原則調整貪婪演算法 A'_{LD} 流程如表 3.3 所示，其隨機選取機制如下：隨機從前 h 低的 LD_i 之候選節點中選一個來修復，若尚未修復之故障節點數小於 h，則所有尚未修復故障節點數皆為候選節點。假設候選節點的集合為 Q，我們使用輪盤法，依「候選節點損失需求量越低，其被選取到的機率越高」之原則來隨機選取，因此每個候選節點 $e \in Q$ 被選取的機率定義為 $P(e):=(LD_e)^{-1} / \sum_{e \in Q} (LD_e)^{-1}$ 。

表 3.3 修正之貪婪演算法 A_{LD} 的建構階段流程

Modified Greedy Algorithm A'_{LD} (Constructive phase)

Data: N_r, K, C, $AD_{i,t}$

Initialization: $currentT_k = 0, currentN_k = o$ for each repairman $k \in \{1,...,K\}$, $N_s = N_r$

1: **While** $N_s \neq \varnothing$ **do**

2: Select a repairman $k \in \{1,2,...,K\}$ with the smallest $currentT_k$

3: Compute LD_i for each unrepaired node $i \in N_s$

4: Sort all unrepaired nodes $i \in N_s$ by LD_i from low to high. The first h low LD_i nodes are candiates.

5: Randomly select a candidate e by a probability $P(e)$

6: Update $N_s, currentT_k, currentN_k, S$

7: **End while**

Result: S

將 R 組 K 條修復排程的路徑解所反向對應出來的層空網路圖，我們稱之為「限縮模式網路」。圖 3.9 為產生限縮模式網路前後的兩個極端的狀況示意圖，左之圖層

空網路可表達所有的可行路徑；而右圖則只有一組 K 條可行修復排程路徑，因此我們須找到介於兩者之間的限縮網路圖來找到新的可行路徑。若以表 3.3 的演算法 A_{LD} 而言，這取決於迭代次數 R 及候選點個數 h 的大小，當 R 或 h 設定愈大，則節線數增加機率愈大，將愈偏向左圖；反之，則愈偏向右圖。由於優先選取損失需求最小節點的解通常會比隨機選取得到的解有更好的表現，因此 R 次迭代中的第一次迭代將以 3.6.1 節貪婪演算法 A_{LD} 的流程來做。

以下以 $N_r=\{A,B,C,D,E\}, K=2, R=3$ 舉例說明，利用演算法 A_{LD} 產生一組修復路徑 $P_1=\{o,A,C,d\}$ 、 $P_2=\{o,B,D,E,d\}$ ，使用修正之演算法 A_{LD} 而產生兩組修復路徑 $P_1=\{o,A,D,d\}$、$P_2=\{o,B,C,E,d\}$ 及另一組 $P_1=\{o,A,B,d\}$、$P_2=\{o,C,E,D,d\}$，將以上產生之路徑以階層節線表示如圖 3.10 的藍色虛線所示，再加上所有階層節點連至虛擬階層訖點的階層節線(灰色虛線)，以增加形成路徑 o-d 的可行性，即為限縮模式網路，圖中黃色及綠色實線的路徑為其中一套新的 K 組人員修復排程路徑。

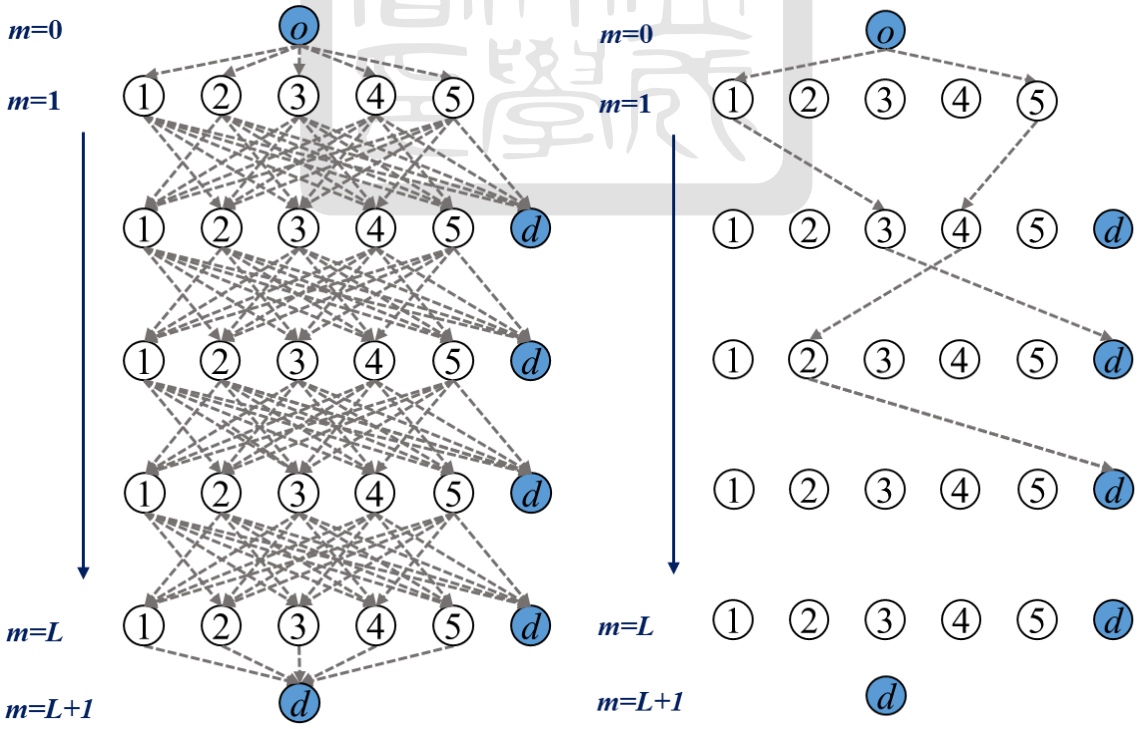

圖 3.9 以演算法 A_{LD} 產生限縮模式網路前後之示意圖

由於在 3.4.2 小節也有提及維修隊修復路徑可以用時空網路圖的時空節線來表示，但因時空節點牽涉時刻，在時空節線的旅行時間相異性高的情況下，多套 K 條維修隊修復路徑經過的時空節點之重複性低，幾乎不會產生新的其它修復路徑，因而並不適用將此方法套用在時空網路圖上。

由於限縮模式網路是以貪婪演算法的方式產生後再以數學模式求解，易陷入區域最佳解。為避免此情況，我們應將限縮模式網路再加上一些可能也不錯的節線，以建構新的網路，再求解新網路所對應的數學模式。為了能有效地找出額外不錯的節線，我們將求解數學模式 M_T^N 或 $M_T^{N_0}$ 過程中存取部分可行解，將這些可行解在網路圖上對應的節線加入限縮模式網路，如此可擴展貪婪解網路圖至於增加其跳脫區域最佳解的可能性。至於可行解的設定方式，可設定一求解時間上限 T_{MFS}，將數學模式在該時限內所找到的可行解之路徑記錄下來。

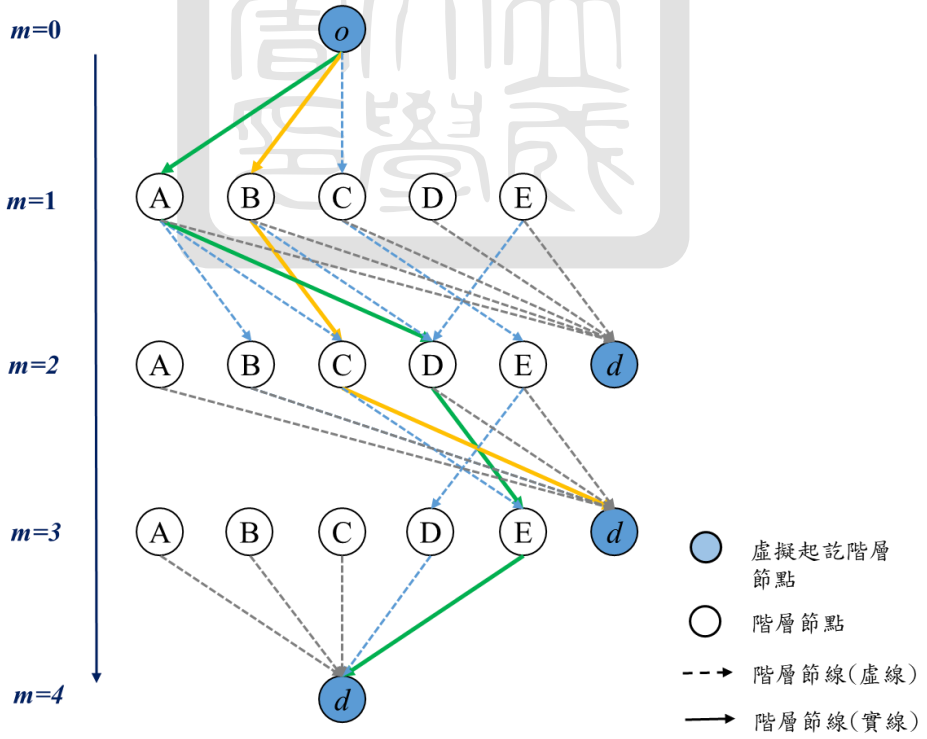

圖 3.10 限縮模式網路示意圖

3.6.3 限縮模式網路下之數學模式

本小節說明在限縮模式網路下可使用的數學模式，可分為以節線或路徑為變數建構之數學模式 $\bar{M}_{L_A}^N$ 或 $\bar{M}_{L_p}^N$。之所以同時測試兩種格式不同但目的相同的數學模式，是鑒於此類路徑規劃相關之整數規劃模式若以節線為變數來建構的話，經常會因為其對偶模式與主模式可行解之對偶差值過大而收斂極為緩慢。反之，以路徑當變數之模式其對偶差值會較好，因此可收斂較快。然而，路徑為變數之數學規劃模式主要困難之處在於必須在網路上產生所有可行的路徑，因為路徑個數會隨著節線個數增加而呈指數成長，因此這種作法必須在限縮網路下才可行。

由於以限縮網路為基礎所建構之模式 $\bar{M}_{L_A}^N$ 其實就是 M_L^N 模式在限縮網路的版本，差別只在於限縮層模式網路的層空節線數較少，因此不再重複說明該數學模式，而僅針對以路徑為變數所建構之數學模式 $\bar{M}_{L_p}^N$ 作說明:

接續先前 3.6.2 小節的限縮模式網路說明，我們可利用深度優先搜尋法(Depth-first search，DFS)找出所有可行之修復路徑，再以這些路徑為變數來建構數學模式。深度優先搜尋法是一種用來遍尋一個網路圖的演算法。由圖上某一節點當成起點開始搜尋，首先先搜尋節線上未被經過的一節點，並儘可能深的搜索，直到該節點的所有節線上節點都已探尋，就回溯到前一個節點，重覆探尋未搜尋的節點，直到找到目的的節點或遍尋全部節點。利用 DFS 可以將同一起訖點間的所有路徑遍尋出來，因此可將其所找出的個別路徑給予一個變數，以此為基礎來重新建構數學模式 $\bar{M}_{L_p}^N$。

觀察層空網路可發現為一個同向子圖（Acyclic Subgraph）的結構，其中虛擬階層起點 $v_{o,0}$ 即為起點，而路徑的長度頂多有 L+1 段，可用 DFS 來找尋所有虛擬起訖階層節點的所有路徑。由於無法確定各組修復員完工的階層，因此必須在 $m=2,3,...,L+1$ 各層分別設置一個虛擬階層訖點 $v_{d,m}$，而這些虛擬階層訖點再全部連至虛擬的總訖點 $v_{d,m'+1}$。因此若要產生經過 m' 個故障節點的修復路徑，則可以以 DFS 找尋 $v_{o,0}-v_{d,m'+1}$

的所有路徑。因為每個故障節點只會被經過一次，所以每探尋一個階層節點 $v_{i,m}$ 需將隸屬於同一故障節點 i 的所有 $m=1,2,....,L$ 階層之節點 $v_{i,m}$ 皆記錄為已拜訪過，如此才不會在後續的搜索中重複拜訪同一故障節點。同理，在回溯前一個節點時，則需將現階層其它隸屬同一故障節點的階層節點皆更新為未拜訪過，如此該點在後續搜索中才仍保有在未來能被拜訪到的機會。

以 DFS 產生所有可行的修復路徑，因在搜索時可限制修復路徑所經過的故障節點數，因此可依使用者的需求來自行定義修復路徑經過故障節點數之下限為某參數 L_{min}。倘若各節線之旅行時間差異不大的話，一組好的修復排程應該會均化其維修隊的工作負荷（亦即每組維修隊所拜訪的個數趨於平均），如此可排除選擇節點數過少的修復路徑。

數學模式

以下定義符號：

集合

P　　限縮多階層網路上產生的修復路徑集合，$p \in P$

參數

$\alpha_{p,i}$　　修復路徑 p 經過節點的次數(1 或 0)

AD_p　　修復路徑 p 對應的節點累積需求量

決策變數

$x_p \in \{0,1\}$　　若修復路徑 p 被選擇則為 1；反之為 0

因路徑 p 為已知資訊，可得路徑 p 上所有節點的修復完工時刻，AD_p 為該路徑 p 所經過（即修復）之節點其可被滿足之總體累積需求量。因此目標式可表達如式(3.16)，最大化所有被選擇的修復路徑 p 之對應路徑的累積需求量 AD_p 加總，即為所

有可滿足的節點累積需求量。

$$Maximize \sum_{p \in P} AD_p x_p \tag{3.16}$$

限制式(3.17)限制最多 K 條修復路徑，因可派遣維修隊組數為 K 組。

$$\sum_{p \in P} x_p \le K \tag{3.17}$$

限制式(3.18)限制每個故障節點只能剛好被一條被選擇的修復路徑所經過，也就是每個故障節點都要被修復一次。

$$\sum_{p \in P} \alpha_{p,i} x_p = 1 \quad \forall i \in N_r \tag{3.18}$$

3.7 模擬退火法(Simulated Annealing，SA)

模擬退火演算法的概念啟發自冶金學的「退火」，是此將材料加熱再經特定速率冷卻的過程，目的是調整材料結晶組織進而改變材料硬度、延展性等性質。當溫度夠高時，材料中的原子可隨機自由移動離開原來內能局部最小值的位置；當溫度緩慢變小時，原子移動範圍受限制會逐漸地向低能量的區域集中，藉此方法使得有較多機會可以找到內能比原先更低的位置。

模擬退火法為一種元啟發式演算法(metaheuristic)，最早由 Kirkpatrick et al. (1983)提出，已被廣泛應用在路徑規劃問題、排程問題等最佳化問題中。由於其具有跳離區域最佳解的機制，相較於 3.6.1 小節提出的貪婪演算法，可解決易陷入區域最佳解的問題。因此，本研究亦將針對節點需求型態問題設計一模擬退火演算法。

編碼方式(encode)

每組可行解我們皆可以以一陣列來儲存，若現在有 3 個維修隊負責搶修作業且有 10 個故障節點需進行修復，可將其編碼以圖 3.11 表示之，每組維修隊以「0」開頭用來標示維修隊修復的起始位置並用來區分維修隊間的修復順序，以此例而言，維修隊 1 修復順序依序為 3、6、10、8，維修隊 2、3 修復順序也可以此類推。若有 k 維修隊

需修復 n 個故障節點，則陣列的長度為 $k+n$，由 k 個 0 及每個故障節點的編號 1,2,…,n 組成，藉由此方式表示成一組可形解。

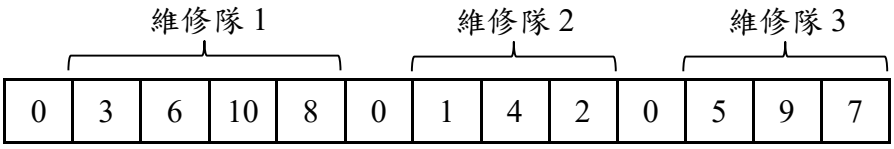

圖 3.11 編碼示意圖

退火程序(annealing schedule)

退火程序指的是降溫過程，首先需設定參數為初始溫度 T_0、冷卻率 $\alpha \in [0,1]$，每經一次迭代溫度會下降一次，我們以 T_t 表示第 t 次迭代後的溫度，降溫機制設定為 $T_t := \alpha T_{t-1} = \alpha^t T_0$，終止條件設定為當溫度小於終止溫度 T_f 時即結束迭代。因此，若已設定固定參數 T_0、α、T_f，最終迭代的次數可計算為 $\lceil \log_\alpha (T_f / T_0) \rceil - 1$。

能量函式

設定能量函式 E 可以求得解 S 的目標值 $E(S)$，由於本研究問題為最大化節點可滿足之需求量，故其值越大表示越接近最佳解，有別於原退火模擬法尋找最小化內能的解之設定，只需在修正後續介紹之接受準則的機率函式即可。

產生鄰近解的方式

每次迭代須以目前解 S 為中心隨機產生一新的鄰近解 S'，在此分為「交換」及「插入」兩種產生的方式。我們設定一交換機率 P_C，隨機產生一個介於 0、1 之間的亂數，若亂數小於 P_C 則以交換方式產生新的鄰近解；反之，則以插入方式產生。首先說明交換產生的方式：針對陣列隨機產生兩個不同且儲存編號非 0 的位置 p_1 及 p_2，將此兩位置儲存編號進行交換，此方式並不影響原來各維修隊修復節點的個數，如圖 3.12 所示。接著說明插入產生的方式，針對陣列隨機產生兩個不同的陣列位置 p_1 及 p_2 且 p_1 儲存編號不為 0，將位置 p_1 儲存的編號插入至位置 p_2，剩餘編號的順序不變，

介於 p_1 及 p_2 之間的編號視情況往前或往後遞補，如圖 3.13。

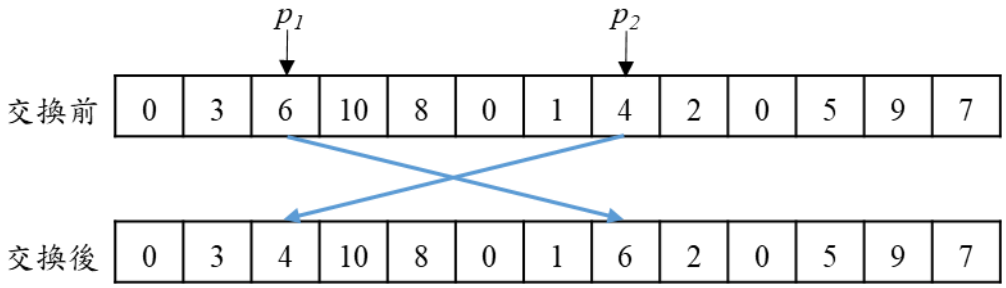

圖 3.12 交換前後示意圖

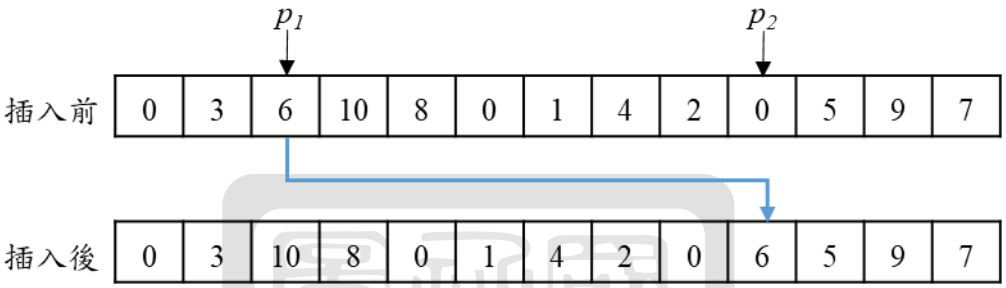

圖 3.13 插入前後示意圖

接受準則

由於本研究問題為最大化目標值，故修正模擬退火法原有的準則，採用 Metropolis 接受準則，來決定是否接受一個能量變動的改變。首先計算目前解 S 的能量值 $E(S)$，以目前解為中心隨機產生新的鄰近解 S' 並計算其能量 $E(S')$，$\Delta E = E(S') - E(S)$ 為兩個解之間的能量差，若 $\Delta E > 0$，代表新的鄰近解的目標值比目前解來得好，接受鄰近解取代目前解；反之，代表所產生的解比目前解差，則利用機率函數和控制溫度參數來判斷是否接受新解。其中，機率函式為 $P(\Delta E, T) := e^{\Delta E/T}$，這也說明模擬退火法有機會跳脫區域最佳解，透過降溫機制來控制收斂的速度，（隨著溫度下降，較不會接受較差的新解；當溫度趨近於 0 時，僅會接受較佳的解進而達到收斂）。

綜上所述，表 3.4 列出模擬退火法流程，其中需決定參數有初始解 S_0，退火程序中的起始溫度 T_0、終止溫度 T_f、冷卻率 α，P_C 為以交換方式產生一鄰近解的機率。

表 3.4 模擬退火法的流程

Simulated Annealing (SA)
Data: N_r, K, C, $AD_{i,t}$ **Parameters**: $S_0, T_0, T_f, \alpha, P_C$ **Initialization:** $S = S_0, T = T_0$ 1: **While** $T \geq T_f$ **do** 2: Generate a new neighborhood solution $S'(S, P_C)$ 3: $\Delta E = E(S') - E(S)$ 3: **If** $\Delta E > 0$, **then** $S = S'$ 4: **Else if** $P(\Delta E, T) > random[0,1]$, **then** $S = S'$ 5: $T = \alpha T$ 6: **End while**

3.8 小結

本章節先討論節點需求型態的問題定義及假設，接著提出網路縮減機制，將原問題網路大小簡化僅包含故障節點之網路。接著，我們提出數種整數規劃模式，延續多組旅行維修員問題之數學模式，將目標式改以最大化可滿足的節點累積需求量，並提出包含排除子迴圈限制式之模式 M_s^N。然而經由數值測試，發現其求解效率不佳而不建議採用。以時空網路為基礎，我們建構了特例模式 $M_T^{N_0}$ 及一般模式 M_T^N；在旅行時間可忽略下，$M_T^{N_0}$ 應比 M_T^N 好，然而此種模式將受 T 值大小影響，所以提出以層空網路為基礎建構 M_L^N。雖然層空展開網路較時空展開網路小，且不受 T 值影響，但卻須以變數記錄每個故障節點的修復完工時刻，並再將之轉換成二元變數，因而導致求解效率不佳。隨著網路增大或維修隊數（K）下降，模式求解時間將驟增，因此我們又提出能結合貪婪演算法及數學模式的數學啟發式演算法以及模擬退火法，期能在短時間內找到還不錯的解。下一章將探討起訖組合需求型態之節點修復排程問題。

第四章 考慮依時變化的起訖組合需求量之網路節點修復排程

本章首先介紹起訖組合需求型態問題及其假設，接著提出多種數學模式及演算法，皆為修正第三章所提出的模式與演算法，因兩者問題的差別只在於滿足的需求型態不同，所以本章的脈絡與第三章雷同，所建構之各數學模式關係整理如圖 4.1 所示，其模式符號上標改為 A，表示起訖組合（類似節線 Arc 之意）需求型態問題。

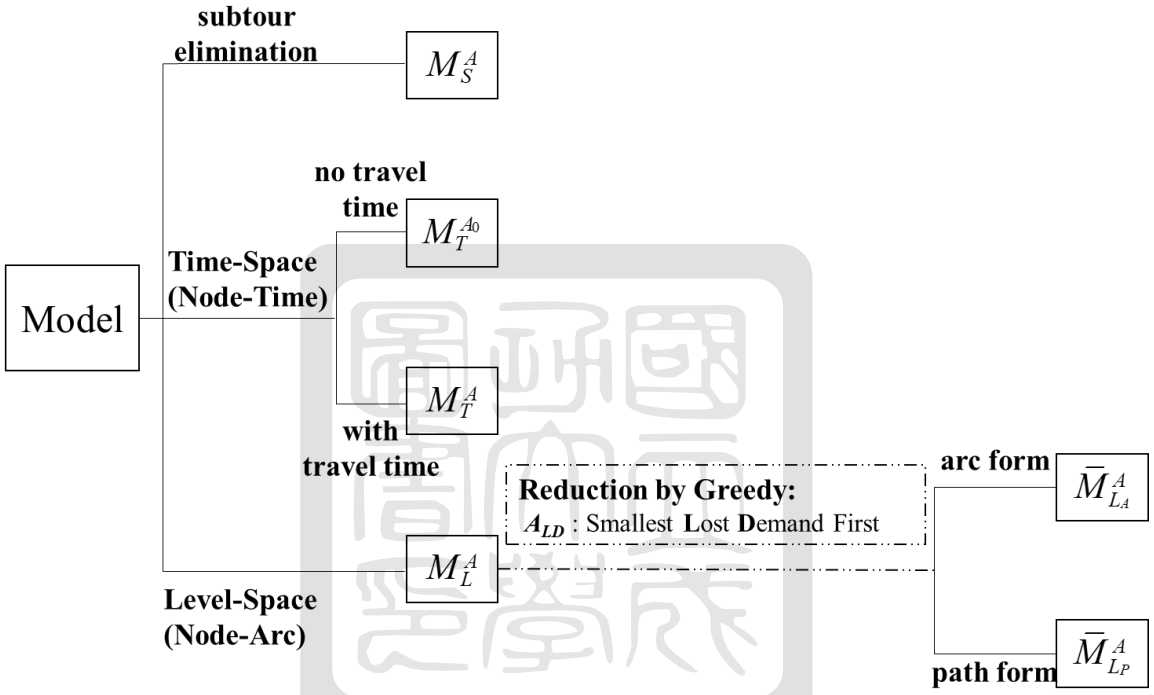

圖 4.1 起訖組合需求型態問題之數學模式示意圖

4.1 問題描述與假設

假設起訖組合需求型態問題之網路 $G^A=(N,E,OD)$，其中 OD 為所起訖組合的集合，大致和上段節點需求型態問題敘述相同，都是要修復所有故障節點以滿足最多的需求量，而主要的差別在於本章之需求量是在起訖組合而非在在單一節點上。在時刻 t 每個起訖點組合(i,j)有其需求量 $D_{(i,j),t}$，若起訖節點 i,j 皆已完成修復可滿足 $D_{(i,j),t}$，則稱起訖點組合(i,j)為「連通」狀態；反之，若有任一起訖點 i,j 尚未修復完工而無

法滿足起訖組合上之需求，則稱起訖點組合(i,j)為「不連通」狀態。我們將起訖點組合(i,j)最早連通的時刻 t 定義為起訖組合(i,j)打通的時刻，且在此時刻起訖點組合(i,j)可滿足的總需求量定義為累積需求量 $AD_{(i,j),t} := D_{(i,j),t} + D_{(i,j),t+1} + \cdots + D_{(i,j),T}$ 。

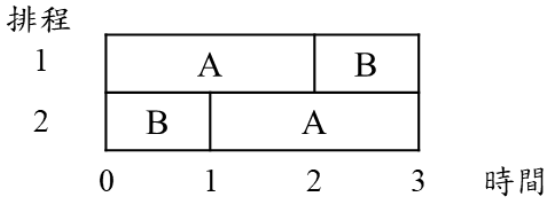

圖 4.2 兩種修復順序排程圖

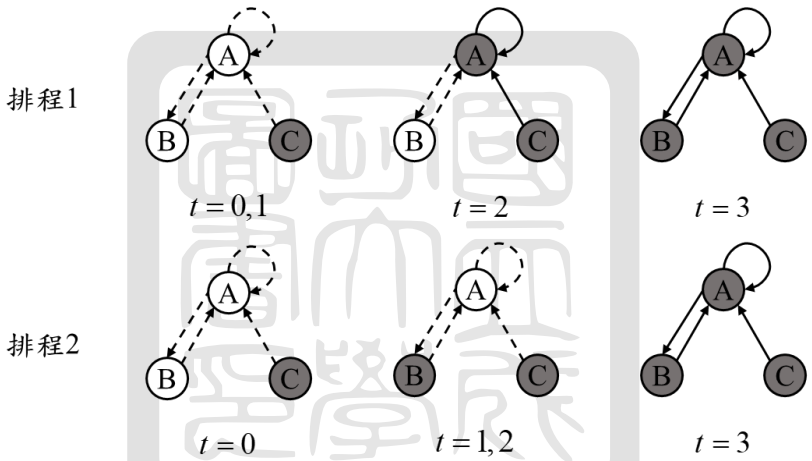

圖 4.3 兩種修復順序下起訖點組合連通變化示意圖

以下舉例說明起訖組合依時變化量的連通的關係：假設一網路有兩個故障節點 A、B 及未故障節點 C，已知節點 A、B 的維修時間分別為 2、1 單位時間，在移動時間可忽略的情況下之修復排程如圖 4.2 所示，圖 4.3 為起訖組合依圖 4.2 的兩種修復排程連通狀態變化示意圖，在 $t = 0$ 為網路的初始狀態，其中實心節點表示正常運作的節點(未發生故障或者是已修復完成)，空心節點則表示尚未修復之故障節點，而起訖組合以有向節線方式表示(起點指向訖點)，虛線為不連通狀態，實線為連通狀態。可觀察到排程 1 起訖組合 OD={(A, A)、(C, A)、(A, B)、(B, A)}的打通時刻分別為 2、2、

3、3 單位時刻，因此排程 1 可滿足總起訖需求量為 $AD_{(A,A),2}+AD_{(C,A),2}+AD_{(A,B),3}+AD_{(B,A),3}$，然而排程2所有起訖組合皆在時刻3打通，可滿足總起訖需求量為 $AD_{(A,A),3}+AD_{(C,A),3}+AD_{(A,B),3}+AD_{(B,A),3}$。因累積需求量隨時間而遞減，$AD_{(A,A),3}\le AD_{(A,A),2}$，$AD_{(C,A),3}\le AD_{(C,A),2}$，所以此例之排程 1 優於排程 2。

4.2 網路縮減

倘若部分節點並未故障，可藉由下面方法來縮減網路：由於起訖組合是建立在起訖節點之節線上，可將起訖組合分為三種情況：(1)起訖點皆為故障節點；(2)起訖點皆為未故障節點；(3)起訖點其中之一為故障節點。其中(2)之可滿足需求量可視為一已知常數，故目標式可只看情況(1)(3)可滿足的起訖組合需求量。假設故障節點的集合為 N_r，未發生故障節點集合為 N_s，以下分步驟進行網路縮減如下：

【步驟一】移除所有情況(2)的起訖組合的集合 $\{(i,j):(i,j)\in OD,\ i,j\in N_s\}$。

【步驟二】對於每個故障節點 i，$\{(i,j):(i,j)\in OD,\ j\in N_s\}\cup\{(j,i):(j,i)\in OD,\ j\in N_s\}$ 情況(3)起訖組合的集合合併成一個起訖組合 (i,i)，若原先有起訖組合 $(i,i)\in OD$，則將需求量重新定義 $D_{(i,i),t}:=D_{(i,i),t}+\sum_{(i,j)\in OD,\ j\in N_s}D_{(i,j),t}+\sum_{(j,i)\in OD,\ j\in N_s}D_{(j,i),t}$，反之，則新增起訖組合 (i,i)，其需求量定義 $D_{(i,i),t}:=\sum_{(i,j)\in OD,\ j\in N_s}D_{(i,j),t}+\sum_{(j,i)\in OD,\ j\in N_s}D_{(j,i),t}$。

【步驟三】移除所有非故障的節點的集合，而將網路圖 $G^A=(N,E,OD)$ 縮減至網路圖 $G^A=(N_r,E',OD')$，其中 $E'\subset E$ 為任兩點故障節點相連的無向節線集合，縮減後的網路依舊是完全圖網路。

我們以例子說明上述步驟，假設有一網路如圖 4.4 所示，首先移除起訖組合(G, E)、(G, G)，再將(G, D)、(E, D)合併到新增的(D, D)，將(C, G)、(C, A)、(A, C)合併到新增的(C, C)，然後將(A, B)合併到(B, B)及(F, E)合併到(F, F)，最後將所有未故障節點

移除，得到圖 4.5。

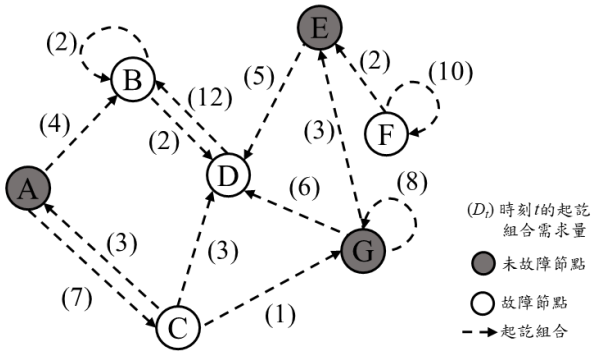

圖 4.4 縮減前之網路圖 G^A

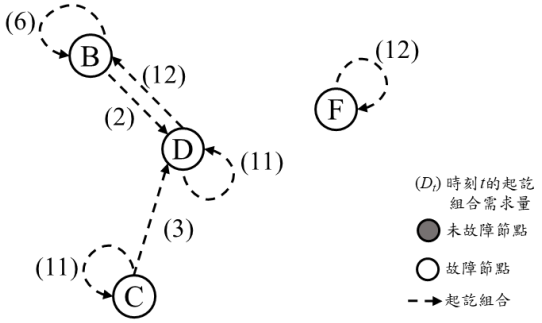

圖 4.5 縮減後之網路圖 G^A

4.3 含排除子迴圈限制式之數學模式 M_S^A

以下新增符號的定義：

集合

OD	所有起訖組合構成之集合，$(i,j)\in OD$

參數

$AD_{(i,j),t}$	起訖組合(i,j)在時刻 t 的累積需求量

決策變數

$s_{(i,t).(j,t')}\in\{0,1\}$	若節點 i 及 j 分別在時刻 t 及 t' 修復完工則為 1；反之為 0

與單一節點需求型態之問題相比，起訖組合需求型態仍舊是以多組維修隊修復所有故障節點，因此其數學模式可直接使用 3.3 小節的限制式(3.2)~(3.3)及 2.1.1 小節的限制式(2.3)~(2.7)，目標式則改為最大化可滿足的起訖組合累積需求量，如式(4.1)所示。式(4.1)之第一項考慮起訖組合(i,j)在起訖節點不相同的情況，需以變數$s_{(i,t).(j,t')}$判斷是否節點 i 及 j 分別在時刻 t 及 t' 修復完工，若是則起訖組合(i,j)在時刻$\max\{t,t'\}$打通；式(4.1)之第二項為起訖組合(i,i)在起訖節點相同的情況，節點 i 修復完工時刻 t 即是起訖組合(i,i)的打通時刻。

$$Maximize \quad \sum_{\substack{(i,j)\in OD,\\ i\neq j}} \sum_{t=c_{oi}}^{T} \sum_{t'=c_{oj}}^{T} AD_{(i,j),\max\{t,t'\}} s_{(i,t),(j,t')} + \sum_{t=c_{oi}}^{T} AD_{(i,i),t} z_{i,t} \tag{4.1}$$

式(4.2)表示當 $s_{(i,t),(j,t')}=1$ 時，則限制變數 $z_{i,t}$ 、 $z_{j,t'}$ 必須為 1，反之則不限制。

$$2s_{(i,t),(j,t')} \le z_{i,t} + z_{j,t'} \quad \forall (i,j)\in OD; i\neq j; t=c_{oi}, c_{oi}+1,...,T;\ \ t'=c_{oj}, c_{oj}+1,...,T \tag{4.2}$$

4.4 以時空網路為基礎建構之數學模式

本節提出以時空網路為基礎建構模式，4.4.1 小節將介紹旅行時間可被忽略下之特例模式，4.4.2 小節則介紹一般模式。

4.4.1 特例模式 $M_T^{A_0}$

以下為新增符號的定義：

集合

OD　　所有起訖組合構成之集合，$(i,j)\in OD$

參數

$AD_{(i,j),t}$　　起訖組合 (i,j) 在時刻 t 的累積需求量

決策變數

$s_{(i,t).(j,t')}\in\{0,1\}$　若節點 i 及 j 分別在時刻 t 及 t' 修復完工則為 1；反之為 0

此數學模式延續 3.4.1 小節的限制式(3.5)~(3.6)，和 4.3 小節新增的目標式與限制式雷同，如目標式(4.3)及限制式(4.4)所示。

$$Maximize \sum_{\substack{(i,j)\in OD,\\ i\neq j}} \sum_{t=p_i}^{T} \sum_{t'=p_j}^{T} AD_{(i,j),\max\{t,t'\}} s_{(i,t),(j,t')} + \sum_{i\in N} \sum_{t=0}^{T-p_i} AD_{(i,i),t+p_i} x_{i,t} \tag{4.3}$$

$$2s_{(i,t+p_i),(j,t'+p_j)} \le x_{i,t} + x_{j,t'} \quad \forall (i,j)\in OD; i\neq j; t=0,1,...,T-p_i;\ \ t'=0,1,...,T-p_j \tag{4.4}$$

4.4.2 一般模式 M_T^A

以下為新增符號的定義：

集合

OD　　所有起訖組合構成之集合，$(i,j)\in OD$

參數

$AD_{(i,j),t}$　　起訖組合(i,j)在時刻 t 的累積需求量

決策變數

$s_{(i,t).(j,t')}\in\{0,1\}$　　若節點 i 及 j 分別在時刻 t 及 t' 修復完工則為 1；反之為 0

此數學模式包含 3.4.2 小節的限制式(3.8)~(3.10)，而目標式為最大化可滿足的起訖組合累積需求量如式(4.5)所示，其第一項為起訖組合(i,j)在起訖節點非同個節點的情況下，需以變數$s_{(i,t),(j,t')}$判斷在起訖組合(i,j)是否在時刻 t 打通，而第二項則為起訖組合(i,i)在起訖節點皆為同個節點的情況下，若有時空節線進入時空節點$v_{i,t}$（即 $\hat{A}^-_{v_{i,t}}$ 中有某條被經過），則節點 i 將在時刻 t 修復完工，而時刻 t 即是起訖組合(i,i)的打通時刻。

$$Maximize \sum_{\substack{(i,j)\in OD,\\ i\neq j}} \sum_{t=c_{oi}}^{T} \sum_{t'=c_{oj}}^{T} AD_{(i,j),\max\{t,t'\}} s_{(i,t),(j,t')} + \sum_{i\in N_r} \sum_{(v_{j',t''},v_{i,t})\in \hat{A}^-_{v_{i,t}}} AD_{(i,i),t}\, x_{(v_{j',t''},v_{i,t})} \tag{4.5}$$

式(4.6)表示當 $s_{(i,t),(j,t')}=1$ 時，則限制時空節點 $v_{i,t}$ 及 $v_{j,t'}$ 皆必須有連進的時空節線，反之則不限制。

$$\begin{aligned} &2s_{(i,t),(j,t')} \le \sum_{(v_{j',t''},v_{i,t})\in \hat{A}^-_{v_{i,t}}} x_{(v_{j',t''},v_{i,t})} + \sum_{(v_{i',t'''},v_{j,t'})\in \hat{A}^-_{v_{j,t'}}} x_{(v_{i',t'''},v_{j,t'})} \\ &\forall (i,j)\in OD; i\neq j; t=c_{oi},c_{oi}+1,...,T; t'=c_{oj},c_{oj}+1,...,T \end{aligned} \tag{4.6}$$

4.5 以層空網路為基礎建構之數學模式 M_L^A

以下為新增符號的定義：

集合

OD　　所有起訖組合構成之集合，$(i, j) \in OD$

參數

$AD_{(i,j),t}$　　起訖組合(i, j)在時刻 t 的累積需求量

決策變數

$s_{(i,t).(j,t')} \in \{0,1\}$　　若節點 i 及 j 分別在時刻 t 及 t' 修復完工則為 1；反之為 0

此數學模式與 3.5 小節模式雷同，因此包含限制式(3.12)~(3.15)及 2.1.2 小節的限制式(2.9)~(2.11)，而目標式則改為最大化可滿足起訖組合累積需求量如式(4.7)所示。

$$Maximize \quad \sum_{\substack{(i,j)\in OD,\\ i\neq j}} \sum_{t=p_i}^{T} \sum_{t'=p_j}^{T} AD_{(i,j),\max\{t,t'\}} s_{(i,t),(j,t')} + \sum_{t=c_{oi}}^{T} AD_{(i,i),t} z_{i,t} \tag{4.7}$$

式(4.8)表示當 $s_{(i,t),(j,t')}=1$ 時，則限制變數 $z_{i,t}$ 、 $z_{j,t'}$ 必須為 1，反之則不限制。

$$2s_{(i,t),(j,t')} \le z_{i,t} + z_{j,t'} \qquad \forall (i,j) \in OD;\ t = c_{oi}, c_{oi}+1, ..., T;\ t' = c_{oj}, c_{oj}+1, ..., T \tag{4.8}$$

4.6 數學啟發式演算法（Matheuristics）

本節針對起訖組合需求型態問題提出數學啟發式演算法，首先在 4.6.1 節將起訖組合需求量簡化成節點需求量，可延用 3.6.1 及 3.6.2 節的方法來產生限縮模式網路，最後在 4.6.2 節提出在限縮模式網路下以節線或路徑為變數之兩種整數規劃模式。

4.6.1 起訖組合需求量簡化為單一節點需求量

將每個時刻 t 每個起訖組合(i, j)的需求量 $D_{(i,j),t}$ 各分一半在起訖節點的需求量上，

定義節點 i 需求量為 $D_{i,t} := \sum_{(i,j)\in OD} 0.5D_{(i,j),t}$，進而可以計算其兩端節點之各別累積需求量 $AD_{i,t}$，如此即可直接套用先前在單一節點的作法來處理。圖 4.6 為將圖 4.5 起訖組合需求量簡化為節點需求量示意圖。

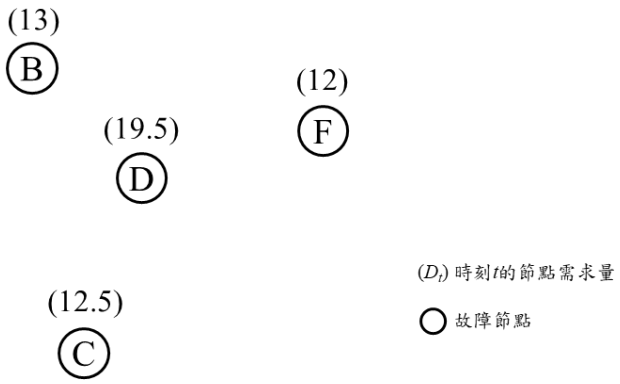

圖 4.6 將起訖組合需求量簡化為節點需求量以圖 4.5 為例

以此方式將起訖組合需求型態問題簡化成單一節點需求型態問題，以表 3.1 的貪婪演算法建構階段流程得到一組解，再藉由表 4.1 的貪婪演算法改善階段流程，以起訖組合需求型態問題的目標來計算目標值用來持續改善解。在產生限縮模式網路仍延用 3.6.2 設計的方法。

表 4.1 貪婪演算法 A_{LD} 的改善階段流程(起訖組合)

Greedy Algorithm A_{LD} (Improvement phase)
Data: *S, C, $AD_{(i,j),t}$*
1: $O \leftarrow$ Compute objective value of solution S
2: **repeat**
3: (S,O) ←*exchange* ($S, O, C, AD_{(i,j),t}$)
4: (S,O) ←*insertion* ($S, O, C, AD_{(i,j),t}$)
5:**Until** (one of these procedures can not improve its input solution)
Result: (S,O)

4.6.2 限縮模式網路下之數學模式

本小節說明在限縮模式網路下可使用的數學模式，可分為以節線或路徑為變數建構之數學模式 $\bar{M}^{A}_{L_A}$ 或 $\bar{M}^{A}_{L_p}$，此兩模式理論上會得到同樣的最佳解，雖然前者看似有較

少的變數，但實務上通常以後者之模式收斂較快，此處之說明與 3.6.3 小節雷同，以下將以路徑為變數建構之數學模式 $\bar{M}_{L_p}^{A}$ 作說明。

以下介紹符號定義:

參數

$AD_{(p,p')}$　　修復路徑組合(p,p')對應的起訖組合累積需求量

決策變數

$s_{(p,p')} \in \{0,1\}$　　若修復路徑組合(p,p')皆被選擇則為 1；其餘情況則為 0

此模式包含 3.6.3 小節的限制式(3.17)~(3.18)，目標式(4.9)為最大化所有被選擇的修復路徑 p 之對應的路徑累積需求量 $AD_{(p,p')}$ 之加總。因路徑 p 為已知資訊，可得路徑 p 上所有經過節點的修復完工時刻，因此若路徑 p,p' 皆有被選到，則在路徑 p 上所有經過的故障節點 i 及在路徑 p' 上所有經過的故障節點 j 皆可知其修復完工時刻。因此可推知起訖組合 $(i,j) \in OD$ 的打通時刻，及可滿足其打通時刻之起訖組合累積需求量。

$$Maximize \sum_{p\in P}\sum_{p'\in P, p'\neq p} AD_{(p,p')}s_{(p,p')} + \sum_{p\in P} AD_{(p,p)}x_p \tag{4.9}$$

式(4.10)代表當變數 $s_{(p,p')}=1$ 時，則限制變數 x_p 及 $x_{p'}$ 等於 1(修復路徑 p,p' 被選擇)，反之則不限制。

$$2s_{(p,p')} \le x_p + x_{p'} \quad \forall p,p' \in P; p \neq p' \tag{4.10}$$

4.7 模擬退火法(Simulated Annealing, SA)

針對起訖組合需求型態問題的模擬退火法和 3.7 節的做法相同，唯一的差別在於能量函式 E 有些微的不同。$E(S)$ 改為計算此解所能滿足起訖組合需求量之總和，其中已知資料應改為尚未修復故障節點集合 N_r、可派遣的維修隊數 K、節線旅行時間矩陣 C 及每對起訖組合 (i,j) 的累積需求量 $AD_{(i,j),t}$。

4.8 小結

本章探討起訖組合需求型態的問題，說明其定義及假設後，我們提出網路縮減機制，將原問題網路大小簡化成為僅包含故障節點之網路。本章所提出之數種整數規劃模式及演算法大多改良自第三章的模式，其數學模式將目標式改成最大化可滿足的起訖組合累積需求量，新增起訖組合連通之相關限制式；數學啟發式演算法則藉由將起訖組合需求量分一半至起訖兩節點的需求量，再套用 3.6.2 小節介紹的方法來產生限縮模式網路，接著再以數學模式 $\bar{M}_{L_A}^{A}$ 或 $\bar{M}_{L_p}^{A}$ 進行求解；模擬退火法則是將能量函式改為可滿足的起訖組合累積需求量之總和。下一章將針對各數學模式及演算法進行測試。

第五章 數值分析

本章節將會針對本研究所建立的數學模式及數學啟發式演算法進行測試與分析，分別在 5.1 節及 5.2 節測試「節點」及「起訖組合」需求型態問題的網路。本研究測試環境為 Window 10 作業系統，搭配 Intel Core i7-6770，3.40GHz*8 處理器與 16GB 記憶體，以 Microsoft Visual Studio 2015 為程式語言介面，使用 C++程式語言撰寫程式，並利用最佳化求解軟體 Gurobi 8.0.1 版求解數學模式，其中求解時間 *CPU Time* 設定為最多 2 小時(7200 秒)。以下首先說明測試網路參數設定，接著針對不同參數設計及網路大小等不同狀況來分析測試結果。

5.1 測試資料參數設定

本研究探討問題的網路為完全圖網路，測試資料已經經過網路縮減，也就是網路的節點已排除非故障節點，表 5.1 為測試資料參數的設定，其中網路節點數和維修隊數分別以 n 和 k 表示之，n 決定網路規模的大小，而維修時間、旅行時間以及需求量皆以表中設定的範圍產生隨機整數；規劃時間 T 的估算設定為平均一組維修隊修復的節點數(若非整數無條件進位)乘上維修時間加上旅行時間範圍的上限值，若在旅行時間可忽略下的情境，旅行時間及規劃時間的設定以括號中的數值表示。後續將以表 5.1 參數設定，針對不同網路規模下以及不同維修隊數的例子來產生測試資料。

表 5.1 測試資料參數資訊

網路結構	參數設定					
	節點數	維修隊數	維修時間	旅行時間	規劃時間	節點/起訖組合需求量(每時刻)
完全圖	n	K	[3,8]	[0,7] (0)	T=15$\lceil n/k \rceil$ (T=8$\lceil n/k \rceil$)	[20,50]

5.2 節點需求型態問題之測試

針對節點需求型態問題，我們先比較可算出精確解的數種整數規劃模式，再比較數學啟發式演算法的求解效率及求解品質。

5.2.1 整數規劃模式比較

本小節比較本研究提出可得精確解之三種整數規劃模式，表 5.2 為其各模式之大小估算，以 *big-O* 估算模式的變數及限制式的個數，可看出模式的大小主要取決於節點數 n、規劃時間 T 以及修復節點數上限 L，而其中 T 及 L 的大小設定又可取決於節點數 n 及維修隊數 k，n 愈大 k 愈小則 T、L 愈大。

表 5.3、5.4 為在旅行時間可忽略下比較模式 M_S^N、$M_T^{N_0}$、M_L^N 的求解表現；表 5.5、5.6 為不可忽略旅行時間下比較模式 M_S^N、M_T^N、M_L^N 的求解表現；表 5.7 為比較以時空網路為基礎建構之特例模式 $M_T^{N_0}$ 和一般模式 $M_T^{N_0}$ 的求解表現，其中 *Num* 為該組網路測試資料數；*Num opt.*代表在該組網路測試資料中求得最佳解的次數；*CPU Time* 為數學模式平均求解時間，若無法在求解時限 7200 (秒)內求得最佳解，該例之求解時間將直接以求解時限 7200 (秒) 計入平均；*B-Gap* 為用最佳化軟體 Gurobi 之測試平均結果（求解時限為 7200 秒）的計算方式如下：

$$B\text{-}Gap = \frac{UB - LB}{UB} \times 100\%$$。*UB* 及 *LB* 為 Gurobi 找到之最佳上下界。

表 5.2 節點需求型態各整數規劃模式之問題大小估算

建構網路	數學模式	二元變數個數	實數變數個數	限制式個數
一般	M_S^N	$O(n^2+nT)$	$O(n)$	$O(n^2)$
時空	$M_T^{N_0}$	$O(nT)$	-	$O(n+T)$
	M_T^N	$O(n^2T)$	-	$O(nT)$
層空	M_L^N	$O(n^2L+nT)$	$O(n)$	$O(n^2+nL)$

從表 5.3~5.6 結果可以發現在網路規模較小的兩種情境下，以時空網路為基礎建

構 $M_T^{N_0}$ 和 M_T^N 皆有較佳的求解效率，均可在極短時間內求得最佳解，然而在網路規模稍微變大時，M_S^N 及 M_L^N 皆無法在求解時限內求得最佳解，其 *Gap* 皆小於 2%。特別注意的是 *B-Gap* 非實際與最佳解的差距，實際的差距(*Optimality Gap*)可在更小。

表 5.3 節點需求型態旅行時間可忽略下的數學模式測試(求解效率)

n	*k*	*T*	*Num*	M_S^N		$M_T^{N_0}$		M_L^N		
				Num opt.	*CPU Time(s)*	*Num opt.*	*CPU Time(s)*	*L*	*Num opt.*	*CPU Time(s)*
12	8	16	3	3	0.10	3	0.03	2	3	0.15
	6	16	3	3	1.26	3	0.08	3	3	15.95
	4	24	3	0	-	3	0.02	4	0	-
24	8	24	3	0	-	3	0.02	4	0	-
	6	32	3	0	-	3	0.04	5	0	-
	4	48	3	0	-	3	0.05	7	0	-

註：- 表示該組網路的 3 個測試資料，模式皆無法在求解時間上限（7200 秒）內求得最佳解。

表 5.4 節點需求型態旅行時間可忽略下的數學模式測試(求解品質)

n	*k*	*T*	*Num*	M_S^N		$M_T^{N_0}$		M_L^N		
				B-Gap (%)	*Gap (%)*	*B-Gap (%)*	*Gap (%)*	*L*	*B-Gap (%)*	*Gap (%)*
12	8	16	3	0.00	0.00	0.00	0.00	2	0.00	0.00
	6	16	3	0.00	0.00	0.00	0.00	3	0.00	0.00
	4	24	3	1.10	0.01	0.00	0.00	4	1.62	0.03
24	8	24	3	2.27	0.10	0.00	0.00	4	2.35	0.03
	6	32	3	4.93	0.25	0.00	0.00	5	4.55	0.30
	4	48	3	6.96	0.52	0.00	0.00	7	7.97	1.30

註：*Gap* 以時空網路的模式求得最佳解為基準。

表 5.5 節點需求型態旅行時間不可忽略下的數學模式測試(求解效率)

n	*k*	*T*	*Num*	M_S^N		M_T^N		M_L^N		
				Num opt.	*CPU Time(s)*	*Num opt.*	*CPU Time(s)*	*L*	*Num opt.*	*CPU Time(s)*
12	8	30	3	3	0.10	3	0.03	2	3	0.15
	6	30	3	3	1.26	3	0.08	3	3	15.95
	4	45	3	3	1648.87	3	0.13	4	0	-
24	8	45	3	0	-	3	0.45	4	0	-
	6	60	3	0	-	3	0.69	5	0	-
	4	90	3	0	-	3	1.88	7	0	-

註：- 表示該組網路的 3 個測試資料，模式皆無法在求解時間上限（7200 秒）內求得最佳解。

表 5.6 節點需求型態旅行時間不可忽略下的數學模式測試(求解品質)

n	k	T	Num	M_S^N		M_T^N		M_L^N		
				B-Gap (%)	*Gap (%)*	*B-Gap (%)*	*Gap (%)*	L	*B-Gap (%)*	*Gap (%)*
12	8	30	3	0.00	0.00	0.00	0.00	2	0.00	0.00
	6	30	3	0.00	0.00	0.00	0.00	3	0.00	0.00
	4	45	3	0.01	0.00	0.00	0.00	4	0.92	0.00
24	8	45	3	2.04	0.14	0.00	0.00	4	2.10	0.00
	6	60	3	3.71	0.44	0.00	0.00	5	3.88	0.25
	4	90	3	8.02	1.55	0.00	0.00	7	7.72	1.52

註：*Gap* 以時空網路的模式求得最佳解為基準。

針對以時空網路為基礎建構之特例模式 $M_T^{N_0}$ 和一般模式 $M_T^{N_0}$ 的比較(表 5.7)，由於 $M_T^{N_0}$ 只能在旅行時間可忽略的情境下求解，因此只針對此情境進行測試，$M_T^{N_0}$ 的求解時間較 M_T^N 的表現好，當網路節點數變大或維修隊數變小時，$M_T^{N_0}$ 的表現仍可在即短時間內求解完畢；M_T^N 的求解時間則有較明顯的遞增趨勢，此結果也說明在旅行時間可忽略下更適合以 $M_T^{N_0}$ 來求解問題，在表 5.2 也可得到合理的驗證，因 $M_T^{N_0}$ 的變數及限制式的個數兩者都較 M_T^N 小。在 n=60, k=1 例子下，M_T^N 在 5 個測試例子中有 1 個未在 7200 秒求到最佳解，其例 *Gap* 值為 0.98%，但由於表中呈現 *Gap* 為所有測試例子的平均值，而呈現 0.98/5=0.20%。

表 5.7 節點需求型態以時空為基礎建構之數學模式比較

n	k	T	Num	$M_T^{N_0}$			M_T^N		
				Num opt.	*Gap (%)*	*CPU Time(s)*	*Num opt.*	*Gap (%)*	*CPU Time(s)*
12	3	32	5	5	0.00	0.01	5	0.00	0.06
	2	48	5	5	0.00	0.01	5	0.00	0.12
	1	96	5	5	0.00	0.04	5	0.00	0.56
24	3	64	5	5	0.00	0.05	5	0.00	1.24
	2	96	5	5	0.00	0.08	5	0.00	4.28
	1	192	5	5	0.00	0.17	5	0.00	24.22
36	3	96	5	5	0.00	0.11	5	0.00	11.39
	2	144	5	5	0.00	0.21	5	0.00	40.64
	1	288	5	5	0.00	0.59	5	0.00	288.87
48	3	128	5	5	0.00	0.26	5	0.00	77.21
	2	192	5	5	0.00	0.50	5	0.00	255.00
	1	384	5	5	0.00	1.08	5	0.00	1605.80
60	3	160	5	5	0.00	0.59	5	0.00	358.09
	2	240	5	5	0.00	0.68	5	0.00	956.67
	1	480	5	5	0.00	4.64	4	0.20	5438.58

註：*Gap* 以時空網路的特例模式求得最佳解為基準。

5.2.2 數學啟發式演算法測試

利用 M_T^N 與數學啟發式演算法在網路規模較大的例子進行比較，兩者方法皆以 A_{LD} 的解當初始解進行求解。數學啟發式演算法以修正之 A_{LD} 迭代次數 R =100、h=5 下產生限縮模式網路後，可套用之數學模式又可分為以節線及路徑為變數建構之數學模式 $\bar{M}_{L_A}^N$ 及 $\bar{M}_{L_P}^N$ 兩者進行比較。我們又再根據有無加入 M_T^N 之數組可行解建構限縮模式網路再進行測試，其中求模式可行解的時間上限設為 T_{MFS} =50($\lceil n/k \rceil$)。測試結果如表 5.8、5.9 所示，其中 *Num* 為該組網路測試資料數；*Num opt.*代表在該組網路測試資料中求得最佳解的次數；*CPU Time* 為數學模式平均求解時間，包含無法在求解時限內求得最佳解之例，其求解時間以求解時限 7200 (秒) 計入平均；*Gap* 以數學模式在求解時間上限（7200 秒）內求得的解為基準。

從表中可發現隨著節點數愈大或維修隊數愈小，M_T^N 在時限內求得最佳解的次數愈少，而數學啟發式演算法套用 $\bar{M}_{L_A}^N$ 及 $\bar{M}_{L_P}^N$ 均可很快在限縮模式網路下求得最佳解，且 $\bar{M}_{L_P}^N$ 的平均求解時間在大部分的例子測試下比 $\bar{M}_{L_A}^N$ 快。從 *B-Gap* 可看出 M_T^N 可求得非常接近最佳解的近似解(除最後一個例子外，*B-Gap* 皆小於 0.3%)，而從 *Gap* 可看出數學啟發式演算法無加入模式可行解的求解品質平均皆可落在 0.5%內，算是很接近求得之最佳解(或 M_T^N 求解 2 小時的近似解)；若再加入模式可行解可再進一步改善解，其 *Gap* 平均皆可落在 0.1%內，但其總體求解時間除了模式 $\bar{M}_{L_A}^N$ 或 $\bar{M}_{L_P}^N$ 的 *CPU Time* 外，需再加上為求 M_T^N 的可行解設定的時間 T_{MFS}，因此求解時間雖花費較久但再得較好的求解品質。

表 5.8 節點需求型態數學啟發式演算法測試(求解效率)

n	k	T	Num	數學模式 M_T^N		數學啟發式演算法 (無加入模式可行解) $\bar{M}_{L_A}^N$	$\bar{M}_{L_P}^N$	數學啟發式演算法 (有加入模式可行解) $\bar{M}_{L_A}^N$	$\bar{M}_{L_P}^N$
				Num opt.	*CPU Time(s)*	*CPU Time(s)*	*CPU Time(s)*	*CPU Time(s)*	*CPU Time(s)*
48	8	90	5	5	33.87	0.17	0.00	2.16	0.04
	6	120	5	5	100.77	0.26	0.00	4.14	0.07
	4	180	5	4	3513.78	0.40	0.00	5.92	0.12
72	8	135	5	5	439.88	0.40	0.00	346.43	144.57
	6	180	5	4	2912.23	1.20	0.11	2.94	0.56
	4	270	5	1	6273.74	1.21	0.05	11.72	1.79
96	8	180	5	3	3554.10	1.21	0.09	35.70	15.89
	6	240	5	2	6168.26	1.79	0.21	25.52	17.36
	4	360	5	2	6580.62	4.03	1.56	26.81	31.07

表 5.9 節點需求型態數學啟發式演算法測試(求解品質)

n	k	T	Num	數學模式		數學啟發式演算法 (無加入模式可行解)	數學啟發式演算法 (有加入模式可行解)
				Num opt.	*B-Gap (%)*	*Gap (%)*	*Gap (%)*
48	8	90	5	5	0.00	0.46	0.00
	6	120	5	5	0.00	0.34	0.00
	4	180	5	4	0.05	0.35	0.05
72	8	135	5	5	0.00	0.10	0.00
	6	180	5	4	0.04	0.12	0.00
	4	270	5	1	0.21	0.24	0.05
96	8	180	5	3	0.08	0.11	0.02
	6	240	5	2	0.14	0.04	0.01
	4	360	5	2	/	0.08	0.00

註1：/ 表示模式在求解時間上限（7200秒）內 還在執行LP relaxation的階段因而無法求得。
註2：*Gap*以數學模式在求解時間上限（7200秒）內求得的解為基準。

5.2.3 模擬退火法測試

以貪婪演算法產生的解當作模擬退火法的初始解，針對不同網路故障節點數及維修隊數，探討不同參數冷卻率 $\alpha \in \{0.90, 0.95, 0.99\}$、起始溫度 $T_0 \in \{2000, 5000, 8000\}$ 設定下的影響，結果如表 5.10 所示。每組網路測試資料設定為 5，其餘參數固定 $T_f = 10^{-5}$、$P_C = 1 - 1/n^2$。 表中對應的數值格式 A\B，其中 A 為 5 個測試資料下改善

貪婪演算法的解的次數，B 為其平均改善程度，若無改善則該例不予納入平均，以百分比(%)表示。舉例來說，3\0.19 代表在 5 個測資中有 3 個測資的模擬退火法有改善了貪婪演算法，而其 3 次的改善程度平均為 0.19%。由表可發現$\alpha = 0.99$的表現最佳，因其改善次數及改善程度較高，T_0設定的大小對不同網路大小及維修對數或冷卻率則無一定影響趨勢。

表 5.10 節點需求型態模擬退火法測試

		k=4			k=3			k=2		
n	α	2000	5000	8000	2000	5000	8000	2000	5000	8000
12	0.90	*2\1.19	2\0.46	5\1.26	1\2.42	4\1.56	2\0.17	0\0.00	0\0.00	1\0.26
	0.95	2\0.94	2\1.20	2\0.73	4\0.87	1\0.52	2\0.40	4\1.25	0\0.00	2\0.34
	0.99	5\3.25	4\4.56	2\2.90	1\2.93	3\4.48	5\3.47	3\5.54	2\9.27	5\3.12
24	0.90	0\0.00	0\0.00	0\0.00	0\0.00	0\0.00	0\0.00	0\0.00	0\0.00	0\0.00
	0.95	0\0.00	2\0.14	0\0.00	1\0.78	0\0.00	0\0.00	0\0.00	0\0.00	1\0.01
	0.99	3\0.92	3\0.70	2\0.48	2\1.67	3\0.18	4\1.16	2\0.46	2\0.36	3\0.33
36	0.90	0\0.00	0\0.00	0\0.00	0\0.00	0\0.00	0\0.00	0\0.00	0\0.00	0\0.00
	0.95	1\0.54	0\0.00	0\0.00	0\0.00	0\0.00	0\0.00	0\0.00	0\0.00	0\0.00
	0.99	3\0.69	0\0.00	3\0.19	1\0.09	2\0.14	0\0.00	0\0.00	0\0.00	0\0.00

註：*說明5個測試資料下改善貪婪演算法的解的次數為2，而改善例子中平均改善程度為1.19(%)。

5.3 起訖組合需求型態問題之測試

針對起訖組合需求型態問題，我們先比較可得精確解的數種整數規劃模式，再比較數學啟發式演算法的求解效率及求解品質。

表 5.11 起訖組合需求型態各整數規劃模式之大小估算

建構網路	數學模式	二元變數個數	實數變數個數	限制式個數
一般	M_S^A	$O(n^2T^2)$	$O(n)$	$O(n^2T^2)$
時空	$M_T^{A_0}$	$O(n^2T^2)$	-	$O(n^2T^2)$
	M_T^A	$O(n^2T^2)$	-	$O(n^2T^2)$
層空	M_L^A	$O(n^2L+n^2T^2)$	$O(n)$	$O(n^2T^2+nL)$

5.3.1 整數規劃模式比較

本小節比較本研究所提出可計算精確解之三種整數規劃模式，表 5.12 為其各模式之大小估算，可發現每個模式的二元變數及限制式的個數以 *big-O* 估算均一致，因 L 通常小於 T，故也可將 M_L^A 二元變數個數 $O(n^2L+n^2T^2)$ 視為 $O(n^2T^2)$ 而限制式個數 $O(n^2T^2+nL)$ 視為 $O(n^2T^2)$。從表 5.11 與表 5.2，可看出兩者需求型態問題的模式大小差別，隨著節點數 n 增加或維修隊數 k 減少，起訖組合需求型態模式的大小會驟增非常快速（相較於節點需求型態模式而言）。

以小規模網路測試模式 M_S^A 、$M_T^{A_0}$ 、M_L^A 及模式 M_S^A 、M_T^A 、M_L^A 的求解表現，我們發現所有模式皆無法在 2 小時內求得最佳解，且模式求解的過程中 *Gap* 收斂極為緩慢，符合整數規劃模式很難在短時間內求得最佳解的預期表現。

5.3.2 數學啟發式演算法測試

利用 M_T^A 與數學啟發式演算法在網路規模較小的例子比較，兩者方法皆以 A_{LD} 的解當初始解進行求解。數學啟發式演算法以修正之 A_{LD} 迭代次數 R =100 下產生限縮模式網路後，可套用之數學模式又分為以節線及路徑為變數建構之數學模式 $\bar{M}_{L_A}^A$ 及 $\bar{M}_{L_P}^A$ 兩者進行比較，而根據有無加入 M_T^A 之數組可行解建構限縮模式網路再進行測試，其中求模式可行解的時間上限設為 T_{MFS}=100($\lceil n/k \rceil$)，測試結果如表 5.12 所示，其中 *CPU Time* 為數學模式求解時間，*Gap* 為以數學模式在求解時間上限（7200 秒）內求得的解為基準。

表 5.12 起訖組合需求型態數學啟發式演算法測試

n	k	T	數學模式 M_T^A B-Gap (%)	數學模式 M_T^A CPU Time(s)	數學啟發式演算法 (無加入模式可行解) Gap (%)	$\bar{M}_{L_A}^A$ CPU Time(s)	$\bar{M}_{L_P}^A$ CPU Time(s)	數學啟發式演算法 (有加入模式可行解) Gap (%)	$\bar{M}_{L_A}^A$ CPU Time(s)	$\bar{M}_{L_P}^A$ CPU Time(s)
12	4	45	686	-	0.00	1547.52	0.01	0.00	1535.97	0.01
	3	60	1110.16	-	0.00	1604.22	0.01	0.00	1589.36	0.01
	2	90	2641.87	-	1.59	-	0.00	0.00	-	0.00
24	4	90	/	-	0.00	-	0.01	0.00	-	0.01
	3	120	/	-	0.00	-	0.01	0.00	-	0.01
	2	180	/	-	0.00	-	0.00	0.00	-	0.01

註1：- 表示模式皆無法在求解時間上限（7200秒）內求得最佳解。

註2：/ 表示模式在求解時間上限（7200秒）內 還在執行 LP relaxation 的階段因而無法求得。

註3：*Gap* 以數學模式在求解時間上限（7200秒）內求得的解為基準。

就以結果來看，數學啟發式演算法以 $\bar{M}_{L_P}^A$ 的求解效率明顯較 $\bar{M}_{L_A}^A$ 好，因此建議以 $\bar{M}_{L_P}^A$ 來求解問題，而有加入模式可行解只有在 n=12 及 k=2 的例子測試有再改善解，然而起訖組合需求的問題大多無法使用模式求的最佳解(因收斂過於緩慢，*B-Gap* 在 7200 秒內仍然極大，但並不是模式的解與最佳解實際的差距)，因數學啟發式演算法 *Gap* 是以數學模式的解為基準，而難以得知求解品質，但至少能在短時間內得到數學模式求解 7200 秒一樣好的解。

5.3.3 模擬退火法測試

以貪婪演算法產生的解當作模擬退火法的初始解，針對不同網路故障節點數及維修隊數，探討不同參數冷卻率 $\alpha \in \{0.90, 0.95, 0.99\}$、起始溫度 $T_0 \in \{2000, 5000, 8000\}$ 設定下的影響，結果如表 5.13 所示。每組網路測試資料設定為 5，其餘參數固定 $T_f = 10^{-5}$、$P_C = 1 - 1/n^2$，表中對應的數值有兩個，前者為五個測試資料下改善貪婪演算法的解的次數；後者則為平均改善程度，若無改善則該例不予納入平均，以百分比(%)表示。由表可發現 $\alpha = 0.99$ 的表現最佳，改善次數及改善程度較高，T_0 設定的大小對不同網路大小及維修對數或冷卻率則無一定影響趨勢。

表 5.13 起訖組合需求型態模擬退火法測試

		k=4			k=3			k=2		
n	α	2000	5000	8000	2000	5000	8000	2000	5000	8000
12	0.90	*2\3.29	2\5.55	1\0.15	0\0.00	0\0.00	1\0.51	1\0.56	1\0.00	3\0.23
	0.95	1\6.55	3\4.69	1\0.50	1\0.16	2\4.61	3\7.28	4\4.67	1\0.15	3\0.49
	0.99	4\3.63	4\6.68	2\7.72	2\5.38	2\11.5	5\7.03	4\12.8	5\10.6	5\5.61
24	0.90	0\0.00	0\0.00	0\0.00	0\0.00	0\0.00	0\0.00	3\0.92	0\0.00	0\0.00
	0.95	0\0.00	2\0.36	0\0.00	0\0.00	2\0.10	1\0.24	2\0.05	0\0.00	1\1.93
	0.99	2\0.21	2\1.28	4\1.05	3\0.39	3\0.23	2\1.71	3\0.28	3\1.79	3\0.86
36	0.90	2\0.45	0\0.00	0\0.00	0\0.00	0\0.00	0\0.00	0\0.00	0\0.00	0\0.00
	0.95	0\0.00	1\0.01	0\0.00	0\0.00	0\0.00	0\0.00	0\0.00	0\0.00	0\0.00
	0.99	0\0.00	2\0.58	3\0.05	1\0.07	3\0.03	3\0.37	2\1.15	0\0.00	1\1.08

註：*說明5個測試資料下改善貪婪演算法的解的次數為2，而改善例子中平均改善程度為3.29(%)。

5.4 小結

一開始先介紹測試網路的參數設定，以 *big-O* 計算各數學模式變數及限制式的個數來估算模式的大小，接著分別針對節點需求型態問題以及起訖組合需求型態問題測試本研究提出之數種數學規劃模式以及數學啟發式演算法。

節點需求型態問題較適用以時空網路為基礎建構之模式 $M_T^{N_0}$ 及 M_T^N，因其求解效率遠比 M_S^N 、M_L^N 好，而 $M_T^{N_0}$ 雖只適用於旅行時間可忽略的情境下，其求解時間卻比 M_T^N 來的更有效率，因此在旅行時間可忽略下建議使用 $M_T^{N_0}$，而在旅行時間不可忽略下則建議使用 M_T^N;在數學啟發式演算法測試中，以 A_{LD} 的解當模式初始解，套用至 $\bar{M}_{L_A}^N$ 或 $\bar{M}_{L_P}^N$ 與 M_T^N 的求解效率相比來得好，而在大部分例子測試中 $\bar{M}_{L_P}^N$ 求解時間又比 $\bar{M}_{L_A}^N$ 快，就求解品質來說，無加入模式可行解已與近似最佳解(*B-Gap*<0.3%)相距不大(*Gap*<0.5%)，而加入模式可行解又可在進一步改善目標值，可得 *Gap*<0.1%。

在起訖組合需求型態問題，在網路規模小的例子中，模式的變數及限制式個數即很大，故在網路規模小的例子中皆無法在時間限制內求得最佳解;在數學啟發式演算法測試中，以 A_{LD} 的解當模式初始解，$\bar{M}_{L_P}^A$ 可在極短時間內求解完畢，明顯比 $\bar{M}_{L_A}^A$ 求解效率好，然而就測試結果而言，加入模式可行解大多無法再進一步改善。

第六章 結論及未來研究

6.1 結論

當人為或天災等事故發生，可能導致服務網路的節點發生大規模故障，本研究探討如何有效地修復毀損之節點，以使修復決策能滿足的節點需求愈大愈好。節點需求依不同服務系統網路又可再分為「單一節點」與「起訖組合」等兩種需求型態，為更符合實務，我們假設節點的服務需求量通常會依時變化（亦即不同時段的需求量有所不同），且具即時性（亦即需求不會等待，當下未滿足的需求，將一律隨即被視為流失），說明這個排程問題與文獻中的 *k*-TRP 更符合實際，並更具挑戰性與困難度。以下臚列本論文具體貢獻，並於 6.2 小節建議未來可嘗試的研究議題：

具體貢獻：

1. 過去文獻經常將節點需求視為一次性的固定常數，而本研究考慮需求會依時變動，且為首度考量「起訖組合」需求的研究，更符合現實情況，無論在問題設定、數學規劃模式與演算法設計等，皆為首創。(節點:第 3 章、起訖組合: 第 4 章)
2. 以 *k*-TRP 之數學模式修改而得「含排除子迴圈限制式」之模式，但此方法求解耗時甚久，不建議採用。(節點:第 3.3 節、起訖組合: 第 4.3 節)
3. 提出「以時空網路建構」之模式，利用 Time-Space 的方式展開網路，時空節點代表某一時刻下的某個節點，以時空節點為變數決定此刻開始修復與否。當節點間的旅行時間還小於其修復時間時，我們特別發展的數學模式可以更有效地求解最佳排程；而在必須考量節點間移動時間的一般模式，則以時空節線表示維修隊移動之路徑，以時空節線為變數決定維修隊在此時刻下經過與否。數值測試結果證實當移動時間可忽略下，特例模式求解效率優於一般模式。此方法可大幅縮減求解時間，但受修復完工上限 T 影響，因此若時段切割過細(T 過大)，將導致展開時空網路圖過大而難以有效求解。(節點:第 3.4 節、起訖組合: 第 4.4 節)

4. 提出「以層空網路建構」之模式，利用 Level-Space 的方式展開網路，階層代表節點的修復順序，而以階層節線表示維修隊移動之路徑。此模式優點在於所展開之網路較時空展開網路小，但因需轉換節點的修復完工時間，導致求解效率不佳且受修復節點上限數 L 影響。(節點:第 3.5 節 、起訖組合: 第 4.5 節)
5. 提出數學啟發式演算法(Matheuristics)：結合貪婪演算法與整數規劃模式，首先產生「限縮層空網路」來加速模式的求解時間：以「優先修復最小損失需求的節點」的貪婪機制，選取在維修隊修復節點的期間而導致其他節點未能滿足需求量最小之節點來修復，同時考量維修時間、旅行時間及依時變化需求。接著以限縮模式網路下，提出兩種數學模式:(1)以節線形式(arc-form):即以層空網路之節線建構，由於求解時間仍耗費甚久;(2)以路徑形式(path-form):以深度優先搜尋法找出所有層空網路可行之修復路徑，並以路徑為決策變數來建構模式。數值測試結果顯示，單一節點需求類型的問題可在短時間內求得一品質還不錯近似最佳解。(節點:第 3.6 節 、起訖組合: 第 4.6 節)
6. 客製化「模擬退火法」: 具跳脫區域最佳解的機制，藉由修改接受準則以適用於本研究問題，以貪婪演算法求得的解當初始解，此方法能在短時間內得到近似最佳解，且為處理起訖組合需求類型問題的最佳建議作法。(節點:第 3.7 節 、起訖組合: 第 4.7 節)

6.2 未來研究

關於本研究網路節點修復排程問題，以下列出未來研究可再深入探討的問題如下:

1. **可持續精進及發展本研究之演算法來產生限縮網路**

層空網路產生限縮節線的方式設計，如何加入新的階層節線，可持續改善模式當下之近似解？或是在無可行解情況下應加入那些階層節線才會有可行解？另外，可嘗試設計針對起訖組合需求型態問題之專屬演算法；目前本研究皆直接套用節點需求型

態之求解方法，在起訖組合網路節點修復問題中還有很大改進空間。

在貪婪演算法的機制設計方面，或可嘗試以下提出的兩種不同貪婪機制：

A. 「選擇較短的節線」之貪婪演算法

貪婪機制

每個節點 i 選其前 l 短節線旅行時間 d_{ij} 的連出節線 (outgoing arc)。

流程說明

圖 6.1 明縮模式網路前後之示意圖，右圖為 l=2 限縮的結果，對於除虛擬起訖階層節點外的每個階層節點 $v_{i,m}, m=1,...,L-1$，我們只考慮其最好的 l 條流出節線，讓每個節點只考慮選擇足夠短的節線來移動。該注意的是，若 l 太小有可能會找不到可行解，但若 l 太大又會使求解時間過久。

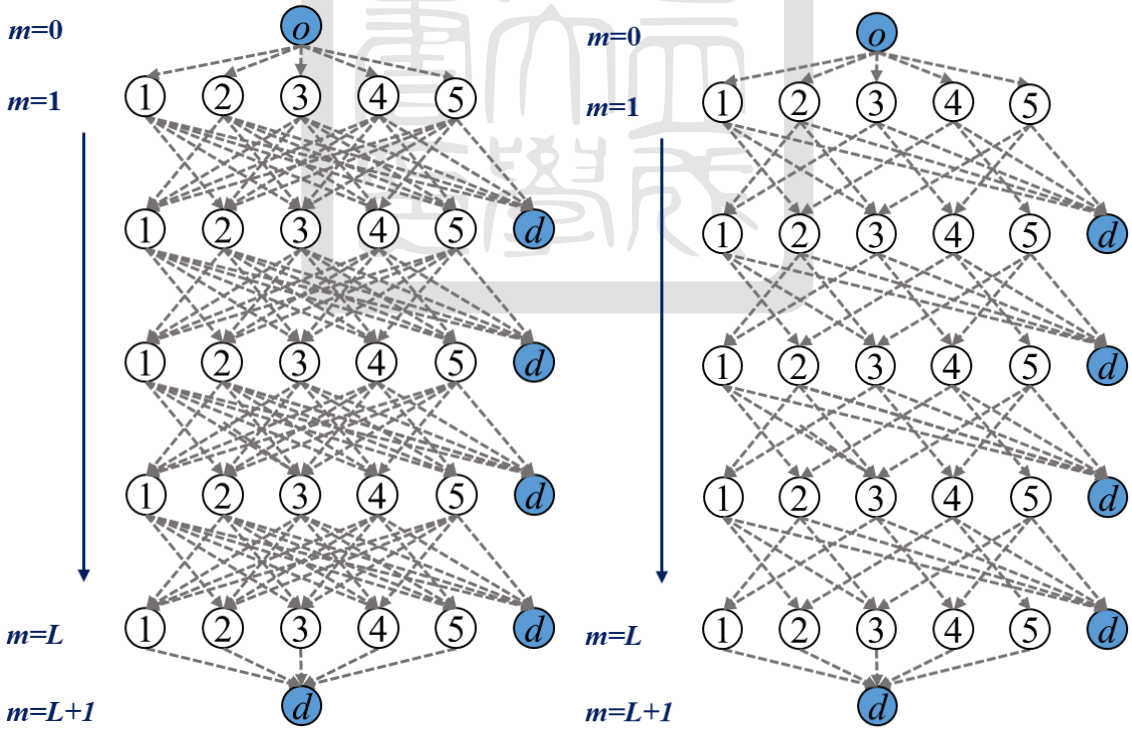

圖 6.1 以演算法 A_{SA} 限縮模式網路前後之示意圖

B. 「節點依需求分群」之貪婪演算法

貪婪機制

將節點依需求分三群(Node Cluster)：

(a)第一群：規劃時間 0~$T/3$ 內需求較大之前 $n/3$ 個節點；

(b)第三群：規劃時間 $2T/3$~T 內需求較大前 $n/3$ 個節點(排除第一群) ；

(c)第二群：排除第一、三群剩餘的 $n/3$ 個節點；

流程說明

以上面分群方式將節點劃分三群，第一群為一開始較需緊急修復的節點集合，因修復後可獲得較大的需求量，而第三群則是反過來以後面較需緊急修復的節點集合。以圖 6.2 說明節點分群之層空網路示意圖，前 $L/3$ 階層將減少到其第三群節點的階層節線，後 $L/3$ 階層則減少到其第一群節點的階層節線，以此方式排除一開始修修復第三群的節點及限制第一群的節點需較早進行修復，圖 6.3 為限縮前後的示意圖。此限縮機制縮減的階層節數有限，若要再進一步減少模式網路，可以選擇較短的節線之方法再限制階層節點連出的節線數。注意的是此分群方式受需完工上限 T 及維修隊修復的上限數 L 大小影響，因此準確估 T 及 L 值可助於提高求解的品質。

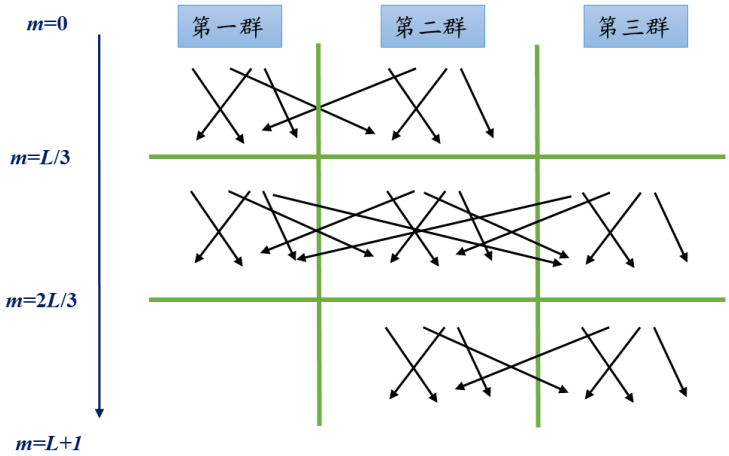

圖 6.2 節點分群之層空網路示意圖

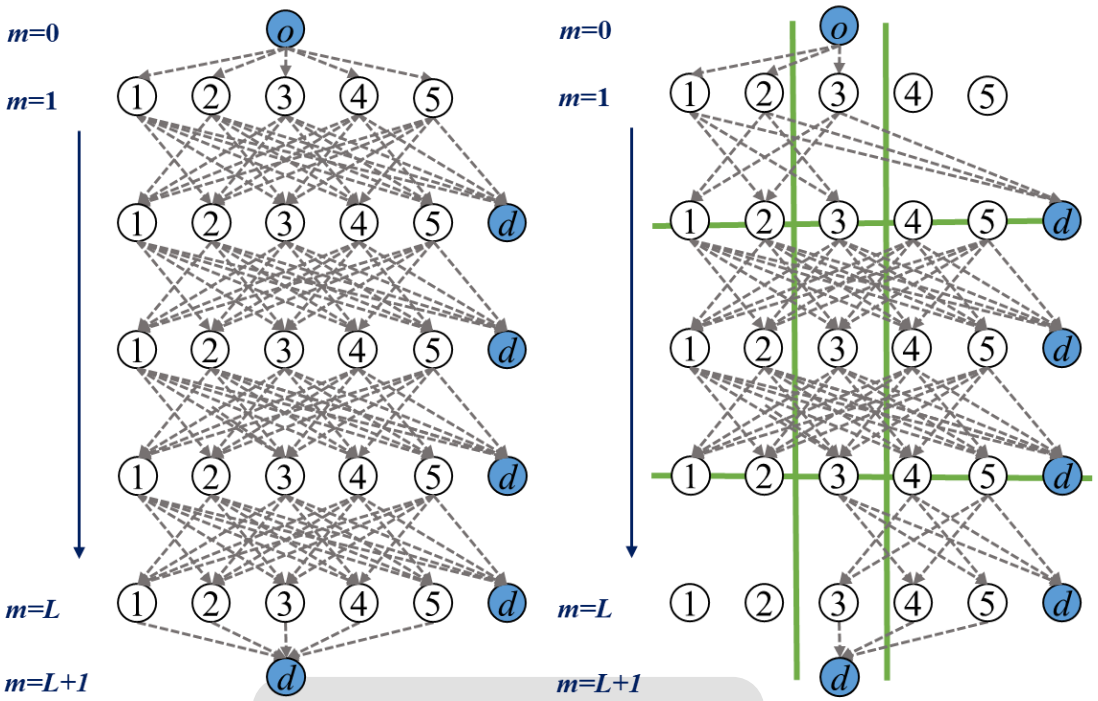

圖 6.3 以演算法 A_{SA} 限縮模式網路前後之示意圖

2. 考慮多組維修隊合作之修復排程研究

以 YouBike 大當機事件維修為例說明，若當初興建租借站時架設較多的停車柱，代表此租借站借還自行車之需求量較大，若仍限制只由單組維修隊來修復，將導致修復時間過長而未能滿足大量的需求量。因此在緊急情況下，為了滿足某節點或某起訖點組合預期即將發生的龐大需求量，應可考慮多組維修隊合作修復，以加速完成節點的修復工作，這方面的作法或可參考不同工作模式(mode)的有限資源專案排程問題（Resource Constrained Project Scheduling Problem，RCPSP）相關文獻。

3. 考慮維修隊工作能力不同之修復排程研究

在現實中各組維修隊所具有的人力、機具等資源並不一定相同，也因此所需修復的節點維修時間因而不同，在未來研究可將此因素列入考量而更能貼近真實的狀況。同樣地，方面的作法亦可參考 RCPSP 相關文獻。

4. 考慮到鄰近正常節點滿足需求之修復排程研究

以 YouBike 服務系統為例，若欲借車或還車的租借站尚未修復前，有部分民眾會願意犧牲一些時間到鄰近可正常營運的場站借車或還車。亦即，原依時變化之服務需求如何考慮使用者行為而調整，在未來研究或可持續探討此因素的影響而更能貼近真實的狀況。

參考文獻

Averbakh, I. (2012). Emergency path restoration problems. *Discrete Optimization*, *9*(1), 58-64.

Angel-Bello, F., Alvarez, A., & García, I. (2013). Two improved formulations for the minimum latency problem. *Applied Mathematical Modelling*, *37*(4), 2257-2266.

Angel-Bello, F., Cardona-Valdés, Y., & Álvarez, A. (2017). Mixed integer formulations for the multiple minimum latency problem. Operational Research, 1-30.

Baez, S., Angel-Bello, F., & Alvarez, A. (2016). Time-dependent formulations for minimizing total completion time in a parallel machine scheduling problem with dependent setup times. *IFAC-PapersOnLine*, *49*(12), 857-862.

Bektas, T. (2006). The multiple traveling salesman problem: an overview of formulations and solution procedures. *Omega, 34*(3), 209-219.

Dewilde, T., Cattrysse, D., Coene, S., Spieksma, F. C., & Vansteenwegen, P. (2013). Heuristics for the traveling repairman problem with profits. *Computers & Operations Research*, *40*(7), 1700-1707.

Fakcharoenphol, J., Harrelson, C., & Rao, S. (2003). The k-traveling repairman problem. In *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms* (pp. 655-664). Society for Industrial and Applied Mathematics.

Jothi, R., & Raghavachari, B. (2007). Approximating the k-traveling repairman problem with repairtimes. *Journal of Discrete Algorithms*, *5*(2), 293-303.

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *science*, *220*(4598), 671-680.

Matisziw, T. C., Murray, A. T., & Grubesic, T. H. (2010). Strategic network restoration. *Networks and Spatial Economics*, *10*(3), 345-361.

Luo, Z., Qin, H., & Lim, A. (2014). Branch-and-price-and-cut for the multiple traveling repairman problem with distance constraints. *European Journal of Operational Research, 234*(1), 49-60.

Nucamendi-Guillén, S., Martínez-Salazar, I., Angel-Bello, F., & Moreno-Vega, J. M. (2016). A mixed integer formulation and an efficient metaheuristic procedure for the k-Travelling Repairmen Problem. *Journal of the Operational Research Society, 67*(8), 1121-1134.

Sahni, S., & Gonzalez, T. (1976). P-complete approximation problems. *Journal of the ACM (JACM)*, *23*(3), 555-565.

Xu, N., Guikema, S. D., Davidson, R. A., Nozick, L. K., Çağnan, Z., & Vaziri, K. (2007). Optimizing scheduling of post−earthquake electric power restoration tasks. *Earthquake engineering & structural dynamics*, *36*(2), 265-284.