

國立政治大學資訊科學系  
Department of Computer Science  
National Chengchi University

碩士論文

Master's Thesis

在機會網路上實現行動代理人之搜尋機制  
Mobile Agent Tracking Technology over Opportunistic  
Network

研究生：林怡萱

指導教授：連耀南

中華民國九十九年六月

June 2010

在機會網路上實現行動代理人之搜尋機制

Mobile Agent Tracking Technology over Opportunistic Network

研究生：林怡萱

Student：Yi-Shiuan Lin

指導教授：連耀南

Advisor：Yao-Nan Lien

國立政治大學

資訊科學系

碩士論文

A Thesis

submitted to Department of Computer Science

National Chengchi University

in partial fulfillment of the Requirements

for the degree of

Master

in

Computer Science

中華民國九十九年六月

June 2010

# 在機會網路上實現行動代理人之搜尋機制

## 摘要

在機會網路上，傳遞資料遠比一般網路困難。運用行動代理人於機會網路上，可大幅提升機會網路的功能，其中的行動代理人平台，需要一個代理人的搜尋機制，方能實現對代理人的控制。本篇論文以「山文誌登山資訊系統」為背景，探討行動代理人在機會網路上的應用。在此登山客追蹤與環境監控系統中，每個登山客都攜帶一個配備有 GPS 功能的小型設備，此設備會在登山客的移動過程中，自動收集位置資訊，並使用短程無線電來和鄰近的登山客交換彼此的資訊。由於行動代理人是附屬於登山客所攜帶的設備上，而且當靠近其他登山客時，才可藉此轉移到另一個設備，故其行動力受制於登山客的移動行為。因登山客行走速率差異不大、所在位置分佈不均，使得行動代理人的轉移極為緩慢且需仰賴不確定的碰面機會，導致行動代理人之搜尋效率極低，無法支援緊急任務。為此，我們提出控制網路的方法，及一個簡單的搜尋演算法，讓搜尋代理人藉此高速網路來快速移動，以提高目標代理人的搜尋效率。在不同的考量及目標下，我們提出幾個控制網路的建置模型，並在證明其為 NP-complete 問題後，提出有效的啟發式演算法來解決此控制點選擇問題。

# Mobile Agent Tracking Technology over Opportunistic Network

## Abstract

Transmitting data on an opportunistic network is much more difficult than that on a general network. The communication capability of an opportunistic network can be greatly enhanced via mobile agent functionality. A mobile agent platform demands a search mechanism to locate and control its agents. In this thesis, we investigate the adoption of mobile agent to opportunistic networks using "CenWits" system as reference model. In CenWits system, each hiker carries a GPS enabled sensor node to collect and exchange movement statistics with its fellow hikers using a short range wireless links. Since mobile agents will be attached to the sensor nodes that hikers carry with, mobile agents can hop from one host to another only when two hosts (hikers) meet together such that their mobility is restrained by the moving behavior of hikers. The little difference in walking speed and the uneven distribution of hikers make the hopping of mobile agents extremely slow and opportunistic. As a consequence, the search of mobile agents is slow and inefficient crippling agility of urgent agent functionality. Therefore, we propose to construct a control network using high speed network for search agents to travel in high speed. Under different objectives and constraints, we propose several control point placement models. After proving them to be NP-Complete, we propose few efficient heuristic algorithms to solve the placement problem. We also propose a simple search algorithm for search agents to search target agents quickly by using a control network.

## 誌謝辭

在我的碩士生涯中，並不如大部份的人那麼平順，因為家中經濟狀況每況愈下，在沒辦法負擔學、宿費與生活費的情況下，從碩二起工讀的時間就長達三年，也影響了論文的進度，延長了畢業的時程。

碩一、二的生活是充滿好奇的，感謝系上開設許多不同領域的課程與 Seminar，讓我們能接觸到各方面的知識技術，也感謝校內不定期舉辦的活動，讓我的課外生活多姿多采，感謝和政、宗銘、逸帆、明翰、永全諸位學長，帶我更快認識系上的行政工作與環境，更感謝他們總是讓研究室充滿歡樂。

在碩三、四正式進入論文研究後，我開始遇到了大大小小的瓶頸，非常感謝連耀南教授總是能藉由討論與提問，想辦法讓我思考得更廣、更深入，尤其針對我的表達能力方面，給了我許多幫助，也在我的論文內容偏離方向時，及時幫助我導正，此外，感謝連耀南教授與陳恭教授，為我許多課堂上不清楚之處解惑，讓我對有興趣的課程更積極學習，感謝威中同學讓我在研究室為論文奮戰、心情煩悶時，有一個能傾吐的對象，感謝立誠、育晟學弟的認真努力，讓我有了學習的目標，感謝玉潔、智杰、彥嵩、筱慈等學弟妹為研究室舒緩壓抑的氣氛，感謝原民中心的同事們一意欣、玫秀、佳凌、嘉玲，總讓我的工作環境充滿歡笑，也感謝原民中心的黃季平老師與林修澈主任，因為對我專業技術的信任，針對資訊相關的問題時常聆聽我的意見，並額外給予我其他工作，讓我在這段期間無須擔心經濟問題。

當研究時程愈拉愈長，原本個性樂觀、積極的自己，曾一度對自己產生極度失望、無力、懷疑等負面的情緒，幸而有淳雯、幸慧、鳳卿、美華幾位好友有形與無形的支持，以及智賢學長、諭祺學弟的鼓勵與意見，加上男朋友舜斌及其家人在精神上、物質上的幫助，若沒有你們大家，我想我沒有辦法挺過這一關，感謝你們。

最後，我要感謝我的家人，謝謝你們的包容，讓我能以對自己負責的態度與堅持，來完成這份論文。

## 目錄

第 1 章 緒論 .....	1
1.1 機會網路 .....	2
1.1.1 機會網路的訊息交換機制 .....	3
1.1.2 機會網路的研究議題 .....	4
1.1.3 機會網路的應用 .....	6
1.2 行動代理人 .....	8
1.2.1 行動代理人的特點 .....	9
1.2.2 行動代理人的優點 .....	10
1.2.3 行動代理人的應用 .....	11
1.3 運用行動代理人於機會網路 .....	12
1.3.1 機會網路上的行動代理人運作 .....	13
1.3.2 行動代理人平台的功能需求 .....	13
1.4 論文組織架構 .....	14
第 2 章 相關研究與挑戰 .....	16
2.1 搜尋行動代理人之相關研究 .....	16
2.1.1 過去的搜尋行動代理人之方法 .....	17
2.1.2 機會網路上搜尋行動代理人之挑戰 .....	19
2.2 「山文誌登山資訊系統」簡介 .....	19
第 3 章 機會網路上行動代理人之搜尋 .....	23
3.1 控制網路概念 .....	24
3.2 利用控制網路的搜尋策略 .....	25
3.3 控制網路的建置 .....	26
3.3.1 環境假設 .....	26

3.3.2	設計考量與目標.....	27
3.4	控制點選擇問題—最大總流量模型 .....	28
3.4.1	最大總流量的數學模型 .....	28
3.4.2	最大總流量模型的缺點 .....	29
3.5	控制點選擇問題—最大涵蓋率模型 .....	30
3.5.1	最大涵蓋率的數學模型 .....	31
3.5.2	最大涵蓋率模型的缺點 .....	32
3.6	控制點選擇問題—最大加權涵蓋率模型 .....	33
3.6.1	最大加權涵蓋率的數學模型 .....	33
第 4 章	問題分析與演算法 .....	35
4.1	數學模型之問題分析 .....	35
4.1.1	問題分析—最大總流量模型 .....	35
4.1.2	問題分析—最大涵蓋率模型 .....	36
4.2	解決方案的演算法 .....	38
4.2.1	CPSP-Flow 解決方案 .....	40
4.2.2	CPSP-Coverage 解決方案 .....	44
4.2.3	CPSP-Utility 解決方案 .....	52
第 5 章	效能評估 .....	58
5.1	小型問題效能評估 .....	58
5.1.1	實驗環境的參數設定 .....	58
5.1.2	實驗結果與分析.....	58
5.2	大型問題效能評估 .....	66
5.2.1	實驗環境的參數設定—sensitivity to the no. of nodes.....	67
5.2.2	實驗結果與分析—sensitivity to the no. of nodes .....	67

5.2.3	實驗環境的參數設定—sensitivity to the no. of trails .....	73
5.2.4	實驗結果與分析—sensitivity to the no. of trails .....	73
第 6 章	結語 .....	79
參考文獻	.....	81





## 表目錄

表 4.1：控制點選擇問題 (CPSP) 的評估指標。	39
表 4.2：Node Potential First (NPF) 演算法之參數表。	44
表 5.1：CPSP-Coverage—NPF vs. Optimal solution (Error)。	64
表 5.2：CPSP-Utility—NPF-U vs. Optimal solution (Error)。	66
表 5.3：啟發式演算法評估 (sensitivity to the no. of nodes)。	72
表 5.4：啟發式演算法評估 (sensitivity to the no. of trails)。	78



## 圖目錄

圖 1.1：Disconnected Mobile Ad Hoc Network (Disconnected MANET)。	4
圖 3.1：在機會網路上搜尋行動代理人 (以「山文誌登山資訊系統」應用情境為例)。	24
圖 3.2：使用控制網路來輔助行動代理人的搜尋。	25
圖 3.3：最大總流量模型下，可能的控制點建置狀況。	30
圖 3.4：最大涵蓋率模型改善控制點之分佈。	31
圖 4.1：登山健行路徑範例。	41
圖 4.2：CPSP-Flow 模型求 sub-optimal 之範例—初始狀態。	42
圖 4.3：CPSP-Flow 模型—sub-optimal solution vs. optimal solution。	43
圖 4.4：Node Potential First (NPF) 演算法之運作流程。	45
圖 4.5：CPSP-Coverage 模型求 sub-optimal 之範例—初始狀態。	46
圖 4.6：CPSP-Coverage 模型求 sub-optimal solution—第一輪比較後，選 B 為控制點。	47
圖 4.7：CPSP-Coverage 模型求 sub-optimal solution—第二輪比較後，選 G 為控制點。	48
圖 4.8：CPSP-Coverage 模型求 sub-optimal solution—第三輪比較後，選 H 為控制點。	49
圖 4.9：CPSP-Coverage 模型求 sub-optimal solution—第四輪比較後，選 I 為控制點。	50
圖 4.10：CPSP-Coverage 模型求 sub-optimal solution—第五輪比較後，不選任何點。	50
圖 4.11：CPSP-Coverage 模型—sub-optimal solution vs. optimal solution。	51
圖 4.12：CPSP-Utility 模型求 sub-optimal solution—初始狀態。	52

圖 4.13：CPSP-Utility 模型求 sub-optimal solution—第一輪比較後，選 B 為控制點。	53
圖 4.14：CPSP-Utility 模型求 sub-optimal solution—第二輪比較後，選 G 為控制點。	54
圖 4.15：CPSP-Utility 模型求 sub-optimal solution—第三輪比較後，選 H 為控制點。	55
圖 4.16：CPSP-Utility 模型求 sub-optimal solution—第四輪比較後，選 I 為控制點。	56
圖 4.17：CPSP-Utility 模型求 sub-optimal solution—第五輪比較後，不選任何點。	56
圖 4.18：CPSP-Utility 模型—sub-optimal solution vs. optimal solution。	57
圖 5.1：CPSP-Coverage—NPF vs. Optimal solution (no. of covered edges)。	59
圖 5.2：CPSP-Coverage—NPF vs. Optimal solution (mean_MC_distance)。	59
圖 5.3：CPSP-Utility—NPF-U vs. Optimal solution (total weight of covered edges)。	60
圖 5.4：CPSP-Utility—NPF-U vs. Optimal solution (mean_weighted_MC_distance)。	61
圖 5.5：CPSP-Coverage—NPF vs. Optimal solution (Error in covered edges)。	63
圖 5.6：CPSP-Coverage—NPF vs. Optimal solution (Error in mean_MC_distance)。	63
圖 5.7：CPSP-Utility—NPF-U vs. Optimal solution (Error in total weight of covered edges)。	65
圖 5.8：CPSP-Utility—NPF-U vs. Optimal solution (Error in mean_weighted_MC_distance)。	65

圖 5.9：啟發式演算法評估 (Avg. number of covered edges) (sensitivity to the no. of nodes)。	68
圖 5.10：啟發式演算法評估 (Avg. total weight of covered edges) (sensitivity to the no. of nodes)。	69
圖 5.11：啟發式演算法評估 (Avg. mean_MC_distance) (sensitivity to the no. of nodes)。	69
圖 5.12：啟發式演算法評估 (Avg. mean_weighted_MC_distance) (sensitivity to the no. of nodes)。	70
圖 5.13：啟發式演算法評估 (Avg. max_MC_distance) (sensitivity to the no. of nodes)。	70
圖 5.14：啟發式演算法評估 (Avg. max_weighted_MC_distance) (sensitivity to the no. of nodes)。	71
圖 5.15：啟發式演算法評估 (Avg. number of covered edges) (sensitivity to the no. of trails)。	74
圖 5.16：啟發式演算法評估 (Avg. total weight of covered edges) (sensitivity to the no. of trails)。	75
圖 5.17：啟發式演算法評估 (Avg. mean_MC_distance) (sensitivity to the no. of trails)。	75
圖 5.18：啟發式演算法評估 (Avg. mean_weighted_MC_distance) (sensitivity to the no. of trails)。	76
圖 5.19：啟發式演算法評估 (Avg. max_MC_distance) (sensitivity to the no. of trails)。	76
圖 5.20：啟發式演算法評估 (Avg. max_weighted_MC_distance) (sensitivity to the no. of trails)。	77

# 第1章

## 緒論

本研究將運用行動代理人技術於機會網路中，以增強其功能與應用。在機會網路上，傳遞資料遠比一般網路困難。運用行動代理人於機會網路上，可大幅提升機會網路的功能，其中的行動代理人平台，需要一個代理人的搜尋機制，方能實現對代理人的控制，例如：暫停、恢復、終止代理人的行為。我們將探討行動代理人平台的需求，並指出為了實現代理人之控制，所須面對的行動代理人搜尋機制之重要性與問題。

本篇論文以「山文誌登山資訊系統」為背景，探討行動代理人在機會網路上的應用。在此登山客追蹤與環境監控系統中，每個登山客都攜帶一個配備有 GPS 功能的小型設備，此設備會在登山客的移動過程中，自動收集位置資訊，並使用短程無線電來和鄰近的登山客交換彼此的移動資訊。由於行動代理人是附屬於登山客所攜帶的設備上，又當兩個登山客互相接近時，行動代理人才可以從其中一個設備轉移到另一個設備，故其行動力受制於登山客的移動行為，使行動代理人的轉移極為緩慢且需仰賴不確定的碰面機會，導致行動代理人之搜尋效率極低，無法支援緊急任務，例如：通知一個團隊成員惡劣天氣的變化。

在過去，雖然已有許多行動代理人的搜尋方法被提出，但由於大多應用在有線網路的環境中，或機會網路的環境特性有很大的差異，故無法將原有的方法直接使用在機會網路環境中，因此，我們提出一個內部能快速交換資訊的高速網路概念（在本論文中，稱為控制網路），及一個利用控制網路來輔助的行動代理人搜尋策略，來讓行動代理人藉此控制網路的轉送，以快速接近並搜尋到特定的行動代理人。

上述之控制網路的建置方法，需依照不同環境背景的考量來設計，本論文將以「山文誌登山資訊系統 (CenWits System)」為例，在此山區應用的環境中，藉由探討各種面向的建置考量與目標，提出 3 個不同的建置模型與啟發式演算法，並透過實驗來驗證各個啟發式演算法與 optimal solution 之效能差異，以及當候選控制點愈來愈多的情況下，這 3 個啟發式演算法的效能比較。

## 1.1 機會網路

機會網路 (Opportunistic Network, OppNet) [16]是由許多移動式節點 (mobile node, MN) 所組成，在典型的機會網路環境中，可能因為移動式節點的分佈位置不均、移動性、省電等原因，而造成整個網路被分成好幾個互不相連的區塊。

以下為機會網路的特點。

- 網路聯繫 (即溝通的機會) 是間歇性的。
- 極少存在一條從來源端到目的端的點對點路徑 (end-to-end path)。
- 兩端點之間的連線中斷與重新連線的情況是時常發生的。
- 網路連結的效能表現是高變動的，或甚至是極端的。

由於以上幾個特點，使得原本舊有的資料傳輸機制，無法直接被使用在機會網路中，為了因應此狀況，就產生了一個新的訊息交換機制—store-carry-forward，藉由這個訊息交換機制，可讓不存在點對點路徑的兩個節點，藉由中間節點的代轉送，讓訊息透過接力的方式來傳送到目的端，當沒有機會使資料往目的端發送時，代傳的節點會為來源端暫存這份資料，並等待適合的代傳節點出現，再將資料一步步地往目的端發送，因此，機會網路中的資料傳輸並沒有一個固定的路線，且即使是位於不同網路區塊的移動式節點，也有機會透過中間節點的代傳，而得以互相通訊。

在機會網路中，是藉由節點本身的移動性，來讓不同的節點之間有「機會」互相接觸，以傳送、接收或代傳資料，但也因為此特性，使得在資料傳輸的過程中，可能

因為環境因素、節點的移動性等原因，而造成移動式節點之間產生間歇性、臨機性的接觸與連結，這將造成每一次的封包傳遞常會有延遲的狀況，所花費的傳輸時間也難以預測，因此，機會網路是屬於耐延遲網路 (Delay Tolerant Networks, DTN) 的一種，不適合使用在即時性需求高的服務，例如：網路電話 (Voice over IP, VoIP)。

### 1.1.1 機會網路的訊息交換機制

運作在傳統網路下的資料交換機制，一般都假設在來源端與目的端之間存在有點對點的路徑 (end-to-end path)，然而在機會網路中，不一定存在這條點對點的路徑，因此，store-carry-forward 機制被提出作為機會網路上的訊息交換機制。

利用 store-carry-forward 訊息交換機制，節點需依靠本身所具備的移動性，且節點扮演的角色必需能隨需求適時地做切換。每個節點都身兼數職，可以承擔 host、router、或 gateway 的任務，當它作為 host 時，具有儲存及保管資料的能力，作為 router 時，此節點便同時具有儲存、攜帶、轉送資料的能力，可以做為同一個網路區塊中其他節點的代傳節點，而作為 gateway 時，則可以視為不同網路區塊之間的代傳節點，如此一來，即使各節點之間處於互不相連的網路區塊，也能藉此機制順利溝通，這種以 store-carry-forward 技術來傳送資料的方式，是機會網路資料傳輸的最大特色。

機會網路可能的應用環境，可以以一個分離的無線隨意網路 (disconnected Mobile Ad Hoc Network, disconnected MANET) [37]來說明，如圖 1.1 所示，該網路由許多移動式節點所組成，當整個網路因節點的移動性，而被切成許多大大小小、互不相連的網路區塊時，由於來源端和目的端之間不存在點對點的路徑，故位於不同區塊的節點是無法互相溝通的，這樣的網路型態即被稱為分離的無線隨意網路。

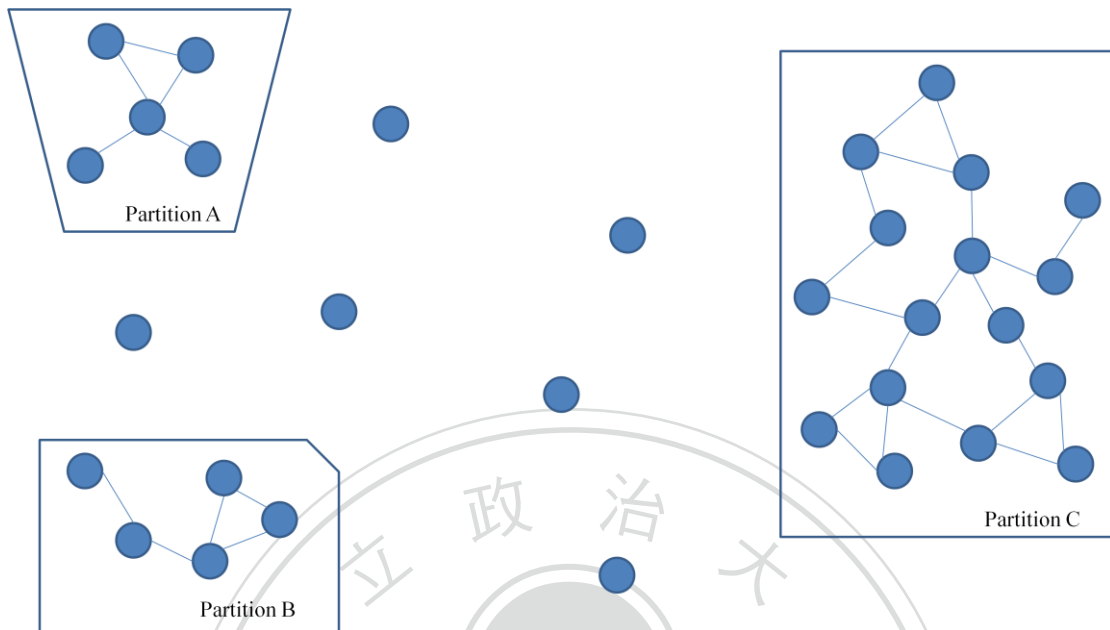


圖 1.1：Disconnected Mobile Ad Hoc Network (Disconnected MANET)。

在此分離的無線隨意網路中，若可以利用移動式節點的移動性及區域發送訊息 (local forwarding) 的能力，讓訊息藉由其他節點的代傳來傳送，那就可以達到資料交換的目的。

### 1.1.2 機會網路的研究議題

由於機會網路本身的特性或限制，使得在資料交換方面的研究，面臨了許多挑戰，以下分別依照不同的網路層級來說明[16]。

#### 1. 網路層

機會網路上的資料交換需依靠移動式節點之間的接觸，且節點同時擁有 host 及 relay 的身份時，整個機會網路才能運作得起來。在網路層中，由於每一節點皆肩負轉送任務，故將額外消耗儲存空間，因此，節點本身的儲存空間大小必需被列入考慮，基於此，產生了兩種完全不同的路由設計理念，分別是 flooding-based (multiple copies)



approach 及 forwarding-based (single copy) approach，分別說明如下。

- Flooding-based (multiple copies) approach：

發送端或中間節點在發出或轉送一個訊息時，會將同一個訊息的多個複本，透過不同的路由路徑來轉送給鄰近節點，直到將訊息送達目的端為止。由於訊息是循著許多不同路徑轉送，較有機會使用到較短的路程，可以降低資料傳送的延遲率，並提高資料傳輸的成功率，但也會造成網路上充斥著同一個訊息的多個複本，耗費轉送節點大量的儲存空間，且佔用較多資料傳輸所需的網路頻寬。

- Forwarding-based (single copy) approach：

發送端或中間節點在發出或轉送一個訊息時，僅轉送到一個鄰近節點，重覆資料轉送的動作，直到將訊息送達目的端為止。這種作法可以減少儲存空間的使用量，並避免網路上相同訊息的過度泛濫，然而，對同一個訊息而言，由於同時只有一個負責保管此訊息的代傳節點，而訊息轉送路徑無法保證為最短路徑，故也會造成延遲時間加長及資料傳遞效率偏低的情形。

## 2. 傳輸層

在機會網路中，節點之間發生連線中斷的狀況是相當頻繁的，且點對點路徑也極少是存在的，故使用在傳統網路上的傳輸層協定（例如：TCP），由於在設計之初，都是假設在點對點路徑已存在的環境下，因此，無法直接轉移到機會網路上使用。

## 3. 包裹層 (Bundle Layer)

這一層是為了支援 store-carry-forward 資料傳輸機制而產生，主要負責資料包裹（或稱為 bundles）的儲存、攜帶及轉送，當只有一個目的端節點時，需提供 unicast bundle delivery 的技術，當有多個目的端節點時，則需提供 multicast、anycast delivery 的方式來傳輸資料。

## 4. 應用層

在機會網路中，由於無法預期節點之間的連線何時會被中斷，故不容易妥善地利用節點接觸時所產生的通訊機會，因此，將傳統網路上的應用直接移植到機會網路中，仍頗具挑戰性。

即使是本質上具有耐延遲特性的網路應用，在傾向於分離的網路環境中，整體的效能也會顯著地下降，舉電子郵件這個高度耐延遲的網路應用來說，雖然使用者通常習慣為了一封回覆去等數小時或數天的時間，然而，由於現有支援電子郵件應用的傳輸層協定（例如：TCP），在設計之初，並沒有被考慮使用在機會網路這樣的環境中，因此，在機會網路中支援電子郵件的應用，仍是相當具有挑戰性的。

即使如此，在機會網路中支援電子郵件的應用，仍是相對簡單的，因為它耐延遲的特性，可以忍受機會網路的高延遲傳輸特性，然而，支援 Web 應用則要更複雜得多，因為高互動性的應用協定（例如：HTTP），不能忍受高延遲。

除了以上所探討的幾點外，另有兩個在機會網路中不可忽略的特殊挑戰。

- Contact – 應該要考慮當兩個移動式節點互相接觸時，可以成功傳送的資料量多寡。
- Storage constraint – 應該要考慮在移動式節點中，是否有足夠的儲存空間，用來暫存準備代傳的資料。

在機會網路中，即使存在了 store-carry-forward 訊息交換機制讓節點之間能順利交換資料，然而，由於與生俱來的網路連線特性，使得資訊交換的傳輸率及延遲時間難以估計，並且由於每個節點內部儲存容量的限制，故也可能造成資料未成功送出前就被丟掉，因此，對保證資料成功交換的挑戰性極大。

### 1.1.3 機會網路的應用

#### 1. 登山服務與山難搜救

由中研院、台灣大學與美國科羅拉多大學合作研發出的「山文誌登山資訊系統

(CenWits System)」[17-18, 48]，主要目的為收集遊客行走路徑與山區的氣候資訊，並將所收集到的資料做進一步的分析、處理與應用，現已實際應用於玉山國家公園，稱為玉山網 (YushanNet) [19, 47]。

將「山文誌登山資訊系統」應用於山難搜救時，可先經由控制中心所搜集到的資訊，來定位待援登山客的行蹤，並將搜救範圍縮小至最有效的區域，以協助搜救人員抓緊黃金救援時間，提升救援速度，並節省搜尋救援工作所耗費的人力與物力成本。

而關於登山服務方面，也可讓國家公園管理處得知遊客活動較頻繁的區域，作為日後增修遊客服務設施的參考，另外，登山客的家屬也可以透過網際網路，來掌握登山客目前的位置與行進狀況。

## 2. 野生動物追蹤

由於野生動物沒有固定路徑，故只能在特定地點找到牠們，例如：水源、巢穴等，在這些野生動物的身上裝置省電、無須額外設定的 GPS 收發器後，研究人員便可藉由裝置之間資料交換的所得，來分析出野生動物的活動範圍，以及不同物種間的互動及遷徙等關係，相關的研究如美國普林斯頓大學的 ZebraNet Project[20, 32]。

## 3. 外太空傳輸

美國國家航空暨太空總署 (NASA) 預計在 2011 年啟用的 InterPlanetary Internet Project 就是這個應用的實際案例[6, 11]。

網際網路 (Internet) 使用 TCP/IP 協定來保證網路中資料傳輸的可靠度，在這個協定中，傳輸資料的來源端和目的端必須在傳輸的過程中保持連線的建立，然而，這在外太空傳輸的過程中是很難做到的，因為距離太遠，再加上行星的運行及衛星的漂浮等情況，故在傳輸過程中，連線的間歇性中斷情形是很普遍的，因此，在這樣的環境下，就相當適合使用機會網路的傳輸特性，當連線中斷時，可以繼續暫存資料，直到下一個可能的節點再傳輸，並且，容許傳輸時間的長時間延遲及高資料錯誤率。

## 4. 開發中國家或偏遠地區的網路傳輸

以往許多技術都是為了已開發國家而研究、發展，然而，這些技術可能無法使用在開發中國家或偏遠地區，因為這些地區的基礎建設可能並不完善，甚至該地的人們可能也沒有一定程度的知識水平，在這樣的應用環境下，成本、部署 (deployment)、功耗等問題就需要被考慮，例如：美國柏克萊大學所研究的 Tier Project[5]，主要的研究目標就是要解決這些開發中國家或偏遠地區的網路通信問題。

## 5. 戰區、災區的通訊網路

以往 Mobile Ad Hoc Network (MANET) 的研究應用，常被設定在無建置基礎建設 (infrastructure)，或基礎建設已遭受損壞的戰區、災區環境中，然而，在這樣的環境中，由移動式節點所組成的許多互不相連的網路區塊，可能會因為移動式節點的移動性與無線傳輸範圍的限制，而造成不同的網路區塊之間無法互相通訊的情況，此時，若能利用節點的移動性，以及機會網路最重要的傳輸機制—store-carry-forward 訊息交換機制，就能適用於戰區、災區環境[31, 38]。

### 1.2 行動代理人

所謂的行動代理人 (mobile agent) 可以視為一支可在異質系統之間移動、執行，並具有特定功能的小型程式。當原本的節點 (home node) 送出行動代理人時，代理人就像使用者的分身一樣，從原本的節點中分離出來，自主地決定要訪問的節點，而無須和 home node 持續保持連絡，它將代表使用者在網路中的節點進行多次跳躍 (multiple hops) 移動與執行任務，最後將使用者所交待的任務結果回傳到 home node。

行動代理人其實也是代理人技術 (agent technology) 的一種。其基本原理就是當行動代理人 (mobile agent) 接收到終端使用者的請求命令時，便即時地處理使用者所交付的任務，並依實際的情況，可從網路一端的主機移動至其他另一端的主機去處理，最後再將結果回傳給使用者。實際上，在網路上移動的行動代理人，其狀態不僅包含本身的程式碼和任務處理資料，還包括了 thread 的狀態及其擁有者的權限。

另外，在終端間移動中的行動代理人，具備了自主性與機動性，即使是在無法和使用者相互通信的環境中，也能將移動前的程式憑藉自己本身的知識，或所收集到的情報，自主地加以分析判斷，並繼續執行處理。

### 1.2.1 行動代理人的特點

代理人通常扮演一個使用者或另一個代理人的角色，並負責和其他代理人之間的溝通 (communication) 與合作 (collaboration/cooperation)。以下為「行動代理人」所必備的特質[49]：

- 自適應性 (Adaptivity)：代理人具備學習能力並能隨經驗增長
- 自主性 (Autonomy)：代理人必須具有主動性，在朝達成目標的方向下，有自主能力採取啟發性之行為
- 協作行為 (Collaborative behavior)：代理人具備可以與其它代理人合作並一起達成目標之能力與行為
- 推論能力 (Inferential capability)：代理人具備有針對抽象定義之目標採取行動之能力，換句話說就是具備推論能力
- 知識水平的通訊能力 (Knowledge-level communication ability)：代理人在與其它代理人溝通時，具備使用人性化語言與溝通方式之能力
- 個性化 (Personality)：代理人可以具有個性化之特質，即具有可以扮演出逼真行為之能力
- 反應性 (Reactivity)：代理人具備選擇性感測與反應之能力
- 持續性 (Temporal continuity)：代理人具備保持身份扮演一致性之特質，並能夠長時間的維持
- 移動性 (Mobility)：代理人具備機動引導之能力，可以從一個主機移至另一個。

## 1.2.2 行動代理人的優點

使用行動代理人技術的好處，包括克服客戶端計算機能力的限制、客制化、與生俱來的存活能力、作為離線用戶的代表，和易於擴展。以下將逐一說明[34]。

- 克服客戶端計算機能力的限制 (Overcoming the limitations of a client computer)：舉例來說，先把代理人送到遠端主機的資料庫，並根據自定義的演算法進行搜尋後，再將資料回傳客戶端，比直接存取遠端資料庫更有效率，並能克服客戶端本身的通訊延遲、總處理能力、儲存容量大小、計算能力等限制，此外，客戶端也可能沒有足夠的儲存空間來暫存大量資料，或者可能沒有足夠的網路頻寬和計算能力。
- 客制化 (customization)：在 client/server 架構中，是很難適應快速變化的，相反的，行動代理人則可以很容易地依照使用者需求客制化，再將此新的需求送到伺服器端執行。其中，伺服器端的代理人系統代表一種解譯器 (interpreter)，負責接受並執行收到的代理人。
- 與生俱來的存活能力 (inherent survivability)：因為行動代理人能攜帶著本身的程式碼和狀態移動，故相較於 client/server 架構有更高的存活率，當一個節點或網路的一部分發生問題時，行動代理人可以離開這個節點，到其他地方再繼續執行原有的任務，雖然有其他方式可以提高 client/server 架構的執行效率，但仍無法適用於大規模的系統架構中，反之，行動代理人原本就是被設計用於 Web 上。
- 作為離線用戶的代表 (representation of a disconnected user)：現今很多使用者都具有移動性，時常在辦公室開始一個工作，而後在另一個地方利用筆記型電腦來繼續之前的工作，並希望在使用者離線的期間仍能繼續執行工作，故可利用送出行動代理人，讓使用者離線後，仍由代理人繼續處理原來的工作，

而後，當使用者再度上線時，再將代理人拉回。

- 易於擴展 (ease of development)：當大量的資料被儲存在分散的伺服器中，遠端存取所需的資料並非一個可行的選擇，甚至基於安全因素，本地資料可能不提供遠端存取，因此，可利用管線技術 (pipelining) 讓不同類型的代理人先後拜訪這些伺服器，並執行必要的動作。

### 1.2.3 行動代理人的應用

行動代理人在接收使用者的任務後，可以提供資訊處理之服務，包含資料的收集、檢索、過濾、交換等功能，在[24]中已提出幾個原因來說明行動代理人在許多應用中，是一個有效率的選擇，其優點有改善 client/server 架構中的頻寬限制、網路延遲，以及減少網路負載、網路斷線的弱點等。以下就現存的行動代理人的應用逐一說明[34, 45, 50]。

1. 電子商務 (Electronic commerce)：使用者可告知行動代理人欲購買的商品名稱、價格、品牌等資訊，並委託其在虛擬商店中，代使用者做搜尋、過濾、下訂單等動作。
2. 軟體發行 (Software distribution)：

當機器的數量不斷地增加，在其上安裝和維護軟體變得更困難，可利用行動代理人依照軟體包的軟件清單，並檢查本機安裝版本，來自定義安裝，以自動完成此動作。

3. 資訊檢索 (Information retrieval)：

使用者僅需輸入欲獲取資料的關鍵字，便可藉由行動代理人的移動，來檢索分散在網路上各主機中資料庫的資料，行動代理人便會在目的端主機資料庫，過濾出所需的資料，並依照檢索結果來選擇移到其他主機繼續處理，或回傳結果給使用者。

4. 系統管理 (System administration) :

行動代理人可以獨立地在大型系統中移動，並執行定期性的管理，例如：備份或其他系統相關的操作，並可在問題發生時，於人為操作的干預前做初步的問題評估，以幫助系統管理員釐清問題發生的原因及後果，此外，也可用於自動執行常規任務，及定期檢查系統和網路。

5. 網路管理 (Network management) :

網路本質上是分散式的，且提供一個適合部署代理人的環境，例如：在主動式網路 (Active Network) 中，可透過行動代理人來移動、傳送通信協定的處理程式，並安裝在原本沒有此通信協定的主機上，使其主機得以運作，即無須事先在各主機上做各通信協定標準化的動作。

6. 網路安全 (Network security) :

將非公開機密化的資訊安置於行動代理人，而後予以加密，並隨著行動代理人在網路中移動，可免於在傳送中遭受竊取、竄改的顧慮，特別是能將解密程式也置於代理人中，因此，在目的地端主機無須事先準備解密程式，則我們可以採用各種雙方已協調好的任意加密形式來作加密動作。

近年來，行動通信和無線網路技術的大幅躍進，使得行動代理人的利用不僅限於遠端資料查詢，還包括了分散式即時系統 (Real Time System) 和工作流程系統 (workflow system) 等環境，促成了其應用的多樣化。

### 1.3 運用行動代理人於機會網路

由於在機會網路上，訊息在傳遞的過程中，相較於傳統網路有更多的不確定性、不可預測性，故訊息的傳遞速度較慢，在這樣的環境中，若採用傳統的集中控制模式並不恰當，因此，我們希望將行動代理人技術導入機會網路中，提升機會網路的功能，以期產生更多可行的應用。例如：利用行動代理人去尋找其他行動代理人，遠比由中控



中心去找行動代理人要來得方便、快速，也無須耗費許多無謂的資訊來回時間。

我們以「山文誌登山資訊系統」為例，在此系統上導入行動代理人技術，可擴大此系統原有的功能，以下列舉幾個可能的應用情境：

- 變更既有登山行程：不同日期出發的登山客，能在發生臨時事件而改變行程時，通知對方新的集合時間及地點。例如：原本約定在兩天後於某定點集合，卻臨時得知有颱風警報或山中起濃霧，此時便可利用行動代理人通知所有人，取消原本的計畫，並馬上趕到另一個地點集合。
- 互動通訊：當某個登山客發現隊友脫隊時，可以先傳送訊息給走失的隊友，並提供初步的協助，或約定下一個集合地點。

### 1.3.1 機會網路上的行動代理人運作

行動代理人於機會網路中的功能與特性如下。

- 存在於移動式節點上，並可進行決策運算。例如：資料匯整，資料搜尋，變換路徑等。
- 可以決定何時轉移到其他特定的移動式節點。
- 可以決定何時啟動、暫停、恢復自己的工作。
- 當轉移到其他的移動式節點時，可被視為一個在節點之間交換的訊息。

### 1.3.2 行動代理人平台的功能需求

雖然將行動代理人技術應用於機會網路中，將帶來許多彈性與優勢，但不可否認地，由於行動代理人的本質仍是一支程式，而只要是軟體就可能有程式撰寫上的缺陷存在，導致破壞系統正常運作，因此，當某些代理人發生執行期的錯誤，而造成系統資源的損耗，影響系統之運作，或其他危及系統的情況（例如：系統當機）時，行動代理人平台必須能在不危及系統安全及正常運作的前提下，去解決這些問題。

行動代理人平台（例如：Aglets [23]、Concordia [44]、MASIF [33]、MOA [35]、

Mole [3]、Voyager [13]、PMADE [36]等)是一個存在於系統上的軟體，具有創建、解譯、執行、複製、轉移、終止行動代理人的能力，且必須提供代理人的控制機制，來讓其擁有者能監控代理人的執行狀態、增加新的任務或撤回舊任務等，同時，行動代理人之間也必須能在任務中進行溝通合作，此外，代理人的控制機制也作為處理代理人發生錯誤，或在其他例外處理時使用。

要實現行動代理人的控制機制，首先，必須存在代理人管理員 (Agent administrator) 元件，此元件具備代理人行為之控制權限，可提供代理人的狀態查詢及其他控制代理人行為之指令。

以上所提到的代理人控制功能，至少需具備以下幾項：

- Status inquiry：代理人狀態查詢功能，用來查詢代理人唯一的識別碼、來源及目的地節點之資訊、被分派執行的命令名稱、優先權值、佔用的系統資源多寡、已耗費的執行時間與完成度，及目前的執行狀態、中間結果等資訊。
- Stop：中止當前任務。
- Suspend：暫停當前執行的任務。
- Resume：恢復之前所執行的任務。
- Update assignment：分派其他新任務。
- Change its itinerary：改變代理人原來的移動路線規劃。
- Kill：強制刪除當前代理人及其任務。

以上所列出的幾點控制機制，最大的困難與前提，即是在於「如何找到所欲控制的行動代理人」，只有先找到某個特定的行動代理人，方能對其進行控制。

## 1.4 論文組織架構

本論文的其餘部分被組織如下。第二章討論搜尋行動代理人之相關研究及挑戰，並簡介「山文誌登山資訊系統」，第三章提出我們的控制網路概念及基於「山文誌登山資訊

系統」為背景的控制網路之建置模型，並在第四章針對相關建置模型，進行問題分析及提出解決方案，之後，在第五章為解決方案之演算法進行效能評估，最後，在第六章總結本論文之研究。



## 第2章

### 相關研究與挑戰

#### 2.1 搜尋行動代理人之相關研究

為了實現行動代理人的離線操作能力和自主性，使得行動代理人平台有必要納入代理人追蹤機制，來讓以行動代理人為基礎的系統中，能實現行動代理人的任務協調、溝通、取消等。

目前已知普遍存在於現有代理人追蹤機制的問題如下[40]。

- Unnecessary communication overhead：多數的作法都需要行動代理人更新自己的位置表，雖然這種作法容易又直接，但當行動代理人長時間漫遊時，會發送許多不必要的位置更新訊息，增加網路負擔。
- Location database server bottleneck：在某些作法中，每當行動代理人移動到新的節點時，便會自動將當前的位置回傳給特定的伺服器，此伺服器具備一個位置更新資料庫來儲存這些位置資訊，故當位置更新訊息多時，因伺服器的儲存空間有限，將成為這類作法之瓶頸。
- High location update and search cost：此搜尋成本主要指所發送的訊息數量、大小、傳送距離，與執行每個搜尋動作所耗費的時間。當行動代理人在節點間的跳接頻繁時，大量的位置更新訊息會造成網路流量加劇，且當行動代理人移動距離長的時候，資訊更新速度將更緩慢，故不足以支援緊急任務之應用。

因此，一個有效率的代理人追蹤機制，應盡可能減少行動代理人的位置搜尋、更新等成本。

### 2.1.1 過去的搜尋行動代理人之方法

關於行動代理人追蹤機制的研究，至少已有多年的研究歷程與成果，因此，現有的追蹤方法有很多，例如：Database logging protocol (DL)、Path proxy (PP)、Blackboard、Shadow、SPC、Optimal location update scheme、Movement Based Location Management (MBLM)、Scalable Hash-Based Mobile Agent Location Mechanism、Blind Search、Intelligent Search 等，以下我們將分別描述這幾種類型的作法。

在 DL [1, 39] 中，使用一個特定伺服器 (home host，或稱為 location database server) 來負責管理位置資訊，並接收漫遊在網路中的行動代理人所傳送過來的位置更新訊息，當要搜尋特定代理人時，首先會向 home host 詢問代理人的位置，並直接和欲搜尋的代理人進行聯繫，屬於 home-based 的作法。在此作法中，因為需利用一個特定伺服器中的資料庫來保留行動代理人的移動軌跡、記錄其最新位置，故被視為此種搜尋方法的效率瓶頸，此外，當行動代理人距離 home host 很遠時，也將提高遠程伺服器更新位置資訊的成本，並造成位置更新速度緩慢的問題，使得當需要和特定代理人溝通時，可能此代理人已離開已知的位置，故此方法並不適用大規模的網路。MOA 和 Aglet 等行動代理人系統平台都是使用此模式。

在 PP [1, 39] 中，使用指向的概念，當行動代理人要移動到一個節點時，會在目前的節點留下其下一個目的地節點的指針，而不是向特定伺服器回報位置資訊，故搜尋代理人時，將由行動代理人原本存在的節點開始，經由這群指針所得到的移動路徑來找到它，然而，當代理人的移動路徑很長時，只要中間有一個指針失效，此行動代理人就無法被找到，且此作法所耗費的搜尋成本可能很高，這取決於代理人的移動路徑長度。Mole 和 Voyager 等行動代理人系統都是使用此模式，來追蹤行動代理人並與其

溝通。

在 Blackboard protocol [41] 中，需維護一個共有的黑板，讓每個節點都能在上面分享訊息交換資訊，當任何行動代理人想傳遞訊息時，便將訊息放到黑板上，而不管接收者在哪或何時才看到此訊息，故需要接收端的行動代理人移動到相應的節點去得到此訊息，將增加不必要的通訊花費。

在 Shadow protocol [2] 中，是結合了 DL 和 PP 兩種作法，和 PP 不同的是，行動代理人是在每個 TTL (time to live) 時間才向相關的 shadow 更新自己目前的位置，而不是每走一個 hop 就更新一次，故可以縮短所記錄的代理人移動路徑。

在 SPC protocol [10] 中，也是結合了 DL 和 PP 兩種作法，在搜尋行動代理人時，會追隨著被留在兩個註冊節點的部分指標來找到它。

在 Optimal location update and search algorithm [26-27] 中，當行動代理人每移動  $d$  個節點時，才更新一次位置資訊，其中  $d$  值為其最佳門檻值，可經用數學方法得出。

在 MBLM [4] 中，將以 SPC 為基礎，並藉由採用一個 optimal location update and search algorithm [27] 來搜尋行動代理人，以減少代理人位置更新訊息的數量。

在 Scalable Hash-Based Mobile Agent Location Mechanism [22] 中，利用一種被稱為 information agents (IAgents) 的特殊代理人，來負責維護一群代理人的位置資訊，而個別 IAgent 所應負責哪些代理人，則是透過一個雜湊函數來決定，當欲搜尋行動代理人時，需先知道此代理人的 ID 及雜湊樹，以計算出負責此代理人的 IAgent，屬於部分中央集權式 (centralized) 作法。

在 Blind Search [28] 中，提出利用 Basic Binary Search (BBS) Algorithm 來搜尋行動代理人的方法，首先假設已知欲搜尋的行動代理人之移動路徑，其中包含預計會經過的節點及順序，此盲目搜尋演算法在運作流程中，會先檢查路徑最中間的節點，若代理人已經過此節點，則往前搜尋，否則往後搜尋，重覆以上動作以逐漸逼近欲搜尋的代理人。此作法雖簡單、容易實作，但其假設使它無法被應用在「不能事先得知被追

蹤的代理人之移動路徑」的情況下。此外，作者為了解決 BBS 演算法可能不小心遺漏行動代理人的缺點，故又提出 Extended Binary Search (EBS) Algorithm，藉由避免一次排除過多的候選節點，來提高搜尋成功率。

在 Intelligent Search[28-29]中，提出利用 Intelligent Binary Search (IBS) Algorithm 來搜尋行動代理人的方法，除了假設已知行動代理人的移動路線（包含預計會經過的節點及順序）外，也假設事先已知每個代理人任務在各個伺服器將花費的服務時間，其中服務時間是預估值，如此一來，可藉由平均服務時間，來猜測目前欲搜尋的行動代理人所在的位置，而非盲目搜尋，故能有效減少搜尋的次數。在行動代理人與網路運作皆正常的情況下，其搜尋效率將高於 Blind Search，反之，由於事先已知的資訊是錯誤的，故使用 Blind Search 可能是較好的選擇。

### 2.1.2 機會網路上搜尋行動代理人之挑戰

以上我們已列出一些現有的搜尋機制，由於這些搜尋機制只考慮行動代理人的移動性，而不考慮節點的移動性，故較適合應用在有線網路下。因此，考慮到網路特性的差異，這些搜尋機制無法直接被使用在機會網路中，主要有以下幾點原因與困難。

- 在機會網路中，行動代理人存在於移動式節點上，但由於移動式節點具有「移動性」的特質，故即使知道欲搜尋的行動代理人位於哪一個移動式節點，也難以預測此移動式節點目前的位置。
- 由於機會網路本身長延遲的特性，可能導致行動代理人的位置更新速度慢，故搜尋代理人所掌握的被搜尋代理人的位置資料可能與實際位置相差甚遠。

## 2.2 「山文誌登山資訊系統」簡介

「山文誌登山資訊系統 (CenWits System)」[17-18, 48]為一個山區遊客追蹤與環境監控系統，主要目標為收集遊客行走路徑與山區的氣候資訊，再將所收集到的資料做進一步的分析、處理與應用，故可追蹤遊客在國家公園內之分佈狀況，及了解國家公園內

之環境變化。此系統由中研院、台灣大學與美國科羅拉多大學所合作研發，目前已被建置於玉山國家公園，使用於遊客的登山活動中，相關的應用有登山服務、山難搜救等。

「山文誌登山資訊系統」除了可提供玉山國家公園管理處了解遊客在國家公園內之活動區域，以作為增修各項遊客服務設施之參考外，更可在不幸發生失蹤山難事件時，快速地調閱出「失蹤者的最後出現時間、地點」，有效地為相關救援單位縮小搜救範圍，避免被口耳相傳的模糊資訊所誤導，以提升在七十二小時黃金救援時間內，救援成功的機率，加強保障遊客之生命安全，並節省大範圍搜尋救援工作所耗費之人力、物力成本。

在國家公園內，各地區地形的差異很大，故無線電訊號的傳送會受到環境狀況與遮蔽物的影響，而影響無線通訊的能力。這是因為電磁波傳播所經過的路徑，可能被許多不同的地形或障礙物所影響 (None Line of Sight, NLOS)，而不再只是很簡單的直線傳播 (Line of Sight, LOS)，無線訊號可能會經過建築物、山丘、或樹葉的反射，而產生多重路徑效應，造成訊號放大或衰減。因此，將使得國家公園內的許多地區都無法接收到無線網路的訊號，若要達到每個地點都能收到無線網路的訊號，將會大幅提升建造成本，困難度也會超乎想像。

此外，山區的人口少、自然景觀珍貴，建置過多基地台除了浪費成本，也嚴重破壞自然景觀與生態，故在這種環境下，傳統行動電話系統的覆蓋率通常不高，使得傳統行動電話系統在山區並不可靠，故可進而假設大多數登山客並無行動電話可用。

又因為目前的網路存在一個很大的缺點，即通信端與接訊端必須要在連線狀態才能傳遞資料，如此當遊客在通訊的死角則無法被追蹤到，為了解決此問題，「山文誌登山資訊系統」使用最新的無線微型收發器，搭配全球衛星定位系統，並藉由機會網路這種「耐延遲網路」技術，讓每位遊客彼此於山徑中相遇時，能彼此交換其所攜行的



路徑與位置等資訊，最後利用架設於各山頭的無線網路中繼站，來收集、儲存遊客的行走路徑等資料，並傳回控制中心，使控制中心能掌握遊客的動向。

「山文誌登山資訊系統」的運作流程如下：

- 每個登山客攜帶一個約 70 克重、火柴盒般大小的無線微型收發器（內含 CPU、記憶體、天線，可執行使用者指定的程式），並搭配全球衛星定位系統（GPS）。
- 每個無線微型收發器會自動記錄各別登山客的移動路徑，換句話說，無線微型收發器會定時由 GPS 感測器取得登山客的位置，並暫存在模組的快閃記憶體中。
- 當登山客在山徑中相遇時，彼此的收發器便會自動記錄相遇的時間及經緯度，並經由 Zigbee 短距無線接取技術，來互相交換對方收發器中所儲存的路徑與位置資訊。
- 而當登山客行經預先架設於各山頭登山口、涼亭等處的無線網路中繼站（基地台）時，所有儲存在無線微型收發器中的資料（包含自己的路徑紀錄，及所有曾相遇過的登山客所攜之收發器中的紀錄），就會自動藉由基地台來傳回到系統內部的控制中心。

在以上的運作流程中，由於考慮到登山客往往是一天或數天的行程，以及山徑並不如平面道路寬敞，又遊客在山徑中相遇的時間是相當短的，因此，基於無線微型收發器的電池耗電量、傳輸距離的限制，以及 Zigbee 基地台數量有限等考量，目前在此系統使用 Zigbee 短距離無線傳輸技術，並結合接力傳送的方式，來達到訊息傳遞的目的。此外，由於是使用 Zigbee 短波通訊協定的公共頻道，來傳輸登山客的軌跡資料，故相較於利用現有的衛星天線、3G，其通訊費用是完全免費的。而雖然 Zigbee 傳輸距離短，但在正式通訊前建立連線的速度比使用藍芽技術快，也較使用 IEEE 802.11 無線通訊協定省電，故更適合使用在這種情境中。

在此系統中，基地台的主要功能是接收無線微型收發器中的登山客 GPS 資料，再透過 3G 網路上傳到系統控制中心。此處的基地台可分為簡易型與太陽能型兩種，簡易型是架設在電源供應充足的管理站或是排雲山莊，太陽能型則是利用太陽能板與充電電池，架設在步道旁的涼亭或其他沒有電源供應的地區。此外，基地台也會收集風速、雨量、風向等氣象資料，並利用網路攝影機來做生態的觀測。

此系統目前所使用的最新無線微型收發器，除了體積小、重量輕（約 70 克）以外，也具有防水、防摔、耐熱等特點，以及緊急呼救的功能，實際使用時，登山客只要在出發前充飽收發器的電力，在登山口啟動收發器，夜宿時關閉，不需要使用者手動操作，無線微型收發器就可以自動提供登山客的路徑資訊給系統控制中心。



## 第3章

### 機會網路上行動代理人之搜尋

在機會網路上，要導入行動代理人技術，首要之事即是實現行動代理人之搜尋機制，然而，以上述討論我們可知，在機會網路上的行動代理人之搜尋，存在著極大的挑戰性。

以「山文誌登山資訊系統」應用情境為例，我們已知每個登山客都會攜帶一個配備有 GPS 功能的小型設備，此設備在本論文中簡稱為「行動節點 (mobile node, MN)」，此行動節點會在登山客的移動過程中，自動收集位置資訊，並使用短程無線電來和鄰近的登山客交換彼此的移動資訊。

然而，由於行動代理人是附屬於登山客所攜帶的行動節點上，故當兩個行動節點 (登山客) 互相接近時，行動代理人才可以從其中一個行動節點轉移到另一個行動節點，其行動力受制於登山客的移動行為，此外，由於登山客行走速率差異不大、所在位置分佈不均，故行動代理人的轉移將極為緩慢且需仰賴不確定的碰面機會，使得行動代理人之搜尋效率極低。

如圖 3.1 所示，在不存在控制網路 (Control Network) 的情況下，若目的端實際上比來源端提早一天出發，不管怎麼走，走在後面的來源端，都很難讓行動代理人經由中間節點的跳接來到達目的端。

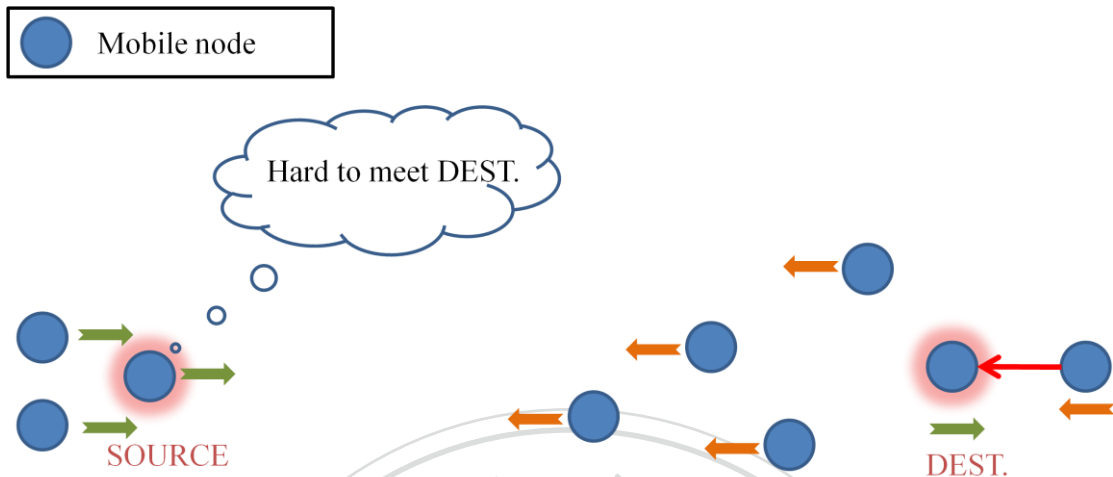


圖 3.1：在機會網路上搜尋行動代理人（以「山文誌登山資訊系統」應用情境為例）。

為了解決在機會網路環境上搜尋行動代理人的困難，我們提出一個高速控制網路的概念，來輔助搜尋代理人的移動，以提高行動代理人的搜尋效率。

### 3.1 控制網路概念

假設在山區存在有固定網路（例如：Internet），我們可利用此網路（在可收到 3G、4G 等系統的信號之處，亦可用 3G、4G 等系統連網）建立控制點（Control Point, CP），形成控制網路（Control Network），讓行動代理人在控制點之間快速移動到所欲搜尋的代理人的前方控制點上，再跳至反向登山客所攜帶的裝置（行動節點）上，隨著反向登山客的行進，與迎面而來的登山客所攜帶的行動代理人互相溝通，進行搜尋任務。

如圖 3.2 所示，假設目的端比來源端提早一天出發，故走在後面的來源端，極難讓行動代理人經由中間其他行動節點的跳接，來到達目的端。因此，需透過控制網路的輔助，先將來源端送出的搜尋代理人傳送到其最近的控制點，再利用控制網路的快速轉送，將行動代理人傳送到「最接近目的端，且位於目的端未來可能會經過的移動

路線上」的控制點，以上動作是藉由搜尋代理人在控制網路內部的移動，來加速行動代理人之搜尋速度。

若此時搜尋代理人已距離目的端夠近，則直接由控制點傳送到目的端，否則，將經由其他與目的端反方向之行動節點的跳接，來逐漸靠近目的端。由於上述的行動節點指的就是由登山客所攜帶的裝置，故事實上是透過登山客的移動，來轉送搜尋代理人，因此，不但轉送速度非常慢，且用來轉送的行動節點必須與目的端的前進方向相反才行。

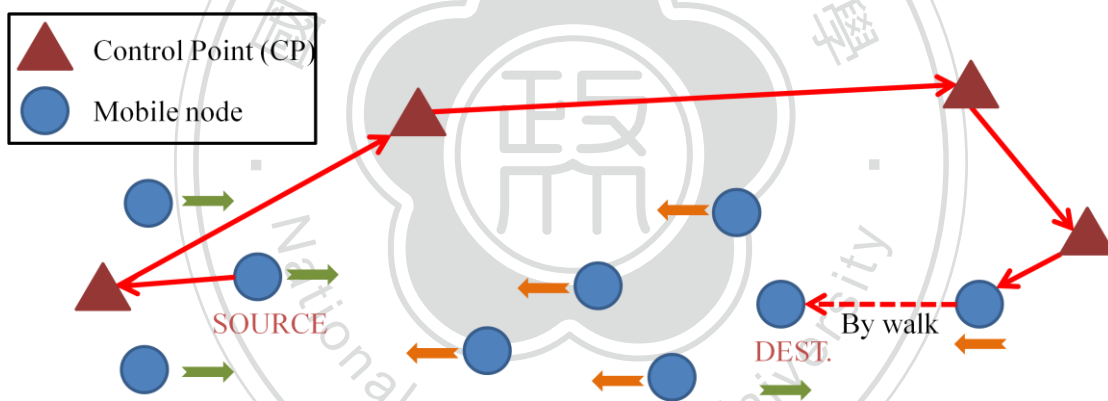


圖 3.2：使用控制網路來輔助行動代理人的搜尋。

### 3.2 利用控制網路的搜尋策略

當有了控制網路後，我們可以修改現有的行動代理人之搜尋機制，來符合機會網路上的需要。

以修改 Basic Binary Search (BBS) Algorithm [28] 為例，運作流程如下。

1. 先利用現有的搜尋演算法，找到最接近行動代理人的控制點。例如：先將控

制點依照行動代理人的移動路徑方向排列，再利用 Basic Binary Search (BBS) Algorithm，從中間的控制點開始找，若發現行動代理人已在先前經過此控制點，則往後找，否則往前找。

2. 找到最接近行動代理人的控制點後，將用來搜尋用的行動代理人，送到行經此控制點的反向行動節點上，再由這些行動節點逐步接近欲搜尋的行動代理人。

在下一節，我們將針對控制網路的建置問題，做各種面向的討論，並各別視不同情況之考量，將其精確地塑造成幾個數學優化模型。

### 3.3 控制網路的建置

控制網路的建置模型，並沒有一定的標準，在不同的環境下，所適合的控制網路皆不盡相同，如要全盤考慮，其面向與範圍將非常龐大與複雜，故本篇論文僅以「山文誌登山資訊系統」為例，來設計適用於山區環境與登山情境的控制網路。

#### 3.3.1 環境假設

以「山文誌登山資訊系統」為背景，我們可假設原本在山上就已存在網路，只是登山客在野外活動中不會、也不能去用到這個網路。因此，在本篇論文中，假設服務範圍內已存在有線或無線封包網路（例如：Internet），是相當合理的。

此外，基於此研究背景的登山應用，我們也作以下幾點環境假設。

1. 登山客遵循某些健行路徑在移動。
2. 人（行動節點）的移動速率差異不大。
3. 原則上候選控制點是位於健行路徑的交界處或端點，但若候選控制點之間的健行路徑仍太長，也可在路徑中間另加候選控制點。

### 3.3.2 設計考量與目標

由於控制點的建置，需考慮山區的地理環境與人為維護等因素，故並非任何地點都可以用來建立控制點，而可以挑選來建立控制點的地方，稱為「候選控制點」。

在上一節的環境假設前提下，我們考量到每個候選控制點的預期花費都不同，且建置控制網路的整體花費有上限，故只能挑選部分地點來建立控制點，而不能、也不需要滿山建置控制點。換句話說，為了建置成本的考量，及避免自然景觀的破壞，在控制網路的建置上，必須有效地限制控制點的數量。

控制點與行動節點如同 2.2 所述之基地台與無線微型收發器，在控制網路中，行動節點會利用無線通訊技術，來和通訊範圍內的行動節點或基地台通訊。

以下我們的控制點選擇問題 (Control Point Selection Problem, CPSP)，將考慮如何在有限的預算資源下，挑選適當的候選控制點，來組成最佳的控制網路。

事實上，評估控制網路的最佳效能評估指標為「搜尋行動代理人所需花費的平均時間」，故我們的主要設計理念為「限制建置成本，並盡可能地縮短每個行動節點與其最近控制點之間的距離」，因此，我們定義了一個評估指標—MC\_distance，來代表「某一個行動節點到它最近的控制點之間的距離」。

然而，雖然將 MC\_distance 設定為目標函數 (objective function) 更符合需求，但因為加入問題模型後的計算太過於複雜，故我們將對控制點選擇問題之模型略做一些簡化，構成不同複雜度的數個模型，使用者可依自身需求，選用任一模型。這些模型的目標與限制條件都遵循以下幾個原則：

1. 限制：整體的建置花費不得超過可用資源之上限。
2. 目標：
  - 盡可能服務到最多的行動代理人或行動節點。
  - 能快速地找到任一個行動代理人或行動節點。

### 3.4 控制點選擇問題—最大總流量模型

基於上述的控制點選擇問題之設計限制、目標，以下我們將提出控制點選擇問題—最大總流量模型。

由於行動代理人搜尋機制的主要任務，就是要發送搜尋代理人來找特定的行動代理人，因此，控制點的可能位置，若有機會涵蓋、服務到最多的行動節點的話，對於快速地找到行動代理人的機會，相信會比較大的。

假設每一條健行路徑的登山客流量為已知，此模型在控制網路的建置策略上，是在限制可用資源的前提下，以獲得最大的行動節點之總流量為目的，來選擇控制點的佈建地點，可歸納如下。

- 限制：整體的建置花費不得超過可用資源之上限。
- 目標：最大化控制點所涵蓋的行動節點之總流量。

在此模型中，我們假設只在健行路徑的交界處或端點存在有候選控制點，且候選控制點之間的距離長度皆相等，故最大化控制點所涵蓋的行動節點之總流量，即能代表最大化地提升服務到行動節點的機會。

#### 3.4.1 最大總流量的數學模型

以下為控制點選擇問題—最大總流量 (CPSP-Flow) 的數學模型。

Given a graph  $G=(V,E)$ , where

- $V=\{v_1,\dots,v_n\}$  is the set of possible nodes for building control points,
- $E=\{e_{ij}|v_i,v_j\in V\}$  is the set of edges (trail segments) between  $v_i$  and  $v_j$ ,
- $W=\{w_i|v_i\in V\}$  is the set of the cost (required resources) of building a control point



on  $v_i$ ,

- $C$  is the available resources,
- $T = \{t_1, \dots, t_m\}$  is the set of hiking trails, each of which is a path, a sequence of nodes, used for hiking, and
- $F = \{f_k | t_k \in T\}$  is the set of the number of hikers per time unit passing the trail,

the Control Point Selection Problem is to find the set of nodes  $S \subseteq V$ , such that

$$\text{maximize } \sum_{i \in V} \sum_{i \in t_k} f_k x_i,$$

subject to

$$\sum_{i \in V} w_i x_i \leq C, \text{ where } x_i = \begin{cases} 1 & \text{if } v_i \in S, \\ 0 & \text{otherwise.} \end{cases}$$

### 3.4.2 最大總流量模型的缺點

由於此模型的设计目標，是選取使通過控制點的總流量最大化的控制點，來建立控制網路，因此，可能導致控制點的佈建地點，無法平均分佈於山區各地，甚至可能只集中在少數幾條行動節點流量較多的路徑上。

如圖 3.3 如示，當控制網路的搜尋範圍，只能涵蓋到所有健行路徑的一小部分時，將使得部分行動節點很容易就遠離了控制網路所能提供服務的區域，而造成行動節點的搜尋成功率大為降低，或搜尋時間大幅增加。

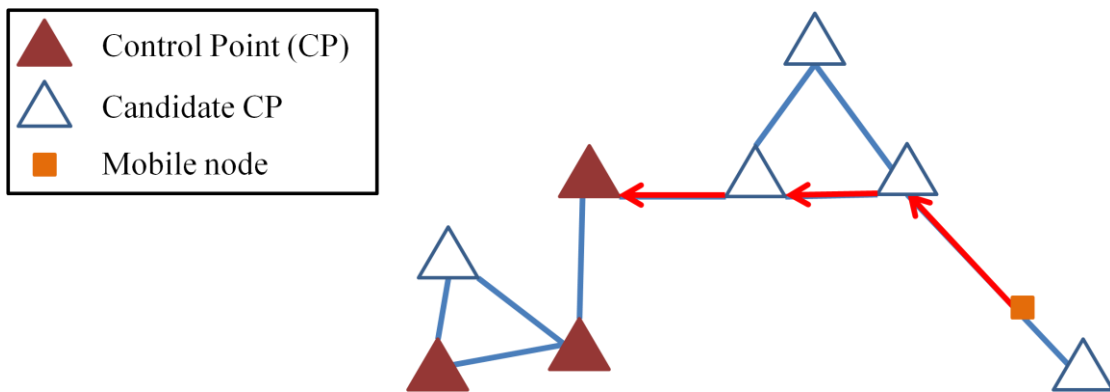


圖 3.3：最大總流量模型下，可能的控制點建置狀況。

由以上的討論可知，此模型的控制網路建置設計，除了可能導致浪費可用資源外，更可能造成搜尋不到行動代理人的嚴重後果。

### 3.5 控制點選擇問題—最大涵蓋率模型

基於上述模型的缺點，我們另外設計了最大涵蓋率模型，在此模型中，我們加入以下兩個設計目標。

1. 必須涵蓋所有健行路徑。因為行走在沒被涵蓋到的路徑上的人，可能永遠無法被找到。
2. 控制點建置位置分佈均勻。因為我們希望今天、明天、前幾天出發的人，都可以很快地被找到。

故此模型的設計如下：

- 1、每一條健行路徑上，至少需建置一個控制點。
- 2、盡量讓佈建的位置平均分散，以擴大控制點所涵蓋的區域。

我們希望此模型所建置出的控制網路，如圖 3.4 所示。

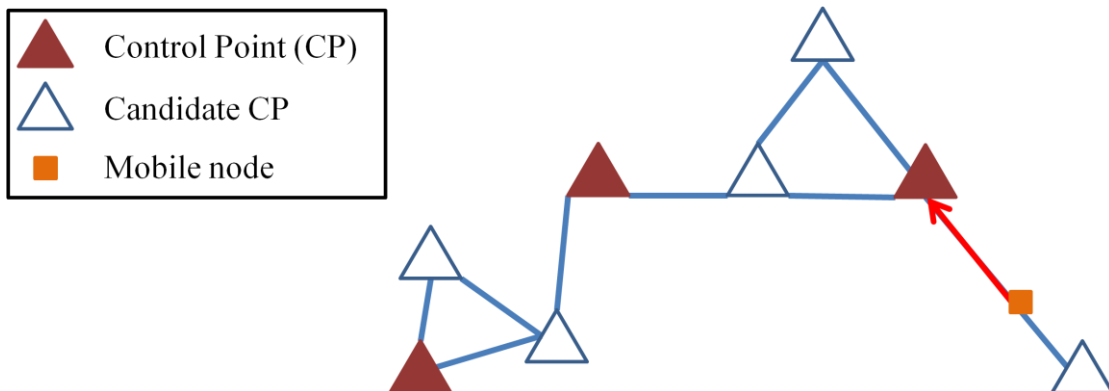


圖 3.4：最大涵蓋率模型改善控制點之分佈。

但由於 Graph 由 node 和 edge 所組成，並沒有面積的概念，所以無法輕易地用較精確的數學模型，來表示此模型的設計概念，故我們將此模型的設計，轉化如下：

- 限制一：整體的建置花費不得超過可用資源之上限。
- 限制二：控制點的建置，需涵蓋到所有的健行路徑。
- 目標：最大化控制點所涵蓋的 edges 數。

在此模型中，我們假設只在健行路徑的交界處或端點存在有候選控制點，且候選控制點之間的 edge 長度皆相等，尚不考慮各 edge 長度不一的情況，故最大化控制點所涵蓋的 edges 數，即能代表盡可能地擴大控制點所涵蓋的區域。

### 3.5.1 最大涵蓋率的數學模型

以下為控制點選擇問題—最大涵蓋率 (CPSP-Coverage) 的數學模型。

Given a graph  $G=(V, E)$ , where

- $V = \{v_1, \dots, v_n\}$  is the set of possible nodes for building control points,

- $E = \{e_{ij} \mid v_i, v_j \in V\}$  is the set of edges(trail segments) between  $v_i$  and  $v_j$ ,
- $W = \{w_i \mid v_i \in V\}$  is the set of the cost of building a control point on  $v_i$ ,
- $C$  is the available resources,
- $T = \{t_1, \dots, t_m\}$  is the set of hiking trails, each of which is a path, a sequence of nodes, used for hiking, and
- $F = \{f_{ki} \mid v_i \in V, t_k \in T\}$  is a passing\_trail matrix, where
 
$$f_{ki} = \begin{cases} t_k & \text{if trail } t_k \text{ pass node } v_i, \\ 0 & \text{otherwise,} \end{cases}$$

the Control Point Decision Problem is to find  $S \subseteq V$ , such that

$$\text{maximize } \left| \bigcup \{e_{ij} \cdot x_i\} \right|,$$

subject to

$$\bigcup \{f_{ki} \cdot x_i\} = T \text{ and } \sum_{i \in V} w_i x_i \leq C, \text{ where } x_i = \begin{cases} 1 & \text{if } v_i \in S, \\ 0 & \text{otherwise.} \end{cases}$$

在上述的數學模型中， $f_{ki}$ 表示在 node  $i$ 上是否有通過 trail  $k$ ，故 $\bigcup \{f_{ki} \cdot x_i\}$ 則表示所有被選中的 node 上所通過的 trail 之聯集，而 $\left| \bigcup \{e_{ij} \cdot x_i\} \right|$ 則表示所有被選中的 node 所鄰近的 edge 之總數，且不計算重複的 edge。

### 3.5.2 最大涵蓋率模型的缺點

此模型為一個控制點的涵蓋率問題，並隱含「每一條 edge 的長度都一樣長」的假設，故在此模型中並沒有考慮到候選控制點之間路段的長度，如此可能導致有些長路段 (例如：2 小時路程)，卻和短路段 (例如：30 分鐘路程) 所設的控制點數量相同，導致

不均勻的控制點分佈。

另外，我們也應考量此路段的重要性，例如：在容易發生山難的地方建置控制點有較高的優先度，讓此路段的控制點較為密集，才能提供更好的服務。

### 3.6 控制點選擇問題—最大加權涵蓋率模型

為了彌補最大涵蓋率模型的不足，我們提出一個考慮更多、更周詳的最大加權涵蓋率模型，首先，我們將環境假設修改為「不只在健行路徑的交界處或端點存在有候選控制點，若候選控制點之間的健行路徑仍太長，也可在路徑中間另加候選控制點」，並加入路段重要性的考量，希望能將候選控制點之間的路段長度、重要性等狀況，也一併考慮進去，做為此路段的重要性權值（權重），主要設計如下。

- 限制一：整體的建置花費不得超過可用資源之上限。
- 限制二：控制點的建置，需涵蓋到所有的健行路徑。
- 目標：最大化控制點所涵蓋 edges 的總權值。

在將此模型轉化為數學表示式之前，我們先定義一個數學函式，如下。

$$P(M) = \left\{ \sum s_{ij} \mid e_{ij} \in M \right\}$$

其中， $s_{ij}$  表示已知的路段  $e_{ij}$  之重要性權值，其數值愈大，代表重要性愈高，故函式  $P(M)$  即為在集合  $M$  中這些路段的重要性權值之總和，也可視為選擇這些路段來建置控制點，可獲得的總收益 (profit)。

#### 3.6.1 最大加權涵蓋率的數學模型

以下為控制點選擇問題—最大加權涵蓋率 (CPSP-Utility) 的數學模型。

Given a graph  $G=(V,E)$ , where

- $V = \{v_1, \dots, v_n\}$  is the set of possible nodes for building control points,
- $E = \{e_{ij} \mid v_i, v_j \in V\}$  is the set of edges(trail segments) between  $v_i$  and  $v_j$ ,
- $S = \{s_{ij} \mid v_i, v_j \in V\}$  is the weight of the edge between  $v_i$  and  $v_j$ , represents the edge length and the probability of mountain incident occurred.
- $W = \{w_i \mid v_i \in V\}$  is the set of the cost of building a control point on  $v_i$ ,
- $C$  is the available resources,
- $T = \{t_1, \dots, t_m\}$  is the set of hiking trails, each of which is a path, a sequence of nodes, used for hiking, and
- $F = \{f_{ki} \mid v_i \in V, t_k \in T\}$  is a passing trail matrix, where
 
$$f_{ki} = \begin{cases} t_k & \text{if trail } t_k \text{ pass node } v_i, \\ 0 & \text{otherwise,} \end{cases}$$

the Control Point Decision Problem is to find  $S \subseteq V$ , such that

$$\text{maximize } P\left(\bigcup \{e_{ij} \cdot x_{ij}\}\right),$$

subject to

$$\bigcup \{f_{ki} \cdot x_i\} = T \text{ and } \sum_{i \in V} w_i x_i \leq C, \text{ where } x_i = \begin{cases} 1 & \text{if } v_i \in S, \\ 0 & \text{otherwise.} \end{cases}$$

## 第4章

### 問題分析與演算法

在這一章中，我們將分別對控制點選擇問題—最大總流量模型、最大涵蓋率模型做問題分析，並在證明這兩個模型皆為 NP-complete 問題後，各別針對最大涵蓋率模型、最大加權涵蓋率模型，提出可行的啟發式演算法，來解決此控制點選擇問題。

#### 4.1 數學模型之問題分析

以下我們將針對控制點選擇問題—最大總流量模型、最大涵蓋率模型做問題分析，以證明這兩個模型皆為 NP-complete 問題。

##### 4.1.1 問題分析—最大總流量模型

以下我們將證明控制點選擇問題—最大總流量模型為 0-1 knapsack problem。

Definition of the 0-1 knapsack problem[9]：

In the following, we have  $n$  kinds of items, 1 through  $n$ . Each kind of item  $j$  has a non-negative value  $p_j$  and a non-negative weight  $w_j$ . The maximum weight that we can carry in the knapsack is  $W$ . Mathematically the 0-1 knapsack problem is as follows.

$$\begin{aligned} & \text{maximize } \sum_{j=1}^n p_j x_j, \\ & \text{subject to } \sum_{j=1}^n w_j x_j \leq W, \quad x_j \in \{0,1\} \end{aligned}$$

而控制點選擇問題—最大總流量模型的最佳化版本，可以被表示如下：

$$\begin{aligned} & \text{maximize } \sum_{i \in V} \sum_{i \in t_k} f_k x_i, \\ & \text{subject to } \sum_{i \in V} w_i x_i \leq C, \text{ where } x_i = \begin{cases} 1 & \text{if } v_i \in S, \\ 0 & \text{otherwise.} \end{cases} \end{aligned}$$

其中， $\sum_{i \in t_k} f_k$  代表所有經過 node  $i$  的健行路徑上的單位時間登山客數量之和，為

已知參數，令  $\sum_{i \in t_k} f_k$  為  $p_i$ ，則可以將 maximize  $\sum_{i \in V} \sum_{i \in t_k} f_k x_i$  轉換為 maximize  $\sum_{i \in V} p_i x_i$ 。再

加上  $\sum_{i \in V} w_i x_i \leq C$ ， $x_i \in \{0,1\}$  的限制，故我們可知，控制點選擇問題—最大總流量模型

可轉化為 0-1 knapsack problem。

然而，由於 0-1 knapsack problem 目前已有許多現成的演算法可用，故不需另行設計演算法，因此，我們的研究主要放在找控制點選擇問題—最大涵蓋率模型、最大加權涵蓋率模型的解。

#### 4.1.2 問題分析—最大涵蓋率模型

以下我們將證明控制點選擇問題—最大涵蓋率模型為 NP-complete 問題。

To show that CPSP-Coverage is NP-complete we must demonstrate two things: (1) that CPSP-Coverage is in NP; and (2) that every language  $A$  in NP is polynomial time reducible to CPSP-Coverage. To do the second we show that a known NP-complete problem, hitting set problem[21], is polynomial time reducible to CPSP-Coverage.

Definition of the Hitting Set Problem (HSP) [21] :

Assume that a ground set  $S$  and a collection  $C$  of subsets  $\{C_1, C_2, C_3, \dots, C_n\}$  are given, where  $C_i$  is a subset of  $S$  and a positive integer  $k \leq |S|$ . The hitting set problem is to find if there is some subset  $S' \subseteq S$  with  $|S'| \leq k$  such that  $(C_i \cap S') \neq \emptyset$



for all  $i=1,2,\dots,n$ . In other words,  $S'$  hits (intersects) every subset in  $C$ .

For example,  $S = \{1, 2, 3, 4\}$  and  $C = \{\{1, 2, 3\}, \{4\}\}$ . The hitting-set for  $S$  and  $C$  consists of  $\{1, 4\}, \{2, 4\}, \{3, 4\}, \{1, 2, 4\}, \{1, 3, 4\}, \{2, 3, 4\}$  and  $\{1, 2, 3, 4\}$  if  $k=4$ . If  $k=2$ , the hitting-set contains  $\{1, 4\}, \{2, 4\}$  and  $\{3, 4\}$ .

The following is the proof that the problem CPSP-Coverage is NP-complete :

(1) CPSP-Coverage is in NP:

We first show that CPSP-Coverage  $\in$  NP, suppose we are given a graph  $G=(V,E)$  and two integer  $k$  and  $c$ . There is a collection  $T$  of subsets  $\{t_1, t_2, \dots, t_m\}$  given, where  $t_i$  is a subset of  $V$ . And there is an integer cost  $w_i$  for all  $i \in V$ . The certificate we choose is the Control Point Set  $V' \subseteq V$  itself. The verification algorithm affirms that  $\sum_{i \in V'} w_i \leq c$ , and then it checks that  $V'$  contains at least one element from each subset in  $T$ . It means that  $(t_i \cap V') \neq \emptyset$  for all  $i=1,2,\dots,m$ . At last it checks that, for each edge  $(u,v) \in E$ , that  $u \in V'$  or  $v \in V'$ , and  $|\bigcup(u,v)|=k$ . The verification can be performed straightforwardly in polynomial time.

(2) CPSP-Coverage is NP-hard:

To prove that CPSP-Coverage is NP-hard, we show that HSP is polynomial time reducible to CPSP-Coverage. To do this reduction, look at the definitions of the two problems, and determine how to construct CPSP-Coverage from HSP.

Let  $U, S, C$  be an instance of HSP. We construct the corresponding instances of

CPSP-Coverage as follows. We form the graph  $G=(V,E)$ , where  $E=\{(i,j):i,j\in S'\}$ , and we define  $C$  in  $U = T$  in  $G$ ,  $S'$  in  $S = V'$  in  $V$ , and cost function  $c(i)=\begin{cases} 0 & \text{if } i\in S', \\ 1 & \text{otherwise.} \end{cases}$  The instance of CPSP-Coverage is then  $\langle G,T,0\rangle$ , which is easily formed in polynomial time.

We now show that a ground set  $S$  has a hitting set if and only if graph  $G$  has a Control Point Set of cost at most 0. Suppose that a ground set  $S$  has a hitting set  $S'$ . Each node in  $S'$  has cost 0 and  $\bigcup(i,j)=E$ . And  $(C_i\cap S')\neq\emptyset$  in  $U$ , thus  $(t_i\cap V')\neq\emptyset$  in  $G$ . Conversely, suppose that graph  $G$  has a Control Point Set  $V'$  of cost at most 0. Since each cost of the node in  $V$  is 0 or 1, the total cost of the hitting set  $S'$  is exactly 0 and node on the hitting set must have cost 0. And  $(t_i\cap V')\neq\emptyset$  in  $G$ , thus  $(C_i\cap S')\neq\emptyset$  in  $U$ . Therefore,  $S'$  contains only nodes in  $V$ . We conclude that  $S'$  is a hitting set in  $S$ .

由以上證明，可知控制點選擇問題—最大涵蓋率模型為 NP-complete 問題，故我們將在之後的章節，提出一個啟發式演算法，來找一個 sub-optimal solution，以解決此問題，並在後續的實驗中，和最佳解做比較，以展示我們解法的成效。

## 4.2 解決方案的演算法

由於 CPSP-Flow 數學模型已有許多現成的 0-1 knapsack problem 之解決方案演算法可用，故在此章節，我們僅提出 CPSP-Coverage 及 CPSP-Utility 數學模型的解決方案演算法。但在那之前，我們仍會針對 CPSP-Flow，舉一例以驗證 3.4.2 的推論。

首先，我們需要定義幾個評估指標，做為 sub-optimal solution 與 optimal solution 比較的依據。在 3.3.2 中，我們已經定義 MC\_distance 為「某一個行動節點到它最近的

控制點之間的距離」。然而，由於在同一條 edge 上，移動方向相反的行動節點，分別與其最近的控制點之間的距離不一定相同，故在之後的實驗中，我們都將假設「在每條 edge 上，往返兩方向各有一個行動節點」，來對與 MC\_distance 相關的評估指標進行計算。下表 4.1 為控制點選擇問題 (CPSP) 的相關評估指標之定義。

表 4.1：控制點選擇問題 (CPSP) 的評估指標。





Name	Definition
number of covered flows	被選為控制點的所有 node 所涵蓋的流量數之和。
number of covered edges	被選為控制點的所有 node 所涵蓋的 edge 總數。
total weight of covered edges	表示被選為控制點的所有 node 所涵蓋的 edge 權重之和。
mean_MC_distance	任一個行動節點到它最近的控制點之平均距離，其距離以「edge 數」表示，算法為「所有行動節點的 MC_distance 之總和/行動節點的總數量」。
max_MC_distance	任一個行動節點到它最近的控制點之最大距離 (edge 數)，即「所有行動節點的 MC_distance 之最大值」。
mean_weighted_MC_distance	與上述的 mean_MC_distance 相同，惟距離以「edge 權重之和」表示，而非 edge 數，並假設「trail 末端的 edge 之權重皆為 1」。
max_weighted_MC_distance	與上述的 max_MC_distance 相同，惟距離以「edge 權重之和」表示，而非 edge 數，並假設「trail 末端的 edge 之權重皆為 1」。

值得注意的是，在我們所提出的3個控制點選擇問題之模型中，sub-optimal solution 在表 4.1 所列的 MC\_distance 相關評估指標上，有機會優於 optimal solution 所求，這其實是可以預料的，因為所謂的 optimal solution 是針對 optimization objective 而言，故當 objective function 不是 MC\_distance 時，這樣的結果就可能會發生。

此外，MC\_distance 相差 1，看起來好像不多，但走路的話，可能是登山客走數十分鐘，甚至是數小時的距離，故在之後的範例及實驗中，對於比較 MC\_distance 相關評估指標的差異時，不可輕忽這些看似微小的差值。

#### 4.2.1 CPSP-Flow 解決方案

如 4.1.1 所述，CPSP-Flow 為 0-1 knapsack problem，故可利用現成的 0-1 knapsack problem 之演算法來求解。我們以圖 4.1 為登山健行路徑之環境狀態為例，其中包含  $t_1$  到  $t_4$  共 4 條 trail，及  $\{A', A, \dots, J, J'\}$  共 18 個 node。

-   $t_1 : A', A, B, C, D, D'$
-   $t_2 : B', B, E, F, J, J'$
-   $t_3 : C', C, E, F, G, G'$
-   $t_4 : H', H, G, I, I'$

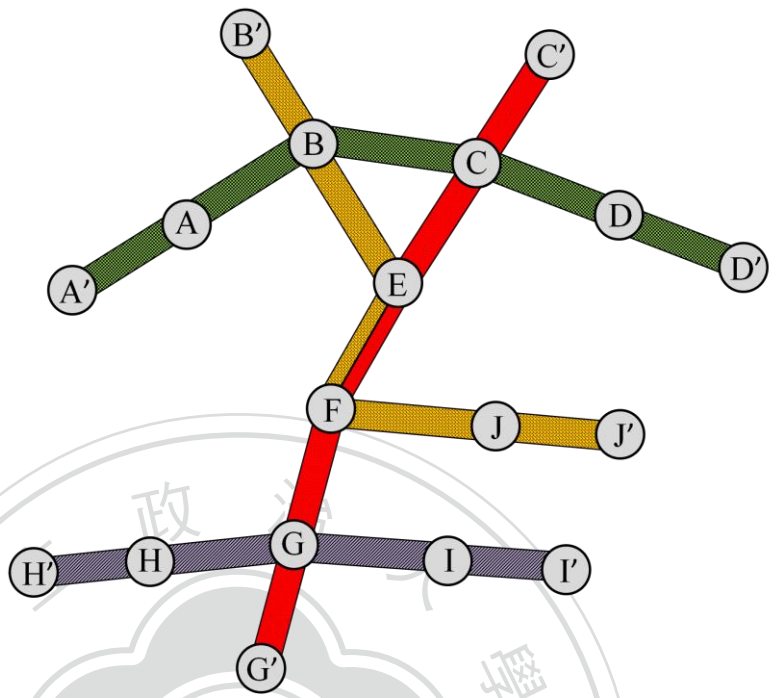






圖 4.1：登山健行路徑範例。

我們假設「所有健行路徑的端點必有控制點」，故只取出圖 4.1 中，端點以外的 node 集合來做為候選控制點之集合 (如圖 4.2)。在此例中，我們分別指定每條 trail 一個固定的登山客流量數，故在候選控制點旁所標示的數值分別代表其 cost 及 flow。

- Given available resources = 6

-   $t_1$  : A', A, B, C, D, D' (Flow= 1)
-   $t_2$  : B', B, E, F, J, J' (Flow= 5)
-   $t_3$  : C', C, E, F, G, G' (Flow= 5)
-   $t_4$  : H', H, G, I, I' (Flow= 1)

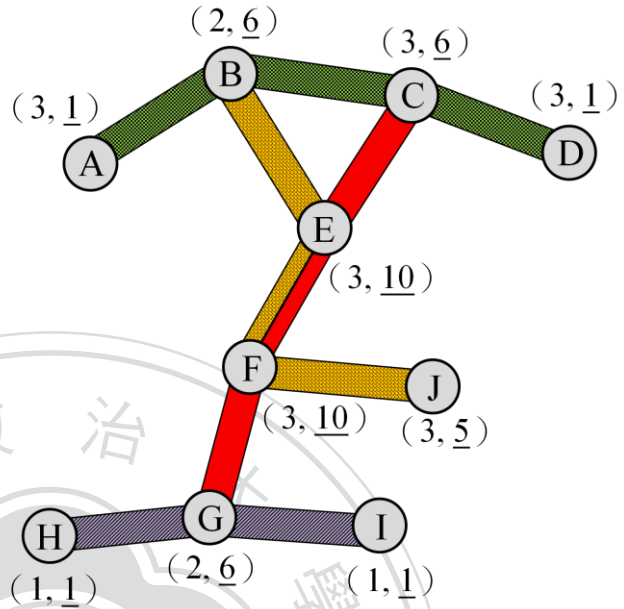
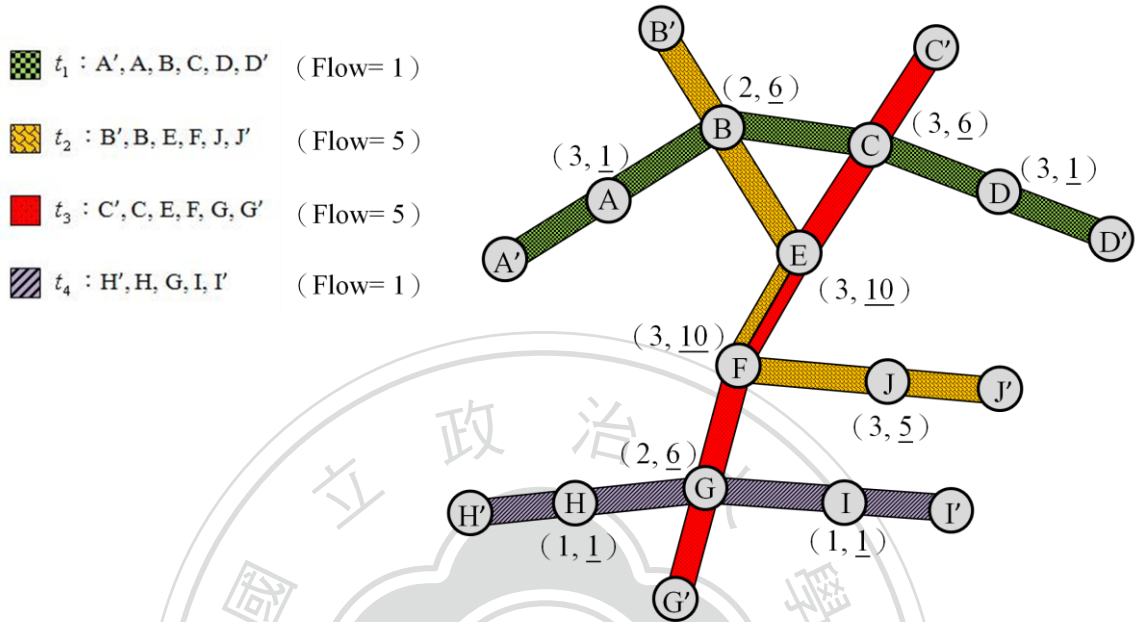


圖 4.2 : CPSP-Flow 模型求 sub-optimal 之範例—初始狀態。

如圖 4.3，CPSP-Flow 利用 0-1 knapsack problem 的 sub-optimal solution 所選出的控制點為 {E, F}，涵蓋的總流量數為 20 (各端點的流量數忽略不計)，皆與 optimal solution 所求相同。故由這些被選出的控制點集合所計算出的 mean\_MC\_distance、max\_MC\_distance 也會與 optimal solution 所求相同。

- Given available resources = 6



0-1 knapsack's solution : {E, F}  
 total consumed cost = 6  
 number of covered flows = 20  
 mean\_MC\_distance = 1.72  
 max\_MC\_distance = 3

Optimal solution : {E, F}  
 total consumed cost = 6  
 number of covered flows = 20  
 mean\_MC\_distance = 1.72  
 max\_MC\_distance = 3

圖 4.3 : CPSP-Flow 模型—sub-optimal solution vs. optimal solution。

由此例可知，CPSP-Flow 可能導致控制點的佈建地點過度集中在某一區域，而未能完全涵蓋所有健行路徑（此例中只涵蓋到  $t_2$ 、 $t_3$  這兩條路徑），而這對於在被漏掉的路徑上移動的行動節點（登山客）而言，由於不易獲得控制網路的幫助，故需要較長的時間，才能被搜尋代理人找到，甚至可能永遠無法被搜尋到。

## 4.2.2 CPSP-Coverage 解決方案

為了找 CPSP-Coverage 的 sub-optimal solution，我們提出一個啟發式演算法，並將此其命名為 Node Potential First (NPF) 演算法，下表 4.2 將列出在 NPF 演算法中所運用到的參數及其意義。

表 4.2：Node Potential First (NPF) 演算法之參數表。

Notation	Definition
edge_degree	node 所連到 edge 的 degree 數。
node_weight	node 被選為控制點，所需花費的 cost。
selected_edge_degree	表示此 node 中，有幾條 edge 已被選取，初始值為 0。
unselected_edge_degree	$edge\_degree - selected\_edge\_degree$ ，表示選擇此 node 所能增加的 covered edge 數。
node_potential	$node\_potential = (unselected\_edge\_degree / node\_weight)$ ，表示此 node 所能 cover edges 的潛力。
trail_hit_count	node 上所通過的 unmark trail 數。

此演算法的設計重點，如下：

- 所選的候選控制點之 node\_weight 總和，不可超過 available resources。
- 優先選擇 node\_potential、trail\_hit\_count 值最大，而 node\_weight 值最小的候選控制點。

下圖 4.4 為 Node Potential First (NPF) 演算法之運作流程，我們將優先選擇



node\_potential、trail\_hit\_count 的值最大，而 node\_weight 最小的候選控制點，並且，為了盡快涵蓋所有的 trail，我們也將在每次選擇一個候選控制點後，更新 node\_potential、trail\_hit\_count 的值，以確保所選的下一個候選控制點能涵蓋到新的 edges 及 trails，如此一來，除了能避免重覆計算所涵蓋到的 edge 數量外，也能達到最快速地涵蓋所有 trail 的目的。

```
Candidate ListV ← all nodes
Iterate until Committed Cost >= Available Resources
{
    Deleting the node from the candidate ListV with node_weight > Remain Resources.
    Selecting the node from the candidate ListV with the largest value of
node_potential, trail_hit_count in descending order, and smallest value of node_weight in
ascending order.
    Updating the value of node_potential and trail_hit_count of every node by marking
covered trails.
}
```

圖 4.4：Node Potential First (NPF) 演算法之運作流程。

以下，我們將以 4.2.1 之環境（圖 4.1）為例，解釋 NPF 演算法的運作流程（圖 4.5 – 圖 4.11）。圖 4.5 為初始狀態，而在候選控制點旁所標示的數值則代表 node\_weight。

- Given available resources = 6

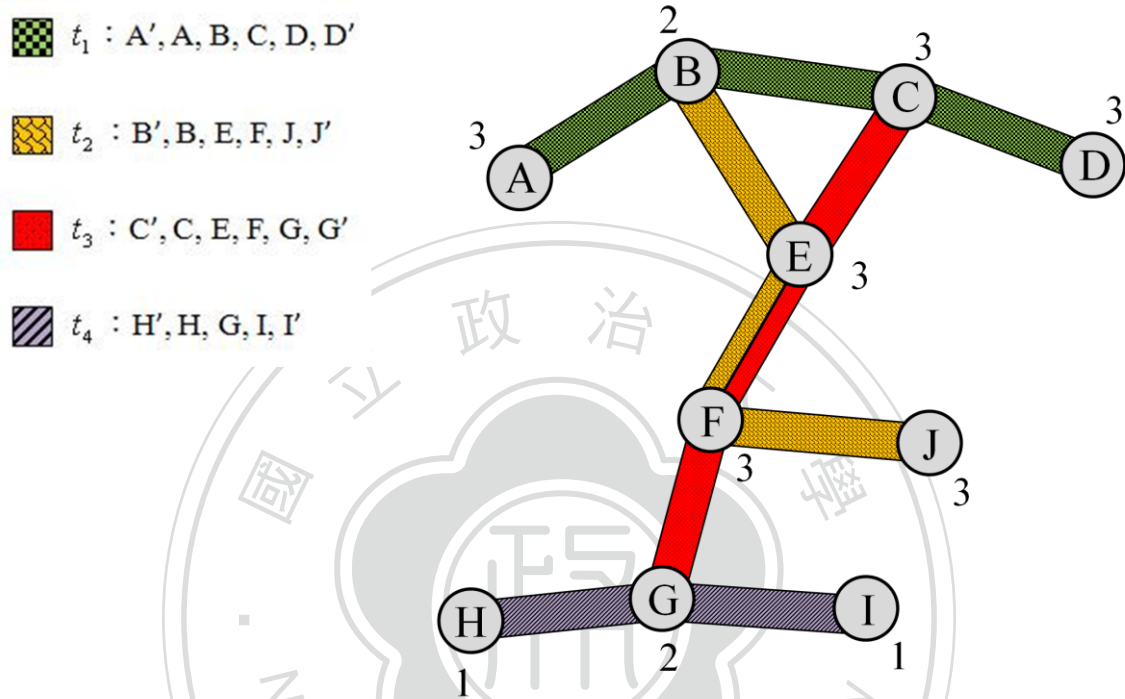
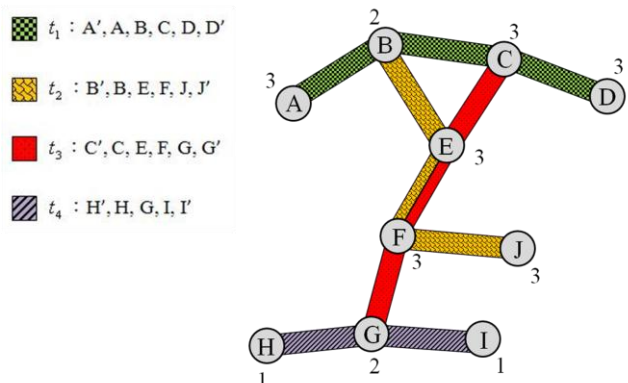


圖 4.5：CPSP-Coverage 模型求 sub-optimal 之範例—初始狀態。

如圖 4.6 所示，在第一輪計算中，先將 node\_potential (1st)、trail\_hit\_count(1st) 由大至小排序，而 node\_weight 由小至大排序，之後，我們將優先選擇 node\_potential (1st)、trail\_hit\_count(1st) 最大，而 node\_weight 最小的候選點，來做為我們的控制點，此例中 B、G 具有相同的優先權，因此，我們任選一個，即以 B 做為第一個控制點，並在更新 node\_potential 與 trail\_hit\_count 的值後，進入下一輪計算。







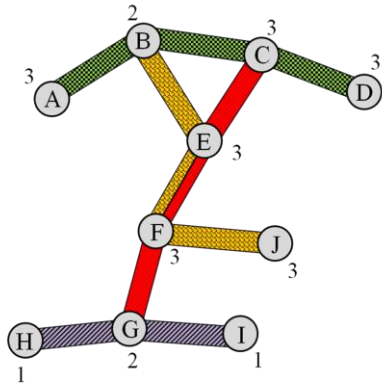
Given available resources = 6  
 candidate ListV = {A, B, ..., J}

	A	B	C	D	E	F	G	H	I	J
edge_degree	1	3	3	1	3	3	3	1	1	1
node_weight	3	2	3	3	3	3	2	1	1	3
node_potential(1st)	1/3	3/2	1	1/3	1	1	3/2	1	1	1/3
trail_hit_count(1st)	1	2	2	1	2	2	2	1	1	1
node_potential(2nd)	0	0	2/3	1/3	2/3	1	3/2	1	1	1/3
trail_hit_count(2nd)	0	0	1	0	1	1	2	1	1	0

圖 4.6：CPSP-Coverage 模型求 sub-optimal solution—第一輪比較後，選 B 為控制點。

依此類推，在第二輪（圖 4.7）、第三輪（圖 4.8）、第四輪（圖 4.9）計算中，將分別選擇 G、H、I 做為控制點，最後，在第五輪（圖 4.10）計算中，由於 available resources 已用盡，故不選擇任何候選點做為控制點，並結束 NPF 演算法。

-   $t_1$  : A', A, B, C, D, D'
-   $t_2$  : B', B, E, F, J, J'
-   $t_3$  : C', C, E, F, G, G'
-   $t_4$  : H', H, G, I, I'

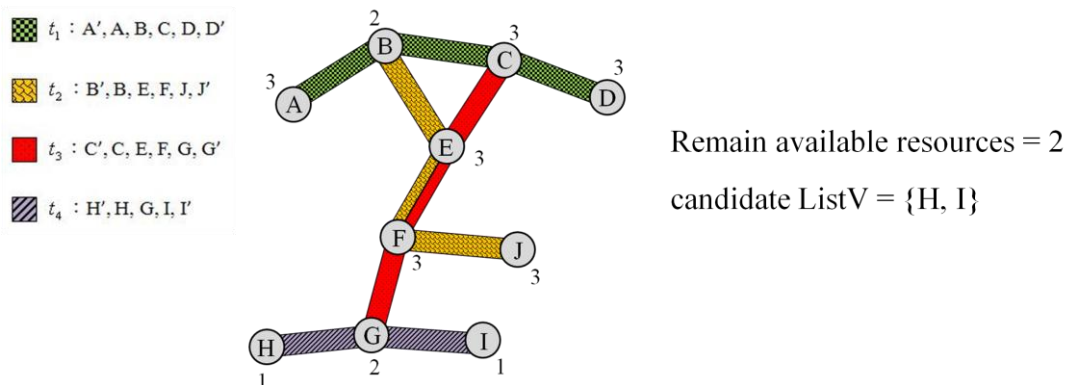


Remain available resources = 4  
 candidate ListV = {A, C-J}

	A	B	C	D	E	F	G	H	I	J
node_weight	3	2	3	3	3	3	2	1	1	3
node_potential(2nd)	0	0	2/3	1/3	2/3	1	3/2	1	1	1/3
trail_hit_count(2nd)	0	0	1	0	1	1	2	1	1	0
node_potential(3rd)	0	0	2/3	1/3	2/3	2/3	0	0	0	1/3
trail_hit_count(3rd)	0	0	0	0	0	0	0	0	0	0

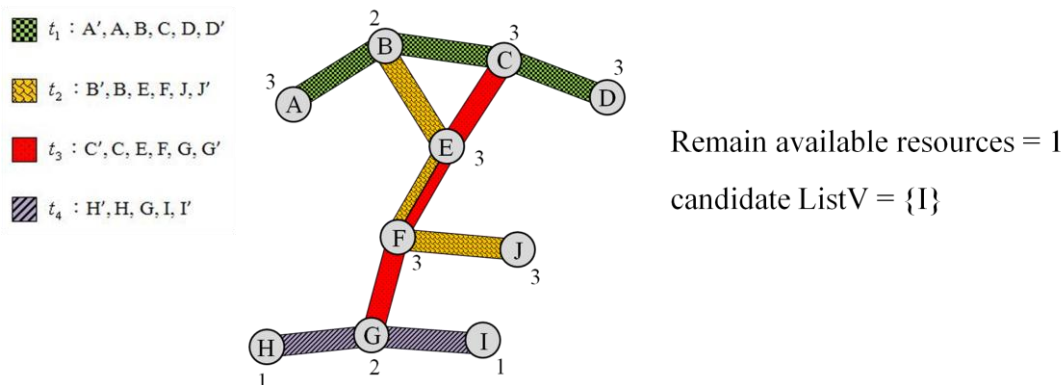
圖 4.7：CPSP-Coverage 模型求 sub-optimal solution—第二輪比較後，選 G 為控制點。





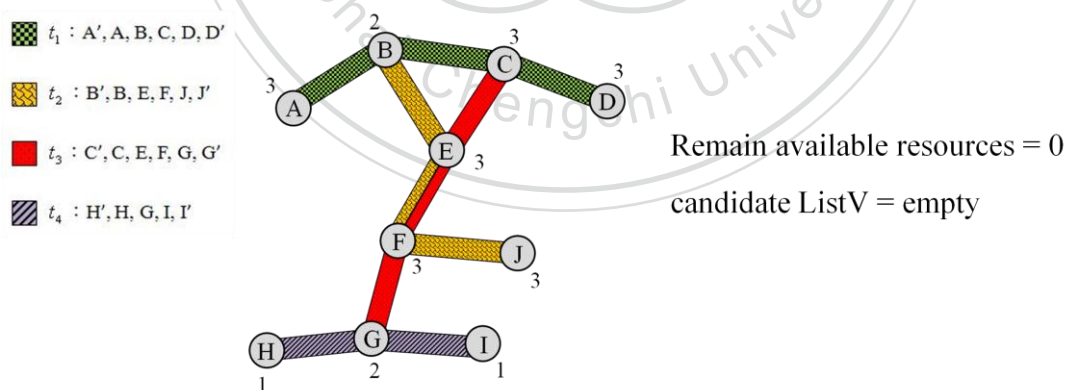
	A	B	C	D	E	F	G	H	I	J
node_weight	3	2	3	3	3	3	2	1	1	3
node_potential(3rd)	0	0	2/3	1/3	2/3	2/3	0	0	0	1/3
trail_hit_count(3rd)	0	0	0	0	0	0	0	0	0	0
node_potential(4th)	0	0	2/3	1/3	2/3	2/3	0	0	0	1/3
trail_hit_count(4th)	0	0	0	0	0	0	0	0	0	0

圖 4.8：CPSP-Coverage 模型求 sub-optimal solution—第三輪比較後，選 H 為控制點。



	A	B	C	D	E	F	G	H	I	J
node_weight	3	2	3	3	3	3	2	1	1	3
node_potential(4th)	0	0	2/3	1/3	2/3	2/3	0	0	0	1/3
trail_hit_count(4th)	0	0	0	0	0	0	0	0	0	0
node_potential(5th)	0	0	2/3	1/3	2/3	2/3	0	0	0	1/3
trail_hit_count(5th)	0	0	0	0	0	0	0	0	0	0

圖 4.9：CPSP-Coverage 模型求 sub-optimal solution—第四輪比較後，選 I 為控制點。





	A	B	C	D	E	F	G	H	I	J
node_weight	3	2	3	3	3	3	2	1	1	3
node_potential(5th)	0	0	2/3	1/3	2/3	2/3	0	0	0	1/3
trail_hit_count(5th)	0	0	0	0	0	0	0	0	0	0


圖 4.10：CPSP-Coverage 模型求 sub-optimal solution—第五輪比較後，不選任何點。


由圖 4.11 可知，CPSP-Coverage 利用 NPF 演算法求得的 sub-optimal solution，其 number of covered edges 為 14，比 optimal solution 少涵蓋 1 個 edge，而 mean\_MC\_distance 則低於 optimal solution 所求，表示任一個行動節點到它最近的控制點之平均距離，比 optimal solution 所求還要近 0.02 個 edge 的距離，此外，max\_MC\_distance 與 optimal solution 所求相同。

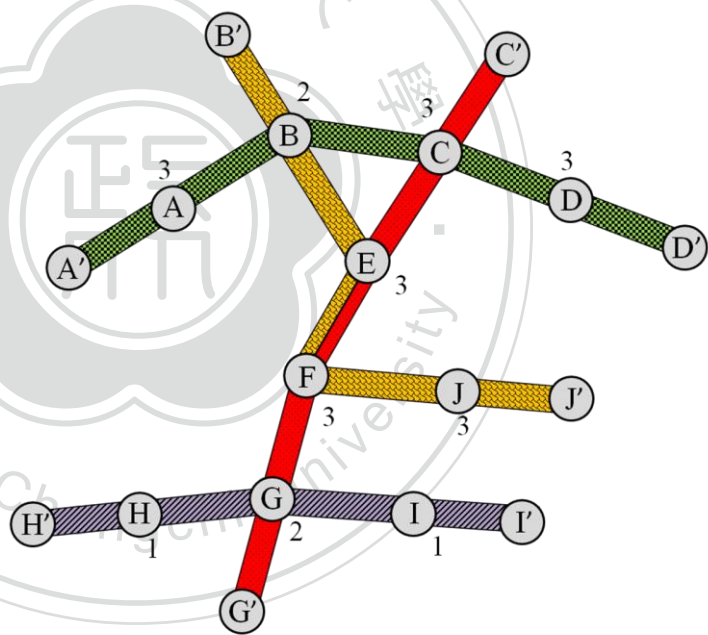
- Given available resources = 6

  $t_1$  : A', A, B, C, D, D'

  $t_2$  : B', B, E, F, J, J'

  $t_3$  : C', C, E, F, G, G'

  $t_4$  : H', H, G, I, I'



Our solution : {B, G, H, I}  
 total consumed cost = 6  
 number of covered edges = 14  
 mean\_MC\_distance = 1.56  
 max\_MC\_distance = 3

Optimal solution : {B, F, H}  
 total consumed cost = 6  
 number of covered edges = 15  
 mean\_MC\_distance = 1.58  
 max\_MC\_distance = 3

圖 4.11：CPSP-Coverage 模型—sub-optimal solution vs. optimal solution。

### 4.2.3 CPSP-Utility 解決方案

在此章節，我們將對 NPF 演算法 (圖 4.4) 稍作修改，稱為 NPF-U 演算法，來解 CPSP-Utility 問題。NPF-U 與 NPF 的差異在於 node\_potential 的計算改為

「 $\text{unselected\_edge\_weight} / \text{node\_weight}$ 」。

以下，我們將以 4.2.1 之環境 (圖 4.1) 為例，解釋 NPF-U 演算法的運作流程 (圖 4.12 - 圖 4.18)。圖 4.12 為初始狀態，在候選控制點旁所標示的數值代表 node\_weight，而在 edge 上標示的數值則代表 edge\_weight。

- Given available resources = 6

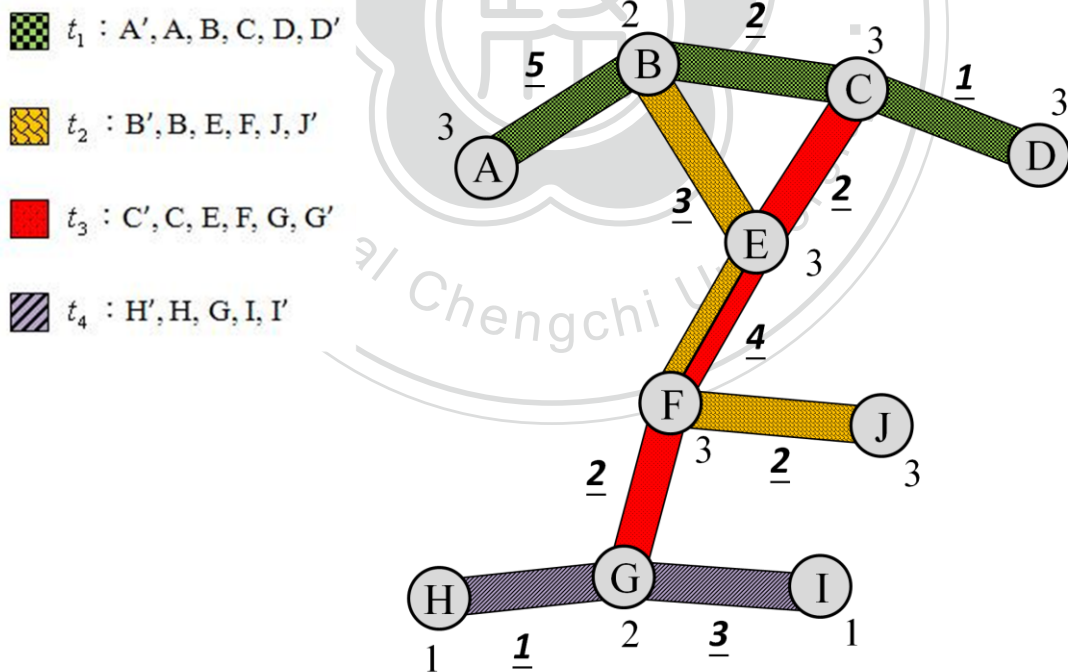


圖 4.12：CPSP-Utility 模型求 sub-optimal solution—初始狀態。



如圖 4.13 所示，在第一輪計算中，先將 node\_potential (1st)、trail\_hit\_count(1st) 由大至小排序，而 node\_weight 由小至大排序，之後，我們將優先選擇 node\_potential (1st)、trail\_hit\_count(1st) 最大，而 node\_weight 最小的候選點，來做為我們的控制點，故選擇 B 做為第一個控制點，並在更新 node\_potential 與 trail\_hit\_count 的值後，進入下一輪計算。

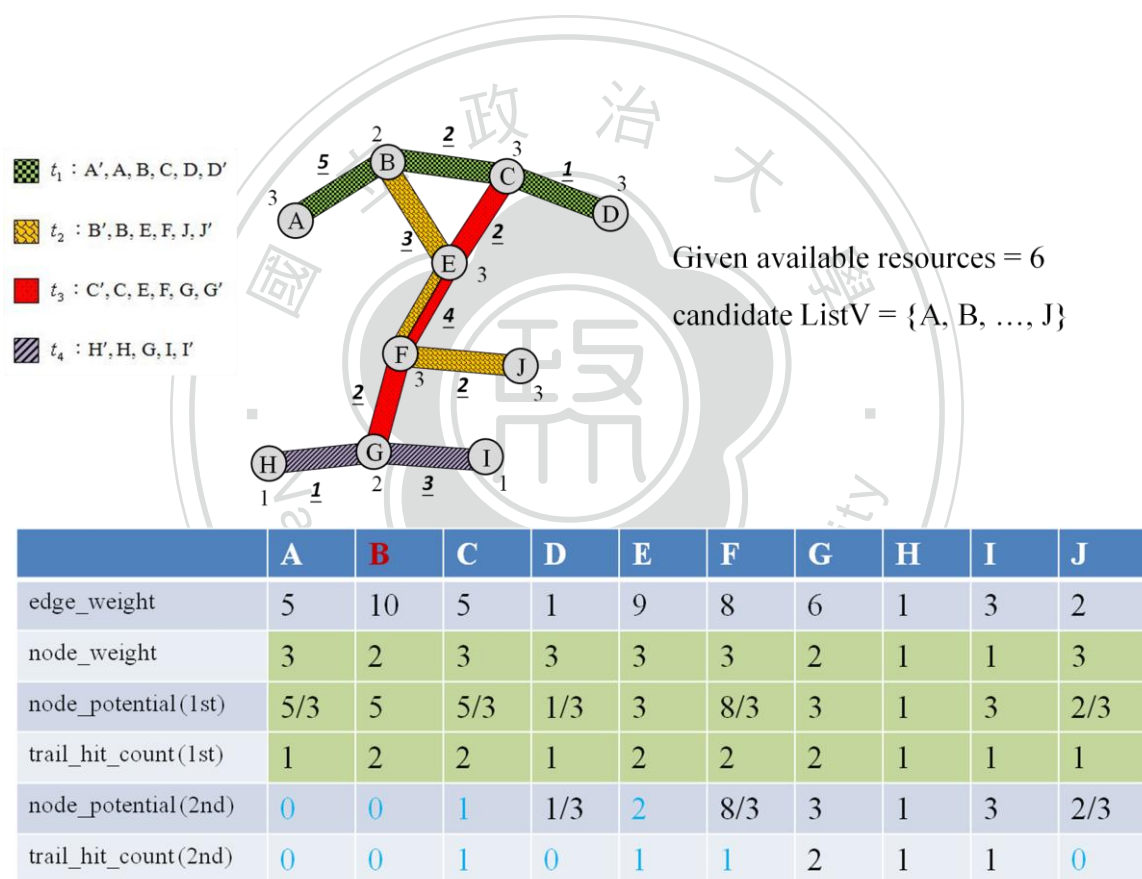


圖 4.13：CPSP-Utility 模型求 sub-optimal solution—第一輪比較後，選 B 為控制點。

依此類推，在第二輪（圖 4.14）、第三輪（圖 4.15）、第四輪（圖 4.16）計算中，將分別選擇 G、H、I 做為控制點，最後，在第五輪（圖 4.17）計算中，由於 available

resources 已用盡，故不選擇任何候選點做為控制點，並結束 NPF-U 演算法。

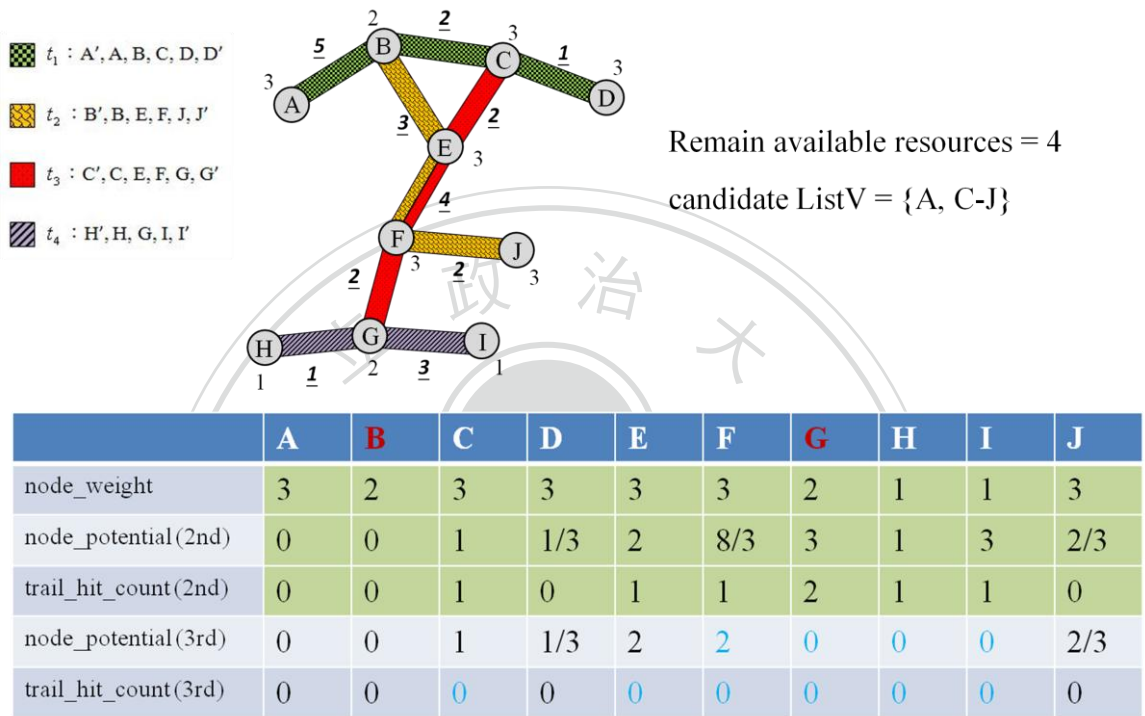
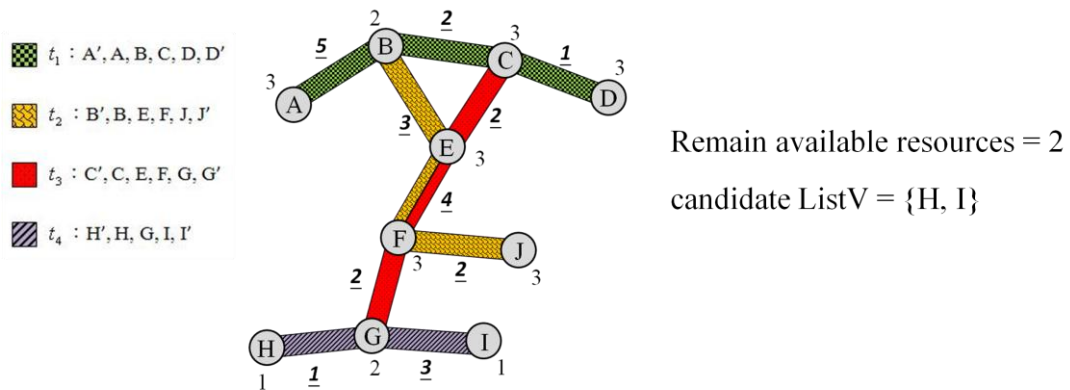
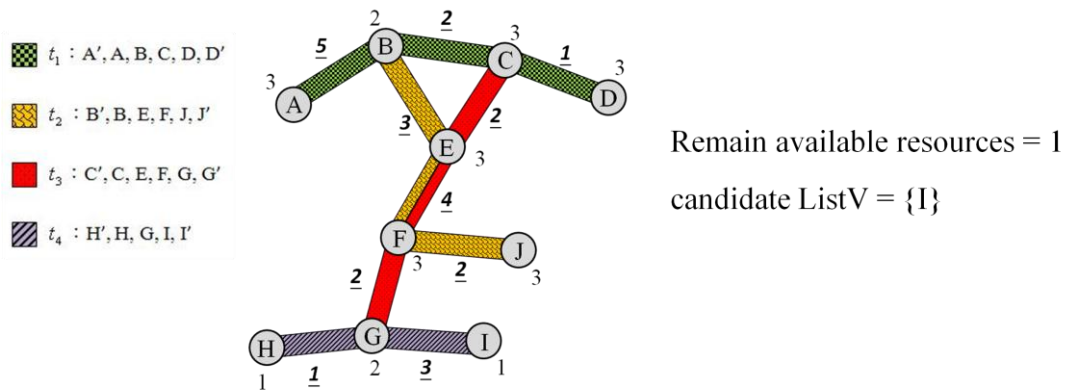


圖 4.14：CPSP-Utility 模型求 sub-optimal solution—第二輪比較後，選 G 為控制點。



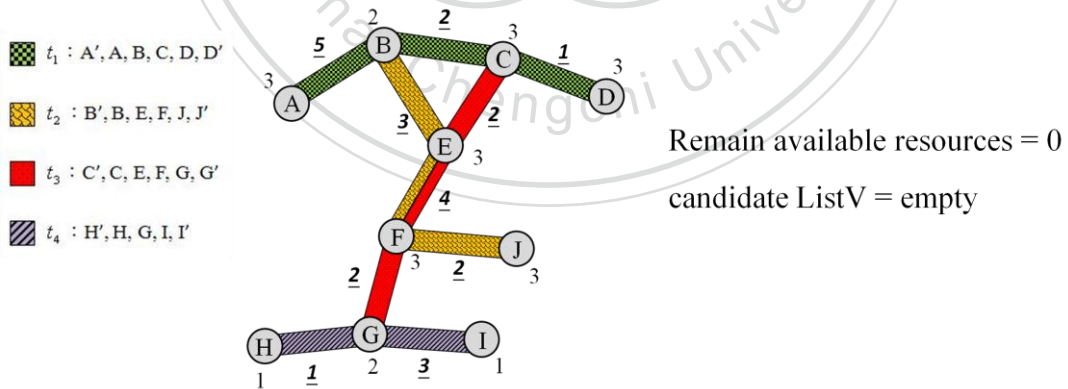
	A	B	C	D	E	F	G	H	I	J
node_weight	3	2	3	3	3	3	2	1	1	3
node_potential(3rd)	0	0	1	1/3	2	2	0	0	0	2/3
trail_hit_count(3rd)	0	0	0	0	0	0	0	0	0	0
node_potential(4th)	0	0	1	1/3	2	2	0	0	0	2/3
trail_hit_count(4th)	0	0	0	0	0	0	0	0	0	0

圖 4.15：CPSP-Utility 模型求 sub-optimal solution—第三輪比較後，選 H 為控制點。



	A	B	C	D	E	F	G	H	I	J
node_weight	3	2	3	3	3	3	2	1	1	3
node_potential(4th)	0	0	1	1/3	2	2	0	0	0	2/3
trail_hit_count(4th)	0	0	0	0	0	0	0	0	0	0
node_potential(5th)	0	0	1	1/3	2	2	0	0	0	2/3
trail_hit_count(5th)	0	0	0	0	0	0	0	0	0	0

圖 4.16：CPSP-Utility 模型求 sub-optimal solution—第四輪比較後，選 I 為控制點。

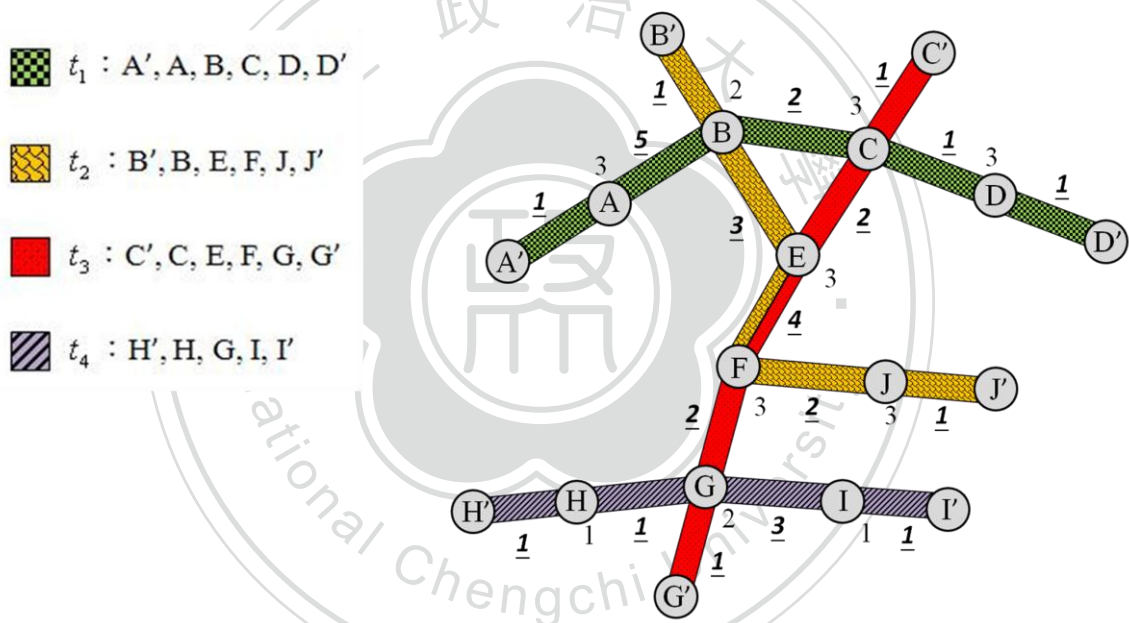


	A	B	C	D	E	F	G	H	I	J
node_weight	3	2	3	3	3	3	2	1	1	3
node_potential(5th)	0	0	1	1/3	2	2	0	0	0	2/3
trail_hit_count(5th)	0	0	0	0	0	0	0	0	0	0

圖 4.17：CPSP-Utility 模型求 sub-optimal solution—第五輪比較後，不選任何點。

由圖 4.18 可知，CPSP-Utility 利用 NPF-U 演算法求得的 sub-optimal solution，其能涵蓋的所有 edge 權重總和為 24，低於 optimal solution 所求 5 個 edge\_weight，而 mean\_weighted\_MC\_distance 則高於 optimal solution 所求 0.03 個 edge\_weight。此外，max\_weighted\_MC\_distance 與 optimal solution 所求相同。

- Given available resources = 6



Our solution : {B, G, H, I}  
 total consumed cost = 6  
 total weight of covered edges = 24  
 mean\_weighted\_MC\_distance = 2.81  
 max\_weighted\_MC\_distance = 7

Optimal solution : {B, F, I}  
 total consumed cost = 6  
 total weight of covered edges = 29  
 mean\_weighted\_MC\_distance = 2.78  
 max\_weighted\_MC\_distance = 7

圖 4.18 : CPSP-Utility 模型—sub-optimal solution vs. optimal solution。

## 第5章

### 效能評估

這一章中，將針對我們所提出的控制點選擇問題之模型做幾個實驗，以評估不同模型下的搜尋行動代理人之效率。

#### 5.1 小型問題效能評估

由於 CPSP-Coverage 及 CPSP-Utility 皆為 NP-complete 問題，因此，我們將分別利用 4.2.2 與 4.2.3 所提出的 NPF 與 NPF-U 啟發式演算法來求得次佳解，並針對在 4.2 中的表 4.1 所列之評估指標，來與其最佳解做比較，以分析各別演算法與最佳解之間的效能差異。因最佳解所耗費的計算資源龐大，故只適合少數的小型問題。

##### 5.1.1 實驗環境的參數設定

我們將隨機產生 5 個 Graph，參數設定如下：

- The total number of candidate control points = 20。
- The total number of trails = 8。
- Available resources = 12。
- The cost of each candidate control point = 1~5。
- The edge\_weight between candidate control point = 1~5。

##### 5.1.2 實驗結果與分析

下圖 5.1 圖 5.2 為 CPSP-Coverage 的實驗結果，分別列出 NPF 演算法與 optimal

solution 所求得的 number of covered edges、mean\_MC\_distance，與 optimal solution 所求相比較。而由於 max\_MC\_distance 與 optimal solution 所求相同，故不以圖表列出。

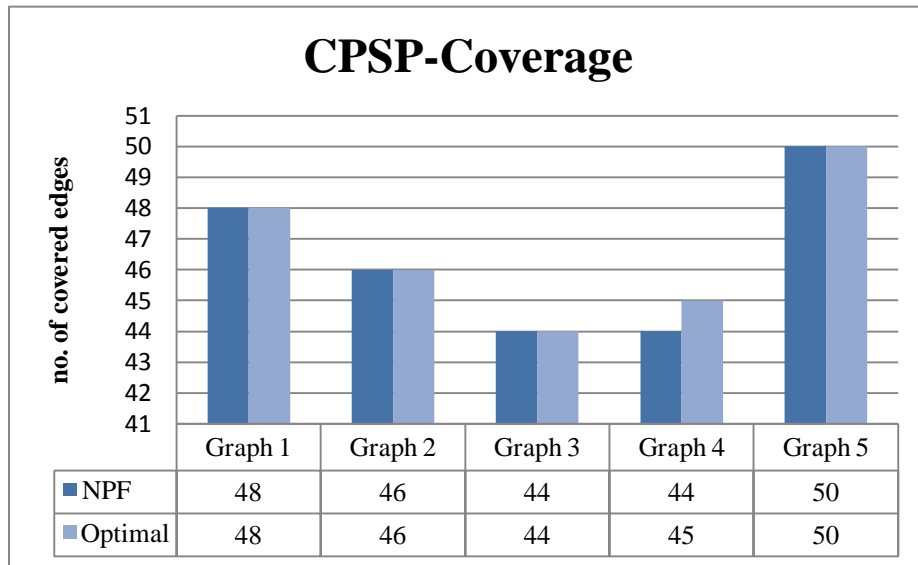


圖 5.1：CPSP-Coverage—NPF vs. Optimal solution (no. of covered edges)。

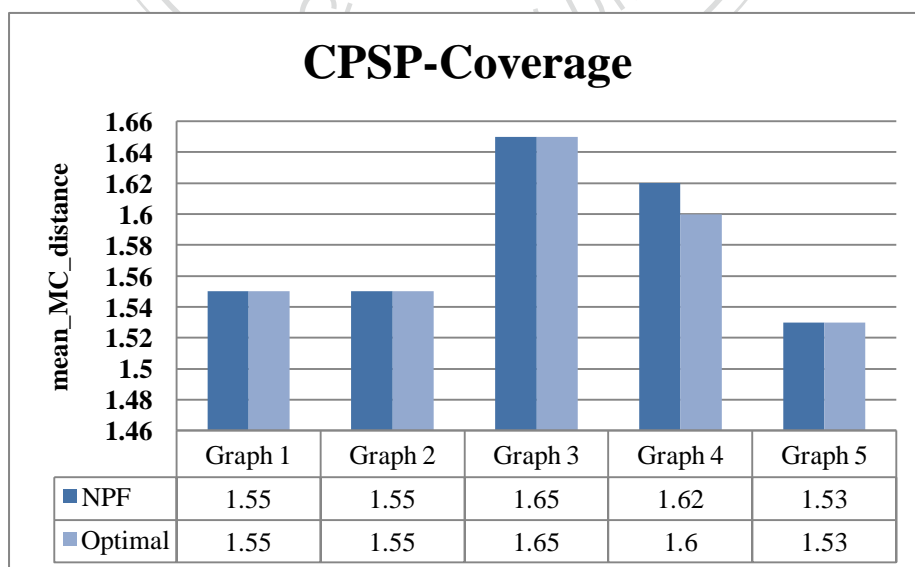


圖 5.2：CPSP-Coverage—NPF vs. Optimal solution (mean\_MC\_distance)。

由圖 5.1 可知，在 Graph 4 中 NPF 演算法所求的 number of covered edges 比 optimal solution 少 1 個 edge，這是因為 NPF 演算法會優先選擇 node\_potential、trail\_hit\_count 較高的候選控制點，然而，就如同 0-1 knapsack problem，這樣做並不能保證所得到的 profit 最高，因此，在某些情況下就會造成 NPF 演算法的 number of covered edges 效率低於 optimal solution。

由圖 5.2 可知，在 Graph 4 中 NPF 演算法所求的 mean\_MC\_distance 比 optimal solution 高 0.02 個 edge，這是由於在此 Graph 中，optimal solution 所選的候選控制點在所有 trail 上出現的比例較 NPF 演算法所選的來得高，故較容易找到距離行動節點較近的控制點，因此，在此情況下就會造成 optimal solution 所求的 mean\_MC\_distance 效率優於 NPF 演算法。

下圖 5.3 圖 5.4 為 CPSP-Utility 的實驗結果，分別列出 NPF-U 演算法與 optimal solution 所求得的 total weight of covered edges、mean\_weighted\_MC\_distance。而由於 max\_weighted\_MC\_distance 與 optimal solution 所求相同，故不以圖表列出。

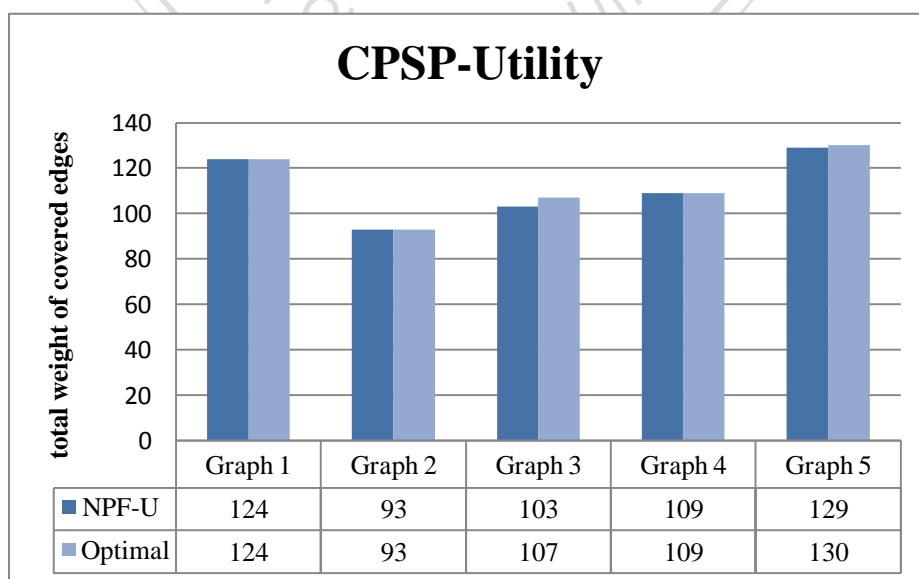


圖 5.3：CPSP-Utility—NPF-U vs. Optimal solution (total weight of covered edges)。



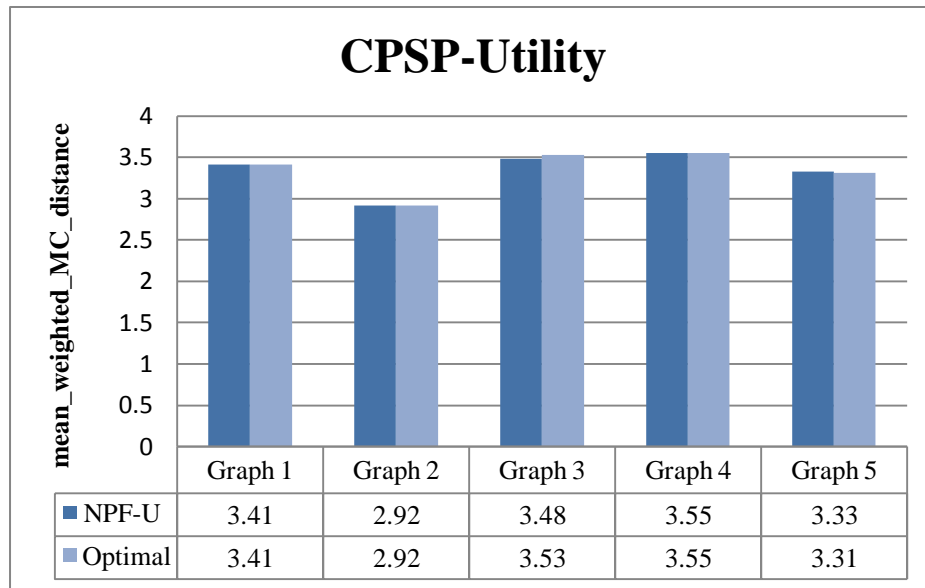


圖 5.4：CPSP-Utility—NPF-U vs. Optimal solution (mean\_weighted\_MC\_distance)。

由圖 5.3 可知，在 Graph 3、Graph 5 中，NPF-U 演算法所求的 total weight of covered edges 分別比 optimal solution 少 4 個、1 個 edge\_weight，原因同上圖 5.1 所述，這也是因為 NPF-U 演算法的控制點選擇機制所導致。

而由圖 5.4 可知，在 Graph 3 中，NPF-U 演算法所求的 mean\_weighted\_MC\_distance 比 optimal solution 低 0.05 個 edge\_weight，這是由於在此 Graph 中，NPF-U 演算法所選的候選控制點在所有 trail 上出現的比例較 optimal solution 所選的來得高，故所求的 mean\_weighted\_MC\_distance 效率高於 optimal solution。反之，在 Graph 5 中，NPF-U 演算法所選的候選控制點在所有 trail 上出現的次數與 optimal solution 相同，故考慮此 Graph 中 edge\_weight 的差異下，NPF-U 演算法的 mean\_weighted\_MC\_distance 效率略低於 optimal solution。

為了分析上圖 5.1-圖 5.4 中，NPF、NPF-U 所求與最佳解的差異程度，我們將表 4.1 的評估指標做正規化，產生以下幾個新的評估指標（單位為%），定義如下：

- Error in covered edge : ( number of covered edges (optimal) – number of covered edges (heuristic)) / number of covered edges (optimal) ◦
- Error in total weight of covered edges : (total weight of covered edges (optimal) – total weight of covered edges (heuristic)) / total weight of covered edges (optimal) ◦
- Error in mean\_MC\_distance : (mean\_MC\_distance (heuristic) – mean\_MC\_distance (optimal)) / mean\_MC\_distance (optimal) ◦
- Error in max\_MC\_distance : (max\_MC\_distance (heuristic) – max\_MC\_distance (optimal)) / max\_MC\_distance (optimal) ◦
- Error in mean\_weighted\_MC\_distance : (mean\_weighted\_MC\_distance (heuristic) – mean\_weighted\_MC\_distance (optimal)) / mean\_weighted\_MC\_distance (optimal) ◦
- Error in max\_weighted\_MC\_distance : (max\_weighted\_MC\_distance (heuristic) – max\_weighted\_MC\_distance (optimal)) / max\_weighted\_MC\_distance (optimal) ◦

由圖 5.5 圖 5.6 以及表 5.1 : CPSP-Coverage—NPF vs. Optimal solution (Error)◦可知，NPF 演算法所求之 number of covered edges，最多比 optimal solution 少 1 個 edge，低於 optimal solution 所求 2.22%，mean\_MC\_distance 所求則最多比 optimal solution 多 0.02 個 edge，高於 optimal solution 所求 1.25%。此外，max\_MC\_distance 與 optimal solution 所求相同。

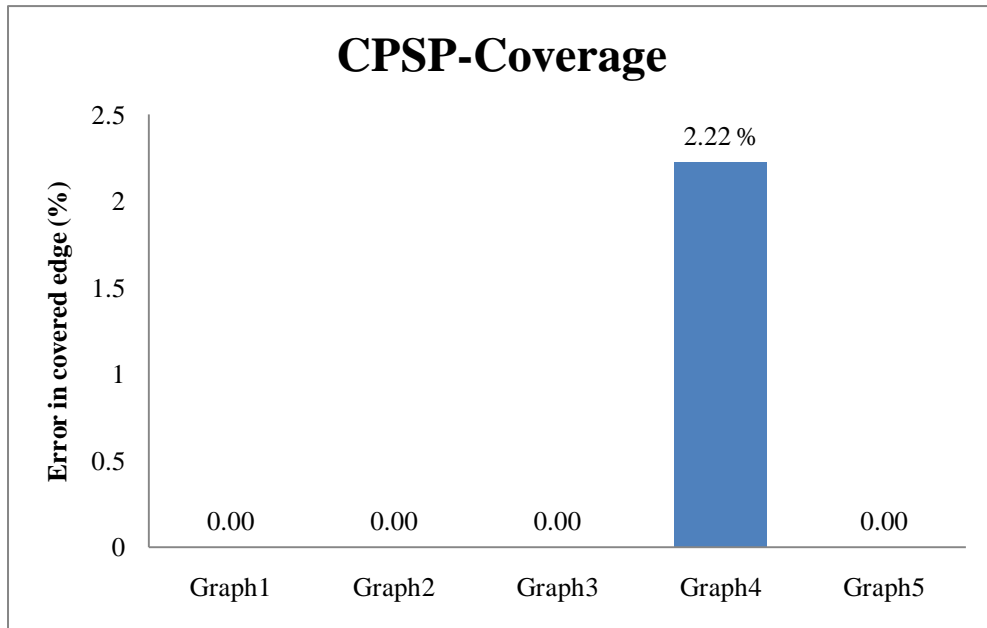


圖 5.5 : CPSP-Coverage—NPF vs. Optimal solution (Error in covered edges)。

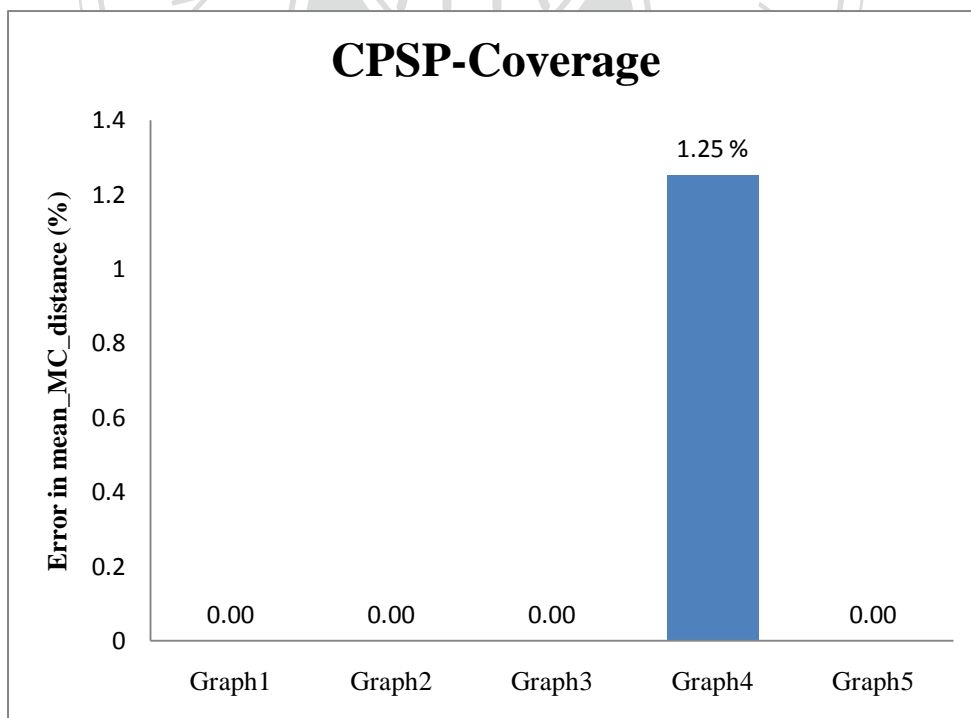


圖 5.6 : CPSP-Coverage—NPF vs. Optimal solution (Error in mean\_MC\_distance)。

表 5.1 : CPSP-Coverage—NPF vs. Optimal solution (Error)。

CPSP-Coverage					
	Graph1	Graph2	Graph3	Graph4	Graph5
no. of covered edges	0	0	0	Difference = 1 Error = 2.22%	0
mean_MC_distance	0	0	0	Difference = 0.02 Error = 1.25%	0
max_MC_distance	0	0	0	0	0

由圖 5.7 圖 5.8 以及表 5.2 可知，NPF-U 演算法所求之 total weight of covered edges，其所涵蓋的 edge 權重最多比 optimal solution 少 4。然而，在 mean\_weighted\_MC\_distance 則有機會優於 optimal solution 所求，這是因為在 CPSP-Utility 模型中，optimization objective 是 total weight of covered edges，而非 mean\_weighted\_MC\_distance。此外，max\_weighted\_MC\_distance 與 optimal solution 所求相同。

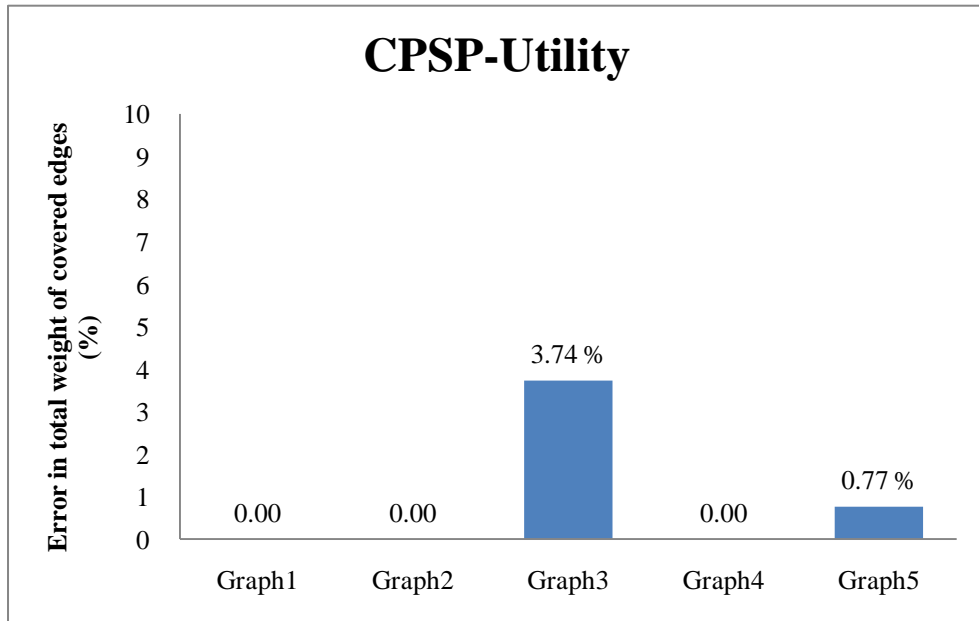


圖 5.7 : CPSP-Utility—NPF-U vs. Optimal solution (Error in total weight of covered edges)。

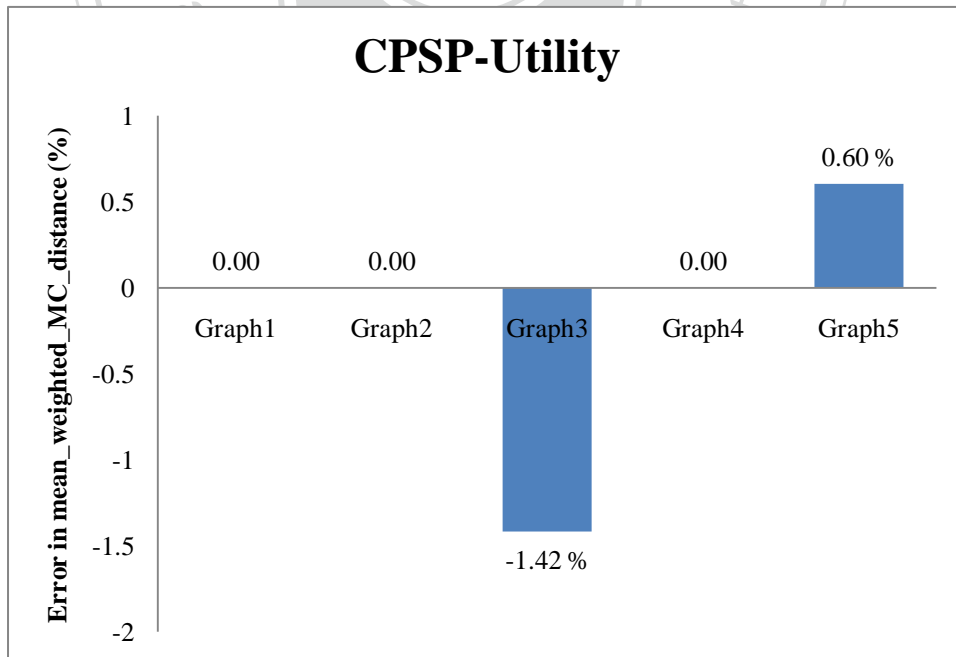


圖 5.8 : CPSP-Utility—NPF-U vs. Optimal solution (Error in mean\_weighted\_MC\_distance)。

表 5.2 : CPSP-Utility—NPF-U vs. Optimal solution (Error)。

CPSP-Utility					
	Graph1	Graph2	Graph3	Graph4	Graph5
total weight of covered edges	0	0	Difference in = 4 Error = 3.74%	0	Difference = 1 Error = 0.77%
mean_weighted_MC_distance	0	0	Difference = -0.05 Error = -1.42%	0	Difference = 0.02 Error = 0.6%
max_weighted_MC_distance	0	0	0	0	0

## 5.2 大型問題效能評估

以下我們將產生大量的問題，模擬在不同 Graph 大小的情況下，0-1 knapsack problem 的啟發式演算法，與 NPF、NPF-U 演算法之效率，並分析 node sensitivity、trail sensitivity 所造成的影響。由於 optimal solution 不易求得，故未與其比較。

我們所用的評估指標如下：

- average number of covered edges
- average total weight of covered edges
- average mean\_MC\_distance
- average mean\_weighted\_MC\_distance
- average max\_MC\_distance
- average max\_weighted\_MC\_distance

### 5.2.1 實驗環境的參數設定—sensitivity to the no. of nodes

本實驗要觀察節點數對演算法的影響，對 10~100 間大小不同的 Graph 各隨機產生 50 個 Graph，並以下列指標評估其效能：

- The total number of candidate control points = 10~100。
- The total number of trails = 8。
- Available resources = 12。
- The cost of each candidate control point = 1~5。
- The edge\_weight between candidate control point = 1~5。
- The flow of each trail = 1~10。

### 5.2.2 實驗結果與分析—sensitivity to the no. of nodes

由圖 5.9 可知，隨著候選控制點的增加，NPF 演算法的 average number of covered edges 效率最高，NPF-U 演算法次之，0-1 knapsack 的啟發式演算法最低。由圖 5.10 可知，隨著候選控制點的增加，NPF-U 演算法的 average total weight of covered edges 效率最高，NPF 演算法次之，0-1 knapsack 的啟發式演算法最低。

由圖 5.11、圖 5.12 可知，隨著候選控制點的增加，NPF 演算法的 average mean\_MC\_distance、average mean\_weighted\_MC\_distance 效率最高，NPF-U 演算法次之，0-1 knapsack 的啟發式演算法最低。在 node sensitivity 下，我們推測可能是因為可以選擇的候選控制點增加了，因此，很容易就可以找到「好」的候選控制點。

由圖 5.13 圖 5.14 可知，隨著候選控制點的增加，原本在 average max\_MC\_distance、average max\_weighted\_MC\_distance 效率較低的 0-1 knapsack 的啟發式演算法，會逐漸與 NPF、NPF-U 演算法愈來愈接近。

根據以上描述我們可知，當必須考慮某些路段的長短及重要性時，NPF-U 的涵蓋效率最高，故較建議使用 CPSP-Utility 模型來建立控制網路，以保證所有在路徑上行

走的登山客，都可以很快地接近控制點。而當不必將路段的重要性列入考慮時，CPSP-Coverage 模型則是較好的選擇，因為 NPF 演算法所求出的 average mean\_MC\_distance、average mean\_weighted\_MC\_distance 效率最高。

表 5.3 為啟發式演算法評估 (sensitivity to the no. of nodes)的詳細數值記錄。

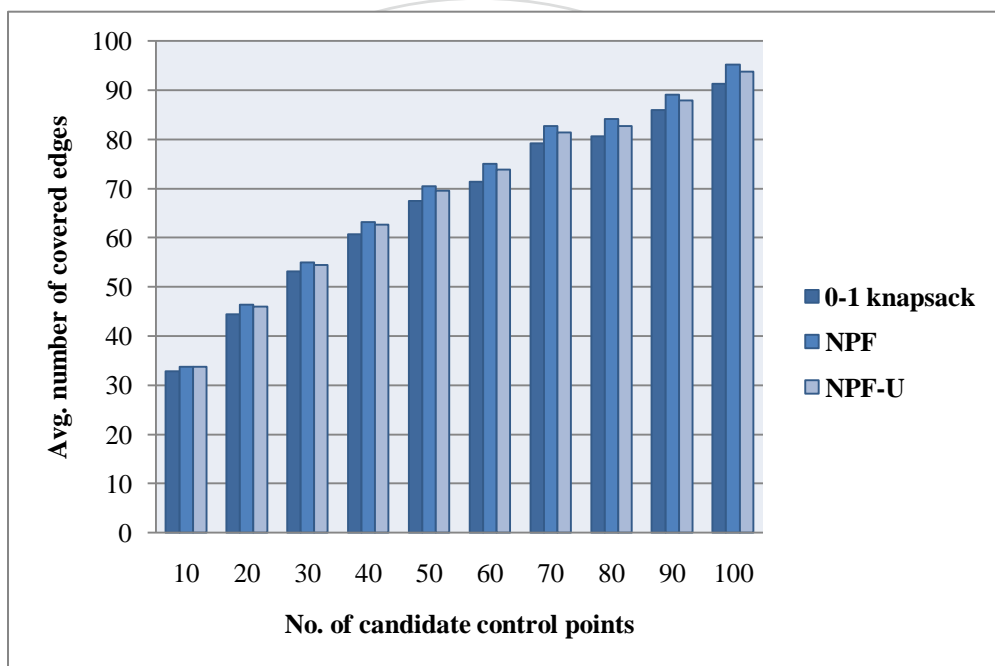


圖 5.9：啟發式演算法評估 (Avg. number of covered edges) (sensitivity to the no. of nodes)。



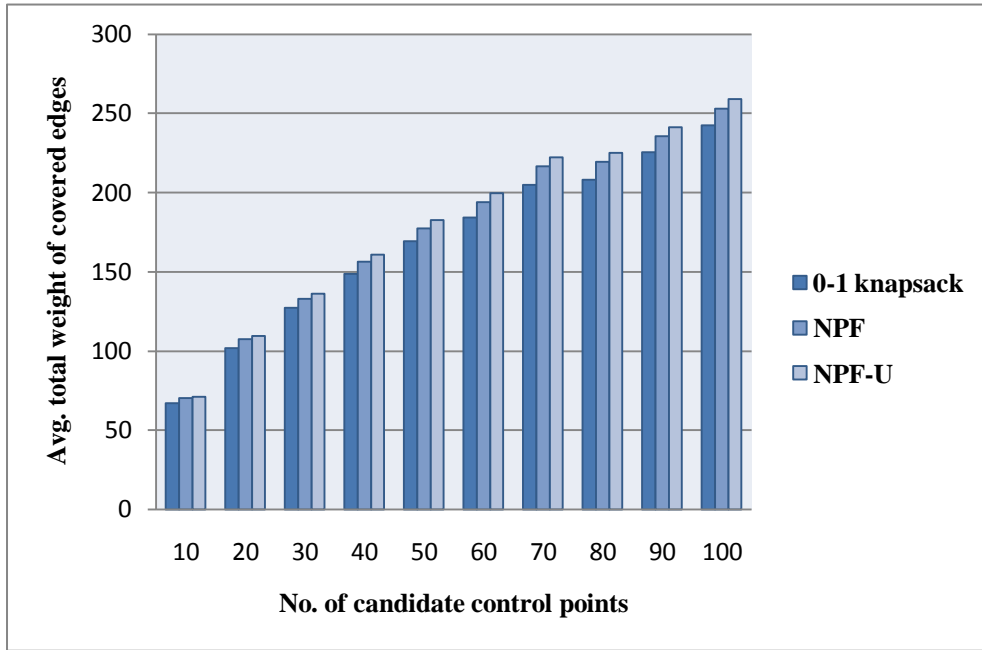


圖 5.10：啟發式演算法評估 (Avg. total weight of covered edges)  
(sensitivity to the no. of nodes)。

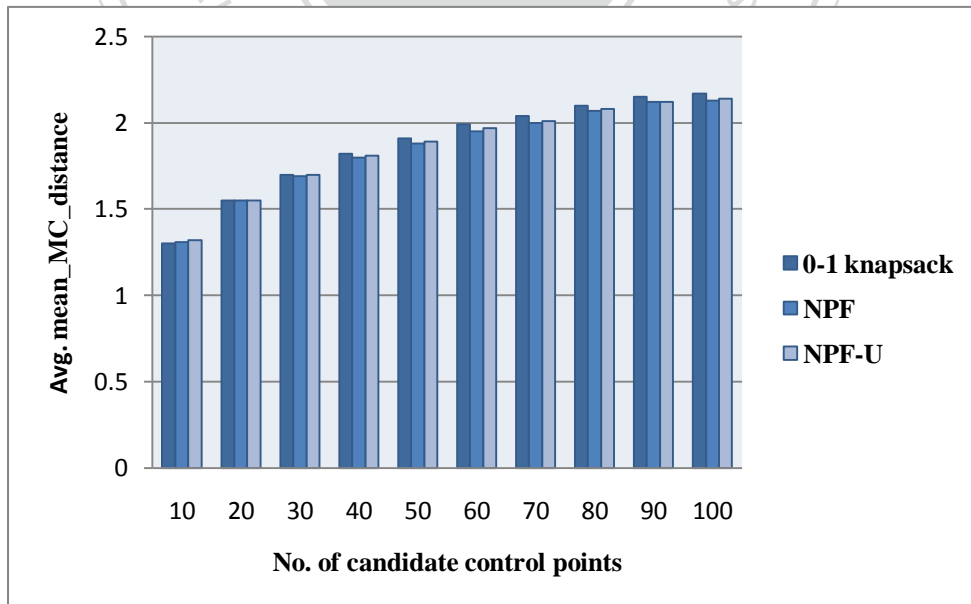


圖 5.11：啟發式演算法評估 (Avg. mean\_MC\_distance)  
(sensitivity to the no. of nodes)。

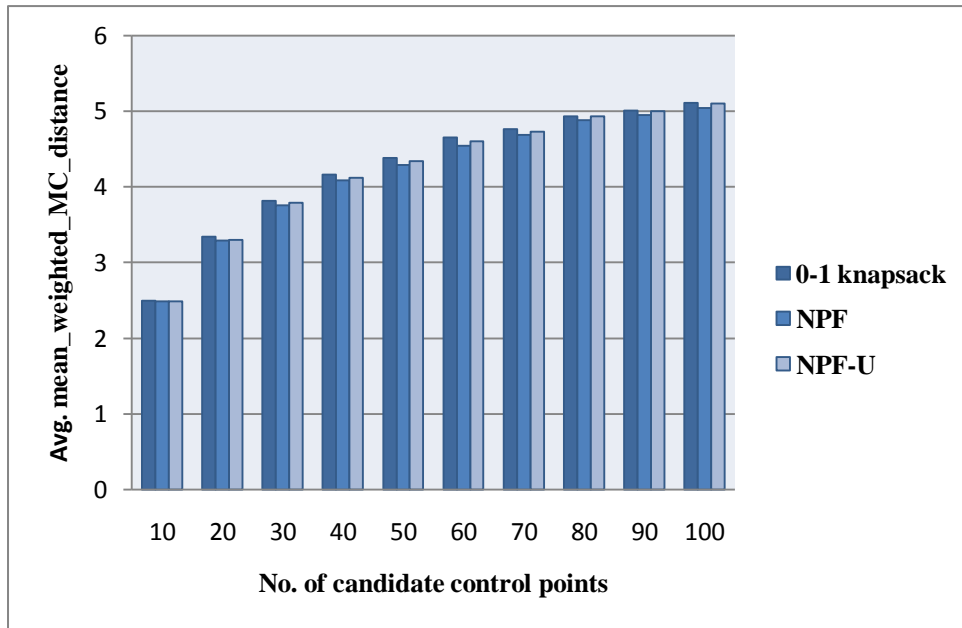


圖 5.12：啟發式演算法評估 (Avg. mean\_weighted\_MC\_distance)  
(sensitivity to the no. of nodes)。

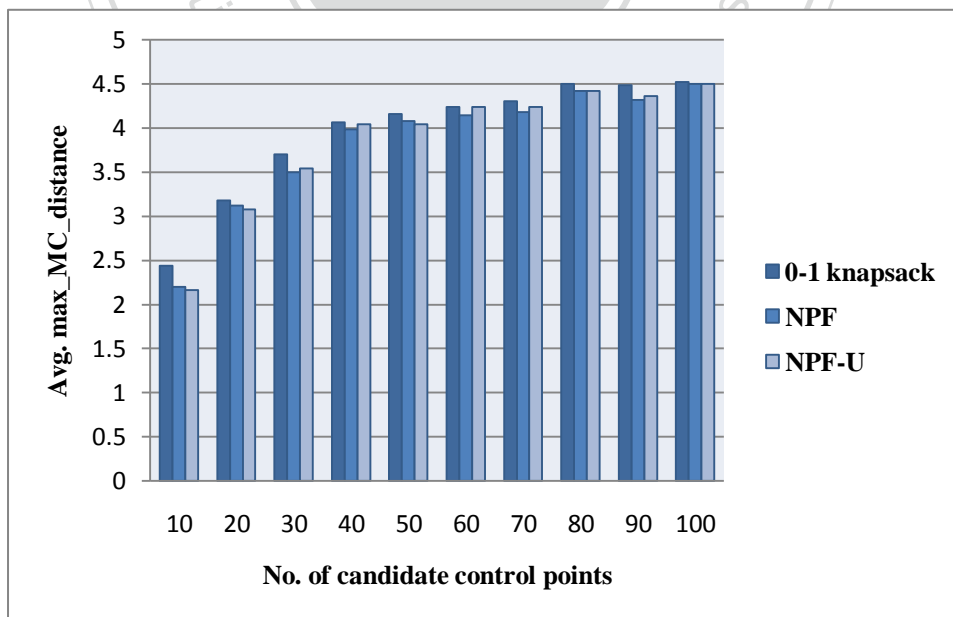


圖 5.13：啟發式演算法評估 (Avg. max\_MC\_distance)  
(sensitivity to the no. of nodes)。

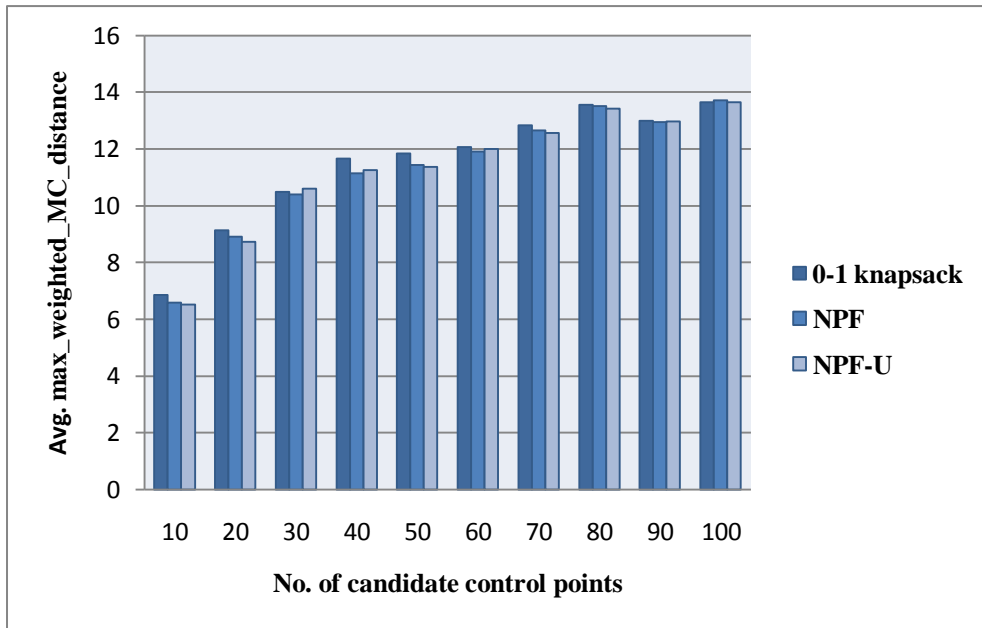


圖 5.14：啟發式演算法評估 (Avg. max\_weighted\_MC\_distance)  
(sensitivity to the no. of nodes)。

表 5.3：啟發式演算法評估 (sensitivity to the no. of nodes)。

No. of nodes	10	20	30	40	50	60	70	80	90	100
<b>Avg. number of covered edges</b>										
0-1 knapsack	32.9	44.44	53.12	60.7	67.48	71.42	79.14	80.64	85.98	91.36
NPF	33.72	46.34	54.92	63.14	70.48	75.02	82.72	84.18	89.1	95.18
NPF-U	33.72	45.94	54.42	62.7	69.56	73.86	81.46	82.76	87.92	93.8
<b>Avg. total weight of covered edges</b>										
0-1 knapsack	67.22	101.9	127.16	148.56	169.44	184.34	205.08	208.24	225.56	242.48
NPF	70.18	107.4	133.14	156.46	177.62	193.82	216.5	219.34	235.48	253.12
NPF-U	70.98	109.7	136.22	160.86	182.86	199.6	222.14	225.1	241.34	259.2
<b>Avg. mean_MC_distance</b>										
0-1 knapsack	1.3	1.55	1.7	1.82	1.91	1.99	2.04	2.1	2.15	2.17
NPF	1.31	1.55	1.69	1.8	1.88	1.95	2	2.07	2.12	2.13
NPF-U	1.32	1.55	1.7	1.81	1.89	1.97	2.01	2.08	2.12	2.14
<b>Avg. mean_weighted_MC_distance</b>										
0-1 knapsack	2.5	3.34	3.82	4.16	4.38	4.65	4.76	4.93	5.01	5.11
NPF	2.49	3.29	3.76	4.09	4.29	4.54	4.69	4.88	4.95	5.04
NPF-U	2.49	3.3	3.79	4.12	4.34	4.6	4.73	4.93	5	5.1
<b>Avg. max_MC_distance</b>										
0-1 knapsack	2.44	3.18	3.7	4.06	4.16	4.24	4.3	4.5	4.48	4.52
NPF	2.2	3.12	3.5	3.98	4.08	4.14	4.18	4.42	4.32	4.5
NPF-U	2.16	3.08	3.54	4.04	4.04	4.24	4.24	4.42	4.36	4.5
<b>Avg. max_weighted_MC_distance</b>										
0-1 knapsack	6.86	9.14	10.5	11.66	11.84	12.06	12.84	13.56	13	13.64
NPF	6.6	8.92	10.4	11.14	11.44	11.92	12.66	13.52	12.96	13.72
NPF-U	6.52	8.74	10.6	11.26	11.36	12	12.56	13.42	12.98	13.64

### 5.2.3 實驗環境的參數設定—sensitivity to the no. of trails

本實驗要觀察登山路徑數對演算法的影響，對 4~40 間大小不同的 Graph 各隨機產生 50 個 Graph，並以下列指標評估其效能：

- The total number of candidate control points = 10。
- The total number of trails = 4~40。
- Available resources = 12。
- The cost of each candidate control point = 1~5。
- The edge\_weight between candidate control point = 1~5。
- The flow of each trail = 1~10。

### 5.2.4 實驗結果與分析—sensitivity to the no. of trails

由圖 5.15 可知，隨著登山路徑數的增加，0-1 knapsack 啟發式演算法的 average number of covered edges 效率，會逐漸與 NPF、NPF-U 演算法接近。由圖 5.16 可知，隨著登山路徑數的增加，NPF-U 演算法的 average total weight of covered edges 效率最高，NPF 演算法與 0-1 knapsack 的啟發式演算法次之。

由圖 5.17 可知，隨著登山路徑數的增加，0-1 knapsack 的啟發式演算法的 average mean\_MC\_distance 效率最高，NPF 演算法次之，NPF-U 演算法最低。由圖 5.18 可知，這 3 個啟發式演算法所求得的 average mean\_weighted\_MC\_distance 會愈來愈低，最後三者幾乎沒有差異。在 trail sensitivity 下，我們推測可能是因為登山路徑數增加了，故交點多的地方，一定比較好，很容易就找到較好的候選控制點。

由圖 5.19 圖 5.20 可知，隨著登山路徑數的增加，這 3 個啟發式演算法所求得的 average max\_MC\_distance、average max\_weighted\_MC\_distance 會愈來愈低，最後三者幾乎沒有差異。

根據以上描述我們可知，當必須考慮某些路段的重要性時，NPF-U 的涵蓋效率最

高，故較建議使用 CPSP-Utility 模型來建立控制網路，以保證所有在路徑上行走的登山客，都可以很快地接近控制點。而當不必將路段的重要性列入考慮時，CPSP-Flow 模型則是較好的選擇，因其求出的 average mean\_MC\_distance 效率最高，而對於 average mean\_weighted\_MC\_distance 則是當登山路徑多到 12 以上時，三個模型的表現差不多。

表 5.4 為啟發式演算法評估 (sensitivity to the no. of trails)的詳細數值記錄。

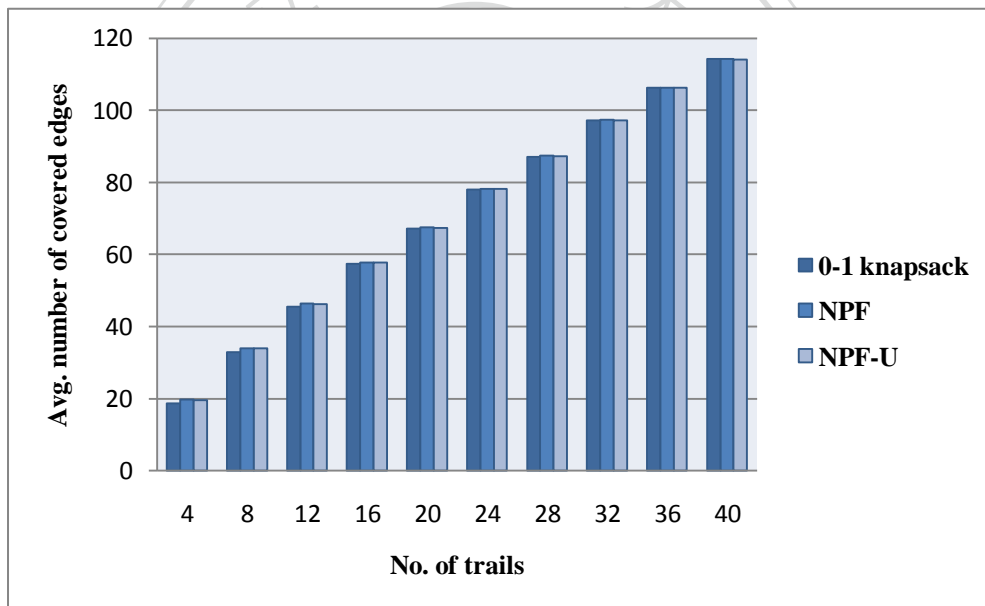


圖 5.15：啟發式演算法評估 (Avg. number of covered edges)  
(sensitivity to the no. of trails)。

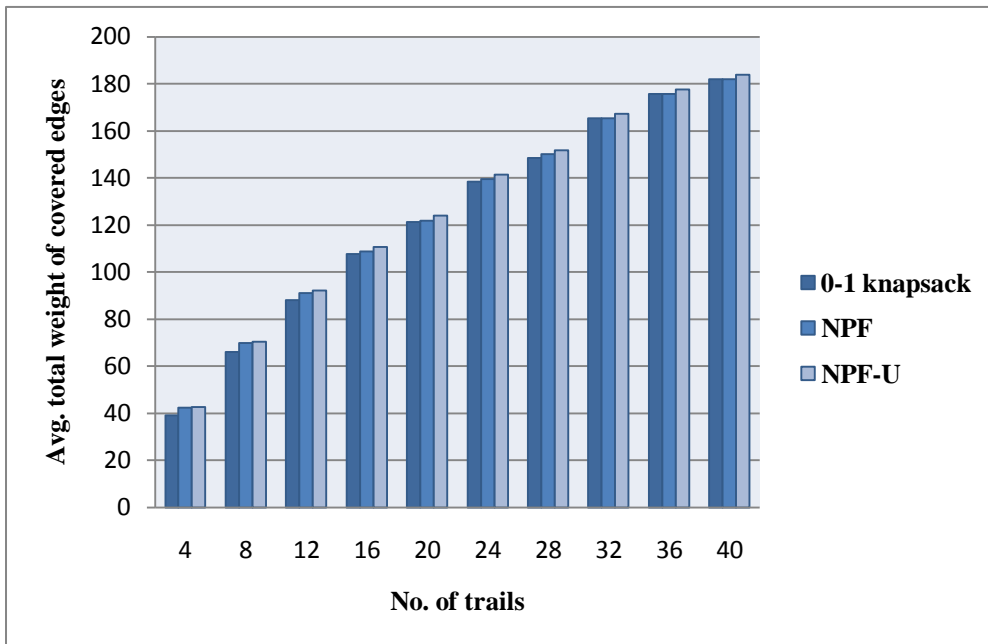


圖 5.16：啟發式演算法評估 (Avg. total weight of covered edges)  
(sensitivity to the no. of trails)。

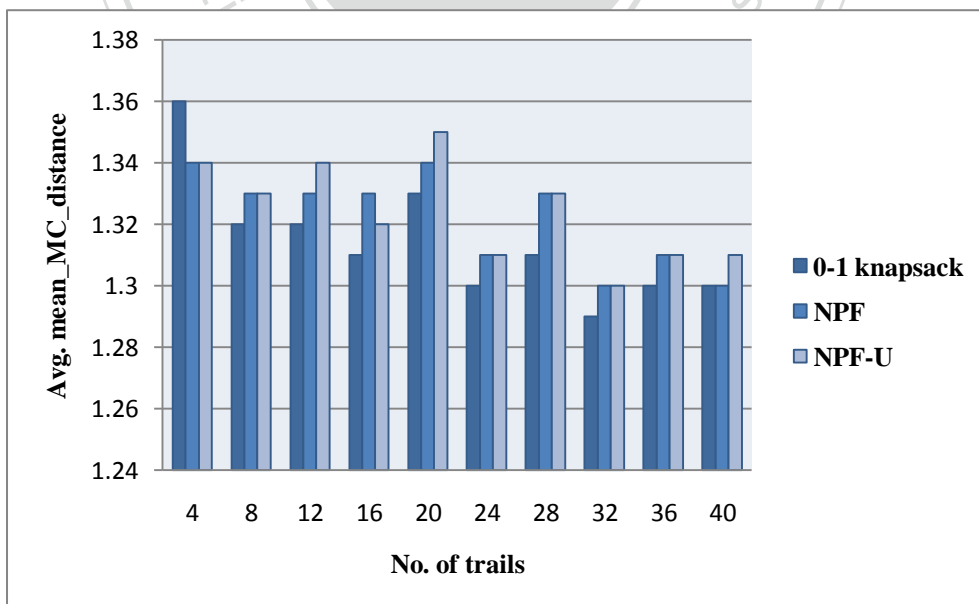


圖 5.17：啟發式演算法評估 (Avg. mean\_MC\_distance)  
(sensitivity to the no. of trails)。

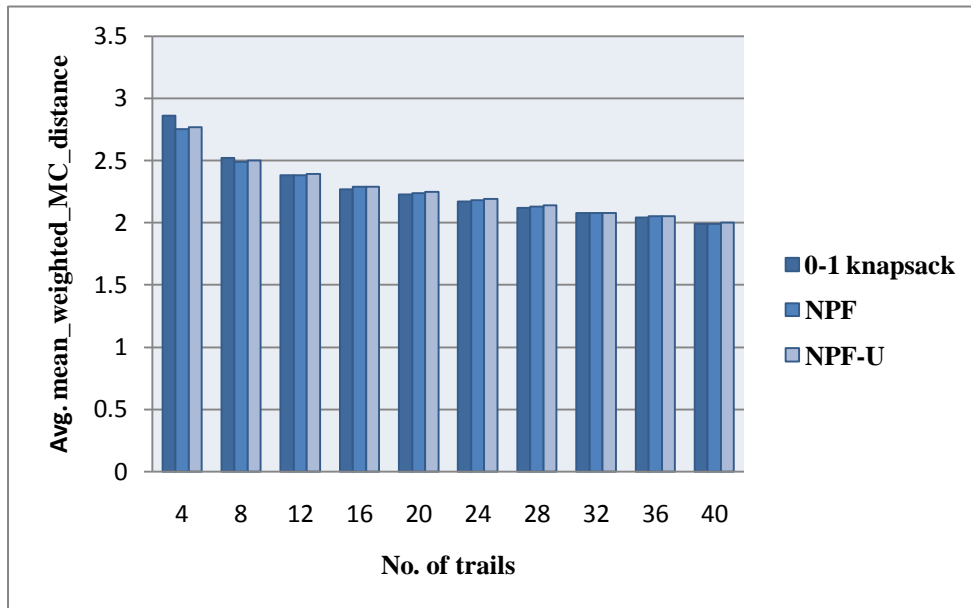


圖 5.18：啟發式演算法評估 (Avg. mean\_weighted\_MC\_distance)  
(sensitivity to the no. of trails)。

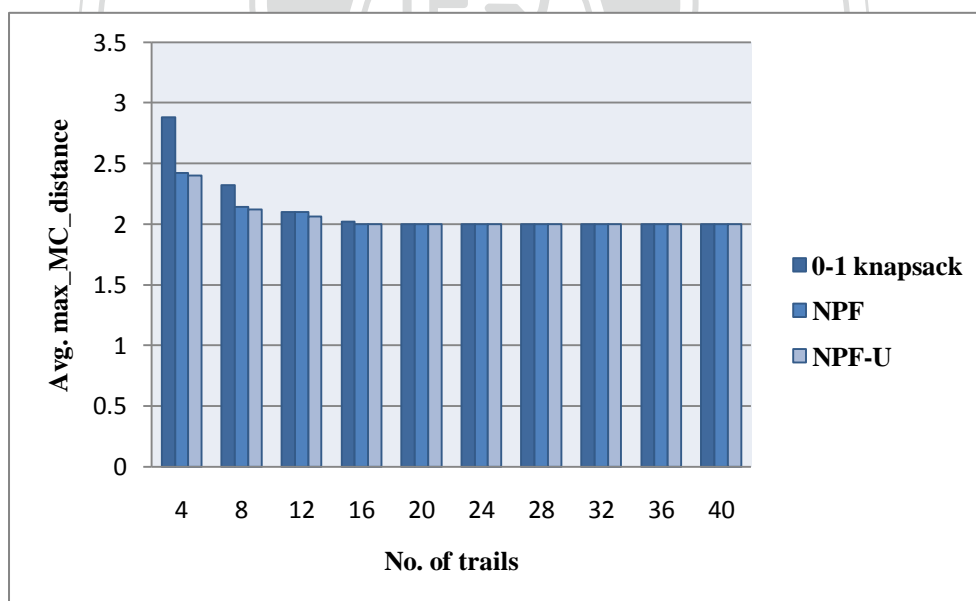


圖 5.19：啟發式演算法評估 (Avg. max\_MC\_distance)  
(sensitivity to the no. of trails)。



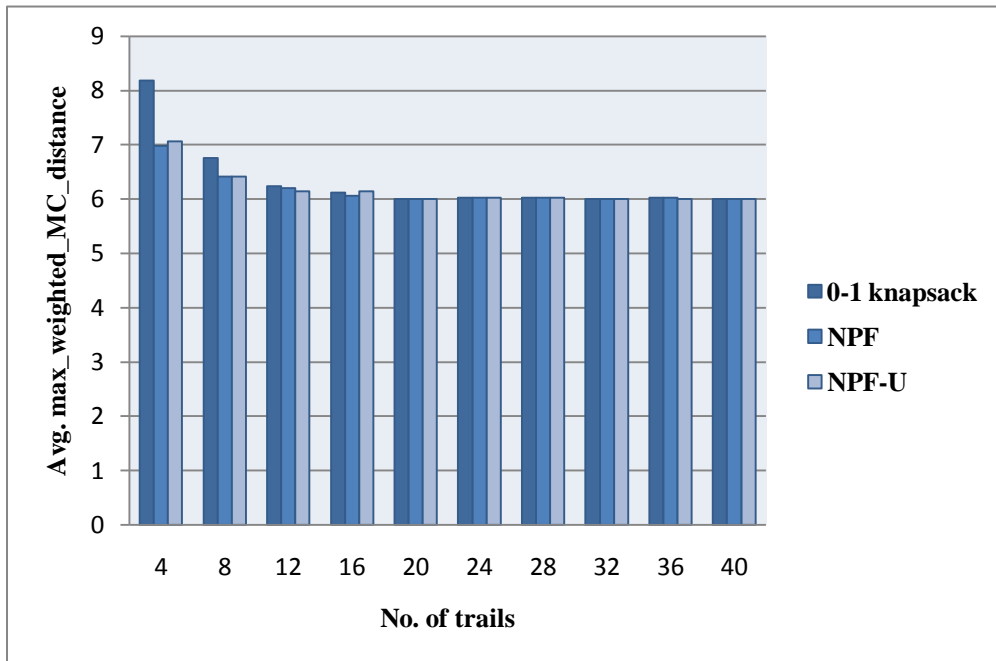


圖 5.20：啟發式演算法評估 (Avg. max\_weighted\_MC\_distance)  
(sensitivity to the no. of trails)。

表 5.4：啟發式演算法評估 (sensitivity to the no. of trails)。

No. of trails	4	8	12	16	20	24	28	32	36	40
<b>Avg. number of covered edges</b>										
0-1 knapsack	18.74	32.9	45.56	57.44	67.16	78.06	87.16	97.3	106.22	114.2
NPF	19.7	33.98	46.48	57.84	67.48	78.28	87.42	97.36	106.28	114.26
NPF-U	19.62	33.9	46.2	57.74	67.3	78.18	87.18	97.3	106.2	114.16
<b>Avg. total weight of covered edges</b>										
0-1 knapsack	39.14	66.06	87.98	107.62	121.34	138.42	148.58	165.36	175.62	182.02
NPF	42.3	69.8	91.12	108.78	121.88	139.64	150.16	165.5	175.68	181.98
NPF-U	42.58	70.38	92.14	110.62	124.06	141.38	151.86	167.32	177.66	183.88
<b>Avg. mean_MC_distance</b>										
0-1 knapsack	1.36	1.32	1.32	1.31	1.33	1.3	1.31	1.29	1.3	1.3
NPF	1.34	1.33	1.33	1.33	1.34	1.31	1.33	1.3	1.31	1.3
NPF-U	1.34	1.33	1.34	1.32	1.35	1.31	1.33	1.3	1.31	1.31
<b>Avg. mean_weighted_MC_distance</b>										
0-1 knapsack	2.86	2.52	2.38	2.27	2.23	2.17	2.12	2.08	2.04	1.99
NPF	2.75	2.49	2.38	2.29	2.24	2.18	2.13	2.08	2.05	1.99
NPF-U	2.77	2.5	2.39	2.29	2.25	2.19	2.14	2.08	2.05	2
<b>Avg. max_MC_distance</b>										
0-1 knapsack	2.88	2.32	2.1	2.02	2	2	2	2	2	2
NPF	2.42	2.14	2.1	2	2	2	2	2	2	2
NPF-U	2.4	2.12	2.06	2	2	2	2	2	2	2
<b>Avg. max_weighted_MC_distance</b>										
0-1 knapsack	8.18	6.76	6.24	6.12	6	6.02	6.02	6	6.02	6
NPF	6.98	6.42	6.2	6.06	6	6.02	6.02	6	6.02	6
NPF-U	7.06	6.42	6.14	6.14	6	6.02	6.02	6	6	6

## 第6章

### 結語

本篇論文的主要貢獻是提出控制網路的概念，並提出一個簡單的行動代理人之搜尋演算法，讓搜尋代理人能利用控制網路，來快速搜尋到目標代理人，使得行動代理人技術的應用及優點，可以移植到機會網路上，以增加「山文誌登山資訊系統」的應用功能。而山難救援只是其中的一個應用，其他還有變更集合地點、互動通訊等許多應用，留待更多的人去發掘。

在建立控制網路的部分，假設地區內網路已存在的前提下，本論文提出數種計算模型以供挑選建置控制點的地點，而控制點之間假設可透過高速網路來傳送行動代理人，到接近目標代理人的控制點，再送至逆向行走登山客所攜帶的行動節點上，如此可以快速搜尋到目標代理人。此外，本篇論文所提出的三個控制網路之建置模型，以 CPSP-Flow 最簡單，CPSP-Coverage 居中，CPSP-Utility 最複雜，但也考慮得最周詳，最能反映實際的情況。

針對不同的控制網路建置模型，我們分別提出一個有效率的啟發式演算法，來解決控制點選擇問題，其中，CPSP-Flow 由於已證明為 0-1 knapsack problem，故使用其現成的演算法，而 CPSP-Coverage、CPSP-Utility 則分別使用 NPF、NPF-U 演算法，最後，利用大量隨機產生的 Graph 進行效能評估。根據實驗結果可知，對於須考慮路段重要性的情況下，較適合用 CPSP-Utility 模型來建立控制網路，以保證所有在路徑上行走的登山客，都可以很快地接近控制點。在所有的模型及演算法中，最接近目標

代理人的控制點，距離目標都在兩個路段之內，大部分短於一個路段。



## 參考文獻

- [1] S. Alouf, F. Huet, and P. Nain, "Forwarders vs. centralized server: an evaluation of two approaches for locating mobile agents," *Performance Evaluation*, vol. 49, no. 1-4, pp. 299-319, Sep. 2002.
- [2] J. Baumann, "A comparison of mechanisms for locating mobile agents," Technical Report TR 1999/11, Univ. of Stuttgart, Faculty of Computer Science, 1999.
- [3] J. Baumann, F. Hohl, K. Rothermel, and M. Straser, "Mole—Concepts of a mobile agent system," *World Wide Web*, vol. 1, no. 3, pp. 123-137, Sep. 1998.
- [4] M. Bavandla. "Improving the performance of location management protocols in a multiregion environment." Master's Thesis, I. I. T. Roorkee, 2004.
- [5] E. Brewer, *et al.* "Technology and Infrastructure for Emerging Regions (TIER)." Internet: <http://tier.cs.berkeley.edu/drupal/about>, Retrieved on Apr. 20, 2011.
- [6] S. Burleigh, A. Hooke, L. Torgerson, K. Fall, V. Cerf, B. Durst, K. Scott, and H. Weiss, "Delay-tolerant networking: an approach to interplanetary internet," *IEEE Communications Magazine*, vol. 41, no. 6, pp. 128-136, Jun. 2003.
- [7] W.-S. E. Chen, C.-W. R. Leng, and Y.-N. Lien, "A novel mobile agent search algorithm," in *Proc. of the International Conference on Computer, Communications and Networks*, 1997, pp. 128-131.
- [8] W.-S. E. Chen, S. T. Su, Y.-N. Lien, H. T. Shu, and H. Liu, "Mobility and management support for mobile agents," in *Proc. of the 2nd International Conference on Autonomous Agents*, 1998, pp. 451-452.
- [9] T. H. Cormem, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to algorithms*: The MIT press, 2001.
- [10] A. Di Stefano and C. Santoro, "Locating mobile agents in a wide distributed environment," *IEEE Transactions on Parallel and Distributed Systems*, vol. 13, no. 8, pp. 844-864, Aug. 2002.
- [11] S. Farrell, *et al.* "InterPlanetary Internet." Internet: <http://www.ipnsig.org/>, Retrieved on Apr. 20, 2011.
- [12] P. Flocchini and M. Xie, "A fully distributed mechanism for locating mobile agents," in *Proc. of the International Multi-Conference on Computing in the Global Information Technology*, 2007, pp. 41-41.
- [13] G. Glass. "Objectspace voyager core package technical overview," in *Mobility: process, computers and agents*, ACM Press/Addison-Wesley Publishing Co., 1999,

- pp. 611-627.
- [14] R. S. Gray, G. Cybenko, D. Kotz, R. A. Peterson, and D. Rus, "D'Agents: Applications and performance of a mobile agent system," *Software: Practice and Experience*, vol. 32, no. 6, pp. 543-573, May. 2002.
  - [15] C. L. Hu and W. S. E. Chen, "A mobile agent-based active network architecture," in *Proc. of the 7th International Conference on Parallel and Distributed Systems*, 2000, pp. 445-452.
  - [16] C.-M. Huang, K.-c. Lan, and C.-Z. Tsai, "A survey of opportunistic networks," in *Proc. of the 22nd International Conference on Advanced Information Networking and Applications*, 2008, pp. 1672-1677.
  - [17] J. H. Huang, S. Amjad, and S. Mishra. "Cenwits Search and Rescue System." Internet: <https://cse1.cs.colorado.edu/~huangjh/Cenwits/index.htm>, Retrieved on Apr. 20, 2011.
  - [18] J. H. Huang, S. Amjad, and S. Mishra, "Cenwits: a sensor-based loosely coupled search and rescue system using witnesses," in *Proc. of the 3rd International Conference on Embedded Networked Sensor Systems*, 2005, pp. 180-191.
  - [19] Y. T. Huang, Y. C. Chen, J. H. Huang, L. J. Chen, and P. Huang, "YushanNet: A delay-tolerant wireless sensor network for hiker tracking in Yushan national park," in *Proc. of the 10th International Conference on Mobile Data Management: Systems, Services and Middleware*, 2009, pp. 379-380.
  - [20] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. S. Peh, and D. Rubenstein, "Energy-efficient computing for wildlife tracking: design tradeoffs and early experiences with ZebraNet," in *Proc. of the 10th International Conference on Architectural Support for Programming Languages and Operating Systems*, 2002, pp. 96-107.
  - [21] R. M. Karp, "Reducibility among combinatorial problems, Complexity of Computer Computations," in *Proc. Sympos. IBM Thomas J. Watson Res. Center*, 1972, pp. 85-103.
  - [22] G. Kastidou, E. Pitoura, and G. Samaras, "A scalable hash-based mobile agent location mechanism," in *Proc. of the 23rd International Conference on Distributed Computing Systems*, 2003, pp. 472-472.
  - [23] D. B. Lange and M. Oshima, "Mobile agents with Java: The Aglet API," *World Wide Web*, vol. 1, no. 3, pp. 111-121, 1998.
  - [24] D. B. Lange and M. Oshima, "Seven good reasons for mobile agents," *Communications of the ACM*, vol. 42, no. 3, pp. 88-89, Mar. 1999.

- [25] S. Lazar, I. P. Weerakoon, and D. P. Sidhu, "A scalable location tracking and message delivery scheme for mobile agents," in *Proc. of the 7th Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*, 1998, pp. 243-248.
- [26] J. Li, H. Kameda, and K. Li, "Optimal dynamic mobility management for PCS networks," *IEEE/ACM Transactions on Networking (TON)*, vol. 8, no. 3, pp. 319-327, Jun. 2000.
- [27] T.-Y. Li and K.-Y. Lam, "An optimal location update and searching algorithm for tracking mobile agent," in *Proc. of the 1st International Joint Conference on Autonomous Agents and Multiagent Systems: part 2*, 2002, pp. 639-646.
- [28] Y. N. Lien and C. W. R. Leng, "On the search of mobile agents," in *Proc. of the 7th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, 1996, pp. 703-707.
- [29] Y. N. Lien, F. Liu, C. W. Leng, and W. S. Chen, "Intelligent Search of Mobile Agents," in *Proc. of the 1997 International Conference on Computer Systems Technology for Industrial Applications*, 1997, pp. 110-116.
- [30] X. Ligong, Z. Zude, and L. Quan, "A novel communication mechanism of mobile agent system," in *Proc. of the 2008 International Workshop on Education Technology and Training & 2008 International Workshop on Geoscience and Remote Sensing*, 2008, pp. 695-699.
- [31] Z. Lu and J. Fan, "Delay/Disruption tolerant network and its application in military communications," in *Proc. of the 2010 International Conference on Computer Design and Applications (ICCD)*, 2010, pp. 231-234.
- [32] M. Martonosi, S. Lyon, L.-S. Peh, V. Poor, and D. Rubenstein. "The ZebraNet Wildlife Tracker." Internet: <http://www.princeton.edu/~mrm/zebranet.html>, Retrieved on Apr. 20, 2011.
- [33] D. Milojicic, *et al.*, "MASIF: The OMG mobile agent system interoperability facility," in *Proc. of the 2nd International Workshop on Mobile Agents*, 1998, pp. 50-67.
- [34] D. S. Milojicic, F. Douglass, and R. Wheeler. *Mobility: processes, computers, and agents*. New York, NY: ACM Press/Addison-Wesley Publishing Co., 1999.
- [35] D. S. Milojicic, W. LaForge, and D. Chauhan, "Mobile objects and agents (MOA)," in *Proc. of the 4th USENIX Conference on Object-Oriented Technologies and Systems*, 1998, pp. 1-14.
- [36] R. Patel and K. Garg, "PMADE-A Platform for Mobile Agent Distribution &

- Execution," in *Proc. of 5th World MultiConference on Systemics, Cybernetics and Informatics (SCI) and 7th International Conference on Information System Analysis and Synthesis (ISAS)*, 2001, pp. 287-293.
- [37] L. Pelusi, A. Passarella, and M. Conti, "Opportunistic networking: data forwarding in disconnected mobile ad hoc networks," *IEEE Communications Magazine*, vol. 44, no. 11, pp. 134-141, Nov. 2006.
- [38] Y. Sasaki and Y. Shibata, "Distributed disaster information system in DTN based mobile communication environment," in *Proc. of the 2010 International Conference on Broadband, Wireless Computing, Communication and Applications*, 2010, pp. 274-277.
- [39] R. Sushil, R. Bhargava, and K. Garg, "Location update schemes for mobile agents," *INFOCOMP Journal of Computer Science*, vol. 7, no. 2, pp. 37-43, 2008.
- [40] R. Sushil, K. Garg, and R. Bhargava, "Comparison of location management techniques for mobile agents," in *Proc. of the 16th IEEE International Conference on Networks*, 2008, pp. 1-5.
- [41] X. Tao, X. Feng, X. Li, G. Zhang, and J. Lu, "Communication mechanism in Mogent system," *Journal of Software*, vol. 11, no. 8, pp. 1060-1065, Aug. 2000.
- [42] H. Wang, G. Zeng, and S. Lin, "An efficient control mechanism of mobile agents," in *Proc. of the 2001 International Conference on Computer Networks and Mobile Computing (ICCNMC)*, 2001, pp. 371-379.
- [43] Y. Wang, F. Zhang, and L. Chen, "An approach to incremental SVM learning algorithm," in *Proc. of the 2008 ISECS International Colloquium on Computing, Communication, Control, and Management*, 2008, pp. 352-354.
- [44] D. Wong, N. Paciorek, T. Walsh, J. DiCelie, M. Young, and B. Peet, "Concordia: An infrastructure for collaborating mobile agents," in *Proc. of the 1st International Workshop on Mobile Agents*, 1997, pp. 86-97.
- [45] L. T. Yang. *Mobile Intelligence*. Hoboken, NJ: John Wiley & Sons, Inc., 2010.
- [46] T.-Y. Yeh and T. I. Wang, "A mechanism for tracking mobile agents in a cluster topology," in *Proc. of the 11th International Conference on Parallel and Distributed Systems*, 2005, pp. 320-327.
- [47] 朱浩華, 黃寶儀, 陳伶志, 藍崑展, 林文和, and 黃致豪. "YuShanNet." Internet: <http://nslab.ee.ntu.edu.tw/~YuShanNet/>, Retrieved on Apr. 20, 2011.
- [48] 陳伶志, 黃寶儀, and 黃致豪. "山文誌登山資訊系統." Internet: [http://nrl.iis.sinica.edu.tw/YushanNet/tw\\_project.php](http://nrl.iis.sinica.edu.tw/YushanNet/tw_project.php), Retrieved on Apr. 20, 2011.
- [49] 陳禹銘, 祝鈞毅, 李雅萍, and 周子勤, "多工協調技術之應用與展望," *機械工*



- 業, no. 285, pp. 93-104, Dec. 2006.
- [50] 楊柏華譯, "<<資訊新知>>:行動代理人技術(上)," *中央研究院計算中心通訊*, vol. 18, no. 23, pp. 184-188, Nov. 2002.

