國 立 成 功 大 學
工 業 與 資 訊 管 理 研 究 所

碩 士 論 文

具有餾化過程的多容量狀態製造網路之可靠度研究

Reliability for Multi-state Capacitated Manufacturing Networks with Distillation Processes

研 究 生: 陳 正 楠
指 導 教 授: 王 逸 琳 博士

中 華 民 國 九 十 七 年 八 月

摘要

在一個製造系統的網路中，時常可見零組件經由一些作業程序而產生品質不相同的成品或半成品，由於這些不同品質的成品或半成品之總數量通常存在著某種特定的比例關係，因此可利用 Fang and Qi (2003) 所導入的特殊 D-node 來將製造過程描述成一個分餾的餾化 (distillation) 過程。其中，進入 D-node 的所有流量必須依照某個既定的比例自 D-node 的流出弧分餾出去。由於 D-node 的分餾比例固定了流量之間的相依關係，導致此製造網路的分析過程較為複雜。另外，因為流量經過 D-node 就必須分送到其所有的流出弧，因此自起源點至需求點的運送過程將不再僅是經過一條簡單路徑，而是一個包含許多條簡單路徑的網路子圖。當網路圖中每一條弧的容量符合一個多狀態的機率分配時，計算不含 D-node 之一般網路的可靠度已是一個 NP-hard 的問題，因此計算含有 D-node 的製造網路之可靠度將更具挑戰性，而此亦為本論文之主要研究議題。本論文首先提出一套前置處理程序對原問題加以簡化，進而提出一套新演算法以計算含有 D-node 的製造網路之網路可靠度，其後亦將探討如何在滿足給定的可靠度門檻值之下計算含有 D-node 的製造網路之最小成本，以及多狀態弧容量之分配網路在最短路徑問題上的相關議題。

關鍵字：網路可靠度；多狀態弧容量；製造網路；餾化過程

Abstract

In a manufacturing network, there often exist operations that produce the same products or semi-products of different qualities. Such a phenomenon can be described by a distillation process using a specialized D-node introduced by Fang and Qi (2003) where the flow enters a D-node has to be distributed according to a pre-specified vector of ratios associated with its outgoing arcs. The existence of D-node complicates the manufacturing process since it creates the relation of flow dependency. In particular, to send flow successfully from a source node to a sink node, an augmenting subgraph containing many paths instead of an augmenting simple path has to be constructed, since the flow passing through a D-node has to be distributed to all of its outgoing arcs. As a result, when the capacity associated with each arc obeys a multi-state probability distribution, calculating the system reliability of shipping a given amount of flows for a multi-state capacitated manufacturing network containing D-nodes becomes a difficult task. This paper focuses on techniques to derive the system reliability for such a multi-state capacitated manufacturing network containing D-nodes. We will first introduce polynomial-time preprocessing techniques that simplify the problem structure, and then propose algorithms for calculating the network reliability. In addition, we also investigate the problem of constructing a manufacturing network with multi-state arc capacity that satisfies the reliability lower bound with minimum total cost, and study the problem of stochastic shortest path using our proposed solution methods as well.

keywords: Network reliability, Multi-state arc capacity, Manufacturing network, Distillation process.

誌謝

首先誠摯的感謝指導教授王逸琳博士，老師悉心的教導使我得以一窺最佳化領域的深奧，不時的討論並指點我正確的方向，使我在這些年中獲益匪淺。老師對學問的嚴謹更是我輩學習的典範。

本論文的完成另外亦得感謝李宇欣教授、林義貴教授、洪一薰教授、黃耀廷教授的大力協助。因為有你們的指導，使得本論文能夠更完整而嚴謹。

兩年裡的日子，實驗室裡共同的生活點滴，學術上的討論、言不及義的閒扯、趕作業而爆肝的革命情感等，再再都是非常珍貴的回憶，感謝眾位學長姐、同學、學弟妹的共同砥礪，有大家的陪伴讓兩年的研究生活變得絢麗多彩。

感謝修杰、群達、橙坤學長、姿君學姐們不厭其煩的指出我研究中的缺失，且總能在我迷惘時為我解惑，也感謝建傑、姿儀、家宜同學的幫忙，陪伴我順利走過這兩年。實驗室的俊賢、志偉以及後來進來的那些學弟妹們當然也不能忘記，你們的幫忙及搞笑我銘感在心。

最後，謹以此論文獻給所有支持及陪伴我走過碩士生涯的人。

目 錄

表 目 錄

圖 目 錄

第一章

緒論

網路流量問題在日常生活中時常出現，如運輸、配送、指派等問題。傳統的網路流量問題皆假設網路中的組成元素（即弧與節點）不會有失效的情況（例如一個轉運節點不會因為特殊原因使得流量流進卻無法流出，流進節點的流量必定與流出節點的流量相同）。然而，在現實生活中的確可能會有某些網路組成元素失效而導致整個系統無法運作的情況，例如電力輸送網路系統中某一個配電所失靈，導致輸送至該配電所的電力無法輸送到需求點。為了能夠確保網路系統能穩定地輸送流量以避免流量無法送達需求點所造成的損害，如何有效地計算出網路可靠度及探討其相關議題即有其必要性與研究價值。

1.1 研究動機

網路可靠度的概念最初是由Moore and Shannon (1956)所提出的，當時電腦記憶體是由許多真空管所組成，一旦某一個真空管損壞便會對電腦造成極大的損害。由於真空管是同時運作的關係，即便每個真空管正常運作的機率頗高，但整體系統正常運作的機率卻可能相當低，因此便引發了如何推估其系統可靠度的研究議題。為了應付現實狀況中廣泛的網路可靠度問題，網路組成元素已無法只用成功與失效（正常運作與否）的二元狀態來描述，取而代之的是網路組成元素可能有多種不同的狀態。舉例來說，Douilliez and Jamoulle (1972)與Frank and Hakimi (1965)曾將多狀態隨機弧容量的概念導入計算最大流量的網路問題，以更符合現實的應用。

近年來網路可靠度的相關議題被許多學者廣泛地研究與討論，而求算可靠度的方法不外乎就是窮舉出minimal paths (MPs)或minimal cuts (MCs)以計算可自起點s運送d單位需求到訖點t的機率。其中，MP(MC)指的是一組滿足其子集合不為$s-t$的路徑(割集)的$s-t$路徑(割集)。如圖1.1所示，$\{a_1, a_5\}$ ($\{a_1, a_4, a_6\}$)為MP(MC)，而$\{a_1, a_3, a_4, a_5\}$ ($\{a_1, a_3, a_4, a_6\}$)便不是MP(MC)。然而，窮舉出所有的MP或MC本身已是一個NP-hard的問題(Colbourn, 1987)，要如何快速並有系統地列出計算可靠度所需要用到的MPs或MCs將更為困難。除了以窮舉出所有的路徑連結方式來找尋MP(MC)的作法之外，另有學者(Yeh, 2005)以窮舉出所有可能的弧容量狀態向量，再用最大流量問題的限制式刪除不可行的狀態向量的方式來找尋可行的MPs；或是提出利用節點分群的方式有系統地找尋所有的MCs以加速尋找可行之容量狀態向量(Yeh, 2006)，進一步縮短計算可靠度的時間。

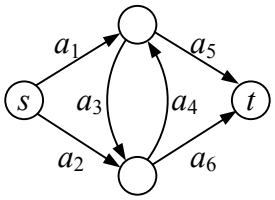


圖 1.1: 網路圖範例

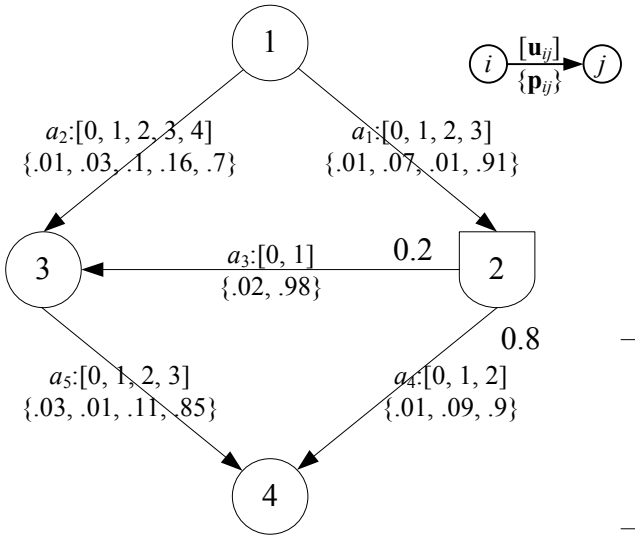
除了試圖加速一般求解網路可靠度的演算法速度之外，文獻中也有針對不同的現實需求而發展新的網路可靠度計算方法之相關研究議題。舉例來說，Lin (2007b)針對物流產業中之上游廠商希望在一定的預算成本內運送不同的商品到需求點的情況提出一個計算網路可靠度的演算法。此外，Lin (2007a)亦用MC的方式，探討在一個節點可能隨機失效且具成本限制的網路下，如何計算輸送多種商品的可靠度指標。

上述問題皆著重在探討如何於種種的限制之下求算出精確的網路可靠度。文獻顯示，計算一般網路可靠度屬於NP-hard的問題(Provan and Ball, 1983)。由於計算較大的網路系統之可靠度必須花費大量時間，因此Ramirez-Marquez and Coit (2005)利用蒙地卡羅法進行網路可靠度的模擬，該方法雖然無法求得精確值，卻可在一定的求解品質之下大幅縮短了求解時間。

雖然網路可靠度的研究在近年來被學者們廣為討論，但其所探討的議題大多還是僅侷限於滿足流量守恆及容量限制之傳統網路架構。除了 Hsieh and Lin (2003) 針對多個起訖點的網路圖探討其可靠度最高的資源分配方式之外，鮮少有人探討特殊網路架構的可靠度計算方式。然而，在現實的製造系統中的確有許多製造過程無法單純地用傳統的網路架構來描述。舉例來說，在一個生產製造過程中，同樣的原料或零組件進入某個工作站處理之後，可能會依照不同的品級分送到下一個工作站。Koene (1982) 將此產品製造過程的分級情況建立模式，定義此處理過程為精化 (refine)，藉此來描述網路圖中流量經過節點之後會依照特定比例分流的情況，並將之命名為 Pure Processing Network。而 Fang and Qi (2003) 更將該特殊節點定義為分配節點 (Distribution Node；D-node)，提出了最小分配成本問題 (Minimum Distribution Cost Problem；MDCP)。

針對具有 D-node 的特殊網路架構 (以下簡稱為分配網路)，大部分的相關文獻均著重於如何有效地求取滿足系統需求的最小成本。然而，產品在經由工作站處理並分級後，的確也可能會因為處理過程的狀況不同而導致其容量狀態不同；因此，如何在各弧具有多容量狀態之分配網路上計算其系統可靠度應有研究的價值。

舉例來說，圖 1.2是一個具有單起訖點組合且含有 D-node 的分配網路圖，其各弧之容量是一組具有多狀態機率分配向量。圖中起點為節點 1，訖點為節點 4，節點 2 為分配節點；流入分配節點 2 的流量有 20% 將流入節點 3，80% 流入節點 4；弧容量向量中每一個狀態各對應到一個機率值。在節點不會失效的前提下，從節點 1 成功運送單一商品 d 單位的需求到節點 4 的機率值即為該系統成功運送 d 單位需求的可靠度。通常，該可靠度的計算方法乃將節點 1 到節點 4 的所有可能路徑列出，判斷該路徑是否可以滿足運送需求量的條件，再將每一條可成功輸送需求量的路徑之機率利用聯集方式加總以計算整個系統的可靠度。然而，因為路徑個數可能會隨著弧的個數呈指數成長，且其行經路徑上一旦經過分配節點又會因為分流比例而造成弧流量不再是整數值；這些情況皆有別於以往網路可靠度的研究假設。基於包含分配節點的分配網路架構之特殊性，探討並發展分配網路上的可靠度計算方式應為一重要之研究議題。



(a)

arc	State probability				
	0	1	2	3	4
a_1	0.01	0.07	0.01	0.91	
a_2	0.01	0.03	0.1	0.16	0.7
a_3	0.02	0.98			
a_4	0.01	0.09	0.9		
a_5	0.03	0.01	0.11	0.85	

(b)

圖 1.2: 多容量狀態之分配網路圖範例

1.2 研究目的

由於網路上各弧之容量具有多種狀態的機率分配，且一旦流量流入D-node必定依照已知的比例流出，因此本論文所探討之網路可靠度是一個全新的研究議題。此外，由於分流比例為實數，因此即便流入D-node的流量為整數，其流出D-node的流量也可能不再是整數；不同於文獻中常見的整數容量狀態的可靠度計算方式，實數之流量將會大幅增加其可靠度計算上的困難。

本研究首先試圖修改Wang and Lin (2005)簡化網路圖的方法，以達到加速計算分配網路之可靠度的目的，接著試圖克服因為D-node所造成計算網路可靠度上的困難，發展一套新方法以計算具多容量狀態之分配網路的系統可靠度，最後並探討網路可靠度在此特殊分配網路之隨機最短路徑與網路設計問題的相關應用。

1.3 研究範圍與限制

關於網路可靠度的實際應用過於廣泛，本研究無法全部涵蓋。為了突顯因為分配網路之架構特殊性而導致可靠度計算方式上的困難，本研究將做出以下限

制，簡化其他的變動因素：

1. 流量守恆，即流入節點的流量必等於流出的流量。
2. 不考慮節點的容量限制，也就是多狀態的情況只發生在弧容量上。
3. 網路中只存在單一起訖點組合。
4. 從D節點流出之弧其分流比例為界於(0,1)之實數，且從該節點流出弧其分流比例總和為100%。
5. 弧上容量狀態發生的機率總和為1。

1.4 論文架構

本論文在第二章回顧了近幾年來可靠度的發展以及包含D-node的分配網路之相關文獻，並說明在具有D-node的網路架構上計算可靠度所遇到的困難；第三章針對網路圖可能出現的五種狀況，提出了簡化與整合多容量狀態分配網路的過程，以試圖降低計算其可靠度的困難；第四章提出了兩套計算多容量狀態分配網路之系統可靠度的演算法；第五章討論了多容量狀態分配網路在最短路徑與網路設計問題上的相關議題，而第六章則對整篇文章進行總結與提出後續可能的相關研究方向建議。

第二章

文獻探討

本章首先介紹網路可靠度的相關基本定義與符號；接著回顧文獻中各種計算可靠度的方法，包括數種d-MPs與d-MCs的尋找方式（d-MPs即為自起點運送至少d單位到需求點的MPs，而d-MCs為該運送方式之割集）；最後闡述含有D-node的分配網路之特殊架構，並說明在此特殊的網路架構下計算網路可靠度可能產生的困難。

2.1 可靠度相關符號與指標定義

假設我們欲探討的網路圖為$G = (N, A)$，其中A表示弧的集合$(|A| = m)$，而N表示節點的集合$(|N| = n)$。$(i, j) \in A$表示一條從節點i指向節點j的弧。在一個弧具有隨機容量狀態的網路圖中，每條弧(i, j)上具有$K_{ij} \geq 2$個以上的狀態$u_{ij}^{k_{ij}}$，其中$k_{ij} \in [1, K_{ij}]$，每一個容量狀態發生的機率為$p_{ij}^{k_{ij}}$，因此每一個弧(i, j)上的容量狀態向量$\mathbf{u}_{ij}$發生的機率可被設為向量$\mathbf{p}_{ij}$，如圖2.1所示。

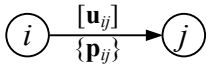


圖 2.1: 弧上機率分配示意圖

目前的文獻中用以衡量系統可靠度的指標可分成兩類：其一為「系統至少能滿足需求量d的可靠度」；假設系統容量狀態為$V(X)$，其中X為弧之容量，則此可靠度指標便是計算$V(X) \geq d$的機率，即$\Pr(V(X) \geq d)$，此指標利用d-MPs下界點的概念來確保該系統至少能滿足d單位的需求。為了要找尋下界點，可行的容

量向量將常需要兩兩比較，因此定義$Y \geq X$為$(y_1, y_2, \ldots, y_m) \geq (x_1, x_2, \ldots, x_m)$，其中$y_k \geq x_k$，$\forall k \in A$；而$Y > X$則表示為$(y_1, y_2, \ldots, y_m) > (x_1, x_2, \ldots, x_m)$，其中$Y \geq X$必須先被滿足且其中至少有一項元素$y_k$與$x_k$滿足$y_k > x_k$。由於先前假設網路中的元素相互獨立，不會影響彼此，因此$\Pr(Y \geq X)$可表示成$\prod_k \Pr(y_k \geq x_k)$。找到所有的下界點$X_i = (x_{i1}, x_{i2}, \ldots, x_{im})$後，將這些下界點的容量狀態取聯集後，系統可靠度便可藉由計算大於或等於該聯集事件的機率而得，即$\Pr(X|X \geq \bigcup_i X_i)$。

另一個計算可靠度的指標為「系統最大流量為d單位的可靠度」，也就是$\Pr(V(X) \leq d)$。根據max-flow min-cut理論，系統的最大流量便是最小割值，因此若要滿足系統最大流量為d單位，則必須找尋最小割值為d單位的容量狀態向量，d-MCs上界點的概念即為了要滿足最小割值為d單位而發展出來。計算此可靠度指標仍是利用兩兩比較後獲得d-MCs上界點，再將所有上界點$X_i = (x_{i1}, x_{i2}, \ldots, x_{im})$聯集後計算小於或等於該聯集事件的機率而得，即$\Pr(X|X \leq \bigcup_i X_i)$。

2.2 網路可靠度相關之文獻

最初之網路可靠度議題旨在探討系統中某一元件失效對系統所造成的影響。本論文一開始在第一章所提及的失效機率即在討論單一元件成功與失效的二元狀態。其後亦有不少的研究探討網路中某些元素(譬如弧之長度或容量)為多狀態情況下之可靠度相關議題。其中，弧具有多容量狀態下的最大流量問題之可靠度議題較被人廣為探討。除了弧可能具有隨機多容量狀態之外，節點也可能具有隨機多容量狀態；而這種情況雖然可將該節點分成兩個完美節點再轉換成用一個隨機多容量狀態的弧來連結，但這種作法需要花費額外的時間與空間。此外，我們亦回顧了更貼近實際狀況的可靠度相關議題。例如考慮商品運送時在每個弧上會有不同花費所衍生的預算限制下之可靠度議題，以及在弧或節點具多容量狀態的網路中同時運送不同產品的可靠度相關議題等等。

2.2.1 二元狀態之網路可靠度

在二元狀態的網路可靠度問題中，可成功連結起訖點的機率即為其系統之可靠度。而最直覺的可靠度計算方式便是窮舉出所有的成功與失效狀態，把成功連結起訖點之所有路徑的機率加總即可得系統可靠度。其計算方式如式子2.1所示：

$$Rel(G) = \sum_{A^S \in O} \prod_{e \in A^S} p_e \prod_{e \notin A^S} (1 - p_e) \tag{2.1}$$

其中 $Rel(G)$ 表示網路圖 $G = (N, A)$ 的可靠度，$A^S \subseteq A$ 代表可運作的弧的集合(S 為成功運作的情況)，O 表示所有成功運作的狀態（路徑集合），而 p_e 就是弧 e 可成功運作的機率。以圖2.2為例，p_e 表示元素 e 成功運作的機率。在此我們假設每條弧成功運作的機率皆為0.99，則可利用每條弧成功與失效的情況列舉出系統運作與否的所有可能組合。由於圖2.2的網路是由6條弧所組成的，因此總共有 $2^6 = 64$ 種可能的系統狀態組合，但其中僅有32種狀態組合能使系統運作。表2.1只列出此例中能使系統成功運作的所有狀態組合，其中 S 為成功運作的情況，F 為失效情況，而這些機率的加總即為此系統之可靠度(0.99979805)。

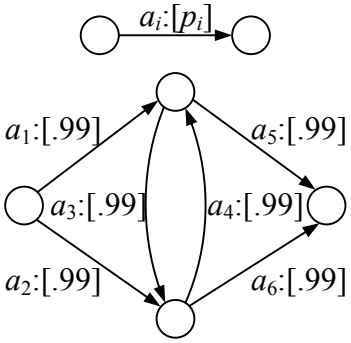


圖 2.2: 二元狀態隨機網路圖

此種可靠度計算方法雖然簡單，但非常沒有效率，因此Moskowitz (1958)和Mine (1959)提出了Factoring或稱為pivotal decomposition等分解方法(如式子2.2所示)來改善其演算效率，利用這種分解方法可省去一些不必要的狀態列舉過程。

$$Rel(G) = p_e Rel(G \cdot e) + (1 - p_e) Rel(G - e) \tag{2.2}$$

表 2.1: 圖 2.2之系統成功狀態列表

	a_1	a_2	a_3	a_4	a_5	a_6	System condition	Probability
1	S	S	S	S	S	S	S	$0.99^6 = 0.941480149$
2	S	S	S	S	S	F	S	$0.99^5 \times 0.01 = 0.0095099$
3	S	S	S	S	F	S	S	$0.99^5 \times 0.01 = 0.0095099$
4	S	S	S	F	S	S	S	$0.99^5 \times 0.01 = 0.0095099$
5	S	S	F	S	S	S	S	$0.99^5 \times 0.01 = 0.0095099$
6	S	F	S	S	S	S	S	$0.99^5 \times 0.01 = 0.0095099$
7	F	S	S	S	S	S	S	$0.99^5 \times 0.01 = 0.0095099$
8	S	S	S	F	S	F	S	$0.99^4 \times 0.01^2 = 9.60596 \times 10^{-5}$
9	S	S	F	S	S	F	S	$0.99^4 \times 0.01^2 = 9.60596 \times 10^{-5}$
10	S	F	S	S	S	F	S	$0.99^4 \times 0.01^2 = 9.60596 \times 10^{-5}$
11	F	S	S	S	S	F	S	$0.99^4 \times 0.01^2 = 9.60596 \times 10^{-5}$
12	S	S	S	F	F	S	S	$0.99^4 \times 0.01^2 = 9.60596 \times 10^{-5}$
13	S	S	F	S	F	S	S	$0.99^4 \times 0.01^2 = 9.60596 \times 10^{-5}$
14	S	F	S	S	F	S	S	$0.99^4 \times 0.01^2 = 9.60596 \times 10^{-5}$
15	F	S	S	S	F	S	S	$0.99^4 \times 0.01^2 = 9.60596 \times 10^{-5}$
16	S	S	F	F	S	S	S	$0.99^4 \times 0.01^2 = 9.60596 \times 10^{-5}$
17	S	F	S	F	S	S	S	$0.99^4 \times 0.01^2 = 9.60596 \times 10^{-5}$
18	F	S	S	F	S	S	S	$0.99^4 \times 0.01^2 = 9.60596 \times 10^{-5}$
19	S	F	F	S	S	S	S	$0.99^4 \times 0.01^2 = 9.60596 \times 10^{-5}$
20	F	S	F	S	S	S	S	$0.99^4 \times 0.01^2 = 9.60596 \times 10^{-5}$
21	S	S	F	F	S	F	S	$0.99^3 \times 0.01^3 = 9.70299 \times 10^{-7}$
22	S	F	S	F	S	F	S	$0.99^3 \times 0.01^3 = 9.70299 \times 10^{-7}$
23	S	F	F	S	S	F	S	$0.99^3 \times 0.01^3 = 9.70299 \times 10^{-7}$
24	F	S	F	S	S	F	S	$0.99^3 \times 0.01^3 = 9.70299 \times 10^{-7}$
25	S	S	F	F	F	S	S	$0.99^3 \times 0.01^3 = 9.70299 \times 10^{-7}$
26	S	F	S	F	F	S	S	$0.99^3 \times 0.01^3 = 9.70299 \times 10^{-7}$
27	F	S	S	F	F	S	S	$0.99^3 \times 0.01^3 = 9.70299 \times 10^{-7}$
28	F	S	F	S	F	S	S	$0.99^3 \times 0.01^3 = 9.70299 \times 10^{-7}$
29	S	F	F	F	S	S	S	$0.99^3 \times 0.01^3 = 9.70299 \times 10^{-7}$
30	F	S	F	F	S	S	S	$0.99^3 \times 0.01^3 = 9.70299 \times 10^{-7}$
31	S	F	F	F	S	F	S	$0.99^2 \times 0.01^4 = 9.801 \times 10^{-9}$
32	F	S	F	F	F	S	S	$0.99^2 \times 0.01^4 = 9.801 \times 10^{-9}$

其中，$Rel(G \cdot e)$ 表示網路圖 $G = (N, A)$ 在已知弧 e 可成功運作的情況下的系統可靠度，$Rel(G - e)$ 表示網路圖 $G = (N, A)$ 在已知弧 e 在失效情況下的系統可靠度。以圖 2.2為例來說明計算流程，我們可依圖 2.3的方式先將 G 拆成 $G \cdot a_3$ 與 $G - a_3$，則 $Rel(G) = p_{a_3} Rel(G \cdot a_3) + (1 - p_{a_3}) Rel(G - a_3) = 0.99 Rel(G \cdot a_3) + 0.01 Rel(G - a_3) = 0.99 \times 0.99979903 + 0.01 \times 0.99970102 = 0.99979805$。

除了以上的可靠度計算方式之外，文獻中另一常用之技巧即為利用

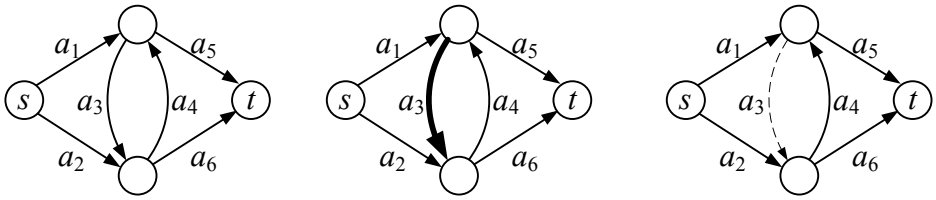


圖 2.3: 圖 2.2之拆解網路圖

最小路徑 (Minimal Path，MP) 和最小割集 (Minimal Cut，MC) 找出系統成功之機率的作法：假設網路圖 G 中的所有的 MP 為 $P_1, ..., P_h$，而 E_i 代表 P_i 上的弧皆可成功運作的事件，則可靠度即為這些事件的聯集機率，可利用計算聯集機率的 inclusion-exclusion 展開方式計算而得，亦即 $Rel(G) = \Pr[E_1 \cup E_2 \cup \cdots \cup E_h] = \sum_{j=1}^{h}(-1)^{j+1} \sum_{I\subseteq\{1,...,h\},|I|=j} \Pr[\bigcap_{i\in I} E_i]$，其中 $\bigcap_{i\in I} E_i$ 表示所有 $i \in I$ 的路徑 P_i 皆可成功運作的事件。以圖 2.2為例，自 s 到 t 共有 4 條 MP，分別為：$P_1 = \{a_1, a_5\}, P_2 = \{a_1, a_3, a_6\}, P_3 = \{a_2, a_4, a_5\}, P_4 = \{a_2, a_6\}$；則可靠度 $Rel(G) = \Pr[E_1 \cup E_2 \cup E_3 \cup E_4]$ 可表示成：$Rel(G) = (-1)^{1+1}\{\Pr[E_1]+\Pr[E_2]+\Pr[E_3]+\Pr[E_4]\}+(-1)^{1+2}\{\Pr[E_1 \cap E_2]+\Pr[E_1 \cap E_3]+\Pr[E_1 \cap E_4]+\Pr[E_2 \cap E_3]+\Pr[E_2 \cap E_4]+\Pr[E_3 \cap E_4]\}+(-1)^{1+3}\{\Pr[E_1 \cap E_2 \cap E_3]+\Pr[E_1 \cap E_2 \cap E_4]+\Pr[E_1 \cap E_3 \cap E_4]+\Pr[E_2 \cap E_3 \cap E_4]\}+(-1)^{1+4}\Pr[E_1 \cap E_2 \cap E_3 \cap E_4] = (-1)^2\{0.9801+0.970299+0.970299+0.9801\}+(-1)^3\{0.96059601+0.96059601+0.96059601+0.941480149+0.96059601+0.96059601\}+(-1)^4\{0.941480149+0.95099005+0.95099005+0.941480149\}+(-1)^5 \times 0.941480149 = 3.900798 - 5.744460199 + 3.784940399 - 0.941480149 = 0.99979805$。由此例可知，若網路圖中有 h 條 MP，光是要產生這些 MP 集合即可能需要花費 2^h 的時間，由此可知可靠度的計算十分耗費時間。

此外，由聯集之 inclusion-exclusion 展開過程可發現以下特性：若聯集事件的個數為奇數個，則其對可靠度的貢獻值為正值；反之（若為偶數個數的事件聯集），則為負值。因此 Satyanarayana and Prabhakar (1978) 便利用此奇偶個數之正負號的對應特性來改善 inclusion-exclusion 展開過程，以加速可靠度的計算。

2.2.2 具隨機多容量狀態弧之可靠度

二元狀態之可靠度僅能判斷起訖點連通的機率，無法進一步分析起訖點間可流通某些單位的機率。在具有多容量狀態弧的最大流量問題中，其弧之容量有多種可能的數值且各有其機率。令 P_{st} 表示所有 MP 所構成的集合且 P_l 表示其中一條 MP，f_l 表示經由 P_l 運送的流量，若系統的需求量為 d 單位，則表示流量經由所有可能的最小路徑 P_l 運送到訖點的流量總和必須至少 d 單位，即 $\sum_l f_l \geq d$。而流量 f_l 將受限於路徑 P_l 上的各弧之最大容量的最小者，若令 $\max(v)$ 表示向量 v 中最大的元素值，則流量 f_l 的限制可表示成 $f_l \leq \min_{(i,j)\in P_l}\{\max(\mathbf{u}_{ij})\}$。由於不同的最小路徑 P_l 所經過的弧可能相同，因此不同的最小路徑所輸送的流量流經相同的弧時，其流量總和也必須受限於該弧的最大容量，即 $\sum_{P_l \text{ passes through } (i,j)} f_l \leq \max(\mathbf{u}_{ij})$, $\forall(i,\, j) \in A$。Yeh (1998) 與 Lin et al. (1995) 即利用上述兩種限制式來找尋可行的 d-MPs，再將流量對應到弧，轉換成弧容量的狀態，即 $x_{ij} = \sum_{P_l \text{ passes through } (i,j)} f_l$, $\forall(i,\, j) \in A$；如此即可換算得到系統中所有的弧其流量所組成的可行狀態向量 $\mathbf{x}_{ij}$。

雖然利用流量限制與滿足需求量來找尋 d-MPs 的方法相當直覺，但仍必須要先找到從起點到訖點間的所有 MP；而且在找到可行流量之後，仍要將流量轉換成弧之容量狀態向量，才能進行可靠度的計算。再者，在一個網路圖中找出起訖點間的所有的 MP 所花費的時間會隨著網路中弧的個數呈指數成長；因此 Yeh (2005) 提出另一解法來避免列舉所有起訖點間之 MP 的必要性。該方法先列出所有滿足 $0 \leq x_{ij} \leq \max(\mathbf{u}_{ij}),\ \forall(i,j) \in A$ 的容量狀態向量 $X = (x_1, x_2, \ldots, x_m)$，再利用流量守恆 $(\sum_i x_{ij} = \sum_{i'} x_{ji'},\ \forall j \in N\backslash\{s,t\})$、起訖點的需求供給量為 d 單位 $(\sum_j x_{sj} = \sum_{j'} x_{j't} = d,\ \forall j, j' \in N\backslash\{s,t\})$ 等限制式刪去不可行之狀態組合，以找尋所有可行的 d-MPs。

除了利用 d-MPs 來計算滿足系統最小需求為 d 單位的可靠度指標外，Lin (2001) 亦利用 d-MCs 來計算系統最大流量為 d 單位的可靠度。MC 是一組弧的集合（割集），若將該組弧集合從網路中移除將造成網路圖的起訖點無法連結。根據 max-flow min-cut 的理論，網路圖中某割集的弧上之容量總和為所有割集中最低

者，其弧容量總和即為此網路的最大流量值。依此理論，我們可找出最大流量為d單位的割集以求得系統可靠度。

以下簡述d-MC尋找的方式：首先找出滿足弧容量總和為d的割集，將其它非割集上的弧容量設定成該弧容量的最大值，依此建構出弧容量狀態向量，再從其中刪除非上界點，即可求得d-MCs。

2.2.3 隨機容量狀態節點之可靠度

除了上一節所探討的節線可能具有隨機的多容量狀態外，本節探討節點具有隨機多容量狀態的可靠度計算方式。我們可將具有隨機多容量狀態的節點切割成兩個不具有隨機多容量狀態的可靠節點，其間以一條具有隨機多容量狀態的弧來連接，如此便可利用前一小節的技巧來計算可靠度。然而，此種網路轉換過程將會花費額外的時間與空間。

除了上述的網路轉換方式之外，Yeh (2001) 先將網路圖中的所有節點視為無隨機多容量狀態的情況以找出d-MPs，再藉由起訖點以外的中間節點之流量守恆以及起訖點的供給需求量為d單位等限制式來逐一找尋可行之節點容量狀態。此外，Lin (2002b) 利用d-MCs在單一商品、無成本預算限制的情況下，探討節點具有隨機容量狀態的可靠度，其方式與前一小節中假設節點完全可靠的可靠度計算方式十分類似。

2.2.4 具預算限制以及多元商品之網路可靠度

為了更能符合現實與管理層面的需求，將網路圖中的弧流量導入單位運輸成本以提供更多資訊的可靠度議題亦值得研究。其中Lin (2004) 考量了單位流量運輸成本，並確保其所找到之d-MPs也必須同時滿足總預算C的限制（亦即尋找(d, C)-MPs），再由(d, C)-MPs推算可滿足d單位需求且花費不超過C之系統可靠度。此外Lin (2002a) 再進一步將多元商品與成本預算限制皆納入考量以求解系統可靠度，其計算原理與過程大致與Lin (2004) 類似。

除了導入運輸成本以及總預算限制之外，Lin (2003) 探討了如何在具有隨機容量節點的網路中運送多元商品的可靠度議題。同時傳遞多元商品的情況

在網際網路的資料傳輸上相當常見，由於其商品種類為兩種或以上且具有成本限制，因此滿足各商品需求的可行MP之搜尋方式將更加複雜。在多元商品的情況下，可行之MP集合可表示成$(\mathbf{d}, C)$-MPs，其中$\mathbf{d}$表示多元的產品需求量向量。限制式$\sum_l f_l^k = d^k,\ \forall k$表示所有種類的商品之需求必須被滿足。由於不同的產品k在不同的組成元素$j' \in N \cup A$上具有不同的容量權重值$w_{j'}^k$，且$w_{j'}^k$不一定為整數，因此流經某網路元素$j' \in N \cup A$之流量應受限於該元素之容量最大值$\max(\mathbf{u}_{j'})$，亦即$\lceil \sum_k (w_{j'}^k \times \sum_{j' \in P_l} f_l^k) \rceil \leq \max(\mathbf{u}_{j'}),\ \forall j' \in N \cup A$；而總成本限制也就表示成$\sum_{j'} \sum_k (c_{j'}^k \times \sum_{j' \in P_l} f_l^k) \leq C$。接著再把流量轉換成容量狀態，即可得$(\mathbf{d}, C)$-MPs。

不同於Lin (2003)的方法，Yeh (2008)將Yeh (2005)處理多容量狀態的可靠度計算方式進一步修正，以探討多元商品且節點具有隨機多容量狀態的情況。其可行之容量狀態搜尋方法仍是在所有可能的弧容量組合中刪去違反以下四類限制式的組合而得：(1)弧與節點皆必須滿足流量守恆，即$\sum_i x_{ij}^k = \sum_j x_{jh}^k = x_j^k,\ \forall i, j, h \in N \backslash \{s, t\}, \forall k$；(2)起訖點的供給與需求必須滿足各類商品的需求量，即$x_s^k = x_t^k = d^k, \forall k$；(3)單位商品流量必須考慮其單位權重所佔的弧容量且同弧上之全部商品流量總和不得超過該弧之容量限制，即$\max\{\sum_k x_{j'}^k, \sum_k w_{j'}^k x_{j'}^k\} \leq \max(\mathbf{u}_{j'}),\ \forall j' \in N \cup A$；(4)各商品在各弧上的流量不能超過該商品的需求量與該弧之弧容量，即$0 \leq x_{j'}^k \leq \min\{d^k, \lceil \frac{\max(\mathbf{u}_{j'})}{w_{j'}^k} \rceil, \max(\mathbf{u}_{j'})\}, \forall k, \forall j' \in N \cup A$。

2.3 具有分配節點的網路問題相關文獻

前面幾個小節所回顧的網路可靠度議題皆建構在一般的網路架構上(亦即滿足節點流量守恆以及弧流量受限於其容量狀態)，而實際應用的網路模式可能會具有更多的額外限制式(side constraints)。舉例來說，Koene (1982)首先將生產製造過程中精化(refining)與混合(blending)的程序導入網路模式中。其中「精化」指的是生產過程中同樣的零組件經過某一個工作站處理後會依照特定的比例分流至其下一個工作站的所有過程；而「混合」則是零組件依特定的比例流至工作站後混合組裝成一項產品的過程。Koene (1982)將此類問題命名為pure processing

network problem(PNP)，並指出該網路模式可被應用於組裝生產線、排程以及電力分配等問題。其後，Chen and Engquist (1986) 利用基底解構的技術修正簡捷法 (Simplex Method) 來求解此類問題，並以數值測試說明其演算法的效率比線性規劃的通用軟體 MINOS 快上數倍；其後，Chang et al. (1989) 更進一步整合網路的資料結構與矩陣的稀疏性以改善 Chen and Engquist (1986) 的演算法，加快了求解 PNP 的速度。

此後 PNP 的相關研究發展停滯了一陣子，直到最近 Fang and Qi (2003) 重新再針對具分流比例限制之最小成本流量問題（或稱最小成本分配問題，Minimum Distribution Cost Problem；MDCP) 加以研究，相關的研究議題才又漸漸受到注意。Fang and Qi (2003) 將餾化（distillation，即精化）與組合（combination，即混合）的過程加以模式化，以描述實際製造過程中相關的處理程序。其實 Fang and Qi (2003) 所定義的特殊網路架構與 Koene (1982) 完全相同，然而其研究卻啟發了後續許多關於此特殊網路架構的新研究主題，其中包括最大分配流量問題（即 MDFP；請參閱 Sheu et al. (2006), Lin (2005)) 與最小分配成本問題（即 MDCP；請參閱 Lin (2006))。上述文獻皆在一個具有特殊節點 D-node 的網路架構上探討最佳化技術，其中 D(Distillation) 代表分配或餾化過程之意，也就是網路架構中具有分配或餾化的處理程序。每一個 D-node 代表著一組餾化過程的流量限制，也就是流量進入 D-node 會依照已知的特定比例自 D-node 分別流至各餾化弧。值得注意的是，上述探討製造網路的相關文獻皆假設網路上的所有參數（如弧容量或弧的長度等）皆為已知的定值，而且完全可靠（亦即不會發生隨機失效的情況）。然而如此的假設太過理想，因為再精密的儀器也會產生不良率，因此不能完全符合真實的製造環境；再者，產品在生產分級分裝運送至下一個加工程序時，也可能會因為運送的交通工具不同，甚至是運送過程中交通的流量無法確定等隨機產生的狀況而導致不可靠的情況發生。因此，本論文之後續章節將分配網路中所有之弧皆具隨機多容量狀態的情況列入考慮，以發展其有效的可靠度計算方式，而該可靠度可被視為一個績效指標以協助生產與製造管理的進行。

2.4 小結

由於分配網路之特殊架構及其餾化程序將使整數值的流量在流過D-node之後可能會成為實數值，導致其流量的分解方式有無限多種；此外，文獻中計算可靠度常用的「流量即容量」之簡化技巧無法適用，因此我們將於第四章提出以弧容量狀態的可能組合方式來計算可靠度。另外，網路圖的複雜程度也會大幅影響可靠度的計算速度，雖然Lin (2005)已提出了許多方法來簡化具有D-node的網路圖，但其方法僅適用於弧容量為確定值的情況，並未考慮多狀態的弧容量與機率分配的特性。因此我們在下一章將先修改Lin (2005)簡化網路圖的方法以簡化本論文所欲探討之具隨機多容量狀態的分配網路。

第三章

簡化與整合分配網路圖

由於分配網路具有特殊結構，我們在計算網路可靠度之前可先利用 Lin (2005) 所提出來的網路圖簡化概念，針對分配節點或是特殊的節點架構進行簡化與整合。因為分配網路上每個弧之弧容量具有多種狀態，因此必須針對弧容量所有組合的可能進行比較，整合後產生新的狀態機率。此一簡化與整合的過程可簡化原分配網路圖的複雜度，省下一些判斷流量是否滿足需求的時間，而加速可靠度的計算。

3.1 基本符號定義與說明

給定一個網路圖 $G = (N, A)$，其中包含 $|N| = n$ 個節點與 $|A| = m$ 條弧，除了起訖節點 s 與 t 外，剩下的節點可分成 N_D 與 N_O 兩群，其中 N_O 表示符合流量守恆的傳統轉運節點，而 N_D 表示餾化節點（D-nodes）所構成的集合。一個 D-node i 只能有一條流入弧 (l, i) 稱為母弧（*mother arc*），並且至少有兩條以上的流出弧稱為餾化弧（*distillation arc*）。每一條餾化弧 (i, j) 可用一個稱為餾化因子（*distillation factor*）的正實數 r_{ij} 來表示自其母弧 (l, i) 所分配到的流量百分比，也就是說對於任一 D-node i 而言，若已知其流入的流量 x_{li}，則可藉由分餾限制式 $x_{ij} = r_{ij}x_{li}$ 來求得其任一餾化弧 (i, j) 上的流量 x_{ij}。我們假設同一母弧分流出去的所有餾化弧之流量的百分比總和為 1；此外，我們將所有相鄰的 D-node 定義為一個 D-group，而 D-group 中最上層的 D-node 稱為母節點（*mother node*）；對每一個餾化弧 (i, j) 定義 $\overline{\mathbf{u}}_{ij} = \mathbf{u}_{ij}/r_{ij}$ 為其標準化的容量向量（*normalized capacity vector*），以方便之後的網路簡化整合過程。

圖 3.1(a) 表示一個未經簡化的分配網路圖，而該網路圖可經由簡化的過程而簡化成 3.1(g) 所呈現的網路圖。其中 3.1(a) 經由簡化虛線部分的單一轉運節點可形成 3.1(b) 的網路圖；3.1(b) 經由簡化虛線部分的迴圈弧可形成 3.1(c) 的網路圖；3.1(c) 經由簡化虛線部分的平行弧可形成 3.1(d) 的網路圖；3.1(d) 經由簡化虛線部分的單一轉運節點可形成 3.1(e) 的網路圖；3.1(e) 經由簡化虛線部分的 D 節點群組可形成 3.1(f) 的網路圖；3.1(f) 經由簡化虛線部分的平行弧可得到最終的 3.1(g) 的網路圖。

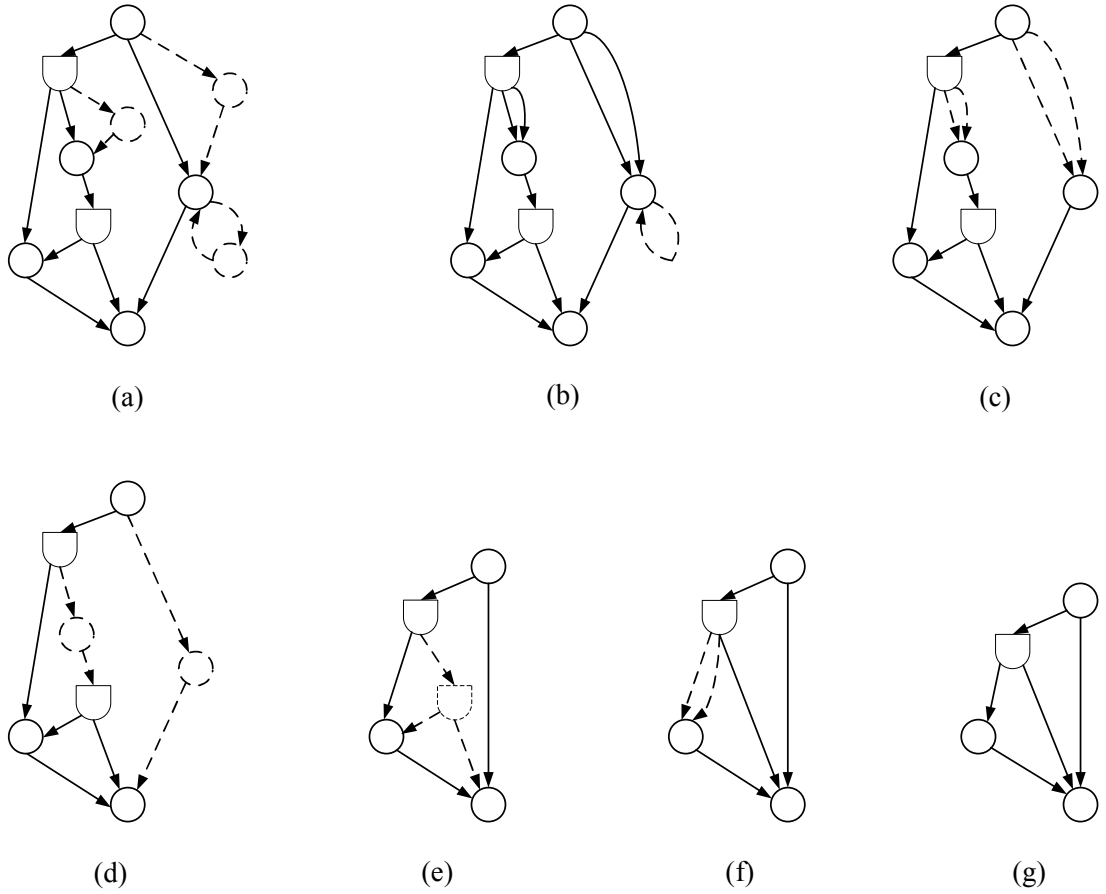


圖 3.1: 簡化網路圖的範例

令弧 (i,j) 之容量含有 K_{ij} 種狀態，以向量 $\mathbf{u}_{ij} = [u_{ij}^{k_{ij}} : k_{ij} = 1, 2, \ldots, K_{ij}]$ 表示之，其個別容量狀態依其大小以嚴格遞增方式排序（即只要 $1 \leq a < b \leq K_{ij}$ 則 $u_{ij}^a < u_{ij}^b$），且個別對應到一個特定的機率值 $p_{ij}^{k_{ij}}$。令 x_{ij} 表示弧 (i,j) 上的流量，

則弧(i,j)之可靠度即為$\sum_{k_{ij}:u_{ij}^{k_{ij}}\geq x_{ij}} p_{ij}^{k_{ij}}$，也就是經由弧$(i,j)$可成功運送流量$x_{ij}$單位的機率。令$\max(v)$表示容量向量$v$內的最大值；為了方便起見，在此先定義一個新的狀態向量$\mathbf{u}_{ij}^{\leq}(x)=[u_{ij}^{k_{ij}}:u_{ij}^{k_{ij}}<x,x],\forall(i,\,j)\in A$，其內含之元素為原向量中小於$x$的部分以及$x$本身，其中$x$為一給定之定值。當兩條弧$(i,j)$與$(j,k)$合併成一條新的弧$(i,k)$時，弧$(i,k)$上的容量向量$\mathbf{u}_{ik}$可藉由特殊的容量狀態向量乘積$\mathbf{u}_{ij}\otimes\mathbf{u}_{iq}=[\min\{u_{ij}^{k_{ij}},u_{jq}^{k_{jq}},\ \forall k_{ij},k_{jq}\}]$計算而得。值得注意的是，容量狀態的個數可能會因為合併的過程而減少（即$\min\{K_{ij},K_{jq}\}\leq K_{iq}\leq K_{ij}\cdot K_{jq}$，且$\max(\mathbf{u}_{iq})\leq\min\{\max(\mathbf{u}_{ij}),\max(\mathbf{u}_{jq})\}$）。而合併兩條平行弧$(i,j)$與$(i,j)'$將使其容量狀態向量成為$\mathbf{u}_{ij}=\mathbf{u}_{ij}\oplus\mathbf{u}_{ij'}=[u_{ij}^{k_{ij}}+u_{ij'}^{k_{ij'}},\ \forall k_{ij},k_{ij'}]$。以下將針對網路圖中可能出現的五種情況進行網路圖簡化與整合。

3.2 簡化D-groups

D-group是一組相鄰的D-node所組成的集合，要對這種情況進行簡化並整合，就必須由D-node的特性下手。由於流進D-node的流量必定依照分餾的比例流出，因此原本屬於不同弧的弧容量在整合形成新弧之時，其弧容量的機率分配也將隨之改變。其處理過程如下：相鄰兩個或多個D-node的分餾比例相乘以形成新的分餾比例，而弧容量的狀態則要先標準化到同一基準才能比較，取其較小的容量瓶頸作為新的容量狀態；各容量狀態之機率則由形成新容量狀態的所有可能之舊容量狀態組合的機率乘積加總而得。

如圖3.2(a)所示，路徑2-4-6可合併成一條弧$(2,\,6)$，而其合併後的容量狀態向量則變成$\mathbf{u}_{26}=r_{46}\mathbf{u}_{24}\otimes\mathbf{u}_{46}=[0,\,0.5,\,1,\,1.5,\,2]\otimes[0,\,1]=[0,\,0.5,\,1]$，也就是將弧$(2,\,4)$上的容量向量餾化到弧$(4,\,6)$上成為$r_{46}\mathbf{u}_{24}=[0,\,0.5,\,1,\,1.5,\,2]$；而合併後之弧$(2,\,6)$的容量狀態向量所對應的機率值為：

$$\hat{p}_{26}^{1}=\sum_{a,b:\min\{u_{24}^{a},u_{46}^{b}\}=0}p_{24}^{a}\cdot p_{46}^{b}=1/5+(1-1/5)\times 1/2=3/5;$$
$$\hat{p}_{26}^{2}=\sum_{a,b:\min\{u_{24}^{a},u_{46}^{b}\}=0.5}p_{24}^{a}\cdot p_{46}^{b}=1/5\times 1/2=1/10;$$
$$\hat{p}_{26}^{3}=\sum_{a,b:\min\{u_{24}^{a},u_{46}^{b}\}=1}p_{24}^{a}\cdot p_{46}^{b}=(1/5+1/5+1/5)\times 1/2=3/10.$$

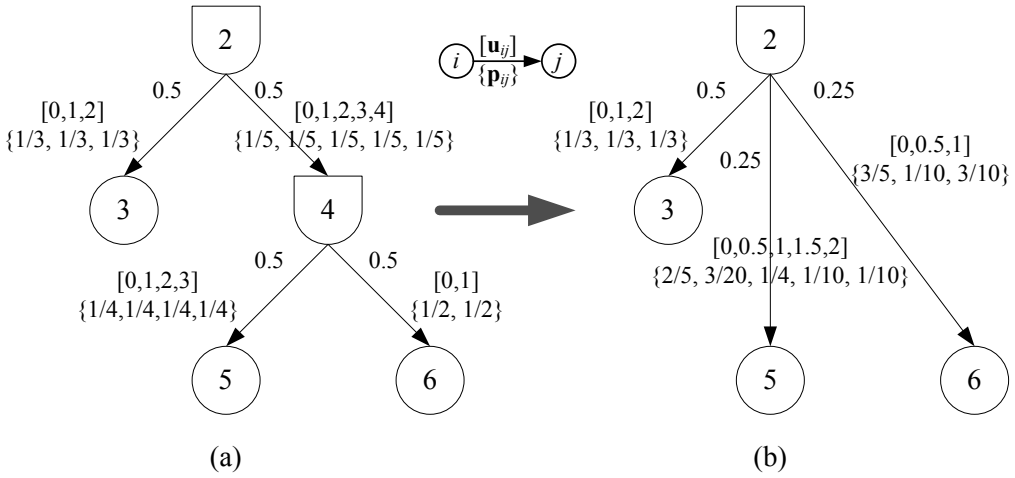


圖 3.2: 簡化D-groups的範例

3.3 簡化單一轉運節點

當網路圖中存在一個O-node且僅有一個流入弧與一個流出弧與其相連，則該O-node可被刪除，而其流入與流出弧亦可被整合成一條從流入弧起點到流出弧終點的弧。新弧上的容量狀態機率分配可依照原流入弧與流出弧的機率分配整合而成；弧之容量狀態取自於原流入弧與流出弧的容量狀態組合中較小者，而其機率則是形成該容量狀態的所有容量狀態組合之機率乘積總和。

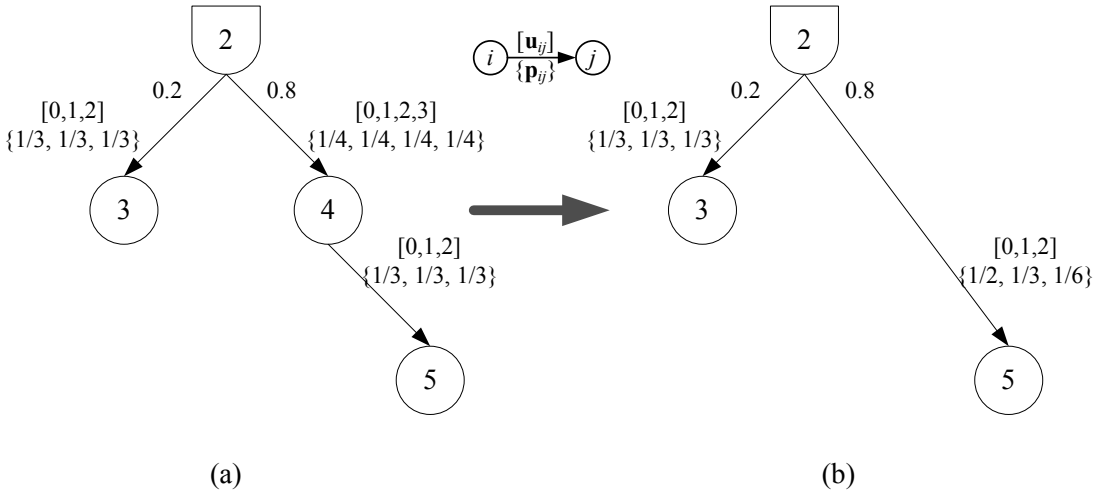


圖 3.3: 簡化單一轉運節點的範例

如圖3.3所示，3.3(a)中的弧(2, 4)與弧(4, 5)可整合成3.3(b)的新弧(2, 5)，而

容量狀態向量整合成$\mathbf{u}_{25} = \mathbf{u}_{24} \otimes \mathbf{u}_{45} = [0, 1, 2, 3] \otimes [0, 1, 2] = [0, 1, 2]$，各容量狀態所對應到的機率則計算如下：

$$\hat{p}_{25}^{1} = \sum_{a,b:\min\{u_{24}^{a},u_{45}^{b}\}=0} p_{24}^{a} \cdot p_{45}^{b} = 1/4 + (1 - 1/4) \times 1/3 = 1/2;$$
$$\hat{p}_{25}^{2} = \sum_{a,b:\min\{u_{24}^{a},u_{45}^{b}\}=1} p_{24}^{a} \cdot p_{45}^{b} = 1/4 \times (1 - 1/3) + (1/4 + 1/4) \times 1/3 = 1/3;$$
$$\hat{p}_{25}^{3} = \sum_{a,b:\min\{u_{24}^{a},u_{45}^{b}\}=2} p_{24}^{a} \cdot p_{45}^{b} = (1/4 + 1/4) \times 1/3 = 1/6.$$

3.4 簡化迴圈弧

雖然我們可以假設起始的網路圖不具有迴圈弧，但迴圈弧可能會因為其它的整合過程而出現(如圖3.1(b))。若某路徑經過自我迴圈弧的話，則其容量狀態向量必不是一組下界點，此時可直接刪除自我迴圈弧。以下之Lemma 1將證明只考慮下界點時的可靠度與額外考慮其他可行向量時之可靠度相同。是故，我們只需利用下界點即可計算可靠度。

Lemma 1. 令系統之容量狀態向量為$X = (x_1, x_2, \ldots)$。假設系統中有一下界點為$X' = (x_1', x_2', \ldots)$與一組可行容量狀態向量$X'' = (x_1'', x_2'', \ldots)$且$X'' > X'$，若將此可行向量一併納入可靠度的計算之中，則可得到下列算式：

(i) $\Pr(X \geq X'') = \Pr(X \geq X' \cap X'')$

(ii) $\Pr(X \geq X' \cup X'') = \Pr(X \geq X')$

Proof:

(i) $\Pr(X \geq X' \cap X'') = \Pr(X \geq X', X \geq X'')$
$= \Pr(x_1 \geq x_1', x_1 \geq x_1'') \times \Pr(x_2 \geq x_2', x_2 \geq x_2'') \times \cdots$
$= \Pr(x_1 \geq x_1'') \times \Pr(x_2 \geq x_2'') \times \cdots = \Pr(X \geq X'')$

(ii) $\Pr(X \geq X' \cup X'') = \Pr(X \geq X') + \Pr(X \geq X'') - \Pr(X \geq X' \cap X'') = \Pr(X \geq X')$

□

由上述證明可得知，若將下界點以外的可行容量狀態向量亦納入可靠度的計算過程的話，則所求得的可靠度等同於僅利用下界點計算而得之可靠度；同理可推論至多組下界點與可行向量的情況，因此在計算可靠度時僅需考慮下界點即可。

以圖 3.4為例，假設系統希望運送 1 單位的需求量從節點 1 至節點 3，若考慮自我迴圈弧的存在，則系統容量狀態向量 $X = (x_1, x_2, x_3)$，其元素依序代表弧 $(1, 2)$, $(2, 2)$, $(2, 3)$ 之某一容量狀態，而可行之容量狀態向量共有兩個，分別是 $X' = (1, 0, 1)$ 與 $X'' = (1, 1, 1)$。由於 $X'' \geq X'$，因此系統的可靠度為 $\Pr(X \geq X' \cup X'') = \Pr(X \geq X') = \Pr(x_1 \geq 1) \times \Pr(x_2 \geq 0) \times \Pr(x_3 \geq 1) = \Pr(x_1 \geq 1) \times 1 \times \Pr(x_3 \geq 1) = \Pr(x_1 \geq 1) \times \Pr(x_3 \geq 1)$，完全不受迴圈弧 $(2, 2)$ 的影響。由此可知，即使我們逕自刪除迴圈弧亦不會影響系統之可靠度。

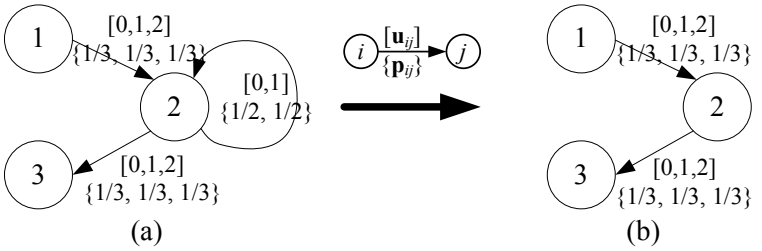


圖 3.4: 簡化迴圈弧的範例

3.5 簡化平行弧

雖然在原本的網路圖中可能不會出現平行弧的情況，但經由簡化與整合之後亦可能出現平行弧，因此本節將針對平行弧的兩種特殊情況分別加以簡化與整合。

3.5.1 僅連結 O 節點之平行弧簡化程序

若是平行弧連結的兩個端點皆為 O 節點，由於每條平行弧的起訖點皆相同，則可將所有相同起訖點的數個平行弧整合成一條新的弧，而新弧上的最大容

量狀態即為原數個平行弧最大容量狀態的加總；其弧之容量狀態為原平行弧上所有容量狀態組合之和，機率為該組合容量狀態機率之乘積總和，如此即可計得新弧上的容量狀態機率分配。

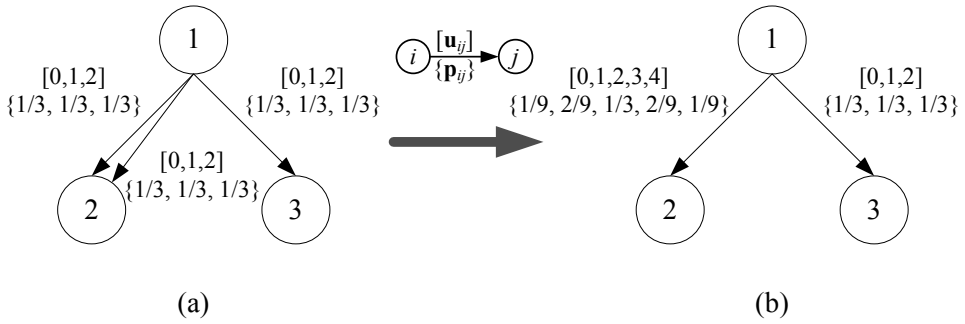


圖 3.5: 簡化連結O節點的平行弧範例

如圖3.5所示，圖3.5(a)的兩條平行弧$(1,\,2)$與$(1,\,2)'$。其中$\mathbf{u}_{12} = \mathbf{u}_{12'} = [0,\,1,\,2]$，合併後新弧的容量狀態向量變成$\mathbf{u}_{12} = \mathbf{u}_{12} \oplus \mathbf{u}_{12'} = [0,\,1,\,2,\,3,\,4]$，且每個容量狀態所對應到的機率值計算如下：

$$\hat{p}_{12}^{1} = \sum_{a,b:u_{12}^{a}+u_{12'}^{b}=0} p_{12}^{a} \cdot p_{12'}^{b} = 1/3 \times 1/3 = 1/9;$$

$$\hat{p}_{12}^{2} = \sum_{a,b:u_{12}^{a}+u_{12'}^{b}=1} p_{12}^{a} \cdot p_{12'}^{b} = 1/3 \times 1/3 + 1/3 \times 1/3 = 2/9 \cong 0.1;$$

$$\hat{p}_{12}^{3} = \sum_{a,b:u_{12}^{a}+u_{12'}^{b}=2} p_{12}^{a} \cdot p_{12'}^{b} = 1/3 \times 1/3 + 1/3 \times 1/3 + 1/3 \times 1/3 = 1/3;$$

$$\hat{p}_{12}^{4} = \sum_{a,b:u_{12}^{a}+u_{12'}^{b}=3} p_{12}^{a} \cdot p_{12'}^{b} = 1/3 \times 1/3 + 1/3 \times 1/3 = 2/9;$$

$$\hat{p}_{12}^{5} = \sum_{a,b:u_{12}^{a}+u_{12'}^{b}=4} p_{12}^{a} \cdot p_{12'}^{b} = 1/3 \times 1/3 = 1/9.$$

3.5.2 連結D-node之平行弧簡化程序

若是平行弧的起始節點為D-node，每條流出弧上的流量會依照特定的分餾比例流出，因此平行弧的整合方式也會受其比例影響。其整合方式首先將D-node流出之平行弧的容量狀態各自依其分餾比例轉換回其所對應之D-node流入弧的容量狀態(亦即將所有連結D-node之各弧容量狀態加以標準化)，接著尋找其中

之瓶頸容量狀態當作新的流入弧之容量狀態，再將該容量狀態依照原平行弧上的分餾比例的總和轉換回流出弧的容量狀態並計算其對應之容量狀態機率。

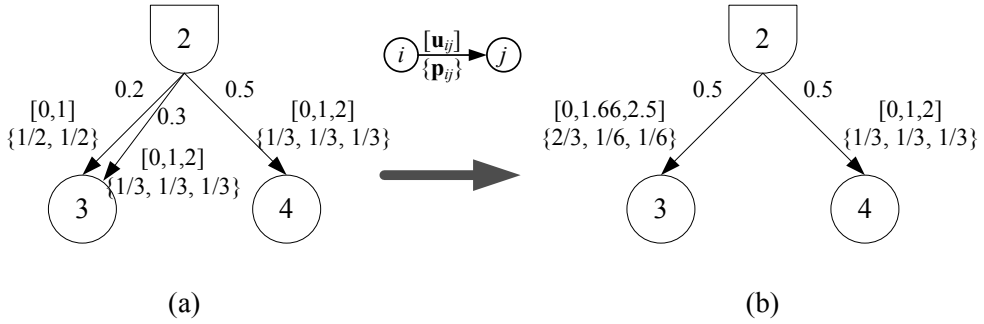


圖 3.6: 簡化平行弧連結一個 D-node 範例

如圖 3.6所示，圖 3.6(a) 有兩條平行弧 $(2,3)$ 及 $(2,3)'$。其中 $\mathbf{u}_{23} = [0,\ 1]$ 而 $\mathbf{u}_{23'} = [0,\ 1,\ 2]$。對於所有的 $2 \times 3 = 6$ 種可能容量狀態組合，我們必須先利用標準化的容量狀態向量找尋瓶頸弧，也就是對 $\overline{\mathbf{u}}_{23} = [0/0.2,\ 1/0.2] = [0,\ 5]$ 與 $\overline{\mathbf{u}}_{23'} = [0/0.3,\ 1/0.3,\ 2/0.3] = [0,\ 3.\overline{3},\ 6.\overline{6}]$ 兩者的容量狀態加以合併，再取其較小者當作瓶頸容量狀態。以此例而言，只會形成三種容量狀態，即 $\overline{\mathbf{u}}_{23} \otimes \overline{\mathbf{u}}_{23'} = [0,\ 3.\overline{3},\ 5]$，故合併後的容量狀態向量就變成 $\mathbf{u}_{23} = (r_{23} + r_{23'}) \times (\overline{\mathbf{u}}_{23} \otimes \overline{\mathbf{u}}_{23'}) = [0,\ 1.\overline{6},\ 2.5]$，而其所對應的機率值計算如下：

$$\hat{p}^1_{23} = \sum_{a,b:(r_{23}+r_{23'})\times\min\{\overline{u}^a_{23},\overline{u}^b_{23'}\}=0} p^a_{23} \cdot p^b_{23'} = 1/2 + 1/2 \times 1/3 = 2/3;$$

$$\hat{p}^2_{23} = \sum_{a,b:(r_{23}+r_{23'})\times\min\{\overline{u}^a_{23},\overline{u}^b_{23'}\}=1.\overline{6}} p^a_{23} \cdot p^b_{23'} = 1/2 \times 1/3 = 1/6;$$

$$\hat{p}^3_{23} = \sum_{a,b:(r_{23}+r_{23'})\times\min\{\overline{u}^a_{23},\overline{u}^b_{23'}\}=2.5} p^a_{23} \cdot p^b_{23'} = 1/2 \times 1/3 = 1/6.$$

3.6 簡化不協調的母弧與餾化弧之容量狀態

在具有 D-node 的分配網路圖中，由於流入 D-node 的流量必須依照特定比例分流至其相鄰節點的特性，瓶頸容量可能會產生在 D-node 的流入弧或流出弧上，藉由標準化容量狀態的整合過程可使所有與 D-node 相連之弧的容量狀態一致之效果。

先將所有與D-node相連的餾化弧之容量狀態向量除以其餾化比例以轉換成其母弧之容量狀態基準，接著比較這些標準化過後之各弧最大容量狀態，取其最小者當瓶頸值。把所有小於該瓶頸值的容量狀態依其大小以遞增排序納入母弧之新容量狀態向量中，再以該母弧上的容量狀態向量分別乘上各餾化弧之原餾化比例，即可獲得各餾化弧之新容量狀態向量。而其最大容量狀態的機率則可由該弧上原本大於或等於該最大容量狀態的所有機率加總而得，其它小於該弧上新的最大容量狀態之原各容量狀態機率則維持不變，而新增的容量狀態機率則設為零。值得注意的是雖然此轉換過程可能會增加新的機率為零之容量狀態，然而這個結果其實反而更有利於後續可靠度的計算，因為這個簡化整合過程將使所有與D-node連結之弧的容量狀態具有一致的影響效果，因此在之後的可靠度計算即可省去母弧與餾化弧各容量狀態影響性之反覆檢驗過程，以縮短可靠度的計算時間。

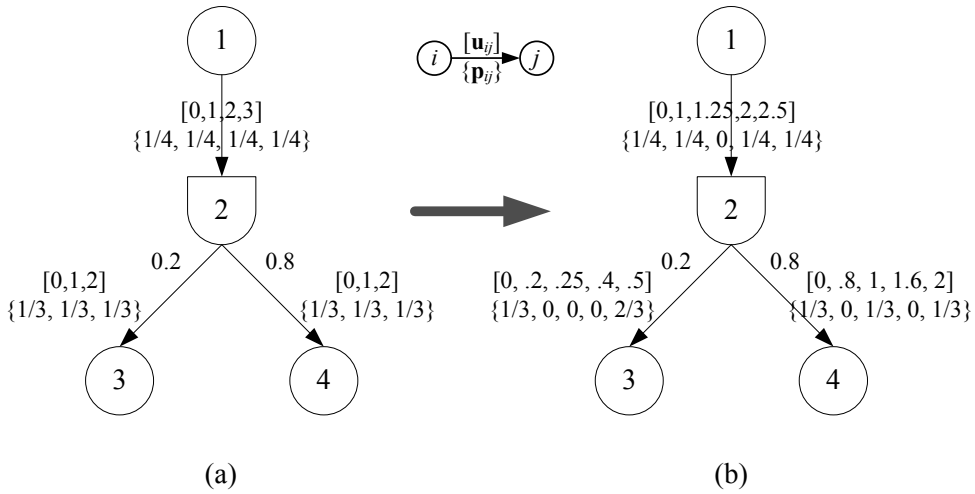


圖 3.7: 簡化不協調容量範例

以圖3.7為例，先找出瓶頸弧之容量狀態$\min\{\max(\mathbf{u}_{12}), \max(\overline{\mathbf{u}}_{23}), \max(\overline{\mathbf{u}}_{24})\} = 2.5$，接著把所有標準化的容量狀態向量$\overline{\mathbf{u}}_{23} = [0/0.2,\ 1/0.2,\ 2/0.2] = [0,\ 5,\ 10]$與$\overline{\mathbf{u}}_{24} = [0/0.8,\ 1/0.8,\ 2/0.8] = [0,\ 1.25,\ 2.5]$結合形成新的母弧容量狀態向量，即$\hat{\mathbf{u}}_{12} = \mathbf{u}_{12}^{\leq}(2.5) \cup \overline{\mathbf{u}}_{23}^{\leq}(2.5) \cup \overline{\mathbf{u}}_{24}^{\leq}(2.5) = [0,\ 1,\ 1.25,\ 2,\ 2.5]$，而餾化弧的容量狀態向量即可由母弧的容量狀態向量乘上分餾比例而計得，即$\hat{\mathbf{u}}_{23} = r_{23}\hat{\mathbf{u}}_{12} = 0.2 \times [0,\ 1,$

1.25, 2, 2.5] = [0, 0.2, 0.25, 0.4, 0.5] 與 $\hat{\mathbf{u}}_{24} = r_{24}\hat{\mathbf{u}}_{12} = 0.8 \times [0, 1, 1.25, 2, 2.5] = [0, 0.8, 1, 1.6, 2]$。母弧 $(1, 2)$ 與餾化弧 $(2, 3)$ 與 $(2, 4)$ 的各容量狀態之機率計算如下：

弧 $(1, 2)$	弧 $(2, 3)$	弧 $(2, 4)$
$\hat{p}_{12}^1 = p_{12}^1 = 1/4;$	$\hat{p}_{23}^1 = p_{23}^1 = 1/3;$	$\hat{p}_{24}^1 = p_{24}^1 = 1/3;$
$\hat{p}_{12}^2 = p_{12}^2 = 1/4;$	$\hat{p}_{23}^2 = 0;$	$\hat{p}_{24}^2 = 0;$
$\hat{p}_{12}^3 = 0;$	$\hat{p}_{23}^3 = 0;$	$\hat{p}_{24}^3 = p_{24}^2 = 1/3;$
$\hat{p}_{12}^4 = p_{12}^3 = 1/4;$	$\hat{p}_{23}^4 = 0;$	$\hat{p}_{24}^4 = 0;$
$\hat{p}_{12}^5 = p_{12}^4 = 1/4.$	$\hat{p}_{23}^5 = p_{23}^2 + p_{23}^3 = 2/3.$	$\hat{p}_{24}^5 = p_{24}^3 = 1/3.$

3.7 小結

簡化與整合過後的分配網路圖可省去許多後續計算可靠度時的反覆檢驗步驟，雖然這些簡化或整合的步驟可能增加各弧之容量狀態個數，然而此舉的確可加速可靠度的計算，這個部分將在第四章介紹可靠度的計算方式時一併說明。

第四章

可靠度的計算與相關應用

由於本文章所提出的網路圖中，存在具有分餾特性的D-node，且因各餾化比例必為界於(0, 1)之間的實數，故當流量流經D-node時便可能不再是整數值。然而，在文獻回顧中所提及計算網路可靠度的演算法，大都只能處理整數值的流量。以最單純的單一商品、不考慮節點失效情況以及成本預算的情況來說，Yeh (1998)與Lin et al. (1995)提出了用$(1)\sum_l f_l = d$; $(2) f_l \leq \min_{(i,j)\in P_l}\{\max(\mathbf{u}_{ij})\}, \forall P_l$; $(3)\sum_{P_l \text{ passes through } (i,j)} f_l \leq \max(\mathbf{u}_{ij}), \forall(i, j)$等限制式來找尋可行的流量運送方式，再轉換成容量狀態向量$x_{ij} = \sum_{P_l \text{ passes through } (i,j)} f_l, \forall(i, j)$。若只考慮整數值的流量，則此方式所產生的可行向量可由路徑P_l所經過的弧以及與需求量d推導出所有可能的整數流量組合；然而，倘若流量為實數值，則以上的流量組合推導方式將會因為實數的稠密性而有無限多種可能，使問題變得無法處理。為了試圖解決因實數流量導致可靠度計算上的困難，本章提出MSE及PART-D兩種可靠度演算法來計算具有D-node之分配網路可靠度。

4.1 MSE(Modified State Enumeration)演算法

在過去的文獻中，針對處理單一商品、不考慮節點失效情況以及成本預算的網路可靠度問題，Yeh (2005)提出用弧容量是否能負荷流量的角度來產生可行的容量狀態向量。本章所提之第一個可靠度演算法在探討可能的弧容量組合時，將檢視其容量向量能否提供一足夠的流量來滿足系統需求，因此將與Yeh (1998)與Lin et al. (1995)的方法部分類似。本章試圖修改Yeh (2005)提出的演算

流程，以配合D-node的特性，發展出可計算具有D-node網路架構的可靠度演算法，稱之為MSE(Modified State Enumeration)法。

4.1.1 MSE演算法步驟

令$CCC = [u_{ij}^{k_{ij}} : (i,j) \in A]$與$QCC = [u_{ij}^{k_{ij}^*} : (i,j) \in A]$分別表示$m$維候選的容量狀態組合向量(Candidate Capacity Combination vector)與合格的容量狀態組合向量(Qualified Capacity Combination vector)。其中，向量中的每一元素代表某一選定的容量狀態；而合格與否則需視其是否為可遵守容量限制並滿足需求等情況之容量向量下界點而定。以下步驟將說明找尋QCC的流程：

Step 0. 將網路圖中與D-node相連的弧之容量狀態進行整合正規化。

Step 1. 針對網路圖中每一條弧(i,j)窮舉其所有之容量狀態向量組合，亦即列舉所有的$CCC = [u_{ij}^{k_{ij}} : (i,j) \in A, k_{ij} = 1,2,\ldots,K_{ij}]$。以表4.1為例，整個系統狀態向量為$X = (u_{a_1}^{k_{a_1}}, u_{a_2}^{k_{a_2}}, u_{a_3}^{k_{a_3}}, u_{a_4}^{k_{a_4}}, u_{a_5}^{k_{a_5}}) = (u_{12}^{k_{12}}, u_{13}^{k_{13}}, u_{23}^{k_{23}}, u_{24}^{k_{24}}, u_{34}^{k_{34}})$，則依序列舉出所有候選的容量狀態組合向量$CCC$為(0, 0, 0, 0, 0), (0, 0, 0, 0, 1), ..., (2.5, 4, 0.5, 2, 3)。

Step 2. 利用下列三條限制式來檢驗每一個CCC：

(1) 起點s的所有流出弧，其提供的弧容量之總和必須大於或等於需求量d單位，即$\sum_{(s,j)\in A} u_{sj}^{k_{sj}} \geq d$；

(2) 終點t的所有流入弧，其提供的弧容量之容量總和必須大於或等於需求量d單位，即$\sum_{(i,t)\in A} u_{it}^{k_{it}} \geq d$；

(3) 對於所有的D-node j，其母弧的容量應等於其所有餾化弧所提供的弧容量總和，且每一條餾化弧的弧容量應等於母弧的弧容量乘上分流比例，即$\sum_{(j,q)\in A} u_{jq}^{k_{jq}}$與$u_{jq}^{k_{jq}} = r_{jq} u_{ij}^{k_{ij}}, \forall (i,j) \in A, (j,q) \in A, k_{jq} = 1, 2, \ldots, K_{jq}$。值得注意的是，在前一章3.6節中曾提及的精簡網路程序可能會增加一些機率為零的容量狀態，然而諸如這些不可行的容量狀態組合皆可在此處使用等號限制式來大幅度刪去。

Step 3. 檢查剩下的CCC是否為可行的狀態向量組合，以確定其必能運送d單位的需求量。

Step 4. 利用上述的限制式與條件刪除不可行的容量狀態組合之後，令剩下的v項CCC為$\overline{CCC}$集合，執行Algorithm 1之比較流程以找尋下界點，即可得到合格的容量組合向量QCC。

Algorithm 1 search for QCC\{input:$\overline{CCC}$, output:QCC\}

Begin
$I = \emptyset$,$QCC = \emptyset$
for $i = 1$ to v and $i \notin I$
  for $j = i + 1$ to v and $j \notin I$
    if $\overline{CCC}_i \geq \overline{CCC}_j$
      $I = I \cup \{i\}$, break;
    else if $\overline{CCC}_j > \overline{CCC}_i$
      $I = I \cup \{j\}$;
  $QCC = \overline{CCC}_i \cup QCC$;
End

令$E_i = \{X|X \geq QCC_i\}$表示大於或等於某一組合格的容量組合向量QCC_i之所有可能的系統容量狀態向量X所構成的集合，而$V(X)$代表整個系統在容量狀態向量為X的情況下可滿足的總需求量，則系統之可靠度即為系統$V(X) \geq d$的機率，且該值亦與所有E_i聯集的機率相同。亦即，$\Pr(V(X) \geq d) = \Pr(\bigcup_i E_i) = \Pr(\bigcup_i \{X|X \geq QCC_i\}) = \Pr(X|X \geq \bigcup_i QCC_i)$

4.1.2 MSE範例演練

圖4.1(b)是圖4.1(a)經簡化整合後所得到的具多容量狀態之分配網路。其中，節點s是起點，節點t是訖點，節點1是分配節點，需求量d=7單位，弧容量的機率分配如圖上所示。依循演算法的Step 1把所有的狀態組合向量列舉出來可得表4.1。

在列出所有可能的容量狀態組合向量之後，我們可根據Step 2的限制式$u_{s,1}^{k_{s,1}} + u_{s,2}^{k_{s,2}} \geq 7$, $u_{3,t}^{k_{3,t}} + u_{4,t}^{k_{4,t}} \geq 7$, $u_{1,2}^{k_{1,2}} + u_{1,3}^{k_{1,3}} = u_{s,1}^{k_{s,1}}$, $u_{1,2}^{k_{1,2}} = r_{1,2}u_{s,1}^{k_{s,1}} = 0.2u_{s,1}^{k_{s,1}}$及$u_{1,3}^{k_{1,3}} = r_{1,3}u_{s,1}^{k_{s,1}} = 0.8u_{s,1}^{k_{s,1}}$將不可行的狀態組合向量刪除。其後，留下來的候選狀

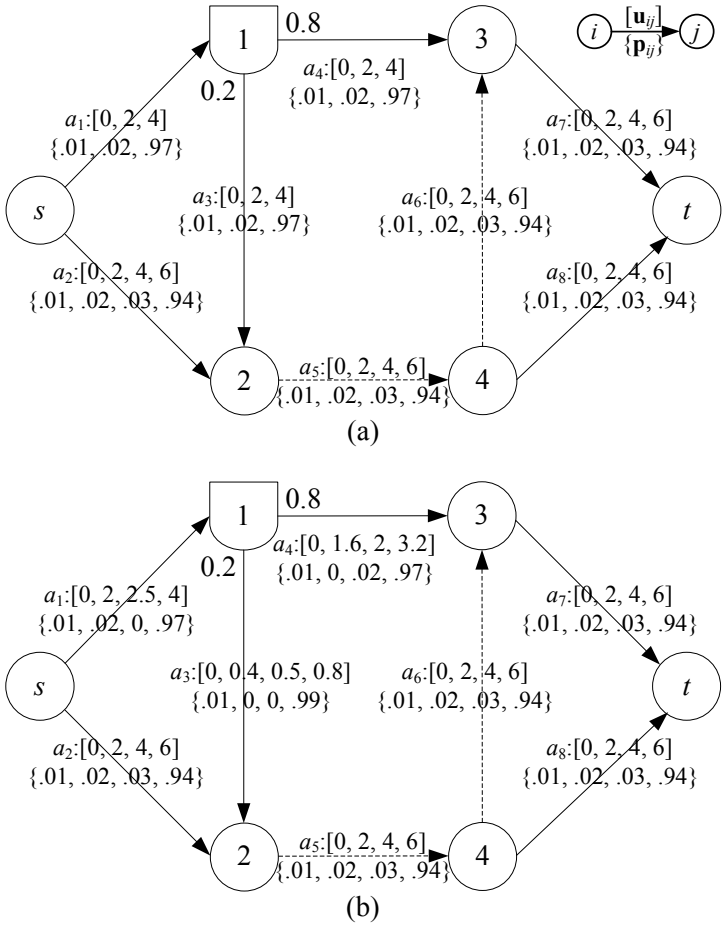


圖 4.1: 簡化後之多狀態分配網路

表 4.1: 窮舉所有可能之弧容量狀態組合列表

a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8
0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	2
0	0	0	0	2	0	0	4
0	0	0	0	3	0	0	6
0	0	0	1	0	0	2	0
0	0	0	1	1	0	2	2
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
4	6	0.8	3.2	6	6	6	6

態組合向量$\overline{CCC}$尚有120組(如表4.2)，再進行Step 3以找出所有的下界點容量狀態向量(如表4.3)，最後得到五個狀態向量$QCC_1 = [2, 6, 0.4, 1.6, 6, 0, 2, 6]$，$QCC_2 = [2, 6, 0.4, 1.6, 6, 2, 4, 4]$，$QCC_3 = [2, 6, 0.4, 1.6, 6, 4, 6, 2]$，$QCC_4 = [4, 4,$

0.8, 3.2, 4, 0, 4, 4] 與 $QCC_5 = [4, 4, 0.8, 3.2, 4, 2, 6, 2]$，而可靠度即可利用表 4.4 的資訊計算而得：$\Pr(E_1 \cup E_2 \cup E_3 \cup E_4 \cup E_5)$=(1) + (2) + (3) + (4) + (5) − (6) − (7) − (8) − (9) − (10) − (11) − (12) − (13) − (14) − (15) + (16) + (17) + (18) + (19) + (20) + (21) + (22) + (23) + (24) + (25) − (26) − (27) − (28) − (29) − (30) + (31) = 0.889952393。

4.2 PART-D(Partitioning with D-nodes) 演算法

Daly and Alexpoulos (2006) 以 Daly (2001) 的演算法為基礎，運用了 Douilliez and Jamoulle (1972) 所提出的 SSP(State Space Partitioning) 法將所有容量狀態向量的組合視為狀態空間，並以上下界分割容量狀態空間方式來求得可靠度。本小節提出的演算法 PART-D(Partitioning with D-nodes) 利用分配節點的特性來加速 Daly and Alexpoulos (2006) 所提出的 F-PARTITION 演算法。我們將依循 Daly and Alexpoulos (2006) 之 F-PARTITION 演算法相關之數學符號定義來闡述 PART-D 之演算法原理，並以數字範例詳細說明 PART-D 演算法的計算過程。

在 PART-D 演算法中，令弧的集合 $A = \{1, \ldots, m\}$，弧 i 上的容量是一組間斷型隨機變數，其值為滿足 $\{b_i(1) < \cdots < b_i(\eta_i)\}$ 之有限集合 $\{b_i(1), \cdots, b_i(\eta_i)\}$。系統的狀態可表示成隨機向量 $X = (X_1, \ldots, X_m)$，狀態空間中的元素則表示成 $x(v) = (b_1(v_1), b_2(v_2), \ldots, b_m(v_m))$，其中 $v_i \in \{1, \ldots, \eta_i\}$。為了簡化符號，令 v_i 代表 $b_i(v_i)$，則 $x(v)$ 可直接表示成 $v = (v_1, \ldots, v_m)$。因此，系統狀態空間 Ω 即可表示成 $\Omega = \otimes_{i=1}^{a}\{1, \ldots, \eta_i\} = \{1, \ldots, \eta_1\} \times \{1, \ldots, \eta_2\} \times \cdots \times \{1, \ldots, \eta_m\}$。令 α 與 β 是 Ω 中的兩個元素，且假設對於所有的 $i \in A$ 而言，$\alpha \leq \beta$ 即為 $\alpha_i \leq \beta_i$，則集合 $R = \{v \in \Omega : \alpha_i \leq v_i \leq \beta_i\}$, $\forall i \in A$ 可視為多維度空間中矩型的間斷區間，簡單表示成 $R = [\alpha, \beta]$，其中 $\alpha[R]$ 與 $\beta[R]$ 表示區間 R 的上下界。若對於 $j \in \{1, \ldots, \eta_i\}$ 以及 $i \in A$，令 $p_i(j) = \Pr\{X_i = b_i(j)\}$ 並假設 X 中的元素相互之間獨立，則任何一個系統狀態的機率可表示成 $\Pr\{X = x\} = \Pr\{X_1 = b_1(v_1), \ldots, X_m = b_m(v_m)\} = \prod_{i=1}^{m} p_i(v_i)$，而系統狀態的間斷區間機率即為 $\Pr\{[\alpha, \beta]\} = \prod_{i=1}^{m} \sum_{j=\alpha_i}^{\beta_i} p_i(j)$。

PART-D 演算法之原理即在系統狀態空間 Ω 中藉由空間切割方式反覆找尋可行區域切割點 v^0 (feasible cut point；即切割之後，大於該點的容量狀態向量

皆為可行)與極限可行指標v_j^*(limiting feasible index；即切割後得到的未知區間下界點，小於該點的容量狀態向量皆不可行)，將未知的區間切割成三部分：(1)可行的區間$F = [v^0, \beta]$；(2)不可行的區間$I_j = \{v \in R : v_i^* \le v_i \le \beta_i \text{ for } i < j, \alpha_j \le v_j < v_j^*, \alpha_i \le v_i \le \beta_i \text{ for } i > j\}, j \in A$；以及(3)未確定的區間$U_j = \{v \in R : v_i^0 \le v_i \le \beta_i \text{ for } i < j, v_j^* \le v_j < v_j^0, v_i^* \le v_i \le \beta_i \text{ for } i > j\}, j \in A$。接著從可靠度下界$P_L$往上累加可行區間的機率值$\Pr\{F\}$，再從可靠度上界$P_U$逐一減去不可行區間的機率值$\Pr\{I_j\}$，直到可靠度上下界相等即得到網路可靠度。以下為PART-D演算法之流程：

Algorithm 2 PART-D

0 Compacting the capacities for all arcs adjacent to D-nodes.

1 Set $P_U = 1$ and $P_L = 0$. Set $\Gamma = \{\Omega\}$.

2 While $\Gamma \neq \emptyset$, do:

2.1 Remove the top set R from Γ and determine if the point $\beta[R]$ is infeasible. If it is, set $P_U = P_U - \Pr\{R\}$ and exit.

2.2 Otherwise, determine v^0 and set $F = [v^0, \beta]$.

2.3 Set $P_L = P_L + \Pr\{F\}$.

2.4 For $j \in A$: Find v_j^* and determine the resulting infeasible intervals I_j.

2.5 For $j \in A$: Set $P_U = P_U - \Pr\{I_j\}$.

2.6 Add all nonempty undetermined sets U_j to Γ.

3 Return $P = P_U = P_L$, the reliability of the network.

在原始的F-PARTITION法中，找尋v^0時必須針對每一條弧上的容量狀態測試每降低一級的容量狀態是否仍可行，而測試可行與否則必須將該狀態向量的組合求解一最大流量問題，因此若狀態向量組合越多就越耗費時間。但若是網路圖中有出現分配節點，且該分配節點之母弧與餾化弧之容量狀態被正規化整合之後(即PART-D中的步驟0)，則經過該分配節點之流量可行與否即可由與該分配節點連結之任一弧的容量狀態向量是否可行來判斷即可，因此可減少測試可行的次數。例如，假設在網路圖中有a條餾化弧且其上皆有b個可能的容量狀態，

則F-PARTITION最多要檢查b^{a+1}次才能判斷出可行的容量狀態向量，但若經過整合正規化之後則只需檢查$(a+1)b$次即可(因為此時母弧與餾化弧最多可能有$(a+1)b$種容量狀態)。同樣的，找尋v^*時原本也需要測試每一條弧上的每種弧容量的組合，但若是網路圖中有分配節點，且該分配節點連結弧上的容量狀態皆已被正規化整合，則分配節點必滿足$v^* = v^0$，也就是不需測試分配節點的連結弧，更可減少測試的次數。以下便用一數字實例實際說明此PART-D演算法的流程。

4.2.1 PART-D範例演練

同樣利用圖4.1來計算需求量$d = 3$單位之網路可靠度。接著依循演算法2計算網路可靠度：

1 Set $P_U = 1$ and $P_L = 0$. Set $\Gamma = \{\Omega\}$.

2.1 Use the set $R = \Omega$, $\beta[R] = (4, 6, 0.8, 3.2, 6, 6, 6, 6)$ is feasible.

2.2 $v^0 = (0, 4, 0, 0, 4, 0, 0, 4)$, $F = [(0, 4, 0, 0, 4, 0, 0, 4), (4, 6, 0.8, 3.2, 6, 6, 6, 6)]$.

2.3 $P_L = P_L + \Pr\{F\} = 0 + \Pr\{[(0, 4, 0, 0, 4, 0, 0, 4), (4, 6, 0.8, 3.2, 6, 6, 6, 6)]\} = 0.912673$.

2.4 $(v_1^*, v_2^*, v_3^*, v_4^*, v_5^*, v_6^*, v_7^*, v_8^*) = (0, 0, 0, 0, 2, 0, 0, 0)$, $I_5 = [(0, 0, 0, 0, 0, 0, 0, 0), (4, 6, 0.8, 3.2, 0, 6, 6, 6, 6)]$.

2.5 $P_U = 1 - \Pr\{I_5\} = 1 - \Pr\{[(0, 0, 0, 0, 0, 0, 0, 0), (4, 6, 0.8, 3.2, 0, 6, 6, 6, 6)]\} = 0.99$.

2.6 $U_2 = [(0, 0, 0, 0, 2, 0, 0, 0), (4, 2, 0.8, 3.2, 6, 6, 6, 6)]$, $U_5 = [(0, 4, 0, 0, 2, 0, 0, 0), (4, 6, 0.8, 3.2, 2, 6, 6, 6)]$, $U_8 = [(0, 4, 0, 0, 4, 0, 0, 0), (4, 6, 0.8, 3.2, 6, 6, 6, 2)]$, $\Gamma = \{U_2, U_5, U_8\}$.

2.1 Use the set $U_2 = [(0, 0, 0, 0, 2, 0, 0, 0), (4, 2, 0.8, 3.2, 6, 6, 6, 6)]$, $\beta[U_2] = (4, 2, 0.8, 3.2, 6, 6, 6, 6)$ is feasible.

2.2 $v^0 = (2, 2, 0.4, 1.6, 2, 0, 2, 2)$, $F = [(2, 2, 0.4, 1.6, 2, 0, 2, 2), (4, 2, 0.8, 3.2, 6, 6, 6, 6)]$.

2.3 $P_L = P_L + \Pr\{F\} = 0.912673 + \Pr\{[(2, 2, 0.4, 1.6, 2, 0, 2, 2), (4, 2, 0.8, 3.2, 6, 6, 6, 6)]\} = 0.931502603$.

2.4 $(v_1^*, v_2^*, v_3^*, v_4^*, v_5^*, v_6^*, v_7^*, v_8^*) = (2, 0, 0.4, 1.6, 2, 0, 2, 0)$, $I_1 = [(0, 0, 0, 0, 2, 0, 0, 0), (0, 2, 0.8, 3.2, 6, 6, 6, 6)]$, $I_3 = [(2, 0, 0, 0, 2, 0, 0, 0), (4, 2, 0, 3.2, 6, 6, 6, 6)]$, $I_4 = [(2, 0, 0.4, 0, 2, 0, 0, 0), (4, 2, 0.8, 0, 6, 6, 6, 6)]$, $I_7 = [(2, 0, 0.4, 1.6, 2, 0, 0, 0), (4, 2, 0.8, 3.2, 6, 6, 0, 6)]$.

2.5 $P_U = 0.99 - \Pr\{I_1\} - \Pr\{I_3\} - \Pr\{I_4\} - \Pr\{I_7\} = 0.988829701$.

2.6 $U_2' = [(2, 0, 0.4, 1.6, 2, 0, 2, 0), (4, 0, 0.8, 3.2, 6, 6, 6, 6)]$, $U_8' = [(2, 2, 0.4, 1.6, 2, 0, 2, 0), (4, 2, 0.8, 3.2, 6, 6, 6, 0)]$, $\Gamma = \{U_5, U_8, U_2', U_8'\}$.

$\vdots$

3 $P_U = P_L = 0.986915609 = P$.

此結果代表可在網路圖4.1上成功運送至少3單位需求量的可靠度為0.986915609。

4.3 小結

本章提出MSE與PART-D兩組演算法以在具有分配節點的網路圖中求算網路可靠度。此兩種演算法與傳統的網路可靠度計算方式的主要不同之處在於它們皆利用弧容量是否能承受流量經過的觀點來找尋可行的容量狀態向量，因此避開了流量有無限多種實數組合的計算困難。此外，第三章所提及的分配網路簡化與整合之程序的確可大幅減少本章兩種可靠度演算法的檢驗步驟。

表 4.2: 剩下的 $\overline{CCC}$ 列表

$\overline{CCC}$	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8
1	2	6	0.4	1.6	6	0	2	6
2	2	6	0.4	1.6	6	0	4	6
3	2	6	0.4	1.6	6	0	6	6
4	2	6	0.4	1.6	6	2	2	6
5	2	6	0.4	1.6	6	2	4	4
6	2	6	0.4	1.6	6	2	4	6
7	2	6	0.4	1.6	6	2	6	4
8	2	6	0.4	1.6	6	2	6	6
9	2	6	0.4	1.6	6	4	2	6
10	2	6	0.4	1.6	6	4	4	4
11	2	6	0.4	1.6	6	4	4	6
12	2	6	0.4	1.6	6	4	6	2
13	2	6	0.4	1.6	6	4	6	4
14	2	6	0.4	1.6	6	4	6	6
15	2	6	0.4	1.6	6	6	2	6
16	2	6	0.4	1.6	6	6	4	4
17	2	6	0.4	1.6	6	6	4	6
18	2	6	0.4	1.6	6	6	6	2
19	2	6	0.4	1.6	6	6	6	4
20	2	6	0.4	1.6	6	6	6	6
21	2.5	6	0.5	2	6	0	2	6
22	2.5	6	0.5	2	6	0	4	6
23	2.5	6	0.5	2	6	0	6	6
24	2.5	6	0.5	2	6	2	2	6
25	2.5	6	0.5	2	6	2	4	4
26	2.5	6	0.5	2	6	2	4	6
27	2.5	6	0.5	2	6	2	6	4
28	2.5	6	0.5	2	6	2	6	6
29	2.5	6	0.5	2	6	4	2	6
30	2.5	6	0.5	2	6	4	4	4
31	2.5	6	0.5	2	6	4	4	6
32	2.5	6	0.5	2	6	4	6	2
33	2.5	6	0.5	2	6	4	6	4
34	2.5	6	0.5	2	6	4	6	6
35	2.5	6	0.5	2	6	6	2	6
36	2.5	6	0.5	2	6	6	4	4
37	2.5	6	0.5	2	6	6	4	6
38	2.5	6	0.5	2	6	6	6	2
39	2.5	6	0.5	2	6	6	6	4
40	2.5	6	0.5	2	6	6	6	6
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
119	4	6	0.8	3.2	6	6	6	4
120	4	6	0.8	3.2	6	6	6	6

表 4.3: 兩狀態向量相互比較之過程

i	j	Qualification	I
1	2	$CCC_2 > CCC_1$	{2}
1	3	$CCC_3 > CCC_1$	{2, 3}
⋮	⋮	⋮	⋮
1	117	$CCC_{117} > CCC_1$	{2, 3, 4, 6, 8, 9, . . . , 117}
1	120	$CCC_{120} > CCC_1$	{2, 3, . . . , 117, 120}
5	7	$CCC_7 > CCC_5$	{2, 3, . . . , 117, 120; 7}
5	10	$CCC_{10} > CCC_5$	{2, 3, . . . , 117, 120, 7, 10}
⋮	⋮	⋮	⋮
5	119	$CCC_{119} > CCC_5$	{2, 3, 4, 6, 8, 9, . . . , 117, 120; 7, . . . , 119}
12	18	$CCC_{18} > CCC_{12}$	{2, 3, 4, 6, 8, 9, . . . , 117, 120; 7, . . . , 119; 18}
12	32	$CCC_{32} > CCC_{12}$	{2, 3, 4, 6, 8, 9, . . . , 117, 120; 7, . . . , 119; 18, 32}
⋮	⋮	⋮	⋮
12	118	$CCC_{118} > CCC_{12}$	{2, 3, 4, 6, 8, 9, . . . , 117, 120; 7, . . . , 119; 18, . . . , 118}
41	42	$CCC_{42} > CCC_{41}$	{2, 3, 4, 6, 8, 9, . . . , 117, 120; 7, . . . , 119; 18, . . . , 118; 42}
41	43	$CCC_{43} > CCC_{41}$	{2, 3, 4, 6, 8, 9, . . . , 117, 120; 7, . . . , 119; 18, . . . , 118; 42, 43}
⋮	⋮	⋮	⋮
41	101	$CCC_{101} > CCC_{41}$	{2, 3, 4, 6, 8, 9, . . . , 117, 120; 7, . . . , 119; 18, . . . , 118; 42, . . . , 101}
47	52	$CCC_{52} > CCC_{47}$	{2, 3, 4, 6, 8, 9, . . . , 117, 120; 7, . . . , 119; 18, . . . , 118; 42, . . . , 101; 52}
47	57	$CCC_{57} > CCC_{47}$	{2, 3, 4, 6, 8, 9, . . . , 117, 120; 7, . . . , 119; 18, . . . , 118; 42, . . . , 101; 52, 57}
⋮	⋮	⋮	⋮
47	106	$CCC_{106} > CCC_{47}$	{2, 3, 4, 6, 8, 9, . . . , 117, 120; 7, . . . , 119; 18, . . . , 118; 42, . . . , 101; 52, . . . , 106}

表 4.4: 計算可靠度所需之事件機率列表

(1)	$\Pr(E_1)$	$= \Pr(X \mid X \geq (2, 6, 0.4, 1.6, 6, 0, 2, 6))$	$= 0.797855676$
(2)	$\Pr(E_2)$	$= \Pr(X \mid X \geq (2, 6, 0.4, 1.6, 6, 2, 4, 4))$	$= 0.798619581$
(3)	$\Pr(E_3)$	$= \Pr(X \mid X \geq (2, 6, 0.4, 1.6, 6, 4, 6, 2))$	$= 0.773920006$
(4)	$\Pr(E_4)$	$= \Pr(X \mid X \geq (4, 4, 0.8, 3.2, 4, 0, 4, 4))$	$= 0.824642285$
(5)	$\Pr(E_5)$	$= \Pr(X \mid X \geq (4, 4, 0.8, 3.2, 4, 2, 6, 2))$	$= 0.807458805$
(6)	$\Pr(E_1 \cap E_2)$	$= \Pr(X \mid X \geq (2, 6, 0.4, 1.6, 6, 2, 4, 6))$	$= 0.773920006$
(7)	$\Pr(E_1 \cap E_3)$	$= \Pr(X \mid X \geq (2, 6, 0.4, 1.6, 6, 4, 6, 6))$	$= 0.734833137$
(8)	$\Pr(E_1 \cap E_4)$	$= \Pr(X \mid X \geq (4, 6, 0.8, 3.2, 6, 0, 4, 6))$	$= 0.750471075$
(9)	$\Pr(E_1 \cap E_5)$	$= \Pr(X \mid X \geq (4, 6, 0.8, 3.2, 6, 2, 6, 6))$	$= 0.719988023$
(10)	$\Pr(E_2 \cap E_3)$	$= \Pr(X \mid X \geq (2, 6, 0.4, 1.6, 6, 4, 6, 4))$	$= 0.758285258$
(11)	$\Pr(E_2 \cap E_4)$	$= \Pr(X \mid X \geq (4, 6, 0.8, 3.2, 6, 2, 4, 4))$	$= 0.766678057$
(12)	$\Pr(E_2 \cap E_5)$	$= \Pr(X \mid X \geq (4, 6, 0.8, 3.2, 6, 2, 6, 4))$	$= 0.742966364$
(13)	$\Pr(E_3 \cap E_4)$	$= \Pr(X \mid X \geq (4, 6, 0.8, 3.2, 6, 4, 6, 4))$	$= 0.727956943$
(14)	$\Pr(E_3 \cap E_5)$	$= \Pr(X \mid X \geq (4, 6, 0.8, 3.2, 6, 4, 6, 2))$	$= 0.742966364$
(15)	$\Pr(E_4 \cap E_5)$	$= \Pr(X \mid X \geq (4, 4, 0.8, 3.2, 4, 2, 6, 4))$	$= 0.791146505$
(16)	$\Pr(E_1 \cap E_2 \cap E_3)$	$= \Pr(X \mid X \geq (2, 6, 0.4, 1.6, 6, 4, 6, 6))$	$= 0.734833137$
(17)	$\Pr(E_1 \cap E_2 \cap E_4)$	$= \Pr(X \mid X \geq (4, 6, 0.8, 3.2, 6, 2, 4, 6))$	$= 0.742966364$
(18)	$\Pr(E_1 \cap E_2 \cap E_5)$	$= \Pr(X \mid X \geq (4, 6, 0.8, 3.2, 6, 2, 6, 6))$	$= 0.719988023$
(19)	$\Pr(E_1 \cap E_3 \cap E_4)$	$= \Pr(X \mid X \geq (4, 6, 0.8, 3.2, 6, 4, 6, 6))$	$= 0.705442811$
(20)	$\Pr(E_1 \cap E_3 \cap E_5)$	$= \Pr(X \mid X \geq (4, 6, 0.8, 3.2, 6, 4, 6, 6))$	$= 0.705442811$
(21)	$\Pr(E_1 \cap E_4 \cap E_5)$	$= \Pr(X \mid X \geq (4, 6, 0.8, 3.2, 6, 2, 6, 6))$	$= 0.719988023$
(22)	$\Pr(E_2 \cap E_3 \cap E_4)$	$= \Pr(X \mid X \geq (4, 6, 0.8, 3.2, 6, 4, 6, 4))$	$= 0.727956943$
(23)	$\Pr(E_2 \cap E_3 \cap E_5)$	$= \Pr(X \mid X \geq (4, 6, 0.8, 3.2, 6, 4, 6, 4))$	$= 0.727956943$
(24)	$\Pr(E_2 \cap E_4 \cap E_5)$	$= \Pr(X \mid X \geq (4, 6, 0.8, 3.2, 6, 2, 6, 4))$	$= 0.742966364$
(25)	$\Pr(E_3 \cap E_4 \cap E_5)$	$= \Pr(X \mid X \geq (4, 6, 0.8, 3.2, 6, 4, 6, 4))$	$= 0.727956943$
(26)	$\Pr(E_1 \cap E_2 \cap E_3 \cap E_4)$	$= \Pr(X \mid X \geq (4, 6, 0.8, 3.2, 6, 4, 6, 6))$	$= 0.705442811$
(27)	$\Pr(E_1 \cap E_2 \cap E_3 \cap E_5)$	$= \Pr(X \mid X \geq (4, 6, 0.8, 3.2, 6, 4, 6, 6))$	$= 0.705442811$
(28)	$\Pr(E_1 \cap E_2 \cap E_4 \cap E_5)$	$= \Pr(X \mid X \geq (4, 6, 0.8, 3.2, 6, 2, 6, 6))$	$= 0.719988023$
(29)	$\Pr(E_1 \cap E_3 \cap E_4 \cap E_5)$	$= \Pr(X \mid X \geq (4, 6, 0.8, 3.2, 6, 4, 6, 6))$	$= 0.705442811$
(30)	$\Pr(E_2 \cap E_3 \cap E_4 \cap E_5)$	$= \Pr(X \mid X \geq (4, 6, 0.8, 3.2, 6, 4, 6, 4))$	$= 0.727956943$
(31)	$\Pr(E_1 \cap E_2 \cap E_3 \cap E_4 \cap E_5)$	$= \Pr(X \mid X \geq (4, 6, 0.8, 3.2, 6, 4, 6, 6))$	$= 0.705442811$

第五章

分配網路之其它可靠度相關議題

除了前一章節所探討的最大容量網路圖之外，若能加入成本屬性則更可貼近現實問題。過去曾有學者將每提供一單位的隨機弧容量之成本列入考慮，因此才會有如何在具成本預算的限制之下求算系統的網路可靠度的研究議題。在設計網路時，其每一條弧都需付出建構成本，要如何設計網路才能以最小成本使其可靠度滿足需求門檻即為本章探討的第一個研究議題。

另外，也曾有學者探討網路圖中弧長度具有隨機多狀態特性之網路可靠度議題。其中 Alexopoulos (1997) 提出了利用狀態空間分割 SSP 的方法來計算網路圖中最短路徑小於某一個數值的機率。此時可將此機率值視為一項績效指標，以提供管理者在不確定的環境中參考的資訊。本章第二部分將試圖把 SSP 的方法運用在具有分配節點的網路圖以計算其可靠度。

5.1 滿足可靠度門檻之最小成本分配網路設計問題

在過去的文獻中，除了探討網路可靠度之外，也有學者研究加入了預算限制的議題，也就是每提供一單位的弧容量就會有固定花費，在滿足需求量並且不得超支預算的前提下，找尋系統的網路可靠度。若以另一角度來看，我們亦可將系統的可靠度當作限制式以找尋滿足系統可靠度之最小成本，也就是用最低的花費使系統滿足需求量的機率到達一定程度，此時成本反而成為著重的目標。因此本小節試圖將最小成本的概念運用在多容量狀態的分配網路設計問題上。

假設在系統建置之前，能夠事先預測可允許設置的網路架構，也就能夠估算整個網路的可靠度。但若是能夠只建置系統中的部分元件而能達到預設的可靠

度門檻，即可降低建構網路的成本。因此，本節即在探討如何建置部分系統而使得網路可靠度得以滿足門檻且成本最小。

令I_{a_j}表示弧a_j設置與否的指標變數，即$I_{a_j}=1$(或$I_{a_j}=0$)表示在此網路中要(或不要)設置弧a_j。則系統建置的指標變數$\mathbf{I}^l(X)=(I^l_{a_1}, I^l_{a_2}, \ldots, I^l_{a_m})$就有$2^m$種組合，其中真正能使得系統連通且為最簡連結狀況的組合為$\overline{\mathbf{I}}^l$，則每一最簡的系統建置狀況$\overline{\mathbf{I}}^l$以及其可能的聯集狀況$\cup_l\overline{\mathbf{I}}^l$可被用來計算網路可靠度$Rel(\cup_l\overline{\mathbf{I}}^l)$。若設置弧$a_j$的成本為$c_{a_j}$，即可計算出建置部分系統$\cup_l\overline{\mathbf{I}}^l$的成本為$\sum_{j=1}^{m} c_{a_j}I^{\cup_l}_{a_j}$，因此在滿足系統需求的機率超過一定門檻$\alpha$的前提之下找尋最小網路建構成本的模式可列出如下：

$$\min \sum_{j=1}^{m} c_{a_j}I^{\cup_l}_{a_j}$$
$$s.t.\ Rel(\cup_l\overline{\mathbf{I}}) \geq \alpha$$

其演算過程為：

Step 1. 列舉所有系統建置的指標變數$\mathbf{I}^l=(I^l_{a_1}, I^l_{a_2}, \ldots, I^l_{a_m})$。

Step 2. 檢視$\mathbf{I}^l$是否能連結起訖點，刪去無法連通起訖點的$\mathbf{I}^l$。

Step 3. 利用4.1小節找尋QCC的方式，兩兩比較找尋最簡的連結狀況$\overline{\mathbf{I}}^l$，令所有的最簡連結狀況所構成的集合為L。

Step 4. 列舉出所有最簡系統連結狀況的組合$\cup_{\text{some } l\in L}\overline{\mathbf{I}}^l$，計算其可靠度$Rel(\cup_{\text{some } l\in L}\overline{\mathbf{I}}^l)$並檢測可靠度是否超過門檻$\alpha$，若是，則可行；反之則不可行。

Step 5. 計算所有可行的$\cup_{\text{some } l\in L}\overline{\mathbf{I}}^l$(即$\{\cup_{\text{some } l\in L}\overline{\mathbf{I}}^l | Rel(\cup_{\text{some } l\in L}\overline{\mathbf{I}}^l(X)) \geq \alpha\}$)的成本$\sum_{j=1}^{m} c_{a_j}I^{\cup_l}_{a_j}$，而其中最小者即為滿足系統可靠度的最小網路建置成本，其所對應到的$\cup_{\text{some } l\in L}\overline{\mathbf{I}}^l$即為滿足系統可靠度的最小成本網路圖。

假設在網路圖4.1中建置一條弧需花費1單位的成本，即$c_{a_j}=1$，則找尋可靠度超過門檻α且成本最小的網路建構方式便可依循上述的步驟說明如下：

Step 1. $\mathbf{I}^1 = (I^1_{a_1}, I^1_{a_2}, \ldots, I^1_{a_8}) = (0, 0, 0, 0, 0, 0, 0, 0)$, $\mathbf{I}^2 = (0, 0, 0, 0, 0, 0, 0, 1)$, $\ldots$, $\mathbf{I}^{256} = (1, 1, 1, 1, 1, 1, 1, 1)$。

Step 2. 剩下可連通的指標變數為$(1, 0, 1, 1, 0, 0, 0, 1)$, $(1, 0, 1, 1, 0, 0, 1, 0)$, $(1, 0, 1, 1, 0, 0, 1, 1)$, $(1, 0, 1, 1, 1, 0, 0, 1)$, $(1, 0, 1, 1, 1, 0, 1, 0)$, $(1, 0, 1, 1, 1, 0, 1, 1)$, $(1, 0, 1, 1, 0, 1, 0, 1)$, $(1, 0, 1, 1, 0, 1, 1, 0)$, $(1, 0, 1, 1, 0, 1, 1, 1)$, $(1, 0, 1, 1, 1, 1, 0, 1)$, $(1, 0, 1, 1, 1, 1, 1, 0)$, $(1, 0, 1, 1, 1, 1, 1, 1)$, $(1, 1, 1, 1, 0, 0, 0, 1)$, $(1, 1, 1, 1, 0, 0, 1, 0)$, $(1, 1, 1, 1, 0, 0, 1, 1)$, $(1, 1, 1, 1, 1, 0, 0, 1)$, $(1, 1, 1, 1, 1, 0, 1, 0)$, $(1, 1, 1, 1, 1, 0, 1, 1)$, $(1, 1, 1, 1, 0, 1, 0, 1)$, $(1, 1, 1, 1, 0, 1, 1, 0)$, $(1, 1, 1, 1, 0, 1, 1, 1)$, $(1, 1, 1, 1, 1, 1, 0, 1)$, $(1, 1, 1, 1, 1, 1, 1, 0)$, $(1, 1, 1, 1, 1, 1, 1, 1)$, $(0, 1, 0, 0, 0, 0, 0, 1)$, $(0, 1, 0, 0, 0, 0, 1, 0)$, $(0, 1, 0, 0, 0, 0, 1, 1)$, $(0, 1, 0, 0, 1, 0, 0, 1)$, $(0, 1, 0, 0, 1, 0, 1, 0)$, $(0, 1, 0, 0, 1, 0, 1, 1)$, $(0, 1, 0, 0, 0, 1, 0, 1)$, $(0, 1, 0, 0, 0, 1, 1, 0)$, $(0, 1, 0, 0, 0, 1, 1, 1)$, $(0, 1, 0, 0, 1, 1, 0, 1)$, $(0, 1, 0, 0, 1, 1, 1, 0)$, $(0, 1, 0, 0, 1, 1, 1, 1)$。

Step 3. 兩兩比較過後最簡的連結方式為$\bar{\mathbf{I}}^1 = (0, 1, 0, 0, 1, 0, 0, 1)$, $\bar{\mathbf{I}}^2 = (0, 1, 0, 0, 1, 1, 1, 0)$, $\bar{\mathbf{I}}^3 = (1, 0, 1, 1, 1, 0, 0, 1)$, $\bar{\mathbf{I}}^4 = (1, 0, 1, 1, 0, 0, 1, 0)$。

Step 4. 除了$\bar{\mathbf{I}}^1$, $\bar{\mathbf{I}}^2$, $\bar{\mathbf{I}}^3$, $\bar{\mathbf{I}}^4$之外，可能的組合還包括$\cup_{l=1,2}\bar{\mathbf{I}}^l$, $\cup_{l=1,3}\bar{\mathbf{I}}^l$, $\cup_{l=1,4}\bar{\mathbf{I}}^l$, $\cup_{l=2,3}\bar{\mathbf{I}}^l$, $\cup_{l=2,4}\bar{\mathbf{I}}^l$, $\cup_{l=3,4}\bar{\mathbf{I}}^l$, $\cup_{l=1,2,3}\bar{\mathbf{I}}^l$, $\cup_{l=1,2,4}\bar{\mathbf{I}}^l$, $\cup_{l=1,3,4}\bar{\mathbf{I}}^l$, $\cup_{l=2,3,4}\bar{\mathbf{I}}^l$, $\cup_{l=1,2,3,4}\bar{\mathbf{I}}^l$，其中$\cup_{l=1,2}\bar{\mathbf{I}}^l = (0, 1, 0, 0, 1, 1, 1, 1)$, $\cup_{l=1,3}\bar{\mathbf{I}}^l = (1, 1, 1, 1, 1, 0, 0, 1)$, $\cup_{l=1,4}\bar{\mathbf{I}}^l = \cup_{l=1,3,4}\bar{\mathbf{I}}^l = (1, 1, 1, 1, 1, 0, 1, 1)$, $\cup_{l=2,4}\bar{\mathbf{I}}^l = (1, 1, 1, 1, 1, 1, 1, 0)$, $\cup_{l=3,4}\bar{\mathbf{I}}^l = (1, 0, 1, 1, 1, 0, 1, 1)$, $\cup_{l=2,3}\bar{\mathbf{I}}^l = \cup_{l=1,2,3}\bar{\mathbf{I}}^l = \cup_{l=1,2,4}\bar{\mathbf{I}}^l = \cup_{l=2,3,4}\bar{\mathbf{I}}^l = \cup_{l=1,2,3,4}\bar{\mathbf{I}}^l = (1, 1, 1, 1, 1, 1, 1, 1)$。除了$Rel(\bar{\mathbf{I}}^3)$不可行之外其他可行狀態之可靠度分別為$Rel(\bar{\mathbf{I}}^1) = 0.912673$, $Rel(\bar{\mathbf{I}}^2) = 0.8852981$, $Rel(\bar{\mathbf{I}}^4) = 0.90354627$, $Rel(\cup_{l=1,2}\bar{\mathbf{I}}^l) = 0.93996945$, $Rel(\cup_{l=1,3}\bar{\mathbf{I}}^l) = 0.912673$, $Rel(\cup_{l=1,4}\bar{\mathbf{I}}^l) = Rel(\cup_{l=1,3,4}\bar{\mathbf{I}}^l) = 0.976883963$, $Rel(\cup_{l=2,4}\bar{\mathbf{I}}^l) = 0.948385484$, $Rel(\cup_{l=3,4}\bar{\mathbf{I}}^l) = 0.885565699$, $Rel(\cup_{l=2,3}\bar{\mathbf{I}}^l) = Rel(\cup_{l=1,2,3}\bar{\mathbf{I}}^l) = Rel(\cup_{l=1,2,4}\bar{\mathbf{I}}^l) = Rel(\cup_{l=2,3,4}\bar{\mathbf{I}}^l) = Rel(\cup_{l=1,2,3,4}\bar{\mathbf{I}}^l) = 0.986915609$。

Step 5. 若設定$\alpha = 0.9$，則可靠度超過門檻值的組合為$\overline{\mathbf{I}}^1$, $\overline{\mathbf{I}}^4$, $\cup_{l=1,2}\overline{\mathbf{I}}^l$, $\cup_{l=1,3}\overline{\mathbf{I}}^l$, $\cup_{l=1,4}\overline{\mathbf{I}}^l$, $\cup_{l=1,3,4}\overline{\mathbf{I}}^l$, $\cup_{l=2,4}\overline{\mathbf{I}}^l$, $\cup_{l=2,3}\overline{\mathbf{I}}^l$, $\cup_{l=1,2,3}\overline{\mathbf{I}}^l$, $\cup_{l=1,2,4}\overline{\mathbf{I}}^l$, $\cup_{l=2,3,4}\overline{\mathbf{I}}^l$, $\cup_{l=1,2,3,4}\overline{\mathbf{I}}^l$，其中最小成本者為$\sum_{j=1}^{8} c_{a_j} I_{a_j}^1 = 3$；若設定$\alpha = 0.95$，則可靠度超過門檻值的組合只有$\cup_{l=1,4}\overline{\mathbf{I}}^l$, $\cup_{l=1,3,4}\overline{\mathbf{I}}^l$與$\cup_{l=2,3}\overline{\mathbf{I}}^l$, $\cup_{l=1,2,3}\overline{\mathbf{I}}^l$, $\cup_{l=1,2,4}\overline{\mathbf{I}}^l$, $\cup_{l=2,3,4}\overline{\mathbf{I}}^l$, $\cup_{l=1,2,3,4}\overline{\mathbf{I}}^l$，而其中最小的成本為$\sum_{j=1}^{8} c_{a_j} I_{a_j}^{\cup_{1,4}} = \sum_{j=1}^{8} c_{a_j} I_{a_j}^{\cup_{1,3,4}} = 7$。

詳細資訊如表5.1所示。

表5.1: 5.1節範例之結果

Indicator Variables	Reliability	Feasibility ($\alpha = 0.9$)	Feasibility ($\alpha = 0.95$)	Cost
$\overline{\mathbf{I}}^1 = (0, 1, 0, 0, 1, 0, 0, 1)$	0.912673	Feasible	Infeasible	3
$\overline{\mathbf{I}}^2 = (0, 1, 0, 0, 1, 1, 1, 0)$	0.8852981	Infeasible	Infeasible	4
$\overline{\mathbf{I}}^3 = (1, 0, 1, 1, 1, 0, 0, 1)$	Infeasible	-	-	-
$\overline{\mathbf{I}}^4 = (1, 0, 1, 1, 0, 0, 1, 0)$	0.90354627	Feasible	Infeasible	4
$\cup_{l=1,2}\overline{\mathbf{I}}^l = (0, 1, 0, 0, 1, 1, 1, 1)$	0.93996945	Feasible	Infeasible	5
$\cup_{l=1,3}\overline{\mathbf{I}}^l = (1, 1, 1, 1, 1, 0, 0, 1)$	0.912673	Feasible	Infeasible	6
$\cup_{l=1,4}\overline{\mathbf{I}}^l = \cup_{l=1,3,4}\overline{\mathbf{I}}^l$ $= (1, 1, 1, 1, 1, 0, 1, 1)$	0.976883963	Feasible	Feasible	7
$\cup_{l=2,4}\overline{\mathbf{I}}^l = (1, 1, 1, 1, 1, 1, 1, 0)$	0.948385484	Feasible	Infeasible	7
$\cup_{l=3,4}\overline{\mathbf{I}}^l = (1, 0, 1, 1, 1, 0, 1, 1)$	0.885565699	Infeasible	Infeasible	6
$\cup_{l=2,3}\overline{\mathbf{I}}^l = \cup_{l=1,2,3}\overline{\mathbf{I}}^l$ $= \cup_{l=1,2,4}\overline{\mathbf{I}}^l = \cup_{l=2,3,4}\overline{\mathbf{I}}^l$ $= \cup_{l=1,2,3,4}\overline{\mathbf{I}}^l = (1, 1, 1, 1, 1, 1, 1, 1)$	0.986915609	Feasible	Feasible	8

5.2 具有分配節點的隨機最短路徑問題

在具有分配節點的隨機分配網路圖中，路徑經過分配節點時會依照其特定比例走向分配節點的下一條弧，也就是只會選定分配節點的其中一條離開弧。因此若已先知道經過分配節點時會經過的離開弧，則呈現出來的便是一般的網路圖而不再具有分配節點，也就是將具有分配節點的網路圖分割成數個不具有分配節點的網路圖。如圖5.1(a)所示，節點1為分配節點，若已知經過節點1時下一條行經的弧是弧a_3，則形成圖5.1(b)的網路圖；若已知經過節點2時下一條行經的弧是弧a_4，則形成圖5.1(c)的網路圖。

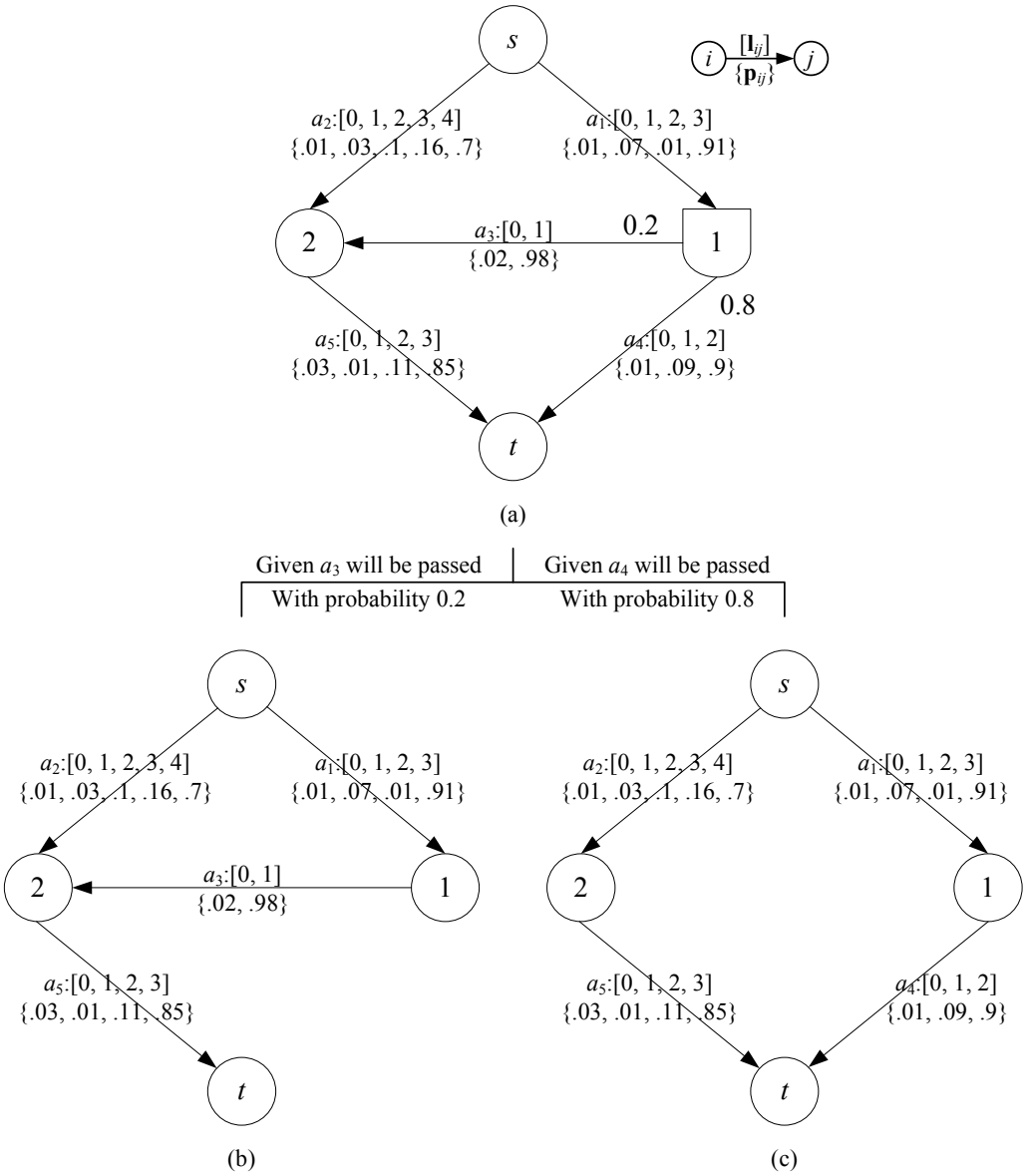

s
i
[l_ij]
{p_ij}
j
a2:[0, 1, 2, 3, 4]
{.01, .03, .1, .16, .7}
a1:[0, 1, 2, 3]
{.01, .07, .01, .91}
2
a3:[0, 1]
{.02, .98}
0.2
1
0.8
a5:[0, 1, 2, 3]
{.03, .01, .11, .85}
a4:[0, 1, 2]
{.01, .09, .9}
t
(a)
Given a3 will be passed
With probability 0.2
Given a4 will be passed
With probability 0.8
(b)
(c)

圖 5.1: 拆解網路圖

將網路圖拆解成一般不具有分配節點的網路圖之後，便可利用下一節所描述的SSP演算法分別計算一般隨機網路圖最短路徑小於某一定值r的機率。而原本具有分配節點的隨機網路圖上最短路徑小於某一定值r的機率可藉由將所有拆解後的網路圖得到的機率再乘上拆解的機率後加總而得。因此若$G=(N,A)$具有分配節點$j' \in N_D$且有$\deg(j')$條離開弧，則網路圖G可拆解成$\prod_{j' \in N_D} \deg(j')$個子網路圖$\bar{G}_i$。對於其中某個子網路圖$\bar{G}_i$而言，形成該子網路圖的機率為$\bar{p}_i$，且在$\bar{G}_i$上最短路徑小於$r$的機率值為$F_i(r)$，則原網路圖$G$的最短路徑小於$r$之機率值即為$F(r)=\sum_{i=1}^{\prod^{j'} \deg(j')} \bar{p}_i F_i(r)$。

5.2.1 狀態空間分割法(SSP)

在Alexopoulos (1997)所提出的方法中，令隨機網路圖中每一條弧$a_i \in A$具有非負的隨機長度X_i且符合一個間斷之機率分配$x_i(1) < x_i(2) < \cdots < x_i(\eta_i)$，其對應的機率分別為$p_i(1), p_i(2), \ldots, p_i(\eta_i)$。隨機弧所組成的狀態空間$\Omega$即為向量$X=(X_1, X_2, \ldots, X_m)$。指標$\alpha_i$, β_i, $w_i \in \{1, 2, \ldots, \eta_i\}$且定義$\Omega$的子集合$R$具有狀態的上界$\alpha=\alpha[R]$與下界$\beta=\beta[R]$且所有的狀態$w$符合$\alpha_i \le w_i \le \beta_i$。因此在子集合區間內的機率即可表示成$\Pr(R)=Pr(X \in R)=\prod_{i=1}^{m} \sum_{k=\alpha_i}^{\beta_i} p_i(k)$，則機率小於某特定值可表示成$F(r)$。

SSP的概念便是在區間$R=(\alpha,\beta)$中，反覆的找尋γ以獲得(1)可行的區間$D=\{w \in R : \alpha_i \le w_i \le \gamma_i \ for\ i \in S\}$與(2)還未確定的區間集合$B_i=\{w \in R : \alpha_k \le w_k \le \gamma_k \ for\ k<i, k \in S; \alpha_k \le w_k \le \beta_k \ for\ k<i, k \notin S; \gamma_i+1 \le w_i \le \beta_i; \alpha_k \le w_k \le \beta_k \ for\ k>i\}, i \in S$ with $\gamma_i < \beta_i$，其中S為A的子集合。為了使未確定的區間B_i愈少，決定γ的方式便是求得以下的背包問題$P_{Knapsack}$的最佳解：

$$\begin{aligned} &\max \ |\{i \in S : w_i = \beta_i\}| && (P_{Knapsack}) \\ &s.t. \quad \sum_{i \in S} x_i(w_i) \le b \\ &\qquad w_i \in \{\alpha_i, \alpha_i+1, \ldots, \beta_i\},\ i \in S \end{aligned}$$

因此分割區間$R=(\alpha,\beta)$的流程即為：先利用下界點α找到最短路徑P與最短路徑長$d(t)$，若$d(t) > r$則此區間不可行，若$d(t) \le r$則令$S=P$與$b=r$並求

解$P_{Knapsack}$問題以獲得可行的區間D與未確定的區間B_i。其中B_i的下界點除了第i項元素為γ_i+1之外，其他皆與原本未確定的區間R相同，則可利用$\alpha[B_i]$求得新的最短路徑P與$d(t)$。同樣當$d(t)>r$時區間B_i為不可行的區間，當$d(t)\le r$時便把區間B_i與最短路徑P和其長度$d(t)$記錄下來。如此一來便可利用SSP的方法來計算最短路徑小於某一定值r的機率$F(r)$。以下為計算$F(r)$的演算法$PARTITION$：

Algorithm 3 PARTITION(r) (Source: Alexopoulos, 1997)

1. Start with the interval $R=\Omega$, empty list L, $F_l(r)=0$, and $F_u(r)=1$. Compute an s-rooted shortest path tree with arc length $x_i(1)$. Let $\ell(j)$ be the predecessor of node j on the tree and let $d(j)$ be the shortest (s,j) path length. If $d(t)>r$, terminate with $F(r)=0$.

2. Identify a shortest path P

3. Do the following:
   Solve problem $P_{Knapsack}$ and compute the feasible interval $D=\{w\in R:\alpha_i\le w_i\le\gamma_i\ for\ i\in S\})$ and the undetermined intervals $B_i=\{w\in R:\alpha_k\le w_k\le\gamma_k\ for\ k<i,\ k\in S;\ \alpha_k\le w_k\le\beta_k\ for\ k<i,\ k\notin S;\ \gamma_i+1\le w_i\le\beta_i;\ \alpha_k\le w_k\le\beta_k\ for\ k>i\}$, $i\in S$ with $\gamma_i<\beta_i$.
   Set $F_l(r)=F_l(r)+\Pr(D)$.
   For $i\in P$ with $\gamma_i<\beta_i$:
   Increase the length of arc i to $x_i(\gamma_i+1)$ and compute a shortest path tree with labels $\bar{\ell}(j)$ and shortest (s,j) path lengths $\bar{d}(j)$.
   If $\bar{d}(t)\le r$, file the record $\{\alpha[B_i];\beta[B_i];(\bar{\ell}(j),j\in N)\}$ in L.
   If $\bar{d}(t)>r$, the set B_i is infeasible; set $F_u(r)=F_u(r)-\Pr(B_i)$.

4. If L is not empty, remove a record $\{\alpha;\beta;(\bar{\ell}(j),j\in N)\}$ from it and go to step 2.

5. **End** with $F_l(r)=F_u(r)=F(r)$.

5.2.2 計算隨機最短路徑績效值$F(r)$

本小節利用SSP的方法來計算圖5.1(a)的績效值$F(r)$。已知圖5.1(a)有0.2的機率成為圖5.1(b)與0.8的機率成為圖5.1(c)，且希望圖5.1(a)的最短路徑不超過$r=4$，則可分別利用SSP求得圖5.1(b)的績效值$F_{(b)}(4)$與圖5.1(c)的績效值$F_{(c)}(4)$，而圖5.1(a)的績效值即為$F(4)=0.2F_{(b)}(4)+0.8F_{(c)}(4)$。圖5.1(b)與圖5.1(c)的資訊如表所示。

表 5.2: 網路圖 5.1(b) 與 5.1(c) 的資訊

圖 5.1(b)	$x_i(1)$, $p_i(1)$	$x_i(2)$, $p_i(2)$	$x_i(3)$, $p_i(3)$	$x_i(4)$, $p_i(4)$	$x_i(5)$, $p_i(5)$
a_1	0, 0.01	1, 0.07	2, 0.01	3, 0.91	
a_2	0, 0.01	1, 0.03	2, 0.1	3, 0.16	4, 0.7
a_3	0, 0.02	1, 0.98			
a_5	0, 0.03	1, 0.01	2, 0.11	3, 0.85	
圖 5.1(c)	$x_i(1)$, $p_i(1)$	$x_i(2)$, $p_i(2)$	$x_i(3)$, $p_i(3)$	$x_i(4)$, $p_i(4)$	$x_i(5)$, $p_i(5)$
a_1	0, 0.01	1, 0.07	2, 0.01	3, 0.91	
a_2	0, 0.01	1, 0.03	2, 0.1	3, 0.16	4, 0.7
a_4	0, 0.01	1, 0.09	2, 0.9		
a_5	0, 0.03	1, 0.01	2, 0.11	3, 0.85	

接著依循 5.2.1 節演算法計算 $F_{(b)}(4)$。

Step 1: $R = \Omega$, $L = \emptyset$, $r = 4$, $F_l(4) = 0$, $F_u(4) = 1$, $(x_1(1), x_2(1), x_3(1), x_5(1)) = (0, 0, 0, 0)$, $d(t) = 0$

Step 2: $P = \{a_2, a_5\}$

Step 3: $x(\beta_5) - x(\alpha_5) = 3 < x(\beta_2) - x(\alpha_2) = 4$,$\gamma_2 = 2$,$\gamma_5 = 4$, $D = \{1 \le w_1 \le 4, 1 \le w_2 \le 2, 1 \le w_3 \le 2, 1 \le w_5 \le 4\}$, $B_2 = \{1 \le w_1 \le 4, 3 \le w_2 \le 5, 1 \le w_3 \le 2, 1 \le w_5 \le 4\}$, $F_l(4) = F_l(4) + \Pr(D) = 0 + 1 \times 0.04 \times 1 \times 1 = 0.04$.

For $\gamma_i < \beta_i (\gamma_2 = 2 < \beta_2 = 5)$,

increase length of arc a_2 to $x_2(3)$, then $(x_1(1), x_2(3), x_3(1), x_5(1)) = (0, 2, 0, 0)$, new shortest path $P = \{a_1, a_3, a_5\}$ and $d(t) = 0 < 4$. $L = \{[B_2] = (1, 3, 1, 1), [B_2] = (4, 5, 2, 4), P = \{a_1, a_3, a_5\}, d(t) = 0\}$.

Step 4: L is not empty, use B_2 and go to Step 2.

Step 2: $P = \{a_1, a_3, a_5\}$.

Step 3: $x(\beta_3) - x(\alpha_3) = 1 < x(\beta_1) - x(\alpha_1) = 3 = x(\beta_5) - x(\alpha_5) = 3$, $\gamma_1 = 4, \gamma_3 = 2, \gamma_5 = 1$, $D = \{1 \le w_1 \le 4, 3 \le w_2 \le 5, 1 \le w_3 \le 2, 1 \le w_5 \le 1\}$, $B_4 = \{1 \le w_1 \le 4, 3 \le w_2 \le 5, 1 \le w_3 \le 2, 2 \le w_5 \le 4\}$, $F_l(4) = F_l(4) + \Pr(D) = 0.04 + 1 \times 0.96 \times 1 \times 0.03 = 0.0688$

For $\gamma_i < \beta_i (\gamma_5 = 1 < \beta_5 = 4)$,

increase length of arc a_5 to $x_5(2)$, then $(x_1(1),\ x_2(3),\ x_3(1),\ x_5(2)) = (0,\ 2,\ 0,\ 1)$, new shortest path $P = \{a_1,\ a_3,\ a_5\}$ and $d(t) = 1 < 4$. $L = \{[B_5] = (1,\ 3,\ 1,\ 2), [B_5] = (4,\ 5,\ 2,\ 4),\ P = \{a_1,\ a_3,\ a_5\},\ d(t) = 1\}$.

Step 4: L is not empty, use B_5 and go to Step 2.

$\vdots$

Step 5: $F_l(4) = F_r(4) = 0.10004672$

得到 $F_{(b)}(4) = 0.10004672$，而 $F_{(c)}(4)$ 同樣依循演算法3計算。

Step 1: $R = \Omega,\ L = \emptyset,\ r = 4,\ F_l(4) = 0,\ F_u(4) = 1,\ (x_1(1),\ x_2(1),\ x_4(1),\ x_5(1)) = (0,\ 0,\ 0,\ 0),\ d(t) = 0$

Step 2: $P = \{a_2,\ a_5\}$

Step 3: $x(\beta_5) - x(\alpha_5) = 3 < x(\beta_2) - x(\alpha_2) = 4,\ \gamma_2 = 2,\ \gamma_5 = 4,\ D = \{1 \le w_1 \le 4, 1 \le w_2 \le 2, 1 \le w_4 \le 3, 1 \le w_5 \le 4\},\ B_2 = \{1 \le w_1 \le 4, 3 \le w_2 \le 5, 1 \le w_4 \le 3, 1 \le w_5 \le 4\},\ F_l(4) = F_l(4) + \Pr(D) = 0 + 1 \times 0.04 \times 1 \times 1 = 0.04.$

For $\gamma_i < \beta_i (\gamma_2 = 2 < \beta_2 = 5)$,

increase length of arc a_2 to $x_2(3)$, then $(x_1(1),\ x_2(3),\ x_4(1),\ x_5(1)) = (0,\ 2,\ 0,\ 0)$, new shortest path $P = \{a_1,\ a_4\}$ and $d(t) = 0 < 4$. $L = \{[B_2] = (1,\ 3,\ 1,\ 1), [B_2] = (4,\ 5,\ 3,\ 4),\ P = \{a_1,\ a_4\},\ d(t) = 0\}$.

Step 4: L is not empty, use B_2 and go to Step 2.

Step 2: $P = \{a_1,\ a_4\}$.

Step 3: $x(\beta_4) - x(\alpha_4) = 2 < x(\beta_1) - x(\alpha_1) = 3,\ \gamma_1 = 3,\ \gamma_4 = 3,\ D = \{1 \le w_1 \le 3,\ 3 \le w_2 \le 5,\ 1 \le w_4 \le 3,\ 1 \le w_5 \le 4\},\ B_1 = \{4 \le w_1 \le 4,\ 3 \le w_2 \le 5,\ 1 \le w_4 \le 3,\ 1 \le w_5 \le 4\},\ F_l(4) = F_l(4) + \Pr(D) = 0.04 + 0.09 \times 0.96 \times 1 \times 1 = 0.1264.$

For $\gamma_i < \beta_i (\gamma_1 = 3 < \beta_1 = 4)$,

increase length of arc a_1 to $x_1(4)$, then $(x_1(4), x_2(3), x_4(1), x_5(1)) = (3, 2, 0, 0)$, new shortest path $P = \{a_2, a_5\}$ and $d(t) = 2 < 4$. $L = \{[B_1] = (4, 3, 1, 1), [B_1] = (4, 5, 3, 4), P = \{a_2, a_5\}, d(t) = 2\}$.

Step 4: L is not empty, use B_1 and go to Step 2.

$\vdots$

Step 5: $F_l(4) = F_r(4)$ =0.2484856

得到$F_{(c)}(4) = 0.2484856$，因此$F(r) = 0.2 \times 0.10004672 + 0.8 \times 0.2484856 = 0.218797824$。

5.3 小結

本章節延伸了網路可靠度的議題，其一為當網路可靠度成為建構網路架構的條件式或門檻時，探討滿足限制式的最小成本，從此角度切入探討可靠度或能賦予更多管理決策上的意涵，也就是用較經濟的方式建構網路架構使得可靠度能滿足要求之門檻。另外，我們亦闡述如何利用SSP的方法來計算具有分配節點的隨機網路圖其最短路徑小於某一特定值r的機率值。

第六章

結論與未來發展方向

6.1 論文總結

具有分配節點的網路架構在現實的生產系統中時常見到，例如零組件經由某個工作站處理後依照特定比例分級，輸送至下一個工作站等應用。文獻中有關分配節點的特殊網路架構相關研究大都著重求解其最小成本等最佳化問題，而未曾針對可靠度方面有所著墨。本篇論文試圖從可靠度的角度切入，在具有分配節點的特殊網路架構下進行可靠度的運算，以提供整體系統可靠度的評估指標。

計算網路可靠度時會因為網路的組成元素過多導致求解的過程困難度大增，因此若能簡化網路圖必能加速求解可靠度，而分配節點其流進的流量必定依照餾化比例流出的特性，恰好能經由整合的過程大量減少網路圖的弧容量狀態組合。本研究修改了Lin (2005)的簡化分配網路方法以符合弧容量具有多狀態機率分配的狀況，先對網路進行簡化與整合處理，以加速其後之可靠度計算。

簡化整合過後的網路圖由於餾化比例的關係，會導致實質有效的弧容量狀態可能變為實數，若直接討論路徑上的流量組合可能會因為實數的稠密性而有無限多種可能，這也將使文獻中大部分的可靠度求解方式無法適用。故本論文以有限個數的弧容量狀態為基礎，提出MSE與PART-D兩種可靠度計算方式，以需求及弧容量限制式來找尋可行的容量狀態向量，進而利用事件聯集的方式計算可行容量組合之聯集機率以獲得可靠度。

除了提出兩種演算法以計算弧具有隨機多容量狀態的分配網路圖可靠度之外，我們更進一步探討如何以最小成本來建構一個可以滿足可靠度門檻的分配網路，以賦予網路可靠度在管理層面上更直觀的用途。

另外，若將研究之問題主體由弧容量改為弧長度時，所探討的問題便轉換成計算具有分配節點隨機網路上最短路徑小於給定門檻值之機率。針對這個問題，我們可依分配節點的特性將網路圖分解成數個不具有分配節點的子網路圖，此時再利用SSP法分別計算子網路圖上的機率值，再乘上形成子網路圖的機率即可。

本論文具體貢獻條列如下：

1. 將分配網路圖之簡化與整合流程擴大延伸至弧具有多容量狀態之分配網路 (Chapter 3)

2. 提出兩演算法以在具有分配節點的網路圖上計算網路可靠度 (Chapter 4)

3. 此兩演算法皆能處理容量狀態具有浮點數與容量間隔非一單位之情況 (Chapter 4)

4. 將可靠度延伸應用至滿足可靠度門檻之最小成本分配網路設計問題 (Section 5.1)

5. 探討分配節點存在於隨機弧長度之網路圖 (Chapter 5)

6. 應用SSP法計算隨機最短路徑之機率值 (Section 5.2)

6.2 未來研究方向

在計算多容量狀態分配網路圖之可靠度時，找尋可行之容量狀態向量的過程是採用窮舉弧容量狀態的方式，其列舉出來的容量狀態向量之數量隨弧的個數呈指數成長；即當弧a_i的狀態向量個數為K_i時，起始的狀態向量個數即為$\prod_{i=1}^{m} K_i$。再利用限制式將不可行的向量從此集合刪除。若能直接從限制式所限制的弧列舉可行的弧容量狀態以組成容量狀態向量將可大幅減少起始的容量狀態向量。拿第四章的範例來說，窮舉其所有的容量狀態向量會得到$4^8 = 65536$種組合 (即$K_i = 4\ \forall i$，且$|A| = 8$)，若針對起點的流出弧與訖點的流入弧限制其弧容量總和必須超過預定的運送量，即$u_{s,1}^{k_{s,1}} + u_{s,2}^{k_{s,2}} \geq 7$與$u_{3,t}^{k_{3,t}} + u_{4,t}^{k_{4,t}} \geq 7$，則在列舉容

量狀態向量組合時$(u_{s,1}^{k_{s,1}}, u_{s,2}^{k_{s,2}})$僅會有$(2, 6), (4, 4)$以及$(4, 6)$等三種組合，而$(u_{3,t}^{k_{3,t}}, u_{4,t}^{k_{4,t}})$亦僅會有$(2, 6), (4, 4), (4, 6), (6, 2), (6, 4)$以及$(6, 6)$等六種組合，因此起始的容量狀態向量便可大幅縮減成$3 \times 4^4 \times 6 = 4608$種組合；若再同時考慮D-node限制式又可再大幅度減少容量狀態向量的組合個數。然而儘管從限制式窮舉容量狀態向量會得到較少的組合狀況，但其仍只是針對起點的流出弧、終點的流入弧以及D-node的流入與流出弧進行限制，並非對每一條弧都有強力的限制，因此面對較複雜的網路圖時，列舉出來的容量狀態向量仍非常可觀。

儘管計算精確可靠度的演算法領域持續蓬勃發展，其演算法的複雜度仍無法克服(即計算網路可靠度仍為NP-hard問題)，因此發展在較短的時間內得到可靠度近似值的啟發式或近似(Approximate)演算法便成為另一項值得研究的主題。如Karger (1995)提出一套近似網路可靠度的演算法，但該演算法只探討了網路中的弧可能成功或失效的二元狀態，並未針對多容量狀態以及具分配節點的特殊網路架構進行研究，因此也可試圖將其方法應用於近似具多容量狀態的分配網路之可靠度。

參 考 文 獻

Alexopoulos, C. State space partitioning methods for stochastic shortest path problems. *Networks*, **30(1)**, 9–21, 1997.

Chang, M. D., Chen, C.-H. J. and Engquist, M. An improved primal simplex variant for pure processing networks. *ACM Transactions on Mathematical Software*, **15**(1), 64–78, 1989.

Chen, C.-H. J. and Engquist, M. A primal simplex approach to pure processing networks. *Management Science*, **32**(12), 1582–1589, 1986.

Colbourn, C. J. *The combinatorics of network reliability.* Oxford, New York: Oxford University Press, 1987.

Daly, M. S. 2001. *State space partition techniques for multiterminal and multicommodity flows in stochastic networks.* Unpublished doctoral dissertation, Georgia Institute of Technology.

Daly, M. S. and Alexopoulos, C. State space partition techniques for multiterminal flows in stochastic networks. *Networks*, **48**(2), 90-111, 2006.

Douilliez, P. and Jamoulle, E. Transportation networks with random arc capacities. *R.A.I.R.O.*, **3**, 45–49, 1972.

Fang, S. C. and Qi, L. Manufacturing network flows: A generalized network flow model for manufacturing process modeling. *Optimization Methods and Software*, **18**, 143–165, 2003.

Frank, H. and Hakimi, S. L. Probabilistic flow through a communications network. *IEEE Trans. Circuit Theor*, **CT-12**, 413–414, 1965.

Hsieh, C. C. and Lin, M. H. Reliability-oriented multi-resource allocation in a stochastic-flow network. *Reliability Engineering and System Safety*, **81**, 155–161, 2003.

Karger, D. R. 1995. A randomized fully polynomial time approximation scheme for the all terminal network reliability problem. In *Stoc '95: Proceedings of the twenty-seventh annual acm symposium on theory of computing* (p. 11-17). New York, NY, USA: ACM.

Koene, J. 1982. *Minimal cost flow in processing networks. a primal approach.* Unpublished doctoral dissertation, Eindhoven University of Technology, Eindhoven, The Netherlands.

Lin, J. C. 2005. *Maximum flows in distribution networks: problem generator and algorithms.* Unpublished master's thesis, National Cheng Kung University.

Lin, J. S., Jane, C. C. and Yuan, J. On reliability evaluation of a capacitated-flow network in terms of minimal pathsets. *Networks*, **25**, 131–138, 1995.

Lin, S. J. 2006. *A network simplex algorithm for the minimum distribution cost problem.* Unpublished master's thesis, National Cheng Kung University.

Lin, Y. K. On reliability evaluation of a stochastic-flow network in terms of minimal cuts. *Journal of Chinese Institute of Industrial Engineers*, **18**(3), 49–54, 2001.

Lin, Y. K. Study on the system capacity for a multicommodity stochastic-flow network with node failure. *Reliability Engineering and System Safety*, **78**, 57–62, 2002a.

Lin, Y. K. Using minimal cuts to evaluate the system reliability of a stochastic-flow network with failures at nodes and arcs. *Reliability Engineering and System Safety*, **75**, 41–46, 2002b.

Lin, Y. K. On the multicommodity reliability for a stochastic-flow network with node failure under budget constraint. *Journal of the Chinese Institute of Industrial Engineers*, **20**(1), 42–48, 2003.

Lin, Y. K. Reliability of a stochastic-flow network with unreliable branches nodes, under budget constraints. *IEEE Transactions on Reliability*, **53**(3), 381–387, 2004.

Lin, Y. K. Generate upper boundary vectors meeting the demand and budget for a p-commodity network with unreliable nodes. *European Journal of Operational Research*, **183**, 253–262, 2007a.

Lin, Y. K. Reliability evaluation for an information network with node failure under cost constraint. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, **37**(2), 180–188, 2007b.

Mine, H. Reliability of physical systems. *IRE Transactions Circuit Theory*, **CT-6**, 138–151, 1959.

Moore, E. F. and Shannon, C. E. Reliable circuit using less reliable relays. *Journal Franklin Institute*, **262, 263**, 191-208, 281-297, 1956.

Moskowitz, F. The analysis of redundancy networks. *AIEE Transactions on Communications Electronics*, **39**, 627–632, 1958.

Provan, J. S. and Ball, M. O. The complexity of counting cuts and of computing the probability that a graph is connected. *SIAM Journal on Computing*, **12**, 777-788, 1983.

Ramirez-Marquez, J. E. and Coit, D. W. A monte-carlo simulation approach for approximating multi-state two terminal reliability. *Reliability Engineering and System Safety*, **87**, 253–264, 2005.

Satyanarayana, A. and Prabhakar, A. New topological formula and rapid algorithm for reliability analysis of complex networks. *IEEE Transactions on Reliability*, **R-27**, 82–100, 1978.

Sheu, R. L., Ting, M. J. and Wang, I. L. Maximum flow problem in the distribution network. *Journal of Industrial and Management Optimization*, **2**(3), 237–254, 2006.

Yeh, W. C. A revised layered-network algorithm to search for all d-minpaths of a limited-flow acyclic network. *IEEE Transactions on Reliability*, **47**, 436–442, 1998.

Yeh, W. C. A simple algorithm to search for all d-mps with unreliable nodes. *Reliability Engineering and System Safety*, **73**, 49–54, 2001.

Yeh, W. C. A novel method for the network reliability in terms of capacitated-minimum-paths without knowing minimum-paths in advance. *Journal of the Operation Research Society*, **56**, 1235–1240, 2005.

Yeh, W. C. A simple algorithm to search for all mcs in networks. *European Journal of Operational Research*, **174**, 1694–1705, 2006.

Yeh, W. C. A simple minimal path method for estimating the weighted multi-commodity multistate unreliable networks reliability. *Reliability Engineering and System Safety*, **93**, 125–136, 2008.