網路最佳化 期末報告

An O($|V|^3$) Algorithm For Finding Maximum Flows In Networks

Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems

學生： 洪佳琦 H34941268

林依巧 R36971234

I. Paper I

1. Title：

An O($|V|^3$) Algorithm For Finding Maximum Flows In Networks

2. Source：

V.M. MALHOTRA, M. Pramodh KUMAR and S.N. MAHESHWARI

Computer Science Program, Indian Institute of Technology, Kanpur 208016, India

3. What this paper is about, what I have learned from this paper...etc：

(1) 定義最大流量問題：

一有方向性的網路圖 $G=(V,E)$，V 是所有節點的集合，其中 s 節點為起始節點，t 為終點；E 則是代表所有節線的集合，$c(u,v)$ 表示每一條節線的流量的容量限制，其為一非負的實數，$f(u,v)$ 則是流過節線 (u,v) 的流量，所以每條節線皆需符合以下條件：

$$0 \le f(u,v) \le c(u,v) \text{ ； } \sum_{(u,w)\in E} f(u,w) - \sum_{(v,u)\in E} f(v,u) = 0$$

意義為經過節線的流量不能大於該節線的容量限制並且流進來的流量等於流出去的流量，如何在這些限制下由起點運送最大的流量至終點，即為網路的最大流量問題。

(2) 文獻與背景：

最大流量的問題已被定義過，也有許多學者研究此類的問題，例如：Dinic、Even、Tarjan…等，都有發表關於求解最大流量問題的演算法。本篇文章探討 per-stage flow problem，此類的問題將原本的網路圖分成幾個階段，每一階段所找到的路徑長度皆相同，也就是後來學者所 Dinic 所提的階段問題，而 Dinic 的演算法其複雜度為每一階段 $O(|V|\bullet|E|)$；Karzanov 則是提出 $O(|V|^2)$ 的演算法；作者希望提出此演算法，每一階段最多需要 $O(|V|^2)$ 步驟，但其邏輯較 Karzanov 提出的簡單。

(3) 演算法內容：

➢ 基本符號介紹：

考慮一網路$G=(V,E)$，其中每一條路經的長度皆一樣，流經的流量為f，所有節點有其潛在流量(flow potential)，以$\rho_f(v)$表示之，潛在流量的意義為該點可以支配的額外流量(extra flow)，其算法為：

$$\rho_f(v)=\min\{\sum_{(v,w)\in E}(c(v,w)-f(v,w)),\ \sum_{(u,v)\in E}(c(u,v)-f(u,v))\}\quad (v\neq s, v\neq t)$$

$$\rho_f(s)=\sum_{(s,w)\in E}(c(s,w)-f(s,w))$$

$$\rho_f(t)=\sum_{(u,t)\in E}(c(u,t)-f(u,t))$$

其中 r 點稱為參考點(reference node)，其潛在流量為參考潛在流量(reference potential)：$\rho_f(r)=\min_{v\in V}\{\rho_f(v)\}$。

➢ (每一階段)演算邏輯與步驟：

整個演算法的邏輯為此(Lemma)：若 r 為網路圖$G=(V,E)$中的參考點，讓流量$f=\rho_f(r)$並且使得下一階段的流量f'其$\rho_{f'}(r)$=0；其意思為在此階段中，計算各節點的潛在流量，找出參考點，如此一來此階段的流量便是$\rho_f(r)$，亦即此流量塞滿此節點。

以上面的邏輯為基礎，求解最大流量問題，其步驟為：

Step 1：計算所有節點的潛在流量$\rho_f(v)$，並在其中找到$\rho_f(r)$。

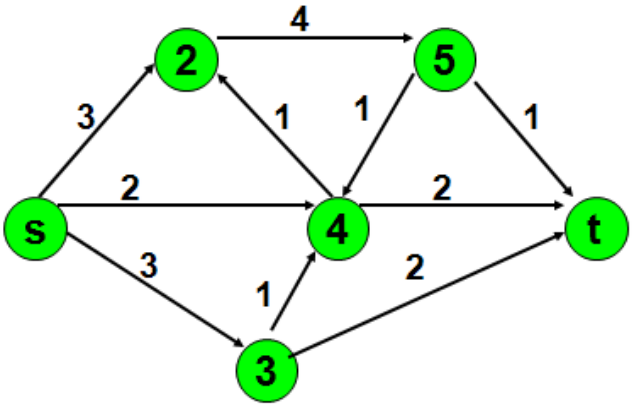
Step 2：由參考點 r 向終點送流量$f=\rho_f(r)$，並向起點方向送f；流經的節線重新計算目前的$c(u,v)-f(u,v)$。

Step3：將被塞滿的節線刪除，重複回到第一步驟，若某節點之流入或流出(incoming or outgoing)節線皆已被塞爆，則將該節點刪去，繼續重複 step1，直到找不到參考點，則此階段停止。

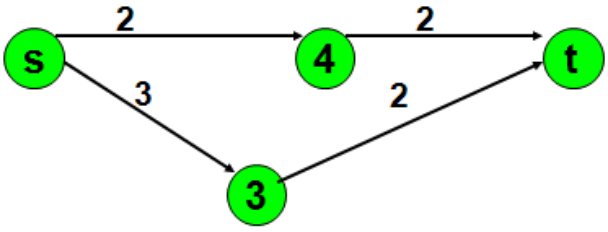
以下為一網路圖(圖一)，以其為例：

首先找出所有的 layer network，圖二為路徑長度為 2 的網路圖，即為第一階段的網路圖，

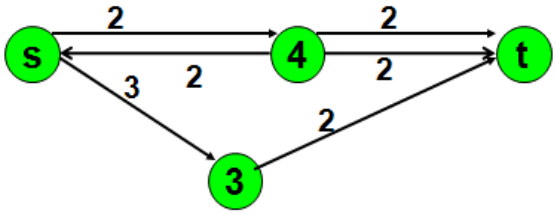
第一步為計算潛在流量，計算的結果為：$\rho_f(s)=5$、$\rho_f(3)=3$、$\rho_f(4)=2$、$\rho_f(5)=2$，因此節點 4 為參考點，第一輪的流量為 2；第二步以節點 4 開始推送 2 單位至終點，因此 $f(4,t)=2$，再由節點 4 往回推送兩單位回起點，$f(4,s)=2$（如圖三），接著計算相關節線的 $c(u,v)-f(u,v)$；第三步發現與節點 4 相連的節線 $c(s,4)-f(s,4)$、$c(4,t)-f(4,t)$ 皆為零，亦即無法再行經此節線，因此刪去節線 (s,4)、(4,t)，再回到第一步驟直到所有點皆被刪除，停止此階段。



（圖一）



（圖二）



（圖三）

(4) 複雜度分析：

此一網路問題分成幾個階段，每一個階段的路徑皆等長，因此最多的情況是 V-1 個階段，即路徑長度從 1 到 V-1；又每一階段中每一輪 i 其複雜度為：$O(|V|+|E_i|)$，其中 $|E_i|$ 是指被刪除的節線數，$|V|$ 是指其他有流量流過但未被塞滿的節線數，因此該階段的總和為 $O(\sum_i(|V|+|E_i|)) = O(|V|^2+|E|) = O(|V|^2)$，又 per-stage flow 問題最多有|V|-1 個階段，此演算法的複雜度即為 $O(|V|^3)$。

II. Paper II

1. Title：

Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems

2. Source：

JACK EDMONDS ;*University of Waterloo, Waterloo, Ontario, Canada*

RICHARD M. KARP ;*University of California, Berkeley, California*

3. What this paper is about, what I have learned from this paper...etc：

(1) 大綱：

本篇文章介紹一個新的最大流量問題的演算法以及最小成本流量的問題，在最大流量問題的部分作者首先呈現出Ford-Fulkerson的演算邏輯，接著指出在某一種情況底下，其演算法的執行會花很多時間，所以作者提出了兩個方法，第一個方法希望使問題無論在何種圖形下，皆僅最多需要 $\frac{1}{4}(n^3-n)$ 個擴張路徑，接著提出一個改良的方法，使得最不好的情況下需要 $n^2\ln n + n^2\ln\bar{c}$ 個擴張路徑。

(2) 定義問題以及Ford-Fulkerson的演算法：

- 問題定義：

有一網路問題N，$\{u,v,...\}$為節點的集合，而$(u,v),u \neq v$稱為節線，網路中有一條回溯的節線(t,s)，其中s為起點，t為終點，除了節線(t,s)外，其餘所有節線皆屬於集合A，而$c(u,v)$為該節線的流量限制，為一非負的數，而$f(u,v)$表示流過的流量，因此所有屬於A集合的節線(u,v)符合$0 \leq f(u,v) \leq c(u,v)$以及所有的節點$u$皆符合$\sum_{(u,w)\in E} f(u,v) - \sum_{(v,u)\in E} f(v,u) = 0$，在這些定義下，希望解決最大流量的問題，即該如何地流可以運送最多單位的流量。

➢ 標籤法則(Labeling method)：

令$u_1,u_2,...u_p$為不同的點，其中(u_i,u_{i+1})或(u_{i+1},u_i)皆為節線，挑出來的結果節線(u_i,u_{i+1})稱為往前的節線(forward arcs)，其餘則為反向的節線(reverse arcs)，而網路中有一流量f，在擴張路徑(augmenting path)流著，擴張路徑則是指一條起點為s到終點t的路徑，這些路徑符合以下情形：

(a) 若$(u_i,u_{i+1}) \in A$且$(u_{i+1},u_i) \notin A$，則$\varepsilon_i = c(u_i,u_{i+1}) - f(u_i,u_{i+1}) > 0$；

(b) 若$(u_i,u_{i+1}) \notin A$且$(u_{i+1},u_i) \in A$，則$\varepsilon_i = f(u_{i+1},u_i) > 0$；

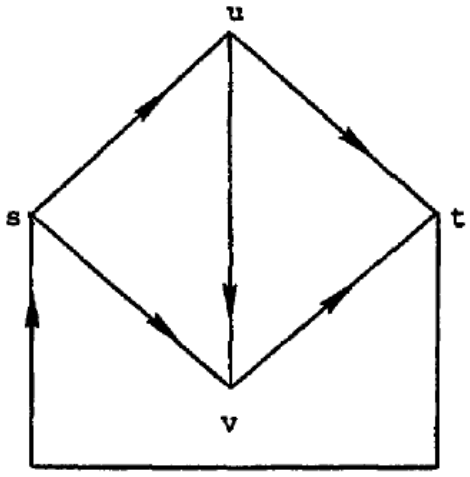
(c) 若$(u_i,u_{i+1}) \in A$且$(u_{i+1},u_i) \in A$

則$\varepsilon_i = c(u_i,u_{i+1}) - f(u_i,u_{i+1}) + f(u_{i+1},u_i) > 0$；

在擴張路徑P中，令$\varepsilon = \min \varepsilon_i > 0$，稱此$(u_i,u_{i+1})$或$(u_{i+1},u_i)$為瓶頸節線，當流量為$f$時，若屬於(a)則$(u_i,u_{i+1})$節線增加流量$\varepsilon$，屬於(b)則減少$(u_{i+1},u_i)$的流量$\varepsilon$，若屬於(c)，則$(u_i,u_{i+1})$節線增加流量$\min(\varepsilon, c(u_i,u_{i+1}) - f(u_i,u_{i+1}))$並且$(u_{i+1},u_i)$減少$\max(0, \varepsilon - c(u_i,u_{i+1}) + f(u_i,u_{i+1}))$，此流量$f$並不一定是最大流量。此方法建立一個流量順序$F = f^0, f^1, f^2, ...$，$f^k$表示建立第k次的流量，此順序到終止時即為最大流量；總結而言，此方法的邏輯在於每一個階段定義出一條擴張路徑，接著運送路徑中的瓶頸流量，再進行下一階段，當不能產生擴張路徑時，則停止演算法而能得到最大流量，此即Ford-Fulkerson的演算法邏輯。

這裡有另一個假設：假設所有節線的容量限制是整數的，且每一階段運送的流量也是整數的，則ε也就會是整數的，因此f^k為整數，所以整體流量F便會是整數。

然而下圖為一特殊的範例，假設$c(u,v)$為 1，其餘屬於 A 集合的節線，其容量限制為 M，我們可知最大流量為 2M，但用此種演算法的邏輯，若選到節線$(s,u)(u,v)(v,t)$或$(s,v)(v,u)(u,t)$，則瓶頸流量為 1，如此一來總是需要花費很多運算的時間，且可以看出當容量限制越大時，所需找的擴張路徑則越多，因此接下來會有兩個精進的做法，改善此問題。



(3) 第一次的精進(refinement)：

➢ 找尋擴張路徑的方法：

令擴張路徑 P 有一流量f，N^f與N擁有相同的節點，且存在節線(u,v)符合下面兩條件的其中一個，組成一由起點至終點的擴張路徑：

$(u,v)\in A$且$c(u,v)-f(u,v)>0$或$(v,u)\in A$且$f(v,u)>0$

而每一次流的量為該次的瓶頸流量，便是P^f的瓶頸節線。

找尋由起點開始至終點有方向性的路徑，其步驟為：

Step1：標記「起點 s」。

Step2：於起點 s 做掃描，去標記尚未標記的節點，並記錄「未標記節點」之前繼者(predecessor)為該節線上之「已標記節點」，重覆此步驟至終

點 t 被標記，且過程中遵守先被標記者，先進行下一次的掃描，最後即找到此條擴張路徑，且此路徑為路徑長度最小者。

Step3：利用各節點之前繼者向前回溯，以搜尋「流量擴張路徑」。

➢ 定理證明：

經過上面說明的方法，找尋出路徑最小者，作者提出其看法，認為每一次運送流量的擴張路徑，由路徑短的開始運送，此方法所需的擴張路徑數目最多僅需要$\frac{1}{4}\left(n^3-n\right)$個，並提出定理一，證明其想法。

✧ 定理一：在標籤法則中找尋最大流量，若每一次找的擴張路徑都是包含最少節線的路徑，則在找尋最大流量的過程中，最多不會找尋超過$\frac{1}{4}\left(n^3-n\right)$個擴張路徑。

要證明這項定理需先明白下面兩個 Lemma：

Lemma1：如果 k<m，且(u,v)是P^k和f^k及P^m和f^m的瓶頸節線，則存在l，使得$k<l<m,(v,u)\in P^l$。

直覺想法：當(u,v)是瓶頸節線時，則下一次的擴張路徑不會出現此節線，若後來希望看到(u,v)成為瓶頸節線，則需要(v,u)流某些流量。

證明：若(u,v)是P^k和f^k的瓶頸節線，則$(u,v)\notin N^{k+1}$，且假設$(u,v)\in N^{k+1}$則$(u,v)\in N^k$或$(v,u)\in P^k$；意思是當第 k 次是瓶頸時，下一次便不會是瓶頸，又因為$(u,v)\in N^{k+1}$且$(u,v)\notin N^k$因此$(v,u)\in P^k$。

Lemma2：若$k<l,(u,v)\in P^k$且$(v,u)\in P^l$，則$\delta^l(s,t)\geq\delta^k(s,t)+2$。其中$\delta^l(s,t)$是指在第$l$次流過起點 s 與終點 t 的距離。

證明：因為$(v,u)\in P^l$，$\delta^l(s,t)=\delta^l(s,v)+1+\delta^l(u,t)$，且$\delta^l(s,v)\geq\delta^k(s,v)$、$\delta^l(u,t)\geq\delta^k(u,t)$，所以

$$\delta^l(s,t)\geq\delta^k(s,v)+1+\delta^k(u,t)=(1+\delta^k(s,u))+1+(1+\delta^k(v,t))=2+\delta^k(s,t)$$

利用上面兩個 lemma 證明定理一：假設$\{l_i\}$是所有包含(u,v)或(v,u)為瓶頸節線的流動順序，因此符合下列情形，

$(u,v) \in P^{l_j}$, j odd and $(v,u) \in P^{l_j}$, j even；

$(u,v) \in P^{l_j}$ j even and $(v,u) \in P^{l_j}$, j odd；

又路徑長度最長為 N-1，也就是 $\delta^{l_j}(s,t) \le n-1$，所以在 $\{l_j\}$ 的順序中，最多發生 (u,v) 或 (v,u) 為瓶頸的次數為 $\frac{1}{2}(n-1)+1=\frac{1}{2}(n+1)$，所以總共的擴充路徑數目為：

$$\frac{n+1}{2}\binom{n}{2}=\frac{n^3-n}{4}.$$

（4） 第二次精進(Refinement)

➢ 參數設定

若將網路圖 N 所有節點，分類為「節點集合 X」與「節點集合 $\overline{X}$ 」，且 $s \in X$ ，$t \in \overline{X}$，並為了精進標籤法則(Labeling method)，而於每回合中均選擇俱有最大擴充量之「流量擴張路徑(flow-augmenting path)」。

而其相關參數設定如下所示:

N	容量(capacity)均為整數之網路圖
X	包含起點 s 之部份節點集合，$s \in X$ 。
$\overline{X}$	包含終點 t 之部份節點集合，$t \in \overline{X}$ 。
M	「集合節點 X 中之節線數」與「集合節點 $\overline{X}$ 中之節線數」之總合上限值，其中限定 $M>1$。
$f^*(t,s)$	最佳最大流量值。
$e(u,v)$	若令 (u,v) 屬於網路圖 N^f 之節線，則 $e(u,v)$ 為該節線上之「殘餘流量」。(註: $e(u,v)=\varepsilon$)

則於此精進標籤法則可歸納出下述之定理。

➢ 定理二

設某一網路圖 N 其所有容量均為整數，並欲採用「第二類精進標籤法則」(註: 下述均簡稱為「標籤法則」)選擇最大流量之節線組合，倘若基於俱最大擴充量之準則下，選擇「流量擴張路徑」，則此最大流量網路問題至多將執行 $1+\log_{M/(M-1)} f^*(t,s)$ 次之擴張。

而後，為證明定理二所提及之擴張次數，故下述將先針對「殘餘流量」、「流量擴張路徑」之搜尋方法等內容做一簡單介紹。

✧ 殘餘流量($e(u,v)$)

若網路圖 N 中可搜尋出一條擴張路徑以推送流量 f，並將該路徑存入網路圖 N^f 中，若其擴張路徑上之瓶頸節線亦為(u, v)，則該瓶頸節線之「殘餘流量」共可分為下述三種情況:

(1) 若 $(u,v)\in A$ 與 $(v,u)\notin A$ 之情況成立，則 $e(u,v)=c(u,v)-f(u,v)$ 。

(2) 若 $(u,v)\notin A$ 與 $(v,u)\in A$ 之情況成立，則 $e(u,v)=f(u,v)$ 。

(3) 若 $(u,v)\in A$ 與 $(v,u)\in A$ 之情況成立，則 $e(u,v)=c(u,v)-f(u,v)+f(v,u)$。

✧ 流量擴張路徑之搜尋方法

若欲於網路圖 N^f 中，依循「Bottleneck extrema」[註]所提及之方法，搜尋出一條由「起點 s」至「終點 t」之擴張路徑，則其相關步驟如下所示:

Step1：標記「起點 s」。

Step2：於起點 s 做掃描，搜尋符合「由『已標記』至『未標記』」之節線，並記錄「未標記節點」之前繼者(predecessor)為該節線上之「已標記節點」。若「未標記節點」不為「終點 t」時，則重覆步驟二;反之，跳至步驟三。

Step3：利用各節點之前繼者向前回溯，以搜尋「流量擴張路徑」。

[註] EDMONDS, J., AND FULKERSON, D. R. Bottleneck extrema. RAND Corp. Memorandum RM-5373-PR (Jan. 1968)

➢ 定理二之證明

若網路圖 N 所有節點，可分類為「節點集合 X」與「節點集合 $\overline{X}$」，且 $s \in X$，$t \in \overline{X}$，則可知

$$c\left(X,\overline{X}\right)=\sum_{\substack{u\in X\\ v\in\overline{X}\\ (u,v)\in A}} f(u,v) \quad , \quad f\left(\overline{X},X\right)=\sum_{\substack{u\in\overline{X}\\ v\in X\\ (u,v)\in A}} f(u,v)$$

故對於任何流量 f 而言，均滿足「兩集合間之容量總合」大於等於「已推送之流量總合」之特性，故其關係式為:

$$c\left(X,\overline{X}\right)\geq f\left(X,\overline{X}\right)-f\left(\overline{X},X\right)=f(t,s) \tag{1}$$

此外，「標籤法則」於各回合中求得之「擴張流量」分別為 $f^0, f^1, ..., f^k, ...$，故可知「第 k 回合之推送量」可由第 k+1 回合與第 k 回合間累計「最大流量值」之差量所得之，而其關係式如下所示:

$$\varepsilon^k = f^{k+1}(t,s)-f^k(t,s) \tag{2}$$

又因起點 u 屬於 X 集合，終點 v 屬於 $\overline{X}$ 集合，第 k 回合之網路圖 N^k 包含該節線 (u,v)，鑑於定理二推送「最大擴充量」之準則，故該節線上之「殘餘流量 $e(u,v)$」應小於等於「第 k 回合之推送量」，而其關係式如下所示:

$$e(u,v)\leq\varepsilon^k \tag{3}$$

而(3)式僅指出「單一節線」殘餘流量與推送量之關係式，倘若將 X 與 $\overline{X}$ 兩集合間所有節線綜合探討之，亦可推知兩集合間之「殘餘流量」應小於等於「第 k 回合之推送量」，而關係式則可表示為:

$$c\left(X,\overline{X}\right)-\left[f^k\left(X,\overline{X}\right)-f^k\left(\overline{X},X\right)\right]$$

$$\leq\varepsilon^k\left[\left\{(u,v)\,|\,u\in X, v\in\overline{X}, (u,v)\in A \quad \text{or } (v,u)\in A)\right\}\right] \leq \varepsilon^k M \tag{4}$$

又因第「k 回合之最大流量值」之數學表式示為:

$$f^k(t,s)=f^k\left(X,\overline{X}\right)-f^k\left(\overline{X},X\right) \tag{5}$$

且依演算法求得之「最佳最大流量值」應小於等於「總運送容量」，而其表示式為:

$$f^*(t,s) \le c\left(X,\overline{X}\right) \tag{6}$$

故可將(6)式減去(5)式，而得下列之數學表式關係式:

$$f^*(t,s) - f^k(t,s) \le c\left(X,\overline{X}\right) - \left[f^k\left(X,\overline{X}\right) - f^k\left(\overline{X},X\right)\right]$$

又可由(4)式得知 $c\left(X,\overline{X}\right) - \left[f^k\left(X,\overline{X}\right) - f^k\left(\overline{X},X\right)\right] \le \varepsilon^k M$

故可推導出下列關係式

$$f^*(t,s) - f^k(t,s) \le \varepsilon^k M \tag{7}$$

即 $$f^*(t,s) - f^k(t,s) \le \left[f^{k+1}(t,s) - f^k(t,s)\right] M$$

同理可知 $$f^*(t,s) - f^{k+1}(t,s) \le \left[f^*(t,s) - f^k(t,s)\right](1 - M^{-1}) \tag{8}$$

又因 $$\left[f^*(t,s) - f^k(t,s)\right](1 - M^{-1}) \le f^*(t,s)(1 - M^{-1})^k$$

則 $$f^*(t,s) - f^k(t,s) \le f^*(t,s)(1 - M^{-1})^k \tag{9}$$

有鑑於網路圖上之容積與每回合推送流量均為整數之特性，並假設於第 k 回合仍未求得最佳最大流量，故「最佳最大流量」與「第 k 回合最大流量」至少存在一單位之差量，而其關係式可表示為:

$$f^*(t,s) - f^k(t,s) \ge 1 \tag{10}$$

故由(9)式之關係式亦可得知

$$f^*(t,s)(1 - M^{-1})^k \ge 1 \tag{11}$$

而後，若欲推估演算法可於第幾回合求得最佳解，則應推算回合指數 k 之上限值，故將(11)式做對數運算，而其推導過程如下所式:

$$\log_{(1-1/M)} f^*(t,s) + k\left[\log_{(1-1/M)}\left(1 - 1/M\right)\right] \ge \log_{(1-1/M)} 1$$

$$= \log_{(1-1/M)} f^*(t,s) + k \ge 0$$

故 $$k \le -\log_{1-1/M} f^*(t,s) = \log_{M/(M-1)} f^*(t,s) \tag{12}$$

因此，由(12)式則得知演算法結束回合之上限次數為:

$$1+\log_{M/(M-1)} f^*(t,s) \tag{13}$$

而後，若令$\bar{c}$為節線上之「平均容量」，則「最佳最大流量」與「兩集合中之節線總數」之上限為:

$$f^*(t,s) \le \bar{c}n^2$$

$$M \le \frac{1}{2}n^2$$

故可推知

$$\log_{M/(M-1)} f^*(t,s) \le \log_{1+2/(n^2-2)}(n^2\bar{c}) = \frac{\ln\left(n^2\bar{c}\right)}{\ln\left(1+2/(n^2-2)\right)} \tag{14}$$

若更進步探討上式中分母之關係式，則可知得:

$$\ln\left(1+\frac{2}{n^2-2}\right) \ge \ln\left(1+\frac{2}{n^2}\right) \ge \frac{2}{n^2} - \frac{1}{2}\left(\frac{2}{n^2}\right)^2 = 2\left(\frac{1}{n^2}-\frac{1}{n^4}\right)$$

故(14)式亦可化整為:

$$\log_{M/(M-1)} f^*(t,s) \le \frac{\ln\left(n^2\bar{c}\right)}{\ln\left(1+2/(n^2-2)\right)} = \frac{2\ln n+\ln\bar{c}}{\ln\left(1+2/(n^2-2)\right)} \le \frac{2\ln n+\ln\bar{c}}{2\left(1/n^2-1/n^4\right)} \tag{15}$$

而後，若欲將演算法「結束回合之上限次數」(即(13)式)以參數「節點總數 n」與「平均容量$\bar{c}$」表示之，並利用(15)式之關係式，則可推知該演算法之擴張次數(即擴張回合數)之上限值，應為:

$$1+\log_{M/(M-1)} f^*(t,s) \le 1+\frac{2\ln n+\ln\bar{c}}{2\left(1/n^2-1/n^4\right)} = 1+\frac{n^4}{2n^2-2}\left(2\ln n+\ln\bar{c}\right) \tag{16}$$

此外，又更進一步化簡關係式(16)，以推估該演算法之「擴張次數」，亦可得知此演算法之複雜度，而化簡過程如下所示:

$$1+\frac{n^4}{2n^2-2}\left(2\ln n+\ln \bar{c}\right)=1+\frac{1}{2}\left[\frac{n^4}{n^2-1}\left(2\ln n+\ln \bar{c}\right)\right]$$

$$=1+\frac{1}{2}\left[\frac{2n^4}{n^2-1}\left(\ln n\right)\right]+\frac{1}{2}\left[\frac{n^4}{n^2-1}\left(\ln \bar{c}\right)\right]$$

$$\approx \frac{1}{2}\left[\frac{2n^4}{n^2}\left(\ln n\right)\right]+\frac{1}{2}\left[\frac{n^4}{n^2}\left(\ln \bar{c}\right)\right]$$

$$=n^2\ln n+n^2\ln \bar{c}$$

$$\Rightarrow O(n^2\ln n+n^2\ln \bar{c})$$

➢ 小結

綜合而言之，若利用「定理二」指出之以最大擴充量法則下選擇「流量擴張路徑」(即「定理二」)，則每回合均可求得一條「擴張路徑」，並由上述一連串之推導過程，可推估於最不理想情況下，至多將執行$n^2\ln n+n^2\ln \bar{c}$次擴張。故可推論該演算法之複雜度為$O(n^2\ln n+n^2\ln \bar{c})$。

4. Conclusion

這兩篇文章分別提出演算法，皆是希望更簡化、更快速找到最大流量問題的解，第一篇文章利用潛在流量的概念，執行一複雜度為$O(|V|^3)$的演算法，而第二篇文章則是改良 Ford-Fulkerson 的演算法，加入擴充路徑的選擇條件，由路徑短的開始選擇，找到擴充路徑所需選取的最多次數。