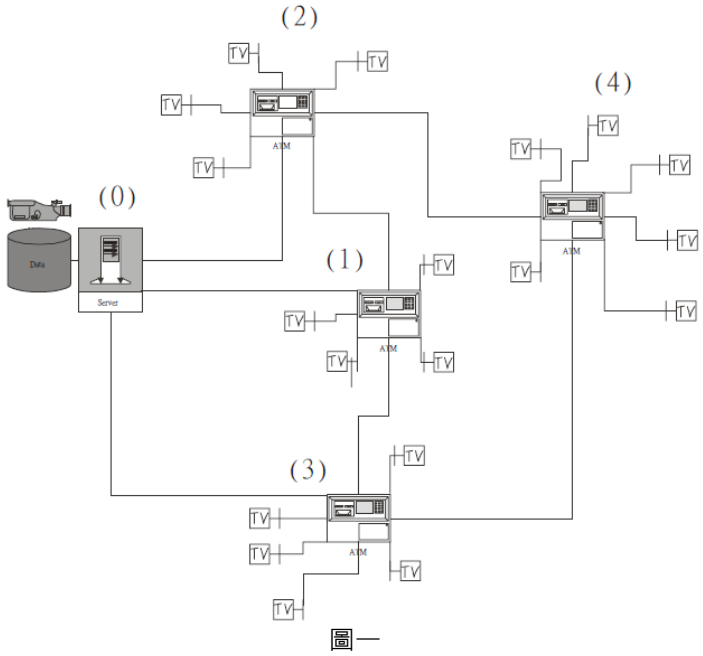
MOD optimal data-routing

110 ORA_project_羅祺育_林芮佑

一、 研究動機

由於近年來上網的人口數快速成長，因此網際網路內容供應商 (ICP，Internet Content Provider) 提供的服務將會是未來的趨勢，MOD (Multimedia on Demand) 即為 ICP 提供的主要服務之一。
近年在通訊方面，由於頻寬問題有大幅的進步，因此如果能進一步探討網路系統之路由最佳化問題，不僅可以更加滿足使用者的需求，對於供應商也能有更大的收益。
本專題為了實現路由最佳化，使用線性規劃及 Branch-and-bound 演算法來實驗，並為了分析其效能，我們探討不同節點數下以暴力演算法與 Branch-and-bound 演算法其各求解之搜尋範圍之比較。

二、 問題定義



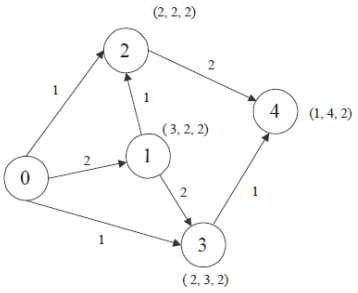
圖一

- **說明：**

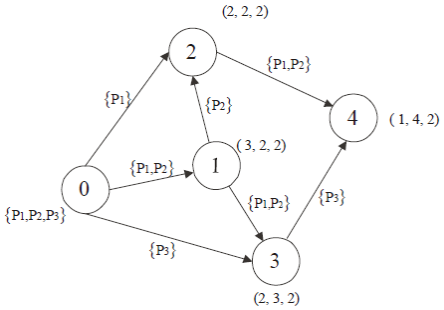
  以圖一為例，節點(0)為伺服器，節點(1)-(4)為集散點 (ATM switch)，分別連接到各自的用戶端，其中集散點會蒐集用戶端對影像的需求並告訴伺服器，且當收到上游傳來的影像，會繼續複製給下游節點。當影片進入某一節點，則視為該節點連接的每個用戶皆可觀看此影片。此外，每一條網路連結邊(i, j)都有一頻寬值r_{ij}，表示進入此網路連結邊的影片總和不可超過此頻寬限制 (本專題假設每影片占用一頻寬單位)。

- **Example：**

  假設此網路路由有 n 個節點 (不含 MOD 伺服器)，且 MOD 伺服器持有$P_1, P_2, \dots, P_m$等 m 部影片。節點 j 對影片的需求人數，以$(b_j^1, b_j^2, \dots, b_j^m)$表示。根據圖一可以畫出圖二的問題模型。其中節點上的向量代表各個影片的需求人數，而連結邊上的數字則代表頻寬。



圖二



圖三

■ Indices：

1. m：影片的種類數量
2. (i, j)：節點 i 連線到節點 j

■ Parameters:

1. r_{ij}：(i, j) 連線之頻寬
2. P_k：第 k 個影片
3. b_j^k：節點 j 對於第 k 個影片的需求人數

■ Decision variable：

x_{ij}^k ：二元變數，表示 (i, j) 邊上是否有影片 k 進入

■ Objective:

$$\text{Maximize} \sum_{(i,j)\in E} \sum_{k=1}^{m} b_j^k \cdot x_{ij}^k$$

■ Constraint:

$$\sum_{k=1}^{m} x_{ij}^k \le r_{ij} \qquad \text{where } x_{ij}^k = \begin{cases} 1 & \text{if } (i,j) \in E(T_k) \\ 0 & \text{otherwise} \end{cases}$$

- $T_k = the\ k_{th}\ video\ deliver\ tree$
- $m = the\ amount\ of\ the\ video$

→ 進入節點 j 的所有影片總和必須小於頻寬

$$\sum_{(i,j)\in E'} x_{ij}^k \le 1 \qquad \text{where } x_{ij}^k = \begin{cases} 1 & \text{if } (i,j) \in E'(T_k) \\ 0 & \text{otherwise} \end{cases}$$

- $E' = the\ edge\ that\ into\ the\ node\ j$

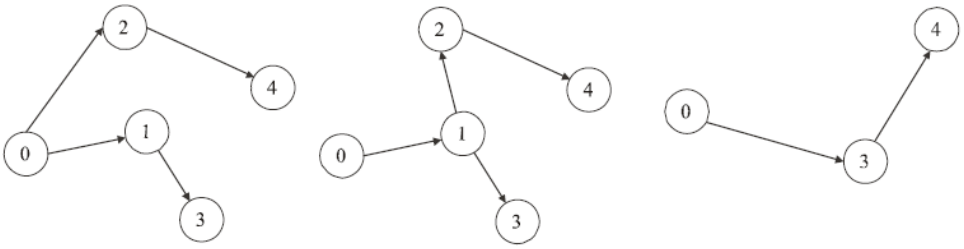
→ 避免會有重複影片進入節點 j

$$\sum_{(i,j)\in E''} x_{ij}^k \ge x_{jr}^k \qquad \text{where } x_{ij}^k = \begin{cases} 1 & \text{if } (i,j) \in E''(T_k) \\ 0 & \text{otherwise} \end{cases}$$

- $E'' = the\ edge\ that\ into\ the\ node\ j$
- $(j, r) = the\ edge\ that\ leave\ the\ node\ j$

→ 避免節點 j 傳出未持有的影片

根據以上限制式即可求出如圖三之解，圖四為此範例可行解所畫出之群播樹。因此，探討此問題相當於求出 m 個以 MOD 伺服器位置為樹根的群播樹 $T_1, T_2, \ldots, T_m$，使得整體所服務的用戶數量為最大。

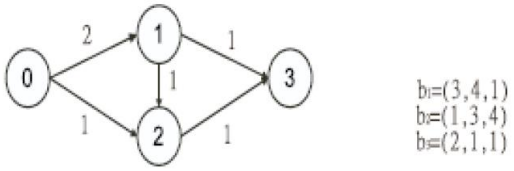


圖四

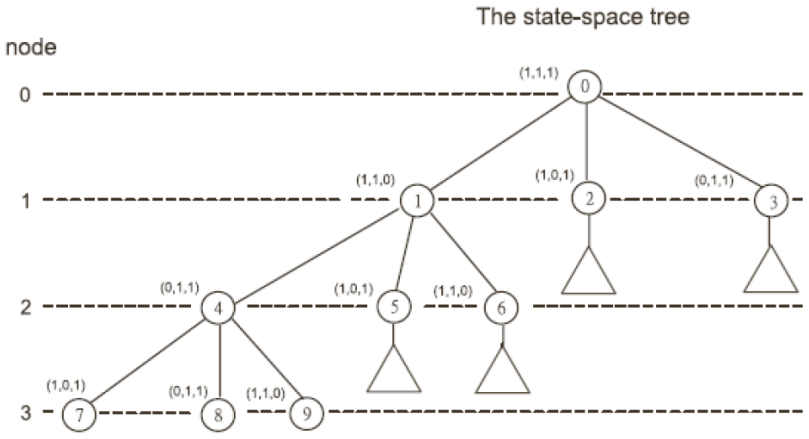
三、 研究方法

➔ 假設網路拓樸圖為有向無環圖 (DAG)，我們設計 branch-and-bound 演算法來求解 MOD 伺服器之路由最佳化問題。

• branch-and-bound 演算法



圖五



圖六

1. 分支方法 (branching method)：

以圖五為例，圖六為根據圖五所畫出之 state-space tree。假設在節點 0 有 3 種影片，所以 state-space tree 樹根上的向量(1,1,1)代表此節點的所有接收情形。接下來考慮節點 1 的所有接收情形，因為節點 1 的上游只有節點 0，且只有 1 條連接邊進入節點 1、頻寬$r_{01} = 2$，因此節點 1 的所有接收情形為 (1,1,0)、(1,0,1)、(0,1,1) 三種，其下游則繼續依此類推，最後可展開成圖六之 state-space tree。

2. 限制方法 (bounding rule)

按照上面的分支方法，每次在分支前，對於下游那些還未決定接收情況的節點，可以採取貪婪的方式估計，即假設該節點之上游可提供所有影片，來估計出該分支之 upper bound。因此，結合分支方法，branch-and-bound 演算法在運行時，在每次分支之前，會先估算此一分支的 upper bound 是否比目前找到的暫時最佳解還好，如果成立即代表此分支有機會找到更好的解，可以繼續分支下去；若不成立，則代表此分支沒有發展的潛力，不必繼續分支。

四、 資料生成

➔ 本專題以隨機方式去產生有階層性的有向無環圖，並得出相關影片數、頻寬...等設定

1. 連結邊：

我們使用的網路節點分布為隨機產生，對於每個節點 i 從 1,2,…,i 隨機挑一數x，代表有多少連結邊進入該節點，並且從該節點的所有上游隨機挑選x個節點來當作連結邊的頭端。

```
node = n
video_number = m

input_number = [0 for i in range(node-1)] # 不包括server

for i in range(len(input_number)):
    input_number[i] = random.randint(1,i+1) # ex:第3個節點的進入點可能有1~3個

node_t = []
node_r = []
upstream = []
for i in range(len(input_number)):
    upstream.append(np.random.choice(range(i+1), input_number[i], False)) # 分別決定每個節點(不包括server)有哪些上游節點

for i in range(node-1):
    for j in range(len(upstream)):
        upstream[j] = list(upstream[j])
        if upstream[j].count(i) > 0:
            node_t.append(i)
            node_r.append(j+1)
        else:
            continue
```

2. 頻寬

假設每條連結邊從 1,…,m-1 隨機挑一數來當作頻寬，m 為影片數量。

```
b = [0 for _ in range(len(node_r))]
for i in range(len(node_r)):
    b[i] = random.randint(1, video_number-1)
```

3. 需求人數

每個節點對於每部影片的需求數量假設為 0,…,20 隨機挑一數。

```
video = [[0 for _ in range(len(node_r))] for _ in range(video_number)]
for i in range(video_number):
    for j in range(len(node_r)):
        video[i][j] = random.randint(0,20)
```

五、 結果與討論

- 結果：針對論文中提供的範例求解最佳路徑

1. 利用 pulp 求解線性規劃之結果：

- The numbers of the users that be served：
```
Status: Optimal
The solution  = 23.0
```
- 群播樹：

video1

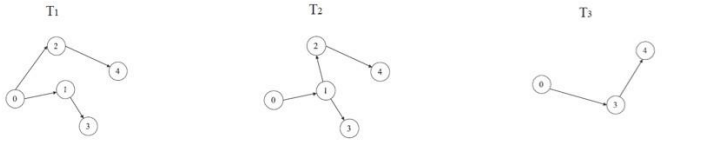
```
video1_e01 = 1.0
video1_e02 = 1.0
video1_e03 = 0.0
video1_e12 = 0.0
video1_e13 = 1.0
video1_e24 = 1.0
video1_e34 = 0.0
```

video2

```
video2_e01 = 1.0
video2_e02 = 0.0
video2_e03 = 0.0
video2_e12 = 1.0
video2_e13 = 1.0
video2_e24 = 1.0
video2_e34 = 0.0
```

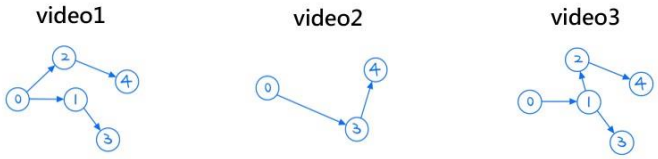
video3

```
video3_e01 = 0.0
video3_e02 = 0.0
video3_e03 = 1.0
video3_e12 = 0.0
video3_e13 = 0.0
video3_e24 = 0.0
video3_e34 = 1.0
```



2. 利用 branch-and-bound 演算法求解之結果：

- Result：
```
最佳提供用戶數量： 23
最佳路徑為: [[1, 0, 1], [1, 0, 1], [1, 1, 1], [1, 1, 1]]
```
- 群播樹：

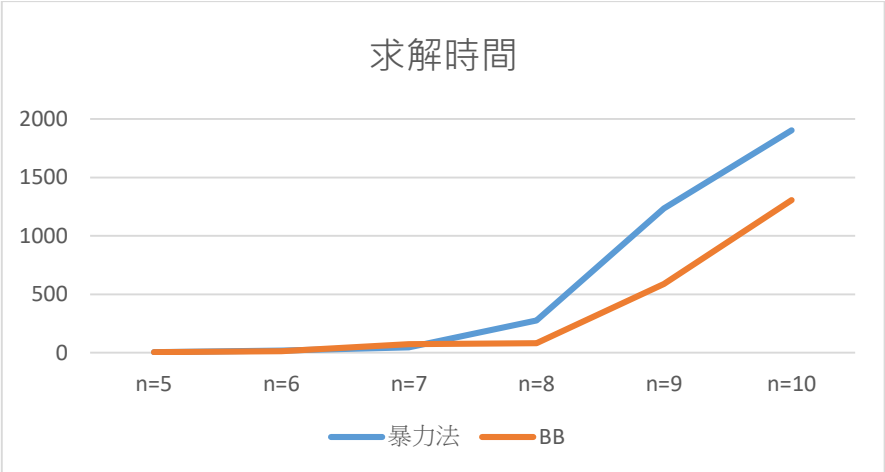


- 效能分析：

1. 根據不同求解方式得出之求解時間：

暴力法	n = 5	n = 6	n = 7	n = 8	n = 9	n = 10
求解時間(s)	4.45	17.90	45.70	275.07	1,235.29	1,903.29

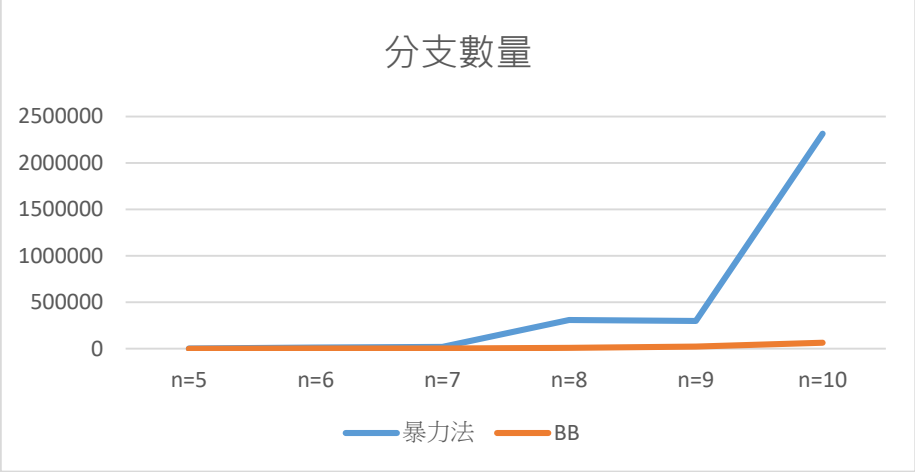
bb	n = 5	n = 6	n = 7	n = 8	n = 9	n = 10
求解時間(s)	3.06	11.05	42.05	81.1	586.41	1,305



2. 根據不同求解方式得出之分支數量：

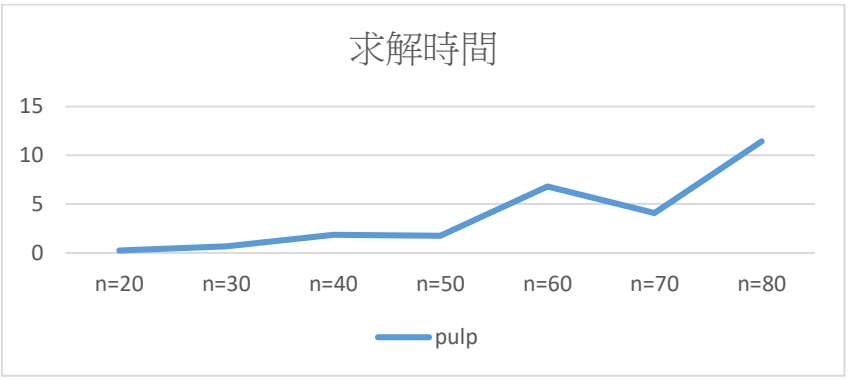
暴力法	n = 5	n = 6	n = 7	n = 8	n = 9	n = 10
分支數	1,557	13,168	20,517	310,176	298,404	2,315,880

BB	n = 5	n = 6	n = 7	n = 8	n = 9	n = 10
分支數	174	1,266	3,058	8,895	24,394	66,124



3. 利用 pulp 在不同情形下的求解時間：

pulp	n = 20	n = 30	n = 40	n = 50	n = 60	n = 70	n = 80
求解時間(s)	0.25	0.68	1.85	1.75	6.79	4.07	11.42



六、 結論

本專題透過 3 種不同演算方法去求解路由最佳化問題，透過效能分析可以發現當節點數越多所花費時間也越大相對的分支數量也會隨之增長，而主要實現的 branch-and-bound 演算法與暴力法求解的分支數量也可以看出該演算法節省了非常多的搜索數量，但當節點數量達到 9 時，求解時間仍然開始大幅度增長，所以當節點數再更進一步增加時，branch-and-bound 演算法就不再適合求解此種問題，可能要使用基因演算法或是禁忌搜尋法等方式來求解。

七、 參考資料

王朱福、林政賢。MOD(Multimedia On Demand)系統中資料繞送之最佳化策略研究。國立屏東教育大學資訊科學系