第二十屆旺宏科學獎

創意說明書

參賽編號：SA20-442

作品名稱：由立體思維解類漢米爾頓路徑問題—以教師介聘為例

姓　　名：林晉湧

關 鍵 字：漢米爾頓路徑、教師介聘

壹、 摘要

本研究旨在應用立體思維解決教師介聘作業，提出擁有品質保證之方法，可求得介聘成功人數之區間。此法除了可使媒合數量最大化外，進而由原模型衍伸出多種策略，可以透過調整參數並於結果與時間中取得平衡。單志願介聘中，透過使用不同模型使**準確率**（介聘成功人數/最多成功人數）介於 88～100%，運算時間與準確率成正相關。多志願介聘則以自訂規則作為範例，套用單志願介聘模型來呈現效果。

教師介聘為學校間之教師調換作業，透過志願選填與其他參與者進行交換。以 110 介聘規則而言，介聘順序為單調→五角調→四角調→三角調→互調，相同者以積分高為優先。現有制度受限於作業期程、業務人員能力，約略簡化問題原型，但即使如此，介聘處理的結果仍不提供數據分析，導致無從分析其品質及過程，因此介聘的結果、數量和方法皆仍有很大的研究空間。

本研究的方法靈感來自俄羅斯方塊的啟發：一塊積木代表一位介聘成功的教師，橫軸位置代表學校間的關係，但目的並非消除方塊，而是將方塊們疊成最接近的指定形狀，即可得最佳解。

經過實驗結果，n、rn、r 會大幅影響時間複雜度（n 為學校數、rn 為任兩校間的最大介聘人數、r 為兩學校間共有幾條連結）。測試資料大多落於$5 \leq n \leq 30$、$5 \leq rn \leq 15$、$r \cong -0.03n^2 + 5.3n - 16$、總介聘人數$\cong rn \times r \times 0.5$。原模型於 $n \leq 8$、$rn \leq 10$、$r \leq 25$ 時，產生的迴路數量約≤ 90，可在一天內執行完。而每個模型因為時間複雜度的區間過大，因此分組進行比較。

貳、 研究動機

教師介聘至今仍缺乏有效率的解法。據目前蒐集的資料、詢問學校人事人員，其流程僅能由各校申請填報並等待官方公布介聘結果，並由於個資法的保護，無法取得原始報名資料，以致於無法驗證其過程，介聘後教師於社群媒體抱怨時有所聞。因此決定嘗試挑戰此難題，突破目前的侷限。

參、 研究目的

一、運用立體思維建立可找出最佳解的模型。
二、使用剪枝法來加速時間並試著逼近最佳解。
三、可藉由調整參數，由時間與解中取得平衡。
四、將單志願模型套入自訂之多志願模型中作為範例。

肆、　研究器材

一、Dev C++（程式主要撰寫軟體）

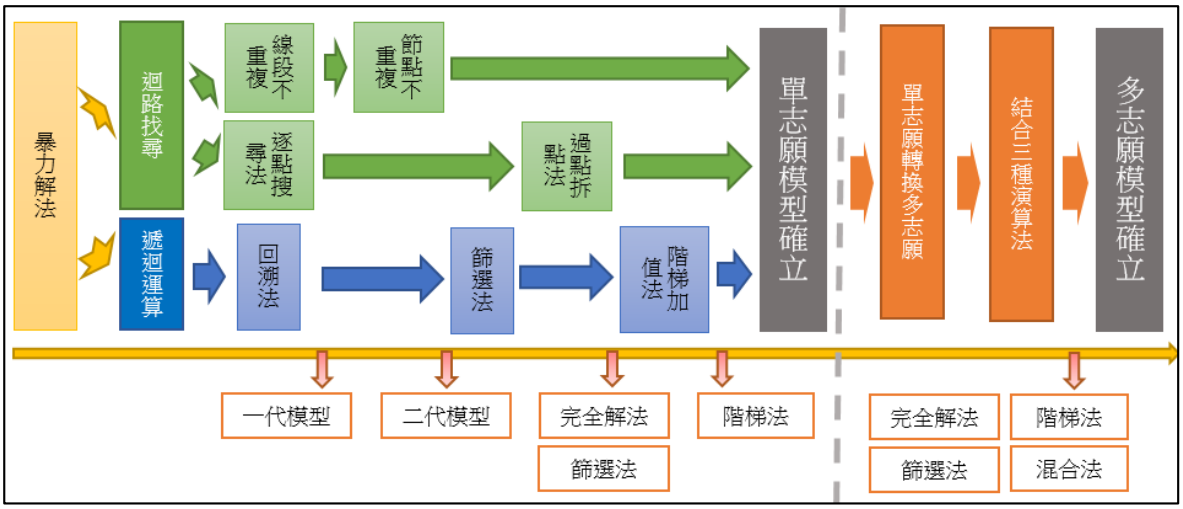
二、Neo4j_Java script（平面有向圖繪製軟體）

三、onshape（立體圖繪製軟體）

四、電腦—intel i5-8265U CPU（程式資料運算裝置）

伍、　研究方法與步驟

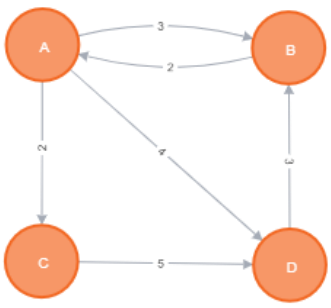
▼圖一、研究歷程



一、問題敘述

教師介聘圖形化如下：有 n 間學校參與介聘，每間學校均有老師想轉至其他學校，若某校共轉出 m 個教師，則某校必須從其他校共轉進 m 位教師，達成**每校之教師數量總和不變**。為了方便觀察，以有向圖作呈現。

▼圖二、例圖



1. 以節點作為學校，箭頭表示幾位教師想介聘，如 A→B＝3，指的是 A 校有 3 位教師想去 B 校，以此類推。
2. 若 A 校最後共轉出 2 位教師，則 A 校必須從其他校共轉進 2 位教師，其他校都必須遵守此原則。
3. 以介聘成功之人數為解。

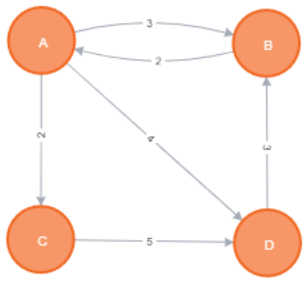
二、名詞與符號定義

1. 最佳解：單志願介聘圖中可得的最多成功人數。
2. 最大解：一演算法由單志願介聘圖可得的最大成功人數。
3. 準確率：最大解／最佳解＊100%
4. 迴路：若未特別註記，泛稱迴路（circuit）與環（cycle）。
5. n：參與介聘的學校數。
6. rn：任兩校間的介聘人數。
7. r：兩學校間共有幾條連結。
8. filter：由 5 種過濾條件排列組合而成。

三、圖形輸入與輸出

為了將圖形輸入程式，採用相鄰矩陣的資料結構儲存。以圖三為例，左列對至上行表示箭頭方向，如黃色格子為 B→A=2。而若兩點不通或相同，則用-1 表示。

▼圖三



▼表一

	A	B	C	D
A	-1	3	2	4
B	2	-1	-1	-1
C	-1	-1	-1	5
D	-1	3	-1	-1

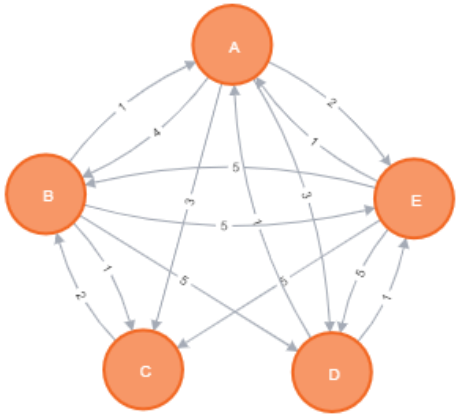
反之，為了將矩陣資料圖形化，使用圖形資料庫表達軟體－neo4j 繪出有向圖：程式計算出結果後，產生相容於 neo4j 的 Javascript 語法，匯入 neo4j 網站後繪出圖形。

▼圖四、矩陣與 C++轉換輸出

```
array:
-1  4  3  3  2
 1 -1  1  5  5
-1  2 -1 -1 -1
 1 -1 -1 -1  1
 1  5  5  5 -1

javascript:
MATCH (n) DETACH DELETE n
CREATE (a:Person { name: "A" }),(b:Person { name: "B"
}),(c:Person { name: "C"}),(d:Person { name: "D"}),(e
:Person { name: "E"}),(a)-[:KNOWS{m:4}]->(b),(a)-[:KN
OWS{m:3}]->(c),(a)-[:KNOWS{m:3}]->(d),(a)-[:KNOWS{m:2
}]->(e),(b)-[:KNOWS{m:1}]->(a),(b)-[:KNOWS{m:1}]->(c)
,(b)-[:KNOWS{m:5}]->(d),(b)-[:KNOWS{m:5}]->(e),(c)-[:
KNOWS{m:2}]->(b),(d)-[:KNOWS{m:1}]->(a),(d)-[:KNOWS{m
:1}]->(e),(e)-[:KNOWS{m:1}]->(a),(e)-[:KNOWS{m:5}]->(
b),(e)-[:KNOWS{m:5}]->(c),(e)-[:KNOWS{m:5}]->(d)
MATCH (all:Person)-[:KNOWS]-(friends)
RETURN all,friends
```

▼圖五、轉換後的程式碼於 javascript 的顯示圖

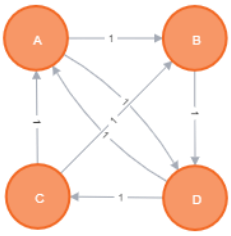


四、立體思維模型

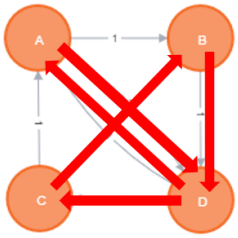
(一) 簡化問題　一人問題

將每條路徑上的介聘數變成 1，試著從中找出想法。對於這個的解法，就是找出一個**通過最多線段的迴路（circuit）**，有點類似尤拉路徑，一筆畫出最長 circuit。（以 n=4 為例）

▼圖六、例圖



▼圖七、解法之一



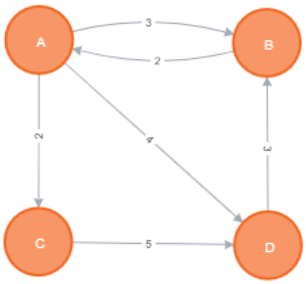
嘗試了多種方式，產生了一個新想法：那如果我把圖中所有的迴路排列組合疊起來，則原問題定能得出最多成功人數！

疊加迴路必得最佳解證——反推法：假設已得最佳解之介聘圖，若要使 1 人介聘，則另 1 人必牽連，以此類推=>隨意由已得最佳解之介聘圖 其中取 1 人，則必能與其他人形成迴路，故可知最佳解之介聘圖可由迴路疊加得到。

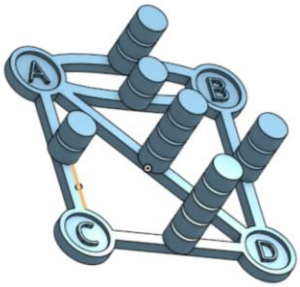
(二) 立體思維　多人問題

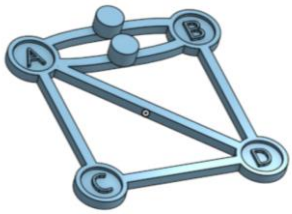
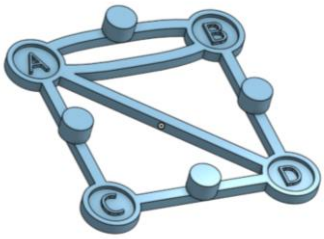
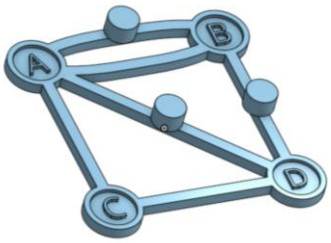
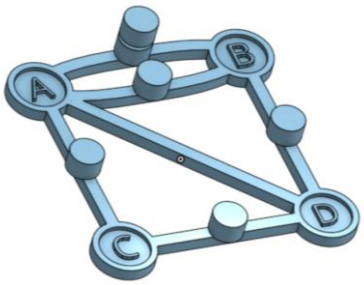
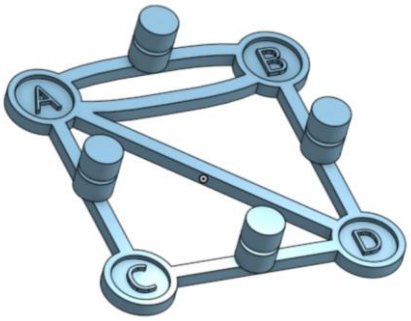
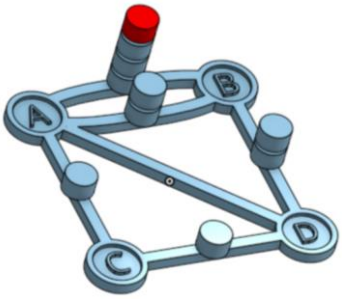
試著把原問題的路徑當作高塔，數值即為層數。如果採用一人問題的解法，只是將很多個迴路（介聘數=1）疊起來，且若層數已達則不得再增蓋，那完全解即最後成功蓋起之總層數的最大值！（以 n=4 為例）

▼圖八、例圖



▼圖九、轉為立體



<table>
<tr><td>先找出圖八所有迴路，迴路之介聘數均為 1，逐點向其他節點進行擴散，並記下得：
（分別命名為迴路 1.2.3….）
L1. A→B→A（圖十）
L2. A→C→D→B→A（圖十一）
L3. A→D→B→A（圖十二）</td><td>▼圖十、迴路 1
  </td></tr>
<tr><td>▼圖十一、迴路 2
  </td><td>▼圖十二、迴路 3
  </td></tr>
<tr><td>接著將得到的迴路疊起來，以不超過某路徑最大值為限。例子：
1. 迴路 1+2（圖十三）。即為其中一解=6（非最佳）。
2. 迴路 2+2（圖十四）。迴路可重複疊加，且恰為最佳解=8。
3. 迴路 1+1+2+3（圖十五）。因 B→A 過量因此不成立。</td><td>▼圖十三、迴路 1+2
  </td></tr>
<tr><td>▼圖十四、迴路 2+2
  </td><td>▼圖十五、迴路 1+1+2+3
  </td></tr>
</table>

(三) 基本模型架構（即一代模型）

基本模型已定→**先迴路**、**再遞迴**，先把圖中之所有迴路找出，接著把迴路遞迴疊加出最佳解。

五、模型改良

鑒於運算時間過慢，決定試著改良模型。分為**最佳解算法加速**及**逼近最佳解算法加速**。

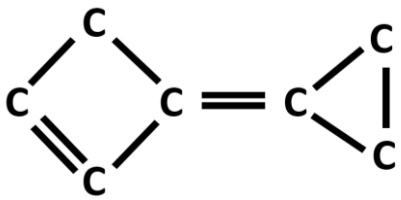
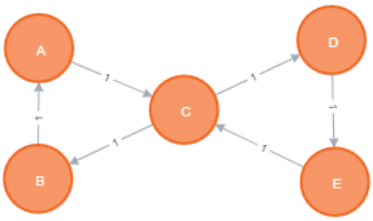
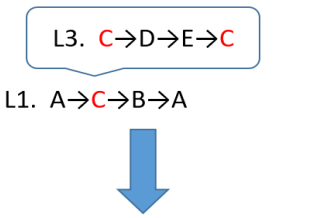
(一) 最佳解算法加速

由原本之模型進行精進，以不影響最佳解為目的。

1. 二代模型__迴路改良一

某次於學習有機化學的烴類時，要算出碳的總價數以求要加幾個氫原子完成化學式（圖十六）；將得到的所有最佳解列出時，解有大量重複的狀況。就此得到靈感：

圖中的所有迴路中，包含 circuit 和 cycle，而 circuit 其實可以用多個 cycle 組成，遞迴時將發生重複計算的狀況。因此藉由去除 circuit 使迴圈數下降，進而遞迴加快。此一舉使研究由類尤拉路徑轉變成類漢米爾頓路徑。

<table>
<tr><td>▼圖十六、烴類
  </td><td>▼圖十七、例圖
  </td></tr>
<tr><td>圖十七中共有 3 個迴路：
L1. A→C→B→A
L2. A→C→D→E→C→B→A
L3. C→D→E→C
其中迴路 2 即迴路 1+3，本來 3 個迴路刪減為 2 個（圖十八）</td><td>▼圖十八、迴路 1.3 合併
  </td></tr>
<tr><td colspan="2">由圖十八可推得，若一個 circuit 通過某點 n≧2 次以上的，則必可拆成 n 個 cycle，如此可知迴路的找法是以**節點不重複**為規則。</td></tr>
</table>

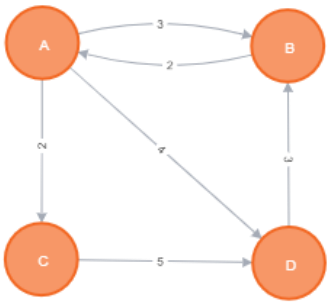
▼表二、迴路改良一__新舊方法總結

之前的方法	現在的方法
以路徑不重複尋找迴路(circuit)	以節點不重複尋找迴路(cycle)

2. 完全解法__迴路改良二

由於在進行剪枝法（於後面有詳細介紹）時，發現找迴路速度竟大於遞迴耗時，因此再次進行改良。檢閱程式碼，發現花時間的地方在於**要由每點往外擴散找迴路，過濾重複太耗時間**，因此決定由此下手。我上網查閱了相關資訊，靈感再度來臨。**如果某點已經往外擴散尋找完，那尋找下一點時某點就可以直接排除**。

▼圖十九



由 A 點開始，找尋所有迴路，接續換 B.C.D。由以下比較可知，舊法比新法多了不少多餘的步驟。

▼圖二十

L1 : A→B →A
L2 : A → C→D →B → A
L3 : A →D →B →A
~~L4 : B → A → B~~
~~L5 : B → A → D → B~~
~~L6 : B → A → C → D → B~~
~~L7 : C → D → B → A → C~~
~~L8 : D → B → A → D~~
~~L9 : D → B → A → C → D~~

L1 : A→B →A
L2 : A → C→D →B → A
L3 : A →D →B →A
（A已找完，刪去A）
（B已找完，刪去B ）
（C已找完，刪去C ）

證：假設依序尋找 ABC….，如果 A 點已經尋找完，代表包含所有 A 點之迴路已經找完，那麼在找 B 點時，若搜尋到了 A 點，那此迴路則必重複，故可將 A 點直接排除；則找 C 點時 A.B 點直接排除，以此類推。

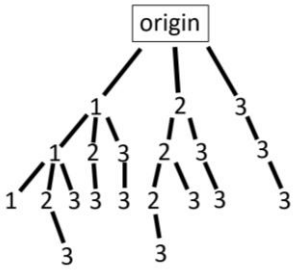
▼表三、迴路改良二__新舊方法總結

之前的方法	現在的方法
以逐點擴散法找迴路	以過點拆點法找迴路

3. 遞迴改良

遞迴的部分並無太多改良，我採回溯寫法，至於空間複雜度與時間複雜度則要看個人的寫法，我採用的方法如圖二十一。

▼圖二十一、遞迴回溯之例	圖中的數字表第幾個迴圈，以 3 個迴路回溯為例。



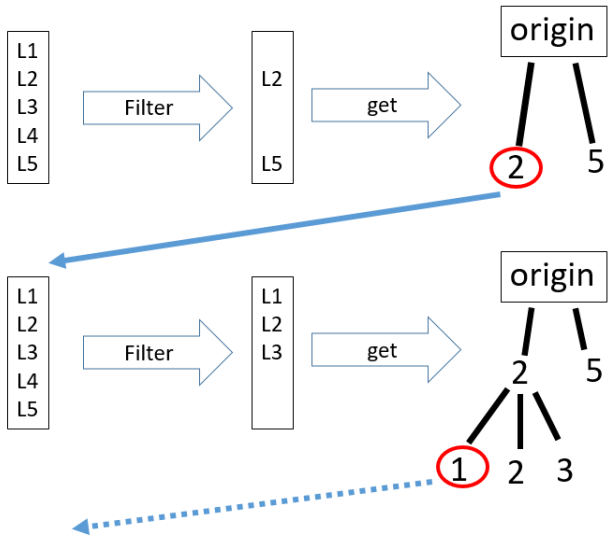
(二) 逼近最佳解算法加速

最佳解的標準模型之速度似乎已經到達極限了，接著要改造模型，雖不會產生最佳解但能有效提速。

1. 遞迴延伸__篩選法（剪枝法）

遞迴中的動作：選迴路→判斷是否可疊並執行，不斷重複這個動作，而我決定下手於選迴路這個階段。以圖二十二為例，**同樣為回溯的方式**，不過於挑迴路時將比較不可能的迴路排除。

▼圖二十二、模型

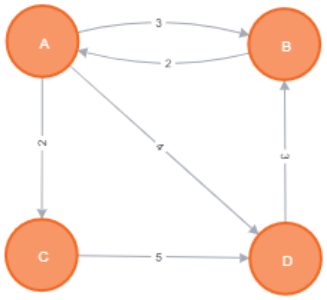


接下來決定篩選方式，我用電腦把圖的迴路算好後，用雙眼與筆試著自己拼拼看，從中不斷嘗試各種方法，得到了一個結論：**挑選順序似乎與迴路於圖中之疊加後剩餘值有相關性**。因此我決定將 filter 用 5 個過濾條件排列組合得之。

▼表四、5 種過濾條件定義

縮寫	過濾條件意義	過濾模式（可能有重複）
min	迴路最小值	取最大
max	迴路最大值	取最大
len	迴路長度	取最大
ave	迴路平均值	取最大
sta	迴路標準差	取最小

▼圖二十三、例圖



以迴路 A→D→B→A 為例

線段	A→D	D→B	B→A
值	4	3	2

min：2

max：4

len：3

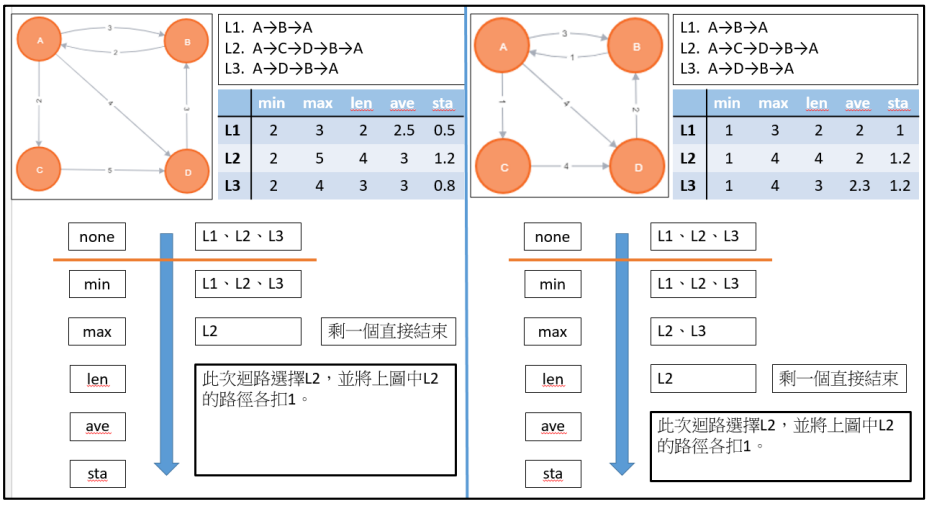
ave：3

sta：0.816496

而 filter 即藉由判斷所有待濾迴路的值決定是否選此迴路。那為何有些取大有些取小？以 min 來說，若此迴路之路徑最小值較大，那這個迴路就可疊比較多次，故先消耗此迴路。

以下為實際操作（filter：min→max→len→ave→sta）：

▼圖二十四、filter 運作的例子



而排列組合的部分可以選或不選與排列順序因此可以得到很多組的 filter。

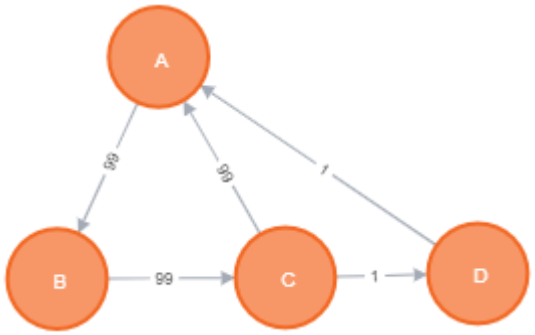
▼表五、filter 的各種模式

挑 5 個(s5)	挑 4 個(s4)	挑 3 個(s3)	挑 2 個(s2)	挑 1 個(s1)
$5! \times C_5^5$	$4! \times C_4^5$	$3! \times C_3^5$	$2! \times C_2^5$	$1! \times C_1^5$

2. 遞迴延伸__階梯法（剪枝法）

為了處理極端介聘圖而進行此改良，以下圖為例。

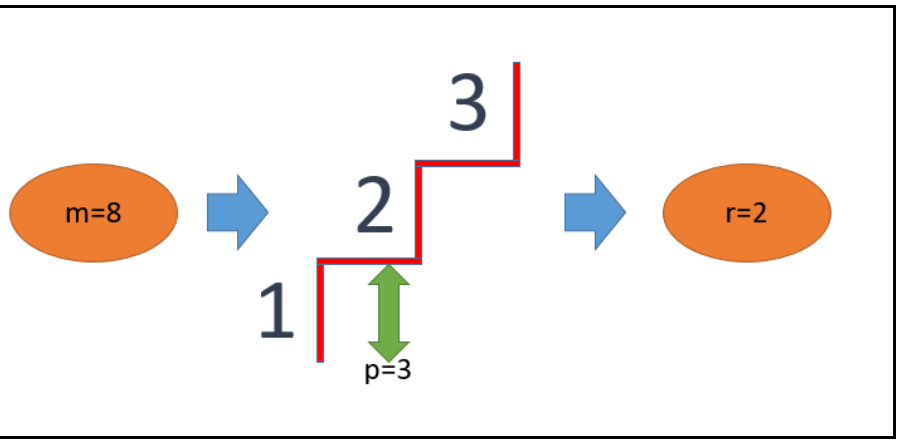
▼圖二十五



如圖如果使用完全解法將造成過多的時間運算，甚至肉眼判斷都比較好。

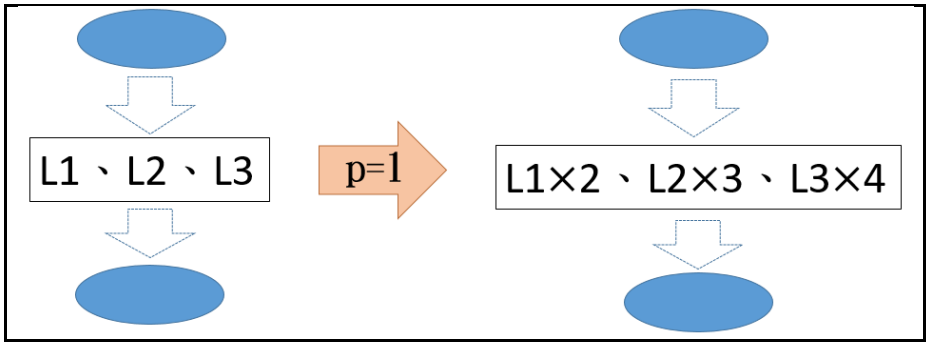
當介聘圖上的某值過大或是過小，運算效率將大幅下降。而階梯加值就宛如稅率，賺的多則要繳的稅就重，提升運算效率。我操作的地方是在遞迴至下一層時，本來迴路上之值都只會-1，但會因為目前此迴路剩餘可加之值而變成-r，而 r 由此迴路上剩餘的最小值 m 與階梯 p 決定：$r = \lfloor m/p \rfloor$，若$r = 0$則定$r = 1$。以下圖為例。

▼圖二十六、$r = \lfloor m/p \rfloor$，若$r = 0$則定$r = 1$

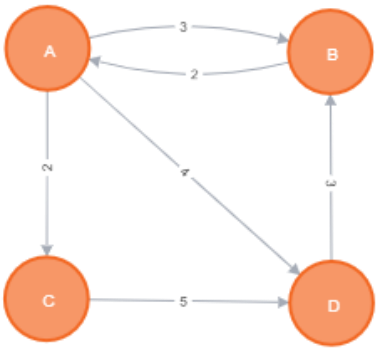
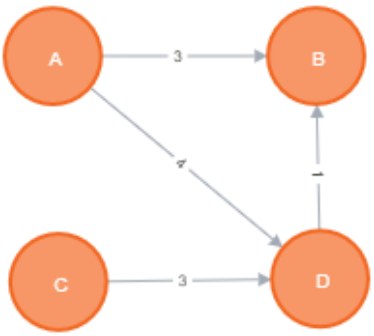


算式的翻譯即為 m 可以爬到第幾層階梯由 p 來控制，而得之解為第 r 層，且 r 至少為 1。下圖為遞迴狀況。

▼圖二十七、遞迴模式



實際操作一次：

▼圖二十八 	要遞迴的迴路為 A→C→D→B→A 且 p=1 而發現迴路中的最小值： A→C = m =2 因此 r=2 此迴路上之值各扣除 2 因此得到下圖
▼圖二十九 	此法會大幅扣除細節的運算，忽略掉細部的+1.-1，但可藉由調整 p 階梯高度來調節精度。

3. 當 p=1 時與 Maximum Flow 的關係

當 p=1 遞迴時，每個迴路均以**不能再填**為目標，與 Maximum Flow 中「找一條可行的路徑並**將其填滿**」的想法有異曲同工之妙，但於本質上仍有些不同。

六、多志願制範例模型

分別延伸了以上的三種單志願演算法：完全解法、篩選法、階梯法，接著再合併三者產生混合法。

(一) 模型規則與內容

此設計可以保障先志願優先順序、後積分高低排序（先志願後積分），但就**不能保證多介聘的最終人數是最多的。**

1. 每人有積分、所在學校、欲介聘學校。
2. 每人得以填最多 n-1 個志願，且志願不得重複。
3. **先志願**：以志願序為基礎，同一志願的人互相進行介聘作業（意即以志願序為分界，將**多志願**切割成**多個單志願**介聘圖進行運算）。
4. **後積分**：志願若相同，則以積分大小為優先順序。
5. 若某人於第 m 個志願介聘成功，則 m+1 以後的志願不納入計算。
6. 得解之取法：以第一志願的最大解為優先，其他志願依序排列，而得到的總合。以下兩表為例，志願序 1.2 的最大解均相同，但**志願序 3 表七大於表六**，於此法下**以表七為基準而淘汰表六**，多志願最最佳解之為 40+10+7+5+3+2+2=69。

▼表六

志願序	1	2	3	4	5	6	7
最大解	40	10	5	4	3	3	2

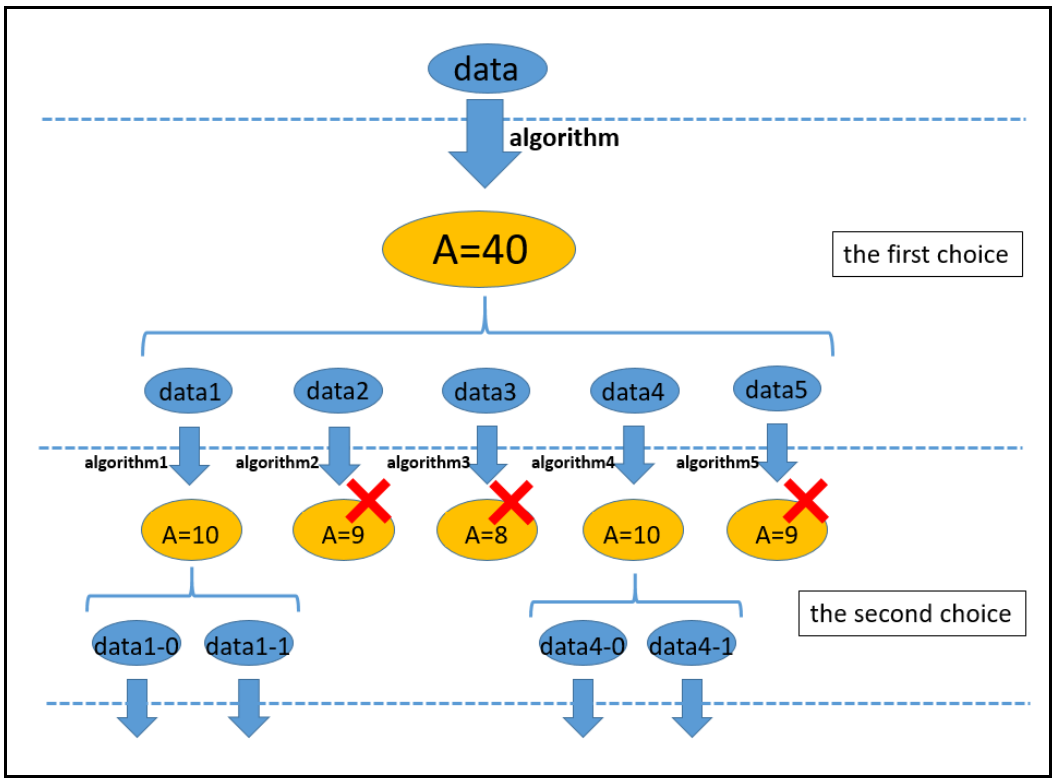
▼表七

志願序	1	2	3	4	5	6	7
最大解	40	10	7	5	3	2	2

(二) 模型架構

由於經由不同演算法得到的最大解可能不只一種，因此再以一個遞迴包住整個架構，以求出多志願的得解。簡而言之，即將多志願問題拆成多個單志願問題去處理。

▼圖三十、多志願範例模型概念



data 為未介聘成功的人的資料。首先由 data 中的第一志願得到單志願介聘圖並經由 algorithm 得到最大解 A=40，共有 5 個最大解。將 data 扣除成功介聘的人後，得到 data1.2.3.4.5，接著重複步驟，分別將其經由 algorithm1.2.3.4.5 以求出最大解。由於定義，data2.3.5 被去除，以此類推第三志願等。

(三) 演算法套用規則

Algorithm 的填法有四種，其中三種是全部填入完全解法、篩選法、階梯法，第四種則是混和以上的混合法。

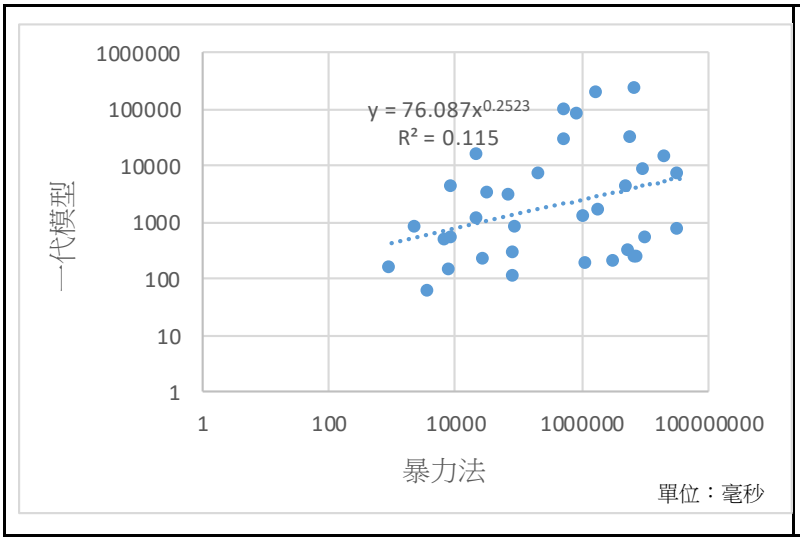
混合法合併以上三種演算法。將由每一個 data 中的**迴路數量**來決定使用哪種演算法（可調參數）：因為介聘數於每條線段的落點均差不多，影響遞迴運算時間較大的為迴路數。而此篇報告是以 45.80 做為區隔：迴路數量若小於等於 45，使用完全解法；介於 46 至 80 則用階梯法；若大於 80 則用篩選法。篩選法則均以 CBA 作為代表（後續會介紹）；階梯法則以 p=1 作為代表。

陸、　實驗結果

一、一代模型與暴力法之比較

進行比較的部分為**總運算時間**與**最大解**。共 35 筆測試資料。

▼圖三十一、一代模型與暴力法之**總運算時間**比較

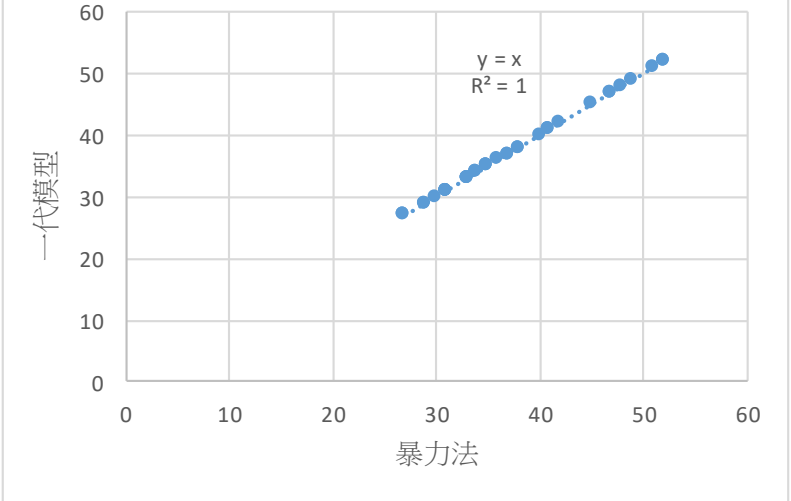


為清楚了解兩軸關係，x.y 軸使用對數刻度表示。

其時間複雜度會受到 n. rn. r 等影響，故不會呈現漂亮的圖形，但趨勢仍明顯顯現。

其最佳趨勢線為 $y = 76.087x^{0.2523}$，為乘冪關係。

▼圖三十二、一代模型與暴力法之**最大解**比較



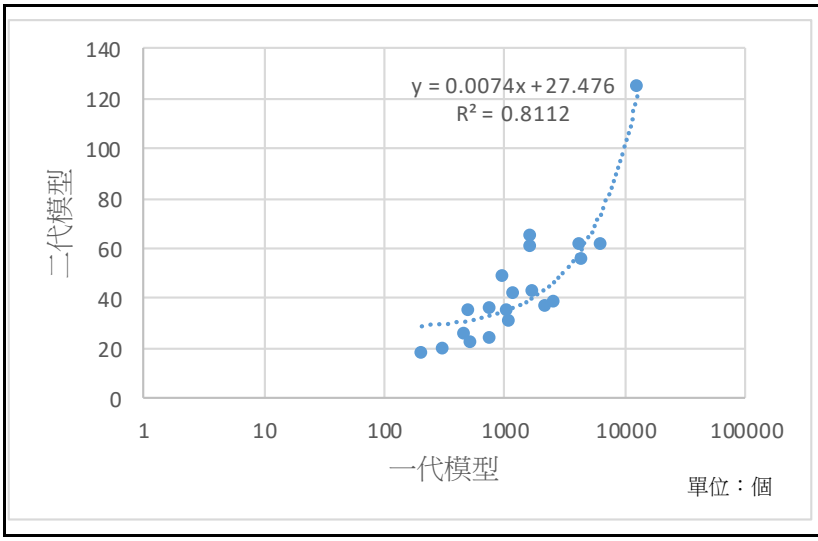
執行了多筆測試資料，答案均 100% 相同，可證明此法有一定可信度。

其最佳趨勢線為$y = x$，為線性關係。

二、二代模型與一代模型之比較

進行比較的部分為**迴路數多寡**與**總運算時間**。 共 20 筆測試資料。

▼圖三十三、二代模型與一代模型之**迴路數**比較

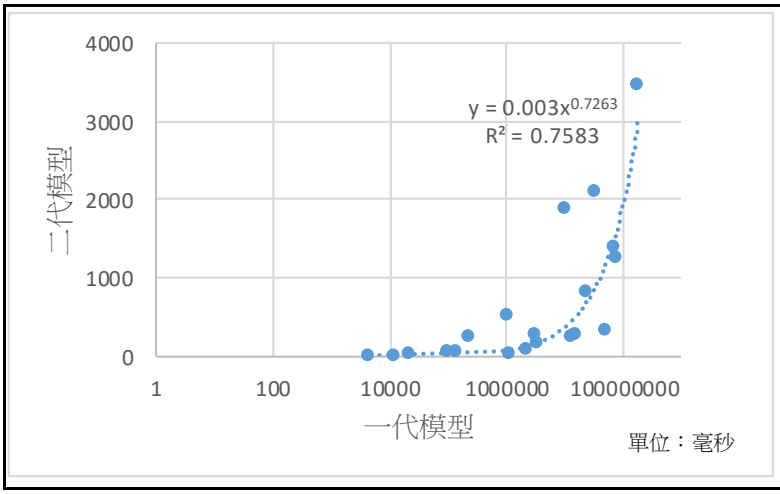


為清楚了解兩軸關係，x 軸以對數刻度表示。

其趨勢有較明顯的顯現。

其最佳趨勢線為$y = 0.0074x + 27.476$，為線性關係。

▼圖三十四、二代模型與一代模型之**總運算時間**比較



為清楚了解兩軸關係，x 軸以對數刻度表示。

其趨勢有較明顯的顯現。

其最佳趨勢線為$y = 0.003x^{0.7263}$，為乘冪關係。

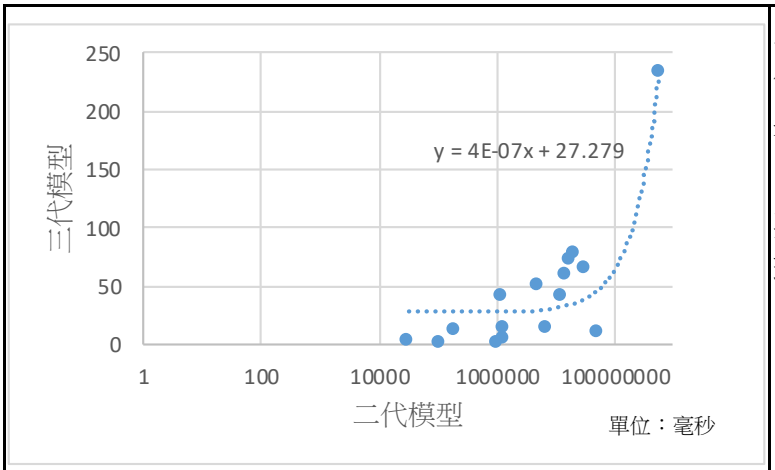
▼表八、小節整理

	一代模型	二代模型
內容	線段不重複	節點不重複
得解	均為最佳解	
迴路數	多	少
找迴路耗時	長	短
遞迴之耗時	長	短

三、三代模型與二代模型之比較

進行比較的部分為**迴路找尋運算時間**(time)。共 16 筆測試資料。

▼圖三十五、三代模型與二代模型之**迴路找尋運算時間**比較



為清楚了解兩軸關係，x 軸以對數刻度表示。

其趨勢有較明顯的顯現。

其最佳趨勢線為$y = 4 \times 10^{-7}x + 27.279$，為線性關係。

▼表九、小節整理

	二代模型	三代模型（完全解法）
內容	逐點擴散法	過點拆點法
得解	均為最佳解	
迴路數	均相同	
找迴路耗時	長	短
遞迴之耗時	均相同	

四、完全解法之虛擬碼及最大時間複雜度估計

此演算法由迴路找尋＋遞迴疊層形成，故分為兩部分處理。虛擬碼之內容請見**附錄一**。

由於此演算法之時間複雜度過於複雜，因此以時間複雜度之**最小**以及**最大**做為代表。而**最小**的時間複雜度一看即知，因此不多做探討。

(一) 能使時間複雜度最大之介聘圖

點與點間越多連結，迴路數大幅上升的可能性就越高，若使迴路數增加、整體平均值增加，那就能使遞迴耗時大幅上升。

以上，推測時間複雜度最大的介聘圖 G，是每個點間都有連結，而兩點間的介聘數都是最高的（$r = n^2 - n$，總介聘人數$= r \times rn$）。

反推，現有另一隨機介聘圖 g，其n. rn之值與 G 相同，則其算得之迴路、遞迴之途經分岔口，G 也是必擁有的，故可確定 G 必為時間

複雜度最大之圖。（此處之介聘圖 G 以下繼續沿用其定義）

(二) 迴路找尋之最大時間複雜度

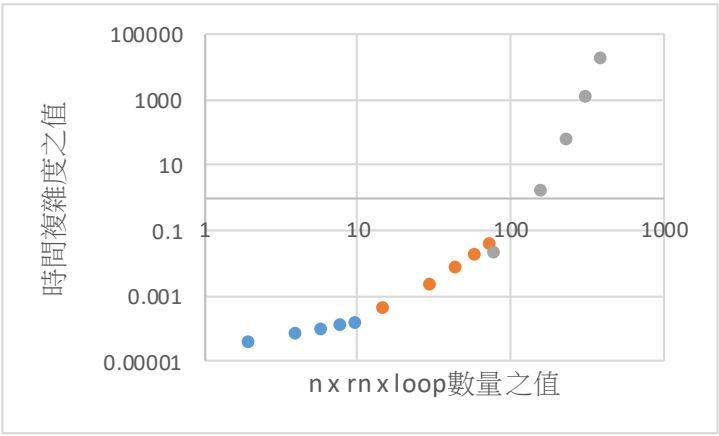
代入介聘圖 G 其最大時間複雜度即為其迴路數量：

O ($\sum_{i=2}^{n} C_i^n \times (i-1)!$)。詳細推理過程請見**附錄二**。

(三) 遞迴疊圖之時間複雜度

由於能力不足，故以圖表呈現。代入介聘圖 G，狀況如下：

▼圖三十六



y 軸是以程式算出之時間複雜度之值，x 軸是 **n × rn × 迴路數** 之值。

藍色：n=2

橘色：n=3

灰色：n=4

詳細推算過程請見**附錄三**。

五、篩選法與完全解法之比較

進行比較的部分為**各過濾條件之所有排列組合的最大解**與**總運算時間**，並選出一套**可調整式**組合。以 A 表示 min 過濾條件，B 表示 max 過濾條件，C 表示 len 過濾條件，D 表示 ave 過濾條件，E 表示 sta 過濾條件。由左至右表過濾順序。共 20 筆測試資料。

(一) 準確率比較

表中的值為 20 筆資料的平均，表示與最佳解的接近程度。

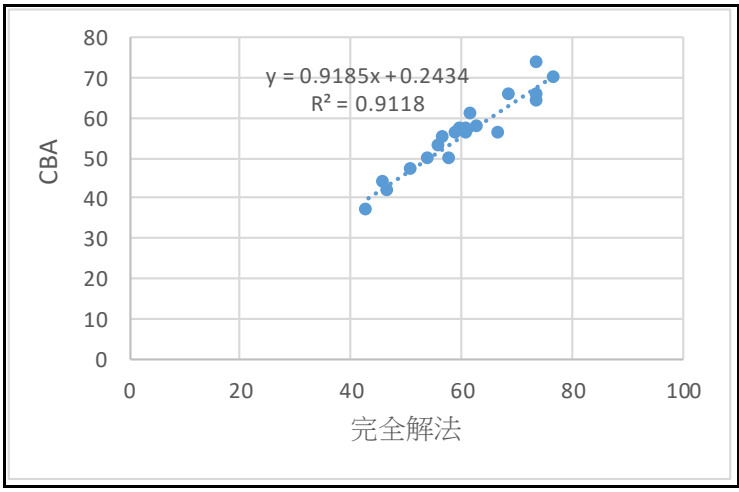
s5, s4,s3 filter 之平均準確率請見**附錄四**。

實驗過程中出了令我意外的問題，s1, s2 filter 的算速超過原方法，因此並不納入計算。之後發現問題所在處為 filter 範圍過廣，導致僅過濾掉些微甚至沒有，而過多的過濾時間造成效能大幅下降。

可以發現 A.C 在開頭的過濾器得到的準確值均偏高，可見的迴路最小值與迴路長度對於遞迴選擇有很大的影響性。

接著我取準確率最高的 CBA 作代表，觀察得解與最佳解的關係。

▼圖三十七、CBA 與完全解法之最大解比較



由圖可得大致為線性關係，故可推得平均準確率的確高達 91%。

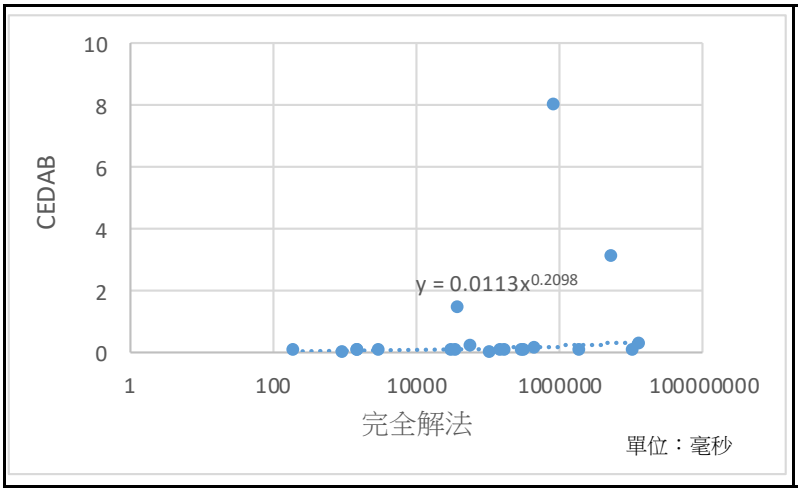
(二) 運算時間

由同一 20 筆測試資料測得之平均值。

s3, s4, s5 filter 之平均總運算時間請見**附錄四**。

可見到完美解與此解的大幅差異，以 C 開頭強佔頭銜。接著同樣取算速最快的 CEDAB 作代表，觀察最佳解與得解之運算時間關係。

▼圖三十八、CEDAB 與完全解法之總運算時間比較



為清楚了解兩軸關係，x 軸以對數刻度表示。

其最佳趨勢線為$y = 0.0113x^{0.2098}$，為乘冪關係。

(三) 最佳可調整式組合

此法之目的是為了使操作者由時間與得解中取得平衡，故將前兩大筆資料平均後經過以下處理得之：

1. 將濾網依照準確率大到小排序。
2. 若有平均準確率相同之濾網，則只保留運算時間最少的。
3. 由頭開始，若第 n+1 個的運算時間大於第 n 個，則去除第 n+1 個。

最後得到以下數據：（完全解法之平均時間為 1723881 ms）

▼表十

	運算時間	準確率			運算時間	準確率
CBA	0.704672	92.23%		BECAD	0.065846	90.91%
ACB	0.123036	92.12%		CBAD	0.040226	90.71%
ACD	0.087555	91.86%		CBADE	0.034689	90.58%
ACBED	0.073246	91.73%		CBDEA	0.034479	90.38%
ACDBE	0.072948	91.60%		CEDAB	0.028473	89.02%

▼表十一、小節整理

	完全解法	篩選法
內容	回溯法	剪枝法
得解	最佳解	低於最佳解
迴路數	均相同	
找迴路耗時	均相同	
遞迴之耗時	長	短

六、階梯法與完全解法比較

進行比較的部分為**最大解**與**總運算時間**。共 14 筆測試資料。

建立於此方法的作用所在，測試資料以此介聘圖之最大值(rn)為基礎區分開來，藉此來探討其中關連性。而階梯值(leap)據其算出的解來範圍性採用。此次的 rn 由 11 至 20、leap 由 1 至 5 測出結果。

▼表十二、階梯法與完全解法之得解之比較

rn	完全	leap=1	leap=2	leap=3	leap=4	leap=5
11	72	72	72	72	72	72
11	88	88	88	88	88	88
11	92	92	92	92	92	92
13	83	83	83	83	83	83
13	69	69	69	69	69	69
13	94	94	94	94	94	94
13	86	86	86	86	86	86
15	97	97	97	97	97	97
15	103	101	102	103	103	103
15	114	113	114	114	114	114
15	74	74	74	74	74	74

20	136	132	135	136	136	136
20	111	111	111	111	111	111
20	115	114	115	115	115	115

▼表十三、階梯法與完全解法之運算時間比較

rn	完全	leap=1	leap=2	leap=3	leap=4	leap=5
11	9185	3.1	1281	4202	10404	13513
11	2804239	4.3305	9885	124997	491922	1420429
11	205147	10.7475	16121	50941	129806	171922
13	45059	1.09	1172	12782	25703	59791
13	167903	18.275	14993	69093	168713	246841
13	1948607	16.48	61003	399083	1372087	2158936
13	3046952	19.92	48910	672683	980487	2558446
15	30890920	5	56759	3085386	8525476	21765574
15	5815700	13.595	121620	1083999	2158673	3924831
15	1378547	0.78	2000	20566	104927	244356
15	28457	1.56	375	2718	10341	19808
20	243889	0.151	218	2640	10700	32221
20	15191274	5.465	41478	1095237	4706909	9008079
20	556139	0.1485	297	2484	12544	33632

表十中算出的準確率相當的高，也如預期：leap 之值若高，則準確率就高，而運算時間也增加。由於設計初衷是為了解決**極端值**與**總圖值過高**，因此於一般圖解上解與時間上的關係並無極為接近的關係，但仍可依據介聘數來推斷 leap 的值應代多少。

▼表十四、小節整理

	完全解法	階梯法
內容	回溯法	剪枝法
得解	最佳解	低於最佳解
迴路數	均相同	
找迴路耗時	均相同	
遞迴之耗時	長	短

七、多志願範例介聘之四種演算法比較

延伸以上之單志願完全解法、篩選法、階梯法。分別有**多志願完全解法(complete)**、**篩選法(filter)**、**階梯法(layer)**、**混合法(mix)** 進行時間與解之比較，共 20 筆測試資料，會顯示的資料有：此演算法之**得解(answer)**、**解的數量(way)**、**耗時(time)**。

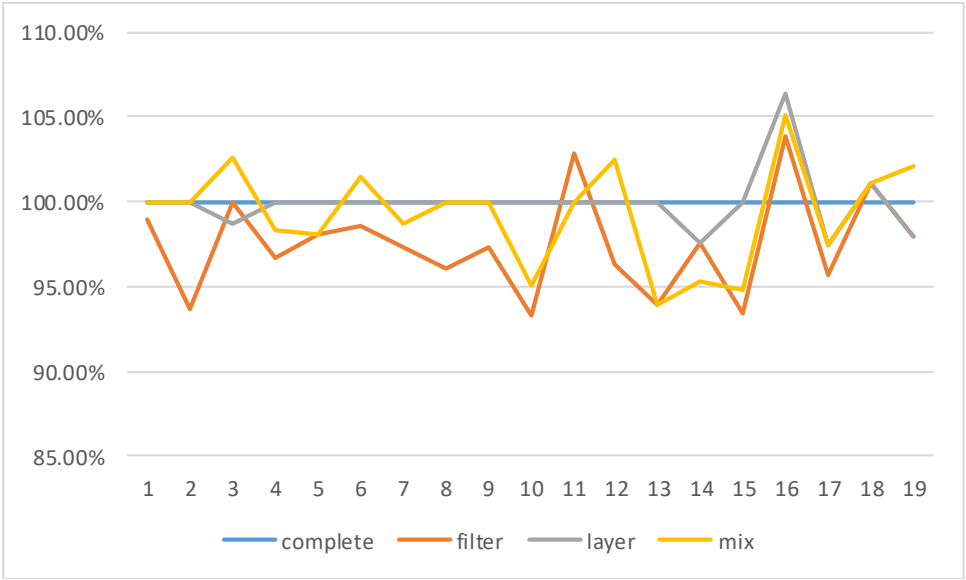
而**暴力法(riot)** 由於速度過慢，因此以一例測試資料作為參考。

▼表十五

	riot	complete	filter	layer	mix
time	2669200	62.1917	0.3572	25.2872	139.3198
answer	51	51	49	50	51
way	1	1	1	1	1

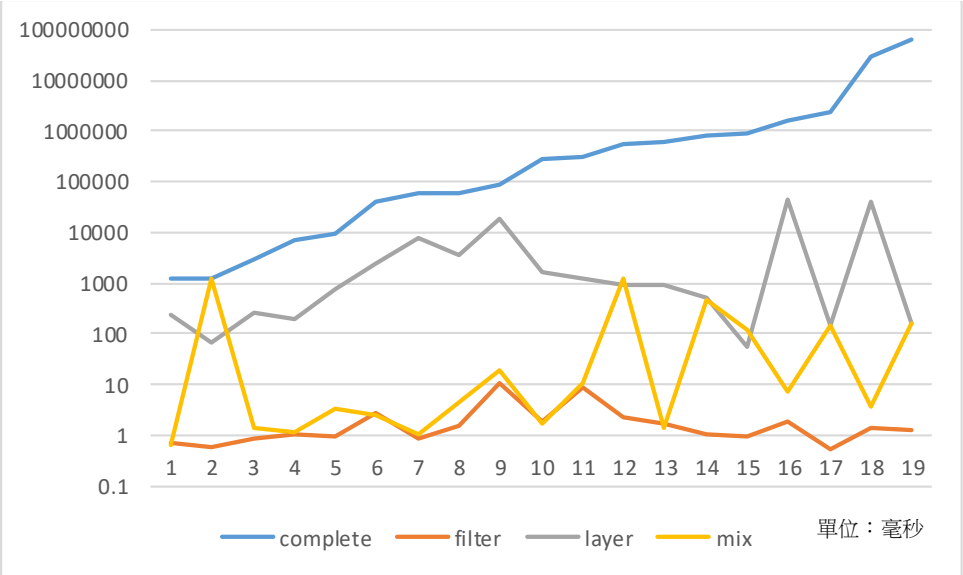
可見得暴力法的效能非常差，且得解經過前面證實與完全解法一模一樣。因此為了避免拖延進度，就不列入之後的比較。

▼圖三十九、四種演算法得解之比較



為方便觀察，將 complete 的得解均訂為 100%。由圖可知 complete 的得解並不會是最大的，layer 最貼近 complete，mix 則徘迴於三者之間。

▼圖四十、四種演算法耗時之比較



整體資料依據 complete 由小至大排序，且 y 軸以對數表示。filter 仍然擁有時間優勢，而 mix 則在三種演算法中擺盪。

▼表十六、四種演算法解的數量之比較

筆數	complete	filter	layer	mix	筆數	complete	filter	layer	mix
1	1	1	1	1	11	1	2	1	2
2	2	1	2	1	12	2	1	2	1
3	2	1	2	1	13	1	1	1	1
4	1	1	2	1	14	2	1	2	1
5	1	1	1	1	15	1	1	1	1
6	1	1	1	1	16	1	1	1	1
7	1	1	1	1	17	1	1	1	1
8	2	1	2	1	18	1	1	1	1
9	1	1	1	1	19	1	1	1	1
10	1	1	1	1	20	1	1	1	1

四種演算法間並無太大的關連性。

柒、 結論

一、利用多個迴路疊加出各種組合的方式可以找出最佳解。

二、單志願模型

(一) 完全解法：

1. 應用處：均可。
2. 迴路以點不重複、過點拆點法，可得最佳效率。
3. 準確率為 100%。
4. 運算時間：暴力法＞完全解法。

(二) 篩選法：

1. 應用處：均可。
2. 使用者可以自行選擇過濾條件，以從解與運算時間中得到平衡
3. CBA 濾網平均擁有高達 92.23%的準確率，且不受其他變數影響。
4. 運算時間：完全解法＞＞篩選法。

(三) 階梯法：

1. 應用處：較適合用於極端值、整體值偏大。
2. 使用者可藉由調整參數（階梯高度）來從解與運算時間中得到平衡。
3. 參數值越大，準確率越高，但相對的運算時間越長。
4. 準確率可極度逼近 100%，運算時間：完全解法＞階梯法。

三、多志願範例模型__先志願後積分

(一) 模擬其中一種狀況，以範例模型演示。

(二) 得解不一定為最多人之解，但於同一志願下必為最多。

(三) 混合法可藉由調整參數決定得解分布與運算時間。

(四) 運算時間：完全解法＞階梯法＞篩選法，混合法摻雜於三演算法中。

捌、 未來展望

一、加入更多測試資料，以找出更精確的關係。

二、設計泛用型工作輪調（例如本例教師介聘）演算法，透過調整參數可適用於各種情境，提供網站頁面供檢視與公評。

玖、 參考資料

一、王修(2010)「應用基因演算法之最佳化教師介聘多角調作業─以台閩地區公立幼稚園教師介聘他縣市為例」。碩士論文，國立高雄師範大學資訊教育研究所。https://hdl.handle.net/11296/3pee7r

附錄

附錄一、完全解法之虛擬碼

▼迴路找尋之虛擬碼

```
LOOP_FINDING(graph)
    loop=[]
    for i=0 to n-1
        SEARCH(i,i,graph,loop)

SEARCH(former,confined,graph,loop)
    if former == confined
        get one loop
        return

    for i=confined to n-1
        if graph[former][i] true and loop not have i
            add i to loop
            SEARCH(i,confined,graph,loop)
            remove i from loop

    there is no loop
    return
```

▼遞迴疊層之虛擬碼

```
LOOP_STACKING(loop_all)
{
    RECURSION(0,0,loop_all)
}

RECURSION(confined,count,loop_all)
{
    for i=confined to loop_all.length-1
        loop=loop_all[i]
        if graph enable stack loop
            RECURSION(i,count+length(loop),loop_all)

    get one answer as count
    return
}
```

附錄二、迴路找尋之最大時間複雜度說明

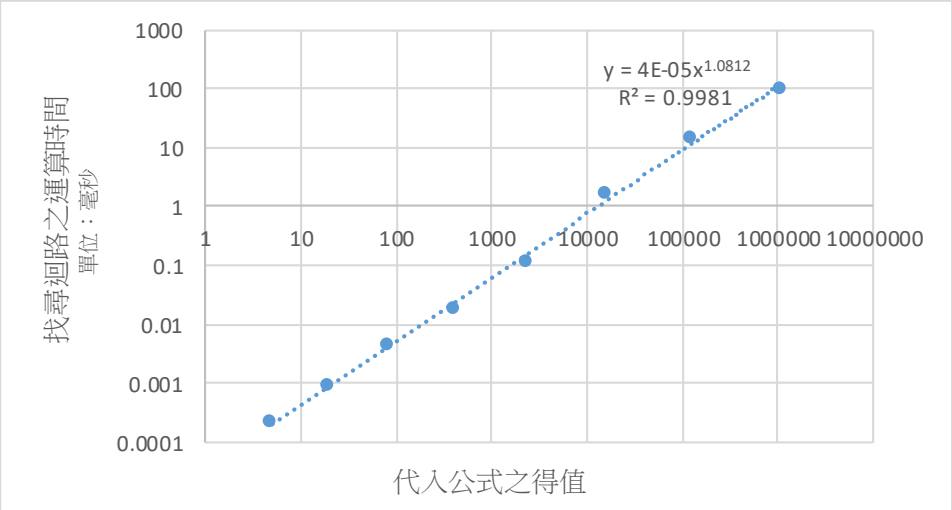
於介聘圖 G 中，每個點都是相通的，故在找到一個點時，即生成一個迴路，而此演算法之設計不會有重複迴路出現→若要算出需要找尋幾次，那算出有幾個迴路即可。

介聘圖 G 之迴路數方程式為 $\sum_{i=2}^{n} C_i^n \times (i-1)!$ 。

推理方法如下：由於介聘圖 G 之特性為各點互通→由 n 個點中取 i 個點來形成迴路，i 個點的排列方法共有 i! 個，但因迴路是環狀，最後還必須再 ÷i，如 A→B→C→D→A＝B→C→D→A→B，而 i 的有效範圍是由 2 個點至 n 個點，故方程式為如此。

除了使用數學推算外，也實際以程式運行作為輔助說明，計算其迴路找尋之時間。

▼圖四十一、迴路找尋之程式運算時間與推得之時間複雜度的關係



$R^2 = 0.9981$，x 之次方項為 1.0812，可見得兩者接近正比關係。藉此可二次驗證上述推算之正確度。

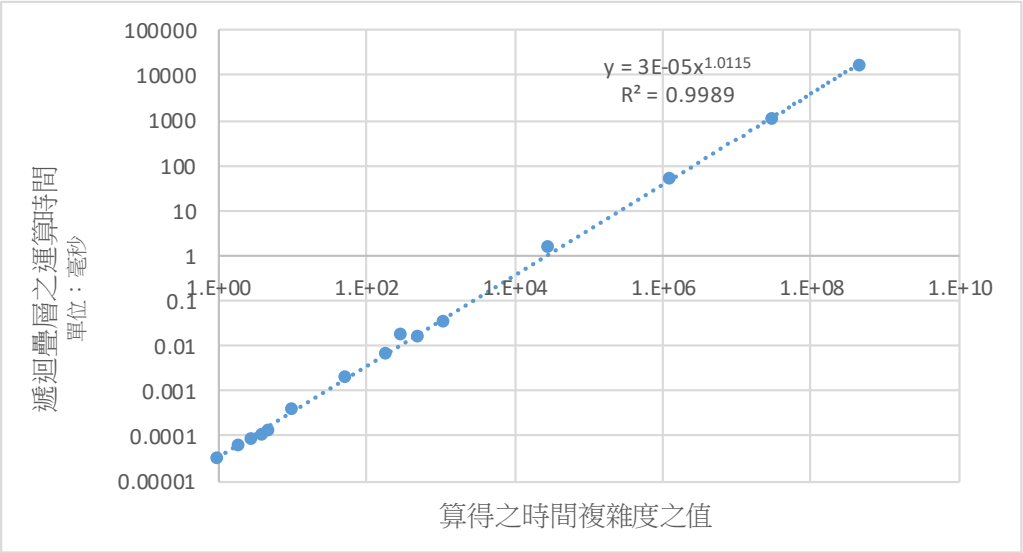
附錄三、遞迴疊層之最大時間複雜度

由於本人的能力不足，因此只能以程式得出運算時間以表現關係圖。

代入介聘圖 G，變因只剩 n.rn 來控制運算時間。N 除了點數，在介聘圖 G 中也控制了迴路數，因此多加入**迴路數**作為因子。

先以程式算出運算時間以及遞迴節點數（即時間複雜度之值），找出兩者的關係。

▼圖四十二、遞迴疊層之程式運算時間與算得之時間複雜度的關係



$R^2 = 0.9989$，x 之次方項為 1.0115，可見得兩者接近正比關係。因此以時間複雜度之值與 n.rn.迴路數進行對比。

嘗試多次拼湊後，以 **n × rn × loop數** 的效果較為規律，因此以其作為 x 軸來呈現圖表。

附錄四、篩選法之測試資料結果

▼表十七、篩選法各種組合之準確率（得解÷最佳解× 100%）

CBA	92.23%	BAED	90.41%	DBC	90.05%	BDCAE	88.48%
CAB	92.14%	BAEDC	90.41%	DBCA	90.05%	BDACE	88.48%
ACB	92.12%	CABDE	90.41%	DBCE	90.05%	BADC	88.39%
ABC	92.08%	CBDE	90.38%	DBCEA	90.05%	BADCE	88.39%
ACD	91.86%	CBDEA	90.38%	DBCAE	90.05%	AEC	86.75%
BCA	91.80%	CBDAE	90.38%	BACD	90.00%	AECB	86.75%
ACBE	91.73%	BCD	90.38%	BACDE	90.00%	AEBC	86.75%
ACBED	91.73%	DEC	90.37%	DABCE	89.98%	AEDC	86.75%
ACDE	91.71%	DECA	90.37%	DCB	89.89%	AECD	86.75%
ACDEB	91.71%	DAEC	90.37%	DCBA	89.89%	AEDCB	86.75%
BAC	91.64%	DEAC	90.37%	DCBE	89.89%	AECDB	86.75%
ACDB	91.60%	DECB	90.37%	DCBEA	89.89%	AEBDC	86.75%
ACDBE	91.60%	DEBC	90.37%	DCBAE	89.89%	AEDBC	86.75%
ACE	91.53%	DECBA	90.37%	BCADE	89.86%	AECBD	86.75%
ACEB	91.53%	DEBCA	90.37%	BCDE	89.82%	AEBCD	86.75%
ACED	91.53%	DAECB	90.37%	BCDEA	89.82%	EBA	85.88%
ACEDB	91.53%	DEACB	90.37%	BCDAE	89.82%	EAB	85.88%

ACEBD	91.53%	DECAB	90.37%	BCE	89.70%	EAC	85.88%
ABCD	91.46%	DEBAC	90.37%	BCEA	89.70%	EDA	85.88%
ABCDE	91.46%	DAEBC	90.37%	BCAE	89.70%	EAD	85.88%
ACBD	91.24%	DEABC	90.37%	BCED	89.70%	EBC	85.88%
ACBDE	91.24%	BCAD	90.36%	BCEDA	89.70%	EDB	85.88%
ABE	91.22%	ADCB	90.35%	BCAED	89.70%	EBD	85.88%
ABEC	91.15%	ADBC	90.35%	BCEAD	89.70%	EDC	85.88%
ABCE	90.92%	ADCBE	90.35%	ABDC	89.58%	EBCA	85.88%
ABCED	90.92%	ADBCE	90.35%	ABDCE	89.58%	EACB	85.88%
ABED	90.92%	DACE	90.33%	CBE	89.35%	EBAC	85.88%
BEA	90.91%	DACEB	90.33%	CBEA	89.35%	EABC	85.88%
BEC	90.91%	BCDA	90.31%	CBAE	89.35%	EDBA	85.88%
BECA	90.91%	DABEC	90.31%	CBED	89.35%	EBDA	85.88%
BEAC	90.91%	DACB	90.28%	CBEDA	89.35%	EADB	85.88%
BECD	90.91%	DCA	90.27%	CBAED	89.35%	EDAB	85.88%
BECDA	90.91%	DCE	90.27%	CBEAD	89.35%	EBAD	85.88%
BECAD	90.91%	DCEA	90.27%	CABE	89.17%	EABD	85.88%
BEACD	90.91%	DCAE	90.27%	CABED	89.17%	EDCA	85.88%
DBA	90.87%	DCEB	90.27%	BDA	89.15%	EADC	85.88%
ABEDC	90.85%	DCEBA	90.27%	BAD	89.06%	EDAC	85.88%
ABECD	90.85%	DCAEB	90.27%	AEB	89.05%	EACD	85.88%
BED	90.82%	DCEAB	90.27%	AED	89.05%	EDCB	85.88%
BEDA	90.82%	ADE	90.26%	AEDB	89.05%	EBDC	85.88%
BEAD	90.82%	ADEB	90.26%	AEBD	89.05%	EDBC	85.88%
BEDC	90.82%	ADEC	90.26%	CEA	89.02%	EBCD	85.88%
BEDCA	90.82%	ADECB	90.26%	CEB	89.02%	EDCBA	85.88%
BEADC	90.82%	ADEBC	90.26%	CED	89.02%	EBDCA	85.88%
BEDAC	90.82%	ABD	90.16%	CEBA	89.02%	EDBCA	85.88%
DAB	90.80%	ABDE	90.16%	CEAB	89.02%	EBCDA	85.88%
CBAD	90.71%	CDA	90.16%	CEDA	89.02%	EADCB	85.88%
ADB	90.68%	CAD	90.16%	CEAD	89.02%	EDACB	85.88%
DBE	90.64%	CDE	90.16%	CEDB	89.02%	EACDB	85.88%
DBEA	90.64%	CADB	90.16%	CEBD	89.02%	EDCAB	85.88%
DBAE	90.64%	CDEA	90.16%	CEDBA	89.02%	EBADC	85.88%
ADBE	90.59%	CDAE	90.16%	CEBDA	89.02%	EABDC	85.88%
ADBEC	90.59%	CADE	90.16%	CEADB	89.02%	EDBAC	85.88%
CBADE	90.58%	CDEB	90.16%	CEDAB	89.02%	EBDAC	85.88%

CABD	90.54%	CDEBA	90.16%	CEBAD	89.02%	EADBC	85.88%
CBD	90.51%	CDAEB	90.16%	CEABD	89.02%	EDABC	85.88%
CBDA	90.51%	CADEB	90.16%	BDE	88.99%	EBCAD	85.88%
DABE	90.50%	CDEAB	90.16%	BDEA	88.99%	EACBD	85.88%
BAE	90.50%	CADBE	90.16%	BDAE	88.99%	EBACD	85.88%
BAEC	90.50%	BACE	90.16%	BADE	88.90%	EABCD	85.88%
BAECD	90.50%	BACED	90.16%	BDEC	88.87%	ECA	85.42%
ADC	90.50%	DABC	90.15%	BDECA	88.87%	ECB	85.42%
ADCE	90.50%	DACBE	90.11%	BDAEC	88.87%	ECD	85.42%
ADCEB	90.50%	DBAC	90.11%	BDEAC	88.87%	ECBA	85.42%
DAC	90.50%	DBACE	90.11%	BADEC	88.78%	ECAB	85.42%
DEA	90.49%	ABDEC	90.08%	CAE	88.69%	ECDA	85.42%
DAE	90.49%	CDB	90.07%	CAEB	88.69%	ECAD	85.42%
DEB	90.49%	CDBA	90.07%	CAED	88.69%	ECDB	85.42%
DEBA	90.49%	CDAB	90.07%	CAEDB	88.69%	ECBD	85.42%
DAEB	90.49%	CDBE	90.07%	CAEBD	88.69%	ECDBA	85.42%
DEAB	90.49%	CDBEA	90.07%	BDC	88.48%	ECBDA	85.42%
DBEC	90.44%	CDBAE	90.07%	BDCA	88.48%	ECADB	85.42%
DBECA	90.44%	CDABE	90.07%	BDAC	88.48%	ECDAB	85.42%
DBAEC	90.44%	DCAB	90.05%	BDCE	88.48%	ECBAD	85.42%
DBEAC	90.44%	DCABE	90.05%	BDCEA	88.48%	ECABD	85.42%

▼表十八、篩選法各種組合之運算時間（完全解法之平均運算時間為1723881）（單位：ms）

CEDAB	0.0284725	ACBD	0.0733483	BADEC	0.1197292
CEBDA	0.0285583	ACDB	0.07356	BDAC	0.12263
CEBD	0.0285925	ACDEB	0.0770758	DBCA	0.12281
CEBA	0.0287283	ACDE	0.0772633	ACB	0.1230358
CEAD	0.0287392	ABCED	0.0779733	ABDCE	0.1325717
CEDA	0.028765	ABCE	0.0815992	ABDC	0.1334558
CED	0.0287658	ABCDE	0.0852908	DCA	0.1357133
CEA	0.0287708	ACD	0.087555	ADCB	0.1398675
CEADB	0.028775	ABCD	0.0886042	ADCBE	0.1399308
CEBAD	0.0288025	ECBD	0.0948258	ADCEB	0.1402508
CEDBA	0.0288067	ECABD	0.09489	ADCE	0.1405292
CEAB	0.0289917	ECBAD	0.0948942	ABDEC	0.1412858
CEDB	0.0289992	ECDBA	0.0949717	ADBC	0.144

CEABD	0.0290075	ECDB	0.095095	ADBCE	0.1440658
CEB	0.0290783	ECAD	0.0951433	ADC	0.1450533
CAEBD	0.029155	ECADB	0.09517	ADBEC	0.1518017
CAEB	0.02923	ECDA	0.0951767	ADEC	0.1532808
CAEDB	0.0293483	ECD	0.0952525	ADECB	0.1533783
CAE	0.0294358	ECDAB	0.09528	ADEBC	0.1536575
CAED	0.0294683	ECBDA	0.095295	BDC	0.1552167
CBDEA	0.0344792	ABEDC	0.0953733	DCB	0.1601467
CBADE	0.0346892	ABECD	0.0957425	DBC	0.1628308
CBDE	0.0347392	EADC	0.0964117	DECB	0.17768
CBDAE	0.0347408	EADCB	0.0964842	DEAC	0.1779892
CABDE	0.0349217	EDCAB	0.0965358	DECBA	0.178095
CDBE	0.0349375	EACDB	0.0965675	DECAB	0.1782217
CDBEA	0.0350342	EABDC	0.096585	DEACB	0.178255
CDABE	0.03508	EBCDA	0.0966283	DECA	0.1783092
CDBAE	0.0351408	EDAC	0.09664	DEBAC	0.1783925
CDEBA	0.0358333	EBDC	0.0967067	DEBCA	0.17851
CDEAB	0.0359	EBCD	0.0967283	DEBC	0.17868
CDAE	0.035905	EDBCA	0.096745	DEC	0.1787658
CDEB	0.035945	EBDCA	0.096745	DEABC	0.17881
CDEA	0.0359975	EDBC	0.0967525	BAC	0.1798983
CDE	0.0361092	EBCAD	0.0967783	DACBE	0.1810083
CADBE	0.036225	EBDAC	0.0967958	DACE	0.1828683
CBEDA	0.0362775	EDCB	0.0967967	DACEB	0.1830408
CBEAD	0.03634	EBACD	0.0968	DABCE	0.1852517
CBEA	0.0363708	EADBC	0.096855	DACB	0.1884817
CDAEB	0.0364133	EABCD	0.0968958	DABC	0.1929475
CBAE	0.0364967	EDBAC	0.0969192	DABEC	0.1932942
CADE	0.036615	EDCBA	0.09692	DAEC	0.1998508
CADEB	0.0366775	EDCA	0.09695	DAEBC	0.2003492
CBED	0.0366867	EACD	0.097015	DAECB	0.2007775
CBAED	0.0366908	EACBD	0.0971783	ABC	0.20665
CBE	0.0367008	EDABC	0.0972917	DAC	0.2086842
CABED	0.0375975	EDC	0.0974408	AECD	0.2598767
CDBA	0.0376525	EDACB	0.09751	AECDB	0.2599392
CABE	0.0376642	EBADC	0.0975875	AEDC	0.2601917
CDB	0.03799	ECAB	0.0980683	AEBCD	0.2602975

CDAB	0.0383183		ECBA	0.0982567		AECBD	0.2603083
CADB	0.0388833		ECB	0.098305		AEDBC	0.2605092
CBD	0.040145		ECA	0.0984833		AEDCB	0.26103
CBAD	0.0402258		ACED	0.0991133		AEBDC	0.26116
CBDA	0.0403142		ACE	0.0991533		AECB	0.2714725
BCED	0.0407017		ACEBD	0.0991875		AEBC	0.2719175
BCEA	0.0407475		ACEB	0.0992483		AEC	0.2730933
BCEDA	0.0407675		ACEDB	0.0993283		BDAE	0.3180075
BCAE	0.0408958		BDCE	0.0995025		BDE	0.3189392
BCE	0.0410258		EACB	0.099715		BDEA	0.3196042
BCEAD	0.0410742		ABEC	0.0997308		BADE	0.3453642
BCAED	0.0411958		EBCA	0.099745		ABDE	0.4477592
BCADE	0.0412217		EBAC	0.09978		DEB	0.4680858
CABD	0.0415208		BDCEA	0.0998008		DEBA	0.4684425
BCDAE	0.0416825		EBC	0.0999133		DEA	0.4685375
BCDE	0.0417817		BDCAE	0.0999658		DEAB	0.4686825
BCDEA	0.0418325		EABC	0.1000542		ADBE	0.4830725
CAD	0.0443342		EAC	0.1000583		DABE	0.4961108
CDA	0.0458667		BDACE	0.1057942		ADE	0.5033533
BCAD	0.0490583		ABED	0.1063258		ADEB	0.503585
BCDA	0.0496825		DCEA	0.1085008		DAE	0.5050817
BCD	0.0501808		BADCE	0.1086717		DAEB	0.5056592
BACED	0.0563567		DCAEB	0.1087108		BCA	0.6634517
BACE	0.0568875		DCAE	0.1087983		CBA	0.7046717
BACDE	0.0569575		DCE	0.10896		CAB	0.7166058
BECAD	0.0658458		DBACE	0.1090525		DBAE	0.724235
BECD	0.0660275		DCEBA	0.1090533		DBE	0.7247258
BECDA	0.0660408		DCEB	0.1090842		DBEA	0.7338642
BEADC	0.0661675		DCEAB	0.1091575		DAB	1.2012167
BEDC	0.0664225		DCBAE	0.1122217		EDB	1.2076133
BEACD	0.0665492		DCBE	0.1124042		EBDA	1.207855
BEDAC	0.0665583		DCBEA	0.1124233		EBAD	1.2089617
BECA	0.0667875		ABE	0.1133075		EADB	1.2096958
BACD	0.0668617		DCABE	0.113695		EAD	1.2097667
BEDCA	0.0669175		DBAC	0.1138992		EDA	1.2099708
BEAC	0.0670958		BDEC	0.1155825		EDAB	1.2100875
BEC	0.0674525		BDEAC	0.1158642		EABD	1.2103525

BAEDC	0.0682308		BDECA	0.115905		EAB	1.2134858
BEAD	0.0683092		DBECA	0.1165067		EBD	1.2148767
BAECD	0.06851		BDAEC	0.1165467		EBA	1.2152617
BEDA	0.0687958		DBAEC	0.1165958		EDBA	1.2212733
BAEC	0.0692458		DBEAC	0.1166233		ABD	1.4198725
BED	0.0698767		DBEC	0.1168017		ADB	1.485925
BAED	0.0704792		BDCA	0.11797		BDA	1.58366
BEA	0.0711767		DBCE	0.1180875		BAD	1.6230833
BAE	0.0727242		DBCEA	0.1185417		DBA	2.5213683
ACDBE	0.0729483		DBCAE	0.118615		AEDB	2907.1265
ACBDE	0.0729483		DCBA	0.1186633		AEBD	2907.7348
ACBED	0.0732458		BADC	0.1191375		AEB	3778.5423
ACBE	0.0732508		DCAB	0.1196975		AED	29021.208