

國立高雄第一科技大學

運籌管理系

AMPL/CPLEX 使用手冊與範例

盧宗成 楊承堯

2008/8/1

(本使用手冊之編寫由 97 年管理學院發展特色計畫補助)

目錄

第一章 如何開始使用 AMPL/CPLEX 軟體?	1
1.1 如何使用 AMPL/CPLEX 求解數學規劃模型.....	1
1.2 AMPL/CPLEX 結果檢視及輸出常見問題.....	7
1.3 使用 AMPL/CPLEX 來執行 CPLEX9.1 版.....	9
第二章 AMPL/CPLEX 基本編譯指令語法說明	10
2.1 編譯指令.....	10
2.2 編譯語法.....	10
2.3 AMPL 程式指令.....	11
2.4 建立 mod 檔，編譯程式碼.....	11
第三章 求解線性問題	13
3.1 成本最小化運輸問題.....	13
3.2 利用 AMPL 模組化設計求解線性問題.....	20
為何要模組化設計?.....	20
3.3 求解多產品運輸問題.....	27
第四章 利用 AMPL/CPLEX 求解網路問題	34
4.1 最小運輸成本問題.....	34

4.2 最大流量問題.....	41
4.3 最短路徑問題.....	47
第五章 利用 AMPL/CPLEX 求解整數規劃問題.....	52
5.1 0-1 問題.....	52
參考文獻.....	61

圖目錄

圖 1：操作流程圖（一）	1
圖 2：操作流程圖（二）	2
圖 3：操作流程圖（三）	2
圖 4：操作流程圖（四）	3
圖 5：操作流程圖（五）	4
圖 6：操作流程圖（六）	4
圖 7：操作流程圖（七）	5
圖 8：操作流程圖（八）	6
圖 9：設定螢幕緩衝區大小.....	7
圖 10：調整高度的設定.....	8
圖 11：求解過程及結果.....	8
圖 12：開啟執行檔.....	9
圖 13：mod 檔	11
圖 14：成本最小化運輸問題執行結果.....	18
圖 15：成本最小化運輸問題各變數值.....	19
圖 16：模組化成本最小化運輸問題執行結果.....	25

圖 17：模組化成本最小化運輸問題各變數值.....	26
圖 18：多產品運輸問題執行結果.....	32
圖 19：多產品運輸問題變數數值.....	33
圖 20：最小運輸配送網路圖.....	34
圖 21：最小運輸成本問題執行結果.....	39
圖 22：最小運輸成本問題變數數值.....	40
圖 23：最大流量運輸網路圖.....	41
圖 24：最大流量運輸問題執行結果.....	45
圖 25：最大流量運輸問題變數數值.....	46
圖 26：最短路徑運輸網路圖.....	47
圖 27：最短路徑運輸執行結果.....	50
圖 28：最短路徑運輸變數數值.....	51
圖 29：整數規劃問題執行結果.....	58
圖 30：未整數規劃下結果.....	59
圖 31：整數規劃下結果.....	60

表目錄

表 1、鋼鐵生產工廠每年生產單位.....	13
表 2、汽車零件廠年需求量.....	13
表 3、單位距離運輸成本.....	14
表 4、汽車零件廠各類型零件年需求量.....	27
表 5、各類產品的年供給量.....	27
表 6、各類型零件運送成本.....	28
表 7、汽車零件廠各類型零件年需求量.....	52
表 8、各類型零件運送成本.....	52
表 9、固定成本.....	53

第一章 如何開始使用 AMPL/CPLEX 軟體?

本系之 AMPL/CPLEX 軟體，目前安裝於管理學院四樓 ERP 實驗室內之 ORACLE/ILOG Server 上 (IP Address: 163.18.24.212)，該伺服器之規格為：Intel Xeon 雙 CPU 1.80GHz、2.6G 記憶體、40GB 硬碟，並可讀取 USB 裝置及 CD-ROM。

1.1 如何使用 AMPL/CPLEX 求解數學規劃模型

使用 AMPL/CPLEX 之步驟如下：

1. 請先將將要求解的*.mod 檔、*.dat 檔壓縮郵寄到自己的學校信箱，這台伺服器嚴格限制只能登入學校信箱（強烈建議將檔案先壓縮再郵寄，以避免產生亂碼導致無法讀取）。
2. 【開始】→【所有程式】→【附屬應用程式】→【通訊】→【遠端桌面連線】。

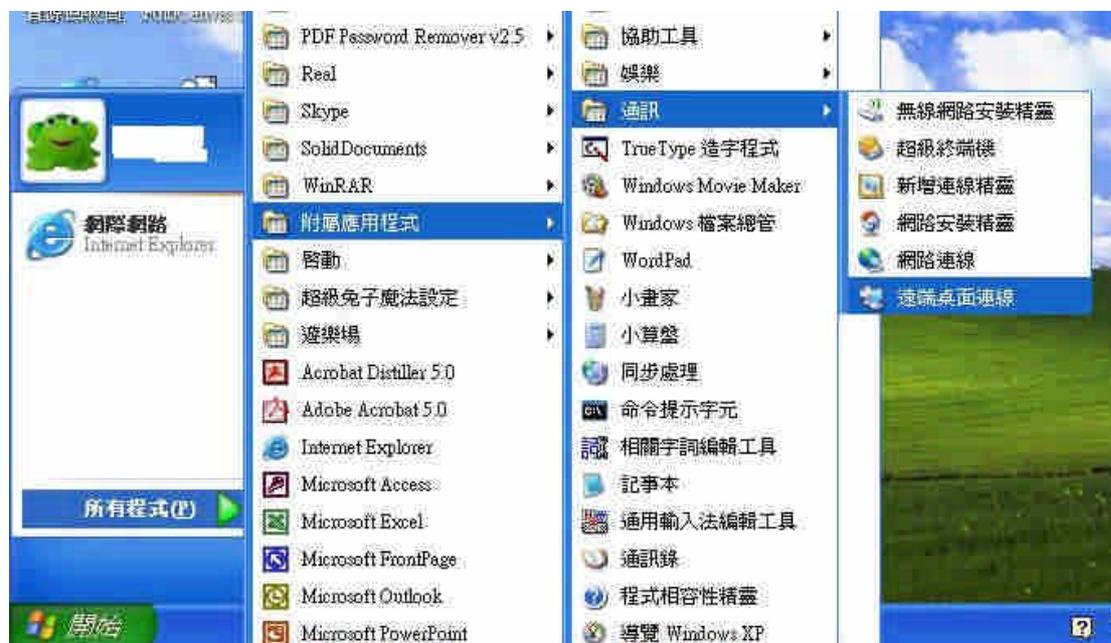


圖 1：操作流程圖（一）

3. 輸入 ILOG Server 之位址：163.18.24.212，按【連線】。



圖 2：操作流程圖（二）

4. 輸入帳號、密碼（帳號、密碼請向負責管理的老師申請），按【確定】。
5. 上網到自己的學校信箱將檔案下載到桌面。
6. 開啟【我的電腦】→【本機磁碟 (C:)】→快速點選【ilmd】以啟動軟體授權。

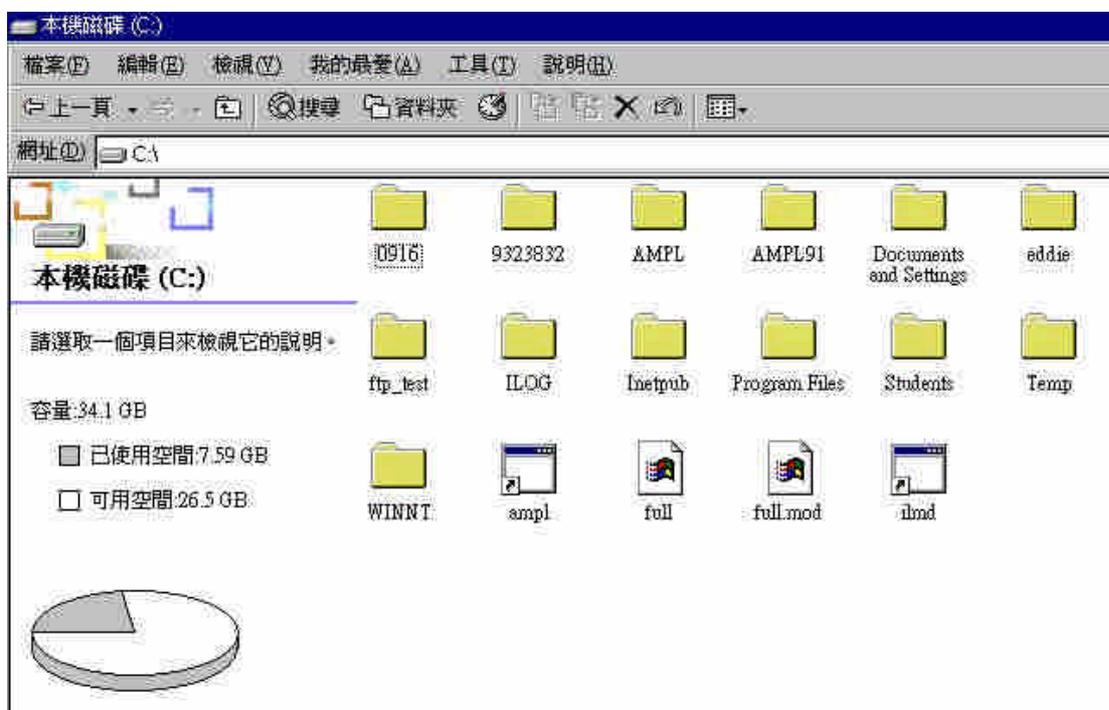


圖 3：操作流程圖（三）

7. 出現底下畫面後，將此畫面『最小化』，使用 AMPL/CPLEX 期間不可結束此授權管理程式。



```
Jun 10 21:43:27 @ ILOG License Manager v2.60 [pid 2284]
Jun 10 21:43:29 @ Using license file "c:\ilog\ilm\access.ilm"
Jun 10 21:43:31 U TOKEN reservation period = 0 sec
Jun 10 21:43:33 l Licensed to "nkfust-taiwan"
Jun 10 21:43:35 F Tokens for AMPL on genuine-ilog: 1 pcwinnt
Jun 10 21:43:35 N Tokens for CPLEX on genuine-ilog: 1 pcwinnt
Jun 10 21:43:35 y Tokens for Configurator on genuine-ilog: 1 pcwinnt
Jun 10 21:43:35 p Tokens for Dispatcher on genuine-ilog: 1 pcwinnt
Jun 10 21:43:35 f Tokens for OPLStudio on genuine-ilog: 1 pcwinnt
Jun 10 21:43:35 e Tokens for Rules on genuine-ilog: 1 pcwinnt
Jun 10 21:43:35 y Tokens for Scheduler on genuine-ilog: 1 pcwinnt
Jun 10 21:43:35 G Tokens for Solver on genuine-ilog: 1 pcwinnt
Jun 10 21:43:35 v Tokens for Views on genuine-ilog: 1 pcwinnt
Jun 10 21:43:35 a Tokens for ViewsCharts on genuine-ilog: 1 pcwinnt
Jun 10 21:43:35 w Tokens for ViewsDataaccess on genuine-ilog: 1 pcwinnt
Jun 10 21:43:35 U Tokens for ViewsDataaccessSQL on genuine-ilog: 1 pcwinnt
-
```

新注半：

圖 4：操作流程圖（四）

8. 打開【AMPL 資料夾】

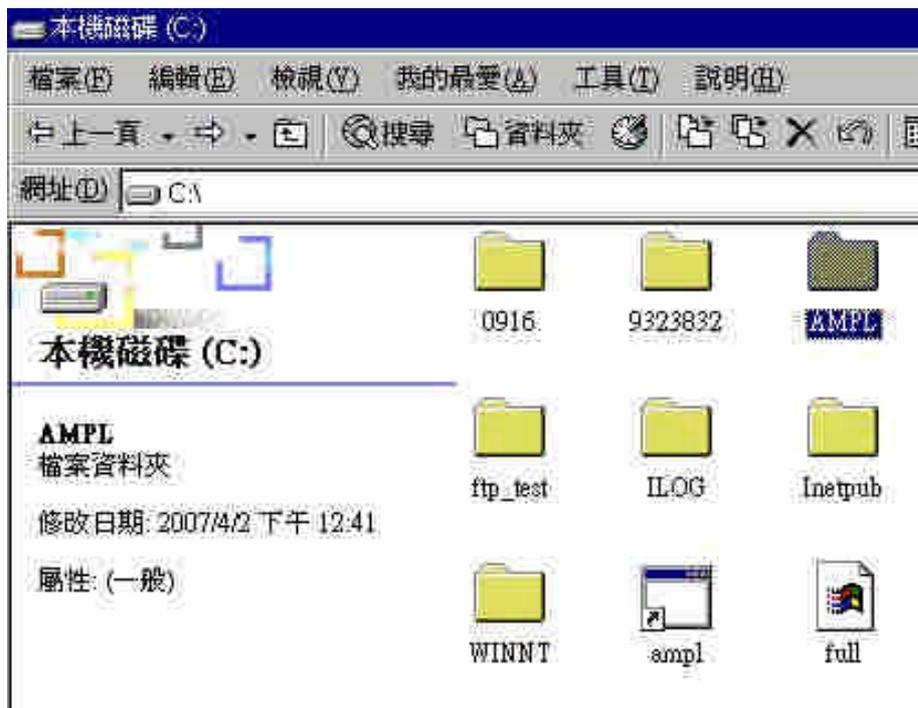


圖 5：操作流程圖（五）

9. 將要求解的*.mod 檔、*.dat 檔解壓縮到 C:\AMPL 資料夾或指定之資料夾中。(如下例將 diet.mod、diet.dat 解壓縮到 C:\AMPL)

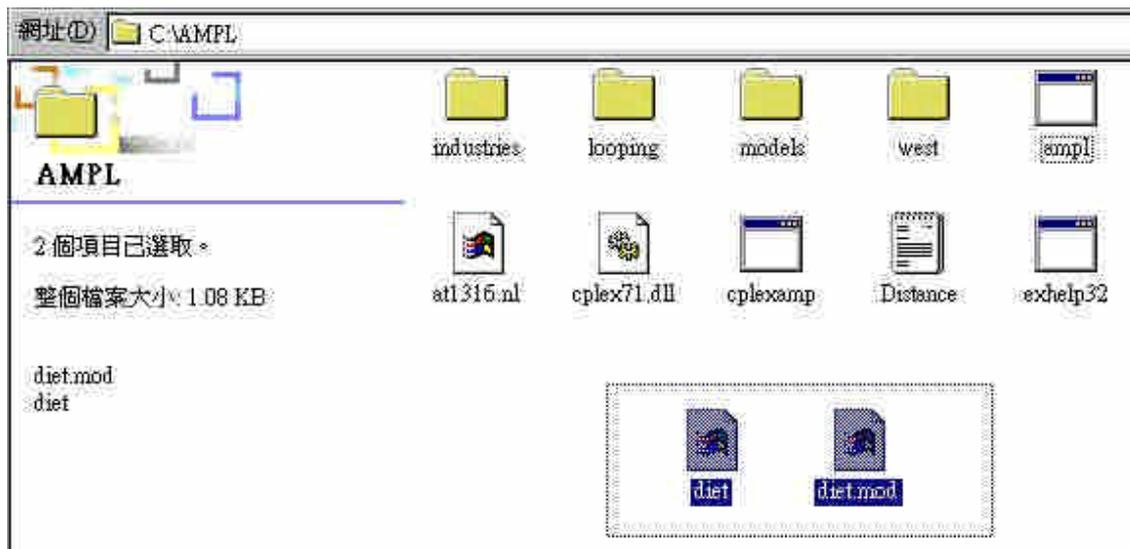


圖 6：操作流程圖（六）

10. 快速點選【AMPL】二次。

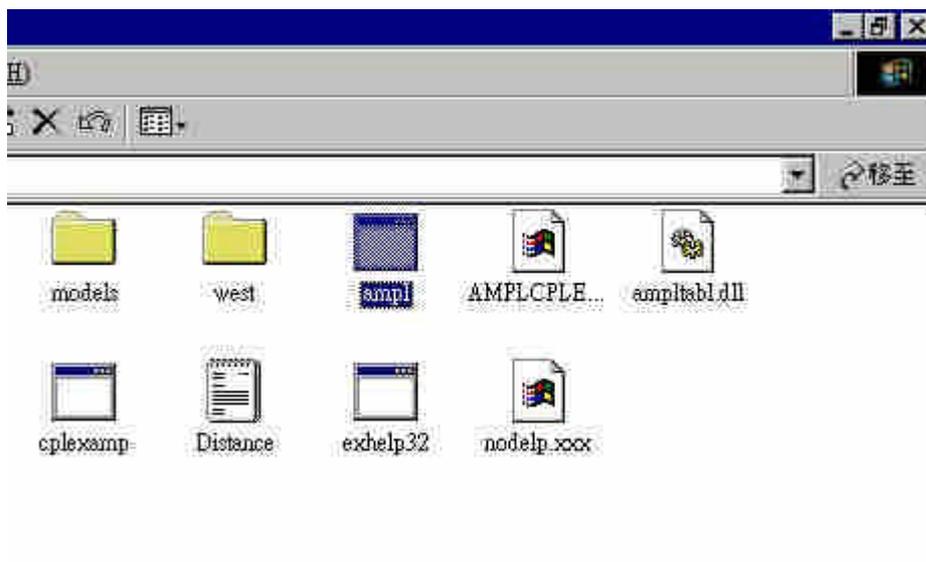


圖 7：操作流程圖（七）

11. 在此視窗中，求解 AMPL 模型。

範例： 輸入”model diet.mod;” ，按 ENTER

輸入”data diet.dat;” ，按 ENTER

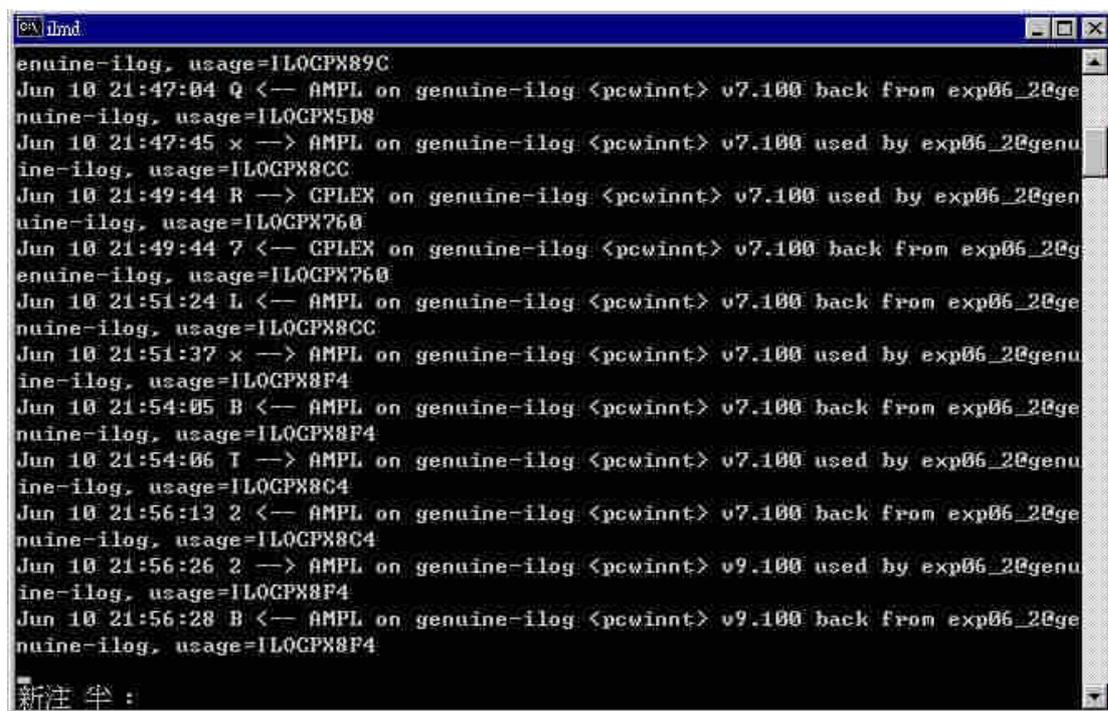
輸入”solve;” ，按 ENTER

詳細之 AMPL 操作指令請參閱 AMPL-A modeling language for mathematical programming 一書（可至系辦或圖書館借閱）。

12. 完成模型求解後，可將答案複製到記事本上，再將檔案壓縮，E-mail 到自己的信箱。

13. 結束 AMPL 軟體前，請輸入”quit;” ，再按 ENTER 以關閉 AMPL 視窗。

14. 登出 ILOG Sever 前，請先將授權管理程式【ilmd】關閉。



```
ilmd
genuine-ilog, usage=ILOCPX89C
Jun 10 21:47:04 Q <- AMPL on genuine-ilog <pcwinnt> v7.100 back from exp06_20ge
genuine-ilog, usage=ILOCPX5D8
Jun 10 21:47:45 x --> AMPL on genuine-ilog <pcwinnt> v7.100 used by exp06_20genu
genuine-ilog, usage=ILOCPX8CC
Jun 10 21:49:44 R --> CPLEX on genuine-ilog <pcwinnt> v7.100 used by exp06_20gen
genuine-ilog, usage=ILOCPX760
Jun 10 21:49:44 7 <- CPLEX on genuine-ilog <pcwinnt> v7.100 back from exp06_20ge
genuine-ilog, usage=ILOCPX760
Jun 10 21:51:24 L <- AMPL on genuine-ilog <pcwinnt> v7.100 back from exp06_20ge
genuine-ilog, usage=ILOCPX8CC
Jun 10 21:51:37 x --> AMPL on genuine-ilog <pcwinnt> v7.100 used by exp06_20genu
genuine-ilog, usage=ILOCPX8F4
Jun 10 21:54:05 B <- AMPL on genuine-ilog <pcwinnt> v7.100 back from exp06_20ge
genuine-ilog, usage=ILOCPX8F4
Jun 10 21:54:06 T --> AMPL on genuine-ilog <pcwinnt> v7.100 used by exp06_20genu
genuine-ilog, usage=ILOCPX8C4
Jun 10 21:56:13 2 <- AMPL on genuine-ilog <pcwinnt> v7.100 back from exp06_20ge
genuine-ilog, usage=ILOCPX8C4
Jun 10 21:56:26 2 --> AMPL on genuine-ilog <pcwinnt> v9.100 used by exp06_20genu
genuine-ilog, usage=ILOCPX8F4
Jun 10 21:56:28 B <- AMPL on genuine-ilog <pcwinnt> v9.100 back from exp06_20ge
genuine-ilog, usage=ILOCPX8F4
新注半:
```

圖 8：操作流程圖（八）

按下螢幕正上方的黃色方塊中，最右邊的 X，即可登出。離開「遠端桌面連線」時，要按「重新開機」以利下一位使用者利用「遠端登入連線」進入本系統。

1.2 AMPL/CPLEX 結果檢視及輸出常見問題

如果問題的答案很多，導致 DOS 介面無法完整地將答案呈現出來，可以以下二種方法得知完整答案。

- A. 使用 IO 轉向指令輸出至文字檔。
- B. 將螢幕緩衝區放大來解決此問題。

設定螢幕緩衝區大小的步驟如下：

1. 按右鍵選【內容】。



圖 9：設定螢幕緩衝區大小

- 調整高度的設定值（預設為 300，最大為 6000），再按兩次【確定】。

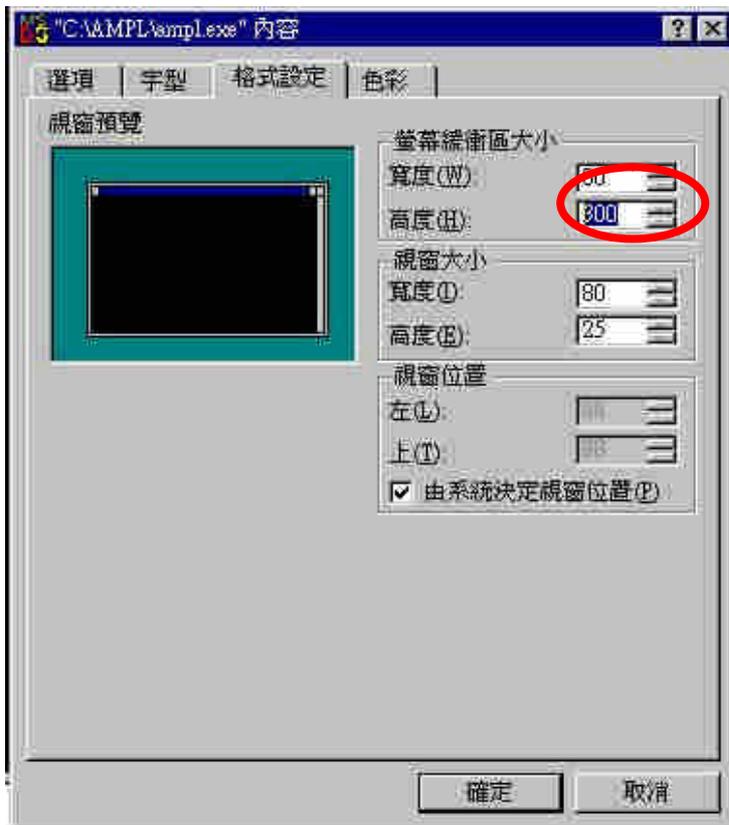


圖 10：調整高度的設定

- 移動視窗右側拉桿以檢視求解過程及結果，如下圖紅色圈圈所示。

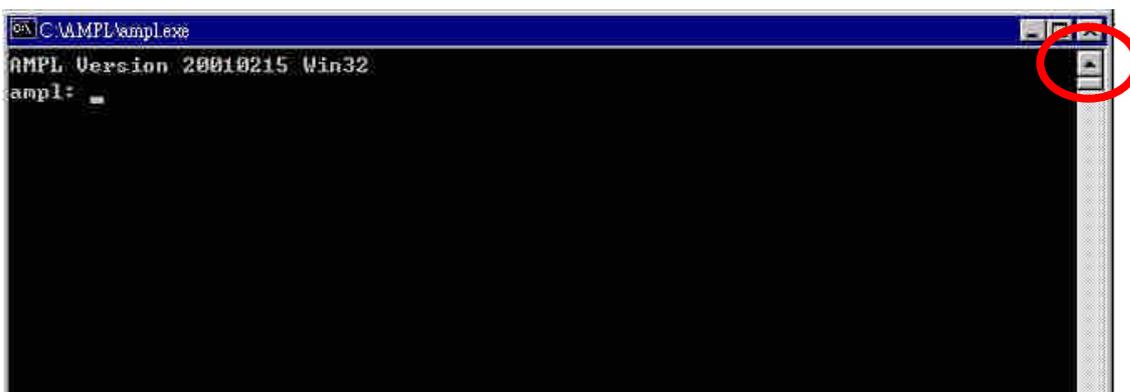


圖 11：求解過程及結果

1.3 使用 AMPL/CPLEX 來執行 CPLEX9.1 版

開啟【C:\AMPL91】資料夾後，快速點選【ampl】圖示二次即可使用 AMPL/CPLEX9.1。

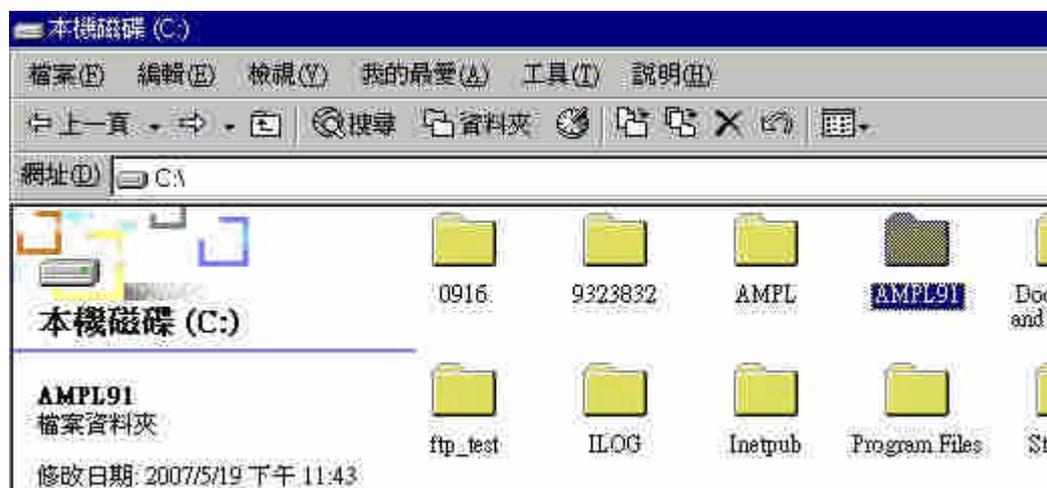


圖 12：開啟執行檔

第二章 AMPL/CPLEX 基本編譯指令語法說明

2.1 編譯指令

var 設定一個變數。

+ - * / 加減乘除四則運算子。

> < = <= >= 比較運算子。

maximize 設定最大化問題。

minimize 設定最小化問題。

subject to 設定一條限制式。

param 設定參數

2.2 編譯語法

var x >= 0; 為設定一個 x 的變數，範圍大於等於 0，變數代號可以自行定義。
(注意語法最後需要 ; 結尾)

x11 + x12 <= 0; 2 變數方程式。變數設定須符合規定。
(注意語法最後需要 ; 結尾)

minimize Z: 以 Z 為代號的最小化問題，代號可以自行定義。
(注意語法最後需要 ; 結尾)

subject to A: 代號 A 之限制式，代號可以自行定義。
(注意語法最後需要 ; 結尾)

2.3 AMPL 程式指令

model	開啟檔案指令
data	開啟資料庫指令
solve	求解指令
display	顯示求解後的變數值指令

2.4 建立 mod 檔，編譯程式碼

開啟一個*.txt 文件檔，將目標式與限制式以*.txt 文件編寫，將編寫好*.txt 檔以 *.mod 存檔。才能使 AMPL 軟體讀取。



圖 13：mod 檔

由於 AMPL 無法讀取有上下標的變數，txt 文件中也無法編寫有上下標之變數，所以使用者要自行定義可被 txt 檔編輯之變數。還有 AMPL 中的指令也不行當作變數代號，以下是常用的幾個基本指令。

Current	complement	integer	solve_result_num
IN	contains	less	suffix
INOUT	default	logical	sum
Infinity	dimen	max	symbolic
Initial	div	min	table
LOCAL	else	option	then
OUT	environ	setoff	union
all	exists	shell_exitcode	while
binary	forall	solve_exitcode	within
by	if	solve_message	
check	in	solve_result	

第三章 求解線性問題

3.1 成本最小化運輸問題

例題一

假設有三家鋼鐵工廠生產鋼捲，其生產單位要供給 7 家汽車零件工廠。下表各表示三家每年生產單位，7 家汽車零件工廠年需求量，與單位距離運輸成本，請以軟體 CPLX 求出最佳化配送目標？

表 1、鋼鐵生產工廠每年生產單位

CARY	Gary, Indiana	1400
CLEV	Cleveland, Ohio	2600
PITT	Pittsburgh, Pennsylvania	2900

表 2、汽車零件廠年需求量

FRA	Framingham, Massachusetts	900
DET	Detroit, Michigan	1200
LAN	Lansing, Michigan	600
WIN	Windsor, Ontario	400
STL	St. Louis, Missouri	1700

FRE	Fremont, California	1100
LAF	Lafayette, Indiana	1000

表 3、單位距離運輸成本

To \ From	CARY	CLEV	PITT
FRA	39	27	24
DET	14	9	14
LAN	11	12	17
WIN	14	9	13
STL	16	26	28
FRE	82	95	99
LAF	6	17	20

設定變數

在表三中，假設 X_{ij} 為三家鋼鐵廠配送到 7 家零件廠的單位數量， $1 \leq i \leq 3$ ， $1 \leq j \leq 7$ ；例如 X_{11} 為 CARY 送往 FRA 的單位數量，以此類推 X_{37} 為 PITT 配送到 LAF 的單位數量。

LP Model 限制式跟目標式

Minimize Cost:

$$\begin{aligned} & 39 X_{11} + 14 X_{12} + 11 X_{13} + 14 X_{14} + 16 X_{15} + 82 X_{16} + 8 X_{17} \\ & + 27 X_{21} + 9 X_{22} + 12 X_{23} + 9 X_{24} + 26 X_{25} + 95 X_{26} + 17 X_{27} \\ & + 24 X_{31} + 14 X_{32} + 17 X_{33} + 13 X_{34} + 28 X_{35} + 99 X_{36} + 20 X_{37} \end{aligned}$$

Subject to:

$$39 X_{11} + 14 X_{12} + 11 X_{13} + 14 X_{14} + 16 X_{15} + 82 X_{16} + 8 X_{17} = 1400$$

$$27 X_{21} + 9 X_{22} + 12 X_{23} + 9 X_{24} + 26 X_{25} + 95 X_{26} + 17 X_{27} = 2600$$

$$24 X_{31} + 14 X_{32} + 17 X_{33} + 13 X_{34} + 28 X_{35} + 99 X_{36} + 20 X_{37} = 2900$$

$$X_{11} + X_{21} + X_{31} = 900$$

$$X_{12} + X_{22} + X_{32} = 1200$$

$$X_{13} + X_{23} + X_{33} = 600$$

$$X_{14} + X_{24} + X_{34} = 400$$

$$X_{15} + X_{25} + X_{35} = 1700$$

$$X_{16} + X_{26} + X_{36} = 1100$$

$$X_{17} + X_{27} + X_{37} = 1000$$

$$\text{All } X_{ij} \geq 0$$

編譯程式碼:

var x11 >= 0; var x21 >= 0; var x31 >= 0; # 變數定義

var x12 >= 0; var x22 >= 0; var x32 >= 0;

var x13 >= 0; var x23 >= 0; var x33 >= 0;

var x14 >= 0; var x24 >= 0; var x34 >= 0;

var x15 >= 0; var x25 >= 0; var x35 >= 0;

var x16 >= 0; var x26 >= 0; var x36 >= 0;

var x17 >= 0; var x27 >= 0; var x37 >= 0;

minimize cost: # 目標式

39*x11 + 14*x12 + 11*x13 + 14*x14 + 16*x15 + 82*x16 + 8*x17
+27*x21 + 9*x22 + 12*x23 + 9*x24 + 26*x25 + 95*x26 + 17*x27
+24*x31 + 14*x32 + 17*x33 + 13*x34 + 28*x35 + 99*x36 + 20*x37;

目標方程式

subject to A: # 限制式 A

x11 + x12 + x13 + x14 + x15 + x16 + x17 = 1400; # 限制式 A 方程式

subject to B: # 限制式 B

x21 + x22 + x23 + x24 + x25 + x26 + x27 = 2600;

subject to C: # 限制式 C

$$x_{31} + x_{32} + x_{33} + x_{34} + x_{35} + x_{36} + x_{37} = 2900;$$

subject to D: # 限制式 D

$$x_{11} + x_{21} + x_{31} = 900;$$

subject to E: # 限制式 E

$$x_{12} + x_{22} + x_{32} = 1200;$$

subject to F: # 限制式 F

$$x_{13} + x_{23} + x_{33} = 600;$$

subject to G: # 限制式 G

$$x_{14} + x_{24} + x_{34} = 400;$$

subject to H: # 限制式 H

$$x_{15} + x_{25} + x_{35} = 1700;$$

subject to I: # 限制式 I

$$x_{16} + x_{26} + x_{36} = 1100;$$

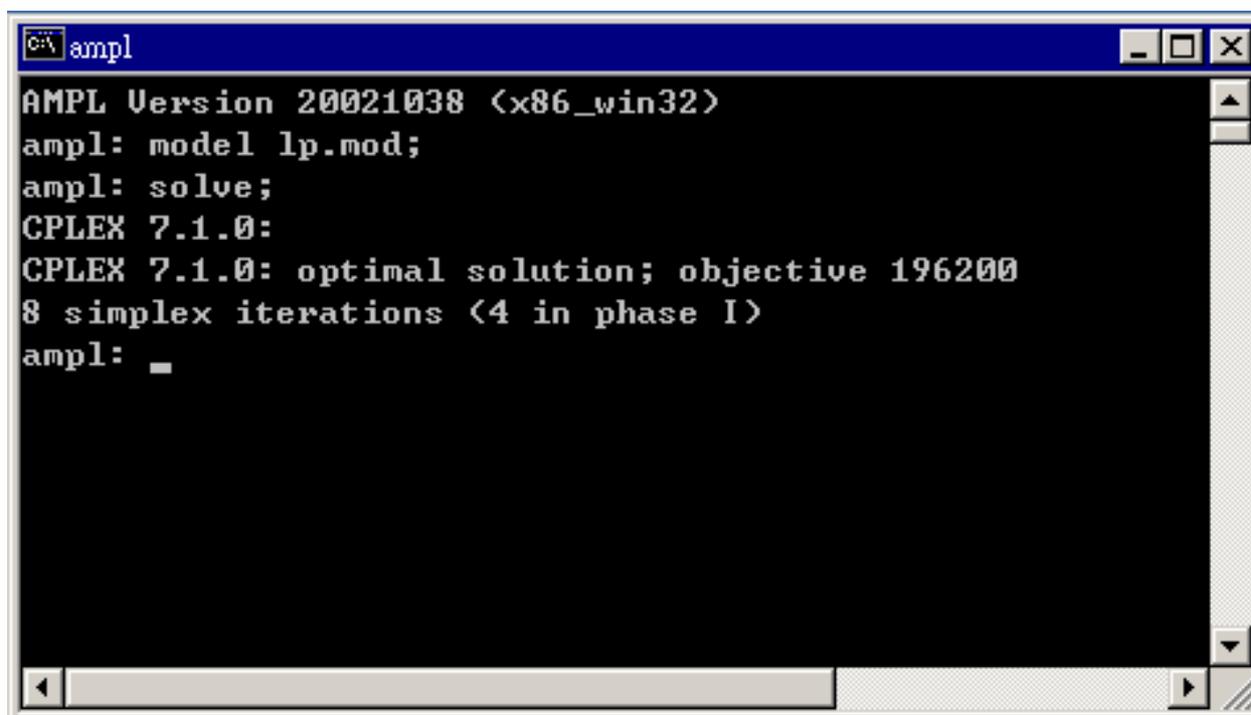
subject to J: # 限制式 J

$$x_{17} + x_{27} + x_{37} = 1000;$$

執行結果

輸入”model lp.mod;” ，按 ENTER (lp.mod 為自行存檔之檔案名)

輸入”solve;” ，按 ENTER



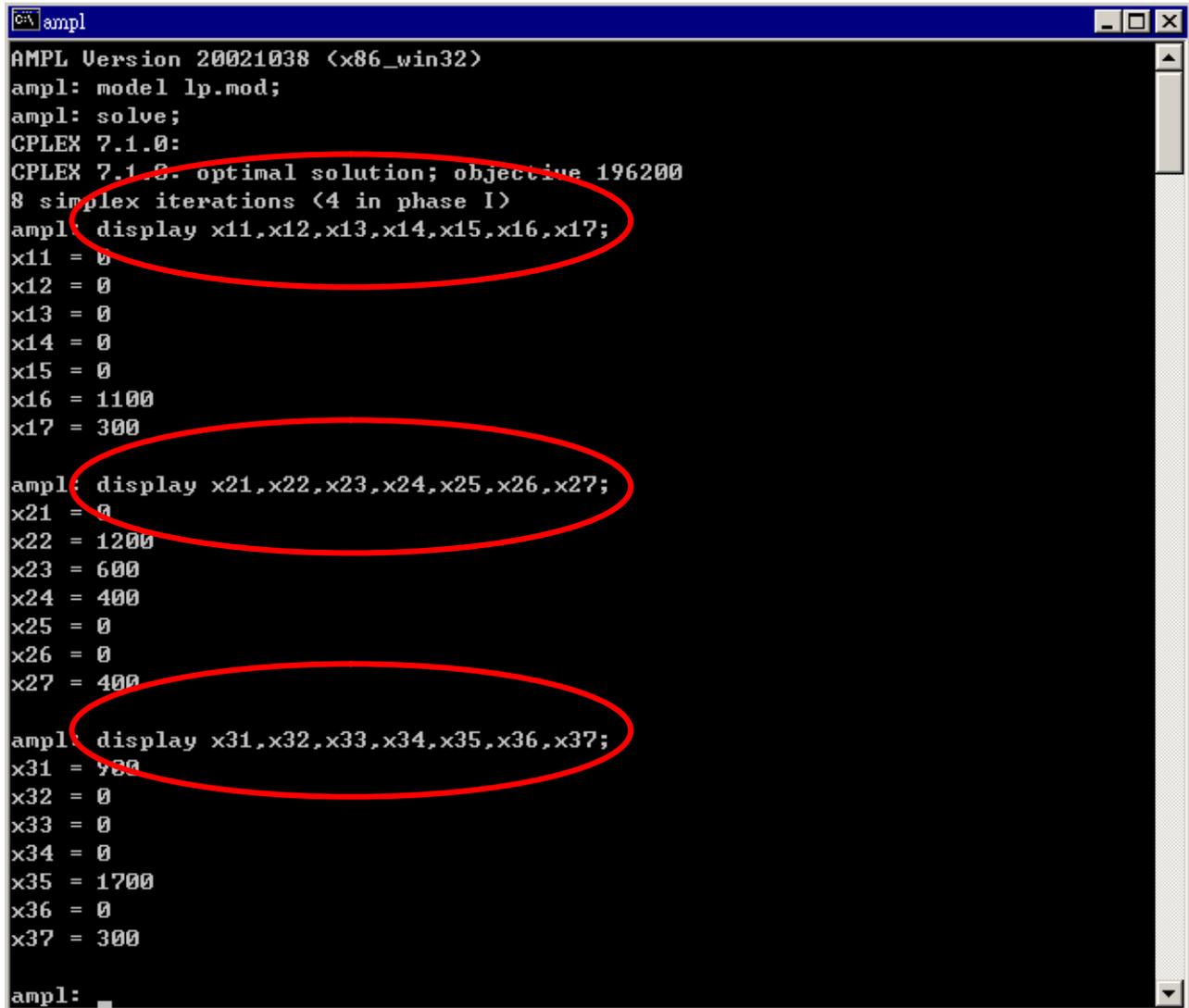
```
C:\> ampl
AMPL Version 20021038 (x86_win32)
ampl: model lp.mod;
ampl: solve;
CPLEX 7.1.0:
CPLEX 7.1.0: optimal solution; objective 196200
8 simplex iterations (4 in phase I)
ampl: _
```

圖 14：成本最小化運輸問題執行結果

顯示各變數值

輸入”display x11,x12,x13,x14,x15,x16,x17;”，按 ENTER （之後變數以此類推）

如果想要一次全部變數值，需要用 data 檔編譯，請自行參考書籍。



```
C:\> ampl
AMPL Version 20021038 (x86_win32)
ampl: model lp.mod;
ampl: solve;
CPLEX 7.1.0:
CPLEX 7.1.0: optimal solution; objective 196200
8 simplex iterations (4 in phase I)
ampl: display x11,x12,x13,x14,x15,x16,x17;
x11 = 0
x12 = 0
x13 = 0
x14 = 0
x15 = 0
x16 = 1100
x17 = 300

ampl: display x21,x22,x23,x24,x25,x26,x27;
x21 = 0
x22 = 1200
x23 = 600
x24 = 400
x25 = 0
x26 = 0
x27 = 400

ampl: display x31,x32,x33,x34,x35,x36,x37;
x31 = 900
x32 = 0
x33 = 0
x34 = 0
x35 = 1700
x36 = 0
x37 = 300

ampl: 
```

圖 15：成本最小化運輸問題各變數值

3.2 利用 AMPL 模組化設計求解線性問題

為何要模組化設計?

在上個例題中，我們可以看到需要對每個變數、每條限制式都定義一個代號。這是非常冗長且無效率的。所以 AMPL 提供可以使用模組化設計來執行，只要設計函數模型在搭配資料檔，這不僅方便，而且當問題需要修改時，只要修改函數模型或資料檔中的其中設定，就可以對整個問題做調整。不像先前那種編譯，要修改就必須一個個進行檢查。以下介紹如何將原編碼轉成模組化編碼。

模組化:

```
set ORIG;

set DEST;

param supply {ORIG} >= 0;

param demand {DEST} >= 0;

check: sum {I in ORIG} supply[i] = sum {j in DEST} demand[j];

param cost {ORIG,DEST} >= 0;

var Trans {ORIG,DEST} >= 0;

minimize Total_Cost:

    sum {i in ORIG, j in DEST} cost[i,j] * Trans[i,j];

subject to Supply {i in ORIG}:

    sum {j in DEST} Trans[i,j] = supply[i];
```

subject to Demand {j in DEST}:

$$\sum \{i \text{ in ORIG}\} \text{Trans}[i,j] = \text{demand}[j];$$

建立目標式的 2 個集合，起點與迄點。

set ORIG;

set DEST;

每個起點為供給點，迄點為需求點，供給、需求都要大於等於零。AMPL 中非負數的值為 param 指令，{ ORIG } 是將 ORIG 集合中元素指向 supply。下一行中的 check: 是要對資料中每個變數做確認，確保所有供給等於所有需求，假使資料中不相等，就會違反限制。

param supply {ORIG} >= 0;

param demand {DEST} >= 0;

check: $\sum \{i \text{ in ORIG}\} \text{supply}[i] = \sum \{j \text{ in DEST}\} \text{demand}[j];$

定義每個起點到迄點之間的運輸成本都要大於等於零，還有運輸量也是要大於等於零。還有定義運數量變數 Trans。

param cost {ORIG,DEST} >= 0;

var Trans {ORIG,DEST} >= 0;

Trans[i,j]，從 i 到 j 之間的運輸單位，cost[i,j]，從 i 到 j 之間的單位成本。相乘積為總運輸成本。

cost[i,j] * Trans[i,j]

完整目標式如下列所示，而 $\sum \{i \text{ in ORIG}, j \text{ in DEST}\}$ 是定義所有 i 為起點，所有 j 為迄點。

minimize Total_Cost:

$$\sum \{i \text{ in ORIG}, j \text{ in DEST}\} \text{cost}[i,j] * \text{Trans}[i,j];$$

也可以編譯成

$$\sum \{j \text{ in DEST}, i \text{ in ORIG}\} \text{cost}[i,j] * \text{Trans}[i,j];$$

subject to Supply {i in ORIG}:

subject to Demand {j in DEST}:

Supply 跟 Demand 是 2 限制式的代號(注意這裡用開頭用大寫，是要區別上面已定義參數 supply 跟 demand)。

在全部 Supply 限制式中，我們需要定義所有從 i 出去的量要等於供給變數的總合，Demand 限制式中也是一樣。完整的限制式如下：

subject to Supply {i in ORIG}:

$$\sum \{j \text{ in DEST}\} \text{Trans}[i,j] = \text{supply}[i];$$

subject to Demand {j in DEST}:

$$\sum \{i \text{ in ORIG}\} \text{Trans}[i,j] = \text{demand}[j];$$

建立 mod 檔，儲存程式碼

開啟一個*.txt 文件檔，將目標式與限制式以*.txt 文件編寫，將編寫好*.txt 檔以 *.mod 存檔。才能使 AMPL 軟體讀取。



test.txt



test.mod

當限制式與目標式用*.mod 檔編輯完成後，接下來就是*.dat 資料檔的編譯。

以下是完整資料檔編譯：

```
param: ORIG: supply :=
```

```
    GARY    1400
```

```
    CLEV    2600
```

```
    PITT    2900 ;
```

```
param: DEST: demand :=
```

```
    FRA     900
```

```
    DET    1200
```

```
    LAN     600
```

```
    WIN     400
```

STL 1700

FRE 1100

LAF 1000 ;

param cost:

	FRA	DET	LAN	WIN	STL	FRE	LAF :=
GARY	39	14	11	14	16	82	8
CLEV	27	9	12	9	26	95	17
PITT	24	14	17	13	28	99	20 ;

資料檔的編譯，AMPL 在讀檔時，會把兩空白中間的數值當作一個單元，所以要區分所有元素與數值，可用空白隔開。如果怕編譯雜亂，可以用 Excel 編譯好，在將數值參數複製到*.txt 檔，再存檔為*.dat 檔。

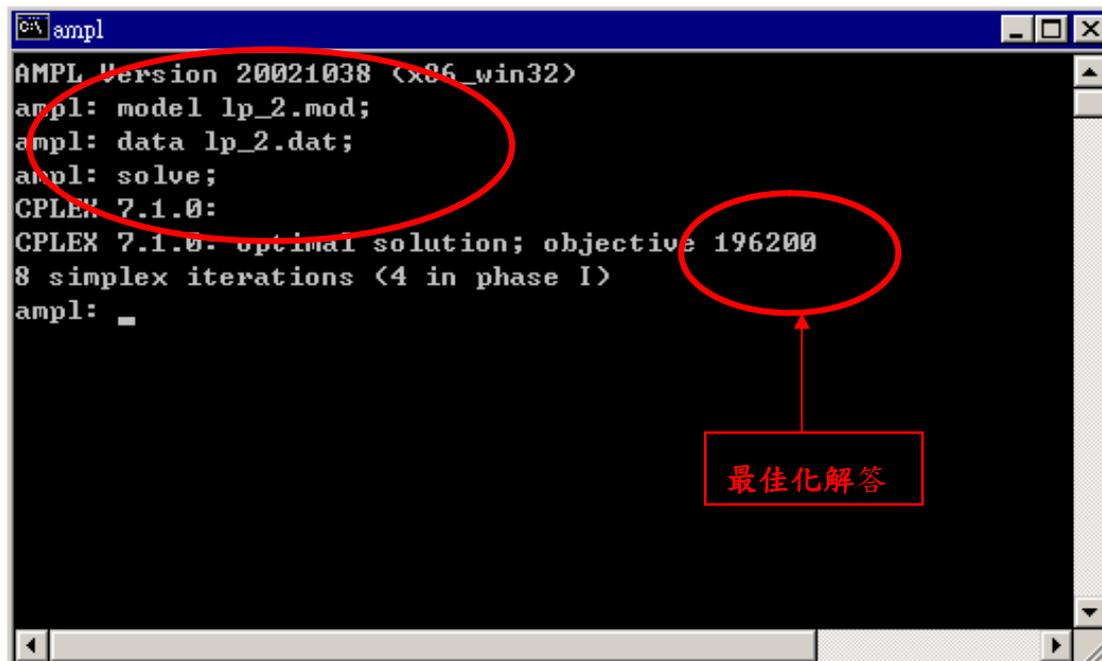


執行結果

輸入”model lp_2.mod;” ，按 ENTER

輸入”data lp_2.dat;” ，按 ENTER

輸入”solve;” ，按 ENTER



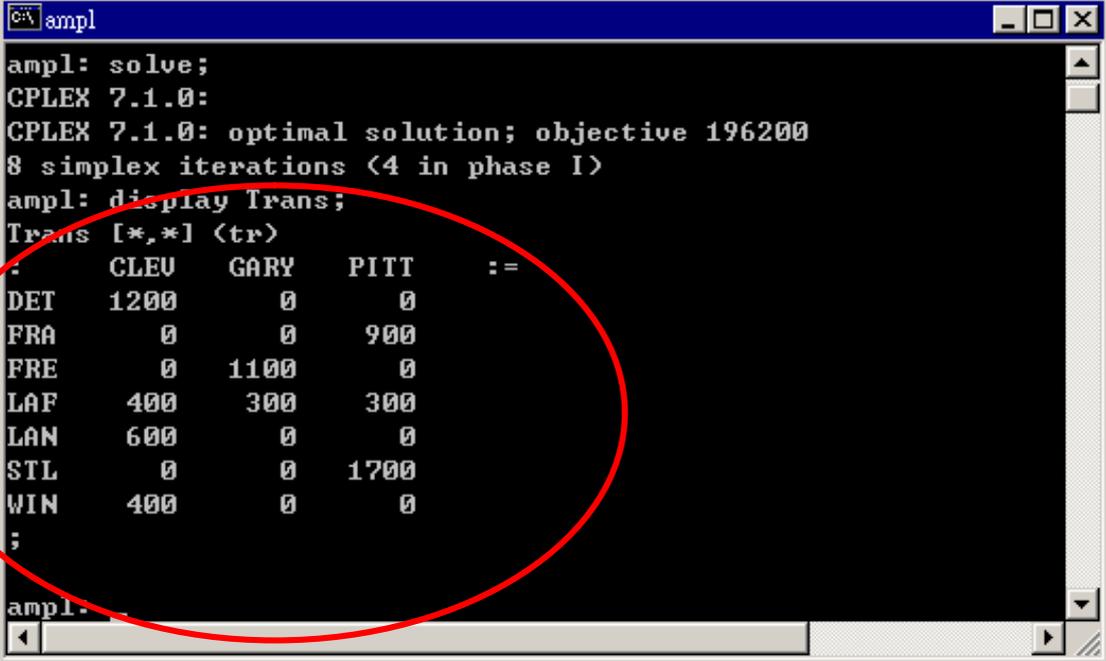
```
C:\> ampl
AMPL Version 20021038 (x86_win32)
ampl: model lp_2.mod;
ampl: data lp_2.dat;
ampl: solve;
CPLEX 7.1.0:
CPLEX 7.1.0: optimal solution; objective 196200
8 simplex iterations (4 in phase I)
ampl: _
```

圖 16：模組化成本最小化運輸問題執行結果

顯示各變數值

輸入”display Trans;”，按 ENTER (Trans 為先前所定義的變數代號)

顯示中的矩陣跟編譯時會有所不同，因為 AMPL 會按照字母順序重新排序。



```
ampl: solve;
CPLEX 7.1.0:
CPLEX 7.1.0: optimal solution; objective 196200
8 simplex iterations (4 in phase I)
ampl: display Trans;
Trans [*,*] (tr)
:      CLEU      GARY      PITT      :=
DET    1200         0         0
FRA         0         0        900
FRE         0       1100         0
LAF     400        300        300
LAN     600         0         0
STL         0         0       1700
WIN     400         0         0
;
```

圖 17：模組化成本最小化運輸問題各變數值

3.3 求解多產品運輸問題

先前的成本最小化問題中，只考慮單一產品。所以在考慮產品多樣性情況下，以先前的模組做以下的修改，可以解決多產品問題。

表 4、汽車零件廠各類型零件年需求量

	FRA	DET	LAN	WIN	STL	FRE	LAF
bands	300	300	100	75	650	225	250
coils	500	750	400	250	950	850	500
plate	100	100	0	50	200	100	250

原先的產品只有單一樣，在這裡把產品係分成三類，每類有不同的運輸成本，如表二所示。

表 5、各類產品的年供給量

	GARY	CLEV	PITT
bands	400	700	800
coils	800	1600	1800
plate	200	300	300

表 6、各類型零件運送成本

(Bands)	FRA	DET	LAN	WIN	STL	FRE	LAF
GARY	30	10	8	10	11	71	6
CLEV	22	7	10	7	21	82	13
PITT	19	11	12	10	25	83	15

(Coils)	FRA	DET	LAN	WIN	STL	FRE	LAF
GARY	39	14	11	14	16	82	8
CLEV	27	9	12	9	26	95	17
PITT	24	14	17	13	28	99	20

(Plate)	FRA	DET	LAN	WIN	STL	FRE	LAF
GARY	41	15	12	16	17	86	8
CLEV	29	9	13	9	28	99	18
PITT	26	14	17	13	31	104	20

由於產品從單一變為多樣，所以在這需要設定一個產品集合。

```
set ORIG;
```

```
set DEST;
```

```
set PROD;
```

```
param supply {ORIG, PROD } >= 0;
```

```
param demand {DEST, PROD } >= 0;
```

在參數方面，也必須考慮到產品種類的問題，所以要加入 PROD 參數。

在流量守衡方面，應為是多種產品，所以檢定是否守衡，是看流進流出的產品量是否相等。

```
check {p in PROD}:
```

```
sum {i in ORIG} supply[i,p] = sum {j in DEST} demand[j,p];
```

限制流量只能為正值。

```
param limit {ORIG,DEST} >= 0;
```

其他參數設定方面，成本與運送單位都必須為正值。

```
param cost {ORIG,DEST,PROD} >= 0;
```

```
var Trans {ORIG,DEST,PROD } >= 0;
```

目標式方面，與先前的最小運輸成本大同小異，只加入了產品參數。

minimize Total_cost:

sum { i in ORIG, j in DEST, p in PROD } cost[i,j,p] * Trans[i,j,p];

限制式方面，如同上，多加入一項產品的限制。

subject to Supply { i in ORIG, p in PROD }:

sum { j in DEST } Trans[i,j,p] = supply[i,p];

subject to Demand { j in DEST, p in PROD }:

sum { i in ORIG } Trans[i,j,p] = demand[j,p];

subject to Multi { i in ORIG, j in DEST }:

sum { p in PROD } Trans[i,j,p] <= limit[i,j];

資料檔設定：

set ORIG := GARY CLEV PITT;

set DEST := FRA DET LAN WIN STL FRE LAF;

set PROD := bands coils plate;

param supply (tr): GARY CLEV PITT :=

bands 400 700 800

coils 800 1600 1800

plate 200 300 300;

param demand (tr): FRA DET LAN WIN STL FRE LAF:=

bands 300 300 100 75 650 225 250

coils 500 750 400 250 950 850 500

plate 100 100 0 50 200 100 250;

param limit default 625; #單一路線上只能運送最大量。

param cost :=

[*,*,bands]: FRA DET LAN WIN STL FRE LAF :=

GARY 30 10 8 10 11 71 6

CLEV 22 7 10 7 21 82 13

PITT 19 11 12 10 25 83 15

[*,*,coils]: FRA DET LAN WIN STL FRE LAF :=

GARY 39 14 11 14 16 82 8

CLEV 27 9 12 9 26 95 17

PITT 24 14 17 13 28 99 20

[*,*,plate]: FRA DET LAN WIN STL FRE LAF :=

GARY 41 15 12 16 17 86 8

CLEV 29 9 13 9 28 99 18

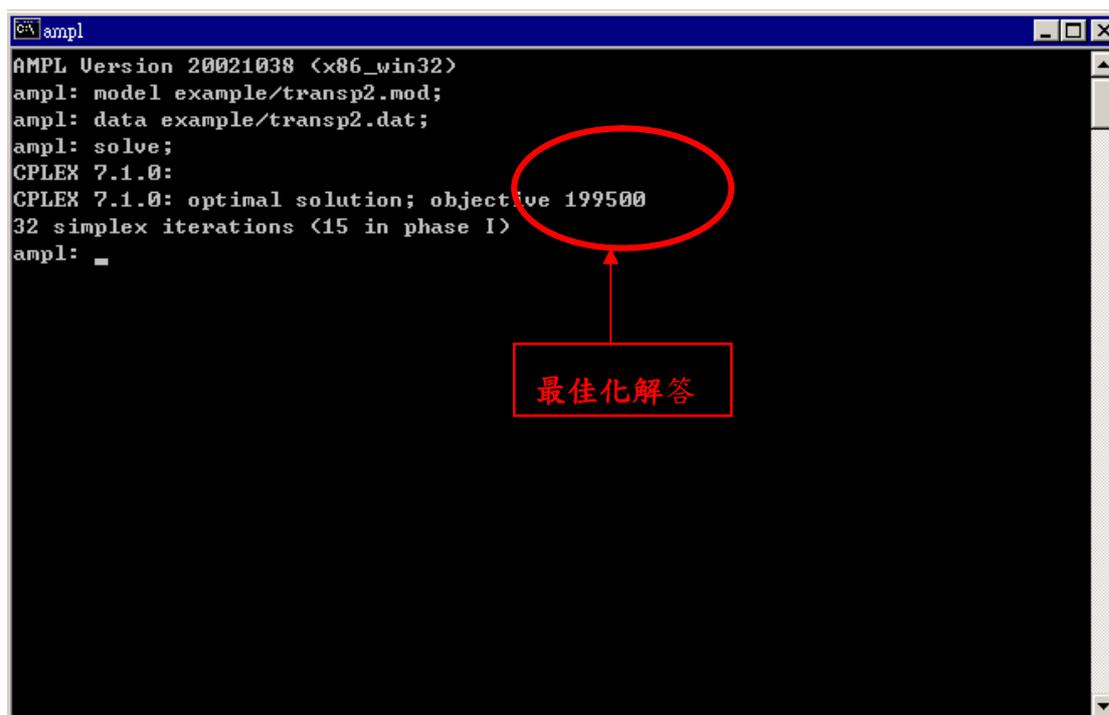
PITT 26 14 17 13 31 104 20;

執行結果

輸入問題模型。

輸入資料檔。

輸入”solve;”



```
ampl
AMPL Version 20021038 (x86_win32)
ampl: model example/transp2.mod;
ampl: data example/transp2.dat;
ampl: solve;
CPLEX 7.1.0:
CPLEX 7.1.0: optimal solution; objective 199500
32 simplex iterations (15 in phase I)
ampl: _
```

圖 18：多產品運輸問題執行結果

數值分析

由下圖得知，三種原料分別由 GARY、CLEV、PITT，送往 FRA、DET、LAN、WIN、STL FRE、LAF 的單位數量。

```
ampl
AMPL Version 20021038 (x86_win32)
ampl: model example/transp2.mod;
ampl: data example/transp2.dat;
ampl: solve;
CPLEX 7.1.0:
CPLEX 7.1.0: optimal solution; objective 199500
32 simplex iterations (15 in phase I)
ampl: display <p in PROD>: <i in ORIG,j in DEST> Trans[i,j,p];
Trans[i,j,'bands'] [*,*] (tr)
:      CLEV  GARY  PITT      :=
DET      0      0    300
FRA    225      0     75
FRE      0      0    225
LAF    225      0     25
LAN      0      0    100
STL    250    400      0
WIN      0      0     75
;

Trans[i,j,'coils'] [*,*] (tr)
:      CLEV  GARY  PITT      :=
DET    525      0    225
FRA      0      0    500
FRE    125    625    100
LAF      0    150    350
LAN    400      0      0
STL    300     25    625
WIN    250      0      0
;

Trans[i,j,'plate'] [*,*] (tr)
:      CLEV  GARY  PITT      :=
DET    100      0      0
FRA     50      0     50
FRE    100      0      0
LAF      0      0    250
LAN      0      0      0
STL      0    200      0
WIN     50      0      0
;

ampl: _
```

圖 19：多產品運輸問題變數數值

第四章 利用 AMPL/CPLEX 求解網路問題

4.1 最小運輸成本問題

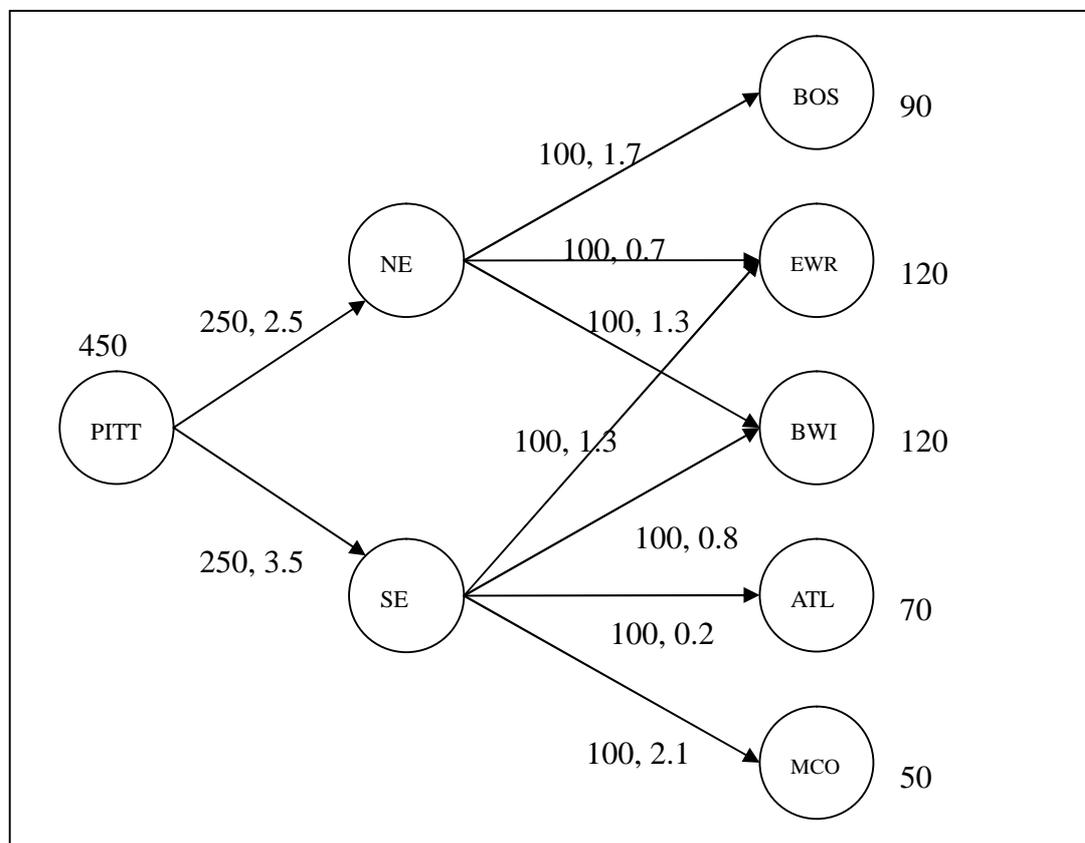


圖 20：最小運輸配送網路圖

上圖表示一家製造商要配送下週產品到倉庫的簡單網路圖形，NE 為北方配送中心，SE 為南方配中心，預計配送 450 個單位的產品到五個倉庫，為 BOS、EWR、BWI、ATL、MCO，分別需求 90、120、120、70、50 單位，每條配送路線上分別有單位路線成本跟配送單位的上限限制。在這網路問題中，目的在找出最低的配送成本路線並符合各倉庫的需求。

在這問題中，為了要描述點到點的問題。我們在這定義兩個集合，分別為城市與路線。每條路線被定義為某一點到下一點的距離，所以在這的編碼可寫成下列所示。

```
set CITIES;
```

```
set LINKS within (CITIES cross CITIES);
```

為了要符合供給與需求的單位數量，所以要有以下限制。

```
param supply {CITIES} >= 0;
```

```
param demand {CITIES} >= 0;
```

在圖一中，每條路線上都有單位成本與上限容量，所以在這我們指派每條路線一個單位成本與上限容量。

```
param cost {LINKS} >= 0;
```

```
param capacity { LINKS } >= 0;
```

在每條路線上，所配送的產品都必須大於零且小於上限容量。

```
var Ship { ( i , j ) in LINKS } >= 0, <= capacity[ i , j ];
```

在目標式中，表示如下；

minimize Total_Cost:

sum { (i , j) in LINKS } cost[i , j] * Ship [i , j];

在限制式中，首先要平衡供給與需求，表示如下；

subject to Balance { k in CITIES }:

supply[k] + sum { (i,k) in LINKS } Ship[i,k] = demand[k] + sum { (k,j) in LINKS }
Ship[k,j];

完整編碼如下所示；

set CITIES;

set LINKS within (CITIES cross CITIES);

param supply { CITIES } >= 0;

param demand { CITIES } >= 0;

check: sum { i in CITIES } supply[i] = sum { j in CITIES } demand[j];

param cost { LINKS } >= 0;

param capacity { LINKS } >= 0;

var Ship { (i,j) in LINKS } >= 0, <= capacity[i,j];

minimize Total_Cost:

$\text{sum } \{(i,j) \text{ in LINKS}\} \text{ cost}[i,j] * \text{Ship } [i,j];$

subject to Balance $\{k \text{ in CITIES}\}$:

$\text{supply}[k] + \text{sum } \{(i,k) \text{ in LINKS}\} \text{Ship}[i,k] = \text{demand}[k] + \text{sum } \{(k,j) \text{ in LINKS}\} \text{Ship}[k,j];$

完整資料檔如下所示；

```
set CITIES := PITT NE SE BOS EWR BWI ATL MCO;
```

```
set LINKS := (PITT,NE) (PITT,SE)
```

```
            (NE,BOS) (NE,EWR) (NE,BWI)
```

```
            (SE,EWR) (SE,BWI) (SE,ATL) (SE,MCO);
```

```
param supply default 0 := PITT 450;
```

```
param demand default 0 :=
```

```
        BOS 90, EWR 120, BWI 120, ATL 70, MCO 50;
```

```
param:    cost    capacity :=
```

```
PITT NE    2.5    250
```

```
PITT SE    3.5    250
```

```
NE BOS     1.7    100
```

```
NE EWR     0.7    100
```

```
NE BWI     1.3    100
```

```
SE EWR     1.3    100
```

```
SE BWI     0.8    100
```

```
SE ATL     0.2    100
```

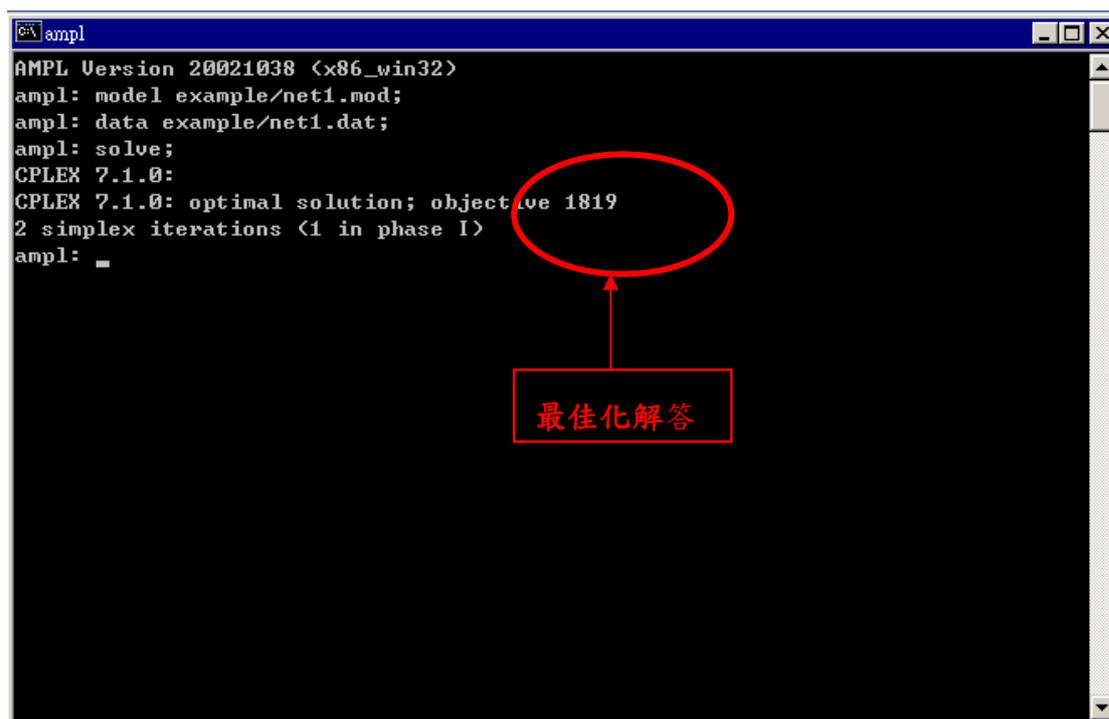
```
SE MCO     2.1    100;
```

執行結果

輸入問題模型。

輸入資料檔。

輸入”solve;”

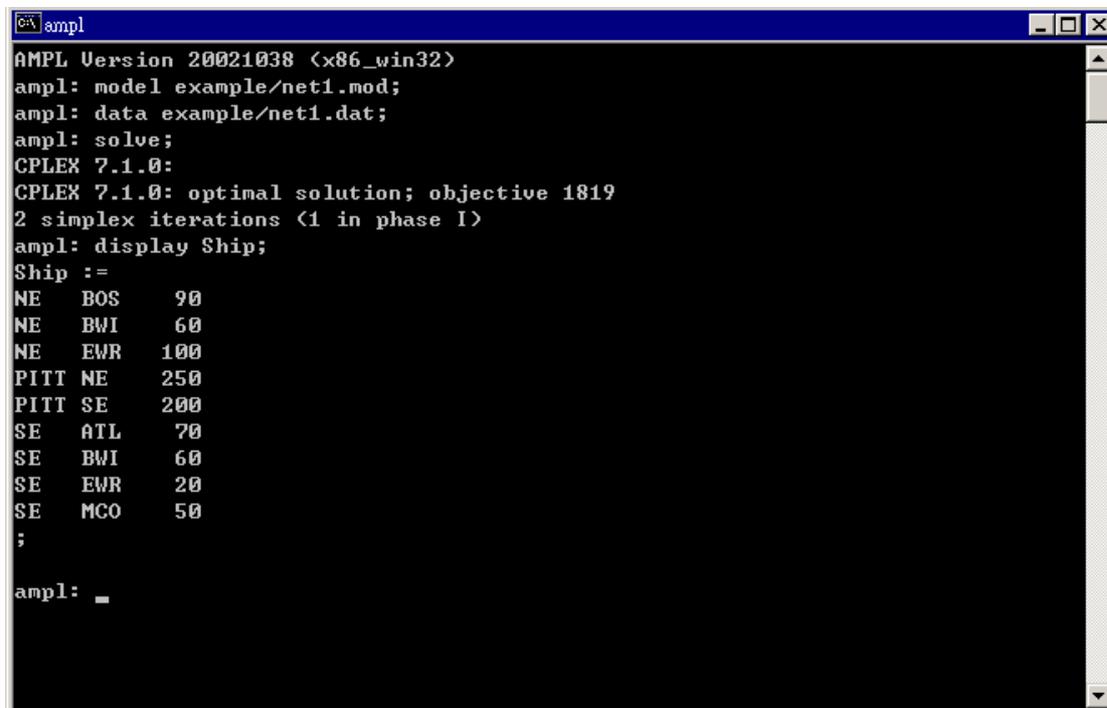


```
ampl
AMPL Version 20021038 (x86_win32)
ampl: model example/net1.mod;
ampl: data example/net1.dat;
ampl: solve;
CPLEX 7.1.0:
CPLEX 7.1.0: optimal solution; objective 1819
2 simplex iterations (1 in phase I)
ampl: _
```

圖 21：最小運輸成本問題執行結果

數值分析

經由 AMPL 運算後得知，每段路線所需運送量。如下圖所示。



```
ampl
AMPL Version 20021038 (x86_win32)
ampl: model example/net1.mod;
ampl: data example/net1.dat;
ampl: solve;
CPLEX 7.1.0:
CPLEX 7.1.0: optimal solution; objective 1819
2 simplex iterations (1 in phase I)
ampl: display Ship;
Ship :=
NE BOS      90
NE BWI      60
NE EWR     100
PITT NE     250
PITT SE     200
SE ATL      70
SE BWI      60
SE EWR      20
SE MCO      50
;
ampl: _
```

圖 22：最小運輸成本問題變數數值

4.2 最大流量問題

在現行規劃問題方面，可細分為最小成本網路問題、運輸及轉運問題、指派問題、最大流量問題、最短路徑問題、最小展開樹問題、多貨品最小成本流量問題、及網路合成問題。而根據預估投入的資源配置來決定各節線的容量，評估網路最大流量與各節線的實際流量，稱為最大流量問題，以下圖二為例，每線段上的數字代表可容許通過的流量上限，並利用 AMPL 軟體試求整個網路 a 到 g 的最大流量。

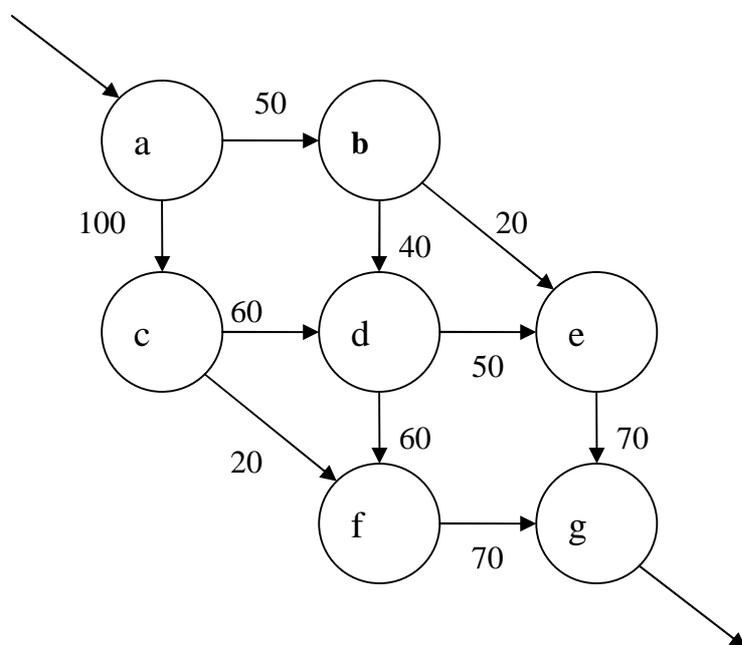


圖 23：最大流量運輸網路圖

以圖二的運輸網路為例，要求得最大流量，首先要設定節點集合；

```
set INTER;
```

```
param entr symbolic in INTER;
```

```
param exit symbolic in INTER, < > entr ;
```

接下來是路線集合，在這行限制式中，是為了要確保所有路線不會同時是入口又是出口；

```
set ROADS within ( INTER diff {exit}) cross ( INTER diff {entr});
```

而在每條運輸容量限制被定義如下；

```
param cap {ROADS} >= 0;
```

```
var Traff {( i,j) in ROADS} >= 0, <= cap[i,j];
```

在目標式中，定義如下；

```
maximize Entering_Traff: sum {(entr,j) in ROADS} Traff[entr,j];
```

在限制式中，為了達到流量守衡，表示如下；

```
subject to Balance {k in INTER diff {entr,exit}}:
```

$$\sum \{(i,k) \text{ in ROADS}\} \text{Traff}[i,k] = \sum \{(k,j) \text{ in ROADS}\} \text{Traff}[k,j];$$

完整編碼：

set INTER;

param entr symbolic in INTER;

param exit symbolic in INTER, <> entr ;

set ROADS within (INTER diff {exit}) cross (INTER diff {entr});

param cap {ROADS} >= 0;

var Traff {(i,j) in ROADS} >= 0, <= cap[i,j];

maximize Entering_Traff: sum {(entr,j) in ROADS} Traff[entr,j];

subject to Balance {k in INTER diff {entr,exit}}:

sum {(i,k) in ROADS} Traff[i,k] = sum {(k,j) in ROADS} Traff[k,j];

資料檔：

```
set INTER := a b c d e f g;
```

```
param entr := a;
```

```
param exit := g;
```

```
param: ROADS: cap:=
```

```
    a d    50,  a c    100
```

```
    b d    40,  b e    20
```

```
    c d    60,  c f    20
```

```
    d e    50,  d f    60
```

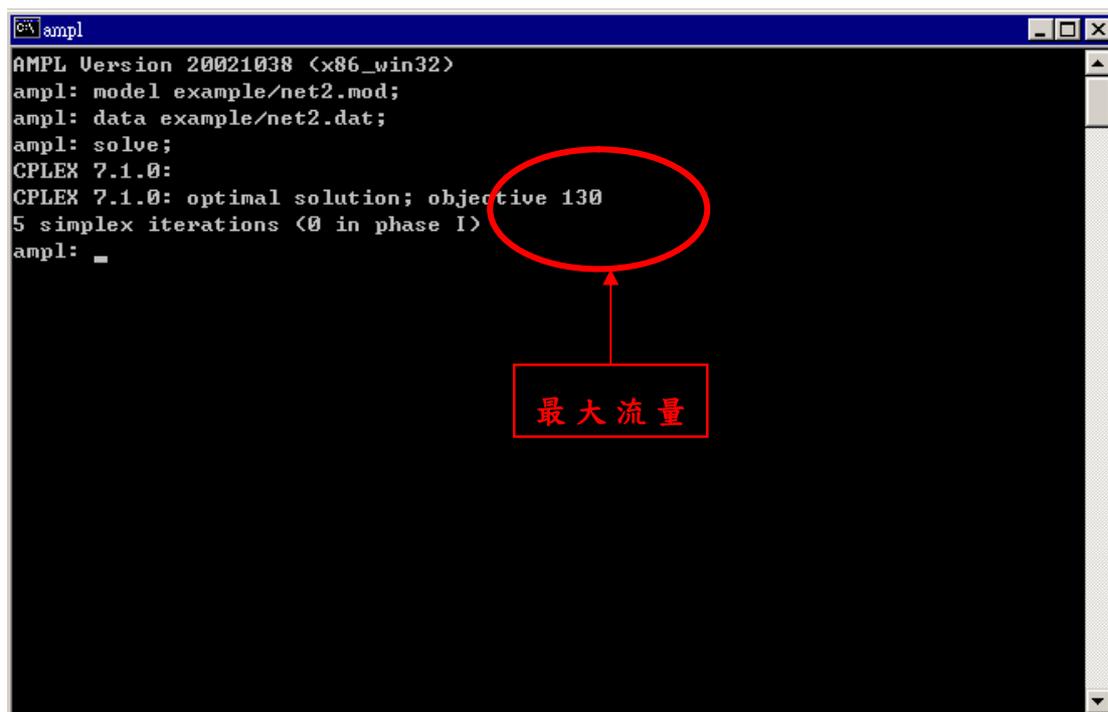
```
    e g    70,  f g    70;
```

執行結果

輸入問題模型。

輸入資料檔。

輸入”solve;”

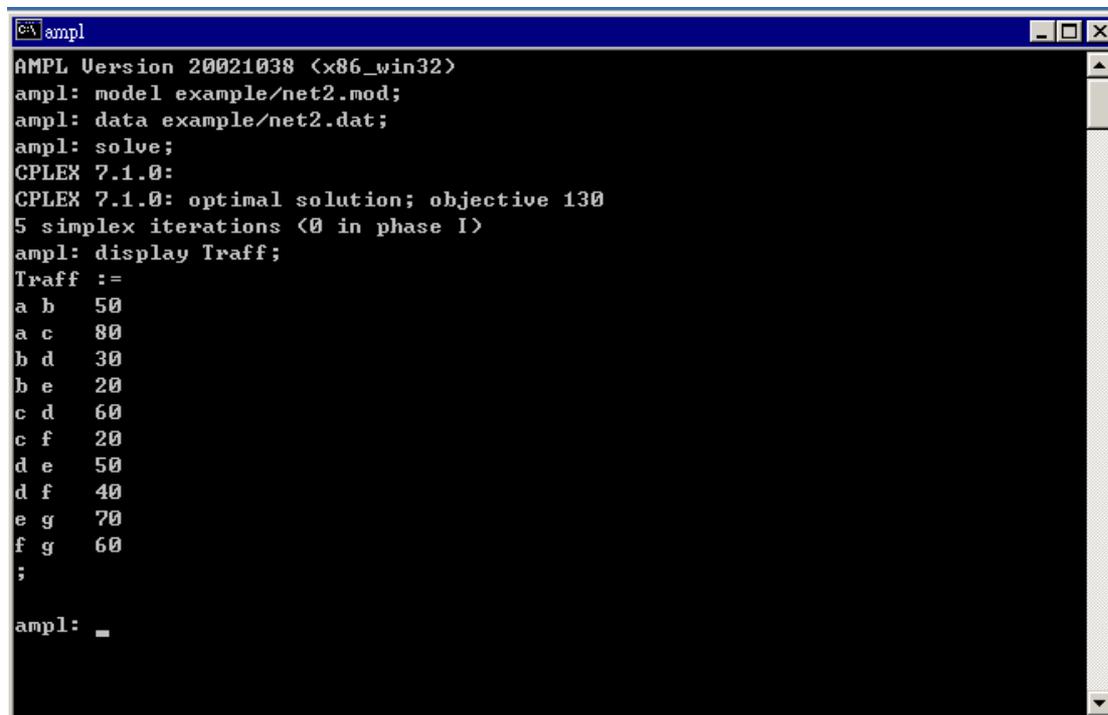


```
ampl
AMPL Version 20021038 (x86_win32)
ampl: model example/net2.mod;
ampl: data example/net2.dat;
ampl: solve;
CPLEX 7.1.0:
CPLEX 7.1.0: optimal solution; objective 130
5 simplex iterations (0 in phase I)
ampl: _
```

圖 24：最大流量運輸問題執行結果

數值分析

經由 AMPL 運算後得知，此題大流量為 130，其每段的流量如圖所示。



```
AMPL Version 20021038 (x86_win32)
ampl: model example/net2.mod;
ampl: data example/net2.dat;
ampl: solve;
CPLEX 7.1.0:
CPLEX 7.1.0: optimal solution; objective 130
5 simplex iterations (0 in phase I)
ampl: display Traff;
Traff :=
a b  50
a c  80
b d  30
b e  20
c d  60
c f  20
d e  50
d f  40
e g  70
f g  60
;
ampl: _
```

圖 25：最大流量運輸問題變數數值

4.3 最短路徑問題

由於交通運輸的便利與普及，所以兩地之間有發生運送或者資訊的傳遞下，最短路徑 (Shortest Path) 的問題隨時都可能會有需求產生。最短路徑是在很多的路徑中，找尋行經距離最短、或者說所花費成本最少的路徑。在本例中，以圖二為例，每線段上的數字代表更路線的長度，試求由 a 到 g 點的最段路徑，並且是由那些線段組成。

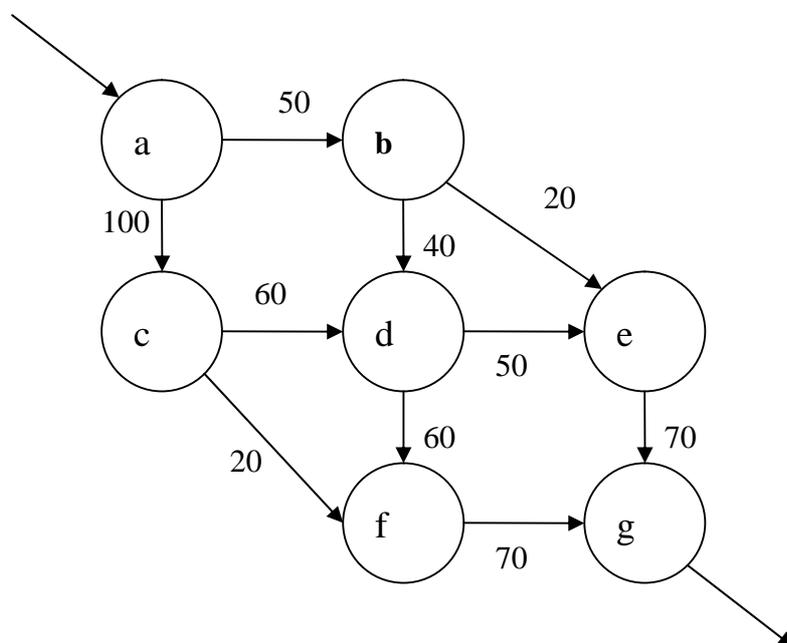


圖 26：最短路徑運輸網路圖

依照圖二的運輸網路圖中，要求得最短路徑問題，只需要修改最大流量中的幾個部分即可。假如路線上代表每線段的距離，所以要設定 2 個參數。

```
param distance {ROADS} >= 0;
```

```
var use {(i,j) in ROADS} >= 0;
```

在目標式方面，改成如下：

```
minimize Total_distance: sum {(i,j) in ROADS} distance[i,j] * use[i,j];
```

在限制式方面，改成如下：

```
subject to Start: sum{(entr,j) in ROADS} use[entr,j]=1;
```

模組：

```
set INTER;
```

```
param entr symbolic in INTER;
```

```
param exit symbolic in INTER, <> entr ;
```

```
set ROADS within (INTER diff {exit}) cross (INTER diff {entr});
```

```
param cap {ROADS} >= 0;
```

```
var Traff {( i,j) in ROADS} >= 0, <= cap[i,j];
```

```
param distance {ROADS} >= 0;
```

```
var use {(i,j) in ROADS} >= 0;
```

```
minimize Total_distance: sum {(i,j) in ROADS} distance[i,j] * use[i,j];
```

```
subject to Start: sum{(entr,j) in ROADS} use[entr,j]=1;
```

```
subject to Balance {k in INTER diff {entr,exit}}:
```

$$\sum \{(i,k) \text{ in ROADS}\} \text{ use}[i,k] = \sum \{(k,j) \text{ in ROADS}\} \text{ use}[k,j];$$

資料檔：

```
set INTER := a b c d e f g;
```

```
param entr := a;
```

```
param exit := g;
```

```
param: ROADS: distance :=
```

```
    a d    50,  a c    100
```

```
    b d    40,  b e    20
```

```
    c d    60,  c f    20
```

```
    d e    50,  d f    60
```

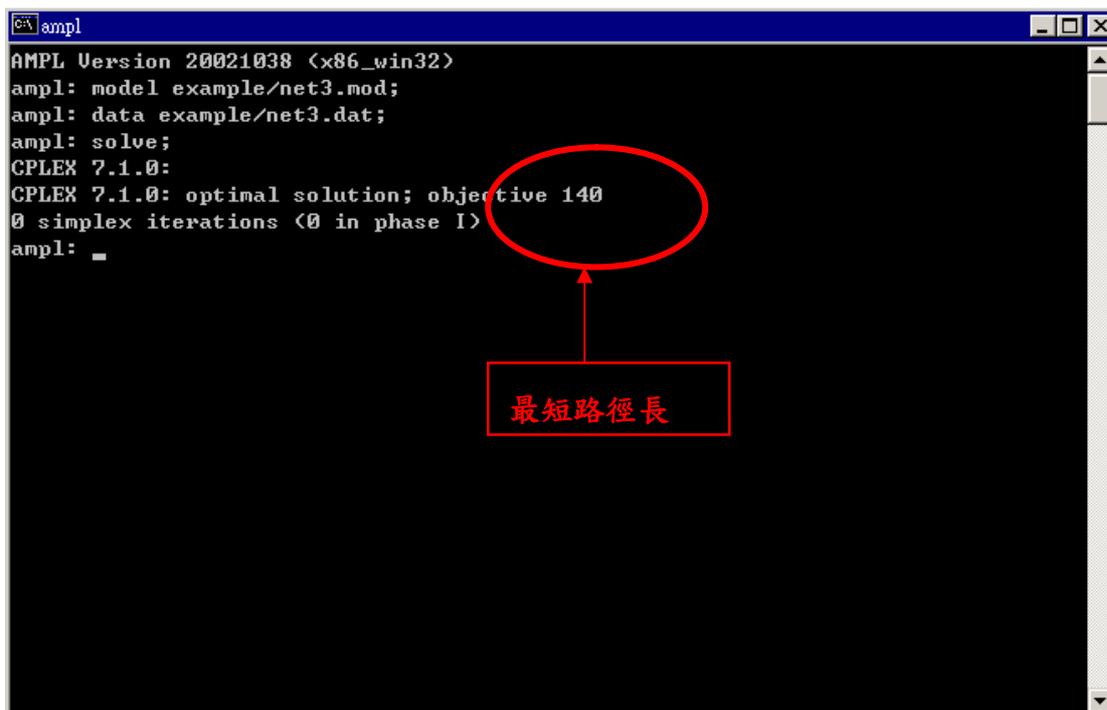
```
    e g    70,  f g    70;
```

執行結果

輸入問題模型。

輸入資料檔。

輸入”solve;”



```
ampl
AMPL Version 20021038 (x86_win32)
ampl: model example/net3.mod;
ampl: data example/net3.dat;
ampl: solve;
CPLEX 7.1.0:
CPLEX 7.1.0: optimal solution; objective 140
0 simplex iterations (0 in phase I)
ampl: _
```

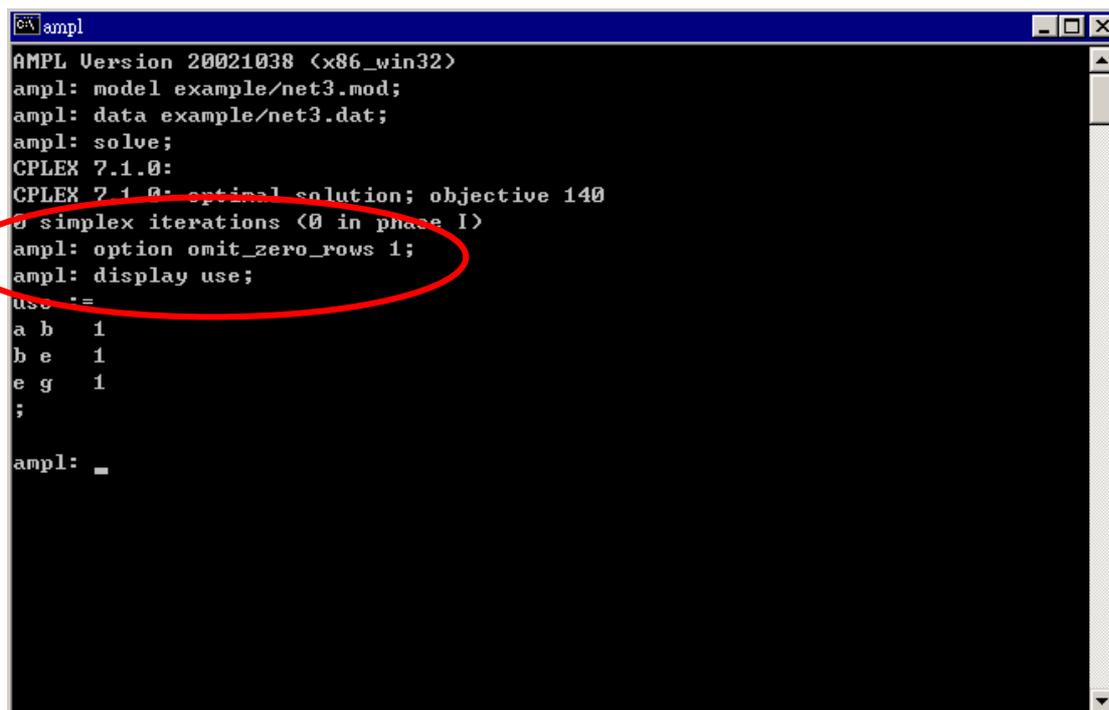
圖 27：最短路徑運輸執行結果

數值分析

```
ampl: option omit_zero_rows 1;    #分行
```

```
ampl: display use;                #顯示數值
```

經由 AMPL 運算後得知，此題最短路徑為 a-b-e-g，總長為 140。



```
ampl
AMPL Version 20021038 (x86_win32)
ampl: model example/net3.mod;
ampl: data example/net3.dat;
ampl: solve;
CPLEX 7.1.0:
CPLEX 7.1.0: optimal solution; objective 140
0 simplex iterations (0 in phase I)
ampl: option omit_zero_rows 1;
ampl: display use;
use ==
a b 1
b e 1
e g 1
;
ampl: _
```

圖 28：最短路徑運輸變數數值

第五章 利用 AMPL/CPLEX 求解整數規劃問題

5.1 0-1 問題

在先前的多樣化產品運輸問題中，是單純的線性問題。所以在這裡做了更多參數的加入，使問題不在是單純的線性問題，本節會已先前的問題加以修改之後，成為整數規劃問題。

由範例 3.多產品運輸問題中，為了要使原先的問題轉成整數規劃問題，在這加入了固定成本，而將原先的運輸成本改由變動成本替代，在編碼中會做不同的更動。

如下所示：

表 7、汽車零件廠各類型零件年需求量

	FRA	DET	LAN	WIN	STL	FRE	LAF
bands	300	300	100	75	650	225	250
coils	500	750	400	250	950	850	500
plate	100	100	0	50	200	100	250

表 8、各類型零件運送成本

(Bands)	FRA	DET	LAN	WIN	STL	FRE	LAF
GARY	30	10	8	10	11	71	6
CLEV	22	7	10	7	21	82	13
PITT	19	11	12	10	25	83	15

(Coils)	FRA	DET	LAN	WIN	STL	FRE	LAF
GARY	39	14	11	14	16	82	8
CLEV	27	9	12	9	26	95	17
PITT	24	14	17	13	28	99	20

(Plate)	FRA	DET	LAN	WIN	STL	FRE	LAF
GARY	41	15	12	16	17	86	8
CLEV	29	9	13	9	28	99	18
PITT	26	14	17	13	31	104	20

表 9、固定成本

	FRA	DET	LAN	WIN	STL	FRE	LAF
GARY	3000	1200	1200	1200	2500	3500	2500
CLEV	2000	1000	1500	1200	2500	3000	2200
PITT	2000	1200	1500	1500	2500	3500	2200

編碼不變動有以下幾項。

```
set ORIG;
```

```
set DEST;
```

```
set PROD;
```

```
param supply {ORIG, PROD } >= 0;
```

```
param demand {DEST, PROD } >= 0;
```

```
check {p in PROD}:
```

```
sum {i in ORIG} supply[i,p] = sum {j in DEST} demand[j,p];
```

```
param limit {ORIG,DEST} >= 0;
```

在成本參數設定中，將原本的成本分成固定與變動兩項，如下所示。

```
param vcost {ORIG,DEST,PROD} >= 0;
```

```
var Trans {ORIG,DEST,PROD } >= 0;
```

vcost 代表變動成本，與運送量(Trans)都必須為正值。

```
param fcost {ORIG,DEST} >= 0;
```

```
var use {ORIG,DEST} binary;
```

fcost 代表固定成本，必須為正值。而 use 是二維變數，1 代表 i 到 j 有通過，0 代表沒有。

在目標是方面，要修改為(變動成本*運送量)+(固定成本*路線是否使用)。

```
minimize Total_cost:
```

$$\begin{aligned} & \text{sum } \{i \text{ in ORIG, } j \text{ in DEST, } p \text{ in PROD}\} \text{vcost}[i,j,p] * \text{Trans}[i,j,p] \\ & + \text{sum } \{i \text{ in ORIG, } j \text{ in DEST}\} \text{fcost}[i,j] * \text{use}[i,j]; \end{aligned}$$

在限制式方面，修改如下。

subject to Supply {i in ORIG, p in PROD }:

$$\text{sum } \{j \text{ in DEST}\} \text{Trans}[i,j,p] = \text{supply}[i,p];$$

subject to Demand {j in DEST, p in PROD }:

$$\text{sum } \{i \text{ in ORIG}\} \text{Trans}[i,j,p] = \text{demand}[j,p];$$

subject to Multi {i in ORIG, j in DEST }:

$$\text{sum } \{p \text{ in PROD}\} \text{Trans}[i,j,p] \leq \text{limit}[i,j] * \text{use}[i,j];$$

資料檔設定：

set ORIG := GARY CLEV PITT;

set DEST := FRA DET LAN WIN STL FRE LAF;

set PROD := bands coils plate;

param supply (tr): GARY CLEV PITT :=

bands 400 700 800

coils 800 1600 1800

plate 200 300 300;

param demand (tr): FRA DET LAN WIN STL FRE LAF:=

bands 300 300 100 75 650 225 250

coils 500 750 400 250 950 850 500

plate 100 100 0 50 200 100 250;

param limit default 625;

param vcost :=

[*,*,bands]: FRA DET LAN WIN STL FRE LAF :=

GARY 30 10 8 10 11 71 6

CLEV 22 7 10 7 21 82 13

PITT 19 11 12 10 25 83 15

[*,*,coils]: FRA DET LAN WIN STL FRE LAF :=

GARY 39 14 11 14 16 82 8

CLEV 27 9 12 9 26 95 17

PITT 24 14 17 13 28 99 20

[*,*,plate]: FRA DET LAN WIN STL FRE LAF :=

GARY 41 15 12 16 17 86 8

CLEV 29 9 13 9 28 99 18

PITT 26 14 17 13 31 104 20;

param fcost: FRA DET LAN WIN STL FRE LAF :=

GARY 3000 1200 1200 1200 2500 3500 2500

CLEV 2000 1000 1500 1200 2500 3000 2200

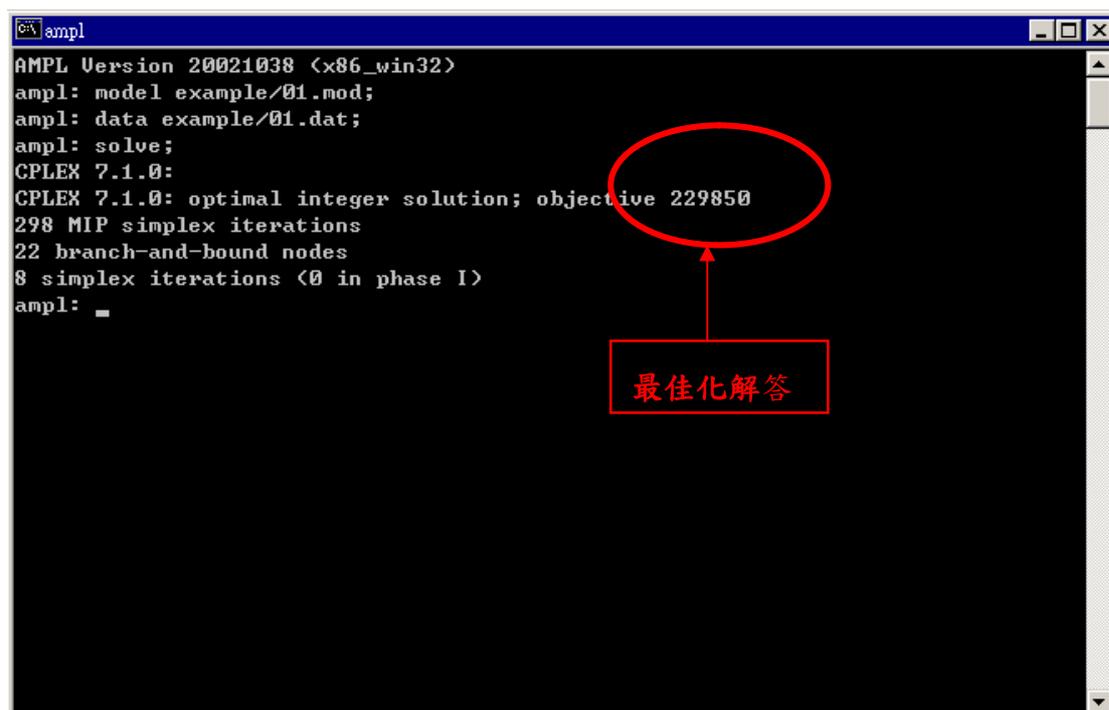
PITT 2000 1200 1500 1500 2500 3500 2200;

執行結果

輸入問題模型。

輸入資料檔。

輸入”solve;”



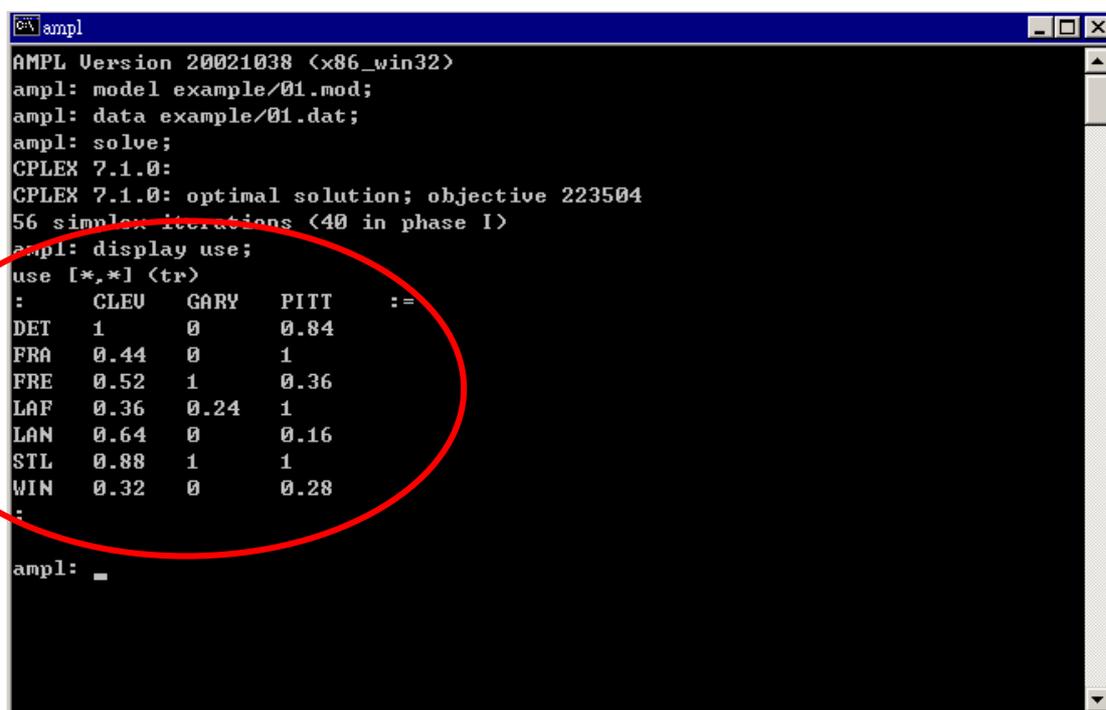
```
AMPL Version 20021038 (x86_win32)
ampl: model example/01.mod;
ampl: data example/01.dat;
ampl: solve;
CPLEX 7.1.0:
CPLEX 7.1.0: optimal integer solution; objective 229850
298 MIP simplex iterations
22 branch-and-bound nodes
8 simplex iterations (0 in phase I)
ampl: _
```

圖 29：整數規劃問題執行結果

數值分析

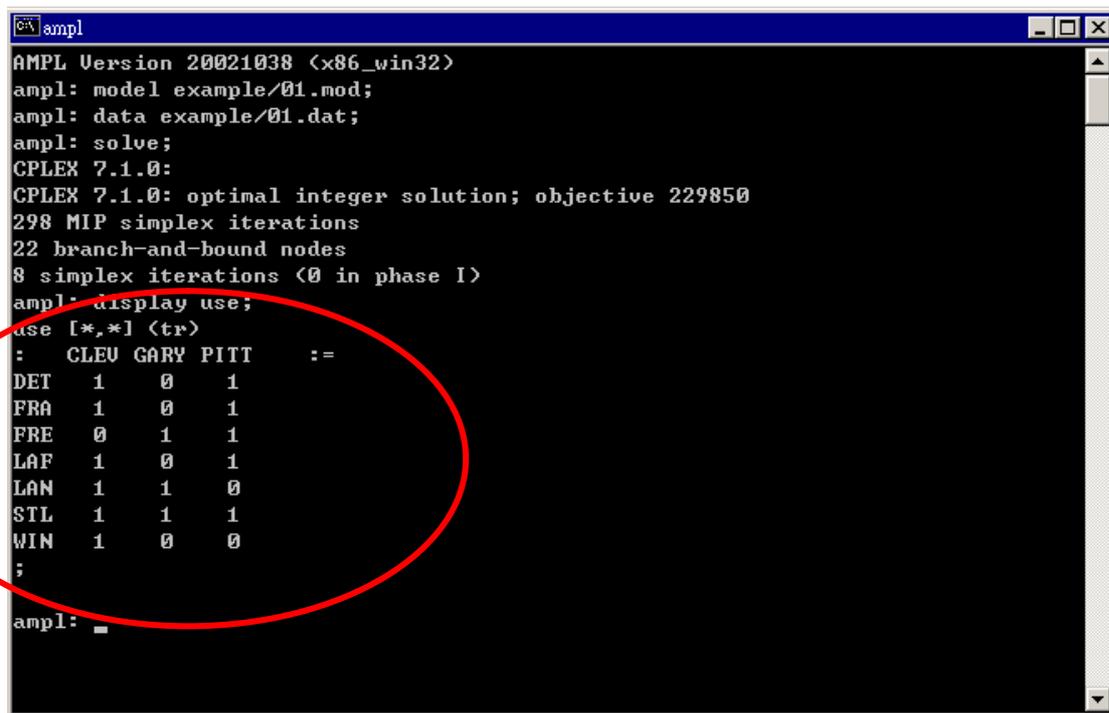
由於在路線決策中，只能有 0 或 1 兩種可能，但是在圖一中，路線使用並不是整數，這是在 `var use {ORIG,DEST} <=1, >=0;`，這段編碼中並未改成 `var use {ORIG,DEST} binary;`。

`binary` 可讓原本的數值轉換成只能是 0 或 1，經由這種表達，即可分辨哪條路線是有被使用，如圖二所示。



```
AMPL Version 20021038 (x86_win32)
ampl: model example/01.mod;
ampl: data example/01.dat;
ampl: solve;
CPLEX 7.1.0:
CPLEX 7.1.0: optimal solution; objective 223504
56 simplex iterations (40 in phase I)
ampl: display use;
use [*,*] (tr)
:      CLEU   GARY   PITT   :=
DET    1      0      0.84
FRA    0.44   0       1
FRE    0.52   1       0.36
LAF    0.36   0.24    1
LAN    0.64   0       0.16
STL    0.88   1       1
WIN    0.32   0       0.28
:
ampl: =
```

圖 30：未整數規劃下結果



```
ampl
AMPL Version 20021038 (x86_win32)
ampl: model example/01.mod;
ampl: data example/01.dat;
ampl: solve;
CPLEX 7.1.0:
CPLEX 7.1.0: optimal integer solution; objective 229850
298 MIP simplex iterations
22 branch-and-bound nodes
8 simplex iterations (0 in phase I)
ampl: display use;
use [*,*] (tr)
:   CLEU GARY PITT   :=
DET  1    0    1
FRA  1    0    1
FRE  0    1    1
LAF  1    0    1
LAN  1    1    0
STL  1    1    1
WIN  1    0    0
;
ampl: _
```

圖 31：整數規劃下結果

參考文獻

Robert Fourer, David M. Gay, and Brian W. Kernighan. (2002). AMPL: a modeling language for mathematical programming, 2nd Edition. Duxbury Press.