朝陽科技大學
資訊工程系

碩士論文

組合雙向拍賣之決策支援

Decision Support For Combinatorial Double Auctions

指導教授：謝富雄 博士

研 究 生：黃仲煒

中華民國 100 年 1 月 27 日

朝陽科技大學資訊工程系

Department and Graduate Institute of Information and Communication Engineering

Chaoyang University of Technology

碩士論文

Thesis for the Degree of Master

組合雙向拍賣之決策支援

Decision Support For Combinatorial Double Auctions

指導教授：謝富雄 (Fu-Shiung Hsieh)

研 究 生：黃仲煒 (Chung-Wei Huang)

中華民國 100 年 1 月 27 日

January 27， 2011

中文摘要

組合式雙向拍賣是使眾多的買賣雙方能夠在一次的拍賣過程中，依照自己的喜愛、偏好，針對不同的商品組合去競標。他們能夠一次選取多樣物品，並且給予這些物品的組合一個金額。改善了傳統只能針對單一商品進行競標的效率。組合式雙向拍賣的目的在讓買賣雙方以合理的價格同時採購或者賣出多樣商品。組合式雙向拍賣最重要的問題就是要在所有的投標中選擇買家與賣家的得標者。本論文使用了 Lagrange 鬆弛法來解決最佳化問題，有效率地找出組合式雙向拍賣的獲勝者。為了驗證所發展的方法，也建置了一個在 NetBeans 的開發平台上，並且運用 Web service 技術的雛型系統。透過此雛型系統來驗證組合式雙向拍賣演算法的效能。主要的結果包括：1.一個組合式雙向拍賣問題的演算法，2.利用 Lagrangian Relaxation 的方法解決此問題，3.利用提出的演算法進行數據的分析，4.使用 Web service 技術結合 NetBeans 平台進行驗證模擬確保例子的效能。

關鍵字：組合式雙向拍賣、最佳化問題、Lagrangian 鬆弛法、整數規劃問題、電子商務

Abstract

In combinatorial double auctions, a bidder can bid on a combination of goods with one limit price for the total combination. This improves the efficiency when the procurement of one good is dependent on the acquisition of another. Combinatorial double auctions in which both sides submit demand or supply bids are much more efficient than several one-sided auctions combined. However, combinatorial double auctions are notoriously difficult to solve from computation point of view. In this paper, we formulate the combinatorial double auction problem and propose an algorithm for finding near optimal solutions by applying the Lagrange Relaxation method. To verify the method developed， we also develop a prototype system. The main results include： (1) a problem formulation for the Combinatorial Double Auctions problem, (2) a solution methodology based on Lagrangian Relaxation and (3) analysis of numerical results based on our solution algorithms.

Keywords ： Combinatorial Double Auctions, Optimization, Lagrangian Relaxation, integer programming, e-commerce

致謝

在朝陽的日子以來，首先感謝謝富雄老師這兩年半在學術上面的教導，使我在資訊系的專業知識和學術研究上受益良多。也感謝國科會計畫編號 NSC-97-2410-H-324-017-MY3 給予本人研究上的資助，外加從研究所生活上也感受到有如業界般的獨立作業、學習，讓我的能力不斷的獲得提昇。

此外，也要感謝實驗室的學長世民及正中，在相同領域上給了我指點及指導。尤其是正中學長，在相同領域之下，給了我一些程式撰寫的概念及教學以及一些閱讀論文的方向跟技巧，讓我解決許多程式及閱讀論文上面的問題。也感謝研究室跟我一起走過來的學長、同學和學弟，讓我在研究所的生活過的非常充實，有許多寶貴的回憶。

最後，我要感謝我的家人給我這個機會讀研究所，這段期間給了我最大的支持與鼓勵，要我不放棄的一直往上爬，讓我能順利的獲得研究所的文憑及學業，在學習上劃上個逗號，也要準備迎接社會上新的磨練及新的學習。

黃仲煒 謹識

中華民國 100 年 1 月 27 日

索引目錄

圖目錄

表目錄

第一章 緒論

1.1 研究背景與動機

最近，在電子商務的爆發性成長下，網路拍賣被大眾投以極大的關注。而傳統的拍賣是單向拍賣機制，由多個買家來競標購買一個賣家的商品。就拍賣的買家、賣家參與者的人數來分類，組合式拍賣可以分為單向拍賣以及雙向拍賣。在單向拍賣中只有單一個賣家對多個買家進行交易，或是單一個買家對多個賣家進行交易，而在雙向拍賣中有多個買家對多個賣家進行交易。組合式拍賣[1][10][11][13][15][20][25]讓參與者可以在單一次的拍賣活動中可同時針對多樣商品進行投標及招標，改善了傳統只能針對單一商品進行競標的效率。

組合式雙向拍賣是使得眾多的買賣雙方能夠在一次的拍賣過程中，依照自己的喜愛、偏好，針對不同種類的商品來進行組合並且競標。他們能夠一次選取多樣物品，並且給予這些商品不同種類的組合一定的金額。組合式雙向拍賣的目的在讓買賣雙方以合理的價格採購或者售出多樣商品。近年來雖然組合式拍賣受到許多學者的注意，但是大部分的研究集中於單一買家和多位賣家進行拍賣的問題。

由於目前關於組合是雙向拍賣的研究仍然不多，因此在本論文將研究組合式雙向拍賣。主要的結果包括：(1)建立組合式雙向拍賣中的數學化模式，(2)利用 Lagrangian Relaxation 的方法論來解決最佳化的問題，(3)提出

解決優勝者決定問題(Winner Detemination Problem,WDP)的演算法，(4)根據結果進行效能。

藉由運用 Lagrangian Relaxation 技術，讓原本的問題可以分解成若干個投標者的子(Bidder's Subproblem)問題，使得問題可以以更有效率的方式來解決，並且在可接受的 CPU 時間內得到計算的結果，演算法經過實驗數據的分析所得到的結果可以產生近似佳解。

1.2.文獻回顧

在現有的文獻中已經有許多有關於組合式拍賣的研究，比較眾所皆知的組合式拍賣例子就是美國聯邦通信委員會 Federal Communications Commission's (FCC)的無線頻譜拍賣，而且此類型的拍賣可以運用在機場時間、鐵路區段[19]以及運輸路線[4]的分配上。就問題的特性以及複雜度而言，組合拍賣和 Set Packing/ knapsack Problem(SPP)包裝/背包問題是密切相關的[21]，可以將組合拍賣的問題轉換為 Set Packing/knapsack Problem (SPP)，而此問題也是一個有名的 NP-complete 問題[2][9][23][24][26]。

許多有關解決組合式雙向拍賣的研究在求解分為兩種，一種是花較長的時間來處理困難的計算並找出其最佳的解，另外一種則是可以快速和希望能找出一個可行的近似解。所以解 SPP 問題對於時間上的要求，會將方法分成兩類型，一種是精確方法(Exact Methods)，另一種是近似方法

(Approximate Methods)。近似解方法所求得的解，必須要探討的就是近似解和最佳解之間是否接近。

已經有很多研究有關組合式拍賣的近似解演算法。Fu-Shiung Hsieh 跟 Shih-Min Tsai[7]研究如何運用近似解演算法來解決組合式反向拍賣[6][8]的問題。組合式反向拍賣是將拍賣應用在採購商品的一種商業模式，目的在讓買家以最低的價格同時採購到所需求的多樣商品，在文獻[7]中有提出使用 Lagrangian Relaxation 的方法及運用 Subgradient 最佳化技術來解決最佳化分配問題和 NP-complete 的問題以有效地尋找到近似解。而在本論文中也是使用同樣的方法及技術以有效率地找尋近似的解。Xia、Koehler 及 Whinston[17]這三位專家所提的論文中認為單向拍賣有許多不錯的特質，像機制的存在以確保最優性，刺激性協調和市場清算的價格，有不少論文提供刺激性協調和估算、個別價格的方法，而這三位專家發現這些方法之間的關係，並分析其優勢與劣勢。

在雙向拍賣中有多個買家與賣家參與競標，每位參與競標者可以針對不同商品的偏好用效用函數(Utility Function)來表示。如何定義每位參與競標者的效用函數是很重要的議題。在大部份過去有關雙向拍賣的研究中，都是假設買家與賣家的效用函數是線性函數或是近似線性函數，而且假設買家與賣家的效用函數是相同的。而在現實中買家與賣家的效用函數是不同的。因此 Jin Ho Choi， Hyunchul Ahn， Ingoo Han 三位作者於文獻[12]

中提出以效用為基礎的雙向拍賣機制(Utility-based double auction， UDA)。在此機制中，買家可以給定價格上限去購買他所期望的商品，而相對賣家也可以擬定價格下限去出售他們的商品。除此之外，商品的數量都必須符合買家與賣家的條件。而文獻[12]並未提出有關於組合式雙向拍賣的問題。本研究最大的不同就是除了能夠設定價格上下限制以外還能夠考慮買賣雙方的商品組合，讓賣家的供給量盡可能的滿足買家的需求量，並且使得他們能夠以雙方最理想的商品進行交易，將交易成功的機率大幅提昇，而交易失敗的機率降低。

1.3 論文架構

本論文共分為六個章節，各章節編排如下

第一章 研究動機及方法簡介

主要在於說明本篇論文的背景、動機、方法及目的的介紹。

第二章 問題描述

描述提出一個組合式雙向拍賣問題。

第三章 解決優勝者決定問題(WDP)的演算法方法說明

說明 Lagrangian 和 Subgradient 及利用 Lagrangian Relaxation 的方法來解組合式雙向拍賣。

第四章 模擬驗證

建置一個雛型系統來驗證所提出的方法。

第五章 實驗結果與分析

針對不同的例子(變數 I、N、K 的設計)去探討近似解與最佳解的百分比差距，以及執行時間的變化趨勢。

第六章 結論與未來展望

本論文的研究結果歸納出來的結論，以及未來研究方向。

第二章 問題描述

2.1 組合式雙向拍賣

組合式雙向拍賣最重要的問題就是要在所有的投標中選擇買家與賣家的得標者。為了運用最佳化的方法來發展解決組合式雙向拍賣問題的演算法，在本文中，首先建立以組合式雙向拍賣問題的數學化模式，將組合式雙向拍賣的最佳化問題公式化成一個整數規劃問題。然後會將此問題利用 Lagrangian Relaxation 的方法來解所提出的問題。

本論文使用了 Lagrangian Relaxation 來解決最佳化問題，有效率地找出組合式雙向拍賣的獲勝者。也建置了一套雛型系統來驗證所提出的方法。先前談到的 WDP 是一種最佳整數規劃的問題，而使用的 Lagrangian Relaxation 來解決 WDP，可以看到以下公式將 WDP 以數學模式來進行公式化。

2.2 數學模式

Winner Detemination Problem(WDP) 數學模式：

$$Max\left[\sum_{i=1}^{I}\sum_{j=1}^{n_i} x_{ij}(p_{ij}-p_{ij}^{WILL})+\sum_{n=1}^{N}\sum_{h=1}^{n_n} y_{nh}(p_{nh}^{WILL}-p_{nh})\right].....(2-1)$$

$$s.t.\sum_{i=1}^{I}\sum_{j=1}^{n_i} x_{ij}q_{ijk} \geq \sum_{n=1}^{N}\sum_{h=1}^{n_n} y_{nh}d_{nhk} \quad \forall k=1,2,3,...,K.....(2-2)$$

s

$$\sum_{j=1}^{n_i} x_{ij} \leq 1 \quad \forall i=1,...,I \;\&\; \sum_{h=1}^{n_n} y_{nh} \leq 1 \quad \forall n=1,2,3,...,N.....(2-3)$$

$$x_{ij} \;\&\; y_{nh} \in \{0,1\}.....(2-4)$$

優勝者決定問題(WDP)所定義的參數：

i = 賣家，n = 買家，j = 賣家 i 投標 j 次，h = 買家 n 招標 h 次。

n_i = 賣家 i 的投標，n_n = 買家 n 的招標。

x_{ij} = 賣家 i 對第 j 次投標交易是否成功的決策變數。

y_{nh} = 買家 n 對第 h 次招標交易是否成功的決策變數。

p_{ij} = 賣家 i 於第 j 次投標的價格，p_{nh} = 買家 n 於第 h 次投標的價格

p_{ij}^{WILL} = 賣家 i 接受第 j 次投標商品最小賣出的價格。

p_{nh}^{WILL} = 買家 n 接受第 h 次招標商品最大買進的價格。

q_{ijk} = 賣家 i 對第 j 次投標所提供商品 k 的量。

d_{nhk} = 買家 n 對第 h 次招標所需求商品 k 的量。

首先介紹此數學模式的第一行公式設定值為 *Max* 是因為買家及賣家都希望買賣雙方所獲得的利潤值為最大，好比以賣家來舉例，假如以賣家角度來思考會以最高售價為考量而來獲得最高利潤值，若以買家角度來思考會以最低價格值來購買到最期望的商品量。

公式(2-1)所表明賣家 i 的總價格減去他所期望以最低成本出售商品的價格即為其獲利，以買家 n 的觀點來說他所期望購買的最高價格減去真正成交的價格即為買家的利潤，而其中的 x_{ij} 為賣家 i 的決策變數，也就是說如果 $x_{ij}=1$ 來說就是賣家成功賣出他的商品，而假如 $x_{ij}=0$ 代表來說就是賣家不能成功賣出其商品，另外說法則是 $x_{ij}=1$ 為交易成功，然則 $x_{ij}=0$ 為交易失敗。

而其中的y_{nh}為買家n的決策變數，用來決策買家如果$y_{nh}=1$表示交易成功，若是$y_{nh}=0$表示交易失敗。

在(2-2)公式中左邊q_{ijk}為第i個賣家於第j次投標所投標的第k項商品的數量，右邊d_{nhk}為第n個買家於第h次招標所招標的第k項商品的數量。公式(2-2)很明顯可以看出賣家i所提供的商品數量必須要滿足買家n所想要購買商品的數量。則其中以k來表示商品的種類。以(2-4)公式可以看出決策變數x_{ij}與y_{nh}不是 1 就是 0，依據上述例子可以很明顯看出此(2-3)公式只能小於或是等於 1，而不能大於 1。

第三章 解決 WDP 的演算法

3.1 Lagrangian Relaxation

Lagrangian Relaxation 是一種鬆弛的技術[14]，將演算法中困難的限制式移到目標式裡面，如果目標是沒有滿足的話，則會確切的懲罰目標式。Lagrangian Relaxation 問題所求得的最佳解的成本將不會小於原始問題所求得最佳解的成本。

Lagrangian Relaxation 就是面對基礎的數學模式，利用「鬆弛(Relaxation)」來解題的策略。將此策略與拍賣問題的演算法一起搭配使用，可以縮短求解時間並且找出一個最佳的近似解。綜合得知，Lagrangian Relaxation 的優點如下：

1.降低問題複雜性：透過鬆弛技術，將限制式移動到目標函數在加上一個懲罰的項目，與原本問題比較起來就少了一個限制式，整個問題看起來就簡單多了。

2.較快的速度：雖然 Lagrangian Relaxation 所求得的解為一個近似解，但是與精確演算法的處理速度比起來，執行效率上快上很多，如果將問題放更大(增加複雜度)，時間上更是有更大的差距。

3.2 Subgradient Method

Subgradient 方法是一個簡單的演算法[22]，減少 non-differentiable convex function。此方法非常類似可微函數的普通 gradient 方法，但有幾個

著名的例外。不同於普通的 gradient 方法，Subgradient 方法不是下降方法：函數值都可以(而且經常)增加。

Subgradient 方法的速度比 Newton's 方法慢，但是更加簡單的，並且可應用於更廣泛的各種問題。結合 Subgradient 方法與原始或雙重分解技術，有時可能將問題制定成一個簡單的分布式演算法。Subgradient 方法起初是在 1970 年的前蘇聯由 N.Z.Shor 這位作者所著作的。一些 Subgradient 方法的基本參考可以在他的書[18]中找到以及另一本書的題目是“Topics in Relaxation and Ellipsoidal Methods”[16]，而在 Subgradient 方法中，Bertsekas[5]這位作者的書也是個不錯的參考。

3.3 Solution Algorithm

接下來利用 Lagrangian Relaxation 演算法代入到所提出的組合式雙向拍賣問題中，以下公式為提出的問題：

$$Max\left[\sum_{i=1}^{I}\sum_{j=1}^{n_i} x_{ij}(p_{ij}-p_{ij}^{WILL})+\sum_{n=1}^{N}\sum_{h=1}^{n_n} y_{nh}(p_{nh}^{WILL}-p_{nh})\right].....(2-1)$$

$$s.t.\sum_{i=1}^{I}\sum_{j=1}^{n_i} x_{ij}q_{ijk} \geq \sum_{n=1}^{N}\sum_{h=1}^{n_n} y_{nh}d_{nhk} \quad \forall k=1,2,3,...,K.....(2-2)$$

$$\sum_{j=1}^{n_i} x_{ij} \leq 1 \quad \forall i=1,...,I \;\&\; \sum_{h=1}^{n_n} y_{nh} \leq 1 \quad \forall n=1,2,3,...,N.....(2-3)$$

$$x_{ij} \;\&\; y_{nh} \in \{0,1\}.....(2-4)$$

要解決 WDP 最佳解的方式是使用 Lagrangian Relaxation 方式來給予一個拉式乘數(Lagrangian Multiplier)λ，此 Lagrangian Relaxation 用於限制(2-2)

分解原始問題，為一個投標者子問題(Bidder's Subproblem 簡稱 BS)的數量。而此 BS 可以解決獨立性的問題。

此 Lagrangian Relaxation 方法結果在 BS 與一個高度分散的決策結構。在 BS 問題之中的相互反應是透過拉式乘數(Lagrangian Multiplier)所反映的，其決定取決於解決 NP-Complete 問題中會發生的對偶問題(Dual Problem)：「資源有限，求最高利潤」 及 「指定產量，求最低成本」，如下公式就是經過 Lagrangian Relaxation 後的原問題：

$$L(\lambda) = Max\left[\sum_{i=1}^{I}\sum_{j=1}^{n_i} x_{ij}(p_{ij} - p_{ij}^{WILL}) + \sum_{n=1}^{N}\sum_{h=1}^{n_n} y_{nh}(p_{nh}^{WILL} - p_{nh})\right] + \sum_{k=1}^{K}\lambda_k(\sum_{n=1}^{N}\sum_{h=1}^{n_n} y_{nh}d_{nhk} - \sum_{i=1}^{I}\sum_{j=1}^{n_i} x_{ij}q_{ijk}$$

$$= Max\left[\sum_{i=1}^{I}\sum_{j=1}^{n_i} x_{ij}(p_{ij} - p_{ij}^{WILL}) + \sum_{n=1}^{N}\sum_{h=1}^{n_n} y_{nh}(p_{nh}^{WILL} - p_{nh}) + \sum_{k=1}^{K}\sum_{n=1}^{N}\sum_{h=1}^{n_n}\lambda_k y_{nh}d_{nhk} - \sum_{k=1}^{K}\lambda_k(\sum_{i=1}^{I}\sum_{j=1}^{n_i} x_{ij}q_{ijk})\right]$$

$$= Max\left[\sum_{i=1}^{I}\sum_{j=1}^{n_i} x_{ij}(p_{ij} - p_{ij}^{WILL}) + \sum_{n=1}^{N}\sum_{h=1}^{n_n} y_{nh}(p_{nh}^{WILL} - p_{nh}) + \sum_{k=1}^{K}\sum_{n=1}^{N}\sum_{h=1}^{n_n}\lambda_k y_{nh}d_{nhk} - \sum_{k=1}^{K}\sum_{i=1}^{I}\sum_{j=1}^{n_i}\lambda_k x_{ij}q_{ijk}\right]$$

$$= Max\left\{\sum_{i=1}^{I}\sum_{j=1}^{n_i} x_{ij}\left[(p_{ij} - p_{ij}^{WILL}) - \sum_{k=1}^{K}\lambda_k q_{ijk}\right] + \sum_{n=1}^{N}\sum_{h=1}^{n_n} y_{nh}\left[(p_{nh}^{WILL} - p_{nh}) + \sum_{k=1}^{K}\lambda_k d_{nhk}\right]\right\}$$

$$= Max\left\{\sum_{i=1}^{I}\sum_{j=1}^{n_i} x_{ij}\left[(p_{ij} - p_{ij}^{WILL}) - \sum_{k=1}^{K}\lambda_k q_{ijk}\right] + Max\sum_{n=1}^{N}\sum_{h=1}^{n_n} y_{nh}\left[p_{nh}^{WILL} - p_{nh}) + \sum_{k=1}^{K}\lambda_k d_{nhk}\right]\right\}$$

$$s.t.\sum_{j=1}^{n_i} x_{ij} \le 1 \quad \forall i = 1,2,3,...,I \;\&\; \sum_{h=1}^{n_n} y_{nh} \le 1 \quad \forall n = 1,2,3,...,N,\; x_{ij} \;\&\; y_{nh} \in \{0,1\}$$

而可以看到以上公式添加了 λ_k 並以暴力拆解的方式進行拆解。此方法包涵了以下三個遵循部份用於解決投標者子問題(BS)的演算法。

(1)給一個 λ，投標者子問題 $L_i(\lambda)$ 及 $L_n(\lambda)$ 的最佳解可以如下解決：

$$x_{ij}=\begin{cases}1\ if\ (p_{ij}-p_{ij}^{WILL})-\sum_{k=1}^{K}\lambda_k q_{ijk}\geq 0\\ 0\ if\ (p_{ij}-p_{ij}^{WILL})-\sum_{k=1}^{K}\lambda_k q_{ijk}<0\end{cases}\ \&\ y_{nh}=\begin{cases}1\ if\ (p_{nh}^{WILL}-p_{nh})+\sum_{k=1}^{K}\lambda_k d_{nhk}\geq 0\\ 0\ if\ (p_{nh}^{WILL}-p_{nh})+\sum_{k=1}^{K}\lambda_k d_{nhk}<0\end{cases}$$

(2)用於解決對偶問題的 Subgradient 方法：

讓 x^l, y^l 對於 Subgradient Method 的近似解來給予反覆式(iteration)的拉式乘數(Lagrangian Multiplier) L^l，讓 x^l, y^l 為子問題的最佳解而將 $\lambda^l{}_k$ 去微分得到 $g^l{}_k$：

$$g_k^l=\frac{\partial L(\lambda)}{\partial \lambda_k}\ \Big|\ Subject.to:\ \lambda_k^l=\sum_{n=1}^{N}\sum_{h=1}^{n_n}y_{nh}d_{nhk}-\sum_{i=1}^{I}\sum_{j=1}^{n_i}x_{ij}q_{ijk}\ \text{where}\ k\in\{1,2,3,...,K\}$$

此 Subgradient Method 是被 Polyak[3]提出並且遵循更新成如下公式：

$$\lambda_k^{l+1}=\begin{cases}(\lambda_k^l+\alpha^l g_k^l)/c\ \ if\ \ \lambda_k^l+\alpha^l g_k^l\geq 0;\\ 0\qquad otherwise.\end{cases}\quad \alpha^l=c\frac{\bar{L}-L(\lambda)}{\sum_k (g_k^l)^2}\qquad 0\leq c\leq 2$$

而 $\bar{L}$ 是一個估計的最佳化對偶成本。如果 α^l 小於門檻值的話，則整個步驟就停止。Polyak[3]證明該方法具有線性收斂。反覆應用於步驟(1)及步驟(2)的演算法可能收斂到最佳對偶解。

(3)從被鬆弛後的問題所得到的解 $((x^*,y^*),\lambda^*)$ 後，使用 Heuristic 演算法來找出一個最佳解 $\bar{x},\bar{y}$，由於鬆弛所得到的解 $((x^*,y^*),\lambda^*)$，可能會導致一種違反限制式條件的狀況，而 Heuristic Method 第一件事情就先檢查所有需求的條件限制：$\sum_{i=1}^{I}\sum_{j=1}^{n_i}x_{ij}q_{ijk}\geq\sum_{n=1}^{N}\sum_{h=1}^{n_n}y_{nh}d_{nhk}\quad\forall k=1,2,3,...,K$ 使否有滿足限制條件。讓 $K^0=\{k\,|\,k\in\{1,2,3,...,K\},\sum_{i=1}^{I}\sum_{j=1}^{n_i}x_{ij}q_{ijk}<\sum_{n=1}^{N}\sum_{h=1}^{n_n}y_{nh}d_{nhk}\}$，$K^0$ 表示違反需求限制的

集合。讓 $I^0 = \{i | i \in \{1,2,3,...,I\}, x_{ij}^* = 0\}$，$I^0$ 表示投標者於解法 x^* 並不是獲勝者，及讓 $N^0 = \{n | n \in \{1,2,3,...,N\}, y_{nh}^* = 0\}$，$N^0$ 表示投標者於解法 y^* 並不是獲勝者，而為了讓 K^0 限制式獲得滿足，那第一選擇 $k \in K^0$ 及 $k = \arg\max_{k \in K^0} \sum_{n=1}^{N} \sum_{h=1}^{n_n} y_{nh}^* d_{nhk} - \sum_{i=1}^{I} \sum_{j=1}^{n_i} x_{ij}^* q_{ijk}$ 而此 Heuristic 演算法所獲得的收益做出的滿足限制條件如下：

選擇 $i \in I^0$ 及 $i = \arg\max_{i \in \{1,2,3,...,I\}, q_{ijk} > 0} (p_{ij} - p_{ij}^{WILL})$ 並且設置 $x_{ij}^* = 1$，而選擇 $n \in N^0$ 及 $n = \arg\max_{n \in \{1,2,3,...,N\}, d_{nhk} > 0} (p_{nh}^{WILL} - p_{nh})$ 並且設置 $x_{nh}^* = 1$。(Arg max 是一種數學術語，意即一系列變數中取其使函數為最大值者)，在執行上述操作之後，的集合 $I^0 \leftarrow I^0 \setminus \{i\}$、$N^0 \leftarrow N^0 \setminus \{n\}$，如果限制式中第 k 項商品還無法完全滿足限制條件的話，將重複步驟(3)，直到限制式滿足的狀態才結束，否則不能完全解決於重複同樣的程序。而使用 $\bar{x}, \bar{y}$ 來表示結果從上述的 Heuristic 演算法所獲得的一個可行解。

第四章 模擬驗證

在本研究中也建置了一個雛型系統來驗證所提出的方法。此雛型系統是在 NetBeans 的開發平台上發展，並運用 Web service 技術來設計。將可以透過此平台來驗證及操作組合式雙向拍賣整體的運作流程。以下來介紹此雛型系統的模擬驗證。

此雛型系統所完成的初始介面可以於圖 4-1 看出，一開始使用已經完成註冊的買家及賣家的帳號登入系統，登入後顯示如圖 4-2 的畫面，可以看到左邊原本只有註冊以及登入的列表更新成只有登入系統才能使用的一些功能，接下來先輸入好一些有關買賣雙家的需求資訊，資訊輸入完成之後使用在上述所提到的解法進行運算，在圖 4-3 及圖 4-4 可以看出已經設定好買家的招標以及賣家所投標的商品。

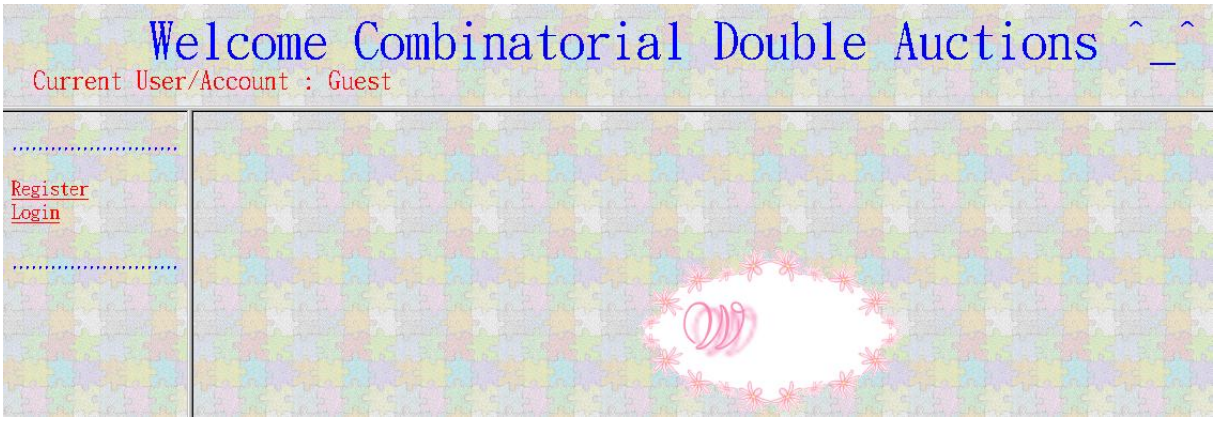
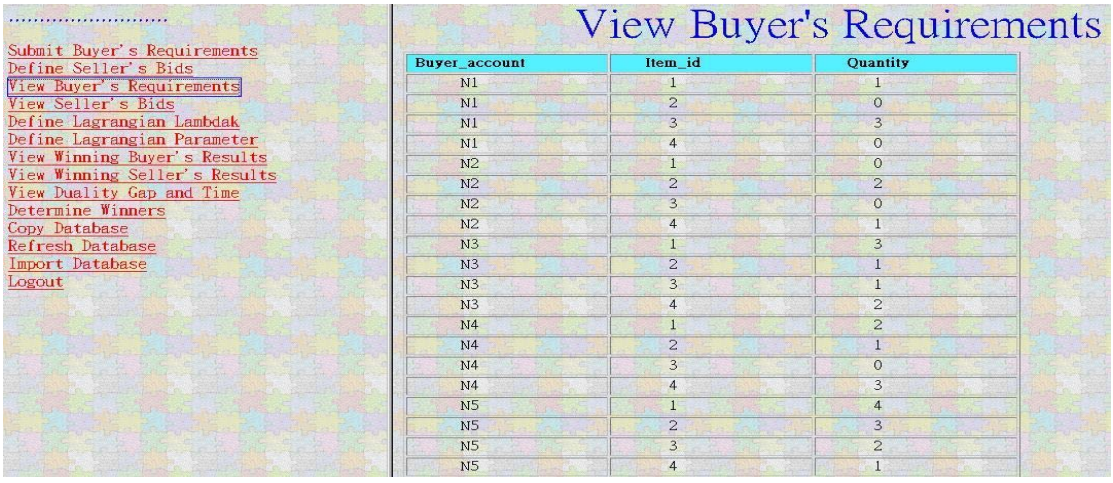


圖 4-1：整體初始網站介面

圖 4-2：整體初始網站登入介面功能



Buyer_account	Item_id	Quantity
N1	1	1
N1	2	0
N1	3	3
N1	4	0
N2	1	0
N2	2	2
N2	3	0
N2	4	1
N3	1	3
N3	2	1
N3	3	1
N3	4	2
N4	1	2
N4	2	1
N4	3	0
N4	4	3
N5	1	4
N5	2	3
N5	3	2
N5	4	1

Buyer_account	Price	Price_will
N1	45	60
N2	50	60
N3	90	60
N4	80	60
N5	100	60

圖 4-3：買家帳戶欄位為 Buyer_account，第一位至第五位買家表示為 N1-N5，Quantity 及 Price 表示為買家*n*的需求商品總數及價格

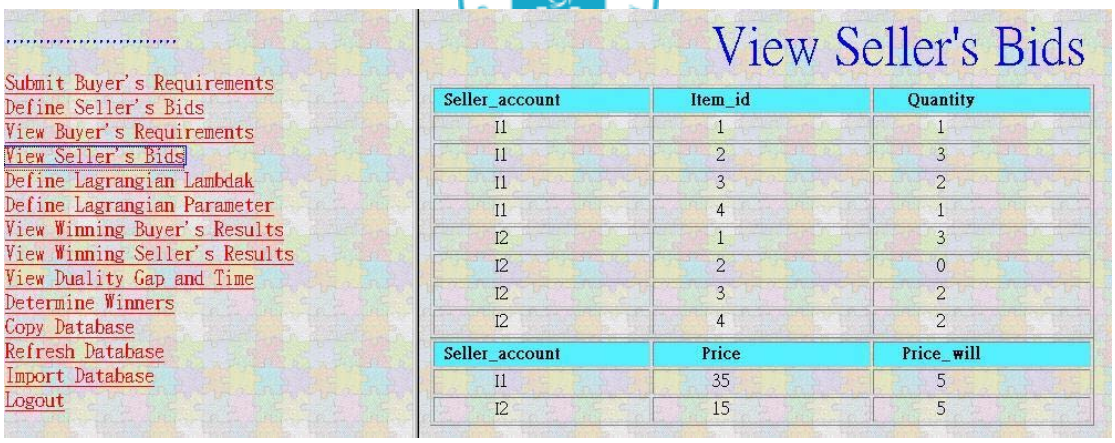


圖 4-4：賣家帳戶欄位為 Seller_account，第一位賣家至第五位賣家表示為 I1-I2，Quantity 及 Price 表示為賣家*i*的出售商品總數及價格

圖 4-3 顯示已經設定好的買家需求商品總數及價格，圖 4-4 顯示已經設定好的賣家所需出售商品總數及價格，接下來設定包括一些關於上述演算法的參數值 Lagragian lambda 值及 Parameter 值，設定好這些指定參數之後，先點選 Determine Winners 連結啟動演算法程式來找出 WDP 的解後，再點選以下圖 4-5 的買家是否得標及圖 4-6 的賣家是否得標的結果。

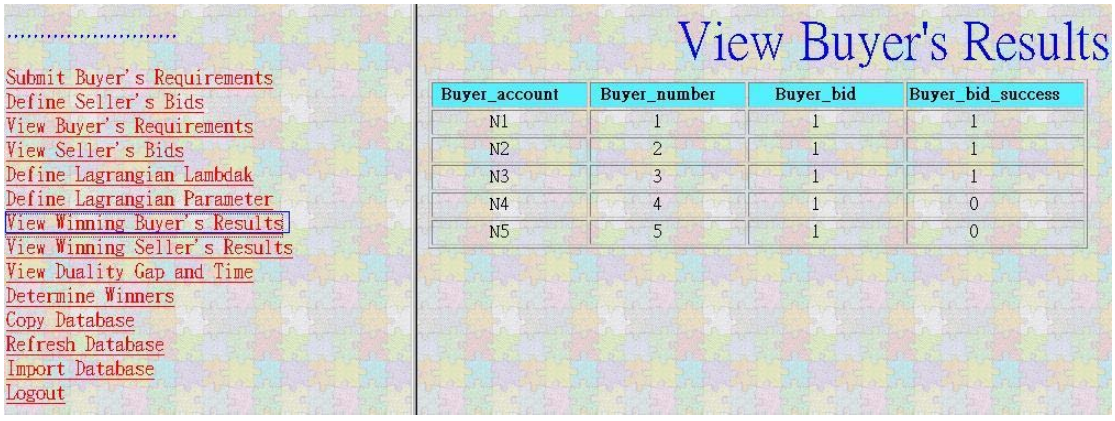


圖 4-5：買家所期望的商品是否有得標或未得標

由圖 4-5 顯示買家所得標的結果為第一位買家至第三位買家(N1-N3)，並可以看出買家(N1-N5)他們所得標的 Buyer_bid_success 為 1 的就表示此筆比

交易成功，若是 Buyer_bid_success 為 0 則此交易失敗。也相對可以看出第四位買家(N4)及第五位買家(N5)他們兩位都未得標。

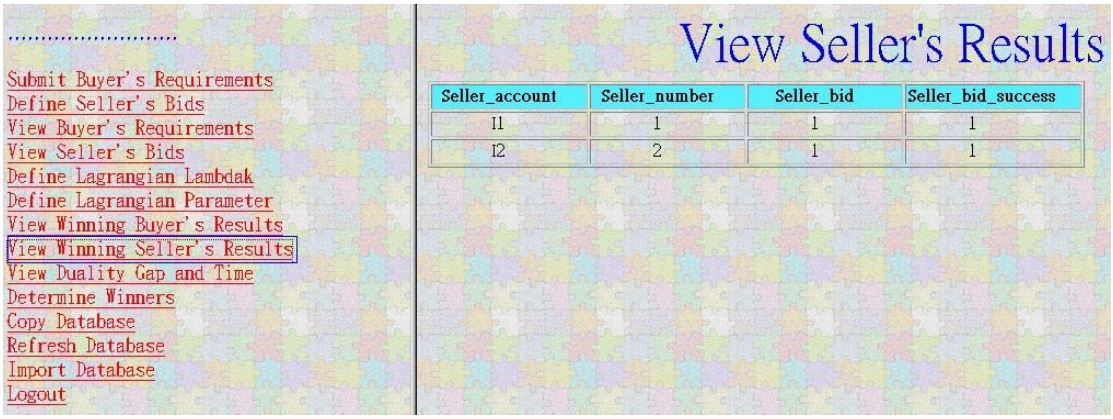


圖 4-6：賣家所出售的商品是否有得標或未得標

由圖 4-5 表示只有第一位至第三位買家(N1-N3)得標，而第四位及第五位買家(N4、N5)都雙雙未得標，很明顯可看出第四位及第五位買家(N4、N5)顯然沒有依照他們所期望的價格得到所期望的商品。從圖 4-6 顯示賣家所得標的結果為第一位賣家及第二位賣家(I1、I2)，並可看出賣家(I1-I2)他們所得標的 Seller_bid_success 為 1 的就表示此筆比交易成功，若是 Seller_bid_success 為 0 則此交易失敗。

由圖 4-5 及圖 4-6 可看出第一位至第三位買家(N1-N3)以及第一、二位賣家(I1、I2)得標，並且兩位賣家(I1、I2)所出售的商品只能滿足三位買家(N1-N3)所期望的商品。由上述圖例表示此雛型系統可以由結合本論文的演算法計算出最匹配的組合。

第五章 實驗結果與分析

本節運用上述雛型系統來進行驗證結果，考慮了兩位賣家及五位買家(I=2，N=5，K=4、I 為賣家，N 為買家，K 為商品)如下：

表 5-1：賣家 I 所提供的商品數量_I2N5K4

Item / Seller	Item 1	Item 2	Item 3	Item 4
Seller 1	1	3	2	1
Seller 2	3	0	2	2
Total	4	3	4	3

根據表 5-1，將有以下參數被設定：

q_{ijk} = 賣家 i 對第 j 次投標所提供商品 k 的量。

$q_{111}=1,\ q_{112}=3,\ q_{113}=2, q_{114}=1$
$q_{211}=3,\ q_{212}=0,\ q_{213}=2, q_{214}=2$

p_{ij} = 賣家 i 於第 j 次投標的價格。

p_{ij} => $p_{11}=35$， $p_{21}=15$

p_{ij}^{WILL} = 賣家 i 接受第 j 次投標商品最小賣出的價格。

$p_{ij}^{WILL}=5$

表 5-2：買家 N 所需求的商品數量_I2N5K4

Item / Buyer	Item 1	Item 2	Item 3	Item 4
Buyer 1	1	0	3	0
Buyer 2	0	2	0	1
Buyer 3	3	1	1	2
Buyer 4	2	1	0	3
Buyer 5	4	3	2	1
Total	10	7	6	7

根據表 5-2，將有以下參數被設定：

d_{nhk} = 買家 n 對第 h 次招標所需求商品 k 的量。

$d_{111}=1,\ \mathrm{d}_{112}=0,\ \mathrm{d}_{113}=3, d_{114}=0$
$d_{211}=0,\ \mathrm{d}_{212}=2,\ \mathrm{d}_{213}=0, d_{214}=1$
$d_{311}=3,\ \mathrm{d}_{312}=1,\ \mathrm{d}_{313}=1, d_{314}=2$
$d_{411}=2,\ \mathrm{d}_{412}=1,\ \mathrm{d}_{413}=0, d_{414}=3$
$d_{511}=4,\ \mathrm{d}_{512}=3,\ \mathrm{d}_{513}=2, d_{514}=1$

p_{nh} = 買家 n 於第 h 次投標的價格

p_{nh} => $p_{11}=45$， $p_{21}=50$ ，
$p_{31}=90$ ， $p_{41}=80$，
$p_{51}=100$

p_{nh}^{WILL} = 買家 n 接受第 h 次招標商品最大買進的價格。

$p_{nh}^{WILL}=60$

初始化 Lagrange multipliers：$\lambda(1) = 3.0, \lambda(2) = 3.0, \lambda(3) = 3.0, \lambda(4) = 3.0$

圖 5-1 為拍賣結果列印：

```
[演算法Strart:]----------------------------------------------------------------------------------
iteration = 1
lambda_k=12.0
賈家得標為：x[1][1]=1 , x[2][1]=0 ,
買家得標為：y[1][1]=1 , y[2][1]=1 , y[3][1]=0 , y[4][1]=0 , y[5][1]=0 ,
L(lambda)=xij((pij-pij_will)-lambda_k*qijk)+ynh((pnh_will-pnh)-lambda_k*dnhk)=9.0+46.0=55.0
g[1]= 0,
g[2]= -1,
g[3]= 1,
g[4]= 0,
sum_gk=2
alpha=2.5 lambda[k]3.0, lambda[k]0.5, lambda[k]5.5, lambda[k]3.0, epsilon=Infinity
----------------------------------------------------------------------------------
[演算法Strart:]----------------------------------------------------------------------------------
iteration = 2
lambda_k=12.0
賈家得標為：x[1][1]=1 , x[2][1]=0 ,
買家得標為：y[1][1]=1 , y[2][1]=1 , y[3][1]=0 , y[4][1]=0 , y[5][1]=0 ,
L(lambda)=xij((pij-pij_will)-lambda_k*qijk)+ynh((pnh_will-pnh)-lambda_k*dnhk)=11.5+48.5=60.0
g[1]= 0,
g[2]= -1,
g[3]= 1,
g[4]= 0,
sum_gk=2
alpha=0.0 lambda[k]3.0, lambda[k]0.5, lambda[k]5.5, lambda[k]3.0, epsilon=0.09090909090909091
----------------------------------------------------------------------------------
[演算法Strart:]----------------------------------------------------------------------------------
iteration = 3
lambda_k=12.0
賈家得標為：x[1][1]=1 , x[2][1]=0 ,
買家得標為：y[1][1]=1 , y[2][1]=1 , y[3][1]=0 , y[4][1]=0 , y[5][1]=0 ,
L(lambda)=xij((pij-pij_will)-lambda_k*qijk)+ynh((pnh_will-pnh)-lambda_k*dnhk)=11.5+48.5=60.0
g[1]= 0,
g[2]= -1,
g[3]= 1,
g[4]= 0,

sum_gk=2
alpha=0.0 lambda[k]3.0, lambda[k]0.5, lambda[k]5.5, lambda[k]3.0, epsilon=0.0

程式結束
當c= 1.0 時, L= 60
顯示得標為：
x*[1][1]
y*[1][1]
y*[2][1]

lambda*[k]={{3.0, 0.5, 5.5, 3.0, }}
L(lambda *)=60.0
----------------------
Time = 0ms(毫秒)
```

圖 5-1：拍賣結果列印_I2N5K4

由圖 5-1 進行推算的拍賣列印結果分析如表 5-3：

表 5-3：拍賣列印的結果分析_I2N5K4

#I	Var	Bid	K1	K2	K3	K4	P
I1	X11	1	1	3	2	1	35
I2	X21	0	3	0	2	2	15
Sum	X		1	3	2	1	

#N	Var	Bid	K1	K2	K3	K4	P
N1	Y11	1	1	0	3	0	40
N2	Y21	1	0	2	0	1	57
N3	Y31	0	3	1	1	2	90
N4	Y41	0	2	1	0	3	80
N5	Y51	0	4	3	2	1	100
Sum	Y		1	2	3	1	

步驟：目前 x11=1，x21=0　　y11=1，y21=1，y31=0，y41=0，y51=0

表 5-3 顯示得標的賣家只有 I1 則買家有 N2 及 N3，計算賣家的總物品量總和之後查詢有沒有大於買家得標的總和，結果發現賣家總和為 1 3 2 1，而買家總和為 1 2 3 1，可看出賣家的物品量缺額為第 K3 項差了 1 項單元物品量要補足買家，則需要進行調解從 I2 下手看是否可以補足與大於其買家的物品量。若是有的則成功得標，I2 設為 1，但又發現說賣家殘餘量不滿足額外的 N4 及 N5 所以使得只有 N1-N3 得標，而沒有其餘的標給 N4、N5，使得雙方都 Match，雙方得標調解成功。

表 5-4：第一次調解_I2N5K4

#I	Var	Bid	K1	K2	K3	K4	P
I1	X11	1	1	3	2	1	35
I2	X21	1	3	0	2	2	15
Sum	X		4	3	4	3	

#N	Var	Bid	K1	K2	K3	K4	P
N1	Y11	1	1	0	3	0	40
N2	Y21	1	0	2	0	1	57
N3	Y31	0	3	1	1	2	90
N4	Y41	0	2	1	0	3	80
N5	Y51	0	4	3	2	1	100
Sum	Y		1	2	3	1	

由表 5-4 進行第一次調解是因為賣家 I1 得標的總和不滿足買家得標的總和量，而發現第三項物品有缺不足買家物品需求量 1 項於 K3 商品，賣家 I1 的 K3 商品得標的總和量只有 2 而買家 N1 及 N2 商品得標總量有 3，經由第一次調解可發現賣家 I1+I2 的商品總和量已經可以滿足買家 I1~I2 商品的需求，則在回頭看看賣家的部份，查看 I2 的殘餘量有沒有滿足買家剩餘的沒得標的殘餘量，而可以發現說賣家目前總量有 4 3 4 3，而買家經過第一次調解的殘餘量則還有 3 1 1 2，從賣家 4 3 4 3 減去 1 2 3 1 還多了 3 1 1 2 的殘餘量，則可以發現說賣家最後的殘餘量有滿足買家 N3 沒得標的殘餘量，則既然賣家的殘餘量有滿足到買家 N3 的殘餘量剛好 Match，但因為賣家沒有多餘的殘餘量可以滿足買家 N4 及 N5 的物品，調解結束。結果如表 5-5：

表 5-5：調解結束_I2N5K4

#I	Var	Bid	K1	K2	K3	K4	P
I1	X11	1	1	3	2	1	35
I2	X21	1	3	0	2	2	15
Sum	X		4	3	4	3	

#N	Var	Bid	K1	K2	K3	K4	P
N1	Y11	1	1	0	3	0	40
N2	Y21	1	0	2	0	1	57
N3	Y31	1	3	1	1	2	90
N4	Y41	0	2	1	0	3	80
N5	Y51	0	4	3	2	1	100
Sum	Y		4	3	4	3	

結果=>　x11=1，x21=1　　y11=1，y21=1，y31=1，y41=0，y51=0

圖 5-2 為 I2N5K4 的程式調解圖：

```
[調解偵測Strart:]------------------------------------------------------------
開始計算賣家得標商品的總和：
sum_x[1]= 1,

sum_x[2]= 3,

sum_x[3]= 2,

sum_x[4]= 1,

開始計算買家得標商品的總和：
sum_y[1]= 1,

sum_y[2]= 2,

sum_y[3]= 3,

sum_y[4]= 1,

查看賣家總和是否有滿足買家總和:
----------------------
sum_xy[1]= 0,
因爲賣家總和有滿足買家總和，所以保留:
sum_xy[2]= 1,
因爲賣家總和有滿足買家總和，所以保留:
sum_xy[3]= -1,
因爲賣家總和並沒有滿足買家總和，賣家目前尚缺以下產品:k[3]= 1, 需要進行調解,
sum_xy[4]= 0,
因爲賣家總和有滿足買家總和，所以保留:
----------------------
因爲已知不滿足的商品，則開始尋找下個賣家來進行補足.
而賣家尚缺以下產品:k[3]= 1,
```

```
[調解開始Strart:]----------------------------------------------
測試Jth是否爲1，若爲1則此投標有效，反則無效
j爲1所以代表有得標，第x[1][1]=1
j爲1所以代表有得標，第x[2][1]=1
測試完Jth之後確認賣家i的k項商品，若爲>0則此投標保留，反則無效
k大於1所以，第x[1][1]*q[1][1][1]=1
k大於1所以，第x[1][1]*q[1][1][2]=3
k大於1所以，第x[1][1]*q[1][1][3]=2
k大於1所以，第x[1][1]*q[1][1][4]=1
k大於1所以，第x[2][1]*q[2][1][1]=3
k大於1所以，第x[2][1]*q[2][1][2]=0
k大於1所以，第x[2][1]*q[2][1][3]=2
k大於1所以，第x[2][1]*q[2][1][4]=2
開始計算賣家得標商品的總和：
sum_x[1]= 4,

sum_x[2]= 3,

sum_x[3]= 4,

sum_x[4]= 3,

開始計算買家得標商品的總和：
sum_y[1]= 1,

sum_y[2]= 2,

sum_y[3]= 3,

sum_y[4]= 1,

查看賣家總和是否有滿足買家總和:
----------------------
sum_xy[1]= 3,
因爲賣家總和有滿足買家總和，所以保留:
sum_xy[2]= 1,
因爲賣家總和有滿足買家總和，所以保留:
sum_xy[3]= 1,
因爲賣家總和有滿足買家總和，所以保留:
sum_xy[4]= 2,
因爲賣家總和有滿足買家總和，所以保留:
程式結束
當c= 1.0 時, L= 60
顯示得標爲：
x*[1][1]
x*[2][1]
y*[1][1]
y*[2][1]

lambda*[k]={(3.0, 0.5, 5.5, 3.0, )}
L(lambda *)=60.0
```

```
check_win(n)_imformation
win[n]=0=>win[3]=>y[3][1]=0
y[3][1]=1
程式結束
當c= 1.0 時, L= 60
顯示得標為：
x*[1][1]
x*[2][1]
y*[1][1]
y*[2][1]
y*[3][1]

lambda*[k]={{3.0, 0.5, 5.5, 3.0, }}
L(lambda *)=60.0
----------------------
開始計算賣家得標商品的總和：
sum_x[1]= 4,

sum_x[2]= 3,

sum_x[3]= 4,

sum_x[4]= 3,

開始計算買家得標商品的總和：
sum_y[1]= 4,

sum_y[2]= 3,

sum_y[3]= 4,

sum_y[4]= 3,

update_differece_x_y(n,h)_imformation
sum_xy[1]= 0,
sum_xy[2]= 0,
sum_xy[3]= 0,
sum_xy[4]= 0,
程式結束
當c= 1.0 時, L= 60
顯示得標為：
x*[1][1]
x*[2][1]
y*[1][1]
y*[2][1]
y*[3][1]

lambda*[k]={{3.0, 0.5, 5.5, 3.0, }}
L(lambda *)=60.0
[調解結束End]------------------------------------------------------------------
程式終止
win_cost_i=(40.0)
win_cost_n=(-5.0)

F(X_bar,Y_bar)=xij((pij-pij_will)+ynh((pnh_will-pnh)=(40.0)+(-5.0)=35.0
L(lambda *)=60.0
Duality GAP=[F(X_bar,Y_bar)-L(lambda *)/F(X_bar,Y_bar)]=0.7142857142857143%
Time = 0ms(毫秒)

Process completed.
```

圖 5-2：調解結果列印_I2N5K4

由原本例子經過調解後的結果列印可以看出，的演算法在 Subgradient 方法收斂後的解為 $x_1^* = 1$(只有第一位賣家得標)及 $y_1^* = 1, y_2^* = 1$(只有第一及第二位買家得標)，其餘賣家及買家都未得標。如果 Subgradient 方法所解出來的就是一個滿足所有限制式的可行解，那就不需要進行 Heuristic 演算法，但是從所得到的解中，可以看出並沒有滿足限制式，g[2]=-1(表示賣家的 k[2]商品多買家的 k[2]商品 1 個單位)，g[3]=1(1 表示賣家的 k[3]商品少買家的 k[3]商品 1 個單位)。所以要執行 Heuristic 演算法來進行調解。然後賣家得到了 $\overline{x_1} = 1, \overline{x_2} = 1$ 及買家得到了 $\overline{y_1} = 1, \overline{y_2} = 1, \overline{y_3} = 1$。從上圖可以看到 Duality gap(對偶間隙)為 0.71%，這個數值是用來判斷所求得的解，是否較接近最佳解。由於結果 x^*, y^* 是個近似解。經由 Duality gap 測試的結果為 0.71% 並於 3%內，則這意味著解決方法產生最佳解。儘管 Duality gap 不為零， 本問題的解 $\overline{X}, \overline{Y}$ 也是最佳的解。 $duality\ gap = \dfrac{\left| F(\overline{X}, \overline{Y}) - L(\lambda^*) \right|}{F(\overline{X}, \overline{Y})}$

where $F(\overline{X}, \overline{Y}) = \sum_{i=1}^{I} \sum_{j=1}^{n_i} \overline{x}_{ij} (p_{ij} - p_{ij}^{WILL}) + \sum_{n=1}^{N} \sum_{h=1}^{n_n} \overline{y}_{nh} (p_{nh}^{WILL} - p_{nh})$

說明了對偶間隙的幾個例子根據問題的大小（I，N，K）。據結果顯示，對偶差距在 0.8%以內。這意味著解決方法產生最佳解，如表 5-6：

表 5-6：Duality Gap

I	N	K	Duality Gap
2	5	4	0.71%
2	5	6	0.89%
3	6	4	0.21%
4	7	5	0.10%
5	8	7	0.09%

在第 5 章目前只列舉一種例子，其餘 4 種例子請參照附錄 1，除了以上五種例子，還進行一些實驗，研究了提出演算法的計算效率。主要是針對以下公式部份：

$$L(\lambda) = Max\left\{\sum_{i=1}^{I}\sum_{j=1}^{n_i} x_{ij}\left[(p_{ij} - p_{ij}^{WILL}) - \sum_{k=1}^{K}\lambda_k q_{ijk}\right] + Max\sum_{n=1}^{N}\sum_{h=1}^{n_n} y_{nh}\left[p_{nh}^{WILL} - p_{nh}) + \sum_{k=1}^{K}\lambda_k d_{nhk}\right]\right\}$$

where $L_I(\lambda) = \max(p_{ij} - p_{ij}^{WILL}) - \sum_{k=1}^{K}\lambda_k q_{ijk}$ & $L_N(\lambda) = \max(p_{nh}^{WILL} - p_{nh}) + \sum_{k=1}^{K}\lambda_k d_{nhk}$

$s.t.\sum_{j=1}^{n_i} x_{ij} \le 1 \quad \forall i = 1,2,3,...,I \;\&\; \sum_{h=1}^{n_n} y_{nh} \le 1 \quad \forall n = 1,2,3,...,N,\ x_{ij} \;\&\; y_{nh} \in \{0,1\}$

由下圖 5-3～圖 5-5 的結果證明一個事實，既計算 $L(\lambda)$ 的 CPU 時間會隨著所設定的 I、N、K(賣家、買家、商品)數量的增加而成一個比例表，下圖 5-3～圖 5-5 的 CPU 時間皆以 ms(毫秒)為主。

圖 5-3 顯示出當將參數 N(買家)及 K(商品)固定，改變 I(賣家)參數時，幾個問題的 CPU 時間。這裡參數 N=15、K=5，I 由 1 到 9。圖相關資料內容，請參照附錄 2。

	I=1	I=3	I=5	I=7	I=9
I的變化影響時間的變化	468	563	656	766	875

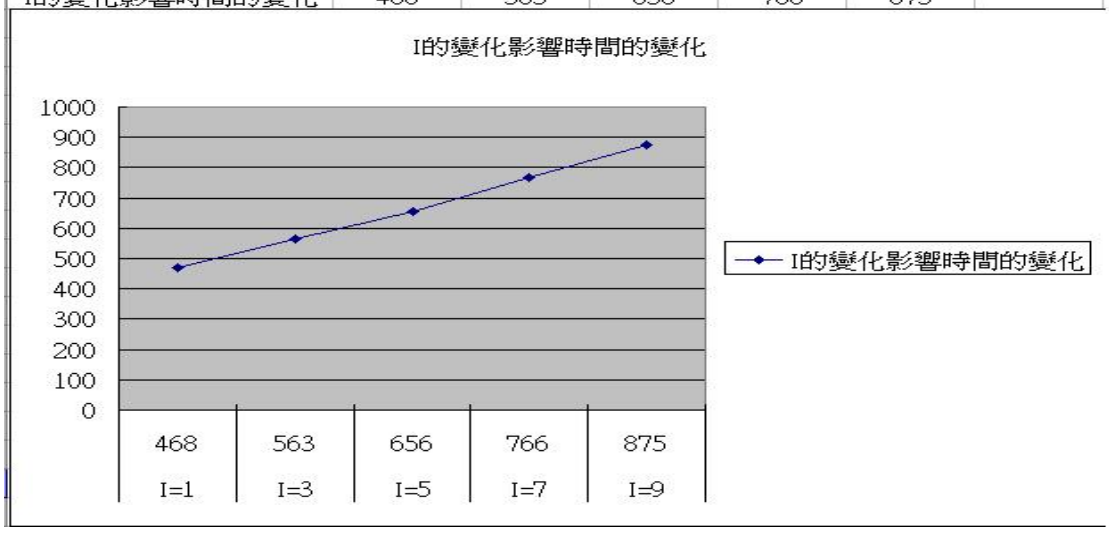


圖 5-3：重複改變 I 參數的 CPU 時間

圖 5-4 顯示出當將參數 I(賣家)及 K(商品)固定，改變 N(買家)參數時，幾個問題的 CPU 時間。這裡參數 I=10、K=5，I 由 2 到 10。圖相關資料內容，請參照附錄 2。

	N=2	N=4	N=6	N=8	N=10
N的變化影響時間的變化	312	375	453	578	657

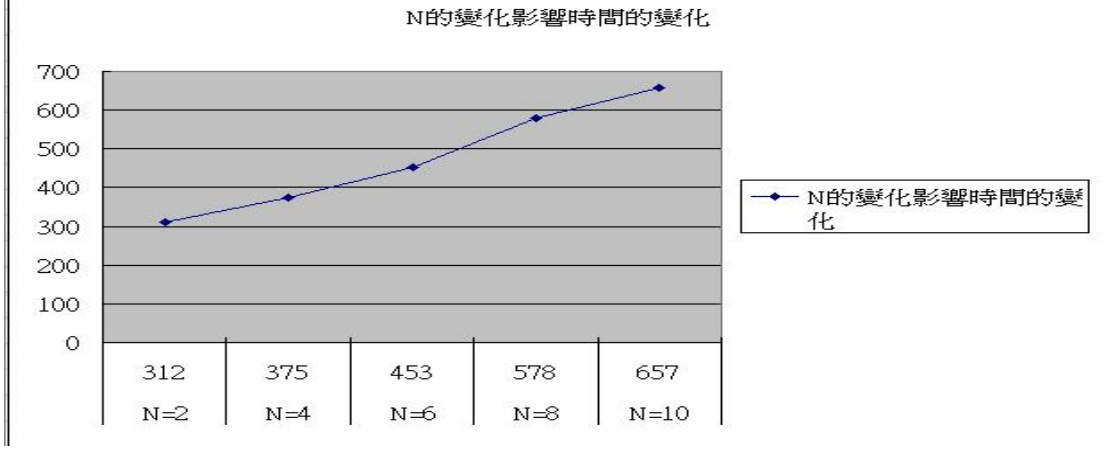


圖 5-4：重複改變 N 參數的 CPU 時間

圖 5-5 顯示出當將參數 I(賣家)及 N(買家)固定，改變 K(商品)參數時，幾個問題的 CPU 時間。這裡參數 I=2、N=4，K 由 5 到 25。圖相關資料內

容，請參照附錄 2。

	K=5	K=10	K=15	K=20	K=25
K的變化影響時間的變化	16	281	531	782	813

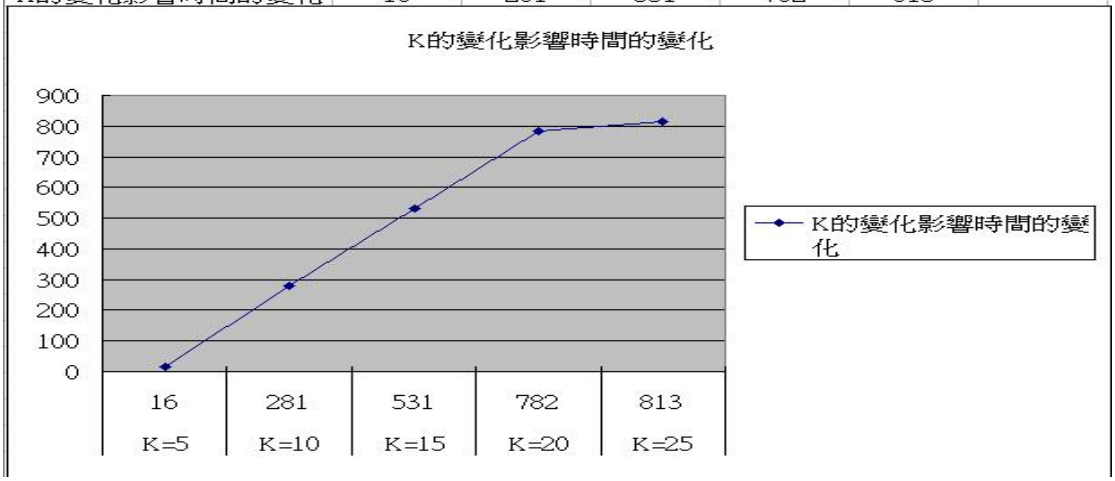


圖 5-5：重複改變 K 參數的 CPU 時間

第六章 結論與未來展望

本論文所探討的問題是在具有多個買家和多個賣家下的組合式雙向拍賣。組合式雙向拍賣讓參與者可以在單一次的拍賣活動中同時對多樣不同的商品進行投標，依照自己的喜愛、偏好，並且針對不同的商品組合去競標以及給予這些物品組合的一定金額。改善了傳統只能針對單一的商品進行競標的效率。主要的結果包括：(1)建立組合式雙向拍賣中的數學化模式，(2)利用 Lagrangian Relaxation 的方法論來解決最佳化的問題，(3)提出解決 WDP 的演算法，(4)根據結果進行效能分析。由運用 Lagrangian Relaxation 技術，讓原本的問題可以分解成若干個投標者的子問題，使得可以更有效的解決。分析和計算的結果顯示，的演算法可以在可接受的 CPU 時間內產生最佳解，也建置一個雛型系統來驗證所提出方法的運算效能。

參考文獻

[1] Alessandro Avenali, Anna Bassanini, “Simulating combinatorial auctions with dominance requirement and loll bids through automated agents,” *Decision Support Systems*, Vol 43, Issue 1, February 2007, pp. 211-228.

[2] A. Andersson, M. Tenhunen, and F. Ygge. “Integer programming for combinatorial auction winner determination,” *Proceedings of the Seventeenth National Conference on Artificial Intelligence*, pp. 39–46, 2000.

[3] B.T. Polyak, “Minimization of unsmooth functionals,” *USSR Computational Mathematics and Mathematical Physics*, Vol 9 Issue 3 (1969), pp. 14–29.

[4] C. G. Caplice, “An Optimization Based Bidding Process: A New Framework for Shipper-Carrier Relationships,” *Thesis in the field of Transportation and Logistics Systems submitted to the Department of Civil and Environmental Engineering, School of Engineering, MIT*, June 1996.

[5] D. P. Bertsekas., “Nonlinear Programming,” *Athena Scientific, second edition,* 1999.

[6] Fu-Shiung Hsieh, Cheng-Chung Hua , “Combinatorial Reverse Auctions with Multiple Buyers,” *Master thesis, Department and Graduate Institute of Information and Communication Engineering*, 2009.

[7] Fu-Shiung Hsieh, Shih-Min Tsai , “A Lagrangian Relaxation Approach For Combinatorial Reverse Auction,” *Master thesis, Department and Graduate Institute of Information and Communication Engineering*, 2009.

[8] Fu-Shiung Hsieh, Shih-Min Tsai , “An Efficient Algorithm for Combinatorial Reverse Auction,” *Master thesis, Department and Graduate Institute of Information and Communication Engineering*, 2008.

[9] Holger H. Hoos and Craig Boutilier. “Solving combinatorial auctions using stochastic local search,” *Proceedings of the Seventeenth National Conference on Artificial Intelligence*, pp. 22–29, 2000.

[10] Jaime Catalan, Rafael Epstein, Mario Guajardo, Daniel Yung, Cristian Martlnez, “*Solving multiple scenarios in a combinatorial auction,*” *Computers & Operations Research*, Vol 36, Issue 10, October 2009, pp. 2752-2758.

[11] Jawad Abrache, Benolt Bourbeau, Teodor Gabriel Crainic, Michel Gendreau, “A new bidding framework for combinatorial e-auctions,” *Computers & Operations Research*,Vol 31, Issue 8, July 2004, pp. 1177-1203.

[12] Jin Ho Choi, Hyunchul Ahn, Ingoo Han. “Utility-based double auction mechanism using genetic algorithms,” *Expert Systems with Applications*,

Vol 34 Issue 1 (2008), pp. 150–158.
[13] Joni L. Jones, Gary J. Koehle, “Combinatorial auctions using rule-based bids,” *Decision Support Systems*, Vol 34, Issue 1, December 2002, pp. 59-74.
[14] Lagrangian (http://en.wikipedia.org/wiki/Lagrangian_relaxation).
[15] Murat Koksalan, Riikka-Leena Leskela, Hannele Wallenius, Jyrki Wallenius, “Improving efficiency in multiple-unit combinatorial auctions: Bundling bids from multiple bidders,” *Decision Support Systems*, Vol 48, Issue 1, December 2009, pp. 103-111.
[16] M. AkgÄul. “Topics in Relaxation and Ellipsoidal Methods,” *Research Notes in Mathematics. Pitman*, Vol 97, 1984.
[17] M. Xia, G.J. Koehler, A.B. Whinston, “Pricing combinatorial auctions,” *European Journal of Operational Research*, Vol 154 (2004), pp. 251–270.
[18] N. Z. Shor. “Minimization Methods for Non-differentiable Functions,” *Springer Series in Computational Mathematics , Springer*, 1985.
[19] P. J. Brewer, “Decentralized Computation Procurement and Computational Robustness in a Smart Market,” *Economic Theory*, 13, pp. 41-92, 1999.
[20] Riikka-Leena Leskela, Jeffrey Teich, Hannele Wallenius, Jyrki Wallenius, “ Decision support for multi-unit combinatorial bundle auctions,” *Decision Support Systems*, Vol 43, Issue 2, March 2007, pp. 420-434.
[21] R.R. Vemuganti, “Applications of set covering, set packing and set partitioning models: a survey,” *D.-Z. Du, Editor, Handbook of Combinatorial Optimization, Kluwer Academic Publishers, Netherlands,*Vol. 1, pp. 573–746, 1998.
[22] Subgradient (http://en.wikipedia.org/wiki/Subgradient_method).
[23] Tuomas Sandholm. “Algorithm for optimal winner determination in combinatorial auctions,” *Artificial Intelligence*, Vol 135, Issues(1-2), , February, pp. 1–54, 2002.
[24] Tuomas Sandholm, Subhash Suri, Andrew Gilpin, and David Levine. “CABOB: A fast optimal algorithm for combinatorial auctions,” *IJCAI*, pp. 1102–1108, 2001.
[25] Valtteri Ervasti, Riikka-Leena Leskela, “Allocative efficiency in simulated multiple-unit combinatorial auctions with quantity support,” *European Journal of Operational Research*, Vol 203, Issue 1, 16 May 2010, pp. 251-260.
[26] Y. Fujishima, K. Leyton-Brown, and Y. Shoham. “Taming the computational complexity of combinatorial auctions: Optimal and approximate approaches,” *Sixteenth International Joint Conference on Artificial Intelligence*, pp. 548–553, 1999.

附錄A

依據第 5 章的表 5-6 中還有其餘四個例子，將會在下面詳細說明，因第 5 章有提出例子 I2N5K4，所以這邊只列出其餘 4 個例子，以例子 I2N5K6 考慮了兩位賣家及五位買家，不過商品提昇至 6 項(I=2，N=5，K=6)，根據例子 I2N5K6 說明，將有以下參數(I 為賣家，N 為買家，K 為商品)：

表 A-1：賣家 I 所提供的商品數量_I2N5K6

Item / Seller	Item 1	Item 2	Item 3	Item 4	Item 5	Item 6
Seller 1	1	1	1	2	0	2
Seller 2	0	2	3	0	1	1
Total	1	3	4	2	1	3

根據表 A-1，將有以下參數被設定：

q_{ijk} = 賣家 i 對第 j 次投標所提供商品 k 的量。

$$q_{111}=1,\ q_{112}=1,\ q_{113}=1, q_{114}=2, q_{115}=0, q_{116}=2$$
$$q_{211}=0,\ q_{212}=2,\ q_{213}=3, q_{214}=0, q_{215}=1, q_{216}=1$$

p_{ij} = 賣家 i 於第 j 次投標的價格。

$$p_{ij} \Rightarrow p_{11}=42,\ p_{21}=25$$

賣家自動接受第一次投標商品最小可接受賣出的價格

$$p_{ij}^{WILL}=15$$

表 A-2：買家 N 所需求的商品數量_I2N5K6

Item / Buyer	Item 1	Item 2	Item 3	Item 4	Item 5	Item 6
Buyer 1	1	0	0	1	0	2
Buyer 2	0	0	3	1	0	0
Buyer 3	0	3	1	0	0	1
Buyer 4	2	1	0	3	2	0
Buyer 5	3	2	1	0	2	3
Total	7	6	5	5	4	6

根據表 A-2，將有以下參數被設定：

d_{nhk} = 買家 n 對第 h 次招標所需求商品 k 的量。

$$d_{111}=1,\ d_{112}=0,\ d_{113}=0, d_{114}=1, d_{115}=0, d_{116}=2$$
$$d_{211}=0,\ d_{212}=0,\ d_{213}=3, d_{214}=1, d_{215}=0, d_{216}=0$$
$$d_{311}=0,\ d_{312}=3,\ d_{313}=1, d_{314}=0, d_{315}=0, d_{316}=1$$
$$d_{411}=2,\ d_{412}=1,\ d_{413}=0, d_{414}=3, d_{415}=2, d_{416}=0$$
$$d_{511}=3,\ d_{512}=2,\ d_{513}=1, d_{514}=0, d_{515}=2, d_{516}=3$$

p_{nh} = 買家 n 於第 h 次投標的價格

$$p_{nh} => p_{11}=50,\ p_{21}=40\ ,\ p_{31}=90,\ p_{41}=80,\ p_{51}=100$$

p_{nh}^{WILL} = 買家 n 接受第 h 次招標商品最大買進的價格。

$$p_{nh}^{WILL}=60$$

初始化 Lagrange multipliers： $\lambda(k)=2.0, \overline{L}=70$

圖 A-1 為拍賣結果列印：

```
[演算法Strart:]----------------------------------------------------------------------------------------
iteration = 1
lambda_k=12.0
買家得標為：x[1][1]=1 , x[2][1]=0 ,
買家得標為：y[1][1]=1 , y[2][1]=1 , y[3][1]=0 , y[4][1]=0 , y[5][1]=0 ,
L(lambda)=xij((pij-pij_will)-lambda_k*qijk)+ynh((pnh_will-pnh)-lambda_k*dnhk)=13.0+46.0=59.0
g[1]= 0,
g[2]= -1,
g[3]= 2,
g[4]= 0,
g[5]= 0,
g[6]= 0,
sum_gk=5
alpha=2.2 lambda[k]2.0, lambda[k]0.0, lambda[k]6.4, lambda[k]2.0, lambda[k]2.0, lambda[k]2.0, epsilon=Infinity
----------------------------------------------------------------------------------------
[演算法Strart:]----------------------------------------------------------------------------------------
iteration = 2
lambda_k=14.4
買家得標為：x[1][1]=1 , x[2][1]=0 ,
買家得標為：y[1][1]=1 , y[2][1]=1 , y[3][1]=0 , y[4][1]=0 , y[5][1]=0 ,
L(lambda)=xij((pij-pij_will)-lambda_k*qijk)+ynh((pnh_will-pnh)-lambda_k*dnhk)=10.6+59.2=69.8
g[1]= 0,
g[2]= -1,
g[3]= 2,
g[4]= 0,
g[5]= 0,
g[6]= 0,
sum_gk=5
alpha=0.04 lambda[k]2.0, lambda[k]0.0, lambda[k]6.48, lambda[k]2.0, lambda[k]2.0, lambda[k]2.0, epsilon=0.18305084745762706
----------------------------------------------------------------------------------------
[演算法Strart:]----------------------------------------------------------------------------------------
iteration = 3
lambda_k=14.48
買家得標為：x[1][1]=1 , x[2][1]=0 ,
買家得標為：y[1][1]=1 , y[2][1]=1 , y[3][1]=0 , y[4][1]=0 , y[5][1]=0 ,
L(lambda)=xij((pij-pij_will)-lambda_k*qijk)+ynh((pnh_will-pnh)-lambda_k*dnhk)=10.52+59.44=69.96
g[1]= 0,
g[2]= -1,
g[3]= 2,
g[4]= 0,
g[5]= 0,
g[6]= 0,
sum_gk=5
alpha=0.01 lambda[k]2.0, lambda[k]0.0, lambda[k]6.5, lambda[k]2.0, lambda[k]2.0, lambda[k]2.0, epsilon=0.0022922636103151375
----------------------------------------------------------------------------------------
```

```
程式結束
當c= 1.0 時, L= 70
顯示得標為：
x*[1][1]
y*[1][1]
y*[2][1]

lambda*[k]={{2.0, 0.0, 6.5, 2.0, 2.0, 2.0, }}
L(lambda *)=69.96
------------------------
Time = 16ms(毫秒)
```

圖 A-1：拍賣結果列印_I2N5K6

由圖 A-1 進行推算的拍賣列印結果分析如表 A-3：

表 A-3：拍賣列印的結果分析_I2N5K4

#I	Var	Bid	K1	K2	K3	K4	K5	K6	P
I1	X11	1	1	1	1	2	0	2	42
I2	X21	0	0	2	3	0	1	1	25
Sum	X		1	1	1	2	0	2	

#N	#Var	Bid	K1	K2	K3	K4	K5	K6	P
N1	Y11	1	1	0	0	1	0	2	50
N2	Y21	1	0	0	3	1	0	0	40
N3	Y31	0	0	3	1	0	0	1	90
N4	Y41	0	2	1	0	3	2	0	80
N5	Y51	0	3	2	1	0	2	3	100
Sum	Y		1	0	3	2	0	2	

結果：目前 x11=1，x21=0　　y11=1，y21=1，y31=0，y41=0，y51=0

其得標的賣家只有 I1 則買家有 N2 及 N3，計算賣家的總物品量總和之後查詢有沒有大於買家得標的總和，結果發現賣家總和為 1 1 1 2 0 2，則買家總和為 1 0 3 2 0 2，則可以看出賣家的物品量缺額為第 K3 項差了 2 項單元物品量要補足買家，則需要進行調解從 I2 下手看是否可以補足與大於其買家的物品量。若是有的則成功得標，I2 設為 1，但又發現說賣家殘餘量不滿足額外的 N4 及 N5 所以使得只有 N1-N3 得標，而沒有其餘的標給 N4、N5，使得雙方都 Match，雙方得標調解成功。

表 A-4：第一次調解_I2N5K4

#I	Var	Bid	K1	K2	K3	K4	K5	K6	P
I1	X11	1	1	1	1	2	0	2	42
I2	X21	1	0	2	3	0	1	1	25
Sum	X		1	3	4	2	1	3	

#N	#Var	Bid	K1	K2	K3	K4	K5	K6	P
N1	Y11	1	1	0	0	1	0	2	50
N2	Y21	1	0	0	3	1	0	0	40
N3	Y31	0	0	3	1	0	0	1	90
N4	Y41	0	2	1	0	3	2	0	80
N5	Y51	0	3	2	1	0	2	3	100
Sum	Y		1	0	3	2	0	2	

由表 A-4 進行第一次調解是因為賣家 I1 得標的總和不滿足買家得標的總和量，而發現第三項物品有缺不足買家物品需求量 2 項於 K3 商品，賣家 I1 的 K3 商品得標的總和量只有 1 而買家 N1 及 N2 商品得標總量有 3，經由第一次調解可發現賣家 I1+I2 的商品總和量已經可以滿足買家 I1~I2 商品的需求，則在回頭看看賣家的部份，查看 I2 的殘餘量有沒有滿足買家剩餘的沒得標的殘餘量，而可以發現說賣家目前總量有 1 3 4 2 1 3，而買家經過第一次調解的殘餘量則還有 0 3 1 0 0 1，從賣家 1 3 4 2 1 3 減去 1 0 3 2 0 2 還多了 0 3 1 0 1 1 的殘餘量，則可以發現說賣家最後的殘餘量有滿足買家 N3 沒得標的殘餘量並且還多了賣家第 K5 項商品 1 個單位量，但因為賣家沒有多餘的殘餘量可以滿足買家 N4 及 N5 的物品，調解結束。則結果如表 A-5：

表 A-5：調解結束_I2N5K6

#I	Var	Bid	K1	K2	K3	K4	K5	K6	P
I1	X11	1	1	1	1	2	0	2	42
I2	X21	1	0	2	3	0	1	1	25
Sum	X		1	3	4	2	1	3	

#N	#Var	Bid	K1	K2	K3	K4	K5	K6	P
N1	Y11	1	1	0	0	1	0	2	50
N2	Y21	1	0	0	3	1	0	0	40
N3	Y31	1	0	3	1	0	0	1	90
N4	Y41	0	2	1	0	3	2	0	80
N5	Y51	0	3	2	1	0	2	3	100
Sum	Y		1	3	4	2	0	3	

結果=>　x11=1，x21=1　y11=1，y21=1，y31=1，y41=0，y51=0

圖 A-2 為 I2N5K6 的程式調解圖：

```
[調解偵測Strart:]-----------
開始計算賣家得標商品的總和：
sum_x[1]= 1,

sum_x[2]= 1,

sum_x[3]= 1,

sum_x[4]= 2,

sum_x[5]= 0,

sum_x[6]= 2,

開始計算買家得標商品的總和：
sum_y[1]= 1,

sum_y[2]= 0,

sum_y[3]= 3,

sum_y[4]= 2,

sum_y[5]= 0,

sum_y[6]= 2,
```

查看賣家總和是否有滿足買家總和:
---
sum_xy[1]= 0,
因爲賣家總和有滿足買家總和，所以保留:
sum_xy[2]= 1,
因爲賣家總和有滿足買家總和，所以保留:
sum_xy[3]= -2,
因爲賣家總和並沒有滿足買家總和，賣家目前尚缺以下產品:k[3]= 2, 需要進行調解,
sum_xy[4]= 0,
因爲賣家總和有滿足買家總和，所以保留:
sum_xy[5]= 0,
因爲賣家總和有滿足買家總和，所以保留:
sum_xy[6]= 0,
因爲賣家總和有滿足買家總和，所以保留:

因爲已知不滿足的商品，則開始尋找下個賣家來進行補足.
而賣家尚缺以下產品:k[3]= 2,

[調解開始Strart:]----------
測試Jth是否爲1，若爲1則此投標有效，反則無效
j爲1所以代表有得標，第x[1][1]=1
j爲1所以代表有得標，第x[2][1]=1
測試完Jth之後確認賣家i的k項商品，若爲>0則此投標保留，反則無效
k大於1所以，第x[1][1]*q[1][1][1]=1
k大於1所以，第x[1][1]*q[1][1][2]=1
k大於1所以，第x[1][1]*q[1][1][3]=1
k大於1所以，第x[1][1]*q[1][1][4]=2
k大於1所以，第x[1][1]*q[1][1][5]=0
k大於1所以，第x[1][1]*q[1][1][6]=2
k大於1所以，第x[2][1]*q[2][1][1]=0
k大於1所以，第x[2][1]*q[2][1][2]=2
k大於1所以，第x[2][1]*q[2][1][3]=3
k大於1所以，第x[2][1]*q[2][1][4]=0
k大於1所以，第x[2][1]*q[2][1][5]=1
k大於1所以，第x[2][1]*q[2][1][6]=1
開始計算賣家得標商品的總和：
sum_x[1]= 1,

sum_x[2]= 3,

sum_x[3]= 4,

sum_x[4]= 2,

sum_x[5]= 1,

sum_x[6]= 3,

開始計算買家得標商品的總和：
sum_y[1]= 1,

sum_y[2]= 0,

sum_y[3]= 3,

sum_y[4]= 2,

sum_y[5]= 0,

sum_y[6]= 2,

```
查看賈家總和是否有滿足買家總和:
----------------------
sum_xy[1]= 0,
因爲賈家總和有滿足買家總和，所以保留:
sum_xy[2]= 3,
因爲賈家總和有滿足買家總和，所以保留:
sum_xy[3]= 1,
因爲賈家總和有滿足買家總和，所以保留:
sum_xy[4]= 0,
因爲賈家總和有滿足買家總和，所以保留:
sum_xy[5]= 1,
因爲賈家總和有滿足買家總和，所以保留:
sum_xy[6]= 1,
因爲賈家總和有滿足買家總和，所以保留:
程式結束
當c= 1.0 時, L= 70
顯示得標爲：
x*[1][1]
x*[2][1]
y*[1][1]
y*[2][1]

lambda*[k]={{2.0, 0.0, 6.5, 2.0, 2.0, 2.0, }}
L(lambda *)=69.96
----------------------
check_win(n)_imformation
win[n]=0=>win[3]=>y[3][1]=0
y[3][1]=1
程式結束
當c= 1.0 時, L= 70
顯示得標爲：
x*[1][1]
x*[2][1]
y*[1][1]
y*[2][1]
y*[3][1]

lambda*[k]={{2.0, 0.0, 6.5, 2.0, 2.0, 2.0, }}
L(lambda *)=69.96
----------------------
```

```
開始計算賣家得標商品的總和：
sum_x[1]= 1,

sum_x[2]= 3,

sum_x[3]= 4,

sum_x[4]= 2,

sum_x[5]= 1,

sum_x[6]= 3,

開始計算買家得標商品的總和：
sum_y[1]= 1,

sum_y[2]= 3,

sum_y[3]= 4,

sum_y[4]= 2,

sum_y[5]= 0,

sum_y[6]= 3,

update_differece_x_y(n,h)_imformation
sum_xy[1]= 0,
sum_xy[2]= 0,
sum_xy[3]= 0,
sum_xy[4]= 0,
sum_xy[5]= 1,
sum_xy[6]= 0,
程式結束
當c= 1.0 時, L= 70
顯示得標為：
x*[1][1]
x*[2][1]
y*[1][1]
y*[2][1]
y*[3][1]

lambda*[k]={{2.0, 0.0, 6.5, 2.0, 2.0, 2.0, }}
L(lambda *)=69.96

[調解結束End]----------------------------------------------------------------
程式終止
win_cost_i=(37.0)
win_cost_n=(0.0)

F(X_bar,Y_bar)=xij((pij-pij_will)+ynh((pnh_will-pnh)=(37.0)+(0.0)=37.0
L(lambda *)=69.96
Duality GAP=[F(X_bar,Y_bar)-L(lambda *)/F(X_bar,Y_bar)]=0.8908108108108106%
Time = 16ms(毫秒)

Process completed.
```

圖 A-2：調解結果列印_I2N5K6

例子 I3N6K4 考慮了三位賣家及六位買家，不過商品降為 4 項(I=3，N=6，K=4)，根據例子 I3N6K4 說明，將有以下參數(I 為賣家，N 為買家，K 為商品)：

表 A-6：賣家 I 所提供的商品數量_ I3N6K4

Item / Seller	Item 1	Item 2	Item 3	Item 4
Seller 1	1	2	3	2
Seller 2	2	1	0	3
Seller 3	2	3	1	1
Total	5	6	4	6

根據表 A-6，將有以下參數被設定：

q_{ijk} = 賣家 i 對第 j 次投標所提供商品 k 的量。

$$q_{111}=1,\ q_{112}=2,\ q_{113}=3, q_{114}=2$$
$$q_{211}=2,\ q_{212}=1,\ q_{213}=0, q_{214}=3$$
$$q_{311}=2,\ q_{312}=3,\ q_{313}=1, q_{314}=1$$

p_{ij} = 賣家 i 於第 j 次投標的價格。

p_{ij} => $p_{11}=47$， $p_{21}=20$， $p_{31}=20$

p_{ij}^{WILL} = 賣家 i 接受第 j 次投標商品最小賣出的價格。

$$p_{ij}^{WILL}=5$$

表 A-7：買家 N 所需求的商品數量_ I3N6K4

Item / Buyer	Item 1	Item 2	Item 3	Item 4
Buyer 1	1	0	3	0
Buyer 2	0	2	0	2
Buyer 3	2	0	1	0
Buyer 4	1	3	0	3
Buyer 5	2	1	0	3
Buyer 6	3	2	1	0
Total	9	8	5	8

根據表 A-7，將有以下參數被設定：

d_{nhk} = 買家 n 對第 h 次招標所需求商品 k 的量。

$$d_{111}=1,\ \mathrm{d}_{112}=0,\ \mathrm{d}_{113}=1, d_{114}=2$$
$$d_{211}=0,\ \mathrm{d}_{212}=2,\ \mathrm{d}_{213}=2, d_{214}=0$$
$$d_{311}=2,\ \mathrm{d}_{312}=0,\ \mathrm{d}_{313}=1, d_{314}=0$$
$$d_{411}=0,\ \mathrm{d}_{412}=3,\ \mathrm{d}_{413}=0, d_{414}=1$$
$$d_{511}=2,\ \mathrm{d}_{512}=1,\ \mathrm{d}_{513}=0, d_{514}=3$$
$$d_{611}=3\ \mathrm{d}_{612}=2,\ \mathrm{d}_{613}=1, d_{614}=0$$

p_{nh} = 買家 n 於第 h 次投標的價格

$p_{nh} \Rightarrow p_{11}=40$， $p_{21}=38$ ， $p_{31}=32$， $p_{41}=80$， $p_{51}=90$， $p_{61}=100$

p_{nh}^{WILL} = 買家 n 接受第 h 次招標商品最大買進的價格。

$$p_{nh}^{WILL}=50$$

初始化 Lagrange multipliers： $\lambda(k)=3.0, \overline{L}=100$

圖 A-3 為拍賣結果列印：

```
[演算法Strart:]----------------------------------------------------------------------------------
iteration = 1
lambda_k=12.0
賈家得標為：x[1][1]=1 , x[2][1]=0 , x[3][1]=0 ,
買家得標為：y[1][1]=1 , y[2][1]=1 , y[3][1]=1 , y[4][1]=0 , y[5][1]=0 , y[6][1]=0 ,
L(lambda)=xij((pij-pij_will)-lambda_k*qijk)+ynh((pnh_will-pnh)-lambda_k*dnhk)=18.0+73.0=91.0
g[1]= 2,
g[2]= 0,
g[3]= 1,
g[4]= 0,
sum_gk=5
alpha=1.8 lambda[k]6.6, lambda[k]3.0, lambda[k]4.8, lambda[k]3.0, epsilon=Infinity
----------------------------------------------------------------------------------
[演算法Strart:]----------------------------------------------------------------------------------
iteration = 2
lambda_k=17.4
賈家得標為：x[1][1]=1 , x[2][1]=0 , x[3][1]=0 ,
買家得標為：y[1][1]=1 , y[2][1]=1 , y[3][1]=1 , y[4][1]=0 , y[5][1]=0 , y[6][1]=0 ,
L(lambda)=xij((pij-pij_will)-lambda_k*qijk)+ynh((pnh_will-pnh)-lambda_k*dnhk)=9.0+91.0=100.0
g[1]= 2,
g[2]= 0,
g[3]= 1,
g[4]= 0,
sum_gk=5
alpha=0.0 lambda[k]6.6, lambda[k]3.0, lambda[k]4.8, lambda[k]3.0, epsilon=0.0989010989010989
----------------------------------------------------------------------------------
[演算法Strart:]----------------------------------------------------------------------------------
iteration = 3
lambda_k=17.4
賈家得標為：x[1][1]=1 , x[2][1]=0 , x[3][1]=0 ,
買家得標為：y[1][1]=1 , y[2][1]=1 , y[3][1]=1 , y[4][1]=0 , y[5][1]=0 , y[6][1]=0 ,
L(lambda)=xij((pij-pij_will)-lambda_k*qijk)+ynh((pnh_will-pnh)-lambda_k*dnhk)=9.0+91.0=100.0
g[1]= 2,
g[2]= 0,
g[3]= 1,
g[4]= 0,
sum_gk=5
alpha=0.0 lambda[k]6.6, lambda[k]3.0, lambda[k]4.8, lambda[k]3.0, epsilon=0.0
----------------------------------------------------------------------------------

程式結束
當c= 1.0 時, L= 100
顯示得標為：
x*[1][1]
y*[1][1]
y*[2][1]
y*[3][1]

lambda*[k]={{6.6, 3.0, 4.8, 3.0, }}
L(lambda *)=100.0
----------------------
Time = 0ms(毫秒)
```

圖 A-3：拍賣結果列印_ I3N6K4

由圖 A-3 進行推算的拍賣列印結果分析如表 A-8：

表 A-8：拍賣列印的結果分析_ I3N6K4

#I	Var	Bid	K1	K2	K3	K4	P
I1	X11	1	1	2	3	2	47
I2	X21	0	2	1	0	3	20
I3	X31	0	2	3	1	1	20
Sum	X		1	2	3	2	

#N	#Var	Bid	K1	K2	K3	K4	P
N1	Y11	1	1	0	3	0	40
N2	Y21	1	0	2	0	2	38
N3	Y31	1	2	0	1	0	32
N4	Y41	0	1	3	0	3	80
N5	Y51	0	2	1	0	3	90
N6	Y61	0	3	2	1	0	100
Sum	Y		3	2	4	2	

結果：目前 x11=1，x21=0，x31=0Y11=1，y21=1，y31=1，y41=0，y51=0，y61=0

其得標的賣家只有 I1 則買家有 N1、N2 及 N3，計算賣家的總物品量總和之後查詢有沒有大於買家得標的總和，結果發現賣家總和為 1 2 3 2，則買家總和為 3 2 4 2，則可以看出賣家的物品量缺額為第 K1 項差了 2 項單元物品量及第 K3 項差了 1 項單元物品量要補足買家，則需要進行調解而因為要先補足 K1 的物品量則往下查詢 I2 及 I3 的 K1 條件發覺說兩項都是有辦法補足買家 K1 不足的物品量，先按照順序先從 I2 下手看是否可以補足與大於其買家的物品量。若是有的則成功得標，I2 設為 1，若是沒有則需要進行第二次調解，則又發現經過第一次調解後發現賣家 I1+I2 的總物品量為 3

335，則買家總物品量為3242，發現說雖然第一項 K1 商品已經滿足買家，但又發現 K3 商品還是差了 1 個單位的物品量，則在從賣家 I3 下手看是否可以滿足其賣家殘餘物品量 K3，結果經由第二次調解可以看到 I1+I2+I3 的物品量總和為 5645，則買家的物品量 N1+N2+N3 的物品總和為 3242，的確賣家的總物品量有滿足買家，則往下看而可以發現說賣家目前總量有 5646，而買家經過第二次調解的殘餘量則還有 1303，從賣家 5646 減去 3242 還多了 2403 的殘餘量，則可以發現說賣家最後的殘餘量有滿足買家 N4 沒得標的殘餘量並且還多了賣家第 K1 及 K2 各 1 項商品單位量，但又發現說賣家殘餘量不滿足額外的 N5 及 N6 所以使得只有 N1-N4 得標，而沒有其餘的標給 N5、N6，使得雙方都 Match，雙方得標調解成功。

表 A-9：第一次調解_ I3N6K4

#I	Var	Bid	K1	K2	K3	K4	P
I1	X11	1	1	2	3	2	47
I2	X21	1	2	1	0	3	20
I3	X31	0	2	3	1	1	20
Sum	X		3	3	3	5	

#N	#Var	Bid	K1	K2	K3	K4	P
N1	Y11	1	1	0	3	0	40
N2	Y21	1	0	2	0	2	38
N3	Y31	1	2	0	1	0	32
N4	Y41	0	1	3	0	3	80
N3	Y51	0	2	1	0	3	90
N4	Y61	0	3	2	1	0	100
Sum	Y		3	2	4	2	

由表 A-9 進行第一次調解是因為賣家第一項物品及第三項物品有缺不

足買家物品需求量 1 項於 K1 及 K3 商品，則賣家 K1 商品得標的總和量只有 1 而買家的 1 商品得標總量有 3，則因為 I2 及 I3 的第 K1 商品都可以滿足並且數量一樣，則按照順序來先從 I2 下手，則經由第一次調解可以發現賣家總物品量為 3335，則買家總物品量為 3242，則賣家第 K1 項商品的確滿足了買家 K1 的商品量，但是又可以發現賣家的 K3 商品還是不滿足買家 K3 商品的物品量 1 項單位，所以必須進行第二次調解。

表 A-10：第二次調解_ I3N6K4

#I	Var	Bid	K1	K2	K3	K4	P
I1	X11	1	1	2	3	2	47
I2	X21	1	2	1	0	3	20
I3	X31	1	2	3	1	1	20
Sum	X		5	6	4	6	

#N	#Var	Bid	K1	K2	K3	K4	P
N1	Y11	1	1	0	3	0	40
N2	Y21	1	0	2	0	2	38
N3	Y31	1	2	0	1	0	32
N4	Y41	0	1	3	0	3	80
N5	Y51	0	2	1	0	3	90
N6	Y61	0	3	2	1	0	100
Sum	Y		3	2	4	2	

經由表 A-10 所進行第二次的調解中而可以發現說賣方目前總量有 5646，而買家經過第二次調解的殘餘量則還有 1303，從賣家 5646 減去 3242 還多了 2404 的殘餘量，則可以發現說賣家最後的殘餘量有滿足買方 N4 沒得標的殘餘量並且還多了賣家第 K1 及 K2 各 1 項商品單位量，但因

為賣家沒有多餘的殘餘量可以滿足買家 N5 及 N6 的物品，調解結束。結果如表 A-11：

表 A-11：調解結束_I3N6K4

#I	Var	Bid	K1	K2	K3	K4	P
I1	X11	1	1	2	3	2	47
I2	X21	1	2	1	0	3	20
I3	X31	1	2	3	1	1	20
Sum	X		5	6	4	6	

#N	#Var	Bid	K1	K2	K3	K4	P
N1	Y11	1	1	0	3	0	40
N2	Y21	1	0	2	0	2	38
N3	Y31	1	2	0	1	0	32
N4	Y41	1	1	3	0	3	80
N5	Y51	0	2	1	0	3	90
N6	Y61	0	3	2	1	0	100
Sum	Y		4	5	4	5	

結果=>　x11=1，x21=1，x31=1　y11=1，y21=1，y31=1，y41=1，y51=0，y61=0

圖 A-4 為 I3N6K4 的程式調解圖：

```
[調解偵測Strart:]------------------------------------------------------------
開始計算賣家得標商品的總和：
sum_x[1]= 1,

sum_x[2]= 2,

sum_x[3]= 3,

sum_x[4]= 2,

開始計算買家得標商品的總和：
sum_y[1]= 3,

sum_y[2]= 2,

sum_y[3]= 4,

sum_y[4]= 2,

查看賣家總和是否有滿足買家總和:
----------------------
sum_xy[1]= -2,
因爲賣家總和並沒有滿足買家總和，賣家目前尚缺以下產品:k[1]= 2, 需要進行調解,
sum_xy[2]= 0,
因爲賣家總和有滿足買家總和，所以保留:
sum_xy[3]= -1,
因爲賣家總和並沒有滿足買家總和，賣家目前尚缺以下產品:k[3]= 1, 需n進行調解,
sum_xy[4]= 0,
因爲賣家總和有滿足買家總和，所以保留:
----------------------
因爲已知不滿足的商品，則開始尋找下個賣家來進行補足.
而賣家尚缺以下產品:k[1]= 2, 而賣家尚缺以下產品:k[3]= 1,

    [調解開始Strart:]-----------------------------------------------
    測試Jth是否爲1，若爲1則此投標有效，反則無效
    j爲1所以代表有得標，第x[1][1]=1
    j爲1所以代表有得標，第x[2][1]=1
    j爲1所以代表有得標，第x[3][1]=1
    測試完Jth之後確認賣家i的k項商品，若爲>0則此投標保留，反則無效
    k大於1所以，第x[1][1]*q[1][1][1]=1
    k大於1所以，第x[1][1]*q[1][1][2]=2
    k大於1所以，第x[1][1]*q[1][1][3]=3
    k大於1所以，第x[1][1]*q[1][1][4]=2
    k大於1所以，第x[2][1]*q[2][1][1]=2
    k大於1所以，第x[2][1]*q[2][1][2]=1
    k大於1所以，第x[2][1]*q[2][1][3]=0
    k大於1所以，第x[2][1]*q[2][1][4]=3
    k大於1所以，第x[3][1]*q[3][1][1]=2
    k大於1所以，第x[3][1]*q[3][1][2]=3
    k大於1所以，第x[3][1]*q[3][1][3]=1
    k大於1所以，第x[3][1]*q[3][1][4]=1
    開始計算賣家得標商品的總和：
    sum_x[1]= 5,

    sum_x[2]= 6,

    sum_x[3]= 4,

    sum_x[4]= 6,

    開始計算買家得標商品的總和：
    sum_y[1]= 3,

    sum_y[2]= 2,

    sum_y[3]= 4,

    sum_y[4]= 2,
```

```
查看賣家總和是否有滿足買家總和:
----------------------
sum_xy[1]= 2,
因爲賣家總和有滿足買家總和，所以保留:
sum_xy[2]= 4,
因爲賣家總和有滿足買家總和，所以保留:
sum_xy[3]= 0,
因爲賣家總和有滿足買家總和，所以保留:
sum_xy[4]= 4,
因爲賣家總和有滿足買家總和，所以保留:
程式結束
當c= 1.0 時, L= 100
顯示得標爲：
x*[1][1]
x*[2][1]
x*[3][1]
y*[1][1]
y*[2][1]
y*[3][1]

lambda*[k]={{6.6, 3.0, 4.8, 3.0, }}
L(lambda *)=100.0
----------------------
check_win(n)_imformation
win[n]=0=>win[4]=>y[4][1]=0
y[4][1]=1
程式結束
當c= 1.0 時, L= 100
顯示得標爲：
x*[1][1]
x*[2][1]
x*[3][1]
y*[1][1]
y*[2][1]
y*[3][1]
y*[4][1]

lambda*[k]={{6.6, 3.0, 4.8, 3.0, }}
L(lambda *)=100.0
```

```
開始計算買家得標商品的總和：
sum_x[1]= 5,

sum_x[2]= 6,

sum_x[3]= 4,

sum_x[4]= 6,

開始計算買家得標商品的總和：
sum_y[1]= 4,

sum_y[2]= 5,

sum_y[3]= 4,

sum_y[4]= 5,

update_differece_x_y(n,h)_imformation
sum_xy[1]= 1,
sum_xy[2]= 1,
sum_xy[3]= 0,
sum_xy[4]= 1,
程式結束
當c= 1.0 時, L= 100
顯示得標為：
x*[1][1]
x*[2][1]
x*[3][1]
y*[1][1]
y*[2][1]
y*[3][1]
y*[4][1]

lambda*[k]={{6.6, 3.0, 4.8, 3.0, }}
L(lambda *)=100.0

[調解結束End]----------------------------------------------------------------
程式終止
win_cost_i=(72.0)
win_cost_n=(10.0)

F(X_bar,Y_bar)=xij((pij-pij_will)+ynh((pnh_will-pnh)=(72.0)+(10.0)=82.0
L(lambda *)=100.0
Duality GAP=[F(X_bar,Y_bar)-L(lambda *)/F(X_bar,Y_bar)]=0.21951219512195122%
Time = 0ms(毫秒)

Process completed.
```

圖 A-4：調解結果列印_I3N6K4

例子 I4N7K5 考慮了四位賣家及七位買家，不過商品提昇至 5 項(I=4，N=7，K=5)，根據例子說明，將有以下參數(I 為賣家，N 為買家，K 為商品)：

表 A-12：賣家 I 所提供的商品數量_I4N7K5

Item / Seller	Item 1	Item 2	Item 3	Item 4	Item 5
Seller 1	3	2	1	0	2
Seller 2	1	3	0	1	1
Seller 3	2	1	2	3	0
Seller 4	2	1	3	2	3
Total	8	7	6	6	6

根據表 A-12，將有以下參數被設定：

q_{ijk} = 賣家 i 對第 j 次投標所提供商品 k 的量。

$$q_{111}=3,\ q_{112}=2,\ q_{113}=1, q_{114}=0, q_{115}=2$$
$$q_{211}=1,\ q_{212}=3,\ q_{213}=0, q_{214}=1, q_{215}=1$$
$$q_{311}=2,\ q_{312}=1,\ q_{313}=2, q_{314}=3, q_{315}=0$$
$$q_{411}=2,\ q_{412}=1,\ q_{413}=3, q_{414}=2, q_{415}=3$$

p_{ij} = 賣家 i 於第 j 次投標的價格。

$p_{ij} \Rightarrow p_{11}=47$，$p_{21}=20$，$p_{31}=20$，$p_{41}=15$

p_{ij}^{WILL} = 賣家 i 接受第 j 次投標商品最小賣出的價格。

$$p_{ij}^{WILL}=5$$

表 A-13：買家 N 所需求的商品數量_I4N7K5

Item / Buyer	Item 1	Item 2	Item 3	Item 4	Item 5
Buyer 1	1	4	1	0	2
Buyer 2	2	0	0	0	1
Buyer 3	1	0	2	1	0
Buyer 4	1	1	3	4	2
Buyer 5	4	3	1	1	2
Buyer 6	2	3	1	0	1
Buyer 7	3	0	2	1	0
Total	14	11	10	7	8

根據表 A-13，將有以下參數被設定：

d_{nhk} = 買家 n 對第 h 次招標所需求商品 k 的量。

$$d_{111}=1,\ d_{112}=4,\ d_{113}=1, d_{114}=0, d_{115}=2$$
$$d_{211}=2, d_{212}=0,\ d_{213}=0, d_{214}=0, d_{215}=1$$
$$d_{311}=1,\ d_{312}=0,\ d_{313}=2, d_{314}=1, d_{315}=0$$
$$d_{411}=1,\ d_{412}=1,\ d_{413}=3, d_{414}=4, d_{415}=2$$
$$d_{511}=4,\ d_{512}=3,\ d_{513}=1, d_{514}=1, d_{515}=2$$
$$d_{611}=2,\ d_{612}=3,\ d_{613}=1, d_{614}=0, d_{615}=1$$
$$d_{711}=3,\ d_{712}=0,\ d_{713}=2, d_{714}=1, d_{715}=0$$

p_{nh} = 買家 n 於第 h 次投標的價格

p_{nh} => $p_{11}=40$， $p_{21}=38$ ， $p_{31}=32$， $p_{41}=80$， $p_{51}=100$， $p_{61}=75$， $p_{71}=85$

p_{nh}^{WILL} = 買家n接受第h次招標商品最大買進的價格。

$$p_{nh}^{WILL} = 50$$

初始化 Lagrange multipliers： $\lambda(k) = 3.0, \overline{L} = 100$

圖 A-5 為拍賣結果列印：

```
[演算法Strart:]----------------------------------------------------------------------------------------
iteration = 1
lambda_k=15.0
買家得標為：x[1][1]=1 , x[2][1]=0 , x[3][1]=0 , x[4][1]=0 ,
買家得標為：y[1][1]=1 , y[2][1]=1 , y[3][1]=1 , y[4][1]=0 , y[5][1]=0 , y[6][1]=0 , y[7][1]=0 ,
L(lambda)=xij((pij-pij_will)-lambda_k*qijk)+ynh((pnh_will-pnh)-lambda_k*dnhk)=18.0+91.0=109.0
g[1]= 1,
g[2]= 2,
g[3]= 3,
g[4]= 2,
g[5]= 1,
sum_gk=19
alpha=-0.47 lambda[k]2.53, lambda[k]2.06, lambda[k]1.59, lambda[k]2.06, lambda[k]2.53, epsilon=Infinity
----------------------------------------------------------------------------------------
[演算法Strart:]----------------------------------------------------------------------------------------
iteration = 2
lambda_k=10.77
買家得標為：x[1][1]=1 , x[2][1]=1 , x[3][1]=0 , x[4][1]=0 ,
買家得標為：y[1][1]=1 , y[2][1]=1 , y[3][1]=1 , y[4][1]=0 , y[5][1]=0 , y[6][1]=0 , y[7][1]=0 ,
L(lambda)=xij((pij-pij_will)-lambda_k*qijk)+ynh((pnh_will-pnh)-lambda_k*dnhk)=25.34+76.43=101.77000000000001
g[1]= 0,
g[2]= -1,
g[3]= 3,
g[4]= 1,
g[5]= 0,
sum_gk=11
alpha=-0.16 lambda[k]2.53, lambda[k]2.22, lambda[k]1.11, lambda[k]1.9, lambda[k]2.53, epsilon=-0.06633027522935771
----------------------------------------------------------------------------------------
程式結束
當c= 1.0 時, L= 100
顯示得標為：
x*[1][1]
x*[2][1]
y*[1][1]
y*[2][1]
y*[3][1]

lambda*[k]={{2.53, 2.22, 1.11, 1.9, 2.53, }}
L(lambda *)=101.77000000000001
----------------------
Time = 0ms(毫秒)
```

圖 A-5：拍賣結果列印_I4N7K5

由圖 A-5 進行推算的拍賣列印結果分析如表 A-14：

表 A-14：拍賣列印的結果分析_I4N7K5

#I	Var	Bid	K1	K2	K3	K4	K5	P
I1	X11	1	3	2	1	0	2	47
I2	X21	1	1	3	0	1	1	20
I3	X31	0	2	1	2	3	0	20
I4	X41	0	2	1	3	2	3	15
Sum	X		4	5	1	1	3	

#N	#Var	Bid	K1	K2	K3	K4	K4	P
N1	Y11	1	1	4	1	0	2	40
N2	Y21	1	2	0	0	0	1	38
N3	Y31	1	1	0	3	2	0	32
N4	Y41	0	1	1	2	4	2	80
N5	Y51	0	4	3	1	1	2	100
N6	Y61	0	2	3	1	0	1	80
N7	Y71	0	3	0	2	0	1	70
Sum	Y		4	4	4	2	3	

結果：目前 x11=1，x21=1，x31=0，x41=0 y11=1，y21=1，y31=1，y41=0，y51=0，y61=0，y71=0

其得標的賣家只有 I1 及 I2 則買家有 N1、N2 及 N3，計算賣家的總物品量總和之後查詢有沒有大於買家得標的總和，結果發現賣家總和為 4 5 1 1 3，則買家總和為 4 4 4 1 3，則可以看出賣家的物品量缺額為第 K3 項差了 3 項單元物品量要補足買家，則需要進行調解而因為要先補足 K3 的物品量則往下查詢 I3 及 I4 的 K3 條件發覺說兩項都是有辦法補足買家 K3 不足的物品量，先按照順序先從 I3 下手看是否可以補足與大於其買家的物品量。若是有的則成功得標，I3 設為 1，若是沒有則需要進行第二次調解，則又發現

經過第一次調解後發現賣家 I1+I2+I3 的總物品量為 6 6 3 4 3，則買家總物品量為 4 4 4 1 3，發現 K3 商品還是差了 1 個單位的物品量，則在從賣家 I4 下手看是否可以滿足其賣家殘餘物品量 K3，結果經由第二次調解可以看到 I1+I2+I3+I4 的物品量總和為 8 7 6 6 6，則買家的物品量 N1+N2+N3 的物品總和為 4 4 4 2 3，的確賣家的總物品量有滿足買家，則往下看而可以發現說賣家目前總量有 8 7 6 6 6，而買家經過第二次調解的殘餘量則還有 3 3 2 4 3，從賣家 8 7 6 6 6 減去 4 4 4 2 3 還多了 4 3 2 4 3 的殘餘量，則可以發現說賣家最後的殘餘量有滿足買家 N4 沒得標的殘餘量並且還多了 3 2 0 0 1 的殘餘量，但又發現說賣家殘餘量不滿足額外的 N5、N6 及 N7 所以使得只有 N1-N4 得標，而沒有其餘的標給 N6、N7，調解結束。

表 A-15：第一次調解_I4N7K5

#I	Var	Bid	K1	K2	K3	K4	K5	P
I1	X11	1	3	2	1	0	2	47
I2	X21	1	1	3	0	1	1	20
I3	X31	1	2	1	2	3	0	20
I4	X41	0	2	1	3	2	3	15
Sum	X		6	6	3	4	3	

#N	#Var	Bid	K1	K2	K3	K4	K4	P
N1	Y11	1	1	4	1	0	2	40
N2	Y21	1	2	0	0	0	1	38
N3	Y31	1	1	0	3	2	0	32
N4	Y41	0	1	1	2	4	2	80
N5	Y51	0	4	3	1	1	2	100
N6	Y61	0	2	3	1	0	1	75
N7	Y71	0	3	0	2	0	1	85
Sum	Y		4	4	4	2	3	

由表 A-15 所進行的第一次調解是因為賣家第三項物品有缺不足買家物品需求量 2 項於 K3 商品，則賣家 K3 商品得標的總和量只有 3 而買家的 K3 商品得標總量有 6，則因為 I3 及 I4 的第 K3 商品都可以滿足並且 I3 可以先用低商品滿足，則按照順序來先從 I3 下手，則經由第一次調解可以發現賣家總物品量為 6 6 3 4 3，則買家總物品量為 4 4 4 2 3，則賣家第 K1 項商品的確滿足了買家 K1 的商品量，但是又可以發現賣家的 K3 商品還是不滿足買家 K3 商品的物品量 1 項單位，所以必須進行第二次調解。

表 A-16：第二次調解_I4N7K5

#I	Var	Bid	K1	K2	K3	K4	K5	P
I1	X11	1	3	2	1	0	2	47
I2	X21	1	1	3	0	1	1	20
I3	X31	1	2	1	2	3	0	20
I4	X41	1	2	1	3	2	3	15
Sum	X		8	7	6	6	6	

#N	#Var	Bid	K1	K2	K3	K4	K4	P
N1	Y11	1	1	4	1	0	2	40
N2	Y21	1	2	0	0	0	1	38
N3	Y31	1	1	0	3	2	0	32
N4	Y41	0	1	1	2	4	2	80
N5	Y51	0	4	3	1	1	2	100
N6	Y61	0	2	3	1	0	1	75
N7	Y71	0	3	0	2	0	1	85
Sum	Y		4	4	4	2	3	

經由第二次的調解中而可以發現說賣家目前總量有 8 7 6 6 6，而買家經過第二次調解的殘餘量則還有 4 4 4 2 3，從賣家 8 7 6 6 6 減去 4 4 4 2 3 還多

了 4 3 2 4 3 的殘餘量，則可以發現說賣家最後的殘餘量有滿足買家 N4 沒得標的殘餘量並且還多了 3 2 0 0 1 的殘餘量，但因為賣家沒有多餘的殘餘量可以滿足買家 N5、N6 及 N7 的物品，則程式終止。調解結束。則結果如表 A-17：

表 A-17：調解結束_ I4N7K5

#I	Var	Bid	K1	K2	K3	K4	K5	P
I1	X11	1	3	2	1	0	2	47
I2	X21	1	1	3	0	1	1	20
I3	X31	1	2	1	2	3	0	20
I4	X41	1	2	1	3	2	3	15
Sum	X		8	7	6	6	6	

#N	#Var	Bid	K1	K2	K3	K4	K4	P
N1	Y11	1	1	4	1	0	2	40
N2	Y21	1	2	0	0	0	1	38
N3	Y31	1	1	0	3	2	0	32
N4	Y41	1	1	1	2	4	2	80
N5	Y51	0	4	3	1	1	2	100
N6	Y61	0	2	3	1	0	1	75
N7	Y71	0	3	0	2	0	1	85
Sum	Y		5	5	6	6	5	

結果=>　x11=1，x21=1，x31=1，x41=1 y11=1，y21=1，y31=1，y41=1，y51=0，y61=0，y71=0

圖 A-6 為 I4N7K5 的程式調解圖：

```
[調解偵測Strart:]------------------------------------------------------------
開始計算賣家得標商品的總和：
sum_x[1]= 4,

sum_x[2]= 5,

sum_x[3]= 1,

sum_x[4]= 1,

sum_x[5]= 3,

開始計算買家得標商品的總和：
sum_y[1]= 4,

sum_y[2]= 4,

sum_y[3]= 4,

sum_y[4]= 2,

sum_y[5]= 3,

查看賣家總和是否有滿足買家總和:
----------------------
sum_xy[1]= 0,
因爲賣家總和有滿足買家總和，所以保留:
sum_xy[2]= 1,
因爲賣家總和有滿足買家總和，所以保留:
sum_xy[3]= -3,
因爲賣家總和並沒有滿足買家總和，賣家目前尚缺以下產品:k[3]= 3, 需要進行調解,
sum_xy[4]= -1,
因爲賣家總和並沒有滿足買家總和，賣家目前尚缺以下產品:k[4]= 1, 需要進行調解,
sum_xy[5]= 0,
因爲賣家總和有滿足買家總和，所以保留:
----------------------
因爲已知不滿足的商品，則開始尋找下個賣家來進行補足.
而賣家尚缺以下產品:k[3]= 3, 而賣家尚缺以下產品:k[4]= 1,
[調解開始Strart:]------------------------------------------------
測試Jth是否爲1，若爲1則鴣趕虷陵幕A反則無效
j爲1所以代表有得標，第x[1][1]=1
j爲1所以代表有得標，第x[2][1]=1
j爲1所以代表有得標，第x[3][1]=1
j爲1所以代表有得標，第x[4][1]=1
測試完Jth之後確認賣家i的k項商品，若爲>0則此投標保留，反則無效
k大於1所以，第x[1][1]*q[1][1][1]=3
k大於1所以，第x[1][1]*q[1][1][2]=2
k大於1所以，第x[1][1]*q[1][1][3]=1
k大於1所以，第x[1][1]*q[1][1][4]=0
k大於1所以，第x[1][1]*q[1][1][5]=2
k大於1所以，第x[2][1]*q[2][1][1]=1
k大於1所以，第x[2][1]*q[2][1][2]=3
k大於1所以，第x[2][1]*q[2][1][3]=0
k大於1所以，第x[2][1]*q[2][1][4]=1
k大於1所以，第x[2][1]*q[2][1][5]=1
k大於1所以，第x[3][1]*q[3][1][1]=2
k大於1所以，第x[3][1]*q[3][1][2]=1
k大於1所以，第x[3][1]*q[3][1][3]=2
k大於1所以，第x[3][1]*q[3][1][4]=3
k大於1所以，第x[3][1]*q[3][1][5]=0
k大於1所以，第x[4][1]*q[4][1][1]=2
k大於1所以，第x[4][1]*q[4][1][2]=1
k大於1所以，第x[4][1]*q[4][1][3]=3
k大於1所以，第x[4][1]*q[4][1][4]=2
k大於1所以，第x[4][1]*q[4][1][5]=3
```

開始計算賣家得標商品的總和：
sum_x[1]= 8,

sum_x[2]= 7,

sum_x[3]= 6,

sum_x[4]= 6,

sum_x[5]= 6,

開始計算買家得標商品的總和：
sum_y[1]= 4,

sum_y[2]= 4,

sum_y[3]= 4,

sum_y[4]= 2,

sum_y[5]= 3,

查看賣家總和是否有滿足買家總和:
----
sum_xy[1]= 4,
因爲賣家總和有滿足買家總和，所以保留:
sum_xy[2]= 3,
因爲賣家總和有滿足買家總和，所以保留:
sum_xy[3]= 2,
因爲賣家總和有滿足買家總和，所以保留:
sum_xy[4]= 4,
因爲賣家總和有滿足買家總和，所以保留:
sum_xy[5]= 3,
因爲賣家總和有滿足買家總和，所以保留:
程式結束
當c= 1.0 時, L= 100
顯示得衧陛G
x*[1][1]
x*[2][1]
x*[3][1]
x*[4][1]
y*[1][1]
y*[2][1]
y*[3][1]

lambda*[k]={{2.53, 2.22, 1.11, 1.9, 2.53, }}
L(lambda *)=101.770000000000001

check_win(n)_imformation
win[n]=0=>win[4]=>y[4][1]=0
y[4][1]=1
程式結束
當c= 1.0 時, L= 100
顯示得標爲：
x*[1][1]
x*[2][1]
x*[3][1]
x*[4][1]
y*[1][1]
y*[2][1]
y*[3][1]
y*[4][1]

lambda*[k]={{2.53, 2.22, 1.11, 1.9, 2.53, }}
L(lambda *)=101.770000000000001

```
開始計算買家得標商品的總和：
sum_x[1]= 8,

sum_x[2]= 7,

sum_x[3]= 6,

sum_x[4]= 6,

sum_x[5]= 6,

開始計算買家得標商品的總和：
sum_y[1]= 5,

sum_y[2]= 5,

sum_y[3]= 6,

sum_y[4]= 6,

sum_y[5]= 5,

update_differece_x_y(n,h)_imformation
sum_xy[1]= 3,
sum_xy[2]= 2,
sum_xy[3]= 0,
sum_xy[4]= 0,
sum_xy[5]= 1,
程式結束
當c= 1.0 時, L= 100
顯示得標為：
x*[1][1]
x*[2][1]
x*[3][1]
x*[4][1]
y*[1][1]
y*[2][1]
y*[3][1]
y*[4][1]

lambda*[k]={{2.53, 2.22, 1.11, 1.9, 2.53, }}
L(lambda *)=101.77000000000001

-----------------------
[調解結束End]-------------------------------------------------------------------
程式終止
win_cost_i=(82.0)
win_cost_n=(10.0)

F(X_bar,Y_bar)=xij((pij-pij_will)+ynh((pnh_will-pnh)=(82.0)+(10.0)=92.0
L(lambda *)=101.77000000000001
Duality GAP=[F(X_bar,Y_bar)-L(lambda *)/F(X_bar,Y_bar)]=0.10619565217391315%
Time = 0ms(毫秒)

Process completed.
```

圖 A-6：調解結果列印_I4N7K5

例子 I5N8K7 考慮了五位賣家及八位買家，不過商品提昇至 7 項(I=5，N=8，K=7)，根據例子說明，將有以下參數(I 為賣家，N 為買家，K 為商品)：

表 A-18：賣家 I 所提供的商品數量_I5N8K7

Item / Seller	Item 1	Item 2	Item 3	Item 4	Item 5	Item 6	Item 7
Seller 1	2	1	2	3	0	1	2
Seller 2	1	3	0	1	1	2	0
Seller 3	3	2	1	0	2	0	3
Seller 4	2	1	3	2	3	3	1
Seller 5	1	1	4	2	4	1	3
Total	9	8	10	8	10	7	9

根據表 A-18，將有以下參數被設定：

q_{ijk} = 賣家 i 對第 j 次投標所提供商品 k 的量。

$$q_{111}=2,\ q_{112}=1,\ q_{113}=2,q_{114}=3,q_{115}=0,q_{116}=1,q_{117}=2$$
$$q_{211}=1,\ q_{212}=3,\ q_{213}=0,q_{214}=1,q_{215}=1,q_{216}=2,q_{217}=0$$
$$q_{311}=3,\ q_{312}=2,\ q_{313}=1,q_{314}=0,q_{315}=2,q_{316}=0,q_{317}=3$$
$$q_{411}=2,\ q_{412}=1,\ q_{413}=3,q_{414}=2,q_{415}=3,q_{416}=3,q_{417}=1$$
$$q_{511}=1,\ q_{512}=1,\ q_{513}=4,q_{514}=2,q_{515}=4,q_{516}=1,q_{517}=3$$

p_{ij} = 賣家 i 於第 j 次投標的價格。

p_{ij} => $p_{11}=60$，$p_{21}=55$，$p_{31}=20$，$p_{41}=18$，$p_{51}=16$

p_{ij}^{WILL} = 賣家 i 接受第 j 次投標商品最小賣出的價格。

$$p_{ij}^{WILL}=5$$

表 A-19：買家 N 所需求的商品數量_I5N8K7

Item / Buyer	Item 1	Item 2	Item 3	Item 4	Item 5	Item 6	Item 7
Buyer 1	1	0	2	0	0	1	0
Buyer 2	1	1	0	3	1	0	2
Buyer 3	1	2	1	2	1	2	1
Buyer 4	2	2	1	1	1	0	1
Buyer 5	2	4	2	3	4	3	4
Buyer 6	2	1	4	0	3	1	1
Buyer 7	1	0	0	0	1	0	2
Buyer 8	1	2	3	1	2	2	1
Total	10	10	12	8	12	8	10

根據表 A-19，將有以下參數被設定：

d_{nhk} = 買家 n 對第 h 次招標所需求商品 k 的量。

$$d_{111}=1,\ \mathrm{d}_{112}=0,\ \mathrm{d}_{113}=2, d_{114}=0, d_{115}=0, d_{116}=1, d_{117}=0$$
$$d_{211}=1, \mathrm{d}_{212}=1,\ \mathrm{d}_{213}=0, d_{214}=3, d_{115}=1, d_{216}=0, d_{217}=2$$
$$d_{311}=1,\ \mathrm{d}_{312}=2,\ \mathrm{d}_{313}=1, d_{314}=2, d_{115}=1, d_{316}=2, d_{317}=1$$
$$d_{411}=2,\ \mathrm{d}_{412}=2,\ \mathrm{d}_{413}=1, d_{414}=1, d_{115}=1, d_{416}=0, d_{417}=1$$
$$d_{511}=2,\ \mathrm{d}_{512}=4,\ \mathrm{d}_{513}=2, d_{514}=3, d_{115}=4, d_{516}=3, d_{517}=4$$
$$d_{611}=2,\ \mathrm{d}_{612}=1,\ \mathrm{d}_{613}=4, d_{614}=0, d_{615}=3, d_{616}=1, d_{617}=1$$
$$d_{711}=1,\ \mathrm{d}_{712}=0\ \mathrm{d}_{713}=0, d_{714}=0, d_{715}=1, d_{716}=0, d_{717}=2$$
$$d_{811}=1,\ \mathrm{d}_{812}=2,\ \mathrm{d}_{813}=3, d_{814}=1, d_{815}=2, d_{816}=2, d_{817}=1$$

p_{nh} = 買家 n 於第 h 次投標的價格

$p_{nh} \Rightarrow p_{11}=40$， $p_{21}=38$ ， $p_{31}=32$， $p_{41}=92$， $p_{51}=140$， $p_{61}=120$, $p_{71}=100$， $p_{81}=130$

p_{nh}^{WILL} = 買家 n 接受第 h 次招標商品最大買進的價格。

$$p_{nh}^{WILL}=60$$

初始化 Lagrange multipliers： $\lambda(k)=2.0, \overline{L}=200$

圖 A-7 為拍賣結果列印：

```
iteration = 1
lambda_k=14.0
買家得標為：x[1][1]=1 , x[2][1]=1 , x[3][1]=0 , x[4][1]=0 , x[5][1]=0 ,
買家得標為：y[1][1]=1 , y[2][1]=1 , y[3][1]=1 , y[4][1]=0 , y[5][1]=0 , y[6][1]=0 , y[7][1]=0 , y[8][1]=0 ,
L(lambda)=xij((pij-pij_will)-lambda_k*qijk)+ynh((pnh_will-pnh)-lambda_k*dnhk)=67.0+114.0=181.0
g[1]= 0,
g[2]= -1,
g[3]= 1,
g[4]= 1,
g[5]= 1,
g[6]= 0,
g[7]= 1,
sum_gk=5
alpha=3.8 lambda[k]2.0, lambda[k]0.0, lambda[k]5.8, lambda[k]5.8, lambda[k]5.8, lambda[k]2.0, lambda[k]5.8, eps:
--------------------------------------------------------------------------------
[演算法Strart:]-------------------------------------------------------------------------------------
iteration = 2
lambda_k=27.2
買家得標為：x[1][1]=1 , x[2][1]=1 , x[3][1]=0 , x[4][1]=0 , x[5][1]=0 ,
買家得標為：y[1][1]=1 , y[2][1]=1 , y[3][1]=1 , y[4][1]=0 , y[5][1]=0 , y[6][1]=0 , y[7][1]=0 , y[8][1]=0 ,
L(lambda)=xij((pij-pij_will)-lambda_k*qijk)+ynh((pnh_will-pnh)-lambda_k*dnhk)=40.8+157.4=198.2
g[1]= 0,
g[2]= -1,
g[3]= 1,
g[4]= 1,
g[5]= 1,
g[6]= 0,
g[7]= 1,
sum_gk=5
alpha=0.36 lambda[k]2.0, lambda[k]0.0, lambda[k]6.16, lambda[k]6.16, lambda[k]6.16, lambda[k]2.0, lambda[k]6.16,
--------------------------------------------------------------------------------
[演算法Strart:]-------------------------------------------------------------------------------------
iteration = 3
lambda_k=28.64
買家得標為：x[1][1]=1 , x[2][1]=1 , x[3][1]=0 , x[4][1]=0 , x[5][1]=0 ,
買家得標為：y[1][1]=1 , y[2][1]=1 , y[3][1]=1 , y[4][1]=0 , y[5][1]=0 , y[6][1]=0 , y[7][1]=0 , y[8][1]=0 ,
L(lambda)=xij((pij-pij_will)-lambda_k*qijk)+ynh((pnh_will-pnh)-lambda_k*dnhk)=37.56+162.07999999999998=199.64
g[1]= 0,
g[2]= -1,
g[3]= 1,
g[4]= 1,
g[5]= 1,
g[6]= 0,
g[7]= 1,
sum_gk=5
alpha=0.07 lambda[k]2.0, lambda[k]0.0, lambda[k]6.23, lambda[k]6.23, lambda[k]6.23, lambda[k]2.0, lambda[k]6.23,
```

```
程式結束
當c= 1.0 時, L= 200
顯示得標為：
x*[1][1]
x*[2][1]
y*[1][1]
y*[2][1]
y*[3][1]

lambda*[k]={{2.0, 0.0, 6.23, 6.23, 6.23, 2.0, 6.23, }
L(lambda *)=199.64
----------------------
Time = 16ms(毫秒)
```

圖 A-7：拍賣結果列印_I5N8K7

由圖 A-7 進行推算的拍賣列印結果分析如表 A-20：

表 A-20：拍賣列印的結果分析_I5N8K7

#I	Var	Bid	K1	K2	K3	K4	K5	K6	K7	P
I1	X11	1	2	1	2	3	0	1	2	60
I2	X21	1	1	3	0	1	1	2	0	55
I3	X31	0	3	2	1	0	2	0	3	20
I4	X41	0	2	1	3	2	3	3	1	18
I5	X51	0	1	1	4	2	4	1	3	16
Sum	X		3	4	2	4	1	3	2	

#N	#Var	Bid	K1	K2	K3	K4	K5	K6	K7	P
N1	Y11	1	1	0	2	0	0	1	0	40
N2	Y21	1	1	1	0	3	1	0	2	38
N3	Y31	1	1	2	1	2	1	2	1	32
N4	Y41	0	2	2	1	1	1	0	1	92
N5	Y51	0	2	4	2	3	4	3	4	140
N6	Y61	0	2	1	4	0	3	1	1	120
N7	Y71	0	1	0	0	0	1	0	2	100
N8	Y81	0	1	2	3	1	2	2	1	130
Sum	Y		3	3	3	5	2	3	3	

結果：目前 x11=1，x21=1，x31=0，x41=0，x51=0　y11=1，y21=1，y31=1，y41=0，y51=0，y61=0，y71=0，y81=0

其得標的賣家只有 I1 及 I2 則買家有 N1、N2 及 N3，計算賣家的總物品量總和之後查詢有沒有大於買家得標的總和，結果發現賣家總和為 3 4 2 4 1 3 2，則買家總和為 3 3 3 5 2 3 3 ，則可以看出賣家的物品量缺額為第 K3、K4、K5、K7 項各差了 1 項單元物品量要補足買家，則需要進行調解而因為要先補足 K3 的物品量則往下查詢 I3 及 I4 的 K3 條件發覺說兩項都

是有辦法補足買家 K3 不足的物品量，先按照順序先從 I3 下手看是否可以補足與大於其買家的物品量。若是有的則成功得標，I3 設為 1，若是沒有則需要進行第二次調解，則又發現經過第一次調解後發現賣家 I1+I2+I3 的總物品量為 6634335，則買家總物品量為 3335233，發現 K4 商品還是差了 1 個單位的物品量，則在從賣家 I4 下手看是否可以滿足其賣家殘餘物品量 K4，結果經由第二次調解可以看到 I1+I2+I3+I4 的物品量總和為 8766666，則買家的物品量 N1+N2+N3 的物品總和為 3335233 ，的確賣家的總物品量有滿足買家，則往下看而可以發現說賣家目前總量有 8766666，而買家經過第二次調解的殘餘量則還有 5562734，從賣家 8766666 減去 3335233 還多了 5431433 的殘餘量，則可以發現說賣家最後的殘餘量只有滿足買家 N4 沒得標的殘餘量並且還多了 4362745 單位，但又發現說殘餘量並不滿足 N5~N8 所以使得只有 N1-N4 得標，而沒有其餘的標給 N5~N8，調解成功

表 A-21：第一次調解_I5N8K7

#I	Var	Bid	K1	K2	K3	K4	K5	K6	K7	P
I1	X11	1	2	1	2	3	0	1	2	60
I2	X21	1	1	3	0	1	1	2	0	55
I3	X31	1	3	2	1	0	2	0	3	20
I4	X41	0	2	1	3	2	3	3	1	18
I5	X51	0	1	1	4	2	4	1	3	16
Sum	X		6	6	3	4	3	3	5	

#N	#Var	Bid	K1	K2	K3	K4	K5	K6	K7	P
N1	Y11	1	1	0	2	0	0	1	0	40
N2	Y21	1	1	1	0	3	1	0	2	38
N3	Y31	1	1	2	1	2	1	2	1	32
N4	Y41	0	2	2	1	1	1	0	1	92
N5	Y51	0	2	4	2	3	4	3	4	170
N6	Y61	0	2	1	4	0	3	1	1	120
N7	Y71	0	1	0	0	0	1	0	2	100
N8	Y81	0	1	2	3	1	2	2	1	130
Sum	Y		3	3	3	5	2	3	3	

由表 A-21 進行的第一次調解是因為賣家第三、四、五、七項物品有缺不足買家物品需求量各 1 項於 K3、K4、K5、K7 商品，則賣家 K3 商品得標的總和量只有 2 而買家的 K3 商品得標總量有 3，則因為 I3 及 I4 的第 K3 商品都可以滿足並且 I3 可以先用低商品滿足，則按照順序來先從 I3 下手，則經由第一次調解可以發現賣家總物品量為 6 6 3 4 3 3 5，則買家總物品量為 3 3 3 5 2 3 3，則賣家第 K3 項商品的確滿足了買家 K3 的商品量，但是又可以發現賣家的 K4 商品還是不滿足買家 K4 商品的物品量 1 項單位，所以

必須進行第二次調解。

表 A-22：第二次調解_I5N8K7

#I	Var	Bid	K1	K2	K3	K4	K5	K6	K7	P
I1	X11	1	2	1	2	3	0	1	2	60
I2	X21	1	1	3	0	1	1	2	0	55
I3	X31	1	3	2	1	0	2	0	3	20
I4	X41	1	2	1	3	2	3	3	1	18
I5	X51	0	1	1	4	2	4	1	3	16
Sum	X		8	7	6	6	6	6	6	

#N	#Var	Bid	K1	K2	K3	K4	K5	K6	K7	P
N1	Y11	1	1	0	2	0	0	1	0	40
N2	Y21	1	1	1	0	3	1	0	2	38
N3	Y31	1	1	2	1	2	1	2	1	32
N4	Y41	0	2	2	1	1	1	0	1	92
N5	Y51	0	2	4	2	3	4	3	4	170
N6	Y61	0	2	1	4	0	3	1	1	120
N7	Y71	0	1	0	0	0	1	0	2	100
N8	Y81	0	1	2	3	1	2	2	1	130
Sum	Y		3	3	3	5	2	3	3	

經由第二次的調解中而可以發現說賣家目前總量有 8766666，而買家經過第二次調解的殘餘量則還有 5562734，從賣家 8766666 減去 3335233 還多了 5431433 的殘餘量，則可以發現說賣家 I1+I2+I3+I4 的殘餘量有滿足買家 N4 沒得標的殘餘量並且還多了 4362745 單位，則查看是否有多的賣家可以滿足，進行第三次調解。

表 A-23：第三次調解_I5N8K7

#I	Var	Bid	K1	K2	K3	K4	K5	K6	K7	P
I1	X11	1	2	1	2	3	0	1	2	60
I2	X21	1	1	3	0	1	1	2	0	55
I3	X31	1	3	2	1	0	2	0	3	20
I4	X41	1	2	1	3	2	3	3	1	18
I5	X51	1	1	1	4	2	4	1	3	16
Sum	X		9	8	10	8	8	7	7	

#N	#Var	Bid	K1	K2	K3	K4	K5	K6	K7	P
N1	Y11	1	1	0	2	0	0	1	0	40
N2	Y21	1	1	1	0	3	1	0	2	38
N3	Y31	1	1	2	1	2	1	2	1	32
N4	Y41	0	2	2	1	1	1	0	1	80
N5	Y51	0	2	4	2	3	4	3	4	170
N6	Y61	0	2	1	4	0	3	1	1	120
N7	Y71	0	1	0	0	0	1	0	2	100
N8	Y81	0	1	2	3	1	2	2	1	130
Sum	Y		3	3	3	5	2	3	3	

既然加上賣家 I5 有辦法滿足買家 N4 的殘餘量，則無法滿足買家 N5~N8 的殘餘量，則程式終止。調解結束。則結果如表 A-24：

表 A-24：調解結束_I5N8K7

#I	Var	Bid	K1	K2	K3	K4	K5	K6	K7	P
I1	X11	1	2	1	2	3	0	1	2	60
I2	X21	1	1	3	0	1	1	2	0	55
I3	X31	1	3	2	1	0	2	0	3	20
I4	X41	1	2	1	3	2	3	3	1	18
I5	X51	1	1	1	4	2	4	1	3	16
Sum	X		9	8	10	8	10	7	9	

#N	#Var	Bid	K1	K2	K3	K4	K5	K6	K7	P
N1	Y11	1	1	0	2	0	0	1	0	40
N2	Y21	1	1	1	0	3	1	0	2	38
N3	Y31	1	1	2	1	2	1	2	1	32
N4	Y41	1	2	2	1	1	1	0	1	92
N5	Y51	0	2	4	2	3	4	3	4	170
N6	Y61	0	2	1	4	0	3	1	1	120
N7	Y71	0	1	0	0	0	1	0	2	100
N8	Y81	0	1	2	3	1	2	2	1	130
Sum	Y		5	5	4	6	3	3	4	

結果=>　x11=1，x21=1，x31=1，x41=1，x51=1　y11=1，y21=1，y31=1，y41=1，y51=0，y61=0，y71=0，y81=0

圖 A-8 為 I5N8K7 的程式調解圖：

```
[調解偵測Strart:]--------------
開始計算賣家得標商品的總和：
sum_x[1]= 3,

sum_x[2]= 4,

sum_x[3]= 2,

sum_x[4]= 4,

sum_x[5]= 1,

sum_x[6]= 3,

sum_x[7]= 2,

開始計算買家得標商品的總和：
sum_y[1]= 3,

sum_y[2]= 3,

sum_y[3]= 3,

sum_y[4]= 5,

sum_y[5]= 2,

sum_y[6]= 3,

sum_y[7]= 3,

查看賣家總和是否有滿足買家總和:
----------------------
sum_xy[1]= 0,
因爲賣家總和有滿足買家總和，所以保留:
sum_xy[2]= 1,
因爲賣家總和有滿足買家總和，所以保留:
sum_xy[3]= -1,
因爲賣家總和並沒有滿足買家總和，賣家目前尚缺以下產品:k[3]= 1, 揭n進行調解,
sum_xy[4]= -1,
因爲賣家總和並沒有滿足買家總和，賣家目前尚缺以下產品:k[4]= 1, 需要進行調解,
sum_xy[5]= -1,
因爲賣家總和並沒有滿足買家總和，賣家目前尚缺以下產品:k[5]= 1, 需要進行調解,
sum_xy[6]= 0,
因爲賣家總和有滿足買家總和，所以保留:
sum_xy[7]= -1,
因爲賣家總和並沒有滿足買家總和，賣家目前尚缺以下產品:k[7]= 1, 需要進行調解,
----------------------
因爲已知不滿足的商品，則開始尋找下個賣家來進行補足.
而賣家尚缺以下產品:k[3]= 1, 而賣家尚缺以下產品:k[4]= 1, 而賣家尚缺以下產品:k[5]= 1, 而賣家尚缺以下產品:k[7]= 1,
```

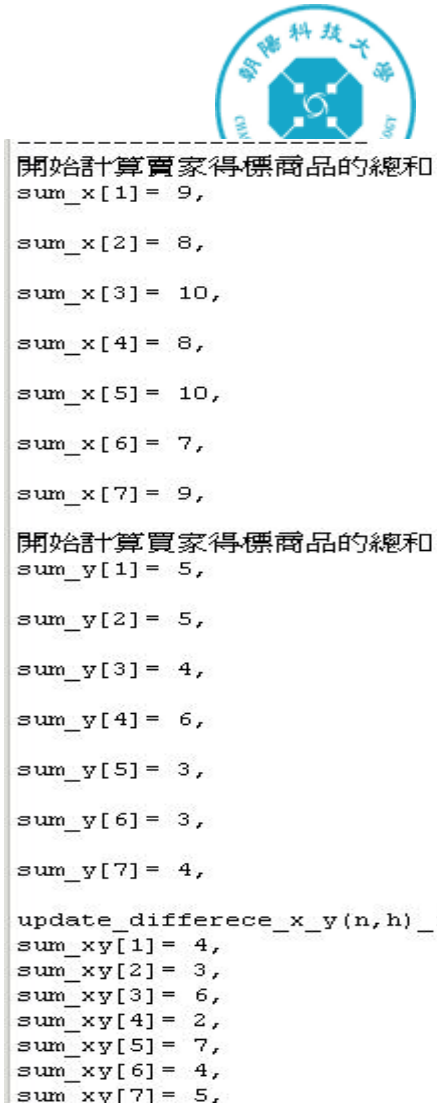
```
[調解開始Strart:]----------------------------------------------------
測試Jth是否為1，若為1則此投標有效，反則無效
j為1所以代表有得標，第x[1][1]=1
j為1所以代表有得標，第x[2][1]=1
j為1所以代表有得標，第x[3][1]=1
j為1所以代表有得標，第x[4][1]=1
j為1所以代表有得標，第x[5][1]=1
測試完Jth之後確認賣家i的k項商品，若為>0則此投標保留，反則無效
k大於1所以，第x[1][1]*q[1][1][1]=2
k大於1所以，第x[1][1]*q[1][1][2]=1
k大於1所以，第x[1][1]*q[1][1][3]=2
k大於1所以，第x[1][1]*q[1][1][4]=3
k大於1所以，第x[1][1]*q[1][1][5]=0
k大於1所以，第x[1][1]*q[1][1][6]=1
k大於1所以，第x[1][1]*q[1][1][7]=2
k大於1所以，第x[2][1]*q[2][1][1]=1
k大於1所以，第x[2][1]*q[2][1][2]=3
k大於1所以，第x[2][1]*q[2][1][3]=0
k大於1所以，第x[2][1]*q[2][1][4]=1
k大於1所以，第x[2][1]*q[2][1][5]=1
k大於1所以，第x[2][1]*q[2][1][6]=2
k大於1所以，第x[2][1]*q[2][1][7]=0
k大於1所以，第x[3][1]*q[3][1][1]=3
k大於1所以，第x[3][1]*q[3][1][2]=2
k大於1所以，第x[3][1]*q[3][1][3]=1

k大於1所以，第x[3][1]*q[3][1][4]=0
k大於1所以，第x[3][1]*q[3][1][5]=2
k大於1所以，第x[3][1]*q[3][1][6]=0
k大於1所以，第x[3][1]*q[3][1][7]=3
k大於1所以，第x[4][1]*q[4][1][1]=2
k大於1所以，第x[4][1]*q[4][1][2]=1
k大於1所以，第x[4][1]*q[4][1][3]=3
k大於1所以，第x[4][1]*q[4][1][4]=2
k大於1所以，第x[4][1]*q[4][1][5]=3
k大於1所以，第x[4][1]*q[4][1][6]=3
k大於1所以，第x[4][1]*q[4][1][7]=1
k大於1所以，第x[5][1]*q[5][1][1]=1
k大於1所以，第x[5][1]*q[5][1][2]=1
k大於1所以，第x[5][1]*q[5][1][3]=4
k大於1猛H，第x[5][1]*q[5][1][4]=2
k大於1所以，第x[5][1]*q[5][1][5]=4
k大於1所以，第x[5][1]*q[5][1][6]=1
k大於1所以，第x[5][1]*q[5][1][7]=3

                開始計算賣家得標商品的總和：
                sum_x[1]= 9,

                sum_x[2]= 8,

                sum_x[3]= 10,

                sum_x[4]= 8,

                sum_x[5]= 10,

                sum_x[6]= 7,

                sum_x[7]= 9,

                開始計算買家得標商品的總和：
                sum_y[1]= 3,

                sum_y[2]= 3,

                sum_y[3]= 3,

                sum_y[4]= 5,

                sum_y[5]= 2,

                sum_y[6]= 3,

                sum_y[7]= 3,
```

查看賈家總和是否有滿足買家總和:
---
sum_xy[1]= 6,
因爲賈家總和有滿足買家總和，所以保留:
sum_xy[2]= 5,
因爲賈家總和有滿足買家總和，所以保留:
sum_xy[3]= 7,
因爲賈家總和有滿足買家總和，所以保留:
sum_xy[4]= 3,
因爲賈家總和有滿足買家總和，所以保留:
sum_xy[5]= 8,
因爲賈家總和有滿足買家總和，所以保留:
sum_xy[6]= 4,
因爲賈家總和有滿足買家總和，所以保留:
sum_xy[7]= 6,
因爲賈家總和有滿足買家總和，所以保留:
程式結束
當c= 1.0 時, L= 200
顯示得標爲：
x*[1][1]
x*[2][1]
x*[3][1]
x*[4][1]
x*[5][1]
y*[1][1]
y*[2][1]
y*[3][1]

lambda*[k]={{2.0, 0.0, 6.23, 6.23, 6.23, 2.0, 6.23, }}
L(lambda *)=199.64
---

check_win(n)_imformation
win[n]=0=>win[4]=>y[4][1]=0
y[4][1]=1
程式結束
當c= 1.0 時, L= 200
顯示得標爲：
x*[1][1]
x*[2][1]
x*[3][1]
x*[4][1]
x*[5][1]
y*[1][1]
y*[2][1]
y*[3][1]
y*[4][1]

lambda*[k]={{2.0, 0.0, 6.23, 6.23, 6.23, 2.0, 6.23, }}
L(lambda *)=199.64



```
----------------------
開始計算賣家得標商品的總和：
sum_x[1]= 9,

sum_x[2]= 8,

sum_x[3]= 10,

sum_x[4]= 8,

sum_x[5]= 10,

sum_x[6]= 7,

sum_x[7]= 9,

開始計算買家得標商品的總和：
sum_y[1]= 5,

sum_y[2]= 5,

sum_y[3]= 4,

sum_y[4]= 6,

sum_y[5]= 3,

sum_y[6]= 3,

sum_y[7]= 4,

update_differece_x_y(n,h)_imformation
sum_xy[1]= 4,
sum_xy[2]= 3,
sum_xy[3]= 6,
sum_xy[4]= 2,
sum_xy[5]= 7,
sum_xy[6]= 4,
sum_xy[7]= 5,
```

```
程式結束
當c= 1.0 時, L= 200
顯示得標為：
x*[1][1]
x*[2][1]
x*[3][1]
x*[4][1]
x*[5][1]
y*[1][1]
y*[2][1]
y*[3][1]
y*[4][1]

lambda*[k]={{2.0, 0.0, 6.23, 6.23, 6.23, 2.0, 6.23, }}
L(lambda *)=199.64
----------------------
[調解結束End]------------------------------------------------------------------------
程式終止
win_cost_i=(144.0)
win_cost_n=(38.0)

F(X_bar,Y_bar)=xij((pij-pij_will)+ynh((pnh_will-pnh)=(144.0)+(38.0)=182.0
L(lambda *)=199.64
Duality GAP=[F(X_bar,Y_bar)-L(lambda *)/F(X_bar,Y_bar)]=0.09692307692307685%
Time = 16ms(毫秒)

Process completed.
```

圖 A-8：調解結果列印_I5N8K7

附錄 B

表 B-1 相關資料_重複改變 I 參數(共 5 個例子)

$\overline{L}=500, C=1.0, \lambda(k)=10.0, N=15, K=5$ 固定，$I=1,3,5,7,9$ 為重複改變值。

(1)I=1，N=15，K=5 資料如下：

Var	(L(lambda)*)/($L(\lambda)^*$)	Duality Gap	Cpu Time
結果	784.29	0.019%	468

(1)I=3，N=15，K=5 資料如下：

Var	(L(lambda)*)/($L(\lambda)^*$)	Duality Gap	Cpu Time
結果	1259.19	0.031%	563

(1)I=5，N=15，K=5 資料如下：

Var	(L(lambda)*)/($L(\lambda)^*$)	Duality Gap	Cpu Time
結果	1736.39	0.029%	656

(1)I=7，N=15，K=5 資料如下：

Var	(L(lambda)*)/($L(\lambda)^*$)	Duality Gap	Cpu Time
結果	2212.8	0.025%	766

(1)I=9，N=15，K=5 資料如下：

Var	(L(lambda)*)/($L(\lambda)^*$)	Duality Gap	Cpu Time
結果	4304.4	0.594%	875

表 B-2 相關資料_重複改變 N 參數(共 5 個例子)

$\bar{L} = 500, C = 1.0, \lambda(k) = 10.0, I = 10, K = 5$ 固定， $N = 2, 4, 6, 8, 10$ 為重複改變值。

(1)I=10，N=2，K=5 資料如下：

Var	(L(lambda)*)/($L(\lambda)^*$)	Duality Gap	Cpu Time
結果	2690	0%	312

(2)I=10，N=4，K=5 資料如下：

Var	(L(lambda)*)/($L(\lambda)^*$)	Duality Gap	Cpu Time
結果	2840	0%	375

(3)I=10，N=6，K=5 資料如下：

Var	(L(lambda)*)/($L(\lambda)^*$)	Duality Gap	Cpu Time
結果	2950	0%	453

(4)I=10，N=8，K=5 資料如下：

Var	(L(lambda)*)/($L(\lambda)^*$)	Duality Gap	Cpu Time
結果	3020	0%	578

(5)I=10，N=10，K=5 資料如下：

Var	(L(lambda)*)/($L(\lambda)^*$)	Duality Gap	Cpu Time
結果	3050	0%	657

表 B-3 相關資料_重複改變 K 參數(共 5 個例子)

$\bar{L} = 500, C = 1.0, \lambda(k) = 10.0, I = 2, N = 4$ 固定， $K = 5, 10, 15, 20, 25$ 為重複改變值。

(1)I=2，N=4，K=5 資料如下：

Var	(L(lambda)*)/($L(\lambda)^*$)	Duality Gap	Cpu Time
結果	790.0	0%	16

(2)I=2，N=4，K=10 資料如下：

Var	(L(lambda)*)/($L(\lambda)^*$)	Duality Gap	Cpu Time
結果	790.0	0%	281

(3)I=2，N=4，K=15 資料如下：

Var	(L(lambda)*)/($L(\lambda)^*$)	Duality Gap	Cpu Time
結果	960.34	0.215%	531

(4)I=2，N=4，K=20 資料如下：

Var	(L(lambda)*)/($L(\lambda)^*$)	Duality Gap	Cpu Time
結果	1416.91	0.793%	782

(5)I=2，N=4，K=25 資料如下：

Var	(L(lambda)*)/($L(\lambda)^*$)	Duality Gap	Cpu Time
結果	2104.56	1.664%	813