FU JEN STUDIES

SCIENCE AND ENGINEERING

No.48, June 2015

CONTENTS

輔仁學誌—理工類

中華民國104年6月　　第四十八期

目　錄

頁次

輔仁學誌 — 理工類

FU JEN STUDIES

-----SCIENCE AND ENGINEERING

發行人 Publisher

江漢聲 校長 （Han-Sun Chiang）

出版者 天主教輔仁大學理工學院

新北市新莊區中正路510號

電話:(02)2905-2411

傳真:(02)2901-4749

輔仁學誌——理工類

第四十八期

FU JEN STUDIES

SCIENCE AND ENGINEERING

No.48, June 2015

輔仁大學理工學院
中華民國104年6月
THE COLLEGE OF SCIENCE AND ENGINEERING
FU JEN CATHOLIC UNIVERSITY
TAIPEI, TAIWAN, R.O.C.

在 CUDA 平台上加速函數對稱性的偵測

王國華　林煥緒

輔仁大學資訊工程系所

摘　要

本篇論文之研究主題是將早期我們所提出之函數對稱性偵測技術 [8] 作平行化，並在 CUDA 統一計算架構 (CUDA–Compute Unified Device Architecture) 下實作並執行對稱性偵測，我們提出兩種 GPU 記憶體存取之最佳化技術 – 全域及共享記憶體最佳化 (GSMO–Global and Shared Memory Optimization) 及全域記憶體最佳化 (GMO–Global Memory Optimization)，在檢查非等效和等效對稱性時，會使用 GSMO 與 GMO 之技術，而在偵測單一變數對稱性時會使用 GMO 之技術。根據實驗結果，對於輸入數目大於 500 與積項數目為 40000 的布林函數，相較於原本在 CPU 上循序執行的檢查方式，GPU 執行的效能在非等效與等效對稱性偵測及單一變數對稱性偵測，分別約有 50 倍及 90 倍的提升，證明我們提出之平行函數對稱性偵測技術在處理大型布林函數時，的確非常有效率。

關鍵字：對稱性偵測、CUDA、GPU、平行計算。

khwang@csie.fju.edu.tw zerox99@csie.fju.edu.tw

1. 前言

對稱性的檢測是針對布林函數的輸入端的對稱性做檢查，它的特性可以應用在邏輯合成 (logic synthesis)[1][2]、技術映射 (technology mapping)[3] 和布林比對 (Boolean matching)[4] 等等，因此一個有效的對稱性檢查演算法是非常重要的。在近幾年已提出許多方法計算函數對稱性的研究 [10]-[15]，如下：有採用重新布局繞線 (rewiring) 的方式分析對稱性 [11, 14]、有使用結構化分析對稱性的方式 [10]、有使用 SAT 工具刪除非對稱性的方式 [12] 和其他的方法 [13, 15]。在過去，我們也提出偵測函數對稱性的演算法 [8]，這個方法是以刪除非對稱性配對集合的方式偵測函數的對稱性，這也是一個非常有效的方法。但在現今的布林函數有著大量的輸入和積項，會使得我們的方法 [8] 計算函數對稱性的效率大幅下降。為了改善此問題，就要將原本循序化的演算法使用平行的概念來加以改良。

本篇論文提出使用圖形處理器 (Graphics Processing Units-GPU) 平行化我們早期研究的演算法 [8]。在 ICCAD (International Conference On Computer-Aided Design) 和 DAC (Design Automation Conference) 的期刊中，也有不少研究 [16-26] 是使用 GPU 去解決 EDA (Electronic Design Automation) 領域的問題。根據 NVIDIA 的技術文件 [9,10] 顯示，現今 GPU 的浮點數運算能力已經比 CPU 高出 8 倍左右。NVIDIA 提供了 CUDA (Compute Unified Device Architecture) 程式語言，讓使用者更容易使用 GPU 的運算資源加速處理大量的資料。我們不只是使用 GPU 平行化早期研究的演算法 [8]，還提出 Global Memory and Shared Memory Optimization (GSMO) 和 Global Memory Optimization (GMO) 兩個技術，讓 GPU 能夠減少讀取全域記憶體的次數，以提升程序的執行效率。在實驗結果的章節裡我們會證明，在 CUDA 平台上並使用 GSMO 和 GMO 兩種技術，是非常有效的。

在接下來的章節之中，第二章介紹與本論文有關之基礎知識；第三章會介紹我們提出的 GSMO 和 GMO 兩種技術；第四章則是我們的實驗結果及分析；最後，我們會有簡單的總結。

2. 背景知識

本節一開始會先介紹函數對稱性的基本定義、我們早期研究的函數對稱性檢查演算法 [8] 和 GPU 的軟體和硬體架構。

2.1 函數對稱性基本介紹

給定一個布林方程式 $f(X)$ 和任意的兩個輸入 x_i 和 x_j ($\{x_i, x_j\} \subset X$)，要對 f 取 x_i 和 x_j 的餘因子，就會產生出四種可能性的組合 $f_{x_i x_j}$、$f_{\bar{x}_i x_j}$、$f_{x_i \bar{x}_j}$ 和 $f_{\bar{x}_i \bar{x}_j}$，透過四種組合的餘因子，可以延伸出不同類型的對稱關係如下：

- 當 $f_{x_i x_j} = f_{\bar{x}_i \bar{x}_j}$ 成立時，就表示 $f(X)$ 針對輸入 x_i 和 x_j 存在等效對稱性關係，表示成 $E(x_i, x_j)$。
- 當 $f_{\bar{x}_i x_j} = f_{x_i \bar{x}_j}$ 成立時，就表示 $f(X)$ 針對輸入 x_i 和 x_j 存在非等效對稱性稱關係，表示成 $NE(x_i, x_j)$。
- 當 $f_{x_i \bar{x}_j} = f_{\bar{x}_i \bar{x}_j}$、$f_{\bar{x}_i x_j} = f_{x_i x_j}$、$f_{\bar{x}_i x_j} = f_{\bar{x}_i \bar{x}_j}$ 和 $f_{x_i \bar{x}_j} = f_{x_i x_j}$ 成立時，就表示 $f(X)$ 針對輸入 x_i 和 x_j 存在單一變數對稱性關係，分別表示成 $SV(x_i, \bar{x}_j)$、$SV(x_i, x_j)$、$SV(x_j, \bar{x}_i)$ 和 $SV(x_j, x_i)$。

以往對稱性之檢查方法，是以計算餘因子的相等性，此作法當函數的輸入數目越大時，會耗費非常大量的計算時間。所以我們先前的研究 [8] 已提出檢查布林函數所有的 on-set 和 off-set 產生的積項配對，刪除不合法的輸入對稱性，留下合法的輸入對稱性。在下面小節會介紹如何利用積項配對移除各種不合法的對稱性配對。

(1) 使用積項配對檢查等效和非等效對稱性

根據我們先前的研究 [8] 可得知不合法的對稱性只會產生在距離在 2 以內的積項配對裡，所以我們只要去檢查距離在 2 以內的積項配對就可以了。一開始先介紹我們使用的符號如下：

- *Distance*(u,v)：表示 u 和 v 兩個積項間的距離，如：*Distance*(u,v) = 1，就表示 u 和 v 兩個積項間的距離是 1。
- *Disjoint*(u,v)：表示 u 和 v 兩個積項間不相交字元的集合，如：*Disjoint*(u,v) = $\{x_i\}$，就表示 u 和 v 兩個積項間 x_i 字元不相交。
- 1-*lit*(u)：表示積項 u 裡字元是 1 的輸入集合，反之 0-*lit*(u) 表示積項 u 裡字元是 0 的輸入集合。
- x_i (u): 表示積項 u 第 i 個字元的數值，如：$u = x_1 \bar{x}_2$，那麼 x_1 (u) = 1。
- *Non-ENE-Symmetries*(u,v)：表示 u 和 v 兩個積項配對可移除之不合法的等效和非等效對稱性配對集合。

給定一個布林函數 f 和兩個積項 $u \in f^{on}$ 和 $v \in f^{off}$，使用下列 3 條規則檢查積項配對並找出非對稱性配對，如下：

Rule 1： *Distance*$(u, v) > 2$，就不存在任何不合法輸入對稱性，所以 *Non-ENE-Symmetries*$(u, v) = \phi$。

Rule 2： *Distance*$(u, v) = 2$ 和 *Disjoint*$(u, v) = \{x_i, x_j\}$，就會存在一組不合法的對稱性配對，所以 *Non-ENE-Symmetries*$(u, v) = \begin{cases} \{x_i, x_j\} : if\ x_i(u) \neq x_j(u) \\ \{x_i, \bar{x}_j\} : if\ x_i(u) = x_j(u) \end{cases}$。

Rule 3： *Distance*$(u, v) = 1$，將會存在多個非對稱性的配對，所以

Non-ENE-Symmetries$(u,v) = \begin{cases} A \times (B \cup \bar{C}) : if\ x_i(u) = 0 \ldots (1) \\ A \times (\bar{B} \cup C) : if\ x_i(u) = 1 \ldots (2) \end{cases}$，當 A = *Disjoint*(u, v) = $\{x_i\}$，B =S (X − *0-lit*(u)) ∩ (X − *1-lit*(v)) − A 和 C = (X − *1-lit*(u)) ∩ (X − *0-lit*(v)) − A。$\bar{B}$ 和 $\bar{C}$ 代表的是 B 和 C 集合裡所有輸入做反向(complemented)。

使用上述的三個規則，就可以對布林方程式 $f(X)$ 中所有的積項配對做檢查，將所有不合法的對稱性配對刪除，留下合法有對稱性的配對。

(2) 使用積項配對檢查單一變數對稱性

根據我們先前的研究 [8] 可得知不合法的單一變數對稱性只會產生在距離在 1 上的積項配對裡，所以我們只要去檢查距離在 1 上的積項配對就可以了。一開始先介紹我們使用的符號如下：

● *Non-SV-Symmetries* (u,v): 表示 u 和 v 兩個積項配對可移除之不合法的單一變數對稱性配對集合。

給定一個布林函數 f 和兩個積項 $u \in f^{on}$ 和 $v \in f^{off}$，使用下列 2 條規則檢查積項配對並找出非對稱性配對，如下：

Rule 1： *Distance*$(u, v) > 1$，就不存在任何不合法輸入對稱性，所以 *Non-SV-Symmetries*$(u, v) = \phi$。

Rule 2： *Distance*$(u, v) = 1$，將會存在多個非對稱性的配對，所以 *Non-SV-Symmetries*$(u, v) = A \times (B \cup \bar{C})$，當 A = *Disjoint*(u, v) = $\{x_i\}$，B = (X − 0-lit(u)) ∩ (X − 0-lit(v)) − A 和 $\bar{C}$ = (X − 1-lit(u)) ∩ (X − 1-lit(v)) − A。$\bar{B}$ 和 $\bar{C}$ 代表的是 B 和 C 集合裡所有輸入做反向。

使用上述的兩個規則，就可以對布林方程式 $f(X)$ 中所有的積項配對做檢查，將所有不合法的對稱性配對刪除，留下合法有對稱性的配對。

2.2 CUDA 編程架構

CUDA 平台是以 CUDA C 語言構成，它是 C/C++ 語言的延伸，是使用編譯器 (nvcc) 進行編譯。CUDA 把 CPU 當作主機端 (host)，把 GPU 當作裝置端 (device)。在 GPU 上運行程序稱為啟動核心 (launch kernel)。一般撰寫 CUDA 的程序，如圖 1，首先會在主機端將需要運算的資料先傳入 GPU 之後，設定程式所需區塊 (block) 和執行緒 (thread) 的數量並啟動核心，運行核心時 CUDA 就會將設定所有的區塊和執行緒組織成一個網格 (grid)，GPU 會以並行的方式處理網格內所有的區塊任務。如圖中 Kernel1 可以顯示，在運行核心前，將程序設定一個區塊有 4 個執行緒和總共有 6 個區塊，運行核心時，CUDA 就將這 6 個區塊組織成 Grid1 並執行。

GPU 處理任務是以區塊為單位，但實際上 GPU 在處理區塊之前，會將區塊內的執行緒以 32 個為單位，切割成若干個 warp。假設一個區塊有 58 個執行緒，GPU 在處理區塊時，就會將之切割成兩個 warp。所以 GPU 實際上處理任務的單位是以 warp 為單位。

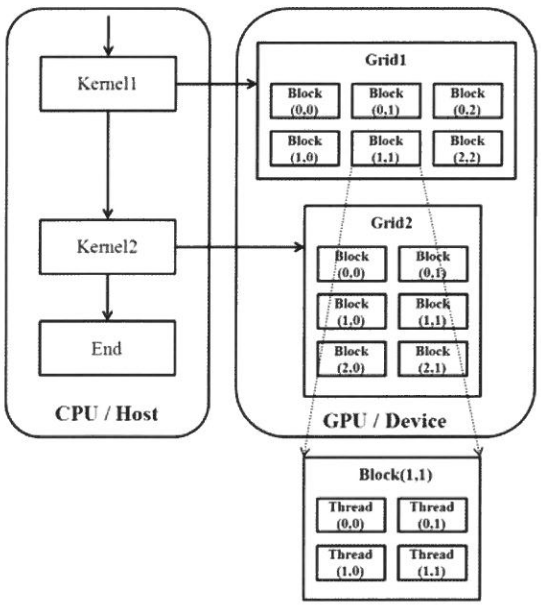


圖 1：CUDA 的編成模型

2.3 GPU 記憶體介紹

GPU 可使用的記憶體包含有暫存器、區域記憶體、共享記憶體、全域記憶體、常數記憶體與材質記憶體，而 CPU 可以透過全域記憶體、常數記憶體與材質記憶體將資料傳遞給 GPU。以下會分開作介紹。

(1) 暫存器

每個執行緒都會有各自的暫存器，其存取的速度是所有記憶體中最快速的，因此當有資料需要經常被存取或計算時，通常都會先將資料存入暫存器裡再進行計算。

(2) 區域記憶體

每個執行緒也都會有各自的區域記憶體，但是其存取速度是非常慢的 (400-600 clock)，所以盡量不要去使用它。

(3) 共享記憶體

每一個區塊都會有各自的共享記憶體，可以讓區塊內的執行緒存取，也就是說區塊內的執行緒如果彼此要溝通，就需要透過共享記憶體來達成，其存取速度僅次於暫存器 (4 clock)，不過一個區塊的共享記憶體的大小最大只有 48KB。

共享記憶體是由 32 個儲存庫所組成的，每個儲存庫的大小是 4bytes。如果當 warp 內超過一個執行緒儲存同一個儲存庫時，就會發生儲存庫衝突，使程序的效能下降，所以在儲存共享記憶體時，最好讓 warp 內執行緒儲存不同編號的儲存庫。

(4) 全域記憶體

在網格裡面所有的區塊和執行緒都可以去存取全域記憶體，它的大小是所有記憶體中最大的。通常 CPU 都是透過全域記憶體將資料傳輸給 GPU，但是存取全域記憶體的速度是非常慢的 (400-600 clock)，所以 GPU 提供一個合併機制，藉由減少存取全域記憶體的次數，提升程式的效能。

當 warp 內的執行緒存取全域記憶體的時候，GPU 會將被存取的全域記憶體以 128bytes 的大小做分段，然後 warp 會使用大小為 128bytes 的 L1 快取去存取全域記憶體上的分段，也就是說一次 L1 快取可以讀取一個分段 (128bytes) 的資料。如果當 warp 內的執行緒都存取全域記憶體上同一個分段內的資料，那麼這個 warp 只需要快取一次即可，換句話說這個 warp 只需要存取全域記憶體一次，就可將 128bytes 的資料讀取完成，反之這個 warp 內的執行緒存取全域記憶體上超過一個分段內的資料，就會發生超過一次的快取失敗，如果快取失敗的次數越多，那麼這個 warp 存取全域記憶體的次數就會增加，就會使程序的執行效率大幅下降，所以最好讓 warp 內的執行緒處理同一個

分段的資料。

(5) 常數記憶體

它是屬於唯讀的記憶體，在網格裡面所有的區塊和執行緒都可以去讀取常數記憶體，它是擁有快取的全域記憶體，如果快取命中，其讀取時間只需 4 clock，但是快取失敗，其讀取時間也需要 400-600clock。它的大小不大，通常都是會存入經常使用的資料，以提高快取的命中率。

(6) 材質記憶體

它也是屬於唯讀的記憶體，在網格裡面所有的區塊和執行緒都可以去讀取材質記憶體，它也是擁有快取的全域記憶體，通常是使用在解決電腦圖學領域的問題。

3. 平行化對稱性偵測的演算法

本章將介紹兩個平行化的演算法 – 檢查函數的等效和非等效對稱性與單一變數對稱性並實作在 CUDA 平台上，我們把需要運算的資料儲存在 GPU 的全域記憶體，根據上一章的介紹可得知 GPU 存取全域記憶體的速度是非常慢的，所以我們提出 GSMO 和 GMO 兩種 GPU 記憶體最佳化的技術，以及也分別使用 CENES 和 CSVS 兩種改良方法，使得 GPU 能夠減少讀取全域記憶體的次數，以提升程序的執行效能。在偵測非等效和等效對稱性時，會使用 GSMO 與 GMO 之技術和 CENES 之改良方法，在偵測單一變數對稱性時，會使用 GMO 之技術和 CSVS 之改良方法。在以下小節，我們會介紹這兩個平行化的演算法的流程圖、如何做任務分配、GSMO 和 GMO 兩種記憶體最佳化技術以及 CENES 和 CSVS 兩種改良方法。

3.1 演算法流程圖介紹

(1) 偵測非等效和等效對稱性之流程

整體流程如圖 2 所示，分成主機端和裝置端兩部分作介紹。

一 . 主機端 :

主機端在輸入布林函數 $f(X)$ 的 f^{on} 和 f^{off} 之後，其步驟如下：

Step 1: 預先處理

根據 $f(X)$ 之 f^{on} 和 f^{off} 之積項數目，先配置 GPU 所需的全域記憶體的儲存空間，將積項常用的資訊會存入常數記憶體，最後再將 $f(X)$ 預設成完全對稱性函數，把其對稱情形儲存在大小為 $2 * (N)^2$ 的表格中，並存入 GPU 的全域記憶體。

Step 2: 全域記憶體和共享記憶體最佳化 (*GSMO*)

使用 GSMO 之技術，將 f^{on} 的積項格式重新編排並存入 GPU 的全域記憶體內。

Step 3: 全域記憶體最佳化 (GMO)

使用 GMO 之技術，將 f^{off} 的積項格式重新編排並傳入 GPU 的全域記憶體內。

Step 4: 運行核心 (Launch Kernel)

將 $f(X)$ 的所有的積項配對做任務分配，並配置使用的區塊數量、執行緒數量和共享記體的大小，交付 GPU 執行所有的任務。

二 . 裝置端 :

Step 1: 判斷 $f(X)$ 的對稱性是否已被完全移除

如果 $f(X)$ 的對稱性表格已為空集合，則表示函數已經不存在任何對稱性，區塊內的執行緒就會直接跳出核心函式，反之則會執行以下三個步驟。

Step 2:f^on 積項寫入共享記憶體

將區塊內執行緒所要處理的 f^on 的積項從全域記憶體寫入共享記憶體。

Step 3: 利用積項配對刪除非對稱性配對

針對距離在 2 以內的積項配對找出並刪除不合法輸入的對稱配對。

Step 4: 檢查 $f(X)$ 的對稱表格 (CENES)

區塊內所有的執行緒處理完第三步驟之後，平均分配區塊內的執行緒去檢查部分布林函數 $f(X)$ 的輸入對稱性表格，並記錄函數各個輸入對稱性的狀態。

GPU 在執行時，區塊內的執行緒會執行以上四個步驟，直到所有區塊都執行完成，GPU 才會將最後的結果回傳 CPU。

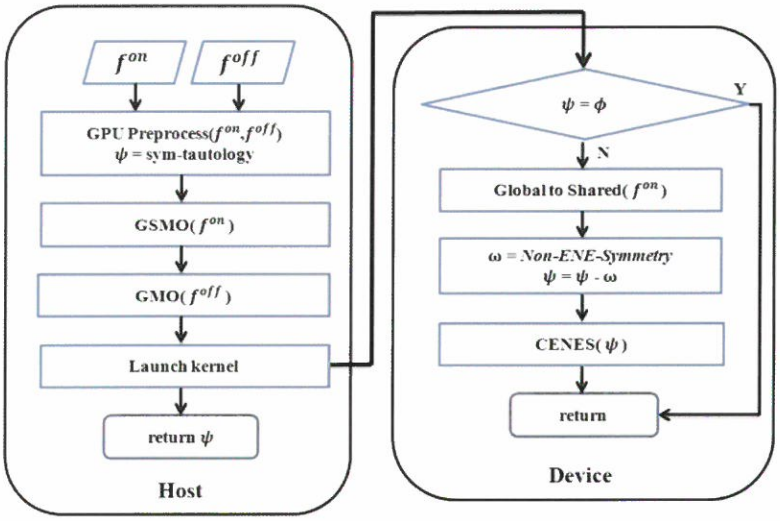


圖 2：偵測函數等效和非等效對稱性之流程圖

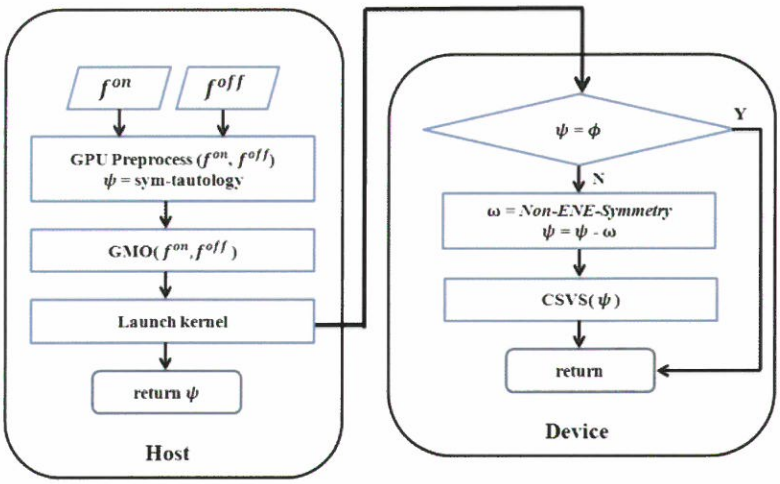


圖 3：偵測函數單一變數對稱性之流程圖

(2) 偵測單一變數對稱性之流程

整體流程如圖 3 所示，分成主機端和裝置端兩部分作介紹。

一 . 主機端 :

主機端在輸入布林函數 $f(X)$ 的 f^{on} 和 f^{off} 之後，其執行步驟與偵測等效和非等效對稱性是大同小異的，主要差別在我們使用 GMO 同時對 f^{on} 和 f^{off} 的積項儲存格式做重新編排並存入 GPU 的全域記憶體內。

二 . 裝置端：

Step 1: 判斷 $f(X)$ 的對稱性是否已被完全移除

如果 $f(X)$ 的對稱表格已為空集合，則表示函數已經不存在任何對稱性，區塊內的執行緒就會直接跳出核心函式，反之則會執行以下兩個步驟。

Step 2: 利用積項配對刪除非對稱性配對

針對距離在 1 的積項配對，刪除所有不合法的單一變數對稱性。

Step 3: 檢查 $f(X)$ 的對稱表格 (CSVS)

區塊內所有的執行緒處理完第二步驟之後，讓區塊內執行緒以 warp 為單位做分配，去檢查部分布林函數 $f(X)$ 對稱性表格，並記錄函數各個輸入對稱性的狀態。

GPU 在執行時，區塊內之執行緒會執行以上三個步驟，直到所有區塊都執行完成，GPU 才會將最後的結果回傳 CPU。

3.2 任務分配

本節探討我們如何對電路所有的積項配對做區塊和執行緒的配置，如圖 4 顯示。給定一個布林函數 $f(X)$，假設其 *on-set* 及 *off-set* 分別有 m 及 n 個積項，總共就會產生 $m \times n$ 個積項配對。我們可將所有的積項配對轉換成一個二維的矩陣，縱軸代表 *on-set* 積項個數，橫軸代表 *off-set* 積項個數。假設一個區塊可處理 p 個 *on-set* 積項和 q 個 *off-set* 積項，就表示一個區塊會

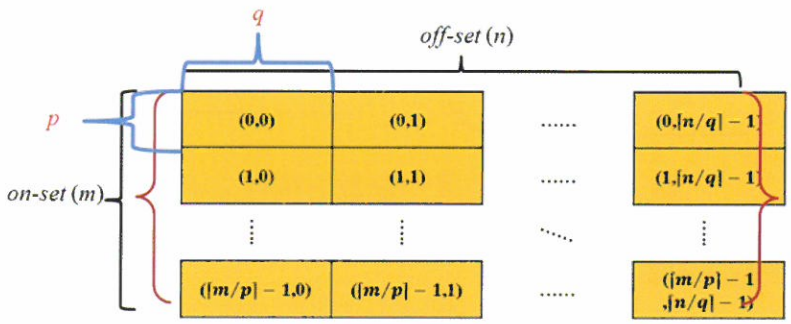


圖 4：GPU 的任務配置圖

有 $p \times q$ 個積項配對。預設一個執行緒可處理一個積項配對，一個區塊將會有 $p \times q$ 個執行緒。使用上述方法分配所有的積項配對，就可以知道每一列會有 $\lceil n/q \rceil$ 個區塊，每一行會有 $\lceil m/q \rceil$ 個區塊。所以總共會有 $\lceil n/q \rceil * \lceil m/q \rceil$ 個區塊。由圖可看出每一列和每一行最後一個區塊都會有多出的部分，這是因為不管如何設定區塊的大小去分配積項配對都很難達到完整分配，所以一定會有多餘執行緒的產生，而且這些執行緒在程式裡並不會被分配到積項配對，就會造成 GPU 的資源浪費。所以在配置區塊的大小時，要盡量減少不必要執行緒的產生。

3.3 全域記憶體與共享記憶體最佳化 (GSMO)

我們在偵測等效和非等效對稱性，會使用到 on-set 積項判斷要移除那些不合法的對稱性。從此可得知 on-set 積項存取的次數是會比 off-set 積項要來的多，所以我們可以在區塊處理積項配對之前，把 on-set 積項儲存至共享記憶體內，以減少 GPU 讀取全域記憶體的次數。在偵測函數的單一變數對稱性，是不需使用 on-set 積項去判斷移除那些不合法的對稱性，所以就不需要將積項存入共享記憶體。

GSMO 的目的就是要提升資料從全域記憶體寫入共享記憶體的效率，而且 GSMO 還有三個優點，第一點就是會減少區塊內閒置執行緒的數量，第二點會使 GPU 減少大量多餘的讀取全域記憶體的次數 (提高 L1 快取的命中率)，第三點就是 GPU 把積項寫入共享記憶體時，不會發生任何的儲存庫衝突。

我們實作 GSMO 的第一個步驟，先在主機端對積項的儲存格式做轉換，第二步驟則是在裝置端將執行緒讀取全域記憶體的方式做改變，那麼我們會先從第二個步驟開始做介紹，這是因為第一個步驟所處理的任務是要改善執行第二個步驟所產生的缺點。假設每個積項需要 32 個 word 的儲存空間，設定每個區塊內有 1024 條執行緒，每個區塊要處理 32 個積項配對，就是每個區塊需要把 32 個 on-set 積項從全域記憶體寫入共享記憶體內，如圖 5 所示，圖中右邊的表格表示 32 個積項所佔用的全域記憶體空間，c0 和 c1 分別表示第零個積項和第一個積項，以下依此類推，總共占用 1024 words 的儲存空間。圖中左邊的表格表示區塊內的執行緒編號，t0 和 t1 分別表示區塊內執行緒都編號 0 和 1，以下依此類推。我們分配每個 warp 內的執行緒共同讀取同一個積項，讓 warp 內的執行緒合作讀取連續的全域記憶體位址，而每一條執行緒只負責讀取一個 word 的資料。這麼做區塊內的執行緒就不會發生閒置，而且讓每個 warp 處理的資料範圍落在 32 個 word 以內，可以使每個 warp 只需讀取全域記憶體一次，就可將 32 個

word 的資料寫入快取中，其他執行緒就不需要作全域記憶體的讀取動作，可大幅提升快取的命中率，所以使用這種方式讀取積項，只需要讀取 32 次全域記憶體，就可將全部大小為 1024 words 的積項存入共享記憶體。在存入共享記憶體的時候，如圖 6 所示，表格外左邊的數字則表示共享記憶體的儲存庫編號。表格外下方的編號為積項編號，如 c0 則代表第 0 個積項，c1 則表示第 1 個積項，以下依此類推。表格內的數字，如果以儲存庫編號 (列) 來看，則表示每個儲存庫負責存取共享記憶體的位址，如果以積項編號 (行) 來看，則表示 32 個積項所儲存的記憶體位址。第零個 warp 內的執行緒會把第 0 個 word 到第 31 個 word 內的資料存入共享記憶體內，第一個 warp 內的執行緒會把第 32 個 word 到第 63 個 word 內的資料存入共享記憶體內，以下依此類推。從上述可得知，這種儲存方式可以讓每個 warp 內的執行緒都使用到不同編號的儲存庫，就可以完全避免儲存庫衝突的產生。

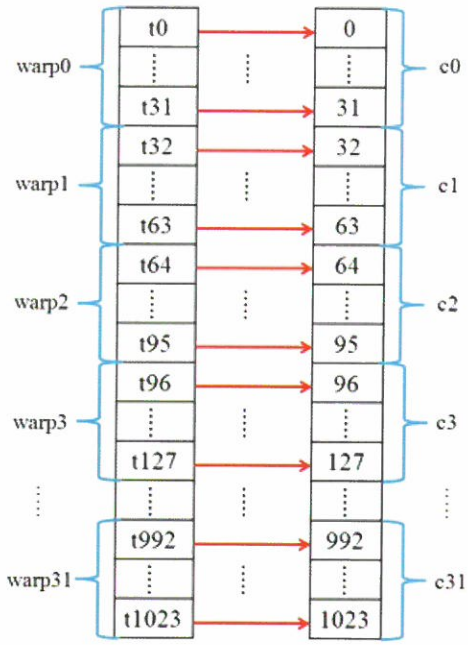


圖 5：裝置端內執行緒讀取積項的模式

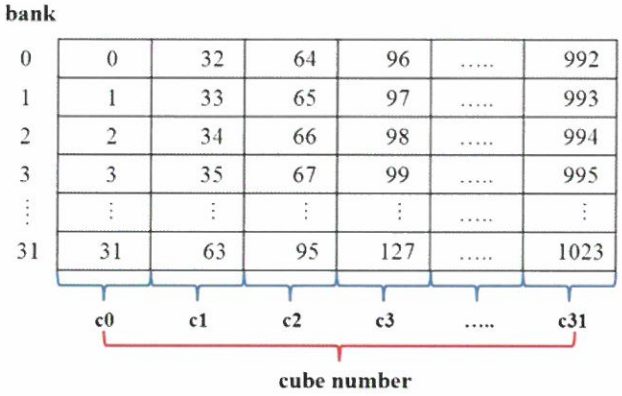


圖 6：儲存庫所對應到的共享記憶體的位置

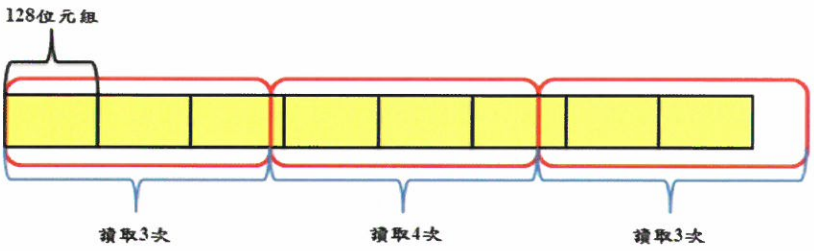


圖 7：各個區塊讀取積項的記憶體次數

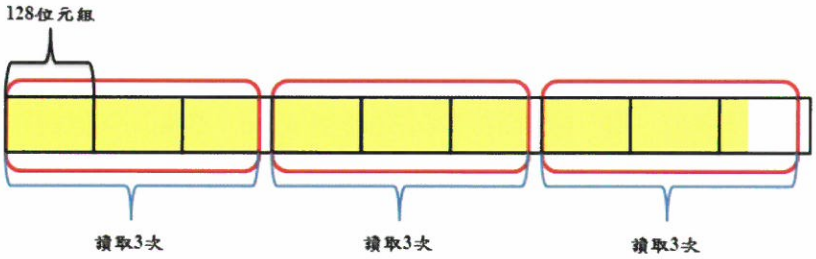


圖 8：改良後各個區塊讀取積項的記憶體次數

利用上述的方法，雖然有效的減少讀取全域記憶體的次數和儲存庫衝突的產生，反而產生一個新的問題。如圖 7 所示，黃色的表格表示函數全部的 on-set 積項所佔用的儲存空間，每一格記憶體的分段為 128 位元組，因為我們使用 warp 讀取全域記憶體的合併機制，所以一格記憶體分段表示一次記憶體的讀取。假設紅色圈住的部分表示

一個區塊處理積項所使用的儲存空間。從圖中清楚看出第二個區塊需要比其他區塊多讀取一次記憶體，因為這個區塊讀取的資料比其它區塊多分佈在 1 個分段上，這是非常不好的情況，因為每一個區塊處理的積項數量是相同的，所以理論上讀取記憶體的次數應該也要相等。為了改善上述的問題，就要在主機端先將積項儲存記憶體的方式做改變，再存入 GPU 的全域記憶體。如圖 8 我們使用空間換時間的方式做改良。將每個區塊處理的第一個積項的起始位址從記憶體的分段上開始儲存。就能保證所有的區塊存取記憶體的次數相等。從圖中可以看出白色的部分是未被使用的記憶體，所以當區塊越多的時候，有可能浪費越多的記憶體空間。

我們提出的 GSMO 之技術可以有效減少所有區塊大量多餘的讀取全域記憶體的次數，而且也可以完全避免儲存共享記憶體時造成的儲存庫衝突。使得讀取全域記憶體和寫入共享記憶體的步驟達到最佳化。

3.4 全域記憶體最佳化 (GMO)

我們在偵測函數對稱性的時候，都會將函數的 on-set 積項和 off-set 積項存入 GPU 的全域記憶體裡，然而 GPU 存取全域記憶體的速度是非常慢的。我們提出 GMO 之技術的目的就是要提升 GPU 讀取全域記憶體的效率，藉由減少 GPU 大量多餘的讀取記憶體的次數 (提高 L1 快取的命中率)，提升程式的效能。

我們實作 GMO 之技術則會先在主機端改變積項的儲存方式，再存入 GPU 的全域記憶體，如圖 9 這是在主機端使用 GMO 對 32 個積項的儲存格式做改變的示意圖，左邊的表格表示積項原本的儲存格式，右邊的表格則表示積項使用 GMO 後的儲存格式。每一格表示記憶體的大小為 4 個位元組，兩邊表格各有 256 個位元組 (64 個 word)。每一個顏色代表一個積項，每一個積項的大小是 2 個 word。從圖中可以發現，原本積項的儲存格式為連續儲存的方式，使用 GMO 轉換後，變成將 32 的積項的第一個 word 依序優先存入 GPU 的全域記憶體，再把第二個 word 依序存入 GPU 的全域記憶體。從圖中可以看出 32 個積項的第一個 word 會存入記憶體的 0 到 31 個 word 上，第二個 word 則會存入記憶體 32 到 63 個 word 上。在裝置端我們使用一個 warp 讀取這 32 個積項，如圖 10，左邊的表格表示一個 warp 裡的 32 條執行緒，我們讓一個執行緒負責處理一個積項，從圖中可以發現當 warp 裡的執行緒要讀取積項的第一個 word 時，因為達到了全域記憶體的合併機制，所以只要存取一次全域記憶體就可完成。現在這個 warp 將 32 個積項全部讀取完成，只需要讀取全域記憶體 2(1*2) 次，如果積項未使用

GMO 做轉換，則需要讀取全域記憶體 4 次才可以完成，可以發現 warp 讀取全域記憶體的次數下降了 1 倍，所以我們提出的 GMO 之技術是能有效下降讀取全域記憶體的次數。

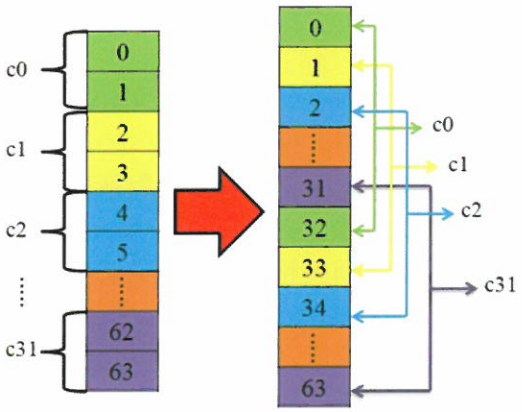


圖 9：積項做 GMO 轉換之示意圖

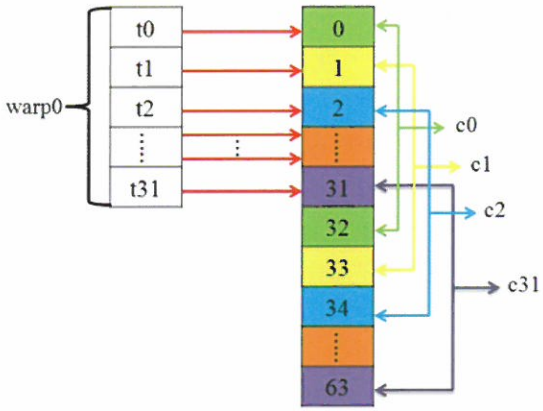


圖 10：warp 讀取積項的模式

3.5 改良函數對稱表格之檢查方法

將積項配對中不合法的對稱性刪除之後，就要檢查函數的對稱表格，如果函數的對稱表格為空集合時，則表示函數的對稱性已經完全被移除，之後尚未被執行的執行

緒，能夠可以在進入核心函式時，就會馬上結束程序。如果將未平行化的方法直接移植在 CUDA 平台上，就會使每一條執行緒處理完積項配對後，都會去檢查函數的對稱表格，這是非常不好的方法。因為我們將函數的對稱表格儲存在全域記憶體裡，如果當函數的輸入越大的時候，函數的對稱性就會越多，檢查函數的對稱表格的時間就會增加，所以我們將檢查函數的對稱表格的方法做改良。

在偵測函數的等效和非等效對稱性，我們使用 CENES 之技術做改良，當區塊內所有執行緒都處理完積項配對的檢查之後，讓區塊內的每個執行緒只檢查部分的函數對稱表格，也就是說每個區塊就只要檢查一次函數對稱性表格即可。假設當函數的輸入的數目為 500，區塊內的執行緒數量為 250，如果每個執行緒都去檢查函數對稱表格，這個區塊就必須作 250 次的函數對稱表格檢查，反之使用 CNENS 之方法，就是區塊內的每個執行緒去檢查函數對稱表格中的不同輸入 (部分表格)，以 500 輸入為例，每個執行緒只需負責檢查兩個輸入，換言之每個區塊只要檢查一次函數對稱表格即可，可大幅下降讀取函數對稱表格之次數。

在偵測函數的單一變數的對稱性，我們使用 CSVS 之技術作改良，因為根據我們的觀察只要積項配對距離是 1，就會存在許多不合法的單一變數對稱性，刪除這些對稱性後，會有很高的機率發生函數大部分的輸入已經不存在任何的對稱性，如果使用 CENES 的方法檢查函數的對稱表格，就會造成大量的執行緒發生閒置，使程式效能大幅下降。假設當函數的輸入的數目和區塊內的執行緒數量各為 320，當任務分配的區塊處理處理完一半的時候，只剩下 100 個輸入含有對稱性，如果使用 CENES 的方法檢查函數的對稱表格，那麼在處理剩下一半區塊的對稱表格之檢查的時候，就會有 220 條執行緒發生閒置，這是非常不好的情況。所以我們針對上述的問題，將 CENES 的方法做改變，不是使用區塊內所有的執行緒平均分配檢查函數各個輸入的對稱表格，而是使用區塊內所有的 warp 平均分配檢查函數各個輸入的對稱表格，這麼做就可以減少執行緒發生閒置的數量。上述的例子區塊內的執行緒數目有 320，總共就會有 10 個 warp，平均分配每個 warp 只檢查對稱表格中的 32 個輸入，就可以有效減少執行緒發生閒置的數量，加快檢查函數各個輸入對稱性的狀態。

4. 實驗結果

4.1 實驗平台與說明

我們已經將偵測函數等效和非等效對稱性與單一變數對稱性的平行化演算法實作完成，實驗所使用的平台為 64 位元架構的主機，CPU 是 INTEL Xeon E5-2620 2.0 GHz 共兩顆，每一顆具有 6 個核心，系統記憶體共有 32GB。顯示卡是 NVIDIA Tesla C2075，它有 448 個核心和 14 個 SM，全域記憶體共有 5GB。作業系統採用 Cent-OS 為 6.4 版本，軟體平台為 *UC Berkeley* 大學所開發的二階邏輯最佳化工具 *Espresso* 套件 [13] 和 NVIDIA 所開發的 CUDA 平台為 5.0 版本。我們使用兩組電路集合，第一組電路為 MCNC *benchmark*，因為此電路集合為二階邏輯電路，其輸入和積項個數都很小，會使得實驗結果無法顯示我們所開發之平行對稱檢查技術的優點。所以我們使用自己開發的工具隨機產生第二組測試電路 : 具有大量輸入和積項數目的大型電路 (輸入數目 >1000)，其實驗結果可清楚顯示平行化技術的確可促進效能提升。

4.2 等效和非等效對稱性之檢查

(1)MCNC benchmark 之實驗結果

第一組實驗是偵測函數的等效和非等效對稱性，使用 MCNC *benchmark*，為了驗證我們提出之平行對稱檢查方法的效果，本實驗有兩組對照組第一組是將原本未平行化的方法在 CPU 上執行 [8]，第二組是將原本的方法平行化並在 GPU 上執行，但不使用 GSMO 和 GMO 兩種技術。在 GPU 上執行時，所設定之區塊大小為 1024 條執行緒，實驗結果如表 1 所示。表中，欄位 #in 和 #out 分別代表電路的輸入和輸出個數，#on 和 #off 分別代表電路的 *on-set* 和 *off-set* 的積項個數，#pair 代表電路中的積項配對數量。symm 代表電路是否存在對稱性，O 表示電路存在對稱性，X 則表示不存在函數對稱性。欄位 CPU、GPU 和 OURS 分別表示第一組、第二組和我們的方法的實驗結果，紀錄時間以秒為單位。以電路 frg2 為例，CPU、GPU 及 OURS 所花費之時間分別為 0.44 秒、0.08 秒及 0.03 秒。以 CPU 為比較基準，GPU 的執行效能可提升 5 倍左右，但 OURS 的執行效能可提升至 14 倍左右。欄位 Total Time 表示所有電路之執行時間時間總和，以 OURS 的執行時間當作基準 (1.0)，相較於 CPU 和 GPU 的方法，OURS 的方法平均效能可分別提升 12.7 倍及 3.3 倍。實驗結果顯示我們的方法在計算對稱性時，比兩組對照組要來的好。因為這些電路都在很短的時間內 (0.5 秒) 執行完畢，無法有效突顯

出我們的方法的實際效益。下一個實驗將使用隨機產生之大量輸入和積項配對數目的電路進行比較。

表 1：MCNC 電路之檢查等效和非等效對稱性之實驗結果

circuit	# in	# out	# of cube		product	time(sec)			CPU/OURS	symm
			on	off		CPU	GPU	OURS		
cordic	23	2	1206	1191	1436346	0.08	0.01	0	-	O
dalu	75	16	2344	1484	3478496	0.17	0.08	0.02	8.50	O
too_large	38	3	1075	541	581575	0.05	0.01	0	-	O
frg2	143	139	4377	3942	17254134	0.44	0.08	0.03	14.67	O
i2	201	1	214	256	54784	0.08	0.04	0	-	O
apex2	39	3	1038	541	559935	0.05	0.01	0	-	O
apex5	117	88	1227	1622	1990194	0.05	0.01	0.01	5.00	O
ex1010	10	10	810	2154	1744740	0.07	0	0	-	X
i8	133	81	3544	986	3494384	0.09	0.04	0.01	9.00	X
i9	88	63	2268	1576	3574368	0.1	0.02	0.01	10.00	X
pdc	16	40	2406	987	2374722	0.03	0.01	0.01	3.00	X
Total						1.27	0.33	0.1		
Ratio (%)						12.7	3.30	1.00		

表 2：大型電路之檢查等效和非等效對稱性之執行時間比較

Method	Input Number		
	500	1000	2000
CPU	389.73	765.00	1420.96
GPU	42.35	63.63	114.51
OURS	7.25	14.37	28.65
CPU / GPU	9.20	12.02	12.41
GPU / OURS	5.84	4.43	4.00
CPU / OURS	53.76	53.24	49.60

(2) 大型的電路之實驗結果

為了驗證我們提出的方法的效果，第二組實驗將會使用工具產生大量輸入和積項的電路進行偵測等效和非等效對稱性的實驗，本實驗使用和第一組實驗相同的兩組對照組。實驗所使用的測試電路的設定如下：有 3 組各包含 100 個電路，其輸入數目分別為 500、1000 和 2000，每個電路的 *on-set* 和 *off-set* 各有 2 萬個積項，其中最少有 5000 個積項配對距離在 2 以內，積項內 don't care 輸入的數量占整體輸入的十分之一。實驗結果如表 2 所示，表中的 CPU、GPU 和 OURS 分別表示第一組、第二組和我們的方法。所紀錄之時間為所有電路的平均執行時間 (以秒為單位)。實驗結果顯示 GPU 的方法在輸入大小為 500、1000 和 2000 的電路，其執行效能比單一 CPU 的執行分別可提升 9.2、12.02 及 12.41 倍。從上述可以得知偵測函數的對稱性的問題是適合平行化的，而我們提出之 GSMO 及 GMO 的技術更進一步將效能提升，結果顯示在 500、1000 及 2000 的輸入大小的電路上，分別可再提升 5.84、4.43 及 4 倍。相對於 CPU 的方式，我們的平行改良技術可提升 53.76、53.24 及 49.6 倍，而且當輸入數目成倍數成長，其執行時間也是線性增長。從實驗結果可以清楚顯示使用 CUDA 平行化加上我們提出的記憶體最佳化之技術是非常有效率的。

4.3 單一變數對稱性之檢查

(1)MCNC benchmark 之實驗結果

第三組實驗是偵測函數單一變數對稱性，使用 MCNC *benchmark*，為了驗證我們提出之平行對稱檢查方法的效果，本實驗有兩組對照組第一組是將原本未平行化的方法在 CPU 上執行 [8]，第二組是將原本的方法平行化並在 GPU 上執行，但不使用 GMO 的技術。在 GPU 上執行時，所設定之區塊大小為 1024 條執行緒，實驗結果如表 3 所示表格中，欄位 #in 和 #out 分別代表電路的輸入和輸出個數，#on 和 #off 分別代表電路的 *on-set* 和 *off-set* 的積項個數，#pair 代表電路中的積項配對數量。symm 代表電路是否存在對稱性，O 表示電路存在對稱性，X 則表示不存在函數對稱性。欄位 CPU、GPU 和 OURS 分別表示第一組、第二組和我們的方法的實驗結果，紀錄之時間以秒為單位。以電路 frg2 為例，CPU、GPU 及 OURS 所花費之時間分別為 0.71 秒、0.16 秒及 0.02 秒。以 CPU 為比較基準，GPU 的執行效能可提升 4 倍左右，但 OURS 的執行效能可提升至 35 倍左右。欄位 Total Time 表示所有電路時間總和，以 OURS 的執行時間當作基準 (1.0)，相較於 CPU 和 GPU 的方法，OURS 的法平均效能可分別提升

43.75 倍及 9.5 倍。實驗結果顯示我們的方法在計算單一變數對稱性時，比兩組對照組要來的好。因為這些電路都在很短的時間內 (1 秒) 執行完畢，無法有效突顯出我們的方法的實際效益。下一個實驗將使用隨機產生之大量輸入和積項配對的電路進行比較。

表 3：MCNC 電路之檢查單一變數對稱性之實驗結果

circuit	# in	# out	# of cube		product	time(sec)			CPU/OURS	symm
			on	off		CPU	GPU	OURS		
cordic	23	2	1206	1191	1436346	0.08	0	0	-	O
dalu	75	16	2344	1484	3478496	0.11	0.08	0.01	11.00	O
too_large	38	3	1075	541	581575	0.06	0.02	0	-	O
frg2	143	139	4377	3942	17254134	0.71	0.16	0.02	35.50	O
i2	201	1	214	256	54784	0.12	0.03	0	-	O
apex2	39	3	1038	541	559935	0.06	0.02	0	-	O
apex5	117	88	1227	1622	1990194	0.09	0.01	0	-	O
ex1010	10	10	810	2154	1744740	0.03	0	0	-	X
i8	133	81	3544	986	3494384	0.14	0.02	0.01	14.00	O
i9	88	63	2268	1576	3574368	0.06	0.01	0	-	O
pdc	16	40	2406	987	2374722	0.12	0.01	0	-	O
Total						1.75	0.38	0.04		
Ratio (%)						43.75	9.50	1.00		

(2) 大型的電路之實驗結果

為了驗證我們提出的方法的效果，第四組實驗將會使用工具產生大量輸入和積項的電路進行偵測單一變數對稱性的實驗，本實驗使用和第三組實驗相同的兩組對照組。實驗所使用的測試電路的設定如下：有 3 組電路集合各包含 100 個電路，其輸入數目分別為 500、1000 和 2000，每個電路的 *on-set* 和 *off-set* 各有 2 萬個積項，其中最少有 5000 個積項配對距離小於 2。每個電路的積項內 don't care 輸入的數量占整體輸入的十分之一。實驗結果如表 4 所示，所紀錄之時間為所有電路的平均執行時間 (以秒為單位)。實驗結果顯示 GPU 的方法在輸入大小為 500、1000 和 2000 的電路，其執行效能比單一 CPU 的執行分別可提升 40.92、44.82 及 48.93 倍。而我們提出之 GMO 的技術進一步將效能提升，結果顯示在 500、1000 及 2000 的輸入大小的電路上，分別可再

提升 2.59、2.17 及 1.9 倍。相對於 CPU 的方式，我們的平行改良技術可提升 106.15、97.38 及 93.03 倍，而且當輸入數目成倍數成長，其執行時間也是線性增長。從實驗結果可以清楚顯示使用 CUDA 平行化加上我們提出的記憶體最佳化之技術是非常有效率的。

表 4：大型電路之檢查單一變數對稱性之執行時間比較

Method	Input Number		
	500	1000	2000
CPU	383.88	686.83	1292.96
GPU	9.38	15.33	26.43
OURS	3.62	7.05	13.90
CPU / GPU	40.92	44.82	48.93
GPU / OURS	2.59	2.17	1.90
CPU / OURS	106.15	97.38	93.03

5. 結論

本本篇論文我們提出 GSMO 和 GMO 兩種技術並使用 CUDA 平行化偵測大型函數的對稱性。GSMO 能夠有效的減少存取全域記憶體的次數，而且在存取共享記憶體時也可以有效地排除儲存庫衝突，使得程式的執行效能能夠提升。GMO 則是讓積項不須存入共享記憶體內，也能夠透過減少存取全域記憶體的次數，提高程式的效能。在執行函數的等效性和非等效對稱性時使用 GSMO 和 GMO 兩種技術，以及在執行單一變數對稱性時使用 GMO 技術，在檢查函數的對稱表格是否為空集合時，我們也將先前研究的方法 [8] 進行改良，並分別提出 CENES 和 CSVS 兩種改良方法。從各種實驗結果得知，我們提出的技術不管在電路的輸入數目的大小和積項數量的多寡，其執行效能都是非常有效的。

目前我們提出的技術是以 C2075 這張顯示卡為主做設計，如果將我們的技術移植在不同計算能力的顯示卡上，就無法確保其技術是最有效果的。因為不同計算能力

的顯示卡其硬體架構和記憶體最佳化的方式都會不一樣，所以就無法保證我們提出的技術是否有效。這個問題是可加以研究討論之未來工作。

參考文獻

1. B.G. Kim and D.L. Dietmeyer, "Multilevel logic synthesis of symmetric switching functions", *IEEE Trans. on Computer-Aided Design*, vol. 10, pp. 436-446, Apr. 1991.
2. Feng Wang and D.L. Dietmeyer, "Exploiting near symmetry in multilevel logic synthesis", *IEEE Trans. on Computer-Aided Design*, vol. 17, pp.772-781, Sept. 1998.
3. V.N. Kravet and K.A. Sakallah, "Constructive library-aware synthesis using symmetries", in *Proc. European Design Test Conf.*, pp. 208-213, 2000.
4. C. Tsai and M. Marek-Sadowska, "Boolean matching using generalized Reed-Muller forms", in *Proc. Design Automation Conf.*, pp. 339-344, June, 1994.
5. F.A. Aloul, A. Ramani, I.L. Markov and K.A. Sakallah, "Solving difficult SAT instances in the presence of symmetry", in *Proc. Design Automation Conf.*, pp.731-736, 2002.
6. C. Scholl, D. Moller, P. Molitor and R. Drechsler, "BDD minimization using symmetries", *IEEE Trans. Computer-Aided-Design*, vol. 18, pp.81-100, Feb. 1999.
7. C.W. Chang, Bo Hu and M. Marek-Sadowska, "In-place delay constrained power optimization using functional symmetries", in *Proc. European Design Test Conf.*, pp. 377-382, 2001.
8. Kuo-Hua Wang and Jia-Hung Chen, "Symmetry Detection for Incompletely Specified Functions", *Design Automation Conf.*, pp. 434-437, 2004.
9. NVIDIA, *NVIDIA CUDA_C_Programming_Guide*_5.0, 2013.
10. NVIDIA, *NVIDIA CUDA_C_Best_Practices_Guide*_5.0, 2012.
11. NVIDIA, *NVIDIA Fermi Compute Architecture Whitepaper*, 2009.
12. CUDA Zone – The resource for CUDA developers, Available at : http://www.nvidia.com/cuda
13. E.M. Sentovich, K.J. Singh, L. Lanvagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P.R. Stephan, R.K. Brayton and A. Sangiovanni-Vincentelli, "SIS: A System for Sequential Circuit Synthesis", *Electronics Research Laboratory*, Memorandum No. UCB/ERL M92/41, 4 May 1992.
14. G. Wang, A. Kuehlmann and A. Sangiovanni-Vincentelli, "Structural detection of symmetries in Boolean functions," in *Proc. Inter. Conf. on Computer Design*, pp. 498-503, 2003.
15. Xiao-Qing Yang and Yu-Liang Wu, "Fast logic restructuring exploring fanout-reconvergent structures and extended symmetry detections," in *Proc. Inter. Conf. on Communications, Circuits and Systems(ICCCAS)*, pp. 1113-1118, 2009.
16. J.S. Zhang, A. Mishchenko, R. Brayton and M. Chrzanowska-Jeske, "Symmetry

detection for large Boolean functions using circuit representation, simulation, and satisfiability," in *Proc. Design Automation Conf.*, pp. 510-515, 2006.
17. N. Kettle and A. King, "An anytime detection algorithm for ROBDDs," in *Proc. Design Automation Conf.*, 2006.
18. Kai-Hui Chang, Lgor L. Markov and V. Bertacco, "Post-Placement Rewiring and Rebuffering by Exhaustive Search for Functional Symmetries," in *Proc. Computer-Aided Design*, pp. 56-63, 2005.
19. A. Mishchenko, "Fast computation of symmetries in Boolean functions," in *Proc. Computer-Aided Design*, pp. 1588-1593, 2003.
20. Z. Feng and P. Li, "Fast Thermal Analysis on GPU for 3D-ICs with Integrated Microchannel Cooling" , *IEEE/ACM Inter. Conf. on Computer-Aided Design*, pp.551-555, Nov. 2010
21. Y. Deng, B.D. Wang and S. Mu, "Taming Irregular EDA Applications on GPUs" , I*EEE/ACM Inter. Conf. on Computer-Aided Design*, pp.539-546, Nov. 2009.
22. M. Garland, "Sparse Matrix Computations on Manycore GPU' s" , in *Proc. of Design Automation Conf. 2008. 45th ACM/IEEE*, pp.2-6, June 2008
23. Damir A. Jamsek, "Designing and Optimizing Compute Kernels on NVIDIA GPUs," in *Proc. Asia and South Pacific Design Automation Conf.*, pp. 224-229, Jan. 2009.
24. B. Suri, U.D. Bordoloi and P. Eles, "A scalable GPU-based approach to accelerate the multiple-choice knapsack problem," in *Proc. Design, Automation & Test in Europe Conf. & Exhibition*, pp. 1126-1129, Mar. 2012.
25. Debapriya C., Andrew D. and Valeria B., "Event-Driven Gate-Level Simulation with GP-GPUs," in *Proc. Design Automation Conf.*, pp. 557-562, 2009.
26. Hao Qian and Yang-Dong Deng, "Accelerating RTL simulation with GPUs," *IEEE/ACM Inter. Conf. on Computer-Aided Design*, pp. 687-693, 2011.
27. Li-Juan Luo, M. Wong and Wen-Mei-Hwu, "An effective GPU implementation of breadth –first search," in *Proc. Design Automation Conf.*, pp.52-55, 2010.
28. Cong, J. and Yi Zou, "Parallel multi-level analytical global placement on graphics processing units," *IEEE/ACM Inter. Conf. on Computer-Aided Design* , pp.681-688, 2009.
29. Yi-Fang Liu and Jiang Hu, "GPU-based parallelization for fast circuit optimization," in *Proc. Design Automation Conf.*, pp.943-946, 2009.
30. S . Rahardja and B.J. Falkowski, "Symmetry conditions of Boolean functions in complex Hadamard transform," *Electronics Letters*, vol. 34, pp. 1634-1635, Aug. 1998.
31. C.C. Tsai and M. Marek-Sadowska, "Generalized Reed-Muller forms as a tool to detect symmetries," *IEEE Trans. Computer*, vol. 45, pp. 33-40, Jan. 1996.
32. B.J. Falkowski and S. Kannurao, "Skew symmetry detection using the Walsh spectral coefficients," in *Proc. Inter. Symposium on Circuit and System*, vol. 2, pp. 321-324, 2000.

Recevied Apr 29, 2014
Revised Apr 29, 2015
Accepted May 6, 2015

Accelerating Functional Symmetry Detection on CUDA Platform

Kuo-Hua Wang and Huan-Hsu Lin

Department of Computer Science and Information Engineering, Fu Jen Catholic University

Abstract

The purpose of this paper is to parallelize our previously proposed functional symmetry detection algorithm and implement it on CUDA (Compute Unified Architecture) platform. We propose two memory access optimization techniques – GSMO (Global and Shared Memory Optimization) and GMO (Global Memory Optimization). Both GSMO and GMO are applied for non-equivalence (NE) symmetry and equivalence (E) symmetry detection, while only GMO is applied to the detection of single-variable (SV) symmetries. By the experimental results, for those Boolean functions with 500 input variables and 40,000 products, our parallel detection algorithm can acquire 50X and 90X performance improvement over the serial one while checking NE&E-symmetries and SV symmetries, respectively. It shows that our parallel symmetry detection technique is indeed effective and efficient for large Boolean functions.

Key words: Symmetry Detection, CUDA, GPU, Parallel Programming.

khwang@csie.fju.edu.tw zerox99@csie.fju.edu.tw

3D 串珠成型之電腦模擬

許健緯　鄭進和

輔仁大學資訊工程學系

摘　要

串珠是一門藝術，只要我們想要的物體，它都可以用手工串織而成。在本論文中研究串珠的作法，用電腦模擬人為串珠規則一步一步串起成型，建構出 3D 串珠成型，在論文中將提出演算法說明如何用電腦模擬每一步驟建立的 3D 串珠多面體，進而成型錐型鏈墜、方型鏈墜、球型鏈墜、迷你雪納瑞與傳統兔子。當中試驗了由三角形、四邊形、五邊形與六邊形所建構而成的 3D 串珠多面體，再由此多面體邊的中點定為每顆串珠位置。最後以電腦圖學技術利用 OpenGL 函式庫將 3D 串珠以多面體與珠體模型成像呈現，經由實驗結果發現手工串織 3D 串珠動作可以由電腦模擬有規律的呈現。

關鍵字：串珠、電腦模擬、電腦圖學。

khwang@csie.fju.edu.tw ssive99@csie.fju.edu.tw

1. 前言

串珠可以串出各種動物，例如:貓、狗、鳳凰或卡通人物，任何人想要的各種物體，都可以透過串的動作去呈現。它是一門藝術，多數人不知串珠為何物，串珠是由具有規則的串珠表 (簡稱規則表)，人拿著材料和釣魚線看著規則表上的步驟一步一步的把它完成形體，其手法類似穿針引線，但卻是更為特殊的手法去做的 [4,6,7,3,8,5]。做串珠靠的都是想像力，一邊看著已串好的實物照片，另一邊跟著串珠規則表做，即使是有經驗的老手，還是會不小心做錯，這時想著有實物參考那多好。

關於 3D 串珠成型電腦模擬之研究 [9,10] 到目前為止並沒有 3D 串珠成型的相關研究，所以本論文之目的在於利用電腦圖學技術，使用電腦模擬人為 3D 串珠成型的方法。我們的方法是以貼面方式來建立 3D 模型的多面體，並沿著串珠規則表的步驟，建立該物體所涵蓋的面，其中利用了球面極座標、貼面公式、座標轉換、特殊部位處理方式等。最後以錐型鏈墜、方型鏈墜、球型鏈墜、迷你雪納瑞、傳統兔子等五種串珠模型做實驗，當中試驗了由三角形、四邊形、五邊形與六邊形所構成之多面體之串珠模型，再由多面體邊的中點定為每顆串珠的位置，並利用電腦圖學技術使用 OpenGL 函式庫繪出點線面所構成之多面體模型、球體模型、及球多面體模型之 3D 串珠 [1,2]。最後，從實驗結果驗證可從串珠規則表有規律轉為電腦 3D 成型模擬。現今 3D 串珠串織步驟之形成顆數皆為 3、4、5 與 6，與本論文研究使用之三角形、四邊形、五邊形與六邊形建構 3D 串珠多面體方式相對應，即本研究可推廣至一般 3D 串珠電腦模擬。

2. 相關研究

串珠成型過程包含三樣元素，第一樣規則表 (如附錄一至五)，第二樣釣魚線，與第三樣珠子材料。因規則表不易看懂，所以先說明規則表，規則表可分為步驟、左線過洞、右線加珠、交叉、形成顆數等五列，分別說明如下：

(1) 步驟

步驟含該回合所需要動作，包含左線過洞含有加珠的動作 (順時針方向去穿珠)，右線加珠則為逆時針方向去穿珠。左線過洞、右線加珠與交叉等動作做完為該步驟之

動作，而該格含有整數的步驟編號外，若該步驟為特殊部位起始時，該步驟會加入其部位文字，如身、耳、腳、尾、鼻等部位特徵。

(2) 左線過洞

當手拿釣魚線時，左手拿的為左線，右手拿的為右線，其過洞意義為穿過旁邊已有的珠子，而左線過洞有四種情況：第一、當該格只含一整數 x，則代表左線過洞 x 顆珠子，或該格含兩整數 x, y，則代表左線過洞 x 顆，右線過洞 y 顆；第二、當該格含有過、跳、或加等三中文字，其為左線過洞、左線跳珠子、或左線加珠子，例如：過 x、跳 y、或加 z (x, y, z 為 3 整數)，則代表左線過洞 x 顆珠子、左線跳 y 顆珠子、或左線加 z 顆珠子；第三、當該格含有加 x 紅 (x 為整數)，則代表左線加 x 顆紅色珠子；第四、該格含加字和 (x, y, z, c) (其中 x, y, z 為整數，c 是顏色)，則代表第 x, y, z 顆是 c 顏色珠子。例如若該格含有過 1 加 4(2, 3, 4 綠)，則代表左線穿過了 1 顆珠子又加 4 顆珠子，而這 4 顆珠子第 1 顆為主色珠子第 2, 3, 4 顆為綠色。最後說明使用小丸之用意，其為小而顏色透明之跳珠，多用於特殊部位，其目的在於固定珠子且不讓人發現珠子，但有一狀況，當左線過洞或加珠數大於右線加珠數時 (如附錄五步驟 15 至 21)，通常開始順時針方向做串珠。

(3) 右線加珠

右線加珠意義為把右線加上珠子，而右線加珠有四種情況：第一、當該格只含一整數 x，代表右線加 x 顆主色珠子；第二、該格含有過、跳、或加等三字，其為右線過洞、右線跳珠子、或右線加珠子，例如：過 x、跳 y、或加 z (x, y, z 為 3 整數)，則代表右線過洞 x 顆珠子、右線跳 y 顆珠子、或右線加 z 顆珠子；第三、該格含 x 紅 (x 為整數)，則代表該格右線加 x 顆紅色珠子；第四、該格含 (x, y, z, c) (其中 x, y, z 為整數，c 是顏色)，則代表第 x, y, z 顆是 c 顏色珠子。

(4) 交叉

左線過洞與右線加珠動作完成之後，左線必須以反方向穿過右線加珠的最後一顆稱為交叉，交叉有三個情況：第一、該格只含一整數，則它會與形成顆數相同；第二、該格為原點，則代表左線與右線必須穿回到該步驟一開始左線與右線的起始位置；第三、該格為跳 x 回 y (x, y 為 2 整數)，則代表有加珠的那條線跳過 x 顆並回穿 y 顆珠子。

(5) 形成顆數

在右線加珠、左線過洞與交叉動作完成後，該步驟總共串織的珠子數目，稱為形成顆數，對本論文之電腦模擬而言，這是重點資訊，該格資訊為構成 3D 串珠模型多面體面之邊數。

在過去有一些電腦串珠成型的研究 [5, 6]，其全以平面 2D 串珠為主，未見有 3D 版串珠成型之電腦模擬，而本論文研究目標為串珠 3D 成型，其做法與 2D 版大為不同。

3. 研究方法

我們的研究方法是先建立 3D 串珠的多面體模型，再取得多面體邊的中點位置，接著將珠子分別以球體與球型多面體表示，將他們的球心置於多面體邊的中點位置，即構成 3D 串珠模型 (如圖 1 與圖 2)，因此我們的研究方法其重點在於如何從規則表建立串珠模型的多面體模型。

圖 1 錐型四面體

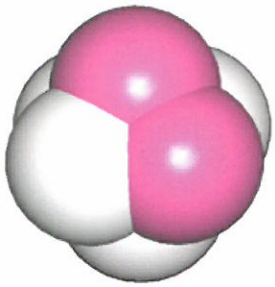


圖 2 錐型球體串珠模型

在這一節中我們將提出演算法說明在串珠建構多面體過程中常見之正三角形、正方形 (或四邊形)、五邊形、與六邊形為主之多面體模型建構，以附錄一至附錄五之規則表為例，說明建構錐型鏈墜、方型鏈墜、球型鏈墜、迷你雪納瑞與傳統兔子等五種 3D 串珠多面體模型的過程，當中會使用到球面極座標、貼面公式、座標轉換與特殊部位處理方式來計算多面體之點線面資料 (其計算方式詳列於第 4 節中)。

在說明附錄一至附錄五規則表之建構多面體步驟之前，有五點特別說明：第一、若該步驟不為特殊處理方式，則其形成顆數上的整數 x，代表建構一正 x 邊形；第二、因已知要建構正多邊形，則我們會先把正多邊形在平面座標系上之標準位置將其頂點

座標與邊的資料儲存起來；第三、在建構串珠多面體模型之座標系，稱之為世界座標系。因此在建構過程中，會常用到座標轉換，從標準位置座標轉為世界座標；第四、通常在規則表中之左線過洞和右線加珠之格子內若是 0，即表示該格不做任何動作；第五、在串織珠子每一步驟，通常都以加珠最後一顆珠子做左線右線交叉，此顆珠子或該珠子對應的多邊形邊作為下一步驟建構新面或新多邊形一個邊的起始。我們建構 3D 串珠多面體的演算法如下：

3D 串珠多面體成型演算法

```
3D 串珠多面體成型演算法
1. input 串珠規則表; //假設 n 是規則表之總步驟數
2. 起始化正三角形、正方形、正五邊形、正六邊形之標準位置(座標);
3. 讀取步驟 1 之形成顆數 x，並將在標準位置之正 x 邊形放入世界座標系中，形成串珠多面體之第一面;
4. i←2; //i 是步驟數
5. while (i≤n) do
6.   if(步驟 i 是處理特殊部位
7.     then 呼叫特殊部位程序(i);
8.     else if(步驟 i 是串珠多面體邊界新的一圈開始
9.           then 建立新一圈的第一面; //第三節中說明
10.          else if((步驟(i-1)不是特殊部位步驟)AND(步驟(i-1).過洞數= = 步驟 i.過洞數+1))
11.           then 補上該圈最後一面; //新一圈的結尾
12.           else 利用步驟 i 之形成顆數 x，將標準位置中之正 x 邊形放入世界座標系中，成為多面體之一面;
13.     //end if
14.  i←i+1;
15. //end while
```

特殊部位演算法

```
特殊部位程序(i)
//步驟 i 是所要處理之特殊部位的開始步驟
1. switch(特殊部位){
2.   case “迷你雪納瑞腳“、“迷你雪納瑞尾“、“兔子尾“:
3.     利用球面極座標找到該部位之世界座標; break; //在第四節中說明
4.   case “兔子前腳“、“兔子後腳“、“兔子耳朵“:
5.     特別的處理方式計算該部位之世界座標; break; //在第四節中說明
6.}
```

在本節中接著分別說明錐型鏈墜、方型鏈墜、球型鏈墜、迷你雪納瑞與傳統兔子之 3D 多面體建構演算法過程 (為簡化說明起見，在以下之圖表中以△、□、梯、㊄與㊅分別代表三角形、正方形、梯形、五邊形與六邊形)。

3.1 錐型鏈墜方法

在附錄一的錐型鏈墜規則表中有 4 個步驟 (實體圖如圖 3)，其步驟 1 右線加 3 顆珠子，形成顆數為 3，亦即把標準位置之正三角形放入世界座標系中，如圖 4，我們得到正三角形 ABC；步驟 2 右線加 2 顆珠子，形成顆數 3，得知要放一正三角形 ADC 與前一步驟所產生之正三角形 ABC 共一邊 (AC)，但此時這兩個正三角形之夾角未知，即步驟 2 產生之正三角形其第 3 點 (如圖 4 中之 D 點) 座標無法算出，因此繼續往下一步驟看；在步驟 3 左線過洞 1 顆珠子，右線加珠 1 顆珠子，形成顆數為 3，即產生一正三角形 DBC 分別與步驟 2 之正三角形 ADC 和步驟 1 之正三角形 ABC 之共邊 CD 與 BC，此時正三角形 ABC 與正三角形 ADC 和正三角形 DBC 之夾角已經出現，我們利用球面極座標算出 D 點座標 (運用 4.1 節中由球面極座標求座標公式算出 D 點的標準座標，再利用 4.3 節公式得到 D 點之世界座標)；在步驟 4 左線過洞 3，把步驟 3、步驟 2 與步驟 1 產生之正三角形沒有相連之邊連成第 4 個正三角形 BDA。最後，對錐型鏈墜規則表共產生 4 個點、六個邊與 4 個正三角形之多面體。

圖 3 錐型鏈墜實體圖

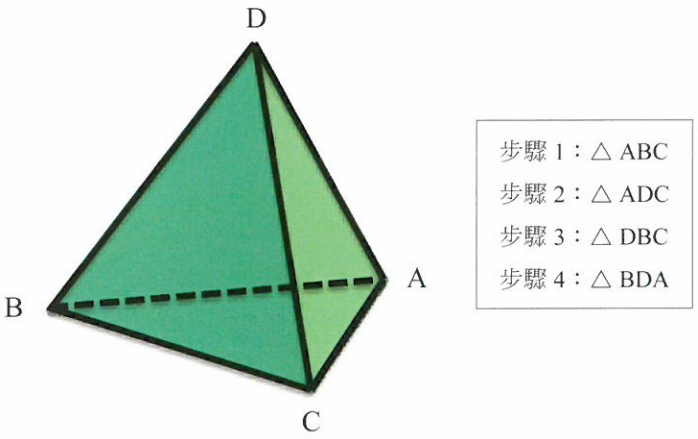


圖 4 錐型鏈墜多面體

3.2 方型與球型鏈墜方法

在附錄二的方型鏈墜規則表中有 6 個步驟 (實體圖如圖 5)，在步驟 1 首先建立一正方形 ABCD(如圖 6)，接著步驟 2 再建立一正方形 ABFE 與前一步驟產生之正方形 ABCD 共一邊 (AB 邊)，但正方形 ABFE 目前與正方形 ABCD 之夾角未定，先往下一步驟 (步驟 3) 查看，它需建立一正方形 ACHE 與前一步驟 (步驟 2) 共一邊 (AE 邊) 此時已知 A、B、C、D 4 點座標，但 E 點 (步驟 2 與步驟 3 產生正方形的共邊上之一頂

點) 座標未知，可是知道 AE 邊與正方形 ABCD(步驟 1 所產生的面) 夾角 90 度，因此可以利用極座標算出 E 點座標 (先用 4.1 節公式由極座標算出 E 點的標準座標 , 再用 4.3 節公式將其轉為世界座標)。有了 E 點座標可以建立一個直角正交局部座標系 UVW(以 A 點為原點，向量 $\overrightarrow{AB}$ 為正 U 軸方向，向量 $\overrightarrow{AB}$ × 向量 $\overrightarrow{AC}$ 方向為正 W 軸方向，W 軸方向 × 正 U 軸方向為正 V 軸方向) ，算出步驟 2 所產生之正方形 ABFE 上之點 F 座標 (運用 4.3 節公式將 UVW 座標系之 F 點座標轉為世界座標)，用同樣方式可以算出步驟 3 所產生之正方形 ACHE 上之未知點 H 座標。同理步驟 4 與步驟 5 分別產生正方形 CDGH 與正方形 DGFB 之未知點 G 的座標可以求出。最後，在步驟 6 中把步驟 2 至步驟 5 未相連之邊合為一正方形 EHGF。在整個方型鏈墜多面體建構過程中，共產生 8 個點、12 個邊與 6 個正方形。

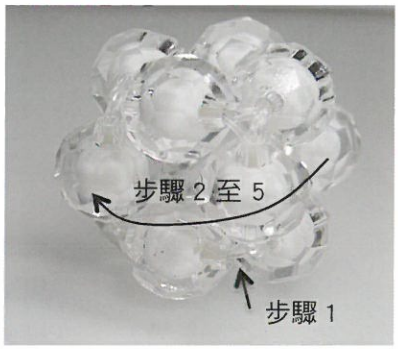
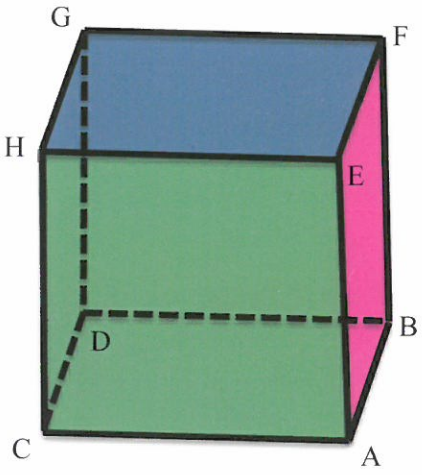


圖 5：方型鏈墜實體圖



步驟 1：△ ABCD
步驟 2：△ ABFE
步驟 3：△ ACHE
步驟 4：△ CDGH
步驟 5：△ DGFB
步驟 6：△ EHGF

圖 6 方型鏈墜多面體

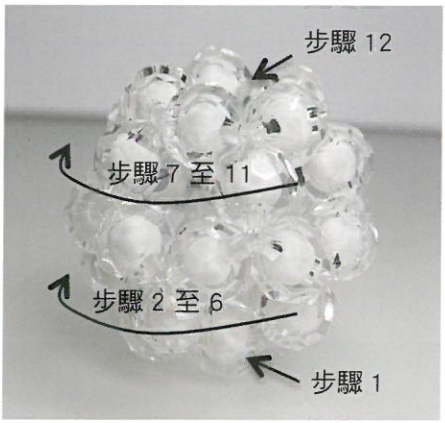
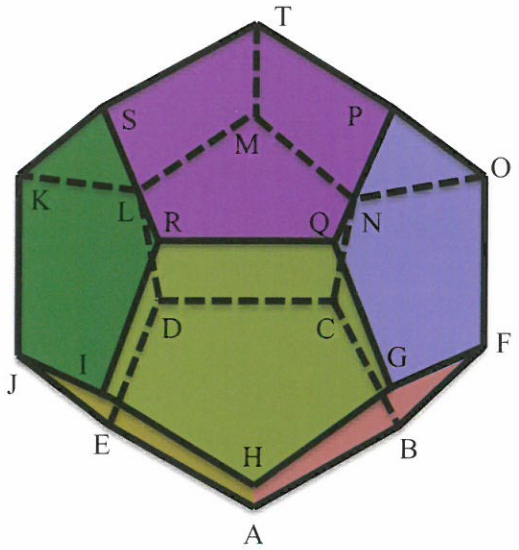


圖 7 球型鏈墜實體圖

在附錄三的球型鏈墜規則表中有 12 個步驟 (實體圖如圖 7)，其建構多面體過程類似前面所提之方型鏈墜。步驟 1 先在底部 (世界座標系 XY 平面上) 放一個正五邊形，接著步驟 2 至步驟 6 產生 5 個相連接之正五邊形分別與底部之正五邊形之上圍成一圈的多面體邊界，然後在步驟 7 至步驟 11 產生 5 個正五邊形放在第一圈的邊界之上，形成第 2 圈多面體邊界，最後在步驟 12 把頂部之五頂點產生一正五邊形。在球型鏈墜多面體產生過程中，其中第 1 圈邊界與第 2 圈邊界在求點座標的方法與附錄二之方型鏈墜多面體找法相同，故不再描述。最後，由球型鏈墜規則表產生之多面體含有 20 個點、30 個邊與 12 個正五邊形，其建構過程如圖 8 所示。



步驟 1：㊄ ABCDE
步驟 2：㊄ ABFGH
步驟 3：㊄ AHIJE
步驟 4：㊄ EJKLD
步驟 5：㊄ DLMNC
步驟 6：㊄ CNOFB
步驟 7：㊄ OPQGF
步驟 8：㊄ GQRIH
步驟 9：㊄ IRSKJ
步驟 10：㊄ KSTML
步驟 11：㊄ MTPON
步驟 12：㊄ TSRQP

圖 8 球型鏈墜多面體

3.3 迷你雪納瑞方法

在附錄四的迷你雪納瑞規則表中有25個步驟，迷你雪納瑞的大概流程如圖9所示。迷你雪納瑞步驟 1 是由臀部開始做。在身體步驟 1 至 13 中逆時針做兩圈 (從臀部往頭的方向看)，皆為正方形組成，其身體與頭只有一邊相連，在頭部步驟 14 至 20 中，為正方形與正五邊形的結合，在嘴部之步驟 21 至 25 做法，則與方型鏈墜相同，最後步驟 25 結束的地方在嘴部。

圖 9 迷你雪納瑞實體圖

現在詳細說明整體步驟流程，在步驟身 1(身體部位) 建立一個正方形，步驟尾 2(尾巴部位) 右線加珠 3 顆與步驟 1 最後一顆加珠共 4 顆，也是建立一個正方形，則身體部位與尾巴部位共用一個邊 (如圖 10)，此時兩正方形夾角未知，我們利用極座標與座標系間座標轉換方式 (套用 4.1 節與 4.3 節中公式) 給定一個角度算出尾巴部位世界座標，該尾巴步驟交叉回原點，即左線又線回到剛才兩正方形共邊處。

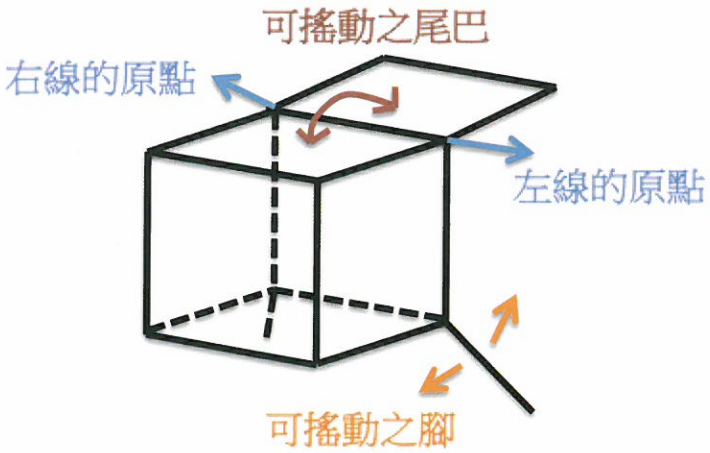
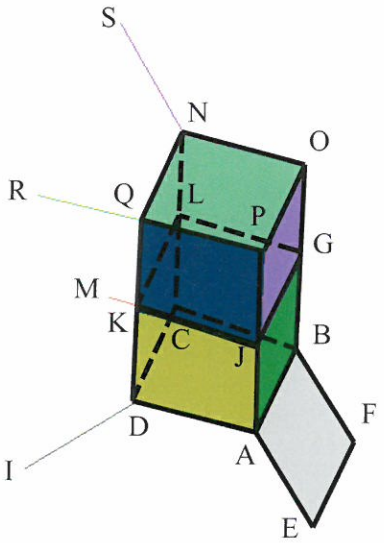


圖 10 迷你雪納瑞尾與腳做法

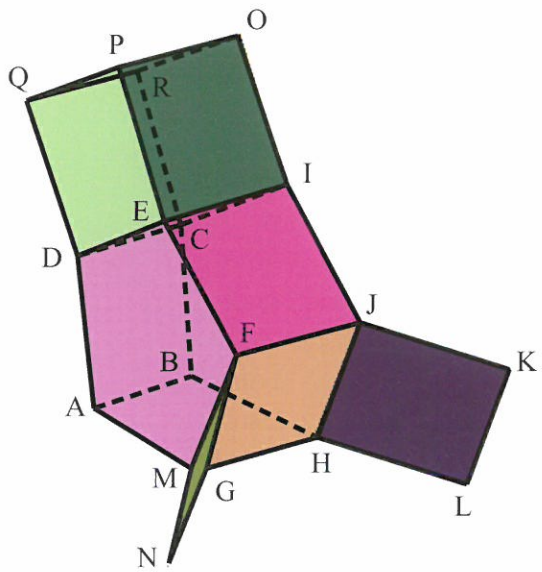
步驟 3 加珠 3 顆，因前一步驟交叉回原點，與步驟 1 和步驟 2 共用一邊，此時身體只有兩面不能固定，需要下一面才能決定。在步驟腳 4 左線過洞 1 加珠 2 顆，第 1 顆是白色珠子而第 2 顆是小丸，其後跳過小丸回穿白色珠子，這時形成了一隻腳，其腳為不固定的一條線段與身體部位共一點 (如圖 10)，我們給定角度，利用局部座標系與世界座標系間之轉換 (套用 4.3 節公式)，算出這條線段之頂點 (世界) 座標。由於小丸很小，因此此腳小丸視為一點，其位置設為腳線段之一頂點位置。



步驟 1：□ ABCD
步驟 2：□ AEFB
步驟 3：□ BGJA
步驟 4：邊 DI
步驟 5：□ JKDA
步驟 6：□ KLCD
步驟 7：邊 CM
步驟 8：□ LGBC
步驟 9：□ LNOG
步驟 10：□ OPJG
步驟 11：□ PQKJ
步驟 12：□ QNLM
步驟 13：邊 QR 、邊 NS

圖 11 迷你雪納瑞身體步驟

步驟 5 右線加珠 2 顆，其一邊 (一顆珠子) 為步驟 1 之正方形的一邊 (步驟 4 的過洞)，與一邊為步驟 4 過洞的邊 (交叉的珠子)，其形成顆數 4，意即將一標準位置之正方形使用貼面公式 (套用 4.2 節公式) 算出世界座標系之正方形。步驟 6 與方型鏈墜相同，使用貼面公式貼上正方形。而步驟腳 7 其方法與步驟 4 相同做法。步驟 8 加珠 1 顆，形成顆數 4，與步驟 1、步驟 3 與步驟 7 中的共邊形成正方形，完成第一圈邊界面。在步驟 9，加 3 紅珠，與前一步驟 8 所生之正方形的一邊相連，其形成顆數 4 為正方形，開始做新的一圈第一面，其面目前為不固定，故先看下一步驟。步驟 10 左線過洞 1，右線加珠 2，形成顆數 4，為一正方形，其左線過洞 1 所產生的正方形與步驟 3 所產生的正方形的邊相連，又與步驟 9 所產生的正方形相連，因此可用球面極座標求出步驟 9 與步驟 10 產生正方形之座標 (套用 4.1 節與 4.3 節公式求得)。步驟 11 至 13 做法相同，使用貼面公式貼上正方形，從步驟 1 至 13 之兩圈多面體建構過程如圖 11 所示。步驟頭 14 左線過 2，右線過 2 加 3，這步驟是先綁緊邊，使之不會太鬆，所以過洞多，而加 3 顆珠子為做頭的開始，又與身體部位共連著一邊，形成顆數 4，為一正方形。因步驟 14 開始是頭部，又身體與頭部位之邊共連一個邊，如果直接接上去會有角度問題，可能導致頭進入了身體，故在步驟 14 開始，在一局部座標系中建構頭部多面體，再給予一角度，轉換成世界座標與身體部位相連即可完成 (套用 4.3 節公式求得)，步驟 14 至 20 做法順序如圖 12。



步驟 14：□ ABCD
步驟 15：㊄ DEFGA
步驟 16：□ GHBA
步驟 17：㊄ HBCIJ
步驟 18：□ FMNG、□ JKLH
步驟 19：□ KLCD
步驟 20：□ JIEF
步驟 21：□ IOPE
步驟 22：□ EPQD
步驟 23：□ DQRC
步驟 24：□ CROI
步驟 25：□ OPQR

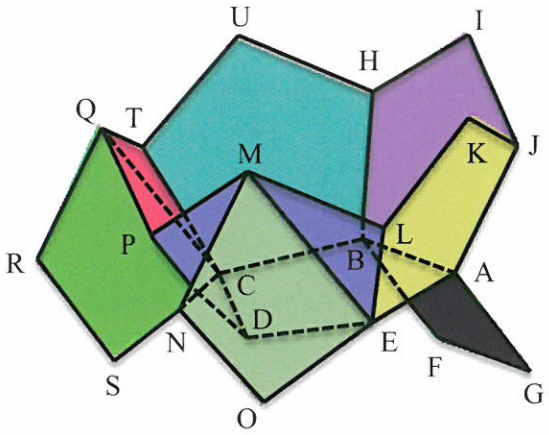
圖 12 迷你雪納瑞頭部 (邊 AB 與圖 11 之身體部位邊 PO 相黏)

在頭部多面體計算過程中，步驟 14 至 16 使用球型極座標公式算出世界座標 (套用 4.1 節與 4.3 節公式求得)，再使用貼面公式 (套用 4.2 節公式) 即算出步驟 15 至 16 之正方形，步驟 17 至 20 使用貼面公式 (套用 4.2 節公式)。最後，步驟 21 至 25 為方型鏈墜之應用。

3.4 傳統兔子方法

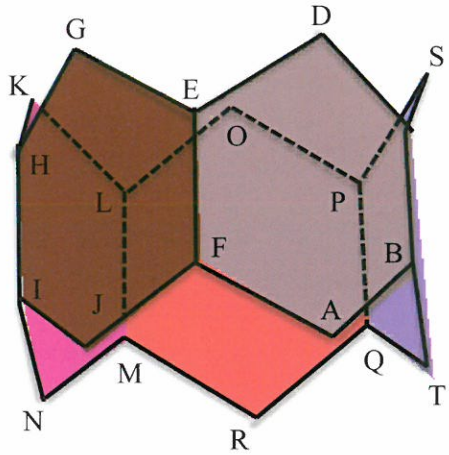
在附錄五的傳統兔子規則表中有 35 個步驟，其做法都與迷你雪納瑞相似，但其中特殊部位之多邊形資訊產生會加以說明。先說明傳統兔子的大概流程 (如圖 13)。傳統兔子在步驟 1 是由臀部開始做，在身體步驟 1 至 9 中，以逆時針方向 (從臀部往頭方向來看) 做一圈，皆為五邊形所組成 (如圖 14)；在身體步驟 10 至 14 中，以逆時針方向再做一圈，皆為六邊形組成 (如圖 15)；在身體步驟 15 至 22 中，以順時鐘方向做一圈，皆為五邊形組成 (如圖 16)；在頭部步驟 23 至 25 中，頭部為步驟 22 上的五邊形去建立球型鏈墜；步驟 35 結束的地方為頭頂。

圖 13 傳統兔子實體圖



步驟 1：㊄ ABCDE
步驟 2：□ ABFE
步驟 3：㊄ BHIJA
步驟 4：㊄ JKLEA
步驟 5：㊄ LMPDE
步驟 6：梯 MNOE
步驟 7：㊄ PQTCD
步驟 8：梯 QRSC
步驟 9：㊄ TUHBC

圖 14 傳統兔子下身



步驟 10：㊅ ABCDE
步驟 11：㊅ EGHIJF
步驟 12：㊅ HKLMN
步驟 13：㊅ LOPQRM
步驟 14：㊅ PSCBTQ

圖 15 傳統兔子中身

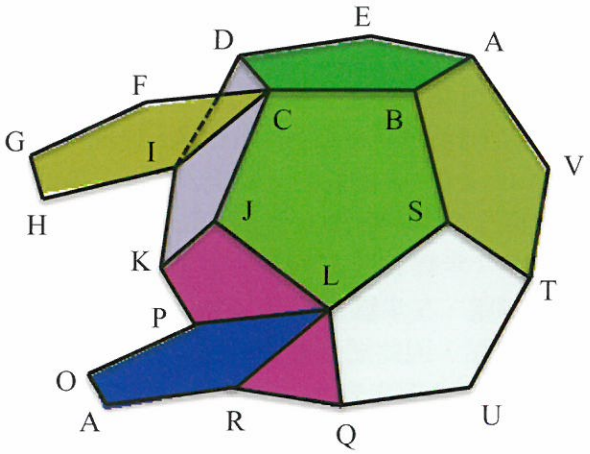


圖 16 傳統兔子上身

接著說明步驟細節，首先就兔子腳部位手工串珠製作過程 (如圖 17)，為了減少線穿過珠子的次數，它是腳與身體一起做的，如步驟 5 右線加珠 4 顆，形成顆數 5 顆，其為五邊形 OABCD，與步驟 6 左線過洞 1(邊長 OF)，右線過洞 2(邊長 DC、CB)，右線加珠 2，形成顆數 6，其為四邊形 OBCD 與四邊形 OBEF，其串法是這樣，但就電腦模擬而言就不易以手工成型順序完成。在本論文中我們是先建立身體之五邊形 OABEF，然後再建立腳四邊形 OBCD。這當中需要平移、旋轉腳四邊形之運算 (詳見 4.4.1 節兔子前腳與後腳特殊部位的處理說明)，算出相關點線面資料；步驟 7 至 8 做腳的方式與步驟 5 至 6 是相反方向去做的，所以不詳加說明；在步驟 9 則貼上五邊形即可完成。

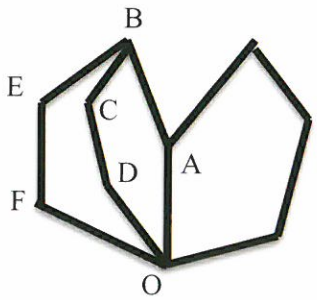


圖 17 傳統兔子後腳

在步驟 10 至 14 中每一步驟形成顆數皆為 6，則開始以六邊形構成邊界為主。這當中涉及到一個問題，它是相鄰 2 個正五邊形上方要架上一個正六邊形，這會造成一個空隙，這正六邊形的 2 個邊與 2 個相鄰的正五邊形無法相接，意即為達成此項工作，無法維持原來正五邊形或是正六邊形的形狀，此為特殊部位點之計算 (詳細計算方式請見 4.4.3 節兔子身體特殊部位的處理說明)；步驟 15 做新的一圈邊界開始為順時針方向進行；步驟 16 至 17 與步驟 18 至 19 做的方式與步驟腳 5 至 6 相似，不多做說明；步驟 20 至 22 皆為正五邊形貼面方式形成邊界；步驟 23 開始做的方式與球型鏈墜相似，是以步驟 22 所建之五邊形為底來做出球型鏈墜，在步驟 23，形成顆數 6，第 3 顆為小粉橘 (兔子鼻子簡稱小粉)，該顆小粉視為一點，因此把形成顆數降 1 改為 5，這步驟 23 建立一正五邊形，在步驟 23 至 30 除了步驟 28 與步驟 29 以外，其餘都和球型鏈墜做法相同；在步驟耳 31 與步驟耳 33 是分別建構特別之 11 多邊形狀的耳朵，因此我們先在平面上先建立好樣版形狀之 11 多邊形狀 (其點座標先算出)，接著運用平移旋轉之座標轉換算出接上頭部之後的真正世界座標位置 (詳見 4.4.2 節兔子耳朵特殊部位的處理說明)。

綜合上述結果，由電腦模擬串珠規則表，以建構面方式一一建立成 3D 多面體，雖然當中有特殊部位 (如腳部、耳朵、與身體一部份) 之建立，以手工串珠該部位之過程由電腦模擬不易，但可以用面或線段直接連上身體卻非常方便，因此電腦模擬串珠規則表建構 3D 串珠模型是有相當的規律性，在本論文中為特殊部位處理方式可應用在其它的串珠動物特殊部位。

以上算出來的模型點線面資料會以 2 種型式存檔：第一種型式檔案以存每一多邊形面或搖擺邊 (如迷你雪納瑞腿) 為主，最後以 3D 多面體型式呈現 3D 模型；第二種型式檔案為呈現球型與球多面體型式出現，因而這種檔案以存球心座標與顏色為主，而球心座標來自原先多面體上每個邊或搖擺邊之中點座標。最後利用 OpenGL 函式庫將模型之多面體，球型與球型多面體織串珠 3D 模型繪出呈現。

4. 相關數學公式與特殊部位處理

4.1 利用球面極座標求座標

已知有一球 (球心在原點 O，半徑 r)，與球面上 2 點 A 與 B。A 點在 x 軸上，B 點在 xy 平面上，$\overrightarrow{OA}$ 與 $\overrightarrow{OB}$ 夾 θ 角，求在球面上一點 C 座標 (x, y, z)，滿足 $\overrightarrow{OC}$ 與 $\overrightarrow{OB}$ 夾 β 角，而且 $\overrightarrow{OC}$ 與 $\overrightarrow{OA}$ 夾 α 角 (如圖 18 所示)。已知 A 點座標 (r, 0, 0)，與 B 點座標 (r cos θ, r sin θ, 0)，而 C 點座標為 (r sin ϕ cos Φ , rsin ϕ sin Φ, r cos ϕ)(其中 ϕ 為 $\overrightarrow{OC}$ 與 + z 軸之夾角，Φ 為 $\overrightarrow{OC}$ 向量投影在 xy 面上與正 x 軸之夾角)，與兩向量的內積

$\overrightarrow{OA} \cdot \overrightarrow{OC} = r^2 \cos\alpha$

$\overrightarrow{OB} \cdot \overrightarrow{OC} = r^2 \cos\beta$

代入得知兩式

$(r,0,0) \cdot (r \sin\phi \cos\Phi , r \sin\phi \sin\Phi , r \cos\phi) = r^2 \cos\alpha$ .....(1)

$(r \cos\theta , r \sin\theta , 0) \cdot (r \sin\phi \cos\Phi , r \sin\phi \sin\Phi , r \cos\phi) = r^2 \cos\beta$ .....(2)

(1) 式整理得到

$\sin\phi \cos\Phi = \cos\alpha$ .....(3)

帶入 (2) 式得到

$\sin\phi \cos\Phi = (\cos\beta - \cos\theta * \cos\alpha) / \sin\theta$ .....(4)

已知

$\cos\Phi = \pm\sqrt{1-\sin^2\Phi}$ 帶入 (3) 式得到

$$\sin\phi\left(\pm\sqrt{1-\sin^2\Phi}\right) = \cos\alpha$$

等號兩邊平方後整理得到

$\sin^2\phi = \sin^2\phi \sin^2\Phi + \cos^2\alpha$

因為

$\cos\phi = \pm\sqrt{1-\sin^2\phi}$

所以

$$\cos\phi = \pm\sqrt{1-(\sin^2\phi \sin^2\Phi + \cos^2\alpha)}$$

$$= \pm\sqrt{(\sin^2\alpha - (\cos\beta - \cos\theta * \cos\alpha)^2 / \sin^2\theta}$$ .....(5)

由 (3) 式、(4) 式與 (5) 式得到 C 點座標 (x,y,z) 為

$$x = r \cos\alpha$$

$$y = r * (\cos\beta - \cos\theta * \cos\alpha) / \sin\theta$$

$$z = r * \left(\pm\sqrt{(\sin^2\alpha - (\cos\beta - \cos\theta * \cos\alpha)^2 / \sin^2\theta}\right)$$

在本論文中藉用已知3點原點O點、A點、B點、與3個角度α、β、θ算出C點座標。在建構錐型鏈墜時，使用 $\alpha = 60^\circ$, $\beta = 60^\circ$, $\theta = 60^\circ$，在建構方型鏈墜時，使用 $\alpha = 90^\circ$, $\beta = 90^\circ$, $\theta = 90^\circ$，在建構球型鏈墜時，使用 $\alpha = 108^\circ$, $\beta = 108^\circ$, $\theta = 108^\circ$，在建構迷你雪納瑞頭部時，使用 $\alpha = 90^\circ$, $\beta = 108^\circ$, $\theta = 90^\circ$，在本論文中所找之 C 點其對應之極角 ϕ 皆小於等於 90°，因此 C 點之 Z 座標採用正值。

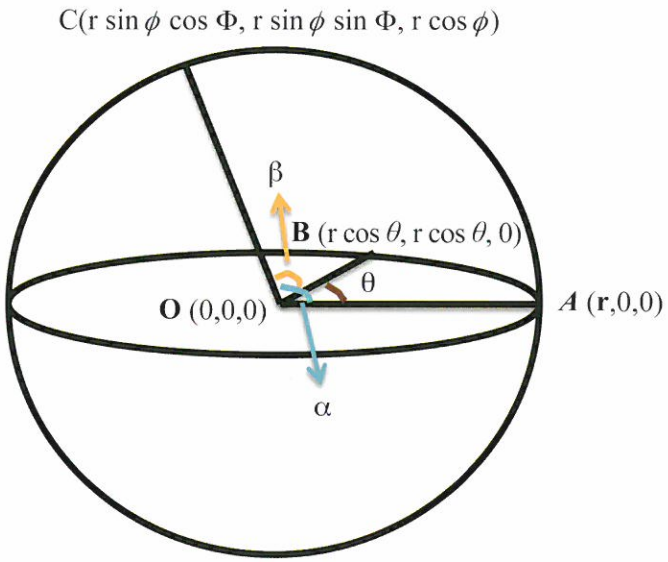


圖 18 球面極座標圖

4.2 貼面公式

使用於 3D 中已有 2 個相同且相鄰正多邊形，如圖 19(1) 之面 2、與面 3，要再貼上一同樣之正多邊形於世界座標系中之面 1 位置。而欲貼上於 2D 平面之標準位置 (於局部座標系) 上之正多邊形，如圖 19(2) 所示標準位置之正多邊形之一頂點位於局部座標系原點 O(0, 0)，已知其相鄰之 A、B 兩點之局部座標。現今之目的是將標準位置上之正多邊形貼上面 1 位置，使得標準位置上之 O 點、A 點、與 B 點分別對準圖 19(1) 中之 O' 點、A' 點、B' 點，因此必須算出在標準位置上其它點 (O, A, B 點外之點) 再貼上 3D 世界座標系後之座標，例如局部座標系標準位置上之點 P(x, y) 對到圖 19(1) 上之 P' 點座標。為此我們另定一個 3 維座標系 UVW，以 O' 點為原點，向量 $\overrightarrow{O'A'}$ 之方向為 +U 方向 (單位向量設為 $\vec{u}$)，而 +W 軸方向定為 $\overrightarrow{O'A'} \times \overrightarrow{O'B'}$ 之方向 (單位向量設為 $\vec{w}$)，因此 +V 軸方向定為 $\vec{w} \times \vec{u}$ (假設單位向量為 $\vec{v}$)，我們知道向量 $\overrightarrow{O'P'} = x\,\vec{u} + y\,\vec{v}$，而且 $\overrightarrow{O'P'} = P' - O'$，所以 $P' = O' + x\,\vec{u} + y\,\vec{v}$，$P'$ 點座標就此得到。同理將整個正多邊形在標準位置上貼到 3D 世界座標系面 1 位置 (在圖 19(1) 中) 的其它點座標皆可得到。

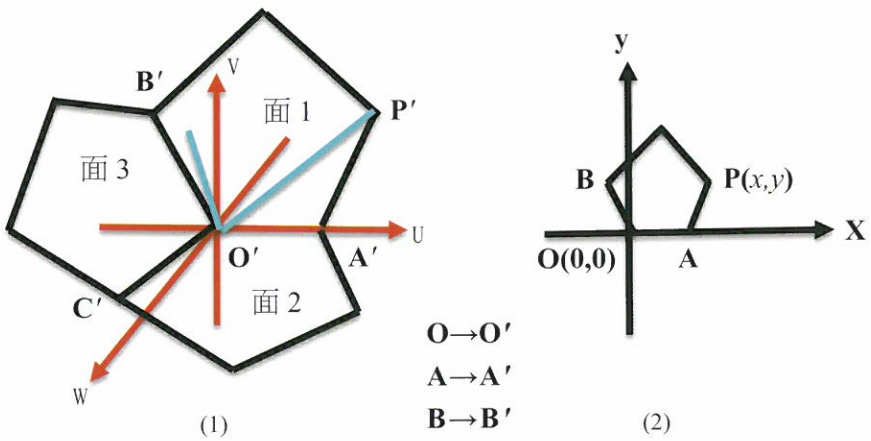


圖 19 正多邊形面貼上世界座標系 (1) 世界座標系中之相關位置
(2) 正多邊形於局部座標系之標準位置

不過，在建構面的時候要注意兩件事情，第一件，向量的方位要對準，如沒對準，有可能會朝反方向算；第二件，貼面的同時，要注意點的順序，如果順序不對，將會出現點位移的現象。

4.3 不同座標系間之座標轉換

給定不共線 3 點 A、B、C 的 XYZ 世界座標，定義一個 UVW 直角正交局部座標系統。其中以 A 為原點，向量 $\overrightarrow{A'B'}$ 方向為正 U 軸方向 (假設其單位向量為 $\vec{u}$)，A、B、C 3 點所在的平面為 UV 平面，而正 W 軸方向為 $\overrightarrow{A'B'} \times \overrightarrow{A'C'}$ (假設其單位向量為 $\vec{w}$)，所以正 V 軸上之單位向量為 $\vec{w} \times \vec{u}$ (假設其單位向量為 $\vec{v}$)。空間中有一點 D，其 UVW 座標為 (*du, dv, dw*) 轉為世界座標 (*dx, dy, dz*)，這兩種座標間轉換如下：

$$\begin{bmatrix} dx \\ dy \\ dz \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & ax \\ 0 & 1 & 0 & ay \\ 0 & 0 & 1 & az \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} ux & vx & wx & 0 \\ uy & vy & wy & 0 \\ uz & vz & wz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} du \\ dv \\ dw \\ 1 \end{bmatrix}$$

其中 A 點之世界座標為 (*ax, ay, az*)，向量 $\vec{u}$ = (*ux, uy , uz*)，向量 $\vec{v}$ = (*vx, vy, vz*)，向量 $\vec{w}$ = (*wx, wy, wz*)。

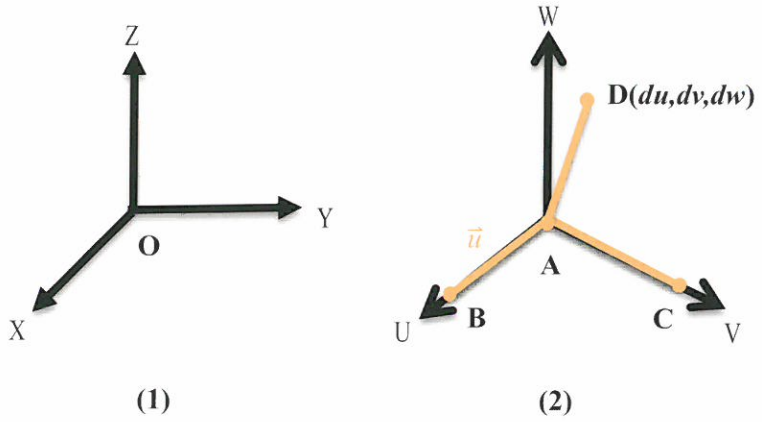


圖 20 (1) 世界座標與 (2) 局部座標系間座標變換

在本論文中會用到已知 A、B、C 3 點世界座標構成一 UVW 座標系，在 UVW 座標系中用極座標方式給定兩角度 α、β，定義一點 D 的 UVW 座標即 (*du, dv, dw*)，因此套用上列式子而得到 D 點之世界座標 (*dx, dy, dz*)。其使用時機有 2 處：其一、搖擺邊給予適當的 α、β 的角度，即可知道世界座標；其二、新的一圈知道兩個步驟形成顆數而得知兩正多邊形，因而得知兩正多邊形角度 α、β，即可知道世界座標。

4.4 特殊部位位置計算

在本論文中特殊部位位置計算有四種，包含兔子的前腳和後腳，兔子耳朵，與兔子身體的一部分，以下詳細說明：

4.4.1 兔子前腳與後腳

兔子之後腳與前腳分別以四邊形與五邊形之多邊形表示，詳細情形分述如下：

在兔子規則表 (在附錄五) 之步驟 5 與步驟 7 為建構兔子後腳 (以正五邊形表示)，需要黏貼於身體後半部分之一正五邊形上 (此多邊形大小與後腳之正五邊形大小一樣)，而遵循規則表上顯示兔子後腳之正五邊形，需以共邊黏貼於身體後半部之一正五邊形，如此則會造成兔子後腳姿勢 (或角度) 不合理，為解決此問題，我們採取折衷方式將後腳改為四邊形表示如圖 21 所示。在圖 21 中原先後腳之正五邊形 BHEFG 欲黏貼上身體之正五邊形 ABCDE 的一邊，我們的解決之道是後腳之多邊形，去除一頂點 H，形成四邊形 BEFG，而以 BE 邊 (原先正五邊形 BHEFG 之一對角線) 黏貼於身體上之正五邊形 ABCDE，如此一來後腳形成一搖擺面，我們再給予適當之搖擺角度即可成型 (套用 4.1 節與 4.3 節公式求得)。

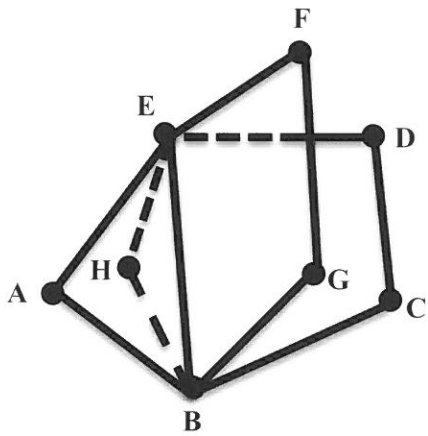


圖 21 兔子後腳資訊：四邊形 BEFG

關於兔子前腳的情況與後腳類似，兔子之兩前腳分別建構於規則表之步驟 16 與步驟 18，而前腳是正六邊形表示，需要黏貼於兔子身體前半部之一正五邊形上。同樣

地若需要以正五邊形與正六邊形以共一邊方式黏貼會出現前腳姿勢不對與共邊長度不一致問題。在本論文中，我們的的解決之道如圖 22 所示，在圖 22 中，原先之前腳正六邊形是多邊形 EFGHIJ 要黏貼身體上之正五邊形 ABCDE，我們採取類似後腳的處理方式，先將前腳之正六邊形去除一頂點 J，前腳形狀就會變成五邊形 EFGHI，欲以邊 EI(即原先六邊形 EFGHIJ 之一對角線 EI) 對準正五邊形 ABCDE 之對角線 BE(點 B 與點 E 相隔一頂點 A)，但是邊 EI 的邊長比邊 BE 長，為此我們將前腳之多邊形 EFGHI 之頂點 I 位置修正移到點 B。最後兔子前腳以五邊形 EFGHB 表示。如此一來前腳之五邊形也是一搖擺面，再給予適當的角度即可成型 (套用 4.1 節與 4.3 節公式求得)。

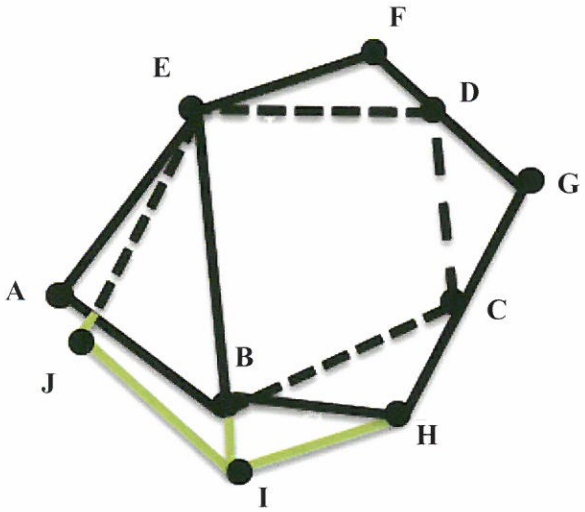


圖 22 兔子前腳資訊：五邊形 EFGHB

4.4.2 兔子耳朵

我們先在平面上標準位置算出兔子耳朵的多邊形位置，然後將此多邊形平移旋轉接到兔子頭部上去。我們的處理方式以多一顆珠子來完成它，這樣有一顆珠子重疊，卻能做出與兔子耳朵真實形狀極其相似，我們以耳朵前有 5 顆珠子，耳後有 4 顆珠子成型 (附錄五步驟耳 31 的耳朵有 8 顆) ，請參考圖 23，以十一邊形 ABCDEFGHIJK 表示兔子耳朵形狀，其中 K 點為座標系原點，A 點在 +X 軸上，點 A 至點 F 位於一圓弧上，而且點 G 至點 K 位於一垂直線 Y 軸上，在本論文中每顆珠子半徑設為 2，因此可以找到一圓。滿足上列條件，算出標準位置上之 11 點 (點 A 至點 K) 的座標。當要產生兔子耳朵多邊形時，將其標準位置座標經由座標變換成世界座標即可完成 (套用

4.1 節與 4.3 節公式求得)。

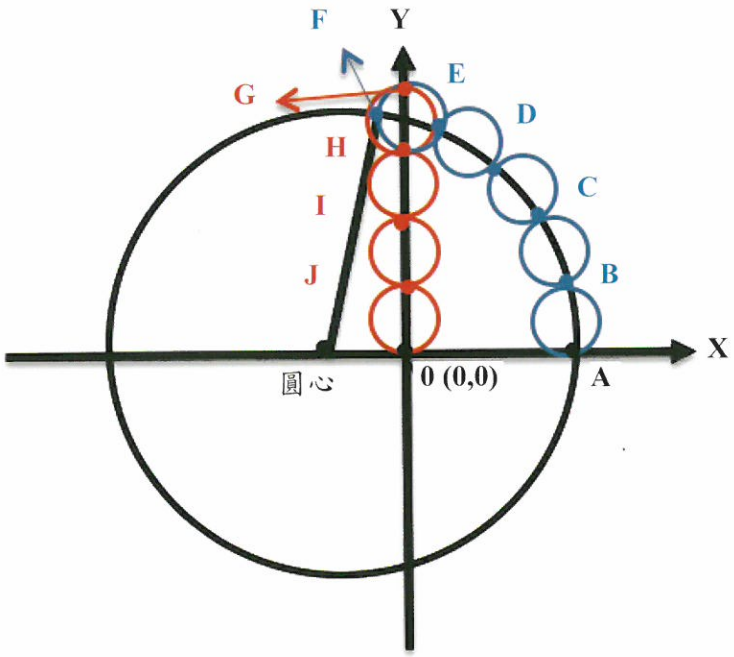


圖 23 兔子耳朵畫法

4.4.3 兔子身體

兔子身體因五邊形要架上六邊形 (如圖 24)，其架上去同時會出現兩個搖擺邊，與迷你雪納瑞頭含有正方形架上五邊形用球極座標方式求出點座標情況不相同，在本論文中我們使用了平面的相對方式去推算，首先利用兔子的建構方式以步驟 1 所產生之正五邊形平貼於 XY 面上，往後之步驟是往 +Z 軸方向建構多面體，因此我們想在 2 個相鄰五邊形上方架上一個六邊形能連接無隙縫，所以六邊形就無法維持正六邊形形狀。以圖 24 為例，面 1 與面 2 為正五邊形，我們想把面 3 六邊形架上去，面 3 上之邊 FA 與邊 AB 要分別面 1 與面 2 相鄰邊，此時我們的目的就是要將此六邊形其它的頂點 C、D、E 座標算出來。我們知道多邊形之邊長是珠子的直徑，在本論文中珠子直徑設為 4，所以我們令 E 點座標 =F 點座標 +(0,0,4)，C 點座標 =B 點座標 +(0,0,4)，而令 D 點座標 =A 點座標 +2*(0,0,Bz-Az)+(0,0,4) ，其中 Az、Bz 分別代表 A 點與 B 點之 z 座標。如此一來，架上去之六邊形各點座標即可得到。

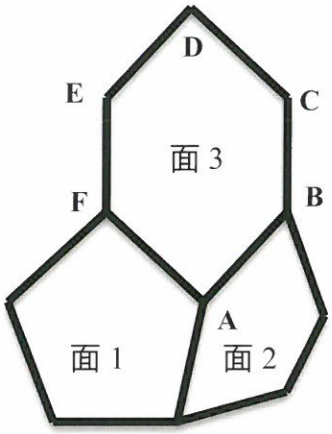


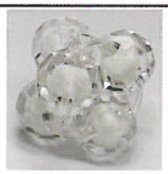
圖 24 五邊形架上六邊形

5. 實驗結果

在本論文中使用錐型鏈墜、方型鏈墜、球型鏈墜、迷你雪納瑞、傳統兔子之串珠規則表 (分別列於為附錄一至五) 做實驗。實驗結果將錐型鏈墜、方型鏈墜、球型鏈墜列表格呈現，其中含有實體圖、多面體圖、球體圖、珠體圖，與其資料量，而迷你雪納瑞、傳統兔子則含有實體圖、多面體圖、球體圖、與珠體圖，並且以四個角度的觀察串珠成像，詳細情形於下面各節中說明。

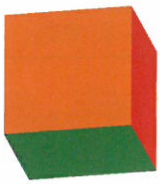
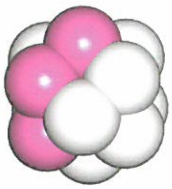
5.1 錐型鏈墜

本實驗以三角形為建構主體的錐型鏈墜，其規則表列於附錄一，實驗結果如下表：

	實體圖	錐型面	錐型（球體）	錐型（珠體）
錐型鏈墜				
資料量	原圖	4 個點、6 個邊、4 個三角面	6 顆球	6 顆球多面體、每個球多面體有 64 個 3 角面

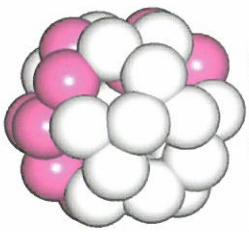
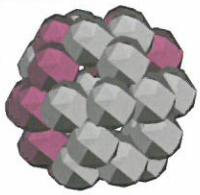
5.2 方型鏈墜

本實驗以正方形為建構主體方型鏈墜，其規則表列於附錄二，實驗結果如下表：

	實體圖	方型面	方型（球體）	方型（珠體）
方型鏈墜				
資料量	原圖	8 個點、15 個邊、6 個正方形	15 顆球	15 顆球多面體，每個球多面體有 64 個 3 角面

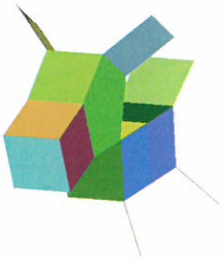
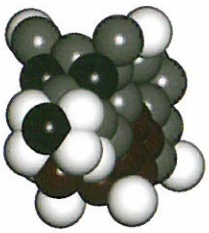
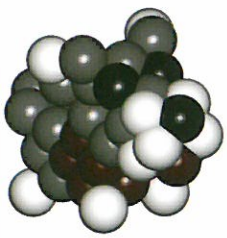
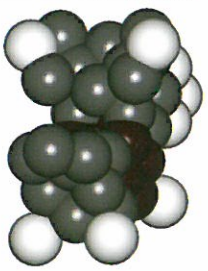
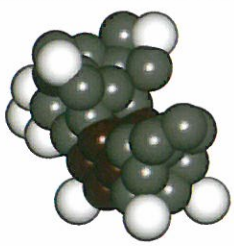
5.3 球型鏈墜

本實驗以正五邊形為建構主體的球型鏈墜，其規則表列於附錄三，實驗結果如下表：

	實體圖	正五多面體	五多面體（球體）	五多面體（珠體）
球型鏈墜				
資料量	原圖	20 個點、30 邊 12 個正五邊形	30 顆球	30 顆球多面體，每個球多面體有 64 個 3 角面

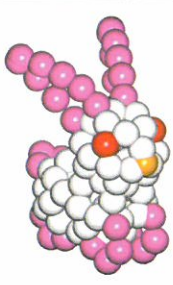
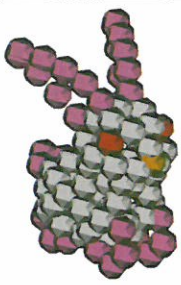
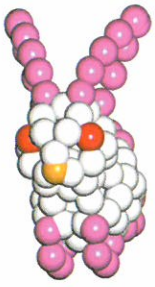
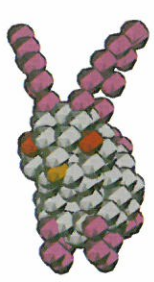
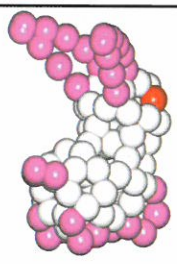
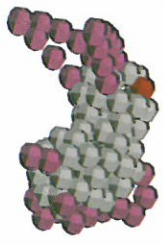
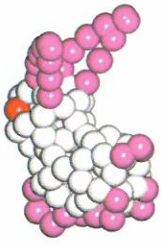
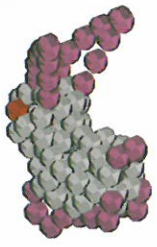
5.4 迷你雪納瑞

本實驗以正方形和五邊形為建構主體迷你雪納瑞的狗型串珠，其規則表列於附錄四，所用的珠子包含深灰色31顆、紅色16顆、黑色3顆、白色8顆，實驗結果如下表(由上至下觀看角度分別為左前、右前、右後、左後)：

實體圖	狗表面	狗串珠(球體)	狗串珠(珠體)
			
			
			
			

5.5 傳統兔子

本實驗以五邊形和六邊形為建構主體的兔型串珠，其規則表列於附錄五，所含蓋的珠子白色 70 顆、粉紅色 35 顆、暗紅色 2 顆、粉橘色 1 顆，實驗結果如下表 (由上至下觀看角度分別為左前、右前、右後、左後)：

實體圖	兔子表面	兔串珠(球體)	兔串珠(珠體)
			
			
			
			

6. 結論

本論文研究著重於人看著串珠規則表 (如附錄一到五)，一步一步手工做出成型，而轉為電腦一步一步模擬人為 3D 串珠成型，而這些有別於過去文獻所提的 2D 方式去建構平面的串珠。在本論文中有四個想法，第一、串珠規則表每一步驟把它設定為一面，進而逐步建構 3D 串珠多面體，最後呈現 3D 珠體圖；第二、使用球體極座標公式算座標；第三、標準位置座標轉到世界座標；第四、特殊部位處理方式。這四個想法實現了本研究以錐型鏈墜、方型鏈墜、球型鏈墜、迷你雪納瑞、與傳統兔子做實驗呈現 3D 串珠模型。

在目前為止未曾見過 3D 串珠電腦模擬成型的相關論文，只有出現專門做平面串珠的應用如 EasyBeadPatterns 軟體模擬 [9]，這有別於本論文所做的 3D 圖型模組。當然本論文也是有一些限制的，如加珠過洞與特殊部位，規則表步驟用電腦模擬不易達成，關於這類問題，本論文採取迂迴做法，變為特殊的處理方式，例如做兔子腳的方式，在規則表中它屬於身體連著腳去做，意即我們所建面的方式就在這個區塊不成立，這需特殊的處理方式去解決。關於串珠模型的特殊部位處理，一般是針對 3D 模型有較尖銳或是搖擺邊界部位的處理 (如動物串珠模型的腳與耳朵部位)，以及動物串珠模型身體邊界面中之相鄰兩圈，每圈邊界都是由同一邊數之多邊形圍繞而成，但相鄰兩圈所用之多邊形邊數不相同，為維持 3D 模型多面體沒有破洞，而需要特別處理 (如本論文中之傳統兔子模型多面體的身體部份)。在本論文中已處理了典型的動物模型特殊部位，但當中還有需要人為調整的資訊，如搖擺面或搖擺邊的方位角度，若是設定值給的不適當，會造成 3D 模型姿勢不自然。

最後，將來希望可以處理更多的樣式串珠表，讓更多的串珠朋友們了解 3D 模型的樣式，並讓更多人知道串珠的美麗與其存在的價值。

參考文獻

1. P.Shirley and S. Marschner,"Fundamentals of Computer Graphics", A. K.Peters, 2009
2. R. Wright,N. Haemel, G.sellers, and B.Lipchak,OpenGL SuperBible, 5th Edition, Addison-Wesley, 2011
3. 內藤朗 (2009) Shopping 趣 -- 串珠迷你吊飾 55 款 , 瑞昇文化事業股份有限公司
4. 林淑惠 (2007) 彩晶造型炫風篇 , 民聖文化事業股份有限公司
5. 容誼珍、蔡淑夏 (2010) 袖珍串珠迷你公仔 , 民聖文化事業股份有限公司
6. 陳湘尹 (2007) 表格串珠 ~ 愛狗造型 , 民聖文化事業股份有限公司
7. 倉上由美子 (2008) 串珠小人國 -- 用串珠辦家家酒 , 瑞昇文化事業股份有限公司
8. 綠川久紀子 (2009) 時尚串珠飾品 DIY, 楓書坊文化出版社
9. http://www.easybeadpatterns.com/(easybeadpatterns 平面串珠軟體)
10. http://www.mydiyclub.com/article-1162-1.html(簡易串珠教學網)

Recevied Oct 7, 2014
Revised Mar 3, 2015
Accepted Mar 10, 2015

附錄一 錐型鏈墜

步驟	1	2	3	4
左線過洞	0	0	1	3
右線加珠	3	2	1	0
交叉	3	3	3	3
形成顆數	3	3	3	3

附錄二 方型鏈墜

步驟	1	2	3	4	5	6
左線過洞	0	0	1	1	2	4
右線加珠	4	3	2	2	1	0
交叉	4	4	4	4	4	4
形成顆數	4	4	4	4	4	4

附錄三 球型鏈墜

步驟	1	2	3	4	5	6
左線過洞	0	0	1	1	1	2
右線加珠	5	4	3	3	3	2
交叉	5	5	5	5	5	2
形成顆數	5	5	5	5	5	5

步驟	7	8	9	10	11	12
左線過洞	1	2	2	2	3	5
右線加珠	3	2	2	2	1	0
交叉	5	5	5	5	5	5
形成顆數	5	5	5	5	5	5

附錄四 迷你雪納瑞

4mm 深灰 *31 4mm 紅 *16 4mm 黑 *3 4mm 白 *8 釣魚線主色灰色

步驟	身 1	尾 2	3	腳 4	5	6
左線過洞	0	0	0	過 1 加 2(1 白 2 小丸	0	1
右線加珠	4	3	3(2 紅)	0	2(1 紅)	2(1 紅)
交叉	4	原點	4	跳 1 回 1	4	4
形成顆數	4	4	4	2	4	4

步驟	腳 7	8	9	10	11	12
左線過洞	加 2(1 白 2 小丸)	2	0	1	1	2
右線加珠	0	1 紅	3 紅	2 紅	2 紅	1 紅
交叉	跳 1 回 1	4	4	4	4	4
形成顆數	2	4	4	4	4	4

步驟	腳 13	頭 14	15	16	17	耳 18
左線過洞	加 2(1 白 2 小丸)	2	0	1	1	過 2 加 3(2 白)
右線加珠	2(1 白 2 小丸)	過 2 加 3	4(2 黑)	2	3(2 黑)	加 3(2 白)
交叉	各跳 1 回 1	4	4	4	4	4
形成顆數	2	4	5	4	5	4

步驟	19	20	嘴 21	22	23	24
左線過洞	0	1 黑	0	1	1	2
右線加珠	1	過 1 黑加 1	3(2 黑)	2(1 白)	2(1 紅)	1 白
交叉	4	4	4	4	4	4
形成顆數	4	4	4	4	4	4

步驟	25					
左線過洞	4					
右線加珠	0					
交叉	4					
形成顆數	4					

附錄五 傳統兔子

4mm 白 *70 4mm 粉 *35 4mm 暗紅 *2 3mm 粉橘 *1 釣魚線 主色白

步驟	身 1	尾 2	3	4	腳 5	6
左線過洞	0	0	0	1	0	1, 2
右線加珠	5	3 粉	4	3	4(2, 3, 4 粉)	2
交叉	5	原點	5	5	5	6
形成顆數	5	4	5	5	5	6

步驟	腳 7	8	9	10	11	12
左線過洞	1	0	2	1	2	2
右線加珠	4(2, 3, 4 粉)	過 2 加 2	2	4	3	3
交叉	6	5	5	6	6	6
形成顆數	6	5	5	6	6	6

步驟	13	14	15	腳 16	17	腳 18
左線過洞	2	3	過 1 加 3	加 4 粉	3, 1 加 2	加 5(2, 3, 4 , 5 粉)
右線加珠	3	2	0	過 1	0	過 1
交叉	6	6	5	6	7	6
形成顆數	6	6	5	6	7	6

步驟	19	20	21	22	頭 23	24
左線過洞	3，1	加 2	加 1	5	0	1
右線加珠	1	過 2	過 3	0	5(3 小粉)	3
交叉	5	5	5	5	6	5
形成顆數	5	5	5	5	6	5

步驟	25	26	27	28	29	30
左線過洞	1	1	2	1，跳 1 小粉	跳 1 小粉過 2	2
右線加珠	3	3	2	3(1 黑)	2(2 黑)	2
交叉	5	5	5	5	5	5
形成顆數	5	5	5	5	5	5

步驟	耳 31	32	耳 33	34	35	
左線過洞	過 7 粉(跳 4 粉)	0	加 8 粉	0	5	
右線加珠	8 粉	2	過 7 粉(跳 4 粉)	1	0	
交叉	原點	5	原點	5	0	
形成顆數	9	5	9	5	0	

Computer Simulation of Three Dimensional Bead Threading

Jian-Wei Syu and Chin-Ho Cheng

Department of Computer Science and Information Engineering
Fu Jen Catholic University
New Taipei City, Taiwan 24205, R.O.C.

Abstract

Threading a string of beads is an art. If we want any string shape of beads, it can be threaded by hand. In this thesis, we focus on 3D bead strings by using the rules of bead stringing. Based on the rules, we simulate the construction of the 3D handmade bead strings step by step. An algorithm is proposed to show that, for a given shape of bead strings, its corresponding 3D beaded polyhedron is constructed first, and then place one ball on the midpoint position of each edge of this polyhedron. Finally, it forms our desired 3D bead string. In our experiments, we test five strings of beads which are the cone, cube, ball, miniature schnauzer, and traditional rabbit shapes. By using computer graphics techniques, the constructed 3D model of each tested bead string is rendered. From the experimental results, it shows that computer simulation of bead stringing can be easily made by using the rules of bead stringing.

Key words: Bead Stringing, Computer Simulation, Computer Graphics

Application of Molecular Imprinting for Improving Odorant Detection by Peptide Libraries

Heau-Shan Gao*, Jen-Yu Chen, Yi-Shan Chien, Chuang-Fu Ha

Department of Chemistry, Fu-Jen Catholic University, Taipei County, Taiwan, R.O.C.

Abstract

A novel method for odorants detection is developed by integrating the functional peptides from combinatorial chemistry and molecular imprinting technology. The molecular imprinting technology is applied to enhance the sensitivity and specificity of binding ability of functional peptide with the target odorant. In the study, the hexapeptide library was prepared by using the combinatory chemistry. The peptides were mixed with the target odorant to prepare the molecularly imprinted peptides, then were coated on a piezoelectric quartz crystal. Based on the positional screening strategy, the imprinted peptides with high binding ability of butyric acid odorant was determined then the best amino acid candidates of each position of hexapeptide were identified. The binding affinity of the odorant with the imprinted peptides and non-imprinted peptides were also determined. The result was shown that the imprinted peptides had higher sensitivity than the non-imprinted peptides. The binding ability of imprinted peptide was 1.2~5.5 fold higher than non-imprinted peptide. The sensitivity of binding between the peptides and the odorants was significantly enhanced by the effect of the molecular imprinting. The combination of peptide library and molecular imprinting technology can be used for development of odorant sensing tools.

Key words: combinatorial chemistry, peptide library, molecular imprinting, odorant

1. Introduction

The mammalian olfactory system can recognize and discriminate large number of different odorant molecules.[1] The odor discrimination occurs during the association of odorous ligands with specific receptors on olfactory sensory neurons. The receptor protein plays an important role in odorant recognition and cell signaling. The literature reports suggest that the extracellular loop and transmembrane domain might be the major part of the odorant-binding domain in olfactory receptors.[2,3] The design of functional polymers that can selectively recognize molecules has become an active area of research in recent years.[4,5]

The recognition and subsequent complementary binding between a substrate and an odorant molecule is the key step in the odor discrimination process. To mimic the olfactory sensing system, a series of synthetic peptides as sensing materials were designed and studied. The peptide library that is composed of thousands or millions of peptide sequences offers a high throughput for drug screening or compound screening.[6,7] However, the peptide library screening approach may have low selectivity and sensitivity to the target compound. Since the conformations of peptides are considered as one of the key factors for absorption effect between peptides and odorant molecules, integrating the molecular imprinting technology shall create the selective recognition sites for the template molecule[8, 9] These recognition sites have a complementary shape and size with the template compound which can enhance the sensitivity and specificity of binding ability of functional peptide with the target odorant. An imprinted functional peptide can form a specific configuration with a complementary target odorant and increase the binding ability with the target odorant.

The objective of our research is to develop a novel method for odorants detection by integrating the functional peptides from combinatorial chemistry and molecular imprinting technology. The butyric acid is selected as a target odorant to investigate the sensitivity and specificity effect of imprinted functional peptide and non-imprinted peptide on their sensing ability for odorant detection. The imprinted peptides with high binding ability of butyric acid are selected by the positional screening process. The potential amino acid candidates corresponded to six positions of hexapeptide are identified.

2. Method

This study integrates the peptide library and molecular imprinting technologies. The hexapeptide library was synthesized by solid phase peptide synthesis method.10,11 Based on the positional screening strategy,[12, 13] the 114 hexapeptide combinatory libraries were prepared (Fig. 1).

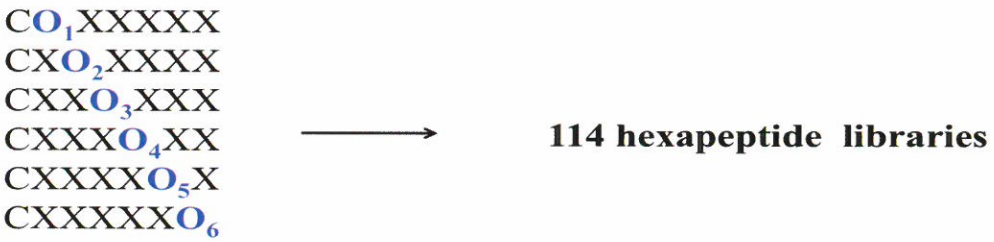


Fig. 1 Positional Scanning of Synthetic Peptide Combinatorial Libraries

O: "Known position" represents a known amino acid which is one of the 19 natural L-amino acids (cysteine was excluded).

X: "Mix position" represents an equimolar mixture of the 19 natural L-amino acids (cysteine was excluded).

C: Additional cysteine residue provides a thio (-SH) group to form a chemical adsorption between the peptide and the gold surface of the piezoelectric crystal.[14]

These hexapeptide libraries were mixed with the target odorant to prepare the molecularly imprinted peptides then coated on a piezoelectric (PZ) quartz crystal.[15,16]

Imprinted functional peptide preparation:

(a)The hexapeptide was dissolved in the butyric acid solution to form the covalently interaction between peptide and the template butyric acid.

(b)The hexapeptide - butyric acid complex was coated on the surface of piezoelectric quartz crystal.

(c)The template butyric acid was removed from surface of piezoelectric quartz crystal.

Non-imprinted peptide preparation:
(a)The hexapeptide was dissolved in the 75% ethanol solution.
(b)The hexapeptide solution was coated on the surface of piezoelectric quartz crystal.

The PZ crystal was served as a signal transducer to determine the binding affinity of synthetic peptides for odorants (Fig. 2).

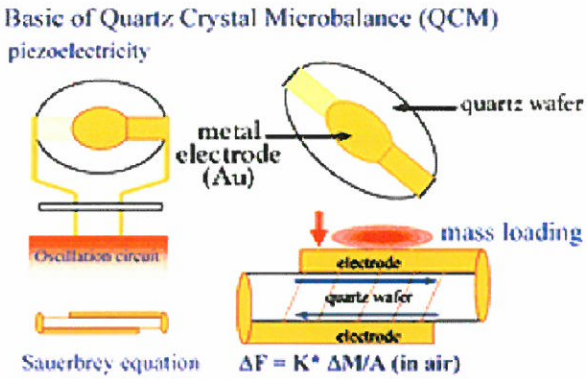


Fig. 2 The Piezoelectric Crystal Served As a Signal Transducer

The sensitivity between peptide and butyric acid on a piezoelectric quartz crystal was determined.

$$\text{Sensitivity} = \frac{\text{The change of frequency } \Delta F \text{ [Hz]}}{\text{Coating amount of peptide } \Delta M \text{ [}\mu\text{g]}}$$

3. Results and Discussion

The conformations of peptides are considered as one of the key factors for absorption effect of peptides on odorant molecules. The molecular imprinting process is applied to enhance the binding ability of peptide with the target odorant (Fig. 3). An imprinted functional peptide can form a specific configuration with a complementary target odorant and enhance the binding ability with the target odorant.

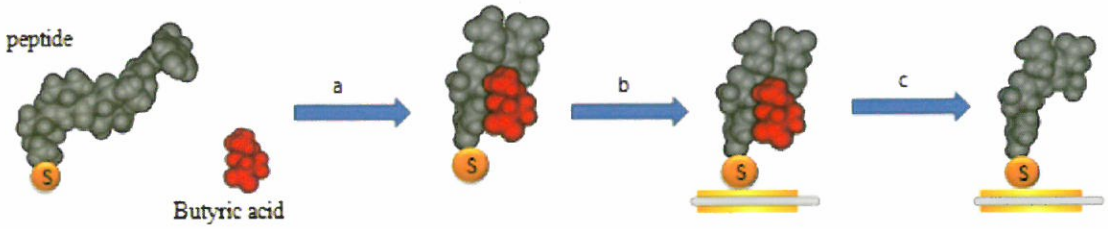


Fig. 3 Schematic Representation of Peptide Molecular Imprinting Process

The binding affinity of the odorant for the imprinted peptides were studied. The imprinted peptides with high binding ability to butyric acid odorant were selected by the positional screening process. The best amino acid candidates corresponded to six positions of hexapeptide were identified.

On the basis of the positional screening approach of the 114 hexapeptide libraries, the result was shown that His and Thr have higher sensitivity to odorant butyric acid in the first position of hexapeptide library (CO_1XXXXX). Therefore, the best amino acid candidate in the first position of hexapeptide library is His or Thr. In the second position of hexapeptide library (CXO_2XXXX), the best amino acid candidate is The or Val. In the third position of hexapeptide library ($CXXO_3XXX$), the best amino acid candidate is Leu or Trp. In the fourth position of hexapeptide library ($CXXXO_4XX$), the best amino acid candidate is Pro or Val. In the fifth position of hexapeptide library ($CXXXXO_5X$), the best amino acid candidate is Gly or Tyr. In the sixth position of hexapeptide library ($CXXXXXO_6$), the best amino acid candidate is Tyr (Fig. 4). The top two amino acid candidates at six positions for hexapeptide are shown in Table 1.

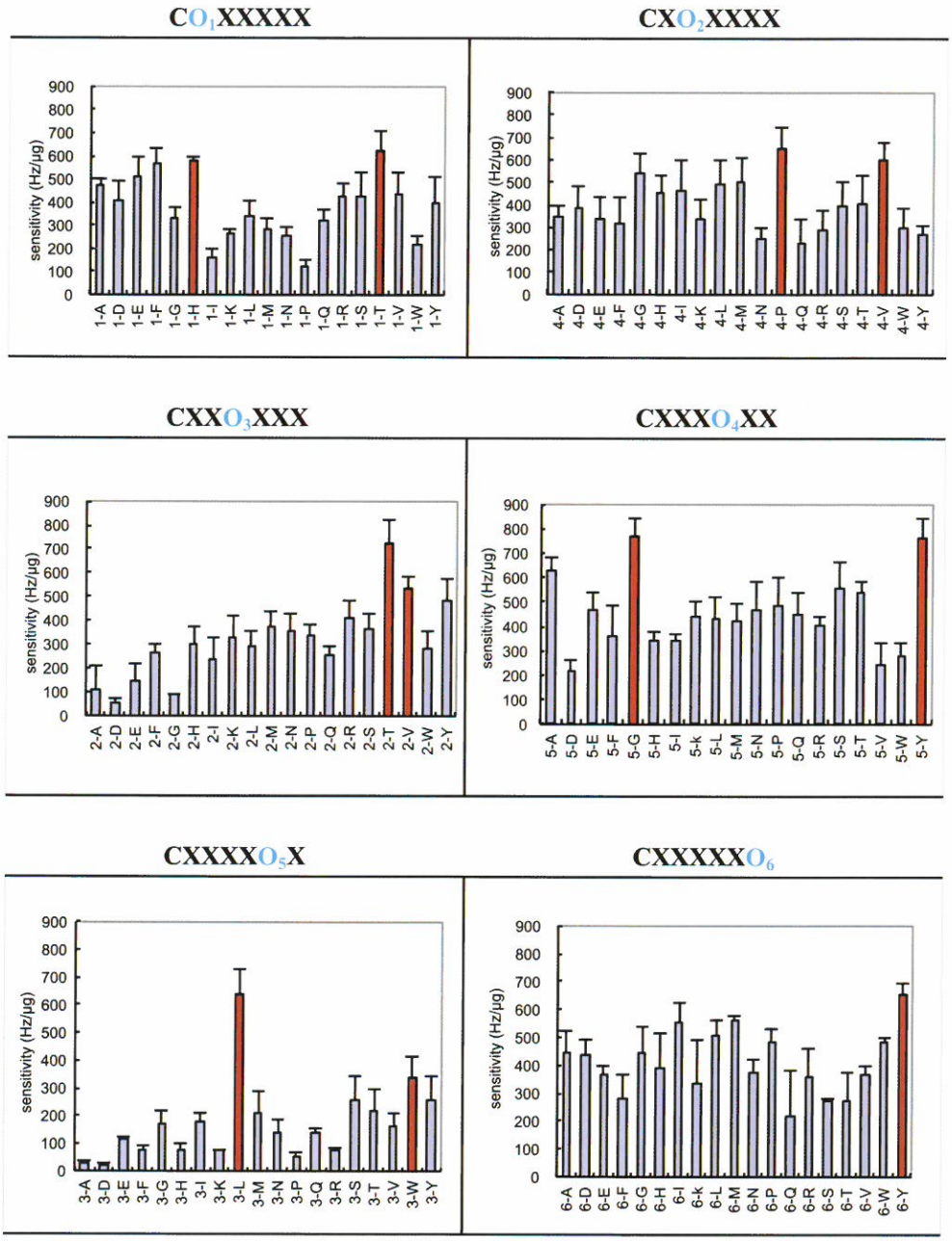


Fig. 4 Binding Ability of Imprinted Hexapeptide Library with Butyric Acid

	X_1	X_2	X_3	X_4	X_5	X_6
1st amino acid candidate	H	T	L	P	G	Y
2nd amino acid candidate	T	V	W	V	Y	

Table 1 The best amino acid candidate in hexapeptide

Based on the data of the amino acid candidates in Table 1, it was shown that the butyric acid molecule was interacted with peptide through the hydrogen bonding between the hydroxyl group of Thr/Tyr and butyric acid.

Assessment of the selectivity and specificity effect of imprinting functional peptides on the odorant, the binding affinity of butyric acid for the imprinted and non-imprinted peptides were studied (Fig. 4). The less sensitivity peptides (1-P, 2-G, 3-D, 4-H, 5-K, and 6-F) at the sixth position of the amino acid candidates were served as a negative control. The result demonstrated that the imprinting functional peptides (red bar) have higher affinity for odorant butyric acid than the non-imprinted peptides (blue bar) at each position of the best amino acid candidate. The sensitivity of imprinted peptide is 1.2~5.5 fold higher than that of non-imprinted peptide in those position of the best amino acid candidate (Table 2). These data indicate that the imprinted peptide have selective recognition.

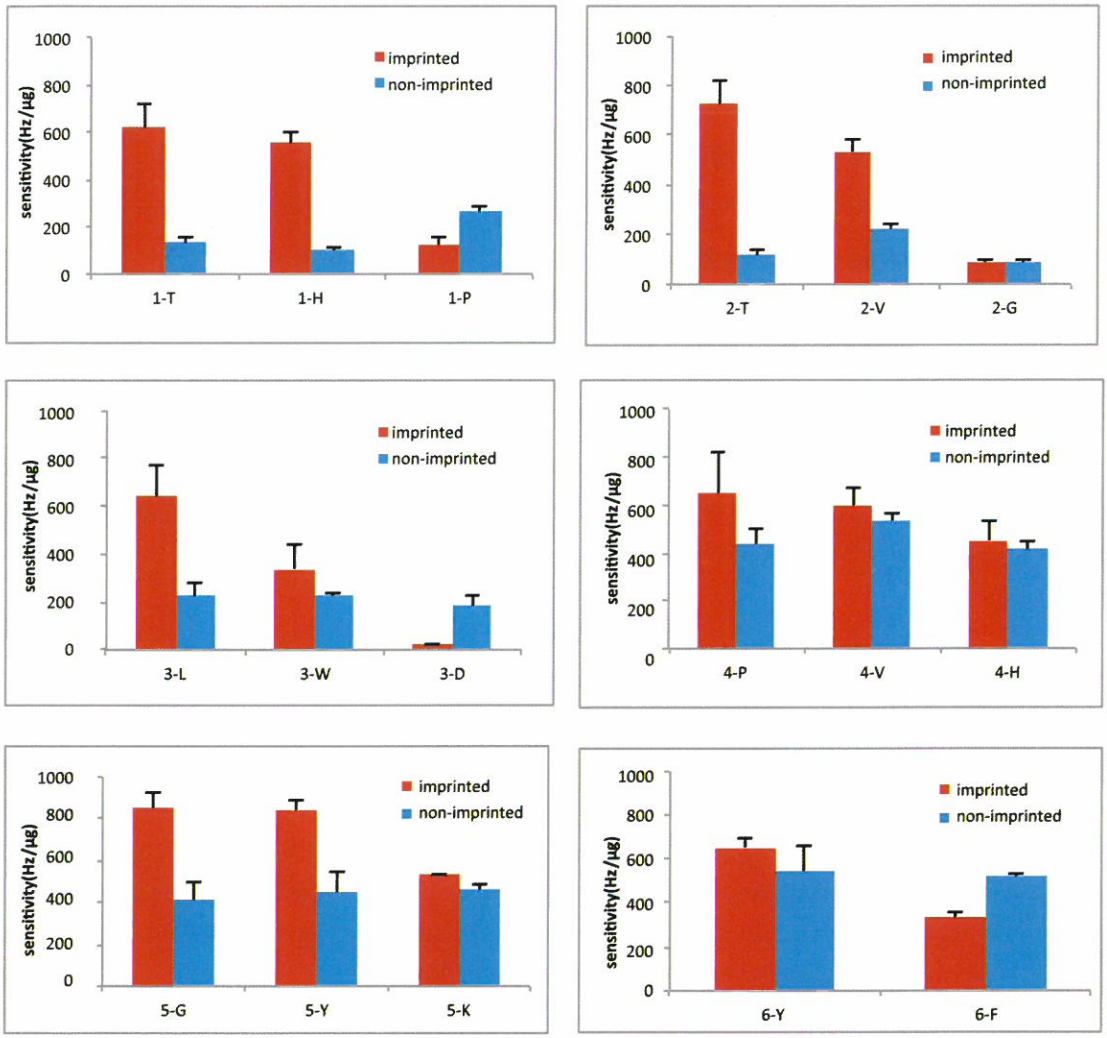


Fig. 5 Binding Affinity of Butyric Acid for the Imprinted and Non-imprinted Peptides

	Sensitivity of Imprinted Peptide (Hz/ug)	Sensitivity of Non-imprinted Peptide (Hz/ug)	Sensitivity Comparison of Imprinted and Non-imprinted Peptide
1-T	624	134	4.7
1-H	557	102	5.5
2-T	725	119	6.1
2-V	536	222	2.4
3-L	640	226	2.8
3-W	338	227	1.5
4-P	652	438	1.5
4-V	660	536	1.2
5-G	855	409	2.1
5-Y	843	447	1.9
6-Y	655	547	1.2

Table 2 Comparison of The Sensitivity of Imprinted and Non-imprinted Peptides

4. Conclusion

The result showed that the molecular imprinting process can enhance the binding affinity of peptide for the odorant and improve the selectivity and specificity of peptides. The combination of peptide library and molecular imprinting technology can be used for development of odorant sensing tools in environmental, food, and medical sample analysis.

Reference

1. Buck, L., Axel, R. **1991**. A Novel Multigene Family May Encode Odorant Receptors: A Molecular Basis for Odor Recognition. *Cell.*, 65, 175–87.
2. Pimple, Y. and Lancet, D. **1999**. The variable and conserved interfaces of modeled olfactory receptor proteins. *Protein Science.*, 8: 969-977.
3. Freitag, J., Ludwig, G., Andreini, I., Rossler, P. and Breer, H. **1998**. Olfactory receptor in aquatic and terrestrial vertebrates. *J. Comp. Physiol. A.*, 183, 635-650.
4. Sellergren, B. **2001**. Techniques and Instrumentation in Analytical Chemistry: Molecularly Imprinted polymers. *Elsevier Amsterdam*, 23:126-138.
5. Dı′ az-Garcı′ a, M. E. and Badı′ a Laı′ no, R. 2005. Molecular Imprinting in Sol-Gel Materials: Recent Developments and Applications. *Microchim. Acta*, 149, 19–36
6. Houghten, R. A., Pinilla, C., Blondelle, S. E., Appel, J. R., Dooley, C.T., Cuervo, J. H., **1991**. Generation and use of synthetic peptide combinatorial libraries for basic research and drug discovery. *Nature.*, 354(7), 84-6.
7. Batra, D. and Shea, K. J. **2003**. Combinatorial methods in molecular imprinting. *Curr. Opin. Chem. Biol.*, 7, 434-442.
8. Hart, B. R., Shea, K. J., **2001**. Synthetic Peptide Receptors: Molecularly Imprinted Polymers for the Recognition of Peptides Using Peptide-Metal Interactions. *J. Am. Chem. Soc.*, 123(9), 2072-3.
9. Haupt, K., Mosbach, K., **2000**. Molecularly Imprinted Polymers and Their Use in Biomimetic Sensors. *Chem. Rev.*, 100(7), 2495-504.
10. Merrifield, R. B., **1963**. Solid Phase Peptide Synthesis. I. The Synthesis of a Tetrapeptide. *J. Am. Chem. Soc.*, 85(14), 2149-54.

11. Merrifield, R. B., **1986**. Solid phase synthesis. *Science*.,232,341
12. Pinilla, C., Appel, J. R., Blanc, P. and Houghten, R. A. **1992**. Rapid identification of high affinity peptide ligands using positional scanning synthetic peptide combinatorial libraries. *BioTechniques*, 13:6, 901- 905.
13. Dooley, C. T. and Houghten, R. A. **1993**. The use of positional scanning synthetic peptide combinatorial libraries for the rapid determination of opioid receptor ligands. *Life Science*, 52, 1509-1517.
14. Zhou, X. C., Zhong, L., Li, S. F. Y., Ng, S. C. and Chan, H. S. O. **1997**. Organic vapor sensors based on quartz crystal microbalance coated with self-assembled monolayers. *Sensors and Actuators B*., 42, 59-65.
15. Skladal, P., **2003**. Piezoelectric Quartz Crystal Sensors Applied for Bioanalytical Assays and Characterization of Affinity Interactions. *J. Braz. Chem. Soc*., 14(4), 491-502
16. O'Sullivan, C. K. and Guilbault, G. G. **1999**. Commercial quartz crystal microbalances – theory and applications. *Biosens. Bioelec*., 14, 663 - 670.

Recevied Oct 7, 2014
Revised Mar 3, 2015
Accepted Mar 10, 2015

應用合成胜肽和分子印模對氣味篩選之研究

高華生*，陳貞妤，簡薏珊，胡創富

輔仁大學 化學系

摘　要

本研究應用組合化學的概念及分子印模 (molecular imprinting) 的策略，設計及製備一系列具有氣體辨識能力的胜肽。本研究以化學混合法製備含有六個胺基酸長度的組合胜肽庫 (combinatorial peptide library)。以胜肽庫為模板分子與氣味分子混合，將模板胜肽聯結於壓電晶體訊號轉換元件製成生物晶片，保留被氣味分子印模的特異構形。進而探討胜肽和目標氣味分子間經印模處理後的吸附效能，會否因胜肽與目標氣味分子形成特殊活性構形，能有效提高生物晶片對氣味分子偵測的靈敏性，增強其辨別度。本研究結果顯示經印模處理後胜肽和目標氣味分子丁酸間偵測的靈敏性高過未經印模處理胜肽約 1.2~5.5 倍。顯示分子印模效應能增強胜肽偵測目標氣味分子的靈敏性。本研究的結果可應用於氣味分子檢測工具的開發。

關鍵字：組合化學、胜肽庫、分子印模、氣味分子。

A Note on the Consistency Between Process Capability Indices and Percentage of Conformance

Sy-Mien Chen and Yu-Sheng Hsu*

Department of Mathematics, Fu-Jen Catholic University
New Taipei City, Taiwan, R.O.C. 24205
**Department of Mathematics, National Central University*
Chung-Li, Taiwan, R. O. C. 32001

Abstract

There are many approaches in capability analysis. Among them, process capability index has been proven useful in many industries to provide information on process performances. The evolution of the theory and application of process capability analysis depends largely on the inventions of new indices. There has been conformance comparisons among competing indices in literature. Very often, when a new index is proposed, authors emphasize the consistency between the new index and the percentage of conformance in order to support the superiority of the new index. However, the power of the consistency between an index and conformance is suspectable. In this paper, based on nine popular indices, we quantify the level of consistency. We also find and discuss regions of consistency under various restrictions of parameters. The main purpose of this paper is to warn index users that even the results from index and percentage of conformance are consistent, it does not mean they are telling you the real quality of a process.

Key words: Symmetric tolerance; Asymmetric tolerance; Consistency; Percentage of conformance; Process capability index.

1. Introduction

It has been four decades since Juran *et al.*(1974) first proposed the idea of process capability index which connects the relationships between the actual process performance and a preset level of production tolerance. From then on, there have been a huge amount of articles about capability indices (*PCI*), whilst proposing new indices and studying their inferential properties are the main issue. For example, articles about the index for process with symmetric tolerance (i.e. when the target is at the midpoint of the specification interval (*LSL,USL*)), are Sullivin (1984, 1985), Kane(1986), Chan et al. (1988), Pearn et al. (1992) and Greenwich and Jahr- Schaffrath (1995). For processes with asymmetric tolerances, (i.e. deviations from target are less tolerable in one direction than the other) many researchers proposed indices by extending those existing indices which were designed for symmetric case. For instance, Kane (1986), Chen et al. (1988), Boyles (1994), Vännman (1997), Chen (1997, 1998), Pearn and Chen (1998), Chen et al. (1999), Pearn et al. (1999), and Pan and Lee (2009).

There is a great and still growing library of literature for process capability analysis. For those who are interested in the field, see the books by Kotz and Johnson (1993), Kotz and Lovelace (1998), Pearn and Kotz (2006). And the review articles, see Tang and Than (1999), Palmer and Tsui (1999), Kotz and Johnson (2002), Spiring *et al.* (2002), Spiring *et al.* (2003), Yum and Kim (2010), Mohammad (2012), Nandini and Saurav (2013), Nandini and Dwivedi (2013), and references therein.

Very often, for two processes with different quality, when the contribution of a research is about providing a new index, authors emphasized that the value of traditional index remain the same while the new index can tell the difference. The conclusion is based on percentage of conforming. For example, Pan and Lee (2009) and references from there in. However, using the consistency between index and percentage of conformance (*POC*) to support the superiority of the new index may be unreliable because the true quality may be completely reversed to the concluded result (as we can see in secion 4). In this paper, we first quantify the level of consistency. Then, we find and discuss regions of consistency under various restrictions of parameters for nine popular process capability indices. From the discussion, we will see that a

high percentage of conformance may not correspond to a high quality process.

The rest of this paper is organized as the following: In section 2, we will present a brief review on some existing (in)capability indices which were proven relatively better than other indices. In section 3, based on nine popular process capability indices, a numerical study will be conducted to investigate the consistency between *PCI* and *POC* for normal processes. In section 4, major findings will be given. Concluding remarks are provided in section 5.

2. A brief review on some existing indicies

There are very fruitful discussions on process capability indices in existing literature. Based on the conclusions from the existing research, we study only those relatively better indices described below :

$C_{pa}(u, v) = \frac{d-|\mu - m| - u|\mu - T|}{3\sqrt{\sigma^2 + v|\mu - T|^2}}$, $u, v \geq 0$ (*Vännman* (1997));

$S_{pk} = \frac{1}{3}\Phi^{-1}\{\frac{1}{2}\Phi(\frac{USL - \mu}{\sigma}) + \frac{1}{2}\Phi(\frac{\mu - LSL}{\sigma})\}$ (*Boyles* (1994);

$C''_{pk} = \frac{d^* - A^*}{3\sigma}$ (Kane (1986));

$C''_{pm} = \frac{d^*}{3\sqrt{\sigma^2 + A^2}}$ (Chen. (1997)).

$C''_{pmk} = \frac{d^* - A^*}{3\sqrt{\sigma^2 + A^2}}$ (Pearn *et al.* (1999)),

$C_{pp} = ((\mu - T)/D)^2 + (\sigma / D)^2$ (Greenwich and Jahr-Schaffrath (1995))

$C''_{pp} = (\frac{A}{D})^2 + (\frac{\sigma}{D})^2$ (Chen (1998))

$C''_{pp}(s, t) = \frac{md(s,t)}{D^2} + (\frac{\sigma}{D})^2$ (Pan and Lee (2009))

where μ is the process mean, σ is the process standard deviation, Φ is the distribution function of the standard normal distribution.

Moreover, $d = \frac{USL - LSL}{2}$ is the half length of the specification limits, $d^* = min\{D_l, D_u\}$, $D = d^*/3$, $A = max\{\frac{d(\mu - T)}{D_u}, \frac{d(T-\mu)}{D_l}\}$, $A^* = max\{\frac{d^*(\mu - T)}{D_u}, \frac{d^*(T-\mu)}{D_l}\}$, here $D_l = T - LSL$, $D_u = USL - T$, $m = \frac{USL + LSL}{2}$ is the midpoint of the specification limits.

Furthermore, the modified desirability function md(s, t) (Mayer & Montegomery (1995)) is defined as

$$md(s,t) = \begin{cases} \frac{d^2}{D_l^s}(T-\mu)^s & \text{if } \mu < T \\ \frac{d^2}{D_u^t}(\mu - T)^t & \text{if } \mu > T. \end{cases}$$

The function $md(s, t)$ can be treated as a piecewise loss function, where $\frac{d^2}{D_l^s}(\mu - T)^s$ and $\frac{d^2}{D_u^t}(T-\mu)^t$ represent the monetary losses incurred at the lower and upper specification limit, respectively. Pan and Lee (2009) suggested that the two parameters s and t can be found by $s = \frac{USL - LSL}{D_u}$ and $t = \frac{USL - LSL}{D_l}$, such that $s > t$ if $D_l > D_u$ and $s < t$ if $D_l < D_u$.

Notice that $C''_{pp}(2, 2) = C''_{pp}$, and $C''_{pp}(2, 2) = C_{pp}$ if the target T is at the mid-point of the specification limits.

Among these indices, $C_{pa}(u, v)$, S_{pk} and C_{pp} are designed for processes with the symmetric tolerance, and the rest are for processes with the asymmetric tolerance. $C_{pa}(1, 3)$ and $C_{pa}(0, 4)$ are most sensitive to process departure from the target value T among all possible u, v (Vännman K. (1997)). The index S_{pk} is a one-to-one transformation of percentage of conformance which represents the actual process yield.

Notice that the values of $C_{pa}(u,v)$, C''_{pm} and C''_{pmk} decrease faster when the process mean shifts away from the target to the closer specification limit than that to the farther specification limit. Indices C''_{pk}, C''_{pm}, C''_{pmk} and $C_{pa}(u, v)$ obtain the maximal values at $\mu = T$, where C_{pp}, C''_{pp} and $C''_{pp}(s,t)$ achieve the minimum at $\mu = T$.

3. Consistency between *POC* and *PCI*

Given two-sided specification limits LSL and USL and target value T, consider two normal processes A and B with different process means μ_A and μ_B with common standard

deviation σ.

The percentages of conformance for processes A and B are defined as

$$POC_A = \Phi\{\frac{USL - \mu_A}{\sigma}\} - \Phi\{\frac{LSL - \mu_A}{\sigma}\}$$

and

$$POC_B = \Phi\{\frac{USL - \mu_B}{\sigma}\} - \Phi\{\frac{LSL - \mu_B}{\sigma}\}$$

respectively. It is obvious that the values of POC_A and POC_B are not affected by the target value T.

To calculate the process capability indices, let $r = \frac{D_l}{D_u}$ be the level of asymmetry of tolerance, then $LSL = T - \frac{2rd}{1+r}$ and $USL = T + \frac{2d}{1+r}$, where d is the half length of the tolerance interval. Throughout this study, we assume that $d = 1$ and the target value $T = 20$ from which the values of LSL and USL can be determined. Moreover, 6 levels of asymmetry are considered, namely $r \in \{0.25, 0.75, 1, 1.25, 2.75, 4\}$. See Table 1 for these preset values. Notice that when $r = 1$, we have the case for the symmetric tolerance.

Table 1: Specification intervals (LSL, USL), $T = 20$

r.	.25	.75	1	1.25	2.75	4
LSL	19.6	19.142857	19	18.89	18.53	18.4
USL	21.6	21.142857	21	20.89	20.53	20.4

Since PCI is considered only for processes which are under statistical control, we need the process means μ_A and μ_B lie in $[LSL, USL]$. Given r and hence LSL and USL, we choose different μ_A and μ_B from $\{LSL + k\cdot(0.05), k = 0,1,..., \frac{USL - LSL}{0.05}\}$. Therefore, for given r, there will be totally 41 different process means to choose from and 1640 ($= 41 \cdot 40$) combinations of (μ_A, μ_B). Moreover, to cover the cases from small to large deviations, the process standard deviation σ are chosen from $\{\frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \frac{1}{5}\}$.

Notice that from the definition of $C_{pa}(1,3)$, if the distance between a process mean and

the target is larger than the half length of the tolerance interval, then the value of $C_{pa}(1,3)$ will be negative, which is not acceptable. We will exclude these cases from our discussion when this occurs and the total number of computations will be adjusted accordingly.

Based on the above settings, the algorithm for the calculations of rates of consistency is descibed below :

Step 1. Choose level r from {.25, .75, 1, 1.25, 2.75, 4}, two different means of process μ_A, μ_B from $\{LSL, LSL+0.05, LSL+2(0.05), \ldots\ldots, USL-0.05, USL\}$, and the common σ from $\{\frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \frac{1}{5}\}$.

Step 2. Compute percentage of conformances POC_A and POC_B for process A and process B, respectively.

Step 3. For each index, compute the values of process capability indices PCI_A and PCI_B for process A and process B, respectively.

Step 4. Let

$S_1 = \{(\mu_A, \mu_B) | (POC_A > POC_B \text{ and } PCI_A > PCI_B) \text{ or } (POC_A < POC_B \text{ and } PCI_A < PCI_B)\}$.

Let $n(S_1)$ denote the number of elements in S_1. Define

$$\text{rate of consistency} = \frac{n(S_1)}{\text{number of pairs } (\mu_A, \mu_B) \text{ leading to nonnegative index}}$$

for process capability indices C_p, C_{pk}, C_{pm}, C_{pmk}, C''_{pk}, C''_{pm}, C''_{pmk}, S_{pk} and $C_{pa}(u, v)$.

Step 5. Let

$S_2 = \{(\mu_A, \mu_B) | (POC_A > POC_B \text{ and } PCI_A < PCI_B) \text{ or } (POC_A < POC_B \text{ and } PCI_A > PCI_B)\}$.

Let $n(S_2)$ denote the number of elements in S_2. Define

$$\text{rate of consistency} = \frac{n(S_2)}{\text{number of pairs } (\mu_A, \mu_B) \text{ leading to nonnegative index}}$$

for process capability indices C_{pp} , C''_{pp} and $C''_{pp}(s', t)$.

The region of inconsistency consists of elements outside S_1 or S_2 for $r = 0.25$, $\sigma = \frac{1}{2}$ and $r = 4$, $\sigma = \frac{1}{2}$ are given in Appendix A and Appendix B respectively.

The rates of consistency for the indices studied can be found from Table 2 to Table 10 in Appendix C. Averaging the 24 entries in each table, we obtain the average rates of consistency for each index. The results can be read from Table 11 in Appendix C.

4. Major findings

Based on the results obtained from Appdices A, B and C, major findings are listed below.

There are two cases where rate of consistency is 1. (i) When $r = 1$, the rates of consistency are all equal to 1 for all indices studied (see Table 2 to Table 10). (ii) The rate of consistency of S_{pk} is 1 (see Table 5) for all choices of r. It is not surprising since

$$S_{pk} = \frac{1}{3}\Phi^{-1}\{\frac{1}{2}\Phi(\frac{USL - \mu}{\sigma}) + \frac{1}{2}\Phi(\frac{\mu - LSL}{\sigma})\} = \frac{1}{3}\Phi^{-1}\{\frac{1}{2}(1+POC)\}.$$

Thus, large POC produces large S_{pk}.

When $r < 1$ and T is close to LSL, then LSL is more important than USL. In this case, good processes will produce μ_A and μ_B such that

$$(\mu_A, \mu_B) \in S_3 = \{(\mu_A, \mu_B) | LSL \leq \mu_A \leq T, LSL \leq \mu_B \leq T\}.$$

For example, if $r = 0.25$, $\sigma = \frac{1}{2}$ then S_3 is located in the regions of consistency (i.e. the lower left corner outside the region of inconsistency, see Appendix A). Thus, process capability indices and percentage of conformance work consistently in S_3, crucial parameter spaces of processes under investigation.

When $r > 1$ and T is close to USL, then USL is more important than LSL. In this case,

good processes will produce μ_A and μ_B such that

$$(\mu_A, \mu_B) \in S_4 = \{(\mu_A, \mu_B) | T \leq \mu_A \leq USL, T \leq \mu_B \leq USL\}.$$

For instance, when $r = 4$, $\sigma = \frac{1}{2}$ then S_4 is located in the regions of consistency (i.e. the top right corner outside the region of inconsistency, see Appendix B). Thus, process capability indices and percentage of conformance perform consistently in S_4, crucial parameter spaces of processes under investigation.

Consider the parameter space

$$S_5 = \{(\mu_A, \mu_B) | (\mu_A - T)^2 + (\mu_B - T)^2 \leq c^2\}.$$

It is expected to have μ_A and μ_B lie in S_5 for small $c > 0$ since then process means will be close to T. When $r = 0.25$, $\sigma = \frac{1}{2}$, for figures in Appendix A, if $c < T$ then the northeast part (where both μ_A and μ_B are greater than T) of S_5 is located in the region of inconsistency. The other majority parts are located in the region of consistency. When $r = 4$, $\sigma = \frac{1}{2}$, for figures in Appendix B, if $c < T$ then the southwest part (where both μ_A and μ_B are smaller than T) of S_5 is located in the region of inconsistency. The other majority parts are located in the region of consistency. In other words, if S_5 is of major concern, then the majority parts of the parameter space is located in the region of consistency.

It is easily seen from Appendices A and B that in the region of inconsistencies, most inconsistencies occur when the target value lies below or above both process means, respectively. Furthermore, even when both means are almost the same and are both close to the target value T, the inconsistency still exists. Which means either *PCI* or *POC* will lead to the wrong conclusion about the real quality of com- peting processes. The more the difference between two process means, the less the inconsistency.

It is interesting to see from Tables 2, 3, 4, and 9 that the rates of consistency of C''_{pk}, C''_{mk}, C''_{pmk} and C''_{pp} are all the same. This is reasonable since C''_{pk}, C''_{mk}, C''_{pmk} are decreasing with A and A^* and C''_{pp} is incresing with A. Consequently, large values of C''_{pk}, C''_{mk}, C''_{pmk} is equivalent to small value of C''_{pp}.

Among the indices studied, the order of indices listed from strong to weak consistency with *POC* are given as : S_{pk}, C''_{pp} (s,t), $C_{pa}(0,4)$, C''_{pk} , C''_{pm} , C''_{pmk} , C''_{pp} , C_{pp}, C_{pa} (1,3) (see Table 11). The relatively new introduced index C''_{pp} (s,t) shows its potential power by high rates of consistency.

For each index discussed except S_{pk}, the rate of consistency is decreasing with $|r - 1|$. Thus, the level of consistency is decreasing when the level of asymmetric is increasing (see Table 2 to Table 10). This suggests that modifications of *PCI* and *POC* are needed to make them consistent when the level of asymmetric is high.

5. Conclusions

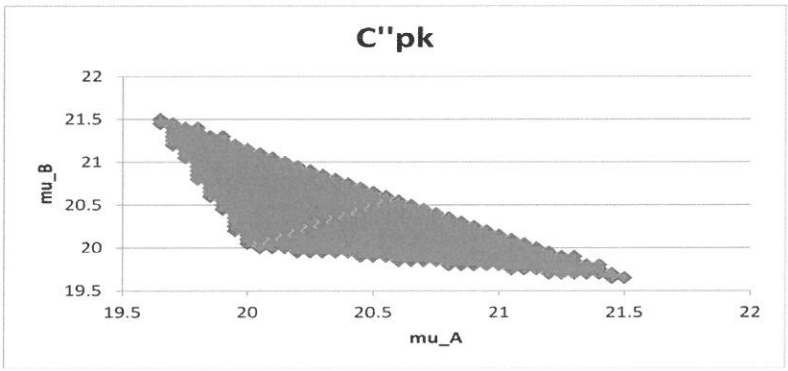
Process capability index (*PCI*) is a successful invention in process capability analysis. Improvements have been made constantly through new proposed indices. The consistency between the new index and the percentage of conformance (*POC*) is often emphasized to support the superiority of the new index.

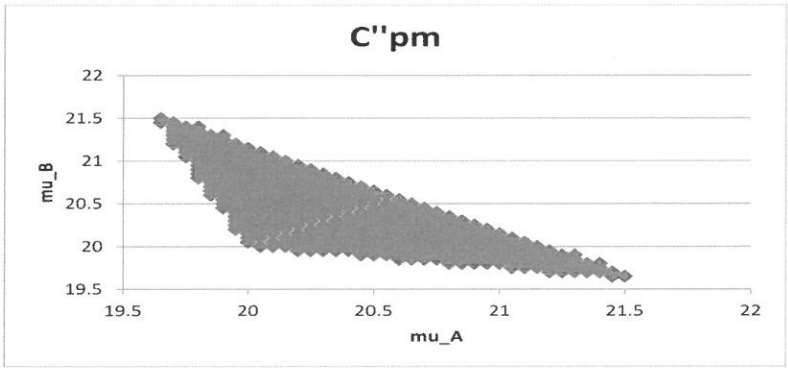
In this paper, we first quantify the level of consistency for nine popular indices. Then, we calculate *PCI* and *POC* under many settings of parameters base which regions of consistency can be obtained. We also discuss properties of regions of consistency for nine popular indices. From Table 2 ～ Table 10, the closer the target value to the end specification limit the worse the consistency between process capability index and percentage of conformance. That is to say, the larger the level of asymmetry the higher the risk of wrong conclusion from process capability index or percentage of conformance.

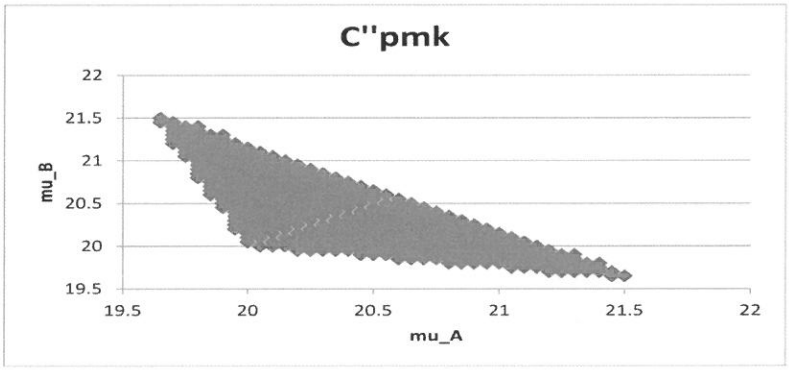
There may be many "conformance comparisons" among competing indices in literature, however, the contribution of this article is giving practitioners an warning that there may be strong inconsistency between percentage of conformance and process capability index. People should not use the value of percentage of conformance blindly to claim that a new index is better than existing indices.

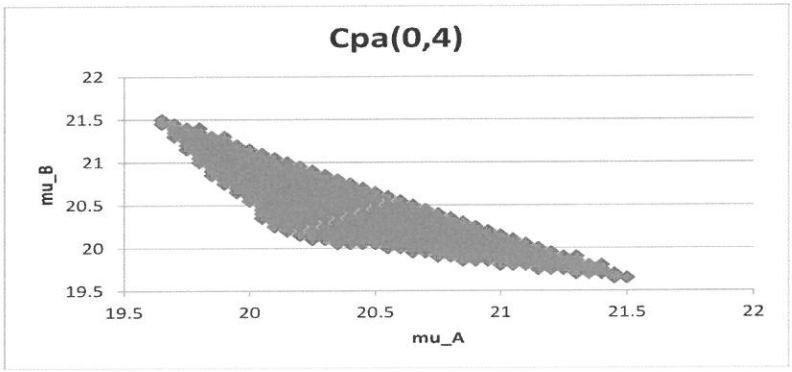
Appendix A

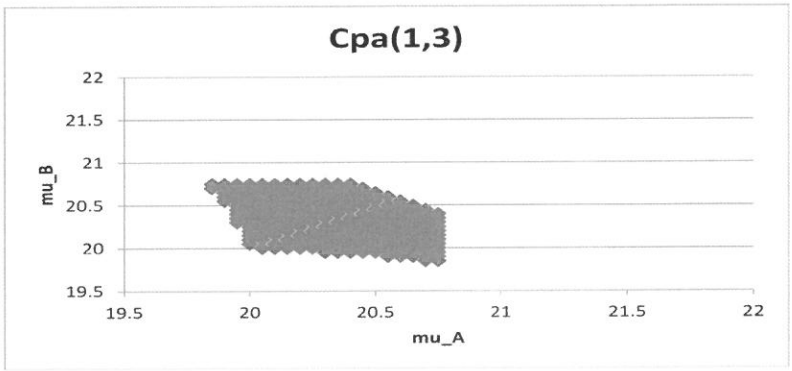
Regions of inconsistency for $r = .25$, $\sigma = \frac{1}{2}$

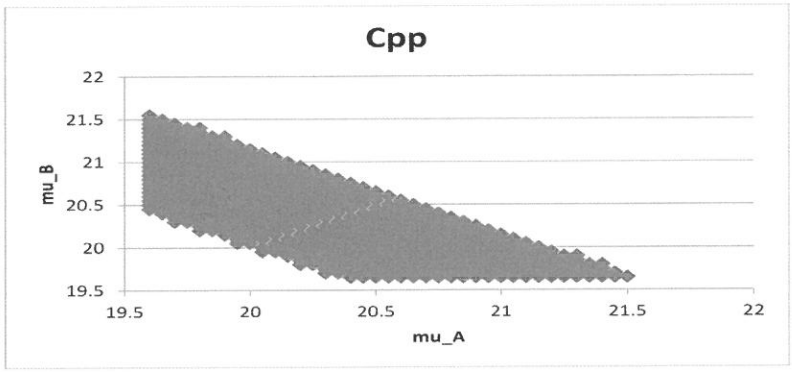


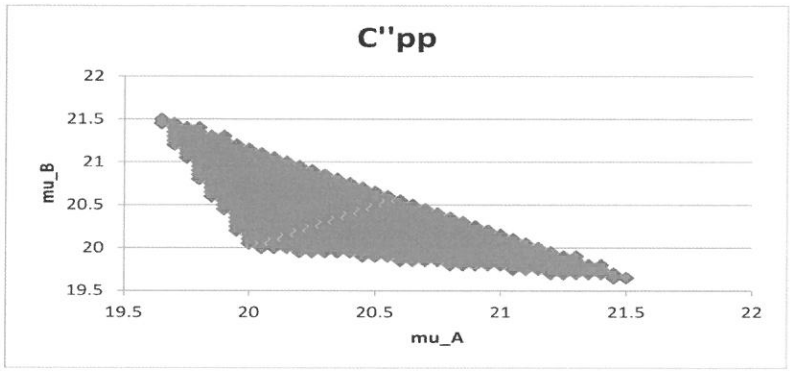


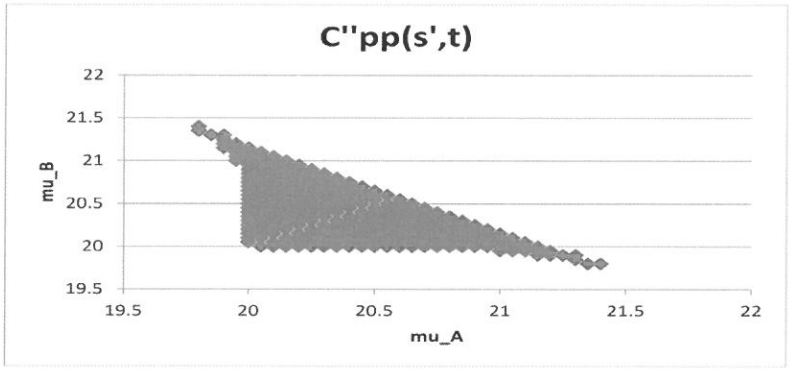



Cpa(0,4)
22
21.5
21
20.5
20
19.5
mu_B
19.5
20
20.5
21
21.5
22
mu_A


Cpa(1,3)
22
21.5
21
20.5
20
19.5
mu_B
19.5
20
20.5
21
21.5
22
mu_A

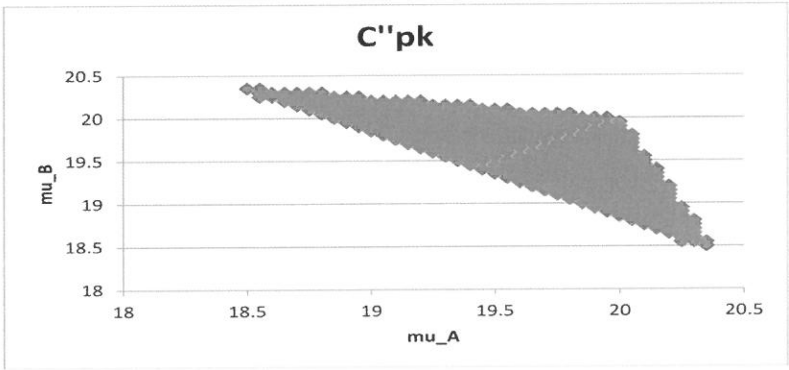

Cpp
22
21.5
21
20.5
20
19.5
mu_B
19.5
20
20.5
21
21.5
22
mu_A

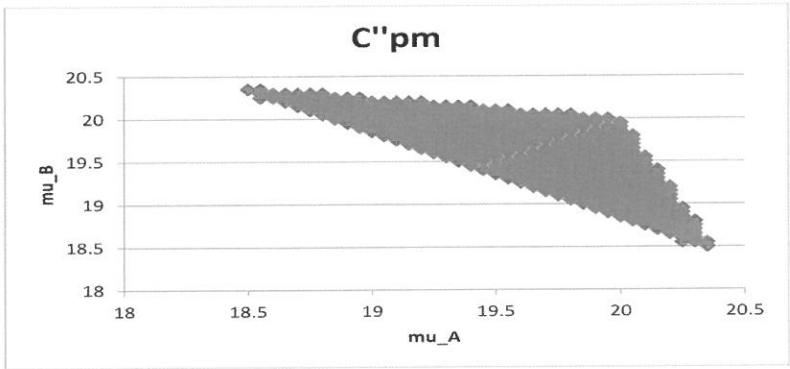

C''pp
22
21.5
21
20.5
20
19.5
mu_B
19.5
20
20.5
21
21.5
22
mu_A

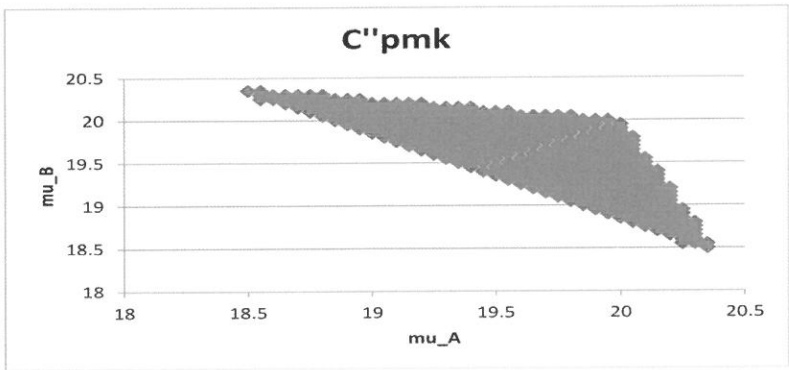

C''pp(s',t)
22
21.5
21
20.5
20
19.5
mu_B
19.5
20
20.5
21
21.5
22
mu_A

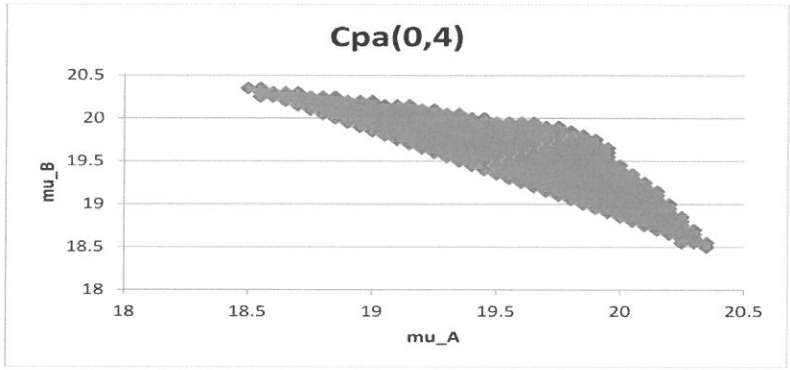
Appendix B

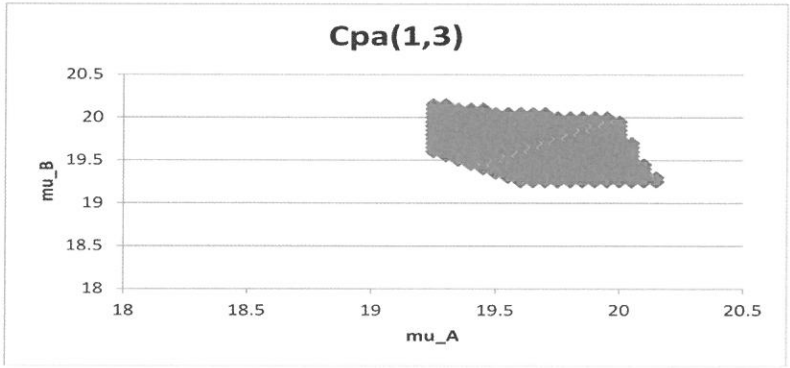
Regions of inconsistency for $r = 4, \sigma = \frac{1}{2}$

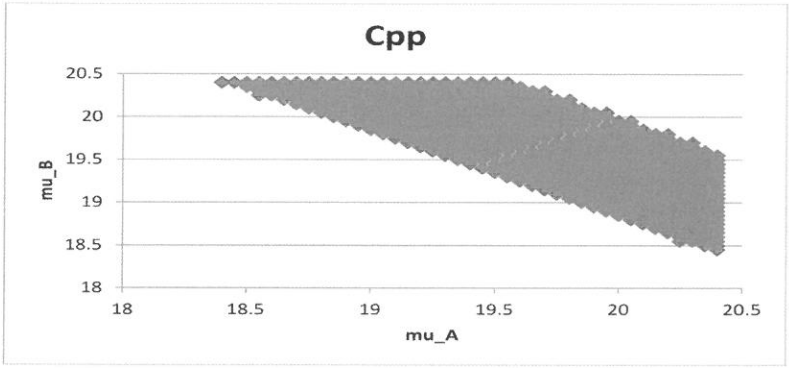


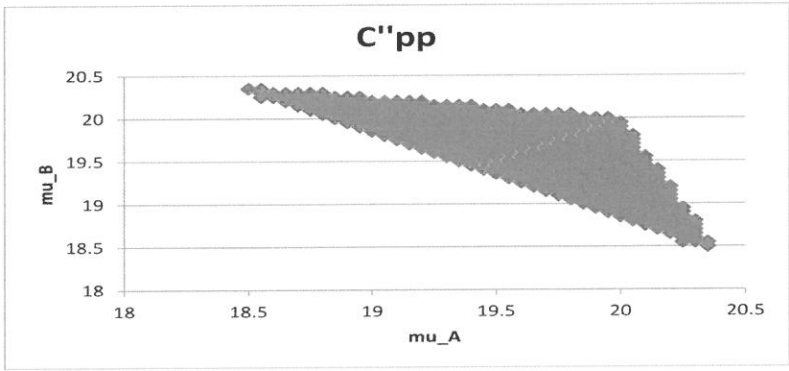


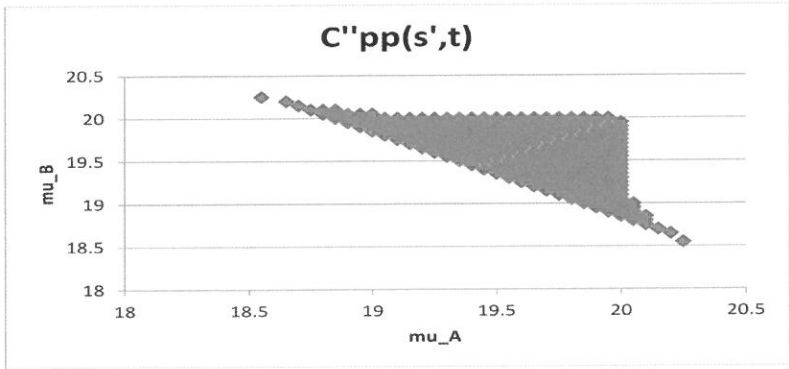



Cpa(0,4)
20.5
20
19.5
19
18.5
18
mu_B
18
18.5
19
19.5
20
20.5
mu_A


Cpa(1,3)
20.5
20
19.5
19
18.5
18
mu_B
18
18.5
19
19.5
20
20.5
mu_A


Cpp
20.5
20
19.5
19
18.5
18
mu_B
18
18.5
19
19.5
20
20.5
mu_A


C''pp
20.5
20
19.5
19
18.5
18
mu_B
18
18.5
19
19.5
20
20.5
mu_A


C''pp(s',t)
20.5
20
19.5
19
18.5
18
mu_B
18
18.5
19
19.5
20
20.5
mu_A

Appendix C

Table 2: Consistency rates between C''_{pk} and *POC*

	r					
σ	0.25	0.75	1	1.25	2.75	4
1/5	0.7243903	0.9439024	1.000000	0.9597561	0.7902439	0.7256098
1/4	0.7256098	0.9451219	1.000000	0.9585366	0.7890244	0.7243903
1/3	0.7243903	0.9439024	1.000000	0.9585366	0.7890244	0.7243903
1/2	0.7231708	0.9426829	1.000000	0.9585366	0.7890244	0.7243903

Table 3: Consistency rates between C''_{pm} and *POC*

	r					
σ	0.25	0.75	1	1.25	2.75	4
1/5	0.7243903	0.9439024	1.000000	0.9597561	0.7902439	0.7256098
1/4	0.7256098	0.9451219	1.000000	0.9585366	0.7890244	0.7243903
1/3	0.7243903	0.9439024	1.000000	0.9585366	0.7890244	0.7243903
1/2	0.7231708	0.9426829	1.000000	0.9585366	0.7890244	0.7243903

Table 4: Consistency rates between C''_{pmk} and *POC*

	r					
σ	0.25	0.75	1	1.25	2.75	4
1/5	0.7243903	0.9439024	1.000000	0.9597561	0.7902439	0.7256098
1/4	0.7256098	0.9451219	1.000000	0.9585366	0.7890244	0.7243903
1/3	0.7243903	0.9439024	1.000000	0.9585366	0.7890244	0.7243903
1/2	0.7231708	0.9426829	1.000000	0.9585366	0.7890244	0.7243903

Table 5: Consistency rates between S_{pk} and *POC*

	r					
σ	0.25	0.75	1	1.25	2.75	4
1/5	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
1/4	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
1/3	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
1/2	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

Table 6: Consistency rates between C_{pa} (0,4) and *POC*

	r					
σ	0.25	0.75	1	1.25	2.75	4
1/5	0.7268293	0.9463415	1.000000	0.9621951	0.7914634	0.7280488
1/4	0.7329268	0.9487805	1.000000	0.9609756	0.7939025	0.7317073
1/3	0.7451220	0.9500000	1.000000	0.9634146	0.8036585	0.7451220
1/2	0.7780488	0.9560975	1.000000	0.9707317	0.8280488	0.7792683

Table 7: Consistency rates between C_{pa} (1,3) and *POC*

	r					
σ	0.25	0.75	1	1.25	2.75	4
1/5	0.3300000	0.9463415	1.000000	0.8800000	0.4350000	0.3300000
1/4	0.3350000	0.8350000	1.000000	0.8800000	0.4350000	0.3350000
1/3	0.3500000	0.8400000	1.000000	0.8750000	0.4500000	0.3500000
1/2	0.3750000	0.8600000	1.000000	0.9050000	0.4800000	0.3750000

Table 8: Consistency rates between C_{pp} and *POC*

	r					
σ	0.25	0.75	1	1.25	2.75	4
1/5	0.6091464	0.8350000	1.000000	0.9073171	0.6597561	0.5963415
1/4	0.6103659	0.8884146	1.000000	0.9060975	0.6585366	0.5951220
1/3	0.6091464	0.8871951	1.000000	0.9060975	0.6585366	0.5951220
1/2	0.6079268	0.8859756	1.000000	0.9054878	0.6579269	0.5945122

Table 9: Consistency rates between C''_{pp} and *POC*

	r					
σ	0.25	0.75	1	1.25	2.75	4
1/5	0.7243903	0.8871951	1.000000	0.9597561	0.7902439	0.7256098
1/4	0.7256098	0.9451219	1.000000	0.9585366	0.7890244	0.7243903
1/3	0.7243903	0.9439024	1.000000	0.9585366	0.7890244	0.7243903
1/2	0.7231708	0.9426829	1.000000	0.9585366	0.7890244	0.7243903

Table 10: Consistency rates between $C''_{pp}(s,t)$ and *POC*

	r					
σ	0.25	0.75	1	1.25	2.75	4
1/5	0.8109756	0.9439024	1.000000	0.9829268	0.8768293	0.8121951
1/4	0.8121951	0.9743903	1.000000	0.9817073	0.8756098	0.8109756
1/3	0.8109756	0.9731708	1.000000	0.9817073	0.8756098	0.8109756
1/2	0.8097561	0.9719512	1.000000	0.9817073	0.8756098	0.8109756

Table 11: Average consistency rates

PCI	S_{pk}	$C''_{pp}(s,t)$	$C_{pa}(0,4)$	C''_{pk}	C''_{pm}
Average	1	.9077977	.7751524	.8568597	.8568597
PCI	C''_{pmk}	C''_{pp}	C_{pp}	$C_{pa}(1,3)$	
Average	.8568597	.8568597	.7739176	.6500559	

Reference

1. Boyles, R. A. (1994) . "Process capability with asymmetric tolerances". *Communications in Statistics - Simulation and Computation* . 23(3), 615-653.
2. Chan, L. K. , Cheng, S. W. and Spiring, F. A., (1988). "A new measure of process capability index: C_{pm}". *Journal of Quality Technology*. 20, 162-175.
3. Chen, K.S. (1997). "A New Process Capability Index for Asymetric Tolerances". *Journalof the Chinese Institute of Industrial Engineers*. 14, 355-362.
4. Chen, K.S. (1998). "Incapability Index with Asymetric Tolerances". *Statistica Sinica*. 8, 253-262.
5. Chen, K.S. , Pearn, W. L., and Lin, P. C (1999). "A new generalization of C_{pm} for processes with asymmetric tolerances". *International Journal of Reliabilty, Quality and Safety Engineering* . 6(4), 383-398.
6. Greenwich, M. and Jahr-Schaffrath, B.L. (1995). "A process incapability index". *Inernational Journal of Quality Reliability Management*. 12, 58-71.
7. Juran J.M., Gryna F.M., Bingham R.S. Jr (1974). *Quality Control Handbook*.. McGraw-Hill, New York.
8. Kane V.E. (1986). "Process capability indices". Jouranal of Quality Technology 18(1):41-52.
9. Kotz, S., Johnson, N. L. (1993). *Process Capability Indices*. London: Chapman and Hall.
10. Kotz, S., Lovelace, C. R. (1998). *Process Capability Indices in Theory and Practice*. London: Oxford University Press Inc.
11. Kotz, S. and Johnson, N. L. (2002). "Process capability indices - a review, 1992-2000". *Journal of Quality Technology*, 34(1), 1-19.
12. Mohammad A. (2012). "Fuzzy Process capability indices: a review". *World Applied Sciences Journal* 16(12), 1734-1740.
13. Mayers, R.H. and Montgomery, D. C. (1995). *Response surface methodology*. New York: John Wiley.
14. Nandini, D. and Saurav, D. P. (2013). "Multivariate Process Capability Index: A Review and Some Results". *Economic Quality Control* . 28(2), 16.
15. Nandini, D. and Dwivedi, P.S. (2013). "Multivariate Process Capability Index: A Review and Some Results". *Economic Quality Control* . 28(2), 151-166.
16. Palmer, K. and Tsui, K.L. (1999). "A Review and Inerpretations of Process Capability Indices' . *Annals of Operations Research* , 87, 31-47.
17. Pan, J.N. and Lee C. Y. (2009). "Development of a new process incapability index with an application to the evaluation of manufacturing risk". *Communications in Statistics:*

Theory & Methods 38(12):1133-1153.

18. Pearn W. L., Kotz S., Johnson N. L. (1992). "Distributional and inferen- tial properties of process capability indices". *Journal of Quality Technology* 24(4):216-233.
19. Pearn W. L., Kotz S. (2006). *Encyclopedia And Handbook of Process Capability Indices: A Comprehensive Exposition of Quality Control Measures*. Series on Quality, Reliability, and Engineering Statistics, Volume 12.
20. Pearn, W. L. and CHEN, K. S. (1998). "New generalization of process capability index C_{pk}". *Journal of Applied Statistics*, Vol. 25, No. 6, 801-810.
21. Pearn, W. L., Lin, P. C. and Chen K. S. (1999). "On the genearalizations of the capability index C_{pmk} for asymmetric tolearnaces". *Far East Jouranl of Theoretical Statistics*. 3. 49-66.
22. Spiring, F. Cheng, S. Yeung A. and Leung B. (2002). "Comment on Process cpability indices– a review, 1992-2000". *Journal of Quality Technology*. 34(1): 23-27.
23. Sullivan, L. P. (1984). "Reducing variablility, a new approach to quality". *Quality Progress*. 17, 15-21.
24. Sullivan, L.P. (1985). Letters. *Quality Progress*. 18, 7 - 8.
25. Spiring, F., Leung B., Cheng, S., and Yeung A.(2003). "A bibliography of Process cpability papers". *Quality and Reliability Engineering International*. 19, 445-460) (DOI:10.1002/qre.538).
26. Tang, L. C. and Than, S. E. (1999). "Computing process capability indices for non-normal data: a review and comparative study". *Quality and Reliability Engineering International*, 15(5). 339-353.
27. Vännman K. (1997). "A general class of capability indices in the case of asym- metric tolerances". *Communications in Statistics: Theory and Methods*, 26:8. 2049-2072.
28. Yum, B.J and Kim K.W. (2010). "A bibliography of the literature on process ca- pability indices:2000-2009". *Quality and Reliability Engineering International*. 27(3): 251-268 (DOI:10.1002/qre.111).

Recevied Oct 8, 2014
Revised Mar 3, 2015
Accepted Mar 10, 2015

製程能力指標與一致性百分比之一致性研究

陳思勉　許玉生*

輔仁大學數學系
*中央大學數學系

摘　要

製程能力指標被廣泛的運用在不同的工業應用上，近幾年不斷地有新的指標被提出。文獻上，當一個新指標被提出時，作者經常以透過該新指標判斷不同製程優劣所得的結果與透過一致性百分比判斷所得之結論相同來突顯該新指標優於既有之指標。然而二種方式所得結果一致並不保證二者所提供的判斷結果是正確的。本文將就此問題討論並藉此研究提醒製程能力指標之使用者在運用時一定要謹慎，以避免做出錯誤的決策。

關鍵字：對稱性公差；非對性公差；一致性；一致性百分比；製程能力指標。

Generalized-Impedance Converter Using Differential Voltage Current Conveyors

Yung-Chang Yin and Hong-Yu Liu

Department of Electrical Engineering, Fu-Jen Catholic University
New Taipei City, Taiwan, R.O.C. 24205

Abstract

A generalized-impedance converter (GIC) circuit using two Differential Voltage Current Conveyors (DVCCs) and three passive components is suggested. The main features of the GIC are its ability to realize synthetic floating inductors, floating capacitors and floating frequency-dependent negative resistors (FDNR) with grounded passive components. The use of the grounded components simplifies the etching process for monolithic or hybrid fabrication. Moreover, the proposed GIC can be used to construct the simulation of higher-order impedance. The inductor in the LCR passive filter circuits can be replaced by the proposed simulation floating inductor from the GIC for the application. Thus, the advantages of these new active RC filters without an inductor can include low component sensitivities and utilize the extensive capability of the primary LCR filter design. Further, the sensitivities of the active inductance, active capacitance and FDNR of the proposed GIC circuit were derived. In order to show the performance of the proposed GIC, a simple biquad filter is experimentally demonstrated. The experimental result of a passive RLC bandpass filter confirming the theory is included.

Key words: differential-voltage current conveyor, inductor simulators, analog circuit design, active filters

1. Introduction

Many active tuned filters have been widely used in signal processing. Several circuit configurations for current-mode filters, voltage-mode filters, sinusoidal oscillators and admittance function simulators using the second-generation current conveyors (CCII) or the four-terminal active current conveyors as active elements have been reported in the literatures [1]~[14]. A recent publication introduced a circuit building block termed differential voltage current conveyor (DVCC). It is a universal building block that was constructed by Pal [15] and then was developed and realized in CMOS technology by Elwan and Soliman[16]. The DVCC has become very popular because of its high signal bandwidth, great linearity and large dynamic range, so it has been used in many analogy signal processing applications for general all-finite linear circuits and applied in the areas of oscillator design, active filters, and cancellation of parasitic elements [15]~[22]. For example: Elwan and Soliman employed two DVCCs and two capacitors to synthesize current-mode bandpass and lowpass filters [16]. Ibrahim, Minaei and Kuntman used three DVCCs and six passive elements to construct current-mode Kerwin uelsman ewcomb biquad [20]. Yin employed two DVCCs and some passive components to construct bandpass, lowpass, notch, allpass and highpass filters [21]. Yin used a DVCC and some passive elements to realize lowpass, bandpass and highpass filters [22]~[25]. Chien used some DVCCs to construct the square and triangular wave generators[26].

Simulated admittance can be applied in the area such as oscillator design, active filters and cancellation of parasitic elements. It is a problem to implement inductors, large-valued capacitors and negative resistors in integrated circuit technology. For this reason, Yin used two DVCCs as active elements and three passive elements to synthesis the one-order grounded inductance, grounded capacitance and negative resistor simulation circuit [27] [28]. However, the DVCC-based higher-order floating inductance and floating capacitance simulation circuit have been not presented. In this study, a new circuit configuration for the simulation of floating inductance, floating capacitance and FDNR using two DVCCs and three passive components is proposed. The proposed circuit employing grounded passive components and requiring no passive component matching conditions simplifies the etching process for monolithic or hybrid fabrication. Meanwhile, the proposed circuit can be also applicable to higher-order impedance

simulation using additional DVCCs. On the other hand, the floating inductor is frequently required for the design of analogue integration circuit building blocks. The area of active filter design and inductor simulation has attracted considerable interest, because the advantages of designing active filters by simulating the inductor of a passive LCR realization of the filter include low component sensitivities and the ability to utilize the extensive knowledge of LCR filter design. Therefore, the main contribution of this paper is to present DVCC-based circuits of simulating floating inductor. Finally, a passive RLC filter using the proposed simulating inductor is experimentally verified.

2. PROPOSED CIRCUIT

The circuit symbol for a DVCC is shown in Fig.1. The port relations of a DVCC can be characterized as $V_x = V_{y1} - V_{y2}$, $I_{Z+} = I_x$, $I_{Z-} = -I_x$ and $I_{y1} = I_{y2} = 0$. The '+' and '-' signs of the current i_z denote the non-inverting and inverting, respectively.

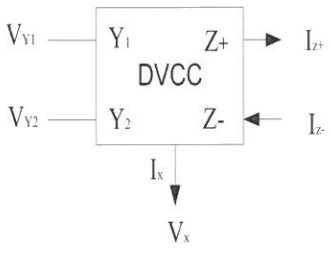


Fig.1. A DVCC symbol

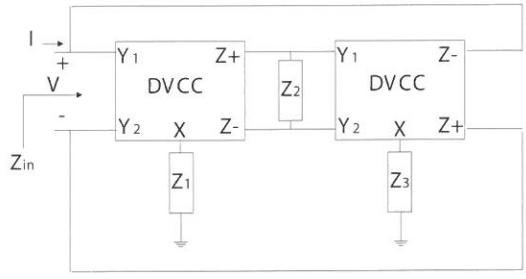


Fig.2: The proposed generalized impedance converter circuit.

The GIC circuit is shown in Fig.2, where $Z_1 \sim Z_3$ are impedances. By routine analysis, one get, for the circuit of Fig.2, the input impedance of the GIC equation is given by

$$Z_{in} = \frac{Z_1 Z_3}{Z_2} \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots (1)$$

By choosing suitable passive components from the impedances Z_1, Z_2 and Z_3 in equation (1),

the floating inductors, floating capacitors and floating frequency dependent negative resistors can be obtained as follows:

(1) If $Z_1 = R_1$, $Z_3 = R_3$, and $Z_2 = (1/sC_2)$, are taken, the impedance of the floating inductor is obtained as

$$Z_{in} = sL_{eq} = s\, C_2 R_1 R_3 \quad \cdots\cdots (2)$$

where $L_{eq} = C_2 R_1 R_3$. The sensitivities of the active inductance are

$$S_{C_2}^{L_{eg}} = S_{R_1}^{L_{eg}} = S_{R_3}^{L_{eg}} = 1$$

(2) If $Z_1 = (1/sC_1)$, $Z_3 = R_3$, and $Z_2 = R_2$ are taken, the impedance of the floating capacitor is obtained by

$$Z_{in} = \frac{1}{sC_{eg}} = \frac{R_3}{sC_1R_2} \quad \cdots\cdots (3)$$

where $C_{eq} = C_1R_2/R_3$. The sensitivities of the active capacitance are

$$S_{C_1}^{C_{eg}} = S_{R_2}^{C_{eg}} = 1,\ S_{R_3}^{C_{eg}} = -1$$

(3) If $Z_3 = (1/sC_3)$, $Z_1 = R_1$, and $Z_2 = R_2$ are taken, the impedance of the floating capacitor is obtained as

$$Z_{in} = \frac{1}{sC_{eg}} = \frac{R_1}{sC_3R_2} \quad \cdots\cdots (4)$$

where $C_{eq} = C_3R_2/R_1$. The sensitivities of the active capacitance are

$$S_{C_3}^{C_{eg}} = S_{R_2}^{C_{eg}} = 1,\ S_{R_1}^{C_{eg}} = -1$$

(4) If $Z_3 = (1/sC_3)$, $Z_1 = (1/sC_1)$, and $Z_2 = R_2$, are taken, the GIC circuit realizes a floating FDNR given by

$$Z_{in} = \frac{1}{sD_{eg}} = \frac{1}{s^2C_1C_3R_2} \quad \cdots\cdots (5)$$

where $D_{eq} = C_3C_1R_2$. The sensitivities of the active capacitance are

$$S_{C_1}^{D_{eg}} = S_{C_3}^{D_{eg}} = S_{R_2}^{D_{eg}} = 1$$

The theoretical characteristic of the FDNR, shown in Fig.2, is given for $Z_3 = (1/sC_3)$, $Z_1 = (1/sC_1)$, and $Z_2 = R_2$. Let $C_1 = 0.1mF$, $C_3 = 1\mu F$ and $R_2 = 100\Omega$. The Matlab analysis is proven to be effective when the optimum curve of the FDNR circuit, shown in Fig.3, is applied. It is

observed from the equation (5) and Fig. 3 that the impedance of the FDNR decreases with increasing frequency.

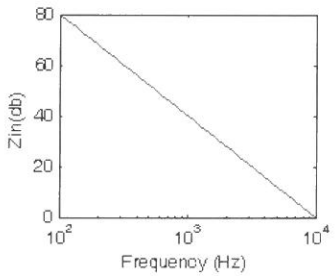


Fig.3: The ideal curve of the FDNR for C_1=0.1mF, C_3=1 μF and R_2=100Ω.

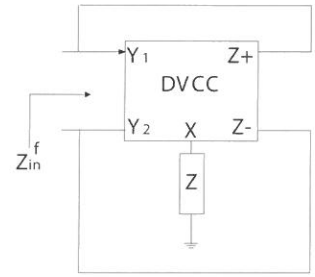


Fig.4: The floating input impedance Z_{in}^{f} is equal to the grounded impedance Z.

In Fig.4, the input impedance Z_{in}^{f} is equal to Z. After replacing Z_2 shown in Fig.2 into Z_{in}^{f}, the GIC circuit contains three DVCCs and three passive components, all of which are grounded. On the other hand, in Fig.2, after replacing Z_2 into Z_{in}, the input impedance Z_{in} ' of the GIC circuit shown in Fig.4 will be equal to below:

$$Z_{in}' = \frac{Z_1 Z_3 Z_5}{Z_2 Z_4} \quad \cdots\cdots(6)$$

Replace Z_5 in Fig.3 into Z_{in}^{f} in Fig.2, then the GIC circuit contains five DVCCs and five passive components, all of which are also grounded. Given the above statement, by replacing Z_5 in Fig.4 into Z_{in}^{f} in Fig.2, the higher-order impedance simulation circuit can be also obtained. The GIC circuit with grounded passive components is constructed. Clearly, by using additional DVCCs, this proposed GIC can be easily expanded to higher-order impedance simulation circuits, which is one of the important properties of the proposed circuit.

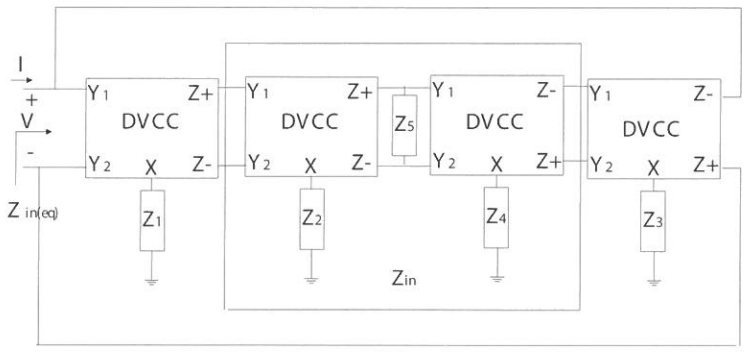


Fig.5: The new GIC circuit construction

3. APPLICATION OF THE PROPOSED SIMULATOR CIRCUIT

The DVCC is implemented by three ICAD844s, three IC op-amps and five resistors. From the Fig-2, if $Z_1=R_1$, $Z_3=R_3$, and $Z_2=(1/sC_2)$ are taken, the impedance of the floating inductor is obtained. The proposed floating inductor simulated was experimentally tested using $R_1 = R_3 = 1K\Omega$, $C_2 = 1\mu F$ and AD844s, so this is equivalent to an inductor with $L_{eq} = 1H$. Then, this simulation floating-inductor can be used in the realization of the series RLC passive bandpass filter as shown in Fig.6 and the transfer function has a biquadratic bandpass characteristic with

$$\frac{v_{out}}{v_{in}} = \frac{(\frac{R}{L_{eg}})s}{s^2 + s(\frac{R}{L_{eg}}) + \frac{1}{L_{eg}C}} \quad \cdots\cdots(7)$$

the central frequency: $\omega_o = (\frac{1}{L_{eg}C})^{1/2}$ $\cdots\cdots(8)$

the quality factor: $\vartheta = R(\frac{L_{eg}}{C})^{1/2}$ $\cdots\cdots(9)$

The sensitivities of ω_o and ϑ according to passive components are:

$$s_{C_1}^{\omega_o} = s_{R_1}^{\omega_o} = s_{R_2}^{\omega_o} = s_C^{\omega_o} = s_C^{\vartheta_o} = -\frac{1}{2},\ s_{R_1}^{\vartheta_o} = s_{R_2}^{\vartheta_o} = s_{C_1}^{\vartheta_o} = \frac{1}{2},\ s_R^{\vartheta_o} = 1$$

all of which are small.

In Fig.7, a second-order bandpass filter was constructed with $R = 1K\Omega$, $C = 1\mu F$ and $L_{eq} = 1H$. The Matlab has simulated the ideal curves of the bandpass filter. The measured values were found using a Hewlett Packard network/spectrum analyzer 4195A. The resonance frequency was monitored and measured and the corresponding inductance value was calculated and compared with the theoretical value calculated using equation (2). The experimental results for the gain and phase responses shown in Fig.8 (a) (b) demonstrate similarity between theoretical calculation and actual measurement. There is a high correlation between the theoretical analyses and the measured results with only minor errors due to the use of passive elements.

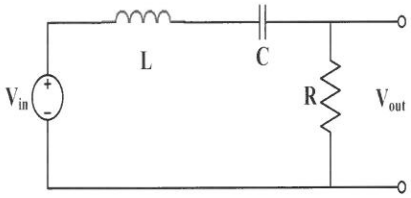


Fig.6 The prototype passive series RLCbandpass filter.

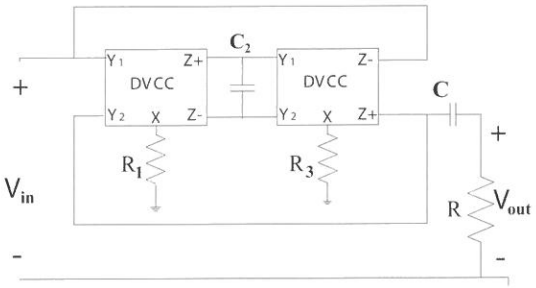
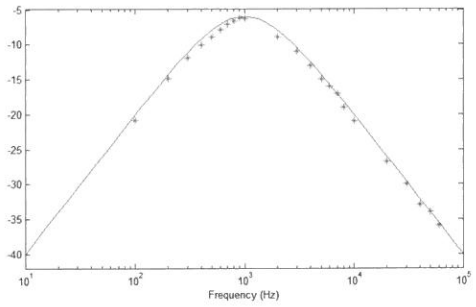
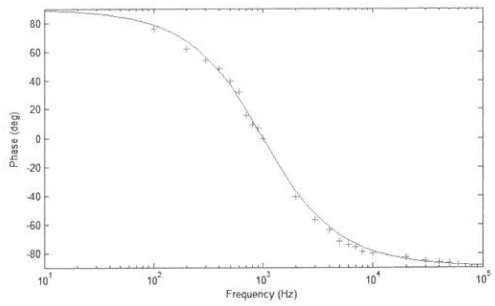


Fig.7 The bandpass filter circuit used to test the inductor realized using the circuit of Fig. 4



(a). The gain response curve of the banspass filter using simulated inductor.



(b). The phase response curve of the banspass filter using simulated inductor.

Fig.8 (a): The gain response curve of the banspass filter using simulated inductor.
(b): The phase response curve of the banspass filter using simulated inductor.
* : Experimental result for banspass gain
+ : Experimental result for bandpass phase
- : Ideal curve

4. CONCLUSION:

Using the concept of the DVCC, a GIC circuit has been presented. The proposed circuit can realize synthetic floating inductors, floating capacitors and floating frequency-dependent negative resistors. By using additional the DVCCs, the proposed GIC circuit can be also

expanded to higher-order impedance simulation circuit. Owing to employing grounded components, the GIC is suited for IC implementation. The advantages of RC active filter transferred by the proposed circuit include low component sensitivities and the ability to utilize the extensive capability of the LCR filter design. Therefore, the experimental results on floating inductor utilizing a passive LCR bandpass filter confirmed the theoretical analysis. The DVCC demonstrates itself as an adaptive active element.

Reference

1. Wilson. B, "Recent Developments in Current Conveyors and Current-Mode Circuits", IEE Proc-G, vol. 137, no. 2, pp.63-77, 1990.
2. Rober G.W. and Sedra A.S., "All Current-Mode Frequency Selective Circuits", Electron. Lett., Vol.25, pp.759-761, 1989.
3. Toumazou C. and Lidgey E.J., "Universal active filter using current conveyous", Electron. Lett.,vol. 22, pp.662-664, 1986.
4. Sun Y. and Fidler J.K., "Versatile active biquad based on second-generation current conveyors", Int. J Electron., vol. 76, pp91-98, 1994.
5. Singh V.K.and Senani R., "New multifunction active configuration employing current conveyors", Electron. Lett., vol. 26, pp.1814-1816, 1990.
6. Robert G.W. and Sedra A.S., "A general class of current amplifier-based biquadratic filter circuits", IEEE Transactions on Circuits and Systems., vol. 39, pp.257-263, 1992.
7. Chang C.M., "Universal active current filters using single second-generation current conveyors", Electron. Lett., vol. 27, no.18, pp.1614-1617, 1991.
8. Chang C.M. and Chen P.C., "Universal active current filter with three inputs and one output using current conveyors", Int. J Electron., vol. 71, no.5, pp.817-819, 1991.
9. Senani R., "New current-mode biquad filter", Int. J Electron., vol. 73, no.4, pp.735-742, 1992.
10. Yin Y.C., "Current-Mode Biquad Using Two CFCCIIps", Fu Jen Studies: Science and Engineering., vol. 3, no.4, pp.367-370, 2003.
11. Yin Y.C., Liou Y.C., "Realization of Current-Mode Highpass Lowpass and Bandpass Biquad Filters using Single CFCCIIp", Fu Jen Studies: Science and Engineering, No.38, pp.89-99, 2004.
12. Yin Y.C., "Realization of Current-Mode Notch and Allpass Filters using Single CFCCII", Fu Jen Studies: Science and Engineering, No.39, pp.11-22, 2005.
13. Gues E.O. and Anday F., "Realization of current-mode universal filter using CFCCIIps", Electron. Lett., vol. 32, pp.1081-1082 , 1996.

14. Chang C.M.and Tu S.H., "Universal current-mode filters employing CFCCIIps" , Int. J Electron., vol. 85, no.6, pp.749-754, 1998.
15. Pal, K., "Modified current conveyors and their applications" , Microelectronics Journal, vol.20, no.4, pp.37-40, 1989.
16. Elwan H.O. and Soliman A.M., "Novel CMOS differential voltage current conveyor and its applications" , IEE Proceeding-Circuits Devices System, No.144, pp.195-200, 1997.
17. Bialokow, M., and Newcomb, R. W., "Generation of all finite linear circuits using the integrated DVCCS" , IEEE Trans., CT-18, pp.733-736, 1971.
18. Bialokow, M., Sienko, W., and Newcomb, R. W., "Active synthesis using the DVCCS/DVCVS" , Int. J. Circuit Theory & Appl., vol.2, pp.23-38. , 1974
19. Nandi. R., "New Ideal Active Inductance and Frequency-Dependent Negative Resistance using D.V.C.C.S./D.V.C.V.S.: Applications in Sinusoidal-Oscillator Realization" , Electron. Lett., vol.14, pp. 551-553, 1978.
20. Muhammed A. Ibrahim, Shahram Minaei and Hakan Kuntman, "A 22.5 MHz current-mode KHN-biquad differential voltage current conveyor and grounded passive elements" , International Journal of Electronics and Communications, pp.311-318, 2004.
21. Yin Y.C., "Realization of Current-Mode Filters Using Two Differential Voltage Current Conveyors" , 2010 Intelligent Living Technology Conference, pp.103-107, 2010.
22. Yin Y.C., "Current-Mode Multifunction Filters using Single Differential Voltage Current Conveyor" , 2010 Workshop on Consumer Electronics Conference, pp.703-707, 2010.
23. Shahram Minaei and Erkan Yuce, "Novel Voltage-Mode All-Pass Filter Based on Using DVCCs" ,Circuit System Signal Procss.,Vol.29, pp.391-402, 2010.
24. Horng J.W., "Lossless inductance simulation and voltage-mode universal biquadratic filter with one input and five outputs using DVCCs" , Analogy Integral Circuit Signal Proceeding, pp.407-413, 2010.
25. Y.C.Yin, "Novel Realization Filter employing Single Differential Voltage Current Conveyor" , Fu Jen Studies: Science and Engineering, No.45, pp.61-70, May 2012.
26. Chien H.C., "Voltage-Controlled dual slop operation square/triangular wave generator and its application as a dual mode operation pulse width modulator employing differential voltage current conveyors" , Microelectronics Journal, Vol.43, Issue 12, pp.962-974, May 2012
27. Y.C.Yin, "Grounded-inductor employing differential-voltage current conveyors" , Fu Jen Studies: Science and Engineering, No.45, pp.53-60, May 2012.
28. Y.C.Yin, "Generalized Active Immittance Simulator using Differential Voltage Current Conveyors" , Fu Jen Studies: Science and Engineering, No.46, pp.115-126, May 2013.

Recevied Oct 8, 2014
Revised Mar 3, 2015
Accepted Mar 10, 2015

使用差動電壓電流傳輸器合成一般化阻抗轉換器

鄞永昌　劉鴻裕

輔仁大學電機工程學系

摘　要

本文提出使用兩個差動電壓電流傳輸器當主動元件，配合三個接地式被動元件，合成一般化阻抗轉換器。此一般化阻抗轉換器，有許多利益。其一：可以合成浮接式電感；其二：可以合成浮接式電容；其三：可以合成相依頻率負阻抗電阻；其四：可以合成出更高階的一般化阻抗轉換器。由於，被動元件為接地式，所以大大地簡化了積體電路的製成。另一方面，電感模擬電路可以用來取代現有被動濾波器的電感值，使其可以被製作成積體電路。所以，這些被動濾波器轉換成主動濾波器之電路，仍可繼續保有原先被動濾波器既有之優點。最後，本文以既有的串接式二階被動帶通濾波器，應用本文所提出的電感模擬電路取代其電感值，以驗證本文之理論預測。

關鍵字：差動電壓電流傳輸器，電感模擬器，類比電路設計，主動濾波器。

A Note on the Number of Odd Dominating Sets in a Tree

Hong-Min Shaw

Department of Mathematics, Fu-Jen Catholic University
New Taipei City, Taiwan, R.O.C. 24205

Abstract

An odd dominating set of a graph is a dominating set that dominates each vertex of the graph an odd number of times. Sutner showed that every graph contains an odd dominating set. Galvin proposed a graphical algorithm to construct a family of odd dominating sets of a given tree. Later Chen *et al.* proved that every odd dominating set of a tree lies in the family that is constructible by Galvin Algorithm. An inductive proof of this result is proposed in this note.

Key words: odd dominating set, Galvin Algorithm

E-mail: hmshaw@math.fju.edu.tw

1. Odd dominating sets

Let $G = (V, E)$ be a finite simple undirected graph. An odd dominating set (od-set in short) of a graph G is a subset of vertices C so that $| N[v] \cap C |$ is odd for every vertex v of G; where $N[v]$ denotes the closed neighborhood of v.

For complete graph K_n, $N[v] = K_n$ for every vertex v, so $OD(K_n) = \{C : | C |$ is odd $\}$. For an od-set C of a cycle C_n, $| N[v] \cap C | \in \{1,3\}$ for every vertex v. If $| N[w] \cap C | = 3$ for some vertex w, then $| N[v] \cap C | = 3$ for every vertex v; thus $C = V(C_n)$. If $| N[u] \cap C | = 1$ for some vertex u, then $| N[v] \cap C | = 1$ for every vertex v; thus n is a multiple of 3 and there are three od-sets of this form.

Denote the set of all od-sets of a graph G by $OD(G)$. Sutner [5] showed that $OD(G) \neq \emptyset$ for every graph G in matrix terminology; from the fact that $OD(G)$ corresponds to a subspace over Z_2, consequently $| OD(G) |$ is a power of 2. Eriksson *et al.* [2] proved $OD(G) \neq \emptyset$ for every graph G in graph terminology. In terms of matrices certain algorithms to construct an od-set (or all od-sets) of a graph are known (cf. [4]). It seems natural to seek for algorithms, in terms of graphs, to do the same job. Galvin [3] gave a graphical algorithm to construct an od-set of a tree. In [1] Chen *et al.* showed that in fact Galvin Algorithm generates all od-sets of a tree We recall Galvin Algorithm in the next section and propose an inductive proof, in the last section, of that Galvin Algorithm produces all od-sets of a tree.

2. Galvin Algorithm

Given a tree $T = (V, E)$, pick up an arbitrary vertex z, and consider the rooted tree T_z (rooted at z). We call a vertex in a rooted tree a node. For a node v in a rooted tree, its parent and children are denoted by $p(v)$ and $K(v)$ respectively.

Galvin Algorithm [3] takes T_z as the input and consists of two phases: bottom-up and top-down.

The bottom-up phase divides the nodes of T_z into three types (original name is parenthesized): out-node (outcast), odd-node (oddball), and even-node (rebel); denoted by Q, P, and R respectively. The classification goes, level by level, from the bottom up as follows.

BU1. All leaves are even(-nodes).
BU2. v is odd if $K(v)\cap P = \emptyset$ and $| K(v)\cap R |$ is odd.
BU3. v is even if $K(v)\cap P = \emptyset$ and $| K(v)\cap R |$ is even.
BU4. v is out if $K(v)\cap P \neq \emptyset$.

To classify a node v, all nodes in $K(v)$ have been classified already; hence BU2, BU3, and BU4 forms a trichotomy. As a consequence v is well-classified.

The top-down phase constructs a subset of nodes C as the output. The construction goes, level by level, from the top down as follows. (z is the root and v is not.)

TD1. If $z \in Q$, then $z \notin C$. If $z \in R$, then $z \in C$.
If $z \in P$, then we may set $z \in C$ or $z \notin C$. (There are two choices available in this case.)
TD2. If $v \in Q$, then $v \notin C$. (So $Q\cap C = \emptyset$.)
TD3. If $v \in R$, then $v \in C$ if and only if $p(v) \notin C$.
TD4. Once each even-node in the working level has been determined to be a member in C or not by TD3, those odd-nodes with the same parent (which is out) could be chosen certain of them to be members in C so that the closed neighborhood of that parent contains odd number of nodes in C.

Next we state a proof (omitted in [3, 1]) that the output C is an od-set of T_z.

According to TD4, $| N[v]\cap C |$ is odd for every out-node v (including $v = z$).

If $z \in R$, then $z \in C$ by TD1 and $K(z)\cap C = \emptyset$ by TD2 and TD3; thus $N[z]\cap C = \{z\}$. If $(z \neq)v \in R \cap C$, then $(K(v)\cup\{p(v)\})\cap C = \emptyset$ by TD2 and TD3; again $N[z]\cap C = \{v\}$. If $(z \neq)v \in R$ and $v \notin C$, then by TD3 $p(v) \in C$ and $N[v] \cap C = (K(v)\cap R)\cup\{p(v)\}$. Since $| K(v)\cap R |$ is even, $| N[v] \cap C |$ is odd.

For $v \in P$, $p(v) \in Q$ or does not exist. Again $K(v)\cap P = \emptyset$. When $v \in C$, $K(v)\cap C = \emptyset$ by TD2 and TD3. When $v \notin C$, $K(v)\cap C = K(v)\cap R$; by BU2 $| K(v)\cap C |$ is odd.

Hence $|N[v] \cap C|$ is odd for both cases. Therefore C is an od-set of T_z (thus, of T too).

3. The number of odd dominating sets in a tree

We just showed that each output C of Galvin Algorithm is an od-set of the input T_z. Denote the set of all possible outputs of Galvin Algorithm by $G(T_z)$. So $G(T_z) \subseteq OD(T_z)$. Later we will show that $G(T_z) \supseteq OD(T_z)$. Thus $OD(T) = OD(T_z) = G(T_z)$ (independent of z).

First let us compute $|G(T_z)|$. For each out-node v, $K(v) \cap P \neq \emptyset$. Suppose there are p odd-nodes and q out-nodes in T_z. The odd-nodes with the same parent forms a block. All odd-nodes are divided into q (or $q+1$) blocks. (The root z itself forms a block when $z \in P$.) A block of k odd-nodes with the same parent has 2^{k-1} choices available by TD4. If the root forms a block, there are 2 choices available by TD1. In total there are 2^{p-q} choices. (No matter what the root is an odd-node or not.) Therefore $|G(T_z)| = 2^{p-q}$.

In [1] Chen *et al.* introduced the quasi all-ones problems and studied whether the root is in the solution to the quasi all-ones problem. Together with the study on whether the root is in the solution to the all-ones problem, they came up with the following result: Suppose there are p odd-nodes and q out-nodes in T_z, then there are exactly 2^{p-q} od-sets of T_z. Since $G(T_z) \subseteq OD(T_z)$ and $|G(T_z)| = 2^{p-q}$, their result $|OD(T_z)| = 2^{p-q}$ implies that $OD(T_z) = G(T_z)$.

Our proof of that $G(T_z) \supseteq OD(T_z)$ does not depend on counting $|OD(T_z)|$ but on mathematical induction. Roughly speaking our proof goes as follows. Given an od-set C of T, we delete some subtree(s) from T_z to obtain a subtree T'_z. For convenience the subscript z will be dropped out. A node (in V') is stable if it receives the same type after the bottom-up phase on both T and T'. To make the argument simpler, we use prime to stand for stuffs in T'. So P', Q', R' are the odd-nodes, out-nodes, even-nodes in T' ; $p'(v)$, $K'(v)$ stand for the parent and the children of v in T'. Consider an od-set C' of T' derived from C. Since $|V'| < |V|$, by inductive hypothesis $OD(T') = G(T')$; thus $C' \in G(T')$. The top-down phase of Galvin Algorithm to construct C' on T' is essentially kept the same and we show it can be extended to construct C on T; thus $C \in G(T)$. Therefore $G(T) \supseteq OD(T)$.

The following lemmas are easily verified.

Lemma 1 *Suppose C is an od-set of a rooted tree. Assume x is a leaf and y = p(x). Then C contains either x or y, but not both.*

Proof. $| N[x] \cap C | = | \{x,y\} \cap C | = 1$. Q.E.D.

Lemma 2 *Suppose T′ is obtained from T by deleting a subtree rooted at some child of a node y. If y is stable, then x is stable for all* $x \in V'$.

Proof. Every leaf of V', except posssibly y, is a leaf of V. By BU1, they are stable. Also for every $x \in V'$, $x \neq y$, $K'(x) = K(x)$. Thus all nodes of level lower than that of y are stable. Since y is assumed to be stable, all nodes of level the same as that of y are stable. Afterwards all nodes of level higher than that of y are stable too. Q.E.D.

Lemma 3

(a) Assume $x \in K(y) \cap Q$. *If* $T' = T\text{-}T_x$, *then y is stable.*
(b) Assume $u,v \in P$, $u \neq v$, *and* $p(u) = p(v) = w$. *If* $T' = T\text{-}T_u$ *(or* $T' = T\text{-}T_v$*), then w is stable.*
(c) Assume $u,v \in R$, $u \neq v$, *and* $p(u) = p(v) = w$. *If* $T' = T\text{-}(T_u \cup T_v)$, *then w is stable.*

Proof.

(a) Note the type of y depends on whether $K(y) \cap P$ is empty or not and on the parity of $| K(y) \cap R |$. Since $K'(y) = K(y) - x$ and $x \in Q$, y is stable.
(b) Clearly $w \in Q$ and $w \in Q'$.
(c) When $K(w) \cap P \neq \emptyset$, $w \in Q$ and $w \in Q'$. When $K(w) \cap P = \emptyset$, note the parities of $| K(w) \cap R |$ and $| K'(w) \cap R' |$ are the same. Therefore w is stable. Q.E.D.

Next we propose an inductive proof of the main theorem.

Theorem 4 *For any vertex z of a tree T,* $OD(T_z) = G(T_z)$.

Proof. Since the root z is specified, we drop the subscript and apply the T and T' terminologies. Let $T(x)$ denote the subtree of T from the root z to all nodes of the same level

as x. We have shown $OD(T) \supseteq G(T)$ in the previous section. Next we will show $OD(T) \subseteq G(T)$. First verify it straightforwardly for trees with at most 4 vertices. Then let T be a rooted tree with more than 4 nodes.

Case 1: There are two different leaves u,v with the same parent w.

Let $T'=T-\{u,v\}$. By Lemma 3(c), w is stable. By Lemma 2, all T'-nodes are stable. For $C \in OD(T)$, either $w \in C$ or $w \notin C$.

In case $w \in C$, by Lemma 1, $u,v \notin C$. Thus $C \in OD(T')$. By inductive hypothesis $C \in G(T')$. Since all T'-nodes are stable, the top down phase to construct C in T' will produce $C \cap T(w)$ in T when it reaches down to the level of w. Go down one more level, $u,v \notin C$ due to TD3. So Galvin Algorithm will output C eventually. Hence $C \in G(T)$.

In case $w \notin C$, by Lemma 1, $u,v \in C$. Let $C' = C-\{u,v\}$. It is easy to check $C' \in OD(T')$. By inductive hypothesis $C' \in G(T')$. Since all T'-nodes are stable, the top down phase to construct C' in T' will produce $C \cap T(w)$ in T when it reaches down to the level of w. Go down one more level, both u and v will be included in C due to TD3. So $C \in G(T)$.

Case 2: Every node is adjacent with at most one leaf.

Every node in the lowest level of a rooted tree is a leaf, called a bottom leaf. Let u be a bottom leaf of T. Set $w = p(u)$ and $s = p(w)$. According to the above condition, $K(w) = u$.

subcase 2.1 $K(s) = \{w\}$.

Note $s \neq z$, otherwise T has only 3 nodes. Hence s has a parent, say t. Let $T' = T-\{u,w,s\}$. Clearly $u \in R$, $w \in P$, and $s \in Q$. By Lemma 3(a), t is stable.

For $C \in OD(T)$, by Lemma 1, either $u \in C$ or $w \in C$, but not both. Since $N[w] = \{u,w,s\}$, $s \notin C$; otherwise $|N[w] \cap C| = 2$, a contradiction. Let $C \cap T' = C'$ and check $C' \in OD(T')$. By inductive hypothesis $C' \in G(T')$.

Since all nodes of T' are stable, the top down phase could construct $C' \cap T'(t)$ $(= C \cap T(t))$ on both T' and T. After the level of t was examined, Galvin Algorithm on T will set $s \notin C$ due to $s \in Q$. By TD4 $w \in C$ if and only if $t \notin C$. By TD3 $w \in C$ if and only if $u \notin C$. Therefore either $u \in C$ or $w \in C$, but not both. So in either case $C \in G(T)$.

subcase 2.2 $K(s) = \{w,x\}$, where x is a leaf.

Note $s \neq z$, otherwise T has only 4 nodes. Hence s has a parent, say t. Let $T' = T - \{u,w,x,s\}$. Clearly $u, x \in R$, $w \in P$, and $s \in Q$. By Lemma 3(a), t is stable.

For $C \in OD(T)$, by Lemma 1, either $u \in C$ or $w \in C$, but not both. Since $N[w] = \{u,w,s\}$, $s \notin C$; otherwise $| N[w] \cap C | = 2$, a contradiction. Hence C contains either $\{x,u\}$ or $\{x,w\}$. Let $C \cap T' = C'$ and check, in both cases, $C' \in OD(T')$. By inductive hypothesis $C' \in G(T')$.

Since all nodes of T' are stable, the top down phase could construct $C' \cap T'(t)$ $(= C \cap T(t))$ on both T and T'. After the level of t examined, for the deleted nodes s,w,x,u, Galvin Algorithm on T sets $s \notin C$ and $x \in C$ due to $s \in Q$ and $x \in R$. By TD4 $w \in C$ if and only if $t \in C$. By TD3 $w \in C$ if and only if $u \notin C$. Therefore $C \in G(T)$.

subcase 2.3 $K(s)$ consists of at most one leaf-child and at least two children that each has a leaf-child. (s could be the root.)

We divide the od-sets C in such a rooted tree T into two classes: C contains a bottom leaf or no bottom leaf is contained in C.

In the former case, let x be a bottom leaf in C and $y = p(x)$. Since $x \in C$, by Lemma 1, $y \notin C$; thus $| N[y] \cap C | = 1$, as a consequence $s \notin C$. Let $T' = T - \{x,y\}$. Clearly $s \in Q$. Since s has another child that is odd, $s \in Q'$ too. Thus s is stable. By Lemma 2 all nodes in T' are stable. Let $C' = C - \{x\}$. Check $C' \in OD(T')$. By inductive hypothesis $C' \in G(T')$.

Since all nodes of T' are stable, the top down phase could construct $C' \cap T(s)$ on both T' and T. By TD2, $s \notin C'$ and $| N'[s] \cap C' |$ is odd. Since $N[s] = N'[s] \cup \{y\}$, when we choose

the same odd-nodes in the level of y to be members of C', y will be excluded to be a member of C; by TD3, $x \in C$. So $C \in G(T)$.

In the latter case, by Lemma 1, all odd-children of s are contained in C. Let y be an odd-child of s, $y = p(x)$, and $T' = T\text{-}\{x\}$. Since $N[y] = \{s,y,x\}$, $x \notin C$, $y \in C$, we get $s \notin C$. Hence C contains $K(s)$. Therefore $C \cap T_s = K(s)$. Check C' $(=C) \in OD(T')$ and s $(\in Q')$ is stable. Thus all nodes of T' but y are stable ($y \in P$ and $y \in R'$). Note that s contains no out-child in T and in T'. By inductive hypothesis C' is constructible on T' by Galvin Algorithm. At the level of y, Galvin Algorithm first include all the even-children ($y \in R'$) of s due to $s \notin C$ and TD3. Next all odd-children of s could be included since C' is constructible. It means $|K(s)|$ fits the parity condition in TD4. When Galvin Algorithm runs on T : Since $y \in P$, the number of even-children of s decreases by one, but the number of odd-children of s increases by one. Hence, by TD4, we could take all odd-nodes in $K(s)$ into C. Namely $C \cap T(y)$ has been constructed. Finally no bottom leaf is contained in C' and in C, by TD3. Thus the given od-set C would be obtained as the output. Therefore $C \in G(T)$. Q.E.D.

As a consequence $|OD(T)| = |G(T_z)| = 2^{p-q}$.

Reference

1. Chen,W.Y.C., Li,X., Wang,C., and Zhang,X. *The minimum all-ones problem for trees*, SIAM J. Comput., Vol. 33, No. 2, pp.379-392 (2004).
2. Eriksson,H., Eriksson,K., and Sjöstrand,J. *Note on the lamp lighting problem*, Adv. Appl. Math., Vol. 27, pp.357-366 (2001).
3. Galvin,F. *Solution to problem* 88-8, Math. Intelligencer, Vol. 11, No.2, pp.31-32 (1989).
4. Martín-Sánchez,Ó. and Pareja-Flores,C. *Two reflected analysis of Lights Out*, Mathematics Magazine, Vol. 74, No. 4, pp.295-304 (2001).
5. Sutner,K. *Linear cellular automata and the Garden-of-Eden*, Math. Intelligencer, Vol. 11, No.2, pp.49-53 (1989).

Recevied Oct 8, 2014
Revised Apr 29, 2015
Accepted May 6, 2015

樹的奇控集個數札記

蕭鴻銘

輔仁大學 數學系

摘　要

一個圖的奇控集是一個制控每個點奇數次的控集。Sutner 證明了每個圖都包含了奇控集。Galvin 提出一個圖網演算法去找出一個樹的奇控集。後來 Chen 等人證明了一個樹的每個奇控集均可由 Galvin 演算法找出來。本文提出這個定理的一個歸納證明。

關鍵字：奇控集 , Galvin 演算法。

在點對點隨選視訊系統中基於緩衝區潛在貢獻的動態鄰居調整機制

呂俊賢　湯景安

輔仁大學資訊工程系

摘　要

面對需要大量傳輸頻寬的影音串流服務，傳統的主從式架構已無法滿足現今使用者的需求，因此將點對點技術應用在影音串流服務上已是個不可避免的趨勢。但是因為使用者進入時間的不同所處在的播放點也不同，如果挑選到播放點差距很遠的鄰居對雙方都可能幫助甚小。即使在挑選播放點接近的鄰居後，也應該考慮因為網路訊務變動可能改變鄰居節點之間的傳輸效能及播放點差距，來再次更換鄰居。在本篇論文中，我們提出了基於緩衝區潛在貢獻的動態鄰居調整機制，作為節點選擇鄰居與發送請求對象的策略。節點挑選鄰居會依照對方跟自己緩衝區的重疊率及負載狀況來算出潛在貢獻，優先選擇潛在貢獻較高的節點當鄰居。並且在挑選節點發送請求時也會優先向潛在貢獻較高的節點發送。利用這些策略可以提升節點之間的上傳頻寬利用率，提升使用者觀看影片的品質，並使得伺服器的負載降低。實驗數據顯示，我們的方法比一般的方法在影片觀看品質 CI 有 26% 左右的改善，並且讓伺服器的負載減少了 24% 左右。

關鍵字：點對點網路，影音串流，動態鄰居選擇策略，
請求選擇策略。

1. 簡介

1.1 影片分享

近年來，隨著網際網路的普及使用者數量也快速增加。而網際網路眾多的應用中，以多媒體串流所佔的頻寬最大 [1]，因此如何使影片傳輸變得更快更有效率便成為了一個很重要的課題。在傳統的主從式架構下，當使用者的數量龐大時，中央伺服器就會無法做及時的處理。而點對點架構不同於主從式架構的地方在於點對點網路架構中沒有客戶端或伺服器的概念，只有平等的同級節點，因此具有系統建置成本較低及擴展性較佳等優點。利用點對點架構所做的多媒體串流系統可負擔大量的使用者，在實務上獲得了非常好的效果，例如 CollectCast[2]、ZIGZAG[3]、和 PPStream [4] 等。

點對點隨選視訊系統 (P2P Video-on-Demand System) 是先把影片儲存在伺服器裡，使用者可以不受時間、空間的限制，透過網路隨選隨看這些影片，系統示意圖如圖 1，系統由下列三種角色組成： Peer 是正在下載影片的節點，影片伺服器擁有完整影片檔的伺服器，而 Tracker 則是會提供一份固定數量的節點清單給新加入節點，作為挑選鄰居之用。點對點隨選視訊系統運作之過程如下：新加入的節點決定要觀看哪部影片後，聯絡負責該影片的 Tracker，並取得一份目前已在系統中的節點清單，依照清單的順序依序向其中節點發出連線請求成為鄰居。接著節點開始觀看影片，依照片段選擇策略選出要下載的影片片段，並向鄰居發出下載請求。若沒有鄰居有此片段，則向影片伺服器要求下載。接收到下載請求的節點或影片伺服器會依照其個別的策略選擇要服務哪個請求。

本論文的研究重點是如何改善點對點隨選視訊系統的鄰居選擇機制和請求選擇機制。鄰居選擇機制一開始建立整個網路的拓樸結構對整體的效能影響很大。我們希望設計一個良好的鄰居選擇機制，並且可以依照影片觀看品質動態調整而不是建立好了就不再更動。而在請求選擇機制方面，大多數的研究都只限於找出頻寬最大或是往返時間最小的節點來對它發出請求，而沒有考慮到對方是否會接受該請求，所以應該要有一個相對應的機制來配合以達到最佳效果。本論文中我們設計一個請求選擇機制，收到請求的一方會依照情況的不同作出相對的回應，目標是希望藉由上述方法來增進整體系統的頻寬使用率以及觀看影片的品質。

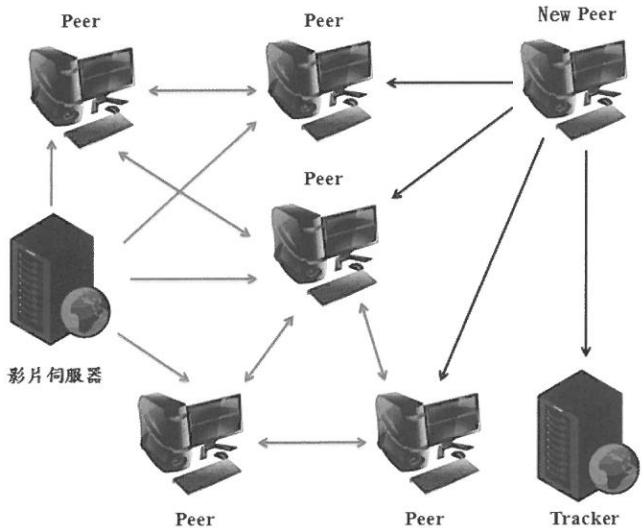


圖 1： 點對點隨選視訊系統

本論文其餘部分的內容如下，第二節將介紹點對點隨選視訊系統的相關研究，第三節說明我們所提出的機制，第四節為效能評估，第五節則是結論。

2. 相關研究

關於點對點隨選視訊系統的鄰居選擇機制主要有下列幾類：第一類認為應該由節點自行測量挑選實際上往返時間較短的節點。如 [5] 認為利用節點的 IP 位置來判斷對方的物理位置，挑選離自己相近或是挑選距離影片伺服器的跳數最小的節點作為鄰居。[6] 實際測量節點的頻寬以及雙方的往返時間來找出最佳的節點來做為鄰居，[7] 則是先找出幾個比較靠近自己的節點再來測量雙方實際的傳輸頻寬及往返時間，只要到達門檻就可以當鄰居，找到足夠的鄰居即停止。第二類方法是認為跟 ISP 合作才能解決最根本的問題。因為點對點串流系統所產生的流量相當大，對 ISP 來說是很大的負擔。[8] 提出的機制可透過 ISP 來知道節點之間的 ISP 跳數和骨幹連接成本，由此可以挑選較近的節點當鄰居來增進系統效能，同時也可以減少 ISP 的花費。[9] 提出的機制是增加一個 ISP Tracker 來讓 ISP 協助節點來尋找鄰居，一樣可以增進系統效能，ISP 也可以依照自己政策來減少花費。另外一類 [10] 認為應該挑選進入系統時間長短較相近的

人做為鄰居，因為他們所需求的片段應該也相近，比較容易找到想要的片段。節點之間可以利用交換控制訊息來傳送彼此的進入系統時間，在挑選鄰居時，只要雙方的進入系統時間差距的絕對值小於門檻以及對方還有剩餘的連接數即可。[11-13] 則將研究焦點放在如何挑選最佳的節點發出請求。[11] 提出先利用測量往返時間對鄰居排序，再依序對鄰居發出測試封包測試可用的頻寬，若超過門檻值就選定此節點發送請求。[12] 認為應該選擇影片播放點最接近的節點發送請求，因為雙方都有很大的機率有對方所要的片段，不僅可以充分利用雙方的上傳頻寬，也可以降低伺服器的負擔。[13] 認為不能只挑看起來上傳頻寬高的節點，而是要挑選實際上對使用者能提供最高頻寬的節點發出請求。

3. 方法設計

本節將描述我們所提出的基於緩衝區潛在貢獻的動態鄰居調整機制，此方法涵蓋兩個部分，一是請求選擇機制，另一部分是動態鄰居選擇機制。

3.1 系統概述

一個新的節點進入系統後由 Tracker 依照其進入系統時間給予一份節點清單，每個節點依照其上傳能力決定最大鄰居數。我們令 w = 節點上傳頻寬 / 影片 bit rate，而該節點所能擁有的最大鄰居數為 $w * K$ (K 為一給定之常數)。每個節點定期計算它每個鄰居的潛在貢獻 (potential)，對某一節點 m 而言，其鄰居節點 i 之潛在貢獻定義為 $P(i)$，計算公式如下：

$$P(i) = \alpha \times (\text{i 對 m 緩衝區貢獻度}) + \beta \times \frac{\text{節點 i 上傳頻寬}}{(\text{影片 bit rate}) \times (\text{i 的佇列長度})}$$

其中 i 對 m 的緩衝區貢獻度為節點 i 緩衝區擁有的片段中節點 m 所沒有的片段個數。在鄰居選擇機制方面，我們假設每個節點會從 Tracker 提供的清單中挑選進入系統時間長短較相近的節點做為鄰居 [10]。因為我們不考慮使用者會暫停或跳動播放點的情況，所以當兩個節點進入系統時間接近則其影片播放點也會是接近的。

在請求選擇機制方面，每個節點會定期計算每個鄰居的潛在貢獻 *P(i)* 和負載量，並依照其潛在貢獻的大小對鄰居排序。當節點要發出請求下載影片片段時，若潛在貢獻最大的鄰居有此片段且負載量小於一個門檻值則對其發出請求，若其負載量超過門檻值則選擇潛在貢獻次之的鄰居。若是所有鄰居負載量都超過門檻值，則對伺服器發出請求。而在片段選擇機制方面，我們假設一個節點最多可以同時發出 3 個請求，其中兩個請求必須是選擇最靠近播放點的片段，而第三個則是隨機選擇欲下載的片段。

3.2 基於緩衝區潛在貢獻的動態鄰居調整機制

在此我們提出一個基於緩衝區潛在貢獻的動態鄰居調整機制，我們利用 Continuity Index (CI) 來評量觀看影片的品質，其定義為一段時間內有多少比例的影片片段及時被播放。我們給定兩個參數 *HIGH* 及 *LOW*，當 CI 大於 HIGH 表示觀看品質優良，若 CI 介於兩者表示品質可接受，若 CI 低於 *LOW* 代表品質不佳。當使用者觀看影片的品質不佳時節點便執行 Algorithm 1 來動態調整鄰居。若現有鄰居數大於門檻值，則把潛在貢獻最小的鄰居都刪除，直到鄰居數降到門檻值。再來向剩下的鄰居發出要求取得每個鄰居的鄰居節點清單，將這些節點置入一候選清單，看是否存在適合作為直接鄰居的節點。我們計算候選清單裡所有節點對我們的潛在貢獻，依照其大小排序，再依序對候選清單裡的節點發出連線請求。若對方接受則加為鄰居，對方不接受則跳過，重複步驟直到候選清單為空。Algorithm 1 的詳細步驟如下：

```
Algorithm 1: (assume executed at peer i)
if  CI < LOW
    if  NumofNeighbors > Threshold
        Remove neighbors with smallest Potential until NumofNeighbors ≤
        Threshold
    Get peer list from all neighbors and put them into a candidate list
    Calculate Potential for each peer in candidate list
    while  candidate list is not empty
        Remove peer p with maximum Potential from the candidate list
        if  P(p) > P(j) (where j is in i's neighbor list and P(j) is the minimum)
            SendRequest( p )
                case Accept: AddNeighbor( p ) and disconnect j
                case Reject: try next peer in candidate list
```

而收到連線請求的節點會執行 Algorithm 2 來決定是否接受這個請求，如果現有鄰居數小於門檻值則無條件加入。當現有鄰居數大於等於門檻值時，如果 CI ≥ *HIGH*，因為目前自己的觀看品質夠好，可以多幫助他人因此無條件加入；如果 *LOW* ≤ CI < *HIGH*，為了保障自身以及現有鄰居的觀看品質，因此拒絕增加新鄰居；如果 CI < *LOW*，因為目前自己的觀看品質不佳，應該要考慮接受對自己較能有貢獻的鄰居。此時先計算對方的潛在貢獻，若大於自己鄰居清單裡潛在貢獻最低值，則刪除此貢獻最低的鄰居，並將對方加為新鄰居。如果對方的潛在貢獻比自己鄰居清單裡潛在貢獻最低的鄰居還要低則拒絕。Algorithm 2 的詳細步驟如下：

```
Algorithm 2: (assume peer m receives a connection request from peer i)
if   NumofNeighbors < Threshold
     Accept ( Peer i )
else
     if   CI ≥ HIGH
          Accept ( Peer i )
     else if   LOW ≤ CI < HIGH
          Reject ( Peer i )
     else
          Calculate P(i)
          if   P(i) > P(j) (where j is in m's neighbor list and P(j) is the minimum)
               Accept ( Peer i ) and disconnect peer j
          else
               Reject ( Peer i )
```

4. 效能評估

4.1 模擬環境

我們撰寫一個模擬程式來評估我們所提出的動態鄰居調整機制之效能。如果節點

收到多個請求會依收到順序服務，所有節點會陸續加入系統，並且在看完影片後馬上離開系統。一個影片伺服器擁有所有的影片片段，而總節點數量會依照節點平均加入間隔時間以及模擬時間變動。影片伺服器的最大上傳頻寬是 20Mbps，節點的下載頻寬為 6Mbps，而上傳頻寬則有 1Mbps 和 2Mbps 兩種，鄰居數量會依照節點的上傳頻寬而不同。每個節點最大可發出的請求數量為 3 個，影片 bit rate 為 1Mbps，長度為 500 秒，分割成 500 個片段，每個片段的大小是 128KB。我們參考 [14] 的觀點把 *LOW* 和 *HIGH* 的值分別設為 0.85 和 0.95，詳細模擬參數如表 1 所示。

表 1. 模擬環境參數設定

伺服器	**1 部，上傳頻寬 20Mbps**
節點進入平均間隔時間	**2 秒**
節點上傳及下載頻寬	**上傳 1Mbps 或 2Mbps，下載 6Mbps**
節點最大可同時發出之請求數量	**3**
影片長度	**500 秒，頻寬為 1Mbps**
片段大小	**128KB**
緩衝區大小	**50 個片段**
LOW, HIGH	**LOW = 0.85, HIGH = 0.95**

4.2 模擬結果

在本實驗中總共比較三種方法。第一種方法是我們所提出的動態鄰居調整機制，第二種方法為非動態調整方法，它是把第一種方法的動態鄰居調整機制關閉，只保留利用潛在貢獻 *P(i)* 的大小順序來發送請求，藉此來比對有無鄰居調整之效能差異，第三種方法為一般方法，進入系統挑選播放點相近的鄰居，但發送請求時為隨機挑選。

4.2.1 改變平均上傳頻寬的影響

圖 2 和圖 3 是在改變節點平均上傳頻寬的情況下，測量其影片觀看品質 CI 和伺服器服務請求量的實驗結果。當節點平均頻寬增加時，代表下載片段的時間可以縮短，片段可以及時到達。從圖 2 中可以發現一般方法在平均頻寬增加的情況下 CI 平穩上升，但是相較於非動態方法還是低了不少。因為一般方法發送請求是隨機挑選，並未像我們的方法除了考慮對方佇列的長度外，還會依照對方的頻寬的大小來調整可接受的負載門檻值，所以 CI 值有 6% 至 20% 左右的改善。而動態調整方法又相對於非動態方法

有更高的 CI 值，因為動態調整方法會在 CI 值低於門檻值時依照緩衝區重疊率和負載一起做考量找出更好的節點當鄰居，而 CI 值也有 5% 至 8% 左右的改善。從圖 3 可以看出，一般方法即使節點平均頻寬增加，對伺服器的請求量還是居高不下，而動態和非動態方法對伺服器的請求量都有明顯的下降。

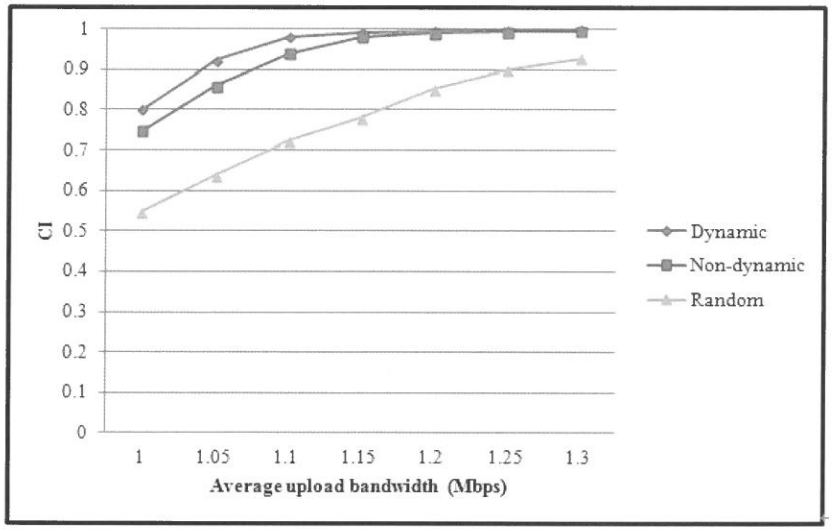


圖 2：平均 CI 比較圖（改變節點平均上傳頻寬）

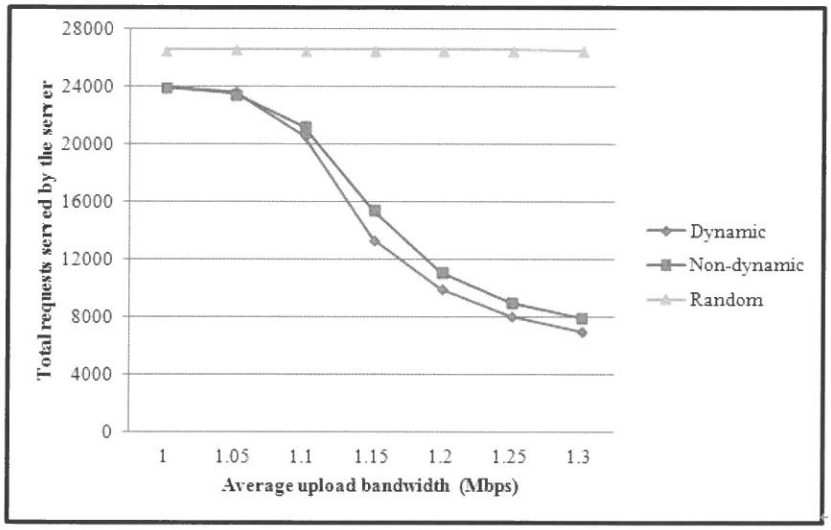


圖 3：伺服器服務請求量比較圖（改變節點平均上傳頻寬）

4.2.2 改變最大鄰居數的影響

圖 4 和圖 5 是在改變最大鄰居數的情況下，測量其影片觀看品質 CI 和伺服器服務請求量的實驗結果。當最大鄰居數增加時，代表可交換片段的節點增加，同時整體可利用頻寬增加。圖中 5(10) 代表有 1Mbps 上傳的節點可擁有 5 個鄰居，而有 2Mbps 上傳的節點可擁有 10 個鄰居。從圖 4 中可以發現，當最大鄰居數增加時，所有的方法的 CI 都逐漸提升。一般方法即使鄰居數夠多，因為它沒有良好的負載平衡機制，CI 值的提升相較於其餘兩個方法低了許多。在鄰居數較少時三種方法並無明顯的差異，因為鄰居數過少整體頻寬不足以支援整體系統所需。當鄰居數變大時，動態方法和非動態相對於一般方法就有明顯的提升，其中動態方法表現最佳，*跟一般方法相比 CI 值有 18% 至 32% 的提升，和非動態方法比較也有 4% 至 8% 的提升。*

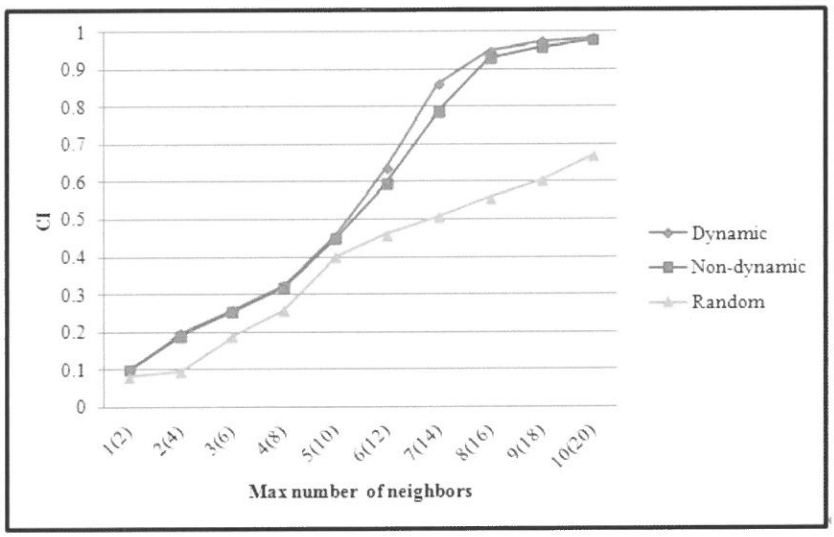


圖 4：平均 CI 比較圖（改變最大鄰居數）

從圖 5 可以看出，一般方法即使最大鄰居數增加，對伺服器的請求量還是居高不下，而動態和非動態方法對伺服器的請求量有明顯的下降，相對於一般方法則減少了 10% 至 25% 左右。

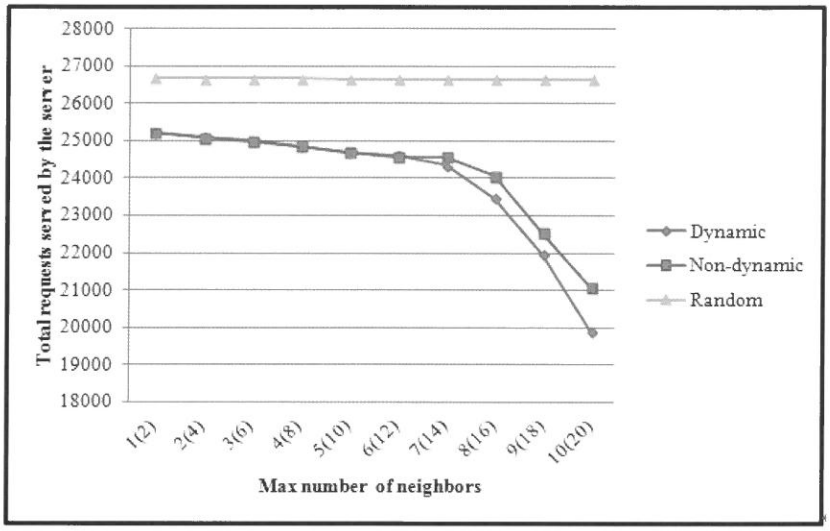


圖 5：伺服器服務請求量比較圖（改變最大鄰居數）

4.2.3 改變緩衝區大小的影響

圖 6 和圖 7 是在改變緩衝區大小下，測量其影片觀看品質 CI 和伺服器服務請求量的實驗結果。當緩衝區大小增加時，和鄰居重疊的機率也變大，可以互相交換片段的機率也增加。從圖 6 和 7 中可以發現一般方法在緩衝區大小增加時 CI 值也逐漸提升，而動態和非動態方法在緩衝區太小時 CI 值反而比一般方法差，這是因為緩衝區太小時能夠互相交換片段的節點也少。我們的方法傾向於優先找其他節點下載，而一般方法只要隨機選到的節點負載太高就會去找伺服器下載，因此在此種情況下一般方法的 CI 值反而較高。但是當緩衝區大小逐漸增加後，在緩衝區有重疊的鄰居夠多的情況下，動態和非動態方法的 CI 值就會大幅提升。但是在緩衝區越來越大的情況下，CI 值又會略微降低，這是因為節點發出的三個請求中有一個是隨機的，緩衝區越大有可能會選到離播放點較遠，幾乎沒有鄰居擁有的片段，此時只能向伺服器下載，造成 CI 值降低，伺服器負載升高的現象。

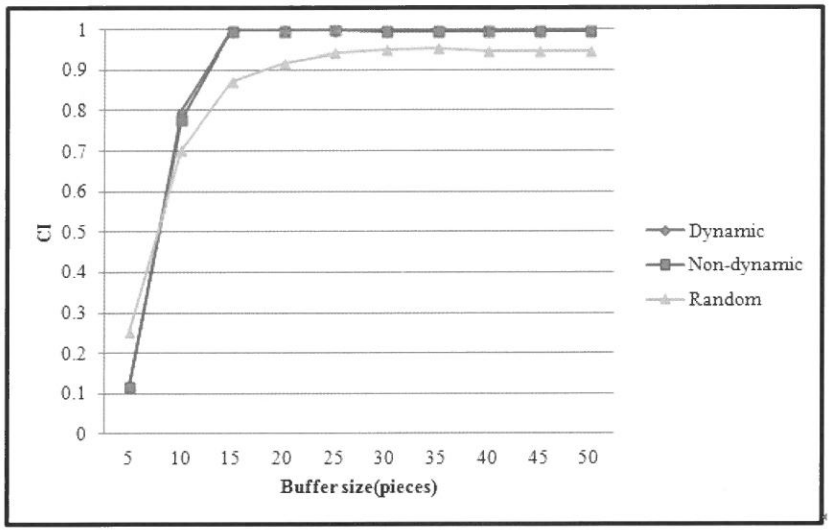


圖 6：平均 CI 比較圖（改變緩衝區大小）

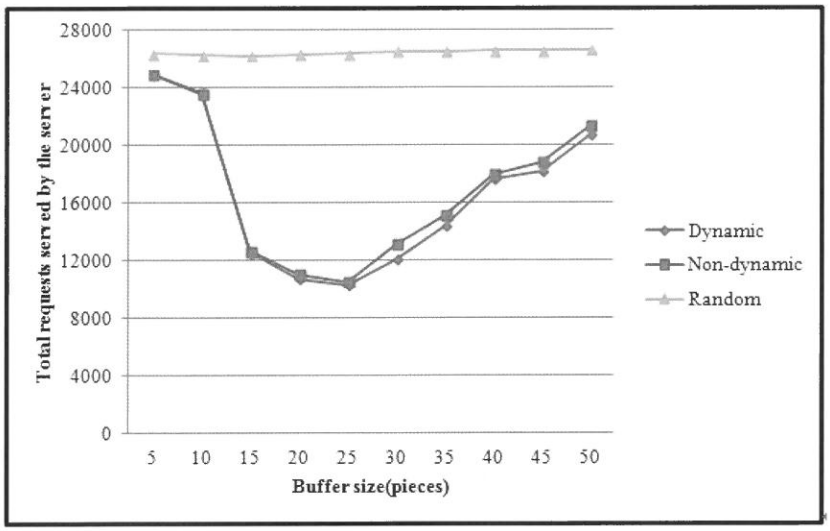


圖 7：伺服器服務請求量比較圖（改變緩衝區大小）

5. 結論

影音串流服務在目前生活中的需求已越來越大，將點對點技術應用在影音串流服務上已是個不可避免的趨勢。在鄰居選擇方面，眾多研究結果顯示挑選播放點接近的鄰居是一個簡單且有效率的方法。但即使在挑選播放點接近的鄰居後，也應該考慮網路訊務變動可能改變鄰居節點之間的傳輸效能及播放點差距，來動態更換鄰居。我們提出了基於緩衝區潛在貢獻的動態鄰居調整機制，作為節點選擇鄰居與發送請求對象的策略。當影片觀看品質不好時，節點將會依照對方跟自己緩衝區的重疊率及負載狀況來算出潛在貢獻，優先選擇潛在貢獻較高的節點當鄰居，並捨去潛在貢獻較低的鄰居節點。而在挑選節點發送請求時會優先向潛在貢獻較高並且負載低的節點發送，若所有鄰居節點的負載都太高則把請求送給影片伺服器。實驗數據顯示，我們的方法比一般的方法在影片觀看品質 CI 平均有 18% 左右的改善，並且在伺服器的負載上面減少了約 20% 左右。

Reference

1. CISCO news: Global Mobile Data Traffic Forecast Update, 2012–2017, http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-520862.html
2. Hefeeda, M., Habib, A., Xu, D., Bhargava, B., and Botev, B., "CollectCast: A Peer-to-Peer Service for Media Streaming," *ACM/Springer Multimedia Systems*, October 2003.
3. D. A. Tran, K. A. Hua, and T. Do, "ZIGZAG: an efficient Peer -to- Peer scheme for media streaming," *in INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, 2003, pp. 1283-1292 vol.2.
4. PPStream, http://www.ppstream.com/
5. Y. Fukushima, T. Yin, K. Inada, and T. Yokohira, "AS-friendly Peer selection algorithms without AS topology information in P2P live streaming," *in Information and Telecommunication Technologies (APSITT), 2010 8th Asia-Pacific Symposium on*, 2010, pp. 1-6.
6. Y. Hao, H. Peilin, and X. Kaiping, "Adaptive topology optimization base on bidirectional Peer selection in Peer -to- Peer media streaming," in *Communications and Networking in China, 2008. ChinaCom 2008. Third International Conference on*, 2008, pp. 574-578.
7. W. Jiannan and Z. Lei, "The Arithmetic of Peers Selecting and Improvement of Peer

Selection Strategies in P2P Live Streaming," in *Parallel Architectures, Algorithms and Programming (PAAP), 2011 Fourth International Symposium on*, 2011, pp. 247-251.

8. Z. Hong, X. Yinlong, H. Yuchong, and L. Xiaobin, "A Peer Selection Mechanism in P2P Networks Based on the Collaboration of ISPs and P2P Systems," i*n Information Science and Engineering (ICISE), 2009 1st International Conference on*, 2009, pp. 1573-1576.
9. M. Z. Masoud, H. Xiaojun, and C. Wenqing, "Constructing a locality-aware ISP-friendly Peer -to- Peer live streaming architecture," *in Information Science and Technology (ICIST), 2012 International Conference on*, 2012, pp. 368-376.
10. I. M. Moraes and O. C. M. B. Duarte, "A Lifetime-Based Peer Selection Mechanism for Peer -to- Peer Video-on-Demand Systems," *in Communications (ICC), 2010 IEEE International Conference on*, 2010, pp. 1-5.
11. H. Tai-Hua, H. Ming-Hung, and M. Yu-Ben, "Adaptive and Efficient Peer Selection in Peer -to- Peer Streaming Networks," in *Parallel and Distributed Systems (ICPADS), 2011 IEEE 17th International Conference on*, 2011, pp. 753-758.
12. W. Zheng, L. Nianwang, K. L. Yeung, and L. Zhibin, "Closest Playback-Point First: A New Peer Selection Algorithm for P2P VoD Systems," *in Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE*, 2011, pp. 1-5.
13. K. Takayama, T. Fujimoto, R. Endo, and H. Shigeno, "Neighbor Selection Based on Transmission Bandwidth on P2P Live Streaming Service," in *Advanced Information Networking and Applications Workshops (WAINA), 2012 26th International Conference on*, 2012, pp. 105-110.
14. A. Habib and J. Chuang, "Service differentiated peer selection: an incentive mechanism for peer-to-peer media streaming," Multimedia, IEEE Transactions on, vol. 8, pp. 610-621, 2006.

Recevied Oct 15, 2014
Accepted Mar 3, 2015

Dynamic Peer Selection Based on Buffer Availability in P2P VoD Systems

Chun-Hsien Lu and Ching-An Tang

Dept. of Computer Science and Information Engineering
Fu Jen Catholic University
New Taipei City, Taiwan, R.O.C. 24205

Abstract

Traditional client-server model has become inadequate for video streaming due to its bandwidth and efficiency bottleneck, and it has been an inevitable trend to apply peer to peer technology in video streaming services. BitTorrent protocol has been the most widely used protocol for file sharing, but it requires modifications, especially in neighbor selection, before it can used suitably for video streaming. In this paper, we proposed a dynamic peer selection algorithm based on buffer availability. Each peer calculates other peers' availability based on their buffer overlap and load, and will give priority to peers with higher availability when sending a request. Simulation results show that this strategy can improve the upload utilization between peers and reduce the server's load, hence enhance the video quality that the user perceives. Our mechanism can improve about 26% on the continuity index, and reduce about 24% of the server's load.

Key words: Peer-to-peer Network, Video Streaming, Dynamic Neighbor Selection

在 CUDA 平台上加速大型函數之布林比對

王國華　張峰銘

輔仁大學資訊工程系所

摘　要

布林比對問題用來檢查兩個函數在作輸入排列組合及相位設定後，是否相等。本篇論文研究主題是以早期所提出的漸進式學習方法 [11] 為基礎，我們首先提出了計算權重的方法，將原本的演算法改良以加快求解的速度。分析先前的布林比對演算法主要的執行時間是花費在 AND 運算與 Reduction 這兩個程序上，本篇論文研究如何將此兩程序平行化。直接將以上程序平行化會有三個問題：原本之資料儲存方式導致 AND 運算效能變慢、處理大型電路時 GPU 記憶體不足、無法實現記憶體合併存取機制 (Memory Coalescing) 導致 GPU 記憶體讀取效率不佳。針對以上三個問題，我們提出三個平行程序最佳化的方法：改變資料儲存方式、利用 CUDA 串流機制降低 GPU 記憶體使用量及轉置資料儲存方式實現記憶體合併存取機制以提升讀取效率。根據實驗結果，權重計算演算法比原本快了 2.6 倍左右，另外，對大型電路作加速測試的實驗會測量不同輸入大小之電路，平行化後求解之加速情形。整體而言，相較於原本循序執行的方式，直接平行化方法大約可達到 5 倍的加速，經最佳化後之方法大約可以達到 30 倍的加速，證明我們提出之平行演算法在對大型函數作布林比對時，的確非常有效率。

關鍵字：布林比對、不完全描述函數、CUDA、GPU。

khwang@csie.fju.edu.tw ssive99@csie.fju.edu.tw

1. 序論

1.1 布林比對問題

給定兩個具有相同輸入數目的布林函數，布林比對 (Boolean Matching) 問題是檢查當經過一些排列組合 (input permutation) 及輸入 / 輸出相設定 (input/output phase assignment) 後，這兩個函數是否相等，也稱為 NPN-equivalence 問題。布林比對問題已被廣泛地應用在邏輯合成 (logic synthesis)、驗證 (verification)[1] 和技術映成 (technology mapping）[2] 之中。假設二個具有 n 個輸入與單一輸出的布林函數，NPN-equivalent 布林比對問題的時間複雜度為 O(n!2^(n+1))，意即可能的對應組合就有 n!2^(n+1) 種。最直接的作法就是把這 n!2^(n+1) 種的對應組合逐一檢查是否相等，但是採取這種直接的作法卻非常地沒有效率。

為了能夠有效率處理布林比對問題，很多的相關的技術已經陸續地被提出 [3-9]，其中包含標準式 (Canonical Form)、特徵值 (Signature) 以及光譜法 (Spectral Method）。標準式之作法是將兩個函數轉換得到各自的表示式，再驗證此二個表示式是否相等，而近年來仍然有相關的改善方法陸續被提出。特徵值之作法為如果二個函數確實可以比對成功，其各自輸入具有的特徵值也必然相同，藉此觀念來排除不可能的對應關係並且縮小解空間的範圍 [10]。光譜法則是將函數量化成向量矩陣表示再判斷是否相等，但是卻必須耗費大量的記憶體使用量。上述所提到的方法，大部分都只能對完全描述函數做處理，只有 [8] 可以應用到不完全描述函數的比對上。1994 年在 DAC(Design Automation Conference) 上 [9] 提出了利用 on-set 與 off-set 作積項配對產生部分對應關係之布林比對演算法，並可以直接應用在不完全描述函數上。

1.2 研究動機

在 [11] 中，改良了 [9] 所提出的不完全描述函數之布林比對演算法，首先利用機器學習的方式移除多餘積項配對，接著利用特徵值等技巧排除不可能的對應關係以縮小其搜尋空間。電路比對之平均時間，由 1797 秒降至 2.77 秒，大幅改善整體效能。為了瞭解在布林比對的過程當中各個步驟所花費之時間，我們測試了 90 組 MCNC benchmark 電路，相關統計資訊如表 1 所示。有將近 79% 以上的時間花費在合法積項

檢查)feasible cubes checking) 與多餘積項移除 (redundant cubes removal) 的動作上。在對輸入越大的布林函數作比對時，產生的積項也會越來越多，其檢查與移除多餘積項所佔的時間比例也隨之提高，若能將這些步驟平行化，便可加速布林比對之效率。

表 1：MCNC benchmark 電路布林比對之相關統計資訊

Pre-Processing	Redundant Pair Checking	Mapping Covered	Feasible Cubes Check	Redundant Cubes Removal	Run Time
4.02%	**1.52%**	**15.18%**	**33.03%**	**46.25%**	**100.00%**

近幾年，NVIDIA 提出了 CUDA(Compute Unified Device Architecture) 運算架構，讓使用者可以將 GPU(Graphic Processing Unit) 應用在一般的資料運算上。GPU 的架構為單一指令 / 多執行緒 (SIMTSingle Instruction Multiple Threads)。在此架構之下，執行一個指令時，執行緒可同時對不同的資料作運算，跟 CPU 相比，其浮點數運算能力高出許多。顯示近年來 CPU 和 GPU 之間計算浮點數能力差距越來越大，以此圖最高階的圖形處理器 GTX680 為例，它擁有 8 個 Multiprocessors (MP)，每個 MP 有 192 個核心 (cores)，共有 1536 個核心，可以同時處理 1536 個執行緒 (threads)。雖然 GPU 每個核心的時脈和處理能力都遠低於 CPU，但總體來說 GPU 有著 CPU 更高的浮點數計算能力。

由於 GPU 發展已逐漸成熟，在 EDA(Electronic Design Automation) 領域中，許多研究也利用 GPU 強大的平行運算能力，將許多問題加以平行化，藉以加速求解 [12-17]。本論文針對 [11] 所提出的演算法加以分析，並將其可平行化部分實作在 GPU 上，藉此加速求解的過程。

在接下來的小節中，第二節將介紹本論文相關之背景知識；第三節則對現有演算法作分析，並加以平行化和最佳化；第四節為實驗結果及分析，第五節是總結本論文研究成果與未來研究方向。

2. 研究背景

本節將要介紹本篇論文之相關背景知識，包含不完全描述函數之布林比對、漸進式學習之方法、CUDA 程序模型及 CUDA 相關硬體架構等。

2.1 不完全描述函數之布林比對

● 不完全描述函數

一個不完全描述函數，包含三種集合，on-set (f^{on})、off-set (f^{off}) 以及 don't care-set(f^{dc})，且 don't care-set 不可為空集合。本論文中使用 $f=(f^{on},f^{off})$ 來表示一個不完全描述函數。

● 函數一致性

兩個具有相同輸入集合的不完全描述函數 f 與 g，若同時滿足 $f^{on} \cap g^{off} = \emptyset$ 與 $f^{off} \cap g^{on} = \emptyset$ 這兩個條件時，我們稱此兩個函數具有一致性 (*consistence*)。用 $f \simeq g$ 表示之。

給定兩個不完全描述函數 $f(X)$ 與 $g(Y)$，其中 $X=\{x_1,x_2,...,x_n\}$ 而 $Y=\{y_1,y_2,...,y_n\}$。對不完全描述函數 f 與 g 作布林比對，對其輸入集合作排列組合及輸入輸出的相位設值後，檢查兩個函數是否滿足一致性。換言之，不完全描述函數之布林比對就是找到一組對應關係 ψ，使得每一個 x_i 都能找到唯一對應的 $y_j(\bar{y_j})$，且滿足 $\tilde{f} \simeq g$，其中 $\tilde{f}=f(\psi(X))$。

● 以多值函數作布林比對

給定兩個布林函數 $f(X)$ 與 $g(Y)$，其中 $X=\{x_1,x_2,...,x_n\}$ 而 $Y=\{y_1,y_2,...,y_n\}$，我們可以用一個多值函數 F 表示可能的輸入對應關係：$F: P_1 \times P_2 \times ... \times ... \times P_n \rightarrow B$ (1)

其中 $P_1=P_2=\cdots=P_n=\{1,2,...,n,n+1,...,2n\}$，而 $B=\{0,1\}$。F 中的第 i 個變數對應到 f 中的 x_i 輸入，P_i 用以表示 x_i 所有可能對應到的 y_j 或 $\bar{y_j}$。每個 x_i 變數的數值 j 代表一種輸入對應關係，其關係如下，當 $j \le n$ 時，x_i 對應到 y_j；當 $j > n$ 時，代表 x_1 對應到 $\bar{y}_{j-n}$。如果不考慮輸入相位設值 (*input phase assignment*) 的情況時，則可以簡化成 $P_i=\{1,2,...,n\}$，每一種對應關係被稱為最小項 (*minterm*)。

範例 2.1：給定具有三個輸入之集合 $X=\{x_1,x_2,x_3\}$ 與 $Y=\{y_1,y_2,y_3\}$。多值對應函數 F 其定

義為，$F:N\times N\times N\rightarrow B$，其中 $N=\{1,2,3,4,5,6\}$，而 $B=\{0,1\}$。令 ψ 為其中一組對應關係最小項為 x_1 對 y_3，x_2 對 y_1，x_3 對 $\bar{y}_2$。最小項 $x_1^{\{3\}}\ x_2^{\{1\}}\ x_3^{\{5\}}$ 可用以表示這組對應關係。

當一個最小項滿足一對一且映成之對應關係，就稱之為合法的 (*feasible*) 最小項，否則就是不合法的 (*infeasible*)。例如，(1,1,2) 與 (1,4,3) 就不是合法的最小項，因為它的 x_1 與 x_2 都對應到 y_1(或 $\bar{y}_1$)，不滿足一對一且映成之對應關係。當一個多值函數只包含所有合法對應的最小項，稱之為完全函數 (*totality function*)。

● 部分設定 (partial assignment) 規則

如前文所提，兩個不完全描述函數之布林比對，是建立在布林函數的一致性關係上。當 $f(X)$ 與 $g(Y)$ 作比對時，如果存在一組對應關係 ψ，使得 $g\simeq f$，其中 $\tilde{f}=f(\psi(X))$ 我們稱之為比對成功。假設有兩個積項 u_i 與 v_j，分別存在於 $f^{on}(f^{off})$ 與 $g^{off}(g^{on})$。根據函數一致性的定義 ($\tilde{f}_{on}\cap g_{off}=\emptyset$ 且 $\tilde{f}_{off}\cap g_{on}=\emptyset$)，在 [11] 中，提出三個部分設定 (*partial assignment*) 產生的規則如下：

Rule 1：每一個 $x_1\in 1\text{-}\ lit\ (u_i)$，及每一個 $y_j\in 0\text{-}\ lit\ (v_j)$，會產生一組部分對應關係（*partial mapping*）為 x_1 對應到 y_j。同理，每一個 $x_1\in 0\text{-}\ lit\ (u_i)$，和 $y_j\in 1\text{-}\ lit\ (v_j)$ 也會產生一組部分對應關係。

Rule 2：每一個 $x_1\in 1\text{-}\ lit\ (u_i)$，及每一個 $y_j\in 1\text{-}\ lit\ (v_j)$ 會產生一組部分對應關係為 x_1 對應到 $\bar{y}_j$。同理，每一個 $x_1\in 0\text{-}\ lit\ (u_i)$，和 $\bar{y}_j\in 0\text{-}\ lit\ (u_j)$ 也會產生一組部分對應關係。

Rule 3：每一組 (u_i, v_j) 積項配對至少存在一組部分對應關係滿足所有對應。

Rule 1 所產生之部分對應關係必定滿足函數一致性，而 **Rule 2** 則是在考慮輸入設值（*input phase assignment*）的情況下所產生之部分對應關係，故對應到的值都和 **Rule 1** 相反。經由 **Rule 1** 與 **Rule 2** 可產生合法對應關係，使其滿足一致性。**Rule 3** 則說明對所有的積項配對均須同時滿足一致性。根據以上三個規則，可以保證在滿足函數一致性的前提下，搜尋是否存在對應關係 ψ，使得 $\psi(u_i)\cdot v_j=0$，換言之，就是使得 $\tilde{f}=f(\psi(X))$ 與 g 滿足一致性。ψ 為一多值 *cover*，記作 $MvCube\ (u_i,v_j)$。如果考慮到輸出設值 (*output phase assignment*) 的情況下，則要取 $f^{on}(f^{off})$ 中的積項 u_i 與 $g^{on}(g^{off})$ 中的積項 v_j 找部分對應關係。

範例 2.2：針對輸入集合 $X = \{x_1, x_2, x_3\}$ 而 $Y = \{y_1, y_2, y_3\}$，給定兩個積項 $u = x_1 \bar{x}_2 x_3$ 與 $v = \bar{y}_1 \bar{y}_2 y_3$。而多值函數 $F: P_1 \times P_2 \times P_3 \rightarrow B$，有三個輸入，其中 $P_1 = P_2 = P_3 = \{1,2,3,4,5,6\}$。0 - lit($u$) = $\{x_2\}$，1 - lit(u)=$\{x_1, x_3\}$，1 - lit(v) = $\{y_3\}$，0 - lit(v) = $\{y_1, y_2\}$。根據 **Rule 1** 與 **Rule 2** 可得到，$MvCube\ (u,v) = x_1^{\{1,2,6\}} + x_2^{\{3,4,5\}} + x_3^{\{1,2,6\}}$。

2.2 漸進式學習之方法

在機器學習 (*machine learning*) 的搜尋系統中可分為兩個階段，一個為學習階段 (*learning phase*)，另一個為應用階段 (*applying phase*)[17]。在學習階段中會將學習到的知識 (*knowledge*) 儲存，並與所學到的知識系統整合，在應用階段就可以利用所學的知識，以避免重複的計算，並可大幅的縮小搜尋空間，加快搜尋的速度。漸進式學習之布林比對演算法就是利用此概念，將原本之布林比對演算法 [9] 作改良，以下對此演算法作說明。

給定一個 *minimum supercube* ξ 及輸入 x_i，符號 *Map1*(ξ ,x_i) 和 *Map0* (ξ ,x_i) 分別代表 x_i 對應 y_j 及 x_i 對應 $\bar{y}_j$ 對應關係之變數集合。假設積項配對為 (u_i ,v_j)，u_i 和 v_j 分別在 f^{on} 和 g^{off} 中，利用所學習到 *minimum supercube* ξ 與產生部份對應關係，以下為改良 **Rule 1** 及 **Rule 2** 之兩個規則：

Rule 4：每一個 1 - *lit*(u_i) 中的 x_i 可對應 0 - *lit*(v_j) ∩ Map1(ξ, x_i) 中的 y_j 以產生一組部分對應 關係。同理，每一個 0 - *lit*(u_i) 中的 x_i 可對應 1 - *lit*(v_j) ∩ *Map*1(ξ, x_i) 中的 y_j 以產生一組部分對應關係。

Rule 5：每一個 1 - *lit*(u_i) 中的 x_i 可對應 1 - *lit*(v_j) ∩ *Map* 0 (ξ, x_i) 中的 $\bar{y}_j$ 以產生一組部分對應關係。同理，每一個 0 - *lit*(u_i) 中的 x_i 可對應 0 - *lit*(v_j) ∩ *Map* 0 (ξ, x_i) 中的 $\bar{y}_j$ 以產生一組部分對應關係。

● 漸進式學習之布林比對演算法

在作布林比對之前，會先計算特徵值，移除掉不可能的對應，以產生初始的對應關係 ψ。接著執行 *Partial-Mapping* 演算法，求得 f^{on} 與 g^{off} 所產生之對應關係。如果是對不完全描述函數，則要做兩次 *Partial-Mapping* 演算法，第一次為 f^{on} 與 g^{off}，第二次為 f^{off} 與 g^{on}，以求得最後的解。如圖 1 所示，為 *Partial-Mapping* 演算法，令 ψ 為目前

所有的合法對應關係，一開始先求得 ψ 的 *supercube* ξ，接著將 f 和 g 裡的積項 u_i 與 v_j 由大到小作排序 (*don't care* 多的積項排前面)。此經驗法則為 *don't care* 多的積項所產生出的對應關係空間會比較小，如此一來可以快速的縮小搜尋空間。ω 為使用 Rule 4 及 Rule 5 經由 u_i 與 v_j 積項配對所產生的部分對應關係，如果 ω 是空集合或者包含於 *MCDB* (*MvCube Database*) 中則跳過下述之步驟，否則就會加入 *MCDB* 中。接著將 ω 和 ψ 作 *AND* 運算 ($\psi = \psi \cap \omega$) 以縮小其搜尋空間，並檢查其是否為合法的對應關係，如果是不合法的對應關係就會被移除，之後進入 *Reduction* 程序，將多餘與被包含的對應關係移除，最後再更新 *supercube* ξ，將所學的知識作整合。

```
Algorithm Partial-Mapping(f, g, ψ)
Input: f, g are Boolean covers to be matched
Output: ψ or ∅
Began
    ξ = ξ(ψ);   MCDB = ∅;   /* Initial Learning */
    Sort the cubes in f and g by the cube size;
    for each cubes u_i ∈ f do
        for each cubes v_j ∈ g do
            ω = SmartMvCube(u_i, v_j, ξ);          /*Apply Phase*/
            if (ω is ∅ or ω ≼ MCDB) continue;    /*Apply Phase*/
            Add ω into MCDB;   /*Learning Phase*/
            if (ψ ⊈ ω) then
                ψ = ψ ∩ ω;      /*AND operation*/
                Reduction(ψ);
                if (ψ is ∅) return ∅;
                ξ = ξ(ψ);      /*Learning Phase*/
            endif
        endfor
    endfor
    return ψ;
End
```

圖 1：漸進式學習之 *Partial-Mapping* 演算法 [11]

2.3 CUDA 程序模型

CUDA (*Compute Unified Device Architecture*) 為 *NVIDIA* 所提出之運算架構，其構想為將程式中只能序列化 (*serial*) 的部分交由 CPU 執行，計算量龐大且能平行化 (*parallel*) 的部分交由 GPU 處理，以加快其運算速度。

CUDA 程式會在兩個不同的平台上運行 [21]，它會將 *CPU* 當作主機端 (*host*)，將 *GPU* 當作裝置端 (*device*)。在裝置端執行的程序稱之為 *kernel*。在執行 *CUDA* 程式時，會先在主機端作初始設定的動作，並將資料傳輸給裝置端作計算，當裝置端的程序運行完畢，再將計算的結果傳回主機端。

● CUDA 執行緒階層架構 (如圖 2)

1. 執行緒 (*thread*)：在 *GPU* 的運行中，最小的基本單位。
2. 區塊 (*block*)：由若干個執行緒組成，*kernel* 在執行時是以區塊為單位。
3. 網格 (*grid*)：由若干個區塊所組成，實際上就是 *kernel* 本身。

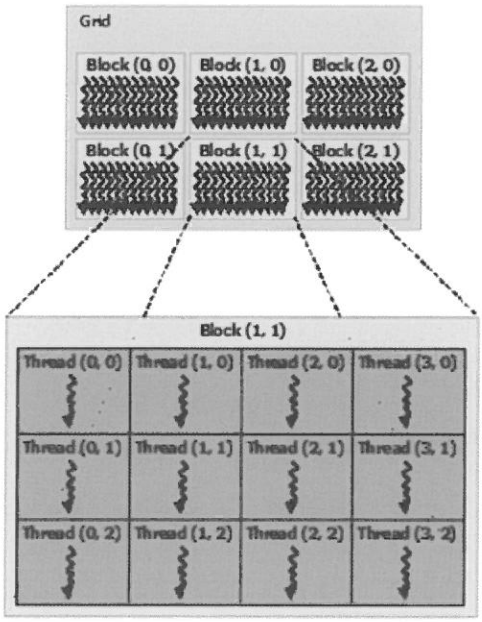


圖 2：*CUDA* 執行緒階層架構圖

● Warp

CUDA 定義 *warp size* 為 32，換言之，一個 *warp* 會有 32 個執行緒。在 *kernel* 運行的時候，表面上是以區塊為單位做運算，實際上 *CUDA* 會將執行的區塊以 32 個執行緒為單位切割成多個 *warp*，切割的分配和順序是以執行緒的索引值作劃分，假

設有三個執行緒的索引值分別為 5、35 及 80，這些執行緒就分別在第 0、1 及 3 個 *warp* 裡。撰寫 *CUDA* 程式，如果使用到條件分支 (*branch*) 的動作要特別注意。當一個 *warp* 裡的執行緒分支的方向不同時，*GPU* 會將所有的分支任務都執行，稱為 *warp divergence*，效能就會大幅下降。

● CUDA 串流

對 *CUDA* 而言，每一個 *kernel* 都是一個任務 (*task*)，假設有一個任務，資料無相依性，*CUDA* 提供了串流機制 (*Stream*)，可以讓這個任務更有效率的運行。*CUDA* 串流將一個任務運行的所有動作視為一個事件 (*Event*)，我們可以將這任務中所有的動作，例如資料複製傳輸、*kernel* 運行等，分為 *n* 段，再將這些動作放進一個特殊的佇列 (*queue*) 中。這樣就可以一邊傳輸，一邊運算。要運行串流，還需使用到 *page-lock host memory*。一般而言，都是使用 *malloc*() 配置記憶體，*CUDA* 提供了一個機制 *cudaHostAlloc*()，讓 *GPU* 也可以配置主機端的記憶體，利用此機制配置的記憶體會對作業系統註冊，可保證此記憶體，不會被作業系統作分頁的動作，以確保執行串流任務時資料傳輸之速度。

2.4CUDA 硬體架構

CUDA 的記憶體模型由暫存器 (*Register*)、區域記憶體 (*Local Memory*)、共享記憶體 (*Shared Memory*)、全域記憶體 (*Global Memory*)、常數記憶體 (*Constant Memory*)、材質記憶體 (Texture Memory) 所組成 (如圖 3 所示)，每一種記憶體都有不同的特性，以下僅針對我們所使用的記憶體加以介紹。

其中，全域記憶體的儲存空間最大，但存取速度也最慢，就效能而言，必須要盡量減少其讀寫次數。但是在做 *GPU* 運算的時候，一定要先將資料複製到全域記憶體，接著 *GPU* 再從全域記憶體讀取資料作計算，所以當要計算的資料越大，*GPU* 就要花費更多的時間存取全域記憶體。為了改善記憶體的讀寫速度，*CUDA* 在記憶體的架構上作了一些調整，一般記憶體是採取以列為主 (*row-major*) 的架構，*CUDA* 的記憶體則是採取以行為主 (*column-major*) 的架構。為了能夠快速的讀取資料，*CUDA* 發展出一種記憶體合併存取機制稱之為 *Memory Coalescing*。在 2.3 節有提到，*CUDA* 在運行的時候是以 *warp* 為單位，而 *warp size* 為 32，所以在讀取記憶體的時候，是以 128 位元組 (4 Bytes×32=128 Bytes) 作一個區段，再以寬度為 128 位元組的 *L1* 快取去存取全域

記憶體。當 *warp* 內的執行緒存取的資料剛好在全域記憶體的同一個區段上，那麼此 *warp* 只要讀取一次 *L1* 快取就可存取完畢。

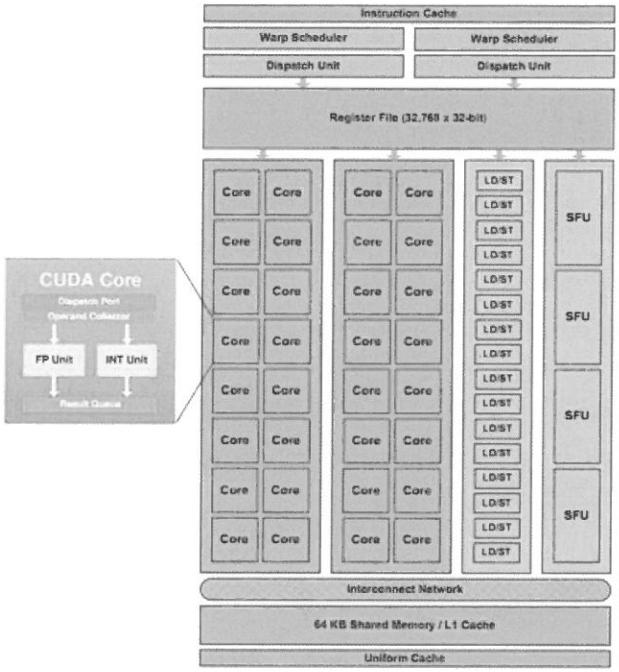


圖 3：*CUDA* 硬體架構架構圖

3. 平行化之布林比對演算法

本節將對我們所採用的漸進式演算法加以分析並提出權重計算排序的方法。接著將演算法直接平行化，並提出三個平行程序最佳化的方法。

3.1Weighted Partial Mapping 演算法

針對原本的演算法我們提出一個方法加以修正，就是先將所有的 (u_i, v_j) 積項配對計算出一個權重 (*weight*)，再依照權重做排序，權重高的積項配對先產生部分對應關

係，如此一來便可確保最先產生的對應關係空間都會比較小。權重的計算規則如下所示：

Rule 1：積項之權重為各字元的權重加總。字元為 0 或 1 時，權重為 1，字元為 *don't care* 時，權重為 2。

Rule 2：積項配對之權重，為兩個積項權重相乘之結果。

範例 3.1：給定兩個 3- 輸入函數 $f = x_2 + x_1 \bar{x}_3$ 而 $\bar{g} = \bar{y}_1 \bar{y}_2 + y_1 \bar{y}_2 y_3$。$u_i$ 和 v_j 分別為 f 和 $\bar{g}$ 中的積項，$u_1 = x_2$、$u_2 = x_1 \bar{x}_3$、$v_1 = \bar{y}_1 \bar{y}_2$ 及 $v_2 = y_1 \bar{y}_2 y_3$。各個積項權重為 $w(u_1) = 5$，$w(u_2) = 4$，$w(v_1) = 4$ 及 $w(v_2) = 3$。4 組積項配對權重分別為 $w(u_1,v_1) = 20$，$w(u_1,v_2) = 15$，$w(u_2,v_1) = 16$ 及 $w(u_2,v_2) = 12$。故得到以積項配對權重由大到小之排序為 (u_1,v_1)、(u_2,v_1)、(u_1,v_2) 及 (u_2,v_2)。

● Weighted Partial Mapping 演算法之分析

我們對漸進式學習之 Weighted Partial Mapping 演算法作分析，並歸納出以下幾點。

1. 演算法相依性高，執行緒間無法直接將所學之知識加以應用，故不可將 *Weighted Partial Mapping* 演算法直接平行化。
2. 動態配置記憶體效能不好，故無法將 *Weighted Partial Mapping* 演算法直接平行化。
3. 每個執行緒暫存器大小有所限制，故無法將 *Weighted Partial Mapping* 平行化。
4. 應針對瓶頸時間之程序 AND 運算與 Reduction 作平行化。
5. 有大量條件分支的程序，會造成 *warp divergence*，故應將合法對應關係檢查分類再平行化。

3.2 部分程序步驟之直接平行化

● AND 運算之平行化

當一個對應關係滿足一對一且映成，就稱之為合法的對應關係，否則就是不合法的對應關係，不合法對應關係的檢查可分為兩種類型：

1. 當其中一個 x_i 輸入無法對應到任何 y_j 輸入時，稱為不合法的對應關係。例如：$(\emptyset,1,2)$，代表 x_1 輸入無法對應到任何 y_j 輸入，稱之為 *empty-type*。
2. 當 x_i 輸入與 y_j 輸入之對應關係不滿足一對一，稱之為不合法的對應關係。例如：$(1,1,2)$，代表 x_1 和 x_2 同時對應到 y_1 輸入，不滿足一對一的關係，稱之為 *incomplete-type*。

Empty-type 之對應關係，只要其中一個輸入 x_i 無法對應到 y_j 即可快速檢查出來。然而 *incomplete-type* 之對應關係需將每一個輸入都檢查才可以判定是否為不合法之對應關係，在實作上會產生大量的 *warp divergence* 情況，平行化效果不佳。故我們把對應關係的檢查分為兩個階段，第一階段用 *GPU* 檢查 *empty-type* 對應關係，第二階段再用 *CPU* 檢查 *incomplete-type* 對應關係。*AND* 之平行化流程如圖 4 所示。

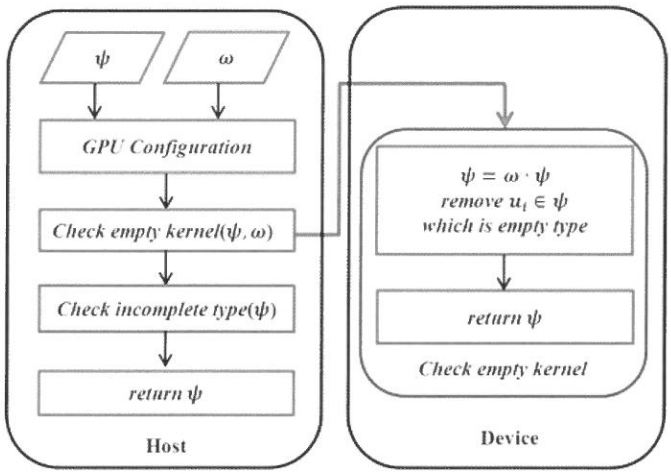


圖 4：*AND* 運算平行化之流程圖

令 ψ 為目前之對應關係集合，ω 為所產生出之部分對應關係。

Step 1：在主機端先作相關參數之設定。計算記憶體大小，配置全域記憶體，並將所需用到的常數複製到常數記憶體。

Step 2：將目前對應關係 ψ 與 partial mapping ω 複製到裝置端，利用 *GPU* 對 ψ 與 ω 作 *AND* 運算並檢查 *empty-type* 不合法之對應關係。

Step 3：將結果傳回主機端，用 *CPU* 檢查 *incomplete-type* 不合法之對應關係。

Step 4：將合法對應關係 ψ 回傳。

任務分配

令 ψ 為上一階段所產生的對應關係集合，ω 為積項配對產生的部分對應集合。假設 ψ 與 ω 的數量分別為 m 與 n，作 *AND* 運算後產生的多值積項數量為 $m \times n$。先把 ψ 與 ω 複製到裝置端，接著將執行緒數量設為 m 個，每一個執行緒都會執行 n 次 *AND*

運算與 *empty-type* 之檢查，最後再將結果回傳到主機端，檢查 *incomplete-type*。如圖 5 所示，每一個方格代表 *AND* 運算所產生的一組對應關係，箭號代表執行緒，在第 1 次的時候，ψ 裡面所有的多值積項 u_i 與 ω 的第 1 組部分對應 v_0，作 *AND* 運算與合法對應關係之檢查，第 2 次的時候，則是所有的 u_i 與 ω 的第 2 組部分對應 v_1 重複做相同之動作，其他依此類推。

● Reduction 之平行化

本節將對 *Reduction* 演算法作進一步的分析。如圖 6 所示，令 ψ 為目前所有可能對應關係之集合，u_i 為 ψ 裡的積項。首先將積項 u_i 由大到小作排序 (*don't care* 多的排在前面)。最後檢查小的積項是否有被大的積項包含，如果有被包含則移除。

Step 1：在主機端先將 ψ 裡的積項 u_i 由大到小作排序。

Step 2：對參數作相關設定。計算記憶體大小，配置全域記憶體，並將所需用到的常數複製到常數記憶體。

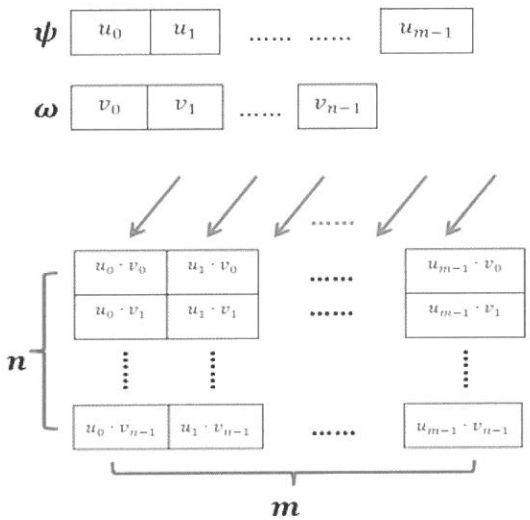


圖 5：合法對應關係檢查之任務分配

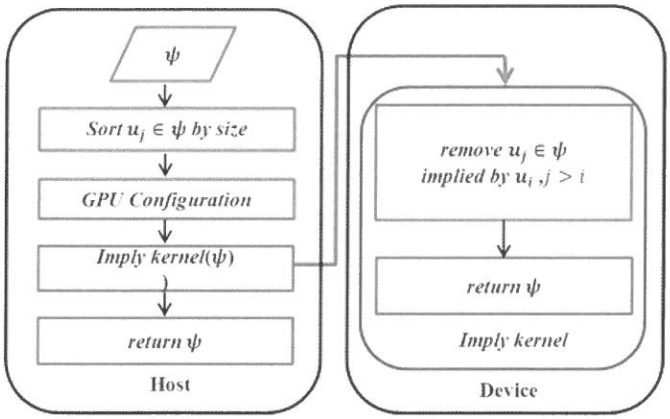


圖 6：Reduction 平行化之流程圖

Step 3：將 ψ 複製到裝置端，利用 *GPU* 檢查 ψ 裡是否有被包含的積項。
Step 4：將計算結果傳回主機端，並將 ψ 作整合。

任務分配

令 ψ 為現階段所產生的所有對應關係 (多值積項)，其數量為 m 個。為了檢查其中是否有多餘之多值積項。首先將 ψ 內的多值積項由大到小作排序。接著把 ψ 複製到裝置端，並設定執行緒數量，檢查其中的多值積項是否有被包含，再將結果回傳到主機端。如圖 7 所示，每一個方格代表一個多值積項，如果有 m 個多值積項，則需要開 m-1 個執行緒，最多需要比較 m-1 次。在第 1 次時，會將第 1 個多值積項 (u_0) 與其他 m-1 個多值積項作運算。第 2 次時，會將第 2 個多值積項 (u_1) 與其他 m-2 個多值積項作運算，其他依此類推。

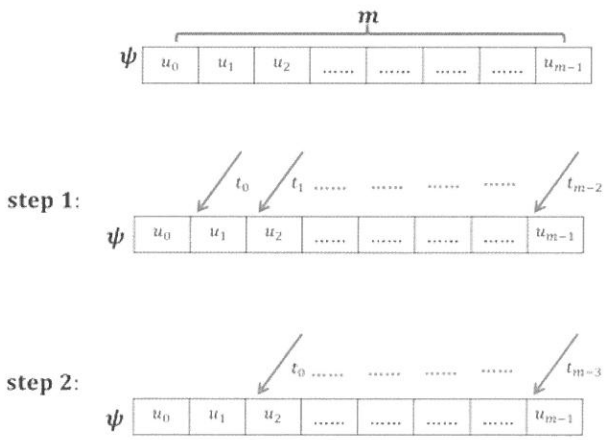


圖 7：多餘對應關係檢查之任務分配

3.3 最佳化之平行化程序

前文所提到之平行化程序會有三個問題：

1. 在平行化 AND 運算中，由於原本資料結構設計，導致檢查 empty-type 對應關係效能不好。
2. 在平行化 AND 運算中，處理大電路會有全域記憶體不足之情形。
3. 在平行化 Reduction 程序中，由於資料儲存之方式，導致記憶體讀取效率不好。

對於這些問題，本節將提出相關之最佳化技術加以解決。

● Empty-type 積項檢查之資料結構轉換

在實作方面，我們是利用 *bit-string* 來表示其對應關係。使用 *bit-string* 的好處是可以用最少的空間儲存對應關係，然而在檢查 *empty-type* 時，會檢查每一個輸入 x_i 與 y_j 之對應關係，在擷取每一個對應關係時會使用到許多的條件判斷。由於 *GPU* 使用的是 *SIMT* 的架構，當有條件判斷時會造成 *warp divergence*，越多的 *warp divergence* 執行的效能就會越慢，為了改善此問題須將資料儲存方式作調整。

資料儲存方式之調整如下，將每個輸入 x_i 對應到 y_j 之關係以 *word* 為單位作劃分。給定兩個輸入數目為 3 的函數 f 與 g，ψ 代表對應關係之集合，假設今天有一組對應關係（多值積項）t 為 (1,3,2)，在不考慮 *input phase assignment* 的情況之下，以 *bit-string* 之形式可表示為 (100 001 010)，而其中 100、001 與 010 分別儲存在三個 *word* 裡面，如圖 8 所示。這樣的調整，會增加儲存空間，但是在作 *empty-type* 檢查時只需判斷 *word* 的值是否為 0 即可，可大幅降低檢查 x_i 對應關係所花費之時間。

● 解決記憶體大小限制之問題 - 利用 Stream 機制降低記憶體使用量

在上一節介紹了合法對應關係檢查之平行方法。然而在對大型布林函數求解之過程，往往會面臨記憶體不足之問題。針對此問題，可以利用 *CUDA Stream* 加以解決。如圖 9 所示，假設目前的對應關係 ψ 有 m 個多值積項，新產生的部分對應關係 ω 有 n 個多值積項 v_i，我們可以將 ω 切成 n 等分，第一次將 v_0 複製到裝置端，開 m 個執行緒執行 *kernel*，結束將結果傳回主機端，第二次將 v_1 複製到裝置端，重複上述步驟直到結束。如此一來，空間複雜度便由 $O(m \times n)$ 降低至 $O(m)$。

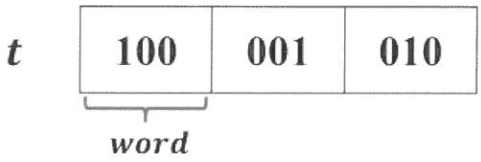


圖 8：改良的 bit-string 表示式

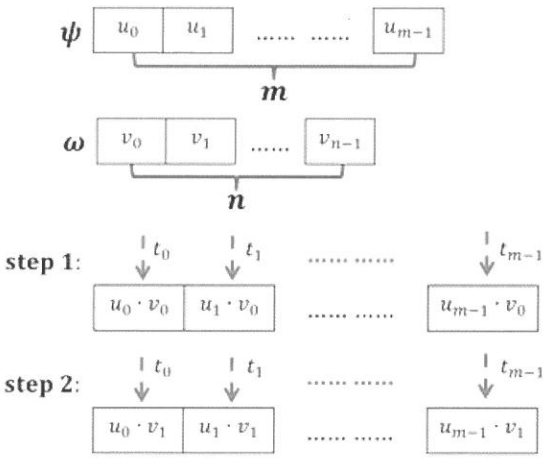


圖 9：利用 Stream 機制降低記憶體使用量

● 全域記憶體最佳化 - 將資料由 row-major 轉換成 column-major

在 2.4 節提到，為了能夠快速的讀取資料，*CUDA* 發展出一種記憶體合併存取機制稱之為 *Memory Coalescing*。只要善用此機制便可用最少的次數讀取全域記憶體。由於 *CPU* 與 *GPU* 記憶體的架構不同，為了提高 *GPU* 中全域記憶體的讀取速度，必須先將對應關係 ψ 作轉置，如圖 10 所示，意即將資料由 *row-major* 轉換成 *column-major*，接著將資料複製到裝置端，再運行 *kernel*，最後將結果傳回主機端。

Row-major 記憶體之讀取情形，如圖 11 所示。紅色的部分代表 1 個 *warp* 讀取的記憶體大小 (128Bytes)，要讀取的多值積項 u_i 大小為 n 個 word，當要讀取每個多值積項的第 1 個變數 (綠色方格部分) 時，以 row-major 之形式必須要讀取很多次才能完成。然而，若將資料轉置成 column-major 的形式，如圖 12 所示，所有多值積項的第 1 個變數，都會擺在連續的位置，故只需讀取 1 次就可完成，依照 *column-major* 之形式，可以用最少的次數讀取全域記憶體，以達到最佳化進而提升整體效能。

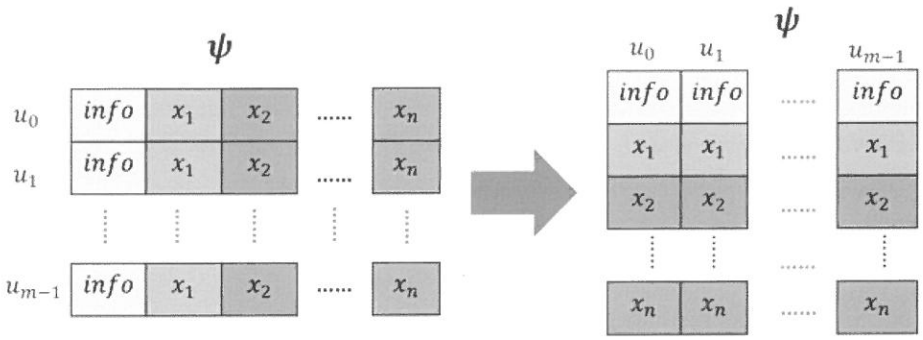


圖 10：row-major 轉 column-major

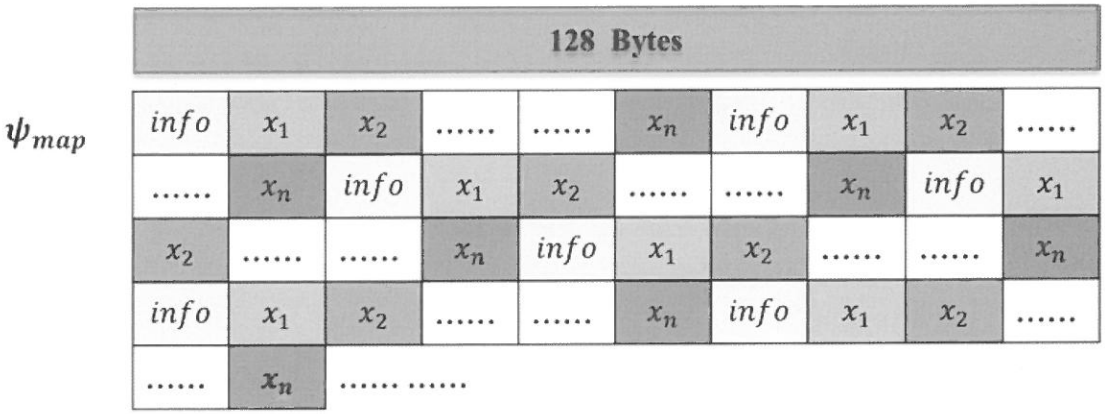


圖 11：row-major 記憶體讀取情形

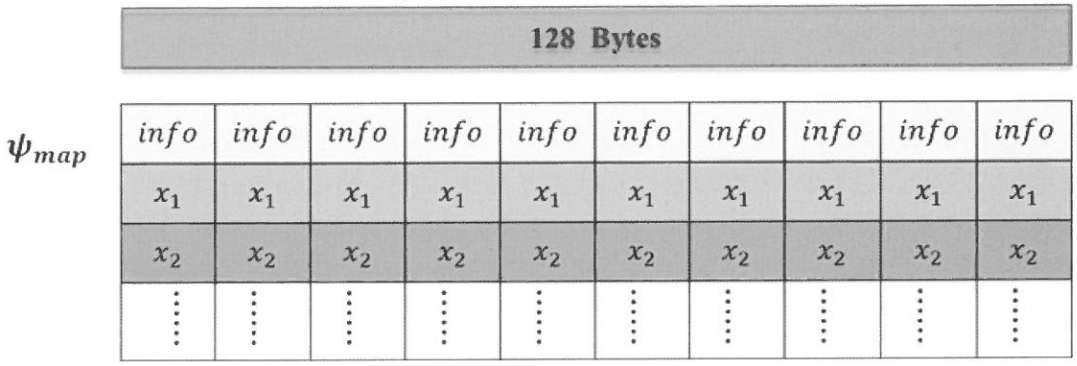


圖 12：column-major 記憶體讀取情形

4. 實驗結果

本節將探討我們所提出之技術的相關實驗結果。包括 *Weighted Partial Mapping* 演算法與 *Partial Mapping* 演算法之比較、在布林比對時將 *AND* 運算與 *Reduction* 程序直接平行化與最佳化之比較。由於現有相關論文與實驗數據並不多，所以並沒有與其他論文比較。

4.1 實驗平台與說明

我們的實驗是建立在 64 位元架構之主機上，在硬體方面，具有兩個六核心的 *INTEL Xeon E5-2620 2.0 GHz* 處理器，系統記憶體共有 32GB。顯示卡為 *NVIDIA Tesla C2075*，全域記憶體大小為 5GB。作業系統採用 *Linux Cent-OS 6.4 x86_64*，軟體平台為 *UC Berkeley* 所開發的二階邏輯最佳化工具 *Espresso* 套件和 *NVIDIA* 釋出的 *CUDA 5.0* 開發工具。

本實驗使用兩組電路集合，第一組電路為 *MCNC benchmark*，此電路為二階邏輯電路，由於這些電路在布林比對的過程當中產生之對應關係 (多值積項) 比較少，無法凸顯平行化演算法之優點，所以我們用隨機的方式產生另一組不完全描述函數之電路，這些電路會在布林比對的過程中產生大量的多值積項，以凸顯平行化技術所提升之效能。

4.2 完全描述函數作布林比對之實驗結果比較

第一個實驗是比較 *Weighted Partial Mapping* 演算法與 *Partial Mapping* 演算法對於布林比對之影響，為了突顯兩種演算法之差異性，本實驗不使用特徵值之技術。我們總共測試了 81 組電路，輸入大小範圍為 3 到 199，時間限制為 8000 秒，若超過限制時間則以 8000 秒計算。相關統計資訊如表 2 所示，第一欄代表執行時間的總和，第二欄代表執行時間之倍率，第三欄代表求出解之電路與所有電路之個數，**Old** 與 **New** 分別表示 *Partial Mapping* 演算法與 *Weighted Partial Mapping* 演算法，由實驗結果顯示在限制時間之內，使用權重計算作排序之方法，可以多解 1 組電路，而且整體執行速度會比原本之方法快 2.6 倍左右。

4.3 改變平均上傳頻寬的影響

第二個實驗是 *CPU* 與 *GPU* 對不完全描述函數作布林比對之比較。在本實驗使用 *Weighted Partial Mapping* 演算法與特徵值之技巧。不完全描述函數產生之方式是將 *MCNC benchmark* 電路以隨機的方式從 *on-set* 與 *off-set* 中分別移除 10%、20% 與 40% 的積項。

表 2：MCNC benchmark 未使用特徵值之布林比對相關統計

Total time(sec)		Ratio	Solved/All	
Old	New	Old/New	Old	New
21978.28	8360.43	>2.63	80/81	81/81

● MCNC benchmark 電路不完全描述函數之布林比對

在執行時間有三個欄位，*CPU* 代表原本之演算法執行結果，*NaiveGPU* 是將演算法中可平行化之部分直接在 *CPU* 上執行，OURS 是我們所提出之最佳化方法在 *CPU* 上運行的結果，*Ratio* 是將 *CPU* 與 *OURS* 之執行時間相除比較加速之倍率。

表 3、表 4 與表 5 分別列出移除 10%、20% 與 40% 積項的不完全描述函數布林比對平行化之相關統計資訊。我們分別對 *AND* 運算與 *Reduction* 以及整體執行時間做分析。移除越多積項的函數其搜尋空間會越大，所產生的對應關係也會越多，執行緒處理的數量也隨之提升。以 *AND* 運算來說，由於發生 *warp divergence* 比例很高，平行化的效果不好。而針對 *Reduction* 而言，最佳化之程序會比直接平行化來的好，以移除 40% 積項之實驗結果，最佳化的執行時間會比直接平行化少一半以上。整體而言，最佳化之效能會比直接平行化快上 1 倍。

● 大型電路不完全描述函數之布林比對

根據上節之實驗結果可知，平行化之加速與否，與比對過程中產生之對應關係 (多值積項) 數量有關。為了突顯我們所提出的方法之效果，本節將使用工具隨機產生大量輸入之 *PLA* 電路進行實驗。電路產生方式是分別將一些電路，隨意加入 0、1 或 *don't care* 以加大電路的輸入數目。每一個電路都會產生輸入大小分別為 50、60、70、80、90、100、150、200 的新電路，由於 *GPU* 硬體架構的限制，將記憶體大小限制在

1GB，在布林比對過程中對應關係 (多值積項) 大小超過 1GB 則結束比對程序。比對時間限制在 40000 秒，若超過 40000 秒則結束比對。

表 3：移除 10% 積項之布林比對相關統計

Procedure	Time(sec)			Ratio	
	CPU	NaiveGPU	OURS	CPU/NaiveGPU	CPU/OURS
AND	24.90	53.01	46.04	0.47	0.54
Reduction	2144.44	426.51	422.16	5.03	5.08
Total	2240.53	555.58	544.41	4.03	4.12

表 4：移除 20% 積項之布林比對相關統計

Procedure	Time(sec)			Ratio	
	CPU	NaiveGPU	OURS	CPU/NaiveGPU	CPU/OURS
AND	29.63	67.86	49.45	0.44	0.60
Reduction	2498.78	715.57	527.50	3.49	4.74
Total	2586.24	845.53	638.99	3.06	4.05

表 5：移除 40% 積項之布林比對相關統計

Procedure	Time(sec)			Ratio	
	CPU	NaiveGPU	OURS	CPU/NaiveGPU	CPU/OURS
AND	68.75	126.04	83.74	0.55	0.82
Reduction	2781.61	1254.66	479.23	2.22	5.80
Total	3153.41	1688.64	870.23	1.87	3.62

本實驗有三組執行結果分別是，*CPU*、*NaiveGPU* 及 *OURS*，其結果如表 6 所示，*#in* 代表電路輸入數目，執行時間為 3 組電路比對時間的加總，*Speed up* 欄位代表與 *CPU* 執行時間相除之倍率，由結果我們可以發現當輸入小於 70 時，在記憶體的限制之下，處理的多值積項數量比較大，執行緒數量隨之提升。*NaiveGPU* 方法大約可達到 5 倍的加速，*OURS* 大約可以達到 30 倍的加速。隨著電路輸入越大，處理的多值積項也隨之

減少，加速的情形也下降。當輸入大小為 150 時，我們提出的最佳化方法還是比直接平行化快了 2.4 倍左右。

表 7 為大型電路各個程序執行時間統計資訊，就 *AND* 運算而言，直接平行化之後效能不好，最佳化後效能雖然有所提升，但與 *CPU* 相比還是不好，原因在於作 *empty-type* 檢查時發生 *warp divergence* 的機率很高，假設一個 *warp* 中有 1 個多值積項不合法對應的變數與其他 31 個不同時，其他 31 個執行緒都要等那 1 個執行緒處理完畢之後才會往下執行。就整體而言，效能無法提升。就 *Reduction* 來說，直接平行化大約可以得到 3.6 的加速，經最佳化後可得到 38 倍的加速。對於比對整體時間而言，直接平行化會加速 3.39 倍，最佳化後可加速 25 倍左右。

5. 結論

本篇論文研究如何在 CUDA 平台上加速大型函數的布林比對問題，在原本以漸進式學習為主的比對演算法之基礎上，首先我們提出了權重計算的方法改良原本的演算法，並將其最耗費時間的程序如 *AND* 運算與 *Reduction* 程序加以平行化，接著提出三個技術將平行程序加以最佳化，主要的技術有資料結構重新設計、利用 CUDA 串流機制減少記憶體的使用量及將記憶體儲存方式作轉置以提升讀取效率。我們測試了許多大型電路，實驗結果證明我們提出的最佳化方法確實有效。

表 6：大型電路之布林比對相關統計

	#in	50	60	70	80	90	100	150	200
CPU	Time(sec)	49346.53	50089.21	10384.33	10991.11	13250.47	4046.18	4970.18	303.42
Naïve	Time(sec)	10603.05	10365.51	1684.08	1938.05	2225.74	2119.81	1798.05	258.9
	Speed up	4.65	4.83	6.17	5.67	5.95	1.91	2.76	1.17
OURS	Time(sec)	1410.24	1696.01	290.77	687.45	901.04	539.71	764.6	152.61
	Speed up	34.99	29.53	35.71	15.99	14.71	7.50	6.50	1.99

表 7：大型電路之布林比對執行時間相關統計

Procedure	Time(sec)			Ratio	
	CPU	NaiveGPU	OURS	CPU/NaiveGPU	CPU/OURS
AND	1096.06	3303.75	1689.38	0.33	0.65
Reduction	145355.71	39464.97	3823.85	3.68	38.01
Total	146566.43	43207.44	5728.39	3.39	25.59

根據實驗結果，我們發現對大型電路做布林比對時，仍然會面臨記憶體不足的情形導致無法完全比對成功。由於硬體上的限制，如果針對不同的顯示卡作修正，顯得不切實際，我們可以開發利用串流機制為主的演算法，解決記憶體不足之問題，以處理更大型的電路。另外，也可對本論文中所採用之布林比對演算法中積項配對之順序加以研究，對於部分的電路而言，以計算權重排序的方式反而會比原本的方法慢上好幾十倍，這表示配對之順序對於不同電路而言會有極大的影響，若能找出最佳的配對順序就可以用最有效率的方式求出解答。上述所討論的這些問題，都可以作為未來研究之方向。

參考文獻

1. J. Mohnke, P. Molitor, and S. Malik, "Application of BDDs in Boolean matching techniques for formal logic combinational verification," International Journal on Software Tools for Technology Transfer, vol. 3, no. 2, pp. 1-10, Springer, May 2001.
2. J. Cong and Y. Y. Hwang, "Boolean matching for LUT-based logic blocks with applications to architecture evaluation and technology mapping," *in Proc. of IEEE Transactions Computer-Aided Design Integrated Circuits Systems*, vol. 20, no. 9, pp. 1077-1090, Sep. 2001.
3. L. Benini and G. De Micheli, "A survey of Boolean matching techniques for library binding," in *Proc. of ACM Transactions on Design Automation of Electronic Systems*, vol. 2, no. 3, pp. 193-226, Jul. 1997.
4. G. Agosta, et. al, "A Unified Approach to Canonical Form-based Boolean Matching," in *Proc. of Design Automation Conference*, pp. 841-846, June 2007.
5. A. Abdollahi and M. Pedram, "A New Canonical Form for Fast Boolean Matching in Logic Synthesis and Verification," in *Proc. of Design Automation Conference*, pp. 379-384, June 2005.
6. G. Agosta, F. Bruschi, G. Pelosi, and D. Sciuto, "A transform-parametric approach to Boolean matching," in *Proc. of IEEE Transactions Computer-Aided Design Integrated*

Circuits Systems, vol.28, no.6, pp. 805-817, Jun. 2009.
7. A. Abdollahi and M. Pedram, "Symmetry detection and Boolean matching utilizing a signature-based canonical form of Boolean function," in *Proc. of IEEE Transactions Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 6, pp. 1128-1137, Jun. 2008.
8. F. Mailhot and G. D. Micheli, "Technology mapping using Boolean Matching and Don't Care Sets," in *Proc. of the Conference on European Design Automation*, pp. 212-216, 1990
9. K. H. Wang and T. T. Hwang, "Boolean Matching for Incompletely Specified Functions," in *Proc. of IEEE Transaction on Computer-Aided-Design of Integrated Circuits and Systems*, Vol. 16. , No. 2, pp. 160-168, Feb, 1997.
10. J. Mohnke, P. Molitor, and S. Malik, "Limits of using signatures for permutation independent Boolean comparison," in *Proc. of ASP-DAC* , pp. 459-464, 1995.
11. K.H. Wang and C.M. Chan, "Incremental learning approach and SAT model for Boolean matching with don' t cares," in *Proc. of ICCAD*, pp.234-239, 2007
12. Kooijman, A. and Vergeest, J., "GPU Implementation of the LFT Shape Matching Algorithm," in *Proc. of Parallel Computing in Electrical Engineering (PARELEC)*, 2011 6th International Symposium.
13. Bertacco, V. and Chatterjee, D., "High performance gate-level simulation with GP-GPU computing," in *Proc. of VLSI Design, Automation and Test (VLSI-DAT)*, 2011 International Symposium.
14. Croix, J.F., Khatri, S.P. and Gulati, K., "Using GPUs to Accelerate CAD Algorithms," in *Proc. of Design & Test, IEEE* (Volume:30 , Issue: 1).
15. Zhang Yuxuan, Xi"an, Wei Tingcun , Kai Yaowen and Fan Xiaoya, "Logic simulation acceleration based on GPU," in *Proc. of Mixed Design of Integrated Circuits and Systems (MIXDES), 2011 the 18th International Conference.*
16. Tsutsui, S. and Fujimoto, N., "Implementation of histogram based sampling algorithm within an EDA scheme with CUDA," in *Proc. of Evolutionary Computation (CEC)*, 2012 IEEE Congress.
17. J. Denzinger, M. Fuchs, C. Goller, and S. Schulz, "Learning from Previous Proof Experience: A Survey," AR-Report AR-99-4, TU München,1999.
18. Rob Farber, "CUDA Application Design and Development," Morgan Kaufmann, 2011
19. Cook Shane, "CUDA programming : a developer's guide to parallel computing with GPUs," Amsterdam ; Boston : Morgan Kaufmann, c2013.
20. Jason Sanders and Edward Kandrot, "CUDA by example : an introduction to general-purpose GPU programming," Upper Saddle River, NJ : Addison-Wesley, 2011.
21. "CUDA C Programming Guide" , http://docs.nvidia.com/cuda/pdf/CUDA_C_Programming_Guide.pdf
22. "CUDA C Best Practice Guide" , http://docs.nvidia.com/cuda/pdf/CUDA_C_Best_Practices_Guide.pdf
23. NVIDIA, *NVIDIA Fermi Compute Architecture Whitepaper,* 2009.

Recevied Oct 23, 2014
Revised Mar 3, 2015
Accepted Mar 10, 2015

Accelerating Boolean Matching for Large Functions on CUDA Platform

Kuo-Hua Wang and Feng-Ming Chang

Department of Computer Science and Information Engineering, Fu Jen Catholic University

Abstract

Boolean matching is to check two Boolean functions are equivalent or not under input permutation and input/output phase assignments. In this paper, we propose a Boolean matching algorithm based on a previously proposed incremental learning approach [11]. Firstly, we analyze the incremental learning approach and then propose a weight computing method to accelerate the matching process. By our analysis, most of the running time is spent on two parts - AND procedure and Reduction procedure. Therefore, we parallelize these two procedures on CUDA (Compute Unified Device Architecture) platform. Directly parallelizing the above two procedures has three issues: (1) Using the original design of data structures results in the performance degradation on AND operations; (2) There is not enough GPU memory while matching large functions; (3) Being unable to implement memory coalescing leads to the reading GPU memory very inefficiently. To solve these problems, we propose three techniques to optimize the native parallel procedure, i.e., redesigning the data structures to improve the AND procedure performance, using CUDA stream technique to reduce the memory usage, and transforming the data storage to realize memory coalescing. By the experimental results, the weight computing method can acquire 2.6X performance improvement over the original one. Besides, we took experiments to verify whether our parallel matching algorithm can expedite the matching process or not for large Boolean functions with different input size. Compared to the sequential matching method, the native parallel method and our optimized parallel method can acquire 5X and 30X speed up, respectively. It shows that our parallel Boolean matching algorithm is indeed effective and efficient for matching large Boolean functions.

Key words: Boolean Matching, Incompletely Specified Functions, CUDA, GPU.

khwang@csie.fju.edu.tw ssive99@csie.fju.edu.tw

基於漢明碼之資料隱藏秘密分享法

吳柏霆　黃貞瑛*
輔仁大學資訊工程系

摘　要

資料隱藏的首要目的是希望秘密資料在傳輸過程中，不會被非法接收者所發現，而當合法接收者收取資料時，可完全解讀所隱藏的秘密資料。(t, n) 秘密分享法可達到上述目的，此方法可將秘密資料，例如一張秘密圖片 (Secret Image)，嵌入 n 張毫無關聯的掩護圖片 (Cover Image)，進而產生 n 張偽裝圖片 (Stego Image)。偽裝圖片雖然嵌入秘密圖片資料，卻與掩護圖片相似，外觀未顯露秘密資料。另外每張偽裝圖片儲存不同部分的秘密資訊，當還原秘密時，不需要 n 張掩護圖片，僅需要其中 t 張偽裝圖片，即可還原秘密資料。由於此方法，具有分散儲存秘密資料的特性，因此稱為秘密分享法。本研究主要目的為提出一個 (t, n) 秘密分享法，具有下列特性：(1) 低失真高品質的偽裝圖片，(2) 完全不失真還原秘密資料，(3) 具有驗證偽裝圖片真偽的功能。本研究運用 (7, 4) 漢明碼 (Hamming Code)，將秘密資料嵌入不同掩護圖片像素值的最低有效位元 (Least Significant Bits)，並且使用 $GF(2^8)$，即伽羅瓦二次冪有限域 (Power-of-Two Galois Fields) 運算。實驗結果顯示，本研究提出的方法，的確可大幅降低偽裝圖片的失真，不易被肉眼察覺，達到保護秘密資料的效果；另外本方法具有完全不失真還原秘密的優點，因此適合隱藏任何類型的數位數據，例如文件等。最後本方法還具有驗證偽裝圖片是否遭到篡改的功能，因此也提升秘密資料解碼的正確性與安全性。

關鍵字：資料隱藏、(t, n) 秘密分享法、漢明碼、伽羅瓦二次冪有限域。

* Corresponding author. Email: jihwang@csie.fju.edu.tw

1. 簡介

隨著資訊的蓬勃發展與網路應用的普及快速，經由網路傳遞資料變得非常便捷，但是若資料本身涉及隱私或具有秘密性，在網路傳輸時，可能遭非法資訊竊盜者攫取。因此，將秘密資料隱藏，再經由網路傳輸，應是一個可行的解決方法。換言之，如何保護及隱藏秘密資料的網路傳輸，是值得探討的網路安全問題。

本研究採用 (t, n) 秘密分享法隱藏資料， Shamir [1] 與 Blakley [2] 首先於 1979 年同年分別提出秘密分享的概念，此方法可將秘密資料分成 n 份，而還原時只需其中 t 份，即可完整還原秘密資料；但是若少於 t 份時，便無法還原秘密資料。由於此種分享與傳遞秘密的方法，對於秘密資料損毀或遺失，具有較高的容忍度。因此已有若干學者利用 (t, n) 秘密分享法以保護秘密資料的傳遞，尤其是傳輸秘密圖片 (Secret Image) 時，稱之為 (t, n) 秘密圖片分享法。

早期的 (t, n) 秘密圖片分享法，常將一張秘密圖片資料，分散嵌入於 n 張雜亂而沒有意義的分享圖片中 [3 – 5]，雖然可確保傳遞圖片資料時，不被他人得知真正的秘密圖片，但是無意義的分享圖片，容易引起他人注意，增加分享圖片被竄改的風險，以致無法正確還原秘密圖片，於是便有偽裝技術的產生。改善的秘密圖片分享法，是將一張秘密圖片嵌入 n 張掩護圖片 (Cover Image)，然後產生 n 張偽裝圖片 (Stego Image)，偽裝圖片雖然嵌入秘密圖片資料，外觀卻未顯露秘密圖片資料，與掩護圖片相似，不易引起他人注意，因此增加安全性。

2004 年 Lin 與 Tsai [6]，使用有限域 GF(251) 之運算於 (t, n) 秘密分享法，偽裝圖片總是比秘密圖片大 4 倍，由於模數 251 運算範圍的值為 [0, 250]，而灰階圖片像素值範圍為 [0, 255]，所以無法不失真還原秘密圖片。2009 年 Su, Chen, 和 Lin [7], 利用向量量化 (Vector Quantization) ，得到掩護圖片的編碼表 (Codebook)，此方法的優點為偽裝圖片的品質非常好，但仍然無法達到不失真還原秘密圖片。2010 年 Lin 和 Chan [8] 提出很好的秘密分享法，使用同一張掩護圖片產生 n 張偽裝圖片，此 n 張偽裝圖片均與原掩護圖片相似，但每張偽裝圖片所使用金鑰值不同，因此嵌入秘密圖片資料也不同。圖片還原時，t 張偽裝圖片可以同時還原秘密及掩護圖片，且秘密及掩護圖片兩者幾乎是不失真還原。不過 Lin 和 Chan [8] 仍然使用有限域 GF(251) 運算於 (t, n) 秘密分享法，所以理論上還是無法保證不失真還原的特性。

目前已有若干文獻提出完全不失真還原秘密圖片的方法，例如：論文 [9] 和 [10] 使用伽羅瓦二次冪有限域 (Power-of-Two Galois Fields) 之運算於 (t, n) 秘密分享法，即 $GF(2^m)$ 有限域，m 為任意整數，可將秘密圖片不失真還原。另外，值得一提的是論文

[11] 和 [12]，使用 (7, 4) 漢明碼 (Hamming Code) 編碼，將秘密圖片資料嵌入掩護圖片像素值的最低有效位元 (Least Significant Bits) 中，使得偽裝圖片品質大幅提升，但是論文 [11] 和 [12] 不具備 (*t, n*) 秘密分享法的性質。

秘密圖片分享法的首要前提，應是高品質的偽裝圖片，及具有完全不失真還原秘密圖片的特性。除此之外，還有一個值得探討的議題，即認證偽裝圖片的真偽，以免誤用被篡改的偽裝圖片，導致還原的秘密圖片不正確。有關認證偽裝圖片真偽之研究，如論文 [6] 使用同位元 (Parity-Bit Checking)，論文 [13] 利用中國同餘定理 (Chinese Remainder Theorem) 驗證，但論文 [6] 和 [13]，都使用模數 251 運算，因此秘密圖片無法不失真還原。論文 [14] 使用 Hash-Based Message Authentication 檢驗，然而此方法增加隱藏位元長度，因此偽裝圖片品質不佳。

綜觀前述，本研究希望提出一個 (*t, n*) 秘密圖片分享法，具有完全不失真還原秘密圖片的特性，並提升偽裝圖片的品質，以避免因偽裝圖片的失真度過高，而被不肖者察覺，進而破壞或竄改。另外，希望所提出的方法，也具有認證偽裝圖片真偽的功能，在還原秘密圖片時，可驗證偽裝圖片是否遭到篡改，以免影響秘密圖片還原的正確性。

2. 相關背景知識

本研究採用 (*t, n*) 秘密圖片分享法隱藏資料，並運用漢明碼嵌入秘密資料於不同掩護圖片中。因此，本節先介紹 (*t, n*) 秘密分享法和漢明碼編碼。

2.1 (*t, n*) 秘密分享法

秘密資料可使用隱藏技術，藉由某一媒介物隱藏，但是若媒介物遭到破壞，便無法還原秘密資料。1979 年 Shamir[1] 提出了 (*t, n*) 秘密分享法，將秘密資料分散隱藏，儲存成 *n* 份，而還原時只需 *n* 份中的任意 *t* 份，便可完整重建原秘密資料；但是如果少於 *t* 份時，便無法還原秘密資料。(*t, n*) 秘密分享法具有分散風險的優點，因此有較高的安全性。使用 (*t, n*) 秘密分享法時，必須先建構分享多項式 $F(x)$，如下所示：

$$F(x) = S + a_1 x + a_2 x^2 + \ldots + a_{t-1} x^{t-1} \quad (1)$$

其中 S 為秘密資料，$a_1, a_2, \ldots, a_{t-1}$ 為任意選擇的隨機數。

另外，在秘密分享法過程中，需預先決定 n 個金鑰值，$k_1, k_2, \ldots, k_n$，分別代入分享多項式，取得 $F(k_1) = y_1, F(k_2) = y_2, \ldots, F(k_n) = y_n$，因此可得 n 對金鑰值與其對應值，即 (k_i, y_i)，其中 $i = 1, 2, \ldots, n$。還原過程中，不失其一般性，若得到前 t 對金鑰值與其對應值 (k_i, y_i)，$i = 1, 2, \ldots, t$，即可使用拉格朗日插值法 (Lagrange Interpolation) 運算，還原多項式 $F(x)$，並可得到 t 個係數值，因此求得秘密資料 S。但是如果少於 t 對金鑰值與其對應值時，便無法找出 $F(x)$ 的 t 個係數，因此也無法得到祕密資料 S。

2.2 漢明碼

漢明碼是一種經常使用在通信領域的錯誤更正碼，一般應用在檢查接收的傳輸位元是否正確。為了偵測產生錯誤傳輸位元的位置，會額外加入檢查位元於傳輸位元中。例如：在 (7, 4) 漢明碼中，4 個傳輸位元序列 $D = [d_4\ d_3\ d_2\ d_1]$，加上 3 個檢查位元序列 $[c_3\ c_2\ c_1]$，形成 1 個字碼 (Codeword) $C = [d_4\ d_3\ d_2\ c_3\ d_1\ c_2\ c_1]$，字碼 C 是由位元序列 D 與生成矩陣 (Generator Matrix)G 所求得，如下列方程式 (2)：

$$C = (G * D^T)^T \pmod 2 \tag{2}$$

其中 $G = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix}$，T 是轉置(Transpose) 運算。

例如：$D = [d_4\ d_3\ d_2\ d_1] = [1\ 1\ 0\ 1]$，由方程式 (2) 求得 $C = [d_4\ d_3\ d_2\ c_3\ d_1\ c_2\ c_1] = [1\ 1\ 0\ 0\ 1\ 1\ 0]$，即 3 個檢查位元序列 $[c_3\ c_2\ c_1] = [0\ 1\ 0]$。

字碼 C 含有資料位元與檢查位元，其特性為：假若傳輸位元正確，C 的徵狀 (syndrome) 為 $[000]^T$，否則會指示傳輸位元錯誤的位置。徵狀的計算公式如下：

$$H * C^T = [000]^T \pmod 2 \tag{3}$$

其中 $H = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$ 是漢明查核矩陣(Check Matrix)。

例如：以上所求 C = [1 1 0 0 1 1 0]，代入方程式 (3)，可得 $[000]^T$。又假設 C 有一個位元錯誤發生在 d_3，其中 d_3 為由右到左的第六個位置，得到 $C' = [d_4\ d_3'\ d_2\ c_3\ d_1\ c_2\ c_1] = [1\ 0\ 0\ 0\ 1\ 1\ 0]$，也就是說，C 和 C′ 只有第六個位置的位元不同，可將 C′ 表示法如下所示：

$$C' = C \oplus E \tag{4}$$

其中 E = [0 1 0 0 0 0 0]，⊕為互斥或 (Exclusive Or) 運算子。當接收者收到 C′ 時，使用查核矩陣檢查，如下所示：

$$H * (C')^T = H * (C \oplus E)^T = (H * C^T) \oplus (H * E^T) = (H * E^T) \pmod 2 \tag{5}$$

其結果為 $[110]^T$，由 $[110]^T$ 位元序列可知，C′ 第六個位置的位元錯誤，修改第六個位置的位元，得到更正字碼 C，由 C 得到正確位元序列 D。

以上說明一般漢明碼通信應用情況，由於本文不會使用到生成矩陣 G，所以字碼 C 的徵狀未必是 $[000]^T$，但是會使用到類似上述方法的概念，詳細過程將在研究方法中說明。

3. 研究方法

本研究方法可分為兩個階段，第一階段為圖片分享，第二階段為圖片還原。第一階段圖片分享，主要是將秘密圖片嵌入 n 張掩護圖片，得到 n 張偽裝圖片，偽裝圖片外觀與掩護圖片相似，可以達成隱藏秘密資料的目的。第二階段為圖片還原，主要是將 t 張偽裝圖片，解碼還原成秘密圖片。

3.1 圖片分享

圖片分享階段，首先決定 (t, n) 秘密分享法之門檻值 t 和 n，並選擇 n 個相異金鑰值：$k_1, k_2, \ldots, k_n$，分別指定於對應的 n 張掩護圖片。我們必須先決定分享多項式，此多項式係數為秘密圖片像素值所組成。然後代入金鑰值 k_i，分別計算分享多項式 $F(k_i)$，其中 $i = 1, 2, \ldots, n$。將函數值 $F(k_i)$ 轉成二進制位元，利用漢明碼將此二位元字串逐一嵌

入第 i 張掩護圖片像素值較低的有效位元，得到偽裝圖片，此階段希望能達成兩個目的，即秘密分享以及減少偽裝圖片的失真。

假設某一秘密圖片及 n 張掩護圖片均為灰階圖片，因此圖片中每一像素含 8 個位元。以下敘述圖片分享步驟：

1. 依序取出秘密圖片的 t 個像素，建構分享多項式如下：

$$F(x) = p_0 + p_1 x + \cdots + p_{t-1} x^{t-1} \pmod{2^8} \quad (6)$$

其中 $p_0, p_1, \ldots, p_{t-1}$ 為秘密圖片的 t 個像素。代入金鑰值 k_i，分別計算分享多項式 $F(k_i)$，$i = 1, 2, \cdots, n$。因此，每一掩護圖片可分別取得函數值 $F(k_i)$。為簡化符號，以 $F(k)$ 表示某一掩護圖片所得函數值，並以二進制表示，得到 $[s_8\ s_7\ s_6\ s_5\ s_4\ s_3\ s_2\ s_1]$ 8 個位元。

2. 計算同位檢測位元

$$p = s_8 \oplus s_7 \oplus s_6 \oplus s_5 \oplus s_4 \oplus s_3 \oplus s_2 \oplus s_1 \quad (7)$$

其中⊕為互斥或運算子。$[p\ s_8\ s_7\ s_6\ s_5\ s_4\ s_3\ s_2\ s_1]$ 9 個位元將會嵌入到掩護圖片的 7 個像素中。

3. 依序取出掩護圖片 7 個像素 C^j，$j = 1, 2, \cdots, 7$，將每一像素 C^j 轉成二進制，即像素 C^j 含有 8 個位元 $[c_8^j\ c_7^j\ c_6^j\ c_5^j\ c_4^j\ c_3^j\ c_2^j\ c_1^j]$，下注標表示由右至左的位元，上注標表示第 j 個像素 C^j，$j = 1, 2, \cdots, 7$。

4. 令 LSB_1 為 7 個像素中最低有效位元序列，即 $LSB_1 = [c_1^7\ c_1^6\ c_1^5\ c_1^4\ c_1^3\ c_1^2\ c_1^1]$，同理 $LSB_2 = [c_2^7\ c_2^6\ c_2^5\ c_2^4\ c_2^3\ c_2^2\ c_2^1]$ 為次低有效位元序列，$LSB_3 = [c_3^7\ c_3^6\ c_3^5\ c_3^4\ c_3^3\ c_3^2\ c_3^1]$ 為第三低有效位元序列。即每一像素 C^j 分別萃取右邊三個位元 $[c_8^j\ c_7^j\ c_6^j\ c_5^j\ c_4^j\ c_3^j\ c_2^j\ c_1^j]$。如下所示：

$$C^1 = [\ c_8^1\ c_7^1\ c_6^1\ c_5^1\ c_4^1\ \mathbf{c_3^1}\ \mathbf{c_2^1}\ \mathbf{c_1^1}\ ]$$
$$C^2 = [\ c_8^2\ c_7^2\ c_6^2\ c_5^2\ c_4^2\ \mathbf{c_3^2}\ \mathbf{c_2^2}\ \mathbf{c_1^2}\ ]$$
$$C^3 = [\ c_8^3\ c_7^3\ c_6^3\ c_5^3\ c_4^3\ \mathbf{c_3^3}\ \mathbf{c_2^3}\ \mathbf{c_1^3}\ ]$$
$$C^4 = [\ c_8^4\ c_7^4\ c_6^4\ c_5^4\ c_4^4\ \mathbf{c_3^4}\ \mathbf{c_2^4}\ \mathbf{c_1^4}\ ]$$

$$C^5 = [\, c_8^5\ c_7^5\ c_6^5\ c_5^5\ c_4^5\ \mathbf{c_3^5}\ \mathbf{c_2^5}\ \mathbf{c_1^5}\,]$$
$$C^6 = [\, c_8^6\ c_7^6\ c_6^6\ c_5^6\ c_4^6\ \mathbf{c_3^6}\ \mathbf{c_2^6}\ \mathbf{c_1^6}\,]$$
$$C^7 = [\, c_8^7\ c_7^7\ c_6^7\ c_5^7\ c_4^7\ \mathbf{c_3^7}\ \mathbf{c_2^7}\ \mathbf{c_1^7}\,]$$

5. 使用 (7, 4) 漢明查核矩陣 H 乘上 LSB_r 序列的轉置，r = 1,2,3，計算每一 LSB_r 序列的徵兆 $[\, v_3^r\ v_2^r\ v_1^r\,]^T$，如下所示：

$$H * [LSB_r]^T = [\, v_3^r\ v_2^r\ v_1^r\,]^T \quad , r = 1,2,3 \tag{8}$$

6. 將步驟 2 所得 9 個位元 $[\, p\ s_8\ s_7\ s_6\ s_5\ s_4\ s_3\ s_2\ s_1\,]$，分成三組，每組 3 個位元，得到 $[\, p\ s_8\ s_7\,]$，$[\, s_6\ s_5\ s_4\,]$，$[\, s_3\ s_2\ s_1\,]$ 三組位元序列。

7. 將三組位元序列分別嵌入於三個 LSB 序列。由步驟 5 已知三個 LSB 序列的徵兆為 $[\, v_3^r\ v_2^r\ v_1^r\,]$，r = 1,2,3，將三個徵兆與三組位元序列各自做互斥或運算，計算三個 LSB 序列各自被改變位元的 $[\, b_3^r\ b_2^r\ b_1^r\,]$ 位置，r = 1,2,3，如下所示：

$$[\, b_3^3\ b_2^3\ b_1^3\,] = [\, v_3^3\ v_2^3\ v_1^3\,] \oplus [\, p\ s_8\ s_7\,]$$
$$[\, b_3^2\ b_2^2\ b_1^2\,] = [\, v_3^2\ v_2^2\ v_1^2\,] \oplus [\, s_6\ s_5\ s_4\,] \tag{9}$$
$$[\, b_3^1\ b_2^1\ b_1^1\,] = [\, v_3^1\ v_2^1\ v_1^1\,] \oplus [\, s_3\ s_2\ s_1\,]$$

8. 改變 LSB_r 序列裡第 $[\, b_3^r\ b_2^r\ b_1^r\,]$ 位置的位元，得到 LSB_r'，r = 1,2,3，例如：

$$[\, b_3^1\ b_2^1\ b_1^1\,] = [\, 1\ 1\ 1\,] = 7_{10}$$
$$LSB_1' = [\, \mathbf{c_1^{7'}}\ c_1^6\ c_1^5\ c_1^4\ c_1^3\ c_1^2\ c_1^1\,]$$
$$[\, b_3^2\ b_2^2\ b_1^2\,] = [\, 1\ 0\ 0\,] = 4_{10} \tag{10}$$
$$LSB_2' = [\, c_2^7\ c_2^6\ c_2^5\ \mathbf{c_2^{4'}}\ c_2^3\ c_2^2\ c_2^1\,]$$
$$[\, b_3^1\ b_2^1\ b_1^1\,] = [\, 0\ 1\ 0\,] = 2_{10}$$
$$LSB_3' = [\, c_3^7\ c_3^6\ c_3^5\ c_3^4\ c_3^3\ \mathbf{c_3^{2'}}\ c_3^1\,]$$

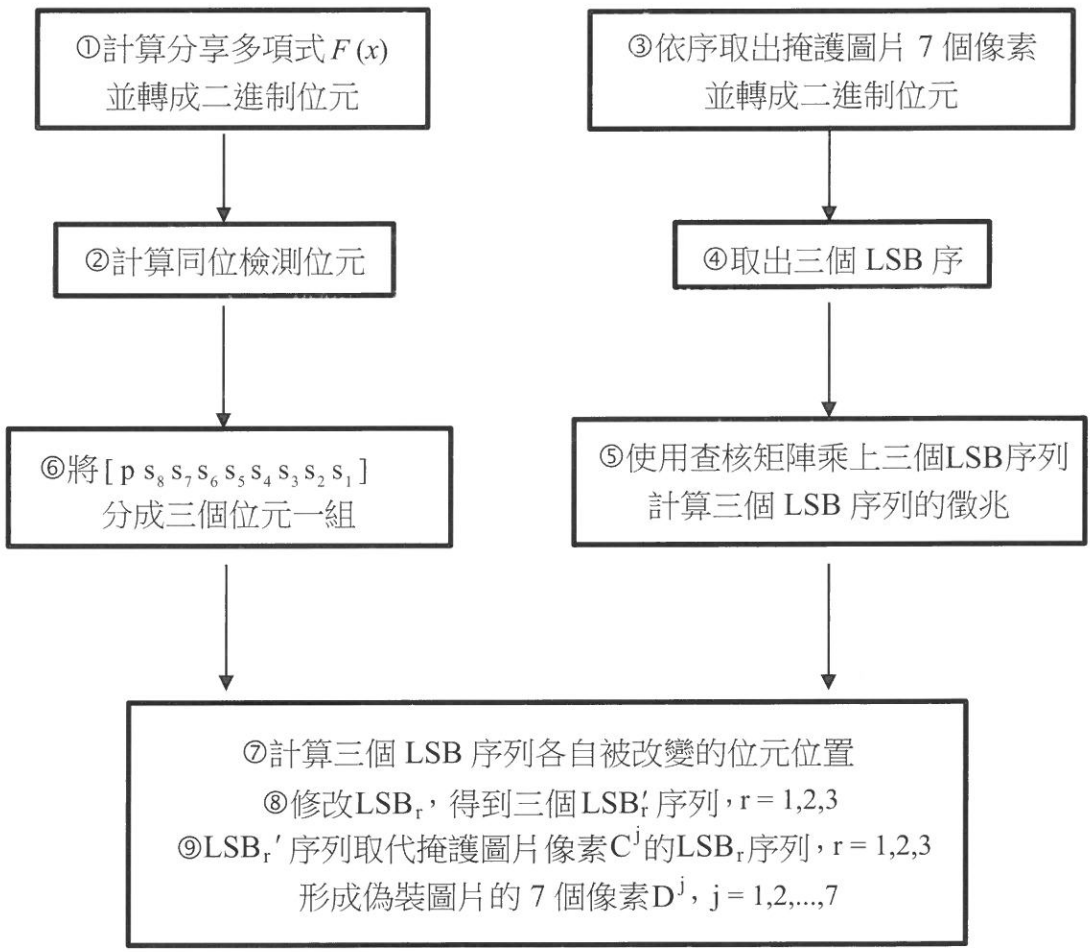
9. 將得到的三組 LSB_r' 序列取代掩護圖片像素 C^j 的 LSB_r 序列，r = 1,2,3，形成偽裝圖片的 7 個像素 D^j，j = 1, 2, …, 7。

$$D^1 = [\, c_8^1\ c_7^1\ c_6^1\ c_5^1\ c_4^1\ c_3^1\ c_2^1\ c_1^1\,]$$
$$D^2 = [\, c_8^2\ c_7^2\ c_6^2\ c_5^2\ c_4^2\ \mathbf{c_3^{2'}}\ c_2^2\ c_1^2\,]$$
$$D^3 = [\, c_8^3\ c_7^3\ c_6^3\ c_5^3\ c_4^3\ c_3^3\ c_2^3\ c_1^3\,]$$
$$D^4 = [\, c_8^4\ c_7^4\ c_6^4\ c_5^4\ c_4^4\ c_3^4\ \mathbf{c_2^{4'}}\ c_1^4\,]$$

$$D^5 = [\, c_8^5\ c_7^5\ c_6^5\ c_5^5\ c_4^5\ c_3^5\ c_2^5\ c_1^5 \,]$$
$$D^6 = [\, c_8^6\ c_7^6\ c_6^6\ c_5^6\ c_4^6\ c_3^6\ c_2^6\ c_1^6 \,]$$
$$D^7 = [\, c_8^7\ c_7^7\ c_6^7\ c_5^7\ c_4^7\ c_3^7\ c_2^7\ \mathbf{c}_1^{7'} \,]$$

10. 前述步驟 1 9，可將秘密圖片的 t 個像素，嵌入某張掩護圖片的 7 個像素。因此，重覆步驟 1 至步驟 9，直到秘密圖片所有像素嵌入到掩護圖片，形成偽裝圖片為止。

以上步驟，是將秘密圖片嵌入於某一掩護圖片，可重複 n 次，完成 n 張相異偽裝圖片，圖一所示為第一階段之圖片分享流程圖。



圖一 圖片分享流程圖，○中之數字為步驟號碼

現舉例說明各步驟如下：

1. 假設 $F(k) = [\, s_8\, s_7\, s_6\, s_5\, s_4\, s_3\, s_2\, s_1\,] = [\, 1\,0\,1\,1\,1\,0\,1\,1\,]$。

2. 計算同位檢測位元 $p = s_8 \oplus s_7 \oplus s_6 \oplus s_5 \oplus s_4 \oplus s_3 \oplus s_2 \oplus s_1 = 0$。$[\, s_8\, s_7\, s_6\, s_5\, s_4\, s_3\, s_2\, s_1\,]$ 9 個位元將會嵌入到掩護圖片的 7 個像素中。

3. 依序取出掩護圖片 7 個像素，將像素值轉成二進制。假設：

$$C^1 = [\, 0\,1\,0\,1\,1\,\mathbf{0}\,\mathbf{1}\,\mathbf{0}\,]$$
$$C^2 = [\, 1\,0\,0\,0\,1\,\mathbf{0}\,\mathbf{0}\,\mathbf{1}\,]$$
$$C^3 = [\, 0\,0\,1\,0\,1\,\mathbf{1}\,\mathbf{0}\,\mathbf{1}\,]$$
$$C^4 = [\, 1\,1\,0\,0\,1\,\mathbf{0}\,\mathbf{0}\,\mathbf{0}\,]$$
$$C^5 = [\, 0\,0\,0\,0\,1\,\mathbf{1}\,\mathbf{1}\,\mathbf{1}\,]$$
$$C^6 = [\, 0\,0\,1\,0\,1\,\mathbf{1}\,\mathbf{0}\,\mathbf{0}\,]$$
$$C^7 = [\, 1\,1\,0\,0\,1\,\mathbf{0}\,\mathbf{1}\,\mathbf{0}\,]$$

4. 取出三個 LSB 序列，如下所示：

$$LSB_1 = [\, c_1^7\ c_1^6\ c_1^5\ c_1^4\ c_1^3\ c_1^2\ c_1^1\,] = [\, 0\,0\,1\,0\,1\,1\,0\,]$$
$$LSB_2 = [\, c_2^7\ c_2^6\ c_2^5\ c_2^4\ c_2^3\ c_2^2\ c_2^1\,] = [\, 1\,0\,1\,0\,0\,0\,1\,]$$
$$LSB_3 = [\, c_3^7\ c_3^6\ c_3^5\ c_3^4\ c_3^3\ c_3^2\ c_3^1\,] = [\, 0\,1\,1\,0\,1\,0\,0\,]$$

5. 計算三個 LSB 序列徵兆 $[\, v_3{}^r\ v_2{}^r\ v_1{}^r\,]$，如下所示：

$$[\, v_3{}^1\ v_2{}^1\ v_1{}^1\,] = H * [\, LSB_1\,]^T = [\, 1\,0\,0\,]^T$$
$$[\, v_3{}^2\ v_2{}^2\ v_1{}^2\,] = H * [\, LSB_2\,]^T = [\, 0\,1\,1\,]^T$$
$$[\, v_3{}^3\ v_2{}^3\ v_1{}^3\,] = H * [\, LSB_3\,]^T = [\, 0\,0\,0\,]^T$$

6. 將步驟 2 所得 9 個位元 $[\, p\, s_8\, s_7\, s_6\, s_5\, s_4\, s_3\, s_2\, s_1\,]$，分成三個位元一組，得到 $[\, p\, s_8\, s_7\,]$，$[\, s_6\, s_5\, s_4\,]$，$[\, s_3\, s_2\, s_1\,]$ 三組位元序列，如下所示：

$$[\, s_3\, s_2\, s_1\,] = [\, 0\,1\,1\,]$$
$$[\, s_6\, s_5\, s_4\,] = [\, 1\,1\,1\,]$$
$$[\, p\, s_8\, s_7\,] = [\, 0\,1\,0\,]$$

7. 將 $[\, p\, s_8\, s_7\,]$，$[\, s_6\, s_5\, s_4\,]$，$[\, s_3\, s_2\, s_1\,]$ 三組位元序列分別嵌入三個 LSB 序列，由步驟 5 已知三個 LSB 序列的徵狀，將 LSB 序列的徵狀與三組位元序列做互斥或運算，計算 LSB_r 改變的位置 $[\, b_3^r\, b_2^r\, b_1^r\,]$，$r = 1,2,3$，如下所示：

$$[\,b_3^1\ b_2^1\ b_1^1\,]=[\,v_3^1\ v_2^1\ v_1^1\,]\oplus[\,s_3\ s_2\ s_1\,]=[\,100\,]\oplus[\,011\,]=[\,111\,]$$
$$[\,b_3^2\ b_2^2\ b_1^2\,]=[\,v_3^2\ v_2^2\ v_1^2\,]\oplus[\,s_6\ s_5\ s_4\,]=[\,011\,]\oplus[\,111\,]=[\,100\,]$$
$$[\,b_3^3\ b_2^3\ b_1^3\,]=[\,v_3^3\ v_2^3\ v_1^3\,]\oplus[\,p\ s_8\ s_7\,]=[\,000\,]\oplus[\,010\,]=[\,010\,]$$

8. 修改 LSB_r 的第 $[\,b_3^r\ b_2^r\ b_1^r\,]$ 位置的位元，得到 LSB_r' 序列，r =1,2,3，如下所示：

$$LSB_1'=[\,\mathbf{c_1^{7'}}\ c_1^6\ c_1^5\ c_1^4\ c_1^3\ c_1^2\ c_1^1\,]=[\,\mathbf{1}\,0\,1\,0\,1\,1\,0\,]$$
$$LSB_2'=[\,c_2^7\ c_2^6\ c_2^5\ \mathbf{c_2^{4'}}\ c_2^3\ c_2^2\ c_2^1\,]=[\,1\,0\,1\,\mathbf{1}\,0\,0\,1\,]$$
$$LSB_3'=[\,c_3^7\ c_3^6\ c_3^5\ c_3^4\ c_3^3\ \mathbf{c_3^{2'}}\ c_3^1\,]=[\,0\,1\,1\,0\,1\,\mathbf{1}\,0\,]$$

9. 將得到的三組 LSB_r' 序列取代掩護圖片像素 C^j 的 LSB_r 序列，r =1,2,3，形成偽裝圖片的 7 個像素 D^j，如下所示：

$$D^1=[\,0\,1\,0\,1\,1\,0\,1\,0\,]_2$$
$$D^2=[\,1\,0\,0\,0\,1\,\mathbf{1}\,0\,1\,]_2$$
$$D^3=[\,0\,0\,1\,0\,1\,1\,0\,1\,]_2$$
$$D^4=[\,1\,1\,0\,0\,1\,0\,\mathbf{1}\,0\,]_2$$
$$D^5=[\,0\,0\,0\,0\,1\,1\,1\,1\,]_2$$
$$D^6=[\,0\,0\,1\,0\,1\,1\,0\,0\,]_2$$
$$D^7=[\,1\,1\,0\,0\,1\,0\,1\,\mathbf{1}\,]_2$$

10. 重覆步驟 1 至步驟 9，直到秘密圖片所有像素嵌入到掩護圖片，形成偽裝圖片為止。

3.2 圖片還原

第二階段為圖片還原，首先需收集 n 張偽裝圖片中的任意 t 張，不失其一般性，收集前 t 張偽裝圖片，和對應的金鑰值 k_i，i =1, 2,..., t。圖片還原階段分別從各偽裝圖片，取出隱藏的鑰匙對應值 $F(k_i)$，得到 t 對金鑰值與其對應值 (k_i, y_i)，i =1, 2,..., t，即可使用拉格朗日插值法運算，重建 $F(x)$ 多項式，其係數即為秘密圖片的像素，將像素依順序組合，可重建秘密圖片。本研究方法使用 GF(2^8) 有限域，其數域範圍為 $[0, 2^8 - 1]$ = [0, 255]，與灰階圖片的像素值一致。建構或重建 $F(x)$ 多項式時，皆能不失真儲存或萃取多項式係數，即秘密圖片像素。因此，可完全不失真還原秘密圖片。秘密圖片還原步驟如下：

1. 依序取出偽裝圖片的 7 個像素 D^j，其中 j = 1, 2, ⋯, 7，並轉成二進制 8 個位元。
2. 取出三個 LSB_r' 序列，其中 r =1,2,3，LSB_r' 序列如下所示：

$$LSB_1' = [\ \mathbf{c_1^{7'}}\ c_1^6\ c_1^5\ c_1^4\ c_1^3\ c_{\mathbf{1}}^2\ c_1^1\ ] = [\ \mathbf{1}\ 0\ 1\ 0\ 1\ 1\ 0\ ]$$
$$LSB_2' = [\ c_2^7\ c_2^6\ c_2^5\ \mathbf{c_2^{4'}}\ c_2^3\ c_2^2\ c_2^1\ ] = [\ 1\ 0\ 1\ \mathbf{1}\ 0\ 0\ 1\ ]$$
$$LSB_3' = [\ c_3^7\ c_3^6\ c_3^5\ c_3^4\ c_3^3\ \mathbf{c_3^{2'}}\ c_3^1\ ] = [\ 0\ 1\ 1\ 0\ 1\ \mathbf{1}\ 0\ ]$$

3. 使用漢明查核矩陣分別乘上三個 LSB_r' 序列，求出 $[\ p\ s_8\ s_7\ ]$，$[\ s_6\ s_5\ s_4\ ]$，$[\ s_3\ s_2\ s_1\ ]$ 三組位元序列，如下所示：

$$H^*[\ LSB_1'\ ]^T = [\ s_3\ \ s_2\ \ s_1\ ]^T = [\ 0\ 1\ 1\ ]^T$$
$$H^*[\ LSB_2'\ ]^T = [\ s_6\ \ s_5\ \ s_4\ ]^T = [\ 1\ 1\ 1\ ]^T$$
$$H^*[\ LSB_3'\ ]^T = [\ p\ s_8\ \ s_7\ ]^T = [\ 0\ 1\ 0\ ]^T$$

4. 檢查 $p = s_8 \oplus s_7 \oplus s_6 \oplus s_5 \oplus s_4 \oplus s_3 \oplus s_2 \oplus s_1$ 是否成立？如不成立，則 $[\ s_8\ s_7\ s_6\ s_5\ s_4\ s_3\ s_2\ s_1\ ]$ 8 個位元有錯誤，此張偽裝圖片不可使用，需要收集其它張偽裝圖片以及對應的金鑰值；如果成立，執行步驟 5。

5.$[\ s_8\ s_7\ s_6\ s_5\ s_4\ s_3\ s_2\ s_1\ ]$ 為 $F(k)$ 的值，收集 t 個 $F(k)$ 值和對應的 t 個金鑰值，使用拉格朗日插值法，還原分享多項式 $F(x) = p_0 + p_1 x + \cdots + p_{t-1} x^{t-1} \pmod{2^8}$ 的 t 個係數值，其中係數值 p_0、p_1、⋯、p_{t-1} 為秘密圖片的像素。

6. 重覆上述步驟 1 到步驟 5，依序取得秘密圖片的所有像素。

7. 將秘密圖片的所有像素依照順序，重新組合將秘密圖片無失真還原。

圖二為圖片還原流程圖：

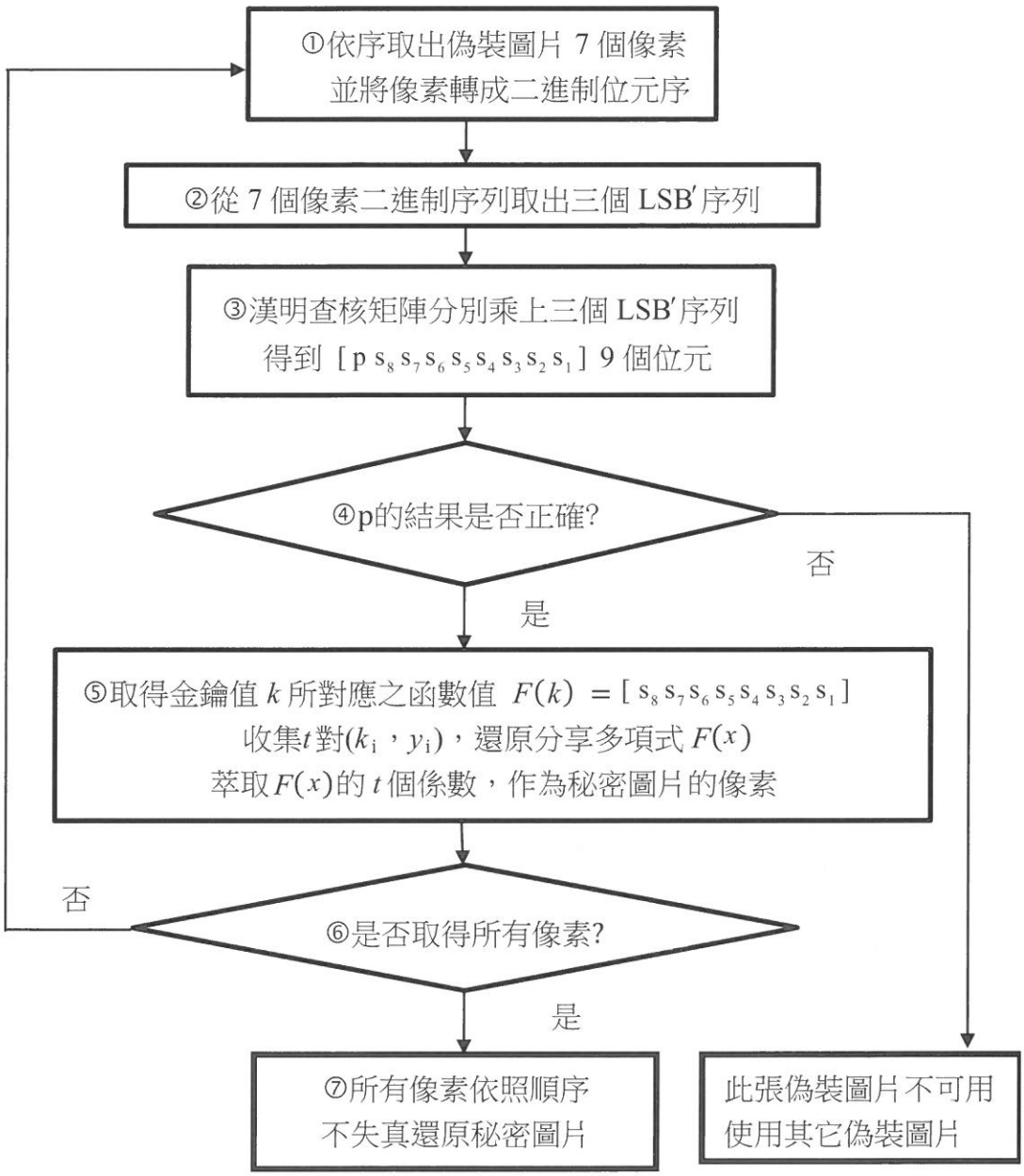


圖 二　圖片還原流程圖，○中之數字為步驟號碼

4. 實驗

由於將秘密圖片嵌入至掩護圖片，形成偽裝圖片，致使偽裝圖片與掩護圖片有所不同而引起失真。本文使用峰值信噪比 PSNR(Peak Signal-to-Noise Ratio) 評估偽裝圖片的失真情形，PSNR 也經常使用在圖像壓縮等領域中，作為測量信號重建品質的常用評估方法，其公式如下：

$$\text{PSNR} = 10 * log_{10}\left(\frac{255^2}{MSE}\right) dB \tag{11}$$

公式中均方差 MSE(Mean Squares Error) 為：

$$\text{MSE} = \frac{1}{H * W}\sum_{x=1}^{H}\sum_{y=1}^{W}(p_{xy} - p'_{xy})^2 \tag{12}$$

其中 H 為掩護圖片的高度，W 為掩護圖片的寬度，p_{xy} 為掩護圖片 (x，y) 位置的像素值，p'_{xy} 為偽裝圖片 (x，y) 位置的像素值。

掩護圖片與偽裝圖片的差異值越大，PSNR 的值越小，代表偽裝圖片失真越大，一般而言，當 PSNR 的值低於 30dB 時，人的眼睛容易察覺圖片失真，因此高品質之偽裝圖片的 PSNR 值至少要 30dB 以上。本文選擇 8 張不同的圖片作為掩護圖片，圖三為 8 張掩護圖片，掩護圖片的大小為 512*512 的灰階圖片，圖四為秘密圖片，秘密圖片的大小為 256*256 的灰階圖片。

(a)Baboon (b) Woman (c) Couple (d)Indians

(e) Goldhill (f) Lena (g) Peppers (h) Walkbridge

圖 三 8 張掩護圖片

圖 四 秘密圖片

影響偽裝圖片的失真程度，具有若干原因，其一是不同的 t 值，其二是嵌入圖片的方法。如何嵌入秘密圖片並且減少偽裝圖片與掩護圖片之間的差異，是本研究的重點之一。本章實驗會設定不同的 t 值，並且使用本研究所提方法，將秘密圖片嵌入掩

護圖片，形成偽裝圖片。

實驗中的參數設定，(t, n) 秘密分享法之參數為 $t = 2, 3, \cdots, 7$，$n = 8$。本研究所提方法，使用伽羅瓦二次冪有限域 $GF(2^8)$，有限域使用的不可約分多項式 (Irreducible Polynomial) 為 $x^8+ x^5+ x^3+ x^2+1$。實驗將秘密圖片嵌入在 8 (= n) 張掩護圖片中。表一列出不同 t 值時，偽裝圖片與掩護圖片所計算的 PSNR 值。

表 一　不同 t 值之偽裝圖片 PSNR 值比較

(t,n)	(2,8)	(3,8)	(4,8)	(5,8)	(6,8)	(7,8)
Baboon	44.54	46.30	47.52	48.51	49.32	49.96
Woman	44.51	46.29	47.55	48.49	49.28	49.97
Couple	44.52	46.27	47.51	48.50	49.28	49.98
Indians	44.49	46.26	47.54	48.51	49.27	49.95
Gold hill	44.51	46.27	47.53	48.47	49.29	49.98
Lena	44.52	46.26	47.53	48.48	49.29	49.94
Peppers	44.53	46.29	47.53	48.51	49.29	49.97
Walkbridge	44.49	46.25	47.51	48.47	49.28	49.95
Average	44.51	46.27	47.52	48.49	49.28	49.96

表一之實驗結果，顯示使用不同的掩護圖片，對 PSNR 值影響不大；影響較大的原因，主要是 t 值。由於 t 值決定分享多項式 $F(x)$ 係數的個數，因此影響建構多項式 $F(x)$ 的多寡。t 值越大表示係數越多，建構分享多項式的個數因此越少，使得嵌入到掩護圖片的像素個數減少，進而提高偽裝圖片的 PSNR 值。反之亦然，當 t 值越小時，分享多項式 $F(x)$ 的係數越少，導致增加 $F(x)$ 的建構個數，使得嵌入到掩護圖片的像素個數增加，因此降低偽裝圖片的 PSNR 值。

圖五是 t =2 時的 8 張偽裝圖片，分別對應到圖三的 8 張掩護圖片。我們肉眼觀察圖五和圖三的對應圖片，實在難以看出偽裝圖片與掩護圖片的差異。由於 t =2 的偽裝圖片，其 PSNR 平均值為 44.51(見表 一)，是所有實驗中品質最低的。因此，當 $t = 3, \cdots,$ 7 時，其 PSNR 值更高 (見表 一)，想必更不易看出偽裝圖片與掩護圖片的差異。

(a) Baboon (b) Woman (c) Couple (d) Indians

(e) Goldhill (f) Lena (g) Peppers (h) Walkbridge

圖五 $t = 2$ 時的 8 張偽裝圖片

由於論文 [9] 和 [10] 與本研究所提出方法，都具有完全不失真還原秘密圖片的優點。因此，我們想以偽裝圖片品質，比較三種方法的優劣，表二以 PSNR 平均值評量各方法的偽裝圖片品質。

表 二 PSNR 比較

t	論文[9]	論文[10]	本研究方法
$t = 3$	42.7	43.97	46.27
$t = 4$	44.4	45.22	47.53
$t = 5$	45.7	46.19	48.49
$t = 6$	46.7	46.98	49.28
$t = 7$	47.5	47.64	49.96

由表二實驗結果，固定 t 值下，本研究方法所產生的偽裝圖片 PSNR 值最高，表示具有最高偽裝圖片品質，與論文 [9] 和 [10] 的方法相比，本文提出方法偽裝圖片的失真最小。這是因為本研究使用 (7, 4) 漢明碼，嵌入秘密圖片至掩護圖片中，每次嵌入時，取掩護圖片 7 像素值的最低三個 LSB 序列，而且每個 LSB 序列含有 7 個位元，最多只更改其中的 1 個位元，形成 LSB′ 序列。由於 LSB′ 序列與 LSB 序列相比，最多僅有 1 個位元不同，因此可大幅降低偽裝圖片的失真，提高偽裝圖片品質。

5. 結論

本研究提出一個 (t, n) 秘密圖片分享法，運用 (7, 4) 漢明碼，GF(2^8) 伽羅瓦二次冪有限域運算，及同位元檢測於演算法中。所提出方法，具有完全不失真還原秘密圖片的特性，並大幅提升偽裝圖片的品質，在還原秘密圖片時，可驗證偽裝圖片是否遭到篡改，以免影響秘密圖片還原的正確性。由於應用有限域 GF(2^8)，擷取資料時，以 8 個位元為單位，對應到電腦最小儲存單位，即位元組 (Byte)，運算更為自然快速，配合本方法不失真還原的特性，可嵌入任何類型的數位數據，例如文書、執行檔、音訊等。由於偽裝圖片的品質與嵌入容量，互為衝突，未來的研究方向，將探討如何取捨兩者，以達到最佳平衡點。

參考文獻

1. P. Shamir, "How to share a secret" ,Communications. ACM, vol. 22, pp. 612-613, 1979.

2. G. R., Blakley, "Safeguarding cryptographic keys" , Proceedings of the National Computer Conference 48: 313–317, 1979.

3. C.C. Thien and J.C. Lin, "Secret image sharing" , Computers & Graphics, vol. 26, pp. 765-770, 2002.

4. R.Z. Wang and C.H. Su, "Secret image sharing with smaller shadow images" , Pattern Recogn. Lett., vol. 27, pp. 551-555, 2006.

5. C.C. Chang, C.C. Lin, C.H. Lin, Y.H. Chen, "A novel secret image sharing scheme in color images using small shadow images" , Inform. Sci. 178 (11), 2433–2447,2008.

6. C.C. Lin, W.H. Tsai, "Secret image sharing with steganography and authentication" , J. Syst. Software, Vol. 3, no. 73, pp. 405-414, 2004.

7. W.K. Su, L.S. Chen, and J.C. Lin, "Secret image sharing based on vector quantization" , International Journal of Circuits, Systems and Signal Processing, vol. 3, 2009.

8. P.Y. Lin and C.S. Chan, "Invertible secret image sharing with steganography" , Pattern Recogn. Lett., vol. 31, pp. 1887-1893, 2010.

9. M.C. Chien and J.I. G. Hwang, "Secret image sharing using (*t, n*) threshold scheme with lossless recovery" , 2012 5th International Congress on Image and Signal Processing, pp.1325 – 1329, 2012.

10. A. E. Trujillo, I. C. Camacho, and M. N. Miyatake, "Improved secret image sharing scheme with payload optimization" ,2012 IEEE 55th International Midwest Symposium on Circuits and Systems, pp. 1132-1135, 2012.

11. C. C. Chang and Y. C. Chou, "A high payload steganographic scheme based on (7, 4) hamming code for digital images" , International Symposium on Electronic Commerce and Security, pp. 16-21, 2008.

12. Y. P. Zhang, J. Jiang, C. Xu, B. Hua, and X. Y. Chen, "A new scheme for information hiding based on digital images", 2011 7th International Conference on Computational Intelligence and Security, pp. 512-516, 2011.

13. C.C Chang, Y.P. Hsieh, C.H. Lin, "Sharing secrets in stego images with authentication", Pattern Recognition, Vol. 41 no.10, pp. 3130-137, 2008.

14. C.N. Yang, T.S. Chen, K.H. Yu, C.C. Wang, "Improvements of image sharing with steganography and authentication", Journal of Systems and Software, Vol. 80, no. 7, pp. 1070-1076, 2007.

Recevied Oct 30, 2014
Revised Apr 29, 2015
Accepted May 6, 2015

A Data Hiding Scheme with Secret Sharing Based on Hamming Codes

Po-Ting Wu and Jen-Ing G. Hwang

Department of Computer Science and Information Engineering
Fu Jen Catholic University
Taipei, Taiwan 242, R.O.C

Abstract

A secret image can be embedded into n cover images to generate n stego images by using the (t,n)-threshold secret sharing method. To reconstruct the secret image requires at least t out of the n stego images. There are two stages in the (t,n)-threshold secret sharing method. The first stage is to embed the secret image, and the second stage is to reconstruct the secret image. When embedding the secret image, how to reduce cover image distortion is a major problem. This paper uses (7,4) Hamming code to embed the secret image into cover images. Because a Hamming code only modifies very few bits of a codeword when it embeds multiple bits, it can significantly reduce stego image distortion to lower the probability of being perceived by the attacker.

The proposed scheme also satisfies the criteria of the revealed secret image is lossless and stego images are meaningful. In addition, the scheme provides an authentication mechanism. The advantage of the lossless recovery of the secret image, due to use of operations in power-of-two Galois fields, which allows that the secret data can be any type of digital data, such as images, audio, video and also documents.

In this paper, we conduct some experiments to demonstrate the validity of the proposed algorithm. The experimental results show that the quality of our stego images is better than those of the other previous algorithms.

Key words: Data hiding, (t,n) secret sharing method, Hamming code, Power-of-two Galois fields