第三十五屆組合數學與計算理論研討會

The 35th Workshop on Combinatorial Mathematics and Computation Theory

論文集

主 辦 單 位 ：靜宜大學
共同主辦單位：演算法與計算理論學會
承 辦 單 位 ：靜宜大學資訊傳播工程系
協 辦 單 位 ：科技部工程科技推展中心、中央研究院資訊科學研究所、中央研究院資訊科技創新研究中心、靜宜大學資訊學院、靜宜大學資訊工程系

會議時間： 中華民國 107 年 5 月 11 日至 5 月 12 日

靜宜大學

大會組織

大會榮譽主席：

唐傳義 校長 (靜宜大學)

大會指導委員：

李家同 榮譽講座教授 (清華大學、靜宜大學、暨南大學)

李德財 院士 (中央研究院)

大會主席：

徐力行 教授 (靜宜大學)

林耀鈴 教授 (靜宜大學)

議程主席：

蔡英德 教授 (靜宜大學)

洪哲倫 教授 (靜宜大學)

宣傳主席：

鍾偉和 教授 (中央研究院)

修丕承 教授 (中央研究院資訊科技創新研究中心)

張志宏 教授 (靜宜大學)

詹毓偉 教授 (靜宜大學)

大會籌辦委員：(按姓氏筆劃順序排列)

王有禮 教授 (臺灣科技大學)

王志宇 教授 (中央研究院資訊科技創新研究中心)

高明達 教授 (中央研究院)

唐傳義 校長 (靜宜大學)

陳健輝 教授 (臺灣大學)

張佑榕 教授 (中央研究院資訊科技創新研究中心)

張貿翔 教授 (弘光科技大學)

張瑞雄 校長 (臺北商業大學)

楊昌彪 教授 (中山大學)

議程委員：(按姓氏筆劃順序排列)

王弘倫 教授 (臺北商業大學)
王炳豐 教授 (清華大學)
王釧茹 教授 (臺北市立大學)
何錦文 教授 (中央大學)
李新林 教授 (中正大學)
官大智 教授 (中興大學)
吳邦一 教授 (中正大學)
林英志 教授 (逢甲大學)
林耀鈴 教授 (靜宜大學)
徐力行 教授 (靜宜大學)
徐熊健 教授 (銘傳大學)
徐讚昇 教授 (中央研究院)
洪宗貝 教授 (高雄大學)
洪春男 教授 (大葉大學)
陳榮傑 教授 (交通大學)
陳彥宏 教授 (臺北市立大學)
黃光璿 教授 (暨南大學)
黃國璽 教授 (高雄海洋科技大學)
張仁俊 教授 (臺北大學)
張鎮華 教授 (臺灣大學)
張肇明 教授 (臺北商業大學)
陳宏賓 教授 (逢甲大學)
林英志 教授 (逢甲大學)
許弘駿 教授 (慈濟大學)
傅恆霖 教授 (交通大學)
彭勝龍 教授 (東華大學)
曾怜玉 教授 (靜宜大學)
詹景裕 教授 (臺北大學)
趙坤茂 教授 (臺灣大學)
楊正宏 校長 (高雄應用科技大學)
楊進雄 教授 (臺北商業大學)
廖崇碩 教授 (清華大學)
蔡英德 教授 (靜宜大學)
蔡錫鈞 教授 (交通大學)
劉嘉傑 教授 (世新大學)
盧錦隆 教授 (清華大學)
賴泳伶 教授 (嘉義大學)
賴寶蓮 教授 (東華大學)
謝孫源 教授 (成功大學)
戴顯權 教授 (成功大學)
顏重功 教授 (世新大學)

目錄

An Approximation Algorithm for the Multi-dimensional Knapsack Problem based on Hopfield Networks

Jen-Chun Chang, Jui-Sheng Chang, Hsin-Lung Wu
Department of Computer Science and Information Engineering
National Taipei University, New Taipei City, Taiwan

Abstract

In this paper, we study the d-dimensional knapsack problem (d-KP). The problem d-KP is an generalized problem of the well-known knapsack problem which is an NP-complete problem. It is also known that there is no fully polynomial-time approximation scheme for d-KP for $d > 1$ unless $P = NP$. We design an approximation algorithm for d-KP based on the Hopfield networks. Experimental results show that our proposed algorithm performs better than a well-known greedy algorithm in many cases.

1 Introduction

The knapsack problem is a well-known problem in combinatorial optimization. The problem states that, given a set of items each labeled with a weight and a value, determine the number of each item to include in a collection so that the total weight is less than or equal to a given limit and the total value is as large as possible. The multi-dimensional knapsack problem denoted as **d-KP** is a variant of the knapsack problem. In this variant, the weight of knapsack item i is expressed by a d-dimensional vector $\vec{w}_i = (w_{i1}, \cdots, w_{id})$ and the knapsack has a d-dimensional capacity vector $\vec{W} = (W_1, \ldots, W_d)$. The goal is to maximize the sum of the values of the items in the knapsack so that the sum of weights in each dimension j does not exceed W_j.

There is no efficient algorithm to find optimal solutions for a given d-dimensional knapsack problems. Instead of finding an optimal algorithm to solve the d-KP, one may try to design an approximate algorithm to solve it. In general, we consider two types of approximate algorithms: polynomial-time approximation scheme (PTAS) and fully polynomial-time approximation scheme (FPTAS). A PTAS is a polynomial-time algorithm which takes an instance of an optimization problem and a parameter $\epsilon > 0$ and produces a solution that is within a factor $1+\epsilon$ of being optimal. FPTAS is a restrictive version of PTAS. A FPTAS is a PTAS which requires the algorithm to be polynomial in both the problem size n and $1/\epsilon$. For the d-dimensional knapsack problem, it was shown by Gens and Levner [3] and independently by Korte and Schrader [5] that the existence of a fully polynomial time approximation scheme even for (2-KP) would imply that P equals NP. Thus, it is hard to design a good efficient approximation algorithm for 2-KP problem. The most obvious idea to find a feasible solution for (d-KP) is based on greedy-type heuristics.

In this paper, we design an efficient approximation algorithm for 2-KP problems based on the well-known Hopfield networks [4]. We compare our algorithm with a common greedy algorithm **GREEDY** for 2-KP. Experimental results demonstrate that our algorithm outperforms the algorithm GREEDY in many cases.

2 Preliminaries

2.1 Hopfield Neural Network

Hopfield neural network (denoted as HNN) is proposed by Hopfield in [4]. Here we use notations used in [1]. A Hopfield neural network is a graph with n nodes denoted by $\{y_1, y_2, \cdots, y_n\}$. Let h_{ij} denote the weight connected with edge (j, i) connecting node y_j to y_i. Let t_i denote the threshold value of each node y_i. Let H be the $n \times n$ matrix whose (i, j)-th entry is h_{ij} and let T be the threshold vector whose the i-th entry is t_i. Let $y_i(t) \in \{-1, 1\}$, $t = 0, 1, 2, \cdots$, denote the state of the i-th node at time t. The state at time t of the Hopfield network is denoted as the vector $Y(t) = (y_1(t), y_2(t), \cdots, y_n(t))$. The state $Y(t+1)$ at time $t + 1$ of the Hopfield network is obtained

from H, T, and $Y(t)$. Each node is updated as follows:

$$y_i(t+1) = \begin{cases} 1 & \text{if } \sum_{j=1}^{n} h_{ij}y_j(t) - t_i > 0 \\ y_i(t) & \text{if } \sum_{j=1}^{n} h_{ij}y_j(t) - t_i = 0 \\ -1 & \text{if } \sum_{j=1}^{n} h_{ij}y_j(t) - t_i < 0 \end{cases}$$

Once the state vector $Y(t)$ is not updated anymore, this state vector $Y(t)$ is called a stable state. Given a HNN, one of methods to show the existence of a stable state of this HNN is based on the method of energy functions. The energy function of a given Hopfield neural network is defined by

$$E(t) = -Y(t)^T \cdot H \cdot Y(t) + 2Y(t)^T \cdot T.$$

H is assumed to be symmetric and the diagonal entries are all non-negative. We define $\Delta E(t) = E(t+1) - E(t)$ and $\Delta Y(t) = Y(t+1) - Y(t)$. The goal is to show that $\Delta E(t) \leq 0$ for all t. Let $\Delta y_i(t) = y_i(t+1) - y_i(t)$ for each i. Note that

$$\Delta y_i(t) = \begin{cases} 2 & \text{if } y_i(t) = -1 \\ & \quad \text{and } \sum_{j=1}^{n} h_{ij}y_j(t) - t_i > 0 \\ 0 & \text{if } y_i(t) = \sum_{j=1}^{n} h_{ij}y_j(t) - t_i \\ -2 & \text{if } y_i(t) = 1 \\ & \quad \text{and } \sum_{j=1}^{n} h_{ij}y_j(t) - t_i < 0. \end{cases}$$

It is not hard to see that

$$\Delta E = -h_{jj}(\Delta y_j(t))^2 - 2(\sum_{j=1}^{n} h_{ij}y_j(t) - t_i)\Delta y_j(t).$$

Since $\Delta y_i(t)$ has three possible values, we continue the analysis by the following three cases.

- In the case that $\Delta y_i(t) = 2$, we have $y_i(t) = -1$ and $y_i(t+1) = 1$. This implies that $\sum_{j\neq i}^{n} h_{ij}y_j(t) - t_i > 0$. Hence $\Delta E < 0$.
- In the case that $\Delta y_i(t) = -2$, we have $y_i(t) = 1$ and $y_i(t+1) = -1$. This implies that $\sum_{j\neq i}^{n} h_{ij}y_j(t) - t_i < 0$. Hence $\Delta E < 0$.
- In the case that $\Delta y_i(t) = 0$, we have $y_i(t) = y_i(t+1)$. This implies that $\sum_{j\neq i}^{n} h_{ij}y_j(t) - t_i = 0$. Thus $\Delta E = 0$.

Based on the above observation, we conclude that $\Delta E \leq 0$.

2.2 Multi-dimensional Knapsack Problems

Knapsack Problem (denoted as KP), is one of well-known NP-complete problems. The problem is defined as follows. n items are given and the i-th item has its value v_i and its weight w_i. The maximum capacity of the knapsack is W. The target is to select a subset of $\{1, 2, \cdots, n\}$ such that the total cost of the selected items is maximized and the total weight is at most W. A given knapsack problem can be modeled as a solution of the following linear programming.

$$\begin{aligned} \text{Maximize } & \sum_{i=1}^{n} v_i x_i \\ \text{subject to } & \sum_{i=1}^{n} w_i x_i \leq W \\ & x_i \in \{0,1\},\ i = 1, \ldots, n \end{aligned}$$

d-dimensional Knapsack Problem (denoted as d-KP) is a generalized problem of the knapsack problem. The problem is defined as follows. n items are given and the i-th item has its value v_i and d weights $w_{i1}, w_{i2}, \ldots, w_{id}$. There are d capacity constraints $R_1, \cdots, R_d$. The target is to select a subset S of $\{1, 2, \cdots, n\}$ such that the total value of the selected items is maximized and satisfies d requirements $\sum_{i\in S} w_{ij} \leq R_j$ for each $j \in \{1, 2, \cdots, d\}$. A given d-dimensional knapsack problem can be modeled as a solution of the following linear programming.

$$\begin{aligned} \text{Maximize } & \sum_{i=1}^{n} v_i x_i \\ \text{subject to } & \sum_{i=1}^{n} w_{ij} x_i \leq R_j \ \forall j \in \{1, 2, \cdots, d\} \\ & x_i \in \{0,1\},\ i = 1, \ldots, n \end{aligned}$$

3 Our Main Approximation Algorithm for d-KP

In this section, we show our approximation algorithm for d-KP. Our algorithm is designed based on Hopfield neural networks. For reader's comprehension, we give our algorithm for 2-KP. It is easy to extend our proposed algorithm to d-KP for $d > 2$.

An $n \times d$ weight matrix (w_{ij}), an n-dimensional value vector $V = (v_1, v_2, \cdots, v_n)^T$, and two capacity constraints R_1, R_2 are given as inputs. Next, we define the matrix H and its corresponding energy function E for this 2-KP. For $j \in \{1, 2\}$, let W_j be the n-dimensional vector $(w_{1j}, w_{2j}, \cdots, w_{nj})^T$.

$$H \doteq V \cdot V^T - \gamma \cdot (\alpha_1 \cdot W_1 \cdot W_1^T + \alpha_2 \cdot W_2 \cdot W_2^T)$$

where α_1 and α_2 are two parameters which control the degree of how the feasible solutions satisfy two given linear constraints. Note that diagonal entries of the matrix H are required to be nonnegative. To obtain this, we use another parameter γ to control this requirement. Precisely, let ℓ be the following value:

$$\ell = \min_{i \in \{1,2,\cdots,n\}} (\frac{v_i^2}{(w_{i1}^2 + w_{i2}^2)}).$$

Now, we require that γ is a parameter chosen from $\{\frac{\ell}{10}, \frac{2\ell}{10}, \cdots, \ell\}$. Let $\vec{1} = (1, 1, \cdots, 1)^T$. Then the threshold vector T is set as

$$T = H \cdot \vec{1}^T.$$

Let us see how it works. We transform the binary vector $X(t) \in \{0, 1\}^n$ into the vector $Y(t) \in \{-1, 1\}^n$ by setting

$$X(t) = \frac{1}{2}\left(\vec{1}^T - Y(t)\right).$$

The original energy function E is defined as

$$E(t) = -X(t)^T \cdot H \cdot X(t)$$

where it captures the given optimization problem 2-KP. However, the state vector in the HNN is a $-1/1$ vector. Thus, we have to transform $X(t)$ into $Y(t)$. Therefore we have

$$\begin{aligned} & 4E(t) \\ = \; & -4(\frac{\vec{1} - Y(t)}{2})^T \cdot H \cdot (\frac{\vec{1} - Y(t)}{2}) \\ = \; & -\vec{1}^T \cdot H \cdot \vec{1} + 2 \cdot Y(t)^T \cdot (H \cdot \vec{1}) - Y(t)^T \cdot H \cdot Y(t). \end{aligned}$$

After eliminating the constant term $-\vec{1}^T \cdot H \cdot \vec{1}$, we define the following new energy function $E'(t)$.

$$\begin{aligned} E'(t) &= -Y(t)^T \cdot H \cdot Y(t) + 2Y(t)^T \cdot H \cdot \vec{1} \\ &= -Y(t)^T \cdot H \cdot Y(t) + 2Y(t)^T \cdot T \end{aligned}$$

where $T = H \cdot \vec{1}$. Therefore, H and T are used to generate the desired Hopfield networks to obtain an approximated feasible solution.

4 Experiments

In this section, we compare our algorithm with the greedy algorithm **GREEDY**. The algorithm **GREEDY** is executed as follows.

1. First it computes the ratio $r_i \doteq \frac{v_i}{(w_{i1} \times w_{i2})}$.
2. It chooses the item with the highest ratio r_i.
3. Add this item into the knapsack if the constrains are satisfied after adding this item and delete it from the item set otherwise.
4. Repeat this procedure until all items are checked.

The test input sets are generated randomly. We show our experimental result in Figure 1 where the value in X-axis represents the first capacity constraint R_1 and the value in Y-axis represents the second capacity constraint R_2. the value in Z-axis represents the ratio between the maximum values obtained by two approximate algorithms and the optimal dynamic programming algorithm of 2-KP. When the ratio between R_1 and R_2 is high, our algorithm outperforms the algorithm GREEDY.

5 Conclusion

In this paper, we study the d-dimensional knapsack problem and design an approximation algorithm based on Hopfield networks. We consider 2-KP in our experimental results. Experimental results show that our proposed algorithm outperforms the algorithm GREEDY when the ratio of two input constrains is high.

References

[1] Kai-Yeung Siu, Vwani Roychowdhury and Thomas Kailath, "Discrete Neural Computation: A Theoretical Foundation", pages 345–385. 1995.

[2] Hans Kellerer, Ulrich Pferschy and David Pisinger, "Knapsack Problems", pages 235–271. 2004.

[3] G. Y. Gens and E. V. Levner. Computational complexity of approximation algorithms for combinatorial problems. In *Mathematical Foundations of Computer Science 1979*, pages 292-300, 1979.

[4] J. J. Hopofield. Neural networks and physical systems with emergent collective computational abilities. In *Proceedings of the National Academy of Sciences*, 79:2554–2558, 1982.

[5] B. Korte and R. Schrader. On the existence of fast approximation schemes. In "Nonlinear Programming", vol. 4, pages 415-437. Academic Press, 1981.

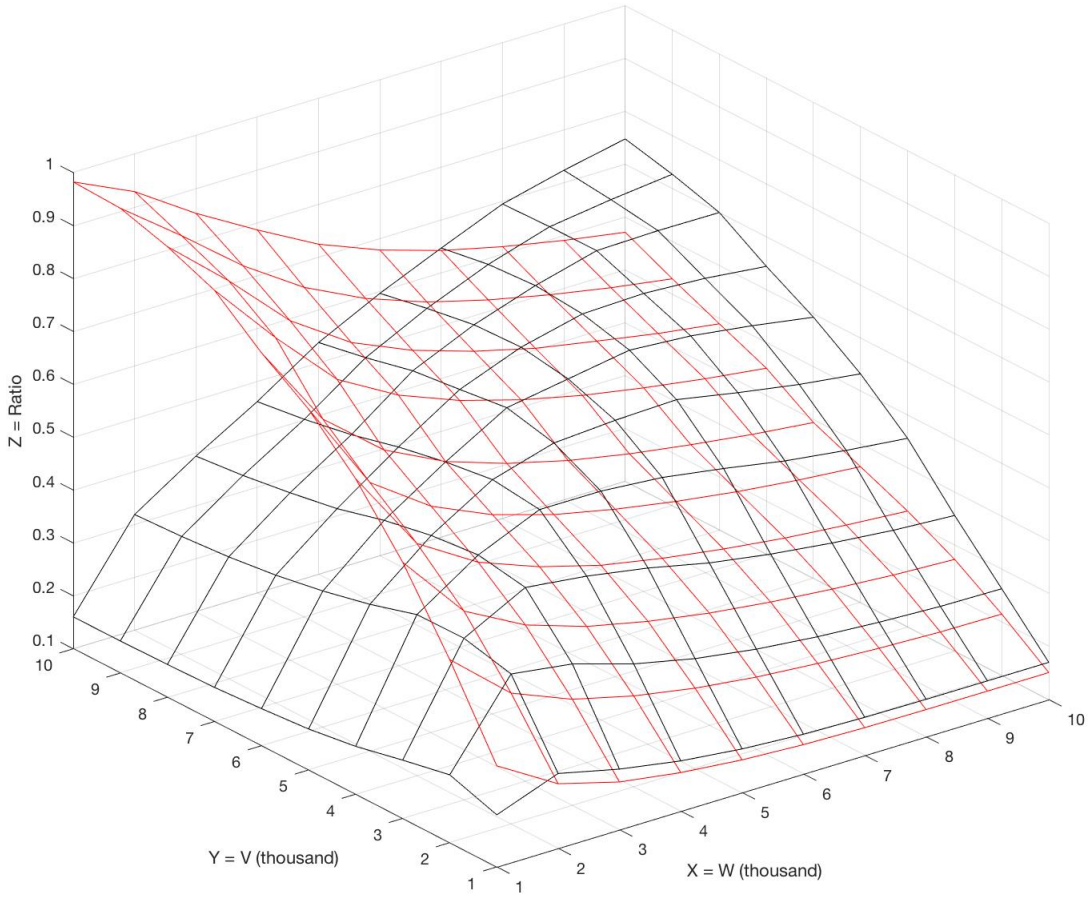


Figure 1: The approximation rates of **GREEDY** and our algorithm. The red curves are rates of the algorithm **GREEDY** and the black ones are rates of our proposed algorithm

書本圖上的(強)彩虹連通

(Strong) Rainbow Connection on Book Graphs

Yung-Ling Lai and Wei-Lin Huang
Department of Computer Science and Information Engineering
National Chiayi University
Taiwan, Chiayi
{yllai, s1050472}@mail.ncyu.edu.tw

摘要

彩虹著色是一種特別的圖形邊著色方式，他所要求的不是相鄰的邊著不同的顏色，而是任意兩點間有一條顏色皆不相同的路徑，稱之為彩虹路徑。若要求圖中任意兩點間都有一條最短路徑為彩虹路徑，則此著色方式稱為強彩虹著色。(強)彩虹連通數則為使圖形達到(強)彩虹著色所需要的最小著色數。由於尋求任意一個圖形的彩虹連通數是一個 NP 困難的問題，因此彩虹著色的研究多半聚焦在特殊圖形上。有關 m 頁書本圖（以 B_m 表示）的彩虹連通數和強彩虹連通數分別已有研究，但所發表的結果有誤，本文將前人在 B_3 的彩虹連通數和 B_m 的強彩虹連通數的結果做一更正。

關鍵字: 邊著色、(強)彩虹連通數、書本圖。

1 背景介紹

圖形著色問題可分為點著色和邊著色。一般的點(邊)著色指的是一圖中相鄰的點(邊)所著的顏色皆不相同。彩虹著色(rainbow coloring)是一種特殊的邊著色方式，由 Chartrand 等[5]在 2008 年所提出。若一個邊著色使圖 G 中任意兩點間都存在有一條顏色皆不相同的路徑，稱之為彩虹路徑(rainbow path)，則該著色方式稱為該圖的彩虹著色(rainbow coloring)，而該圖即為彩虹連通圖(rainbow connected graph)。一個圖形 G 的彩虹連通數(rainbow connection number)即為使圖形 G 為彩虹連通所需要的最少邊著色數，以 $rc(G)$ 來表示。

若一個邊著色使圖 G 中任意兩點間存在一條最短路徑為彩虹路徑，稱之為強彩虹路徑（strong rainbow path），則此種著色方式稱為強彩虹著色(strong rainbow coloring)，該圖即為強彩虹連通圖(strong rainbow connected graph)。而一個圖形的強彩虹連通數(strong rainbow connection number)即為使圖形為強彩虹連通所需要的最少邊著色數，以 $src(G)$ 表示。

對任意圖找出其彩虹連通數或強彩虹連通數都是 *NP* 困難(*NP-Hard*) 的問題[4]，就算是一個圖形的彩虹連通數為 2 時，要去判斷這個已經著色的圖形是否為為彩虹連通時，這個問題依然是 *NP* 完全(NP-Complete)的問題。對於任意固定的 $k \geq 2$ 時，判斷一個任意圖形的彩虹連通數是否為 k，一樣也是 NP 完全的問題[2]。

然而，也有一些特定圖形的彩虹連通數是已經知道的，比如：完全多分圖 (complete multipartite graphs)[3][13]；風扇圖(fan graphs)和太陽圖(sun graphs)[16]；輪子圖(wheel graphs)和彼得森圖(Petersen graph)[1]；荊棘圖(thorn graph)[12]；線圖(line graphs)[9][10]等等。

同樣的，也有一些圖形的強彩虹連通數已被求出，如蝴蝶圖(butterfly graph)和貝奈斯圖(Benes graph)[2]；阿貝爾群中的凱萊圖(Cayley graphs on Abelian groups)[8]等。一個圖形的直徑為圖中任意兩點間最大的距離，以 $diam(G)$ 表示。一個非孤立點(nontrivial)的連通圖 G，若其圖形的邊數為 m，依照定義任兩點間都要有一條彩虹路徑，而強彩虹路徑亦為彩虹路徑，且若每個邊的顏色均不同，對連通圖來說任意兩點間必有強彩虹路徑，因此 $diam(G) \leq rc(G) \leq src(G)$ [5]這個結果是顯而易見的。而 $src(G) = m$ 若且唯若 G 為樹圖，且不存在有強彩虹連通樹圍為 $m-1$ 的圖形，[11]刻畫了強彩虹連通數為 $m-2$ 的圖形，[6]則證明了對任意整數 a 與 b，存在圖形 G 使 $rc(G) = a$，$src(G) = b$ 的條件是 $a = b \in \{1,2\}$ 或 $3 \leq a \leq b$。

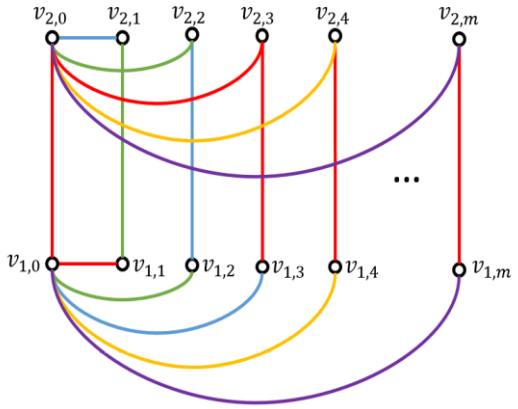
彩虹連通應用在許多地方，最著名的就是在網路加密上的應用，我們可以將圖上所有的點視為機構，邊視為網路連線，邊上所給的顏色視為在該連線上所使用的加密方式。若兩機構間要傳遞一需要加密的重要資訊，為了防止密碼被破解，每經過一個中繼點就必需換一種加密方式；而一條沒有重複的加密方式的路徑才會是安全路徑。由於加密需要成本，因此會期待用最少的加密方式來使任意兩機構間都能有一條安全路徑，這裡所需要的最少加密方式就相當於圖形中的彩虹連通數。若要求所傳遞的路徑是最短的，則使任意兩機構間皆存在安全路徑所需的最少加密方式，即為相對圖形上的

強彩虹連通數。

m 頁書本圖(m-Book graph) B_m[7]，是由星圖(star graph)[15] S_m 與路徑圖 P_2 (path graph)做笛卡爾積(Cartesian product)所形成的圖形，其中星圖時常被當做實作高速計算網路的拓樸模型，書本圖則是建置更大型的高速計算網路的拓樸模型。高速計算網路通常領導著科學和人類的進步，計算出來的資料需要整合且資料通常需要加密，因此研究書本圖的(強)彩虹連通數是非常重要的。本文即修訂前人對書本圖的彩虹連通數和強彩虹連通數所提出的結果。

2 預備知識

兩個圖 $G_1=(V_1,E_1)$ 與 $G_2=(V_2,E_2)$ 的笛卡爾積(cartesian product)記做 $G_1\square G_2$，其點集合為 $\{(u,v)\,|\,u\in V_1,v\in V_2\}$，而兩頂點 (u_1,v_1) 與 (u_2,v_2) 相鄰，若且唯若 $v_1=v_2$ 且 $(u_1,u_2)\in E_1$ 或 $u_1=u_2$ 且 $(v_1,v_2)\in E_2$。路徑圖(path graph)是一棵有著兩個節點維度為 1，其餘節點維度為 2 的樹。一個 m 節點的星圖(star graph) S_m 可定義為一棵樹具有一個內部節點和 m 個葉節點。而一個 m 頁書本圖 B_m，是由長度為 1 的路徑圖與 m 個頂點的星圖做笛卡爾積的結果，即為 $P_2\square S_m$。若 $V(P_2)=\{u_1,u_2\}$ 且 $V(S_m)=\{v_i\,|\,0\le i\le m\}$，$E(S_m)=\{v_0v_i\,|\,1\le i\le m\}$。為了描述方便，本文將 B_m 的頂點集合表示為 $V(B_m)=\{v_{i,j}\,|\,1\le i\le 2,\ 0\le j\le m\}$，$E(B_m)=\{v_{i,0}v_{i,j}\,|\,1\le i\le 2,0\le j\le m\}\bigcup\{v_{1,j}v_{2,j}\,|0\le j\le m\}$，如圖一所示。



圖一：B_m 的表示方法

在 2014 和 2015 年，書本圖的彩虹連通數與強彩虹連通數分別被提出，他們所提出的定理如下：

定理（Sy[17]）一個 m 頁書本圖 B_m（$m\ge 3$）的彩虹連通數為 $rc(B_m)=4$。

定理（Rao[14]）一個 m 頁書本圖 B_m（$m\ge 3$）的強彩虹連通數為 $src(B_m)=m+1$。

然而這些定理都有部分錯誤，本論文的目的即修正此二定理之錯誤。

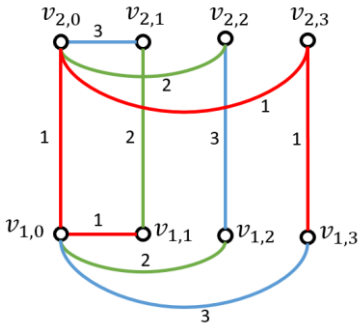
3 主要成果

根據定理（Sy[17]），$rc(B_3)=4$。這是錯誤的，我們將定理修正如定理 1。

定理 1. 對一個 m 頁書本圖 B_m，當 $2\le m\le 3$ 時，$rc(B_m)=3$；對於所有的 $m\ge 4$，$rc(B_m)=4$。

證明：由於 $B_2\subset B_3$，又 $diam(B_m)=3$，本定理直接證 $rc(B_3)=3$ 即可。考慮邊著色 $c:E(B_3)\to\{1,2,3\}$ 如下：

$$c(v_{1,0},v_{1,j})=j,1\le j\le 3;$$
$$c(v_{2,0},v_{2,j})=4-j,1\le j\le 3;$$
$$c(v_{1,j},v_{2,j})=j+1,0\le j\le 2;$$
$$c(v_{1,3},v_{2,3})=1;$$



圖二：B_3 的著色方法

由圖二可以看出，$v_{1,0}$ 到 $v_{1,j}$ $1\le j\le 3$，$v_{2,0}$ 到 $v_{2,j}$ $1\le j\le 3$ 及 $v_{1,i}$ 到 $v_{2,i}$ $0\le i\le 3$ 距離均為 1，故必存在彩虹路徑，而 $v_{1,0}$ 到 $v_{2,j}$ $1\le j\le 2$ 則可有 $v_{1,0}$，$v_{2,0}$，$v_{2,j}$ 的彩虹路徑且 $v_{1,0}$，$v_{1,3}$，$v_{2,3}$ 亦為彩虹路徑，即 $v_{1,0}$ 到圖上任意點均有彩虹路徑，反之亦然。接下來考慮 $v_{1,j}$ 到 $v_{2,k}$，其中 $j\ne k$ 則 $v_{1,j}-v_{1,0}-v_{2,0}-v_{2,k}$，$v_{1,j}-v_{2,j}-v_{2,0}-v_{2,k}$，或 $v_{1,j}-v_{1,0}-v_{1,k}-v_{2,k}$ 中至少一條彩虹路徑，故 c 為 B_3 上的 3-彩虹著色。又 $diam(B_3)=3$，故 $rc(B_3)=3$。■

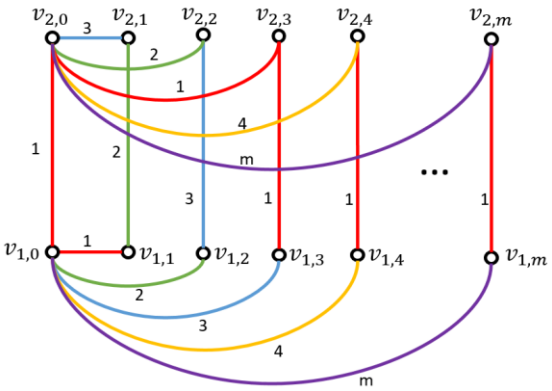
至於強彩虹著色部分，根據定理（Rao[14]），所有的書本圖 $src(B_m)=m+1$，但我們的定理 2 將證明其實 $src(B_m)=m$。

定理 2. $src(B_m)=m$ ，對於所有的 $m\geq 3$ 。

證明：考慮邊著色 $c':E(B_m)\to\{1,2,...,m\}$ 如下：

$$c'(v_{1,0},v_{1,j})=j, 1\leq j\leq m;$$
$$c'(v_{2,0},v_{2,j})=4-j, 1\leq j\leq 3;$$
$$c'(v_{2,0},v_{2,j})=j, 4\leq j\leq m;$$
$$c'(v_{1,j},v_{2,j})=j+1, 0\leq j\leq 2;$$
$$c'(v_{1,j},v_{2,j})=1, 3\leq j\leq m.$$

考慮任意兩點 $v_{i,j}$ 到 $v_{l,k}$ 若 $i=l$ 或 $j=k$ ，則兩點間最短路徑唯一，且該路徑均為彩虹路徑，故不失一般性可考慮 $v_{1,j}$ 與 $v_{2,k}$ 且 $j\neq k$ 。又當 $0\leq j,k\leq 3$ 時，因 c' 與 c 完全相同，且因 B_3 為二分圖，距離為 2 的點必沒有長度為 3 的路徑，由定理 1 的證明得知，此著色法亦為 B_3 的強彩虹著色。故我們只需要考慮當 $j\geq 4$ 或 $k\geq 4$ 時即可，分為以下幾種情況討論。



圖三. B_m 的著色表示方法

(1) 當 $j=0, k\geq 4$ 時，$v_{1,0}-v_{2,0}-v_{2,k}$ 為強彩虹路徑。

(2) 當 $j=1, k\geq 4$ 時，$v_{1,1}-v_{2,1}-v_{2,0}-v_{2,k}$ 皆為強彩虹路徑。

(3) 當 $j=2, k\geq 4$ 時，$v_{1,2}-v_{2,2}-v_{2,0}-v_{2,k}$ 為強彩虹路徑。

(4) 當 $j=3, k\geq 4$ 時 $v_{1,3}-v_{1,0}-v_{2,0}-v_{2,k}$ 為強彩虹路徑。

(5) 當 $j\geq 4, k\neq 3$ 時 $v_{1,j}-v_{1,0}-v_{2,0}-v_{2,k}$ 為強彩虹路徑。

(6) 當 $j\geq 4, k=3$ 時 $v_{1,j}-v_{1,0}-v_{1,3}-v_{2,3}$ 是強彩虹路徑。

因為任意兩點間均存在一條最短路徑為彩虹路徑，故 c' 為 B_m 的一個強彩虹著色，即 $src(B_m)\leq m$ 。由於 B_m 是由兩個 S_m 組成，又因 S_m 當中任意兩點的最短距離在 2 以內，在 B_m 中並未減少在同一個 S_m 中兩點的距離，且因 S_m 為樹圖，因此 $src(B_m)\geq src(S_m)=m$ 。故由夾擠定理得 $src(B_m)=m$ 。 ■

4 結論

本文中，修正了前人所提出的結果，得到 B_3 的彩虹連通數及 B_m 的強彩虹連通數。根據這些結果，未來將可拓展至堆疊書本圖（stack book graphs）的彩虹著色及強彩虹著色中。

參考文獻

[1] P. Ananth and M. Nasre, New hardness results in rainbow connectivity, *arXiv: 1104.2074 [cs.CC] 2011*.

[2] A. Arputhamary and M. Helda Mercy, Strong Rainbow Edge Coloring of Some Interconnection Networks, *Procedia Computer Science 57 , 338-345, 2015*.

[3] Y. Caro, A. Lev, Y. Roditty, Z. Tuza and R. Yuster. On rainbow connection, *The Electronic Journal of Combinatorics, 15, \# R57, 2008*.

[4] S. Chakraborty, E. Fischer, A. Matsliah and R. Yuster. Hardness and algorithms for rainbow connection, *Journal of Combinatorial Optimization, 1-18, 2009*.

[5] G. Chartrand, G. L. Johns, K. A. McKeon and P. Zhang. Rainbow connection in graphs, *Mathematica Bohemica, 133 (1), 85-98, 2008*.

[6] X. Chen and X. Li, A solution to a conjecture on the rainbow connection number, *Ars Combin. 104 , 193–196, 2012*.

[7] J. A. Gallian, J. A. Dynamic Survey DS6: Graph Labeling. *Electronic J. Combinatorics, DS6, 1-58, Jan. 3, 2007*.

[8] H. Li, X. Li and S. Liu, The (strong) rainbow connection numbers of Cayley graphs on Abelian groups, *Comput. Math. Appl. 62, 4082-4088, 2011*.

[9] X. Li and Y. Sun, Rainbow connection numbers of line graphs, *Ars Combin.100, 449–463. 2011*.

[10] X. Li and Y. Sun, Upper bounds for the rainbow connection numbers of line graphs, *Graphs & Combin., 28(2), 251–263, 2012*.

[11] X. Li and Y. Sun, On the strong rainbow connection of a graph, *Bull. Malays. Math. Sci. Soc. 36, 299–311, 2013*.

[12] Y. Liu and Z. Wang, Rainbow Connection Number of the Thorn Graph, *Applied Mathematical Sciences, 8 (128), 6373-6377, 2014*.

[13] K. A. McKeon, G. Chartrand, G. L. Johns and P. Zhang. The Rainbow Connectivity of a Graph, *Networks, 54(2), 75-81, 2009*.

[14] K. S. Rao, R. Murali and S.K. Rajendra. Rainbow and Strong Rainbow criticalness of some standard graphs. *International Journal Of Mathematics And Computer Research, 829-836 ISSN :2320-7167, 2015*.

[15] S. Skiena, "Cycles, Stars, and Wheels." §4.2.3 in *Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica*. *Reading, MA: Addison-Wesley, 83, 144-147, 1990*.

[16] S. Sy, G. H. Medika and L. Yulianti. The Rainbow Connection of Fan and Sun, *Applied Mathematical Sciences, 7(64), 3155-3159, 2013*.

[17] S. Sy, R. Wijaya and S. Supangken. Rainbow Connection Numbers of Some Graphs. Combinatorics, *Applied Mathematical Sciences, 8(94), 4693 - 4696, 2014*。

Online Linear Approximation via Follow-the-Leader Strategy

Jen-Chun Chang, Cheng-Kang Liu, Hsin-Lung Wu
Department of Computer Science and Information Engineering
National Taipei University, New Taipei City 237, Taiwan.
jcchang@mail.ntpu.edu.tw
s710583111@webmail.ntpu.edu.tw
hsinlung@mail.ntpu.edu.tw

Abstract

In this paper, we study an online prediction problem: at time t, an example x_t is given, the learner makes a prediction $\hat{y}_t$, and receives the labeled value y_t. The loss at this step is $(\hat{y}_t - y_t)^2$. We assume that $y_t = \alpha x_t + \beta + \epsilon_t$ for two fixed but unknown constants α and β and for some value ϵ_t with $|\epsilon_t| \leq B$. In addition, we assume that $x_t = t$. We propose an online algorithm based on the Follow-the-Leader strategy and show a logarithmic regret bound for this algorithm.

1 Introduction

In this paper, we consider a prediction problem in which the t-th example is $x_t \in \mathbb{R}$ and its label y_t is the value such that $y_t = \alpha x_t + \beta + \epsilon_t$ for two fixed but unknown constants α and β and for some value ϵ_t with $|\epsilon_t| \leq B$. The examples are given in an online way. At time t, the example $x_t \in \mathbb{R}$ is given to the learner. The learner predicts a value $\hat{y}_t$ before observing the label y_t. We require that the memory of the learner is limited. That is, the size of the learner's memory is constant. Next, the square loss is used to measure the accuracy of a prediction method. Precisely, for a labeled example (x_t, y_t), the loss of a prediction $\hat{y}_t$ is $(y_t - \hat{y}_t)^2$. The regret of the learner is defined as

$$\text{Regret} \doteq \sum_{t=1}^{T}(y_t - \hat{y}_t)^2 - \min_{a,b}\sum_{t=1}^{T}(ax_t + b - y_t)^2.$$

Note that the best pair (a, b) which obtains the minimum value $\sum_{t=1}^{T}(ax_t + b - y_t)^2$ is just the least-square solution of the linear regression problem for the data set $\{(x_t, y_t) : 1 \leq t \leq T\}$. Thus our goal is to design a prediction method such that the regret to the linear regressor is as small as possible.

In this paper, we propose an online algorithm to solve this problem. Our algorithm is designed based on the so-called Following-the-Leader (**FTL**) strategy and obtains logarithmic regret under the restriction that $x_t = t$. Precisely, under the restriction that $x_t = t$, the regret of our algorithm is $O(B^2 \log T)$ where B is the upper bound of the error term ϵ_t and T is the number of iterations.

For the bounded case that there exists a fixed constant C such that $|x_t| \leq C$ for all t, this problem can be solved by using the online algorithm proposed by Zinkevich [2] or the logarithmic regret algorithms developed by Hazan et al. in [1]. However, their methods cannot be applied to the unbounded case.

2 Preliminaries

Given a data set $\{(x_1, y_1), (x_2, y_2), \cdots, (x_T, y_T)\}$, we assume that there exist three constants α, β, and B such that, for each t, $y_t = \alpha x_t + \beta + \epsilon_t$ for some ϵ_t with $|\epsilon_t| \leq B$. Let X_t, E_t, Y_t be matrices defined by

$$X_t = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_t \end{bmatrix}, E_t = \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_t \end{bmatrix}, Y_t = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_t \end{bmatrix}.$$

Suppose, at time t, we have collected t labeled examples $\{(x_1, y_1), (x_2, y_2), \cdots, (x_t, y_t)\}$. The best linear function $y = a_t x + b_t$ which minimizes the sum of square errors $\sum_{i=1}^{t}(a_t x_i + b_t - y_i)^2$ is the least-square solution

of the linear regression problem for the data set $\{(x_1, y_1), (x_2, y_2), \cdots, (x_t, y_t)\}$. Precisely, the least solution $(a_t, b_t)^T = (X_t^T X_t)^{-1} X_t^T Y_t$. Therefore, we conclude that

$$a_t = (\overline{y})_t - b_t(\overline{x})_t$$

and

$$b_t = \frac{\sum_{i=1}^{t}(x_i - (\overline{x})_t)(y_i - (\overline{y})_t)}{\sum_{i=1}^{t}(x_i - (\overline{x})_t)^2}$$

where $(\overline{x})_t = \frac{1}{t}\sum_{i=1}^{t} x_i$ and $(\overline{y})_t = \frac{1}{t}\sum_{i=1}^{t} y_i$.

3 The Main Online Algorithm

Our algorithm is designed based on the Following-The-Leader (FTL) strategy. That is, at time $t+1$, the algorithm computes the least-square solution (a_t, b_t) for the data set $\{(x_1, y_1), \ldots, (x_t, y_t)\}$ and makes a prediction

$$\hat{y}_{t+1} = a_t x_{t+1} + b_t.$$

In order to compute (a_t, b_t), the algorithm needs to store the data set $\{(x_1, y_1), \ldots, (x_t, y_t)\}$. However, the memory space of the algorithm is limited and hence is not allowed to store the whole labeled examples. To overcome this, we observe that it is sufficient to maintain following four values to compute (a_t, b_t).

- $V_{xy}(t) = \sum_{t=1}^{T} x_t y_t$
- $V_x(t) = \sum_{t=1}^{T} x_t$,
- $V_y(t) = \sum_{t=1}^{T} y_t$, and
- $V_{xx}(t) = \sum_{t=1}^{T} x_t^2$.

Indeed,

$$b_t = \frac{tV_{xy}(t) - V_x(t)V_y(t)}{tV_{xx}(t) - V_x(t)^2}$$

and

$$a_t = \frac{V_y(t) - b_t V_x(t)}{t}.$$

Therefore, after predicting the y_{t+1}, $V_{xy}(t+1), V_x(t+1), V_y(t+1), V_{xx}(t+1)$ can be updated easily in an online sense.

Based on the above observation, we propose the following constant-memory-size online linear approximation algorithm.

Algorithm 1: FTL-based Online Linear Approximation

Input: data set $(x_t, y_t) \in \mathcal{S}$ comes sequentially , $\forall t = 1, 2, ..., T$
Output: $\widehat{y} \in \mathcal{Y}$
Set
$V_{xy}(0) = 0, V_x(0) = 0, V_y(0) = 0, V_{xx}(0) = 0$
Randomly generate a_1, b_1
for $t = 1; t \leq T$ **do**
  Receive x_t then computes $\widehat{y}_t = a_t x_t + b_t$ to make a prediction
  Receive the true value y_t and suffer a loss $\ell_t(a_t, b_t) = (\widehat{y}_t - y_t)^2$
  Update four values
  $V_{xy}(t) = V_{xy}(t-1) + x_t y_t$,
  $V_x(t) = V_x(t-1) + x_t$,
  $V_y(t) = V_y(t-1) + y_t$,
  $V_{xx}(t) = V_{xx}(t-1) + x_t^2$,
  Compute $b_t = \frac{tV_{xy}(t) - V_x(t)V_y(t)}{tV_{xx}(t) - V_x(t)^2}$ and $a_t = \frac{V_y(t) - b_t V_x(t)}{t}$

3.1 Performance Analysis for the case that $x_t = t$

Our goal is to bound the following regret:

$$\text{Regret} \doteq \sum_{t=1}^{T}(y_t - \hat{y}_t)^2 - \min_{a,b}\sum_{t=1}^{T}(ax_t + b - y_t)^2.$$

It is not hard to derive by induction that

$$\begin{aligned} & \sum_{t=1}^{T}(y_t - \hat{y}_t)^2 - \min_{a,b}\sum_{t=1}^{T}(ax_t + b - y_t)^2 \\ = & \sum_{t=1}^{T}\ell_t(a_t, b_t) - \min_{a,b}\sum_{t=1}^{T}\ell_t(a, b) \\ \leq & \sum_{t=1}^{T}\left(\ell_t(a_t, b_t) - \ell_t(a_{t+1}, b_{t+1})\right). \end{aligned}$$

Our target is to bound $\ell_t(a_t, b_t) - \ell_t(a_{t+1}, b_{t+1})$. Note that $y_t = \alpha x_t + \beta + \epsilon_t$ for some fixed but unknown constants α and β. For convenience, we define the following terms:

- $A_t \doteq \sum_{i=1}^{t}(x_i - \overline{x}_t)^2$ and
- $B_t \doteq \sum_{i=1}^{t}(x_i - \overline{x}_t)(\varepsilon_i - \overline{\varepsilon}_t)$.

Lemma 1. *A_t and A_{t-1} can be bounded in $\Theta(t^3)$.*

Proof. According to the setting of special case $x_i = i, \forall i$.

$$A_{t-1} = \sum_{i=1}^{t-1}(x_i - \overline{x}_{t-1})^2 \leq \sum_{i=1}^{t-1} t^2 \leq t^3.$$

So, similar with A_t. □

Lemma 2. *B_t and B_{t-1} can be bounded in $\mathcal{O}(Bt^2)$.*

Proof. According to the setting of special case $x_i = i, \forall i$.

$$\begin{aligned} B_{t-1} &= \sum_{i=1}^{t-1}(x_i - \overline{x}_{t-1})(\varepsilon_i - \overline{\varepsilon}_{t-1}) \leq \sum_{i=1}^{t-1}(t)(2B) \\ &\leq 2B\sum_{i=1}^{t-1} t \leq 2Bt^2. \end{aligned}$$

The bound for B_t is similar to B_{t-1}. □

In addition, it is easy to derive following equations:

$$a_t = \beta + \frac{B_t}{A_t} \tag{1}$$

$$a_{t-1} = \beta + \frac{B_{t-1}}{A_{t-1}} \tag{2}$$

$$b_t = \alpha + \overline{\varepsilon}_t - \overline{x}_t\frac{B_t}{A_t} \tag{3}$$

$$b_{t-1} = \alpha + \overline{\varepsilon}_{t-1} - \overline{x}_{t-1}\frac{B_{t-1}}{A_{t-1}} \tag{4}$$

Then, we have

$$\begin{aligned} & \ell_t(a_{t-1}, b_{t-1}) - \ell_t(a_t, b_t) \\ = \;& (a_{t-1}x_t + b_{t-1} - y_t)^2 - (a_t x_t + b_t - y_t)^2 \\ = \;& (a_{t-1}x_t + b_{t-1} - \alpha x_t - \beta - \epsilon_t)^2 \\ & -(a_t x_t + b_t - -\alpha x_t - \beta - \epsilon_t)^2 \\ = \;& [\frac{B_{t-1}}{A_{t-1}}x_t + \overline{\epsilon}_{t-1} - \frac{B_{t-1}}{A_{t-1}}\overline{x}_{t-1} - \epsilon_t]^2 \\ & -[\frac{B_t}{A_t}x_t + \overline{\epsilon}_t - \frac{B_t}{A_t}\overline{x}_t - \varepsilon_t]^2 \\ = \;& [\frac{B_{t-1}}{A_{t-1}}(x_t - \overline{x}_{t-1}) + \frac{B_t}{A_t}(x_t - \overline{x}_t) + \overline{\varepsilon}_{t-1} + \overline{\varepsilon}_t - 2\varepsilon_t] \\ & \times[\frac{B_{t-1}}{A_{t-1}}(x_t - \overline{x}_{t-1}) - \frac{B_t}{A_t}(x_t - \overline{x}_t) + \overline{\varepsilon}_{t-1} - \overline{\varepsilon}_t] \end{aligned}$$

Moreover, it is not hard to obtain following lemmas.

Lemma 3. $|\overline{x}_{t-1} - \overline{x}_t| = \frac{x_t - \overline{x}_{t-1}}{t}$.

Lemma 4. $|\overline{\varepsilon}_{t-1} - \overline{\varepsilon}_t| = \frac{\varepsilon_t - \overline{\varepsilon}_{t-1}}{t}$ *can be bounded in* $\mathcal{O}(\frac{B}{t})$.

Furthermore, we observe that

$$\begin{aligned} & |B_{t-1} - B_t| = |\sum_{i=1}^{t-1}(x_i - \overline{x}_{t-1})(\varepsilon_i - \overline{\varepsilon}_{t-1}) - \sum_{i=1}^{t-1}(x_i - \overline{x}_{t-1} \\ & + \overline{x}_{t-1} - \overline{x}_t)(\varepsilon_i - \overline{\varepsilon}_{t-1} + \overline{\varepsilon}_{t-1} - \overline{\varepsilon}_t) - (x_t - \overline{x}_t)(\varepsilon_t - \overline{\varepsilon}_t)| \\ & = |-\sum_{i=1}^{t-1}(x_i - \overline{x}_{t-1})(\overline{\varepsilon}_{t-1} - \overline{\varepsilon}_t) - \sum_{i=1}^{t-1}(\varepsilon_i - \overline{\varepsilon}_{t-1})(\overline{x}_{t-1} - \overline{x}_t) \\ & - \sum_{i=1}^{t-1}(\overline{\varepsilon}_{t-1} - \overline{\varepsilon}_t)(\overline{x}_{t-1} - \overline{x}_t) - (x_t - \overline{x}_t)(\varepsilon_t - \overline{\varepsilon}_t)| \\ & = |-\sum_{i=1}^{t-1}(x_i - \overline{x}_{t-1})(\frac{\varepsilon_t - \overline{\varepsilon}_{t-1}}{t}) - \sum_{i=1}^{t-1}(\varepsilon_i - \overline{\varepsilon}_{t-1})(\frac{x_t - \overline{x}_{t-1}}{t}) \\ & - \sum_{i=1}^{t-1}(t-1)\frac{(\varepsilon_t - \overline{\varepsilon}_{t-1})(x_t - \overline{x}_{t-1})}{t^2} - (x_t - \overline{x}_t)(\varepsilon_t - \overline{\varepsilon}_t)| \end{aligned}$$

.

In addition, we also observe that

$$|\frac{1}{A_{t-1}} - \frac{1}{A_t}| = |\frac{A_t - A_{t-1}}{A_t A_{t-1}}| = |\frac{t-1}{t^2}\frac{(x_t - \overline{x}_{t-1})^2}{A_t A_{t-1}}|.$$

Now we bound $[\frac{B_{t-1}}{A_{t-1}}(x_t - \overline{x}_{t-1}) + \frac{B_t}{A_t}(x_t - \overline{x}_t) + \overline{\varepsilon}_{t-1} + \overline{\varepsilon}_t - 2\varepsilon_t]$. According to the assumption that $x_t = t$, Lemma 1 and, Lemma 2, we have

$$\begin{aligned} & |(\frac{B_{t-1}}{A_{t-1}}(x_t - \overline{x}_{t-1}) + \frac{B_t}{A_t}(x_t - \overline{x}_{t-1}) + (\overline{\varepsilon}_{t-1} + \overline{\varepsilon}_t) \\ & - 2\varepsilon_t| \\ & \leq |(\frac{Bt^2}{t^3}t) + (\frac{Bt^2}{t^3}t) + 2B - 2B| \\ & \leq |2B| \end{aligned}$$

Next, we bound $[\frac{B_{t-1}}{A_{t-1}}(x_t - \overline{x}_{t-1}) + \frac{B_t}{A_t}(\overline{x}_t - x_t) + \overline{\varepsilon}_{t-1} - \overline{\varepsilon}_t]$. By Lemma 1, Lemma 2, Lemma 3, and

Lemma 4, we have

$$
\begin{aligned}
&|\frac{B_{t-1}}{A_{t-1}}(x_t-\overline{x}_{t-1})-\frac{B_t}{A_t}(x_t-\overline{x}_t)+(\overline{\varepsilon}_{t-1}-\overline{\varepsilon}_t)|\\
&=|[(\frac{B_{t-1}}{A_{t-1}}-\frac{B_t}{A_t})(x_t-\overline{x}_{t-1})+\frac{B_t}{A_t}(x_t-\overline{x}_{t-1})]-\frac{B_t}{A_t}(x_t-\overline{x}_t)\\
&+(\overline{\varepsilon}_{t-1}-\overline{\varepsilon}_t)|\\
&=|(\frac{B_{t-1}}{A_{t-1}}-\frac{B_t}{A_t})(x_t-\overline{x}_{t-1})+\frac{B_t}{A_t}(\overline{x}_t-\overline{x}_{t-1})+(\overline{\varepsilon}_{t-1}-\overline{\varepsilon}_t)|\\
&=|(\frac{B_{t-1}}{A_{t-1}}-\frac{B_t}{A_t})|t+\frac{B}{t}+\frac{B}{t}|\\
&=[|\frac{B_{t-1}-B_t}{A_{t-1}}+B_t|\frac{1}{A_{t-1}}-\frac{1}{A_t}|]t+2\frac{B}{t}\\
&\leq|\frac{Bt}{t^3}+(Bt^2)(\frac{1}{t^5})|+2\frac{B}{t}\\
&\leq\frac{B}{t^2}+2\frac{B}{t}\\
&\leq\frac{B}{t}
\end{aligned}
$$

Now, after combining two upper bounds, we conclude that $\ell_{t-1}(a_{t-1},b_{t-1})-\ell_{t-1}(a_t,b_t)=O(B^2/t)$. So far, we can obtain our main theorem.

Theorem 1. *Suppose that $x_t=t$. The regret bound of the FTL-based Online Linear Approximation is $O(B^2\log T)$.*

Proof.

$$
\begin{aligned}
&\sum_{t=1}^{T}(y_t-\hat{y}_t)^2-\min_{a,b}\sum_{t=1}^{T}(ax_t+b-y_t)^2\\
=&\sum_{t=1}^{T}\ell_t(a_t,b_t)-\min_{a,b}\sum_{t=1}^{T}\ell_t(a,b)\\
\leq&\sum_{t=1}^{T}(\ell_t(a_t,b_t)-\ell_t(a_{t+1},b_{t+1}))\\
\leq&O(B^2\sum_{t=1}^{T}1/t)\\
=&O(B^2\log T).
\end{aligned}
$$

□

References

[1] E. Hazan, A. Agarwal, and S. Kale. Logarithmic regret algorithms for online convex optimization. *Machine Learning*, Volume 69, No,2-3, pages 169-¡V192, 2007.

[2] M. Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the Twentieth International Conference of Machine Learning*, pages 928–936, 2003.

改善傳感器網路壽命的分散式情境感知協定之研究

陳大仁*,[1]，徐力行[§,2]，許明陽[1]，邱威閔[1]
[1]Department of Information Management,
National Taichung University of Science and Technology, Taichung, Taiwan,
[2]Department of Computer Science and Information Engineering,
Providence University
danny@nutc.edu.tw*, lihhsing@gmail.com[§]

摘要

無線感測網路(WSNs)是由許多有限電量的感測器組成，經常佈署在人煙稀少的位置，如何有效地節省感測器能源並延長網路壽命成為相當重要的議題。其中最常見的有效能節能方法為叢集方法，目的將無線感測網路劃分為許多叢集，每個叢集推選出一個叢集頭匯集感測器的資料，最後發送給基地台。此方法可以有效地降低整體網路能量的損耗，但是會導致叢集頭快速死亡，因此我們提出 CEMST 演算法，叢集頭的選擇考慮節點重疊度(overlapping degrees)、密度(density)與剩餘電能(residual energy)，並且傳輸的路徑使用 Dijkstra 最短路徑演算法和最小生成樹(MST)中的 Borůvka 演算法，減緩並平衡叢集的電能損耗，並延長感測網路的壽命。

1. 簡介

無線感測網路技術[1, 2, 3, 15]主要面臨的問題是降低與均衡感測節點間的電能損耗，以及延長網路壽命，因此有許多研究者針對不同的應用與環境，提出適當的策略和協定。

目前許多無線感測網路採用叢集(Cluster)架構[6]，每個叢集有一個節點成為叢集頭(Cluster head)，叢集頭的工作可以分為兩個部分：叢集內(intra-cluster)通訊與叢集間(inter-cluster)通訊，如圖 1。

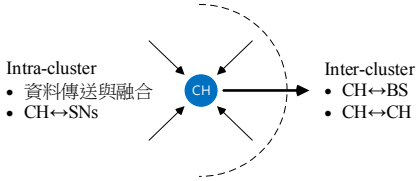


圖 1、叢集頭的負責工作

叢集內傳輸的工作分為收集節點資料，以及資料融合與傳送。前者，叢集頭收集叢集內所有節點的資料，許多文獻以單跳的方式傳輸資料，若沒有明確的方法限制叢集內節點的數目，可能會造成成員太多，加重叢集頭的負擔。因此有效控制叢集是相當重要的。後者，叢集頭融合叢集內節點的數據，可以有效減少傳輸所需的電能。

叢集間的通訊協定主要是處理叢集之間、以及叢集與基地台之間的通訊。研究[7]發現叢集間多跳模式，可在無線通道中更節能。但是愈靠近基地台叢集頭的資料處理負荷將會愈重，造成基地台附近的叢集頭快速死亡，形成熱點(Hot-Spot)區域。有學者提出改進的方法如 UCR[11]，減緩熱點問題的發生。叢集頭的工作比一般節點繁重，如何有效選擇叢集頭以及善用叢集頭資源顯得相當重要。

本文提出了 CEMST(Clustered Energy-efficient Minimum Spanning Tree)方法，目標是建立理想的叢集結構。CEMST 分為三個階段：叢集形成、路徑建置和重新路由。首先，叢集形成階段中，選擇叢集頭乃基於節點的重疊度、密度和殘存電能，找出相關度愈高且電能愈多的節點擔任叢集頭。叢集範圍是根據叢集頭的密度和基地台距離。路徑建置分為叢集內與叢集間兩部分，叢集內使用 Dijkstra 自我穩定演算法規劃最短路徑，叢集間使用 Borůvka-MST [5]演算法，規劃最小成本生成樹。最後，重新路由方法針對叢集內斷開的節點重新連接或是更換電能即將耗盡的叢集頭，從而降低重新組態的電能消耗，使得整個網路能擁有更長的壽命。

2. 文獻探討

無線感測網路中，叢集結構可分為兩類:第一類為叢集內單跳協定，包括 LEACH、HEED、UCR、Hybrid、FBR…等協議，第二類為叢集內多跳協定，包括 DSBCV…等。

2.1. 叢集內單跳

LEACH(**L**ow **E**nergy **A**daptive **C**lustering **H**ierarchy) [8]是叢集架構中最具有代表性的協定之一，每個叢集皆有一個叢集頭收集叢集內節點的感測資料，於資料融合後發送到基地台。全部的節點收集一次資料並送回基地台的過程稱為一個『回合』(Round)，在每回合中分為兩個階段：設定階段(Set-up phase)和穩定階段(Steady-state phase)。設定階段主要是選擇叢集頭和形成叢集，選擇叢集頭時，每個節點自行決定在第 r 回合隨機產生介於 0 到 1 的數值，若低於門檻值 $T(n)$ 則此節點將成為叢集頭。

$$T(n) = \begin{cases} \frac{P}{1-P[r \bmod (1/p)]}, & n \in G \\ 0, & otherwise \end{cases} \quad (1)$$

其中，P 為預期成為叢集頭的機率，r 當前的回合數。$r \bmod (1/p)$為一個週期當選為叢集頭的數量，G 是定義為在 $r \bmod (1/p)$回合中沒有被選為叢集頭的節點所形成之集合，$n = 1,2,\ldots$。n是無線感測

網路的節點數目。每回合中，如果節點擔任過叢集頭，則會把自己的$T(n)$設置為 0，使得剩餘的節點機率增加，因此保證節點有機會當選叢集頭。形成叢集階段各叢集頭使用 CSMA MAC 協定[4, 9]、廣播 ADV(Advertisement Message)訊息，其包含叢集頭本身的識別碼。非叢集頭的節點根據接收到的信號強度(Received Signal Strength Indication, RSSI)判定叢集頭與非叢集頭之間的距離，並且找出距離最近的叢集頭，發出 Join-REQ 訊息申請加入叢集。當叢集頭收到各成員請求的訊息後，將為所有成員規劃出分時多工存取(TDMA)排程，並且廣播排程 ADV_SCH 訊息告知叢集內所有成員所分配到的傳送時間排程。穩定狀態中的叢集成員按照自己分配的時槽傳送資料，叢集頭進行資料融合後發送給基地台。

HEED(Hybrid Energy-Efficient Distributed Clustering Approach) [24]類似於 LEACH，但選擇叢集頭是參考節點剩餘電能，並根據公式(2)計算節點成為叢集頭(CH)的機率：

$$CH_{prob} = C_{prob} \times \frac{E_{residual}}{E_{max}} \quad (2)$$

C_{prob}為初始網路中預計叢集頭數目佔全部節點的比例；$E_{residual}$為目前剩餘電能；E_{max}為最大電能。每個節點計算成為叢集頭的機率，再透過廣播與通訊範圍內的鄰居節點比較，機率最高的節點將會擔任叢集頭。

UCR(Unequal Cluster-based Routing)[11]將整個網路劃分為許多不同大小的叢集，緩解熱點問題。叢集範圍根據叢集頭距離基地台的長度來決定，叢集頭距離基地台愈短時，所擁有的叢集範圍愈小，叢集競爭範圍由公式(3)產生：

$$R_i = (1 - c\frac{d_{max} - d(s_i, BS)}{d_{max} - d_{min}})R_0 \quad (3)$$

R_0 是預定的最大競爭範圍，$(1-c)R_0$是最小競爭範圍，其中 c 是一個介於 0 到 1 之間的常數。d_{max}和d_{min}表示網路中節點與基地台最長和最短的距離，$d(s_i,\ BS)$表示叢集頭與基地台之間的距離。

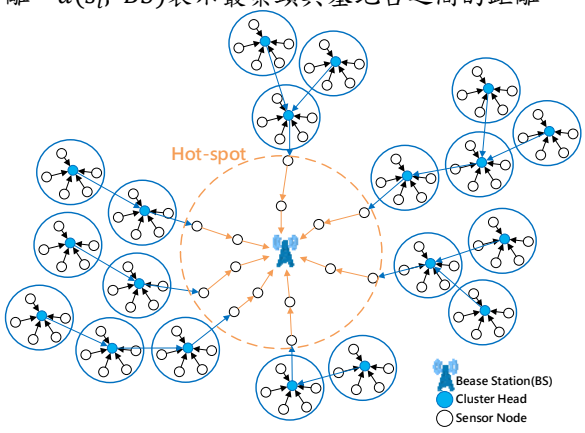


圖 2、混合多跳路由

混合路由(Hybrid routing)[12]結合平面多跳路由 (Flat multi-hop routing) 和分層多跳路由 (Hierarchical multi-hop routing)並且考慮熱點問題。平面多跳路由的每個節點透過多跳的最短路徑，將資料轉送到基地台。分層多跳路由，如 LEACH，將網路節點分為兩層，第一層為叢集頭(CH)，第二層為叢集成員(cluster member, CM)。CM 會選擇最靠近它的 CH，加入叢集並且將資料發送給叢集頭。混合多跳路由首先定義熱點的區域為基地台的最大傳輸範圍，分為熱點區域外與熱點區域兩部份，如圖 2。熱點區域外(藍色區域)減少資料量進入熱點區域，因為發送功率與資料量成正比，因此透過分層多跳路由的資料壓縮機制降低資料量，減少電能消耗。熱點區域內(澄色區域)盡可能降低每個單位的傳輸功率，採用平面多跳路由減緩熱點區域內的能量。

流量平衡路由 FBR (**F**low-**B**alanced **R**outing)[13]目標是同時實現電能效率和覆蓋保持(coverage preservation)。叢集形成時，考慮節點的感測區域重疊面積，簡稱重疊度，並選擇高重疊度的節點擔任叢集頭，目的是延長網路的覆蓋壽命。網路是以多層級骨幹(backbone)構建，使得叢集頭擁有多條路徑可達基地台，當骨幹上的叢集頭電能耗盡時，只對斷開的節點進行部份重建。叢集考慮到重疊度(overlapping degree)，假設每個節點的感測範圍半徑為r，當節點感測範圍有部分或完全重疊時，稱此節點為朋友節點，如圖 3，節點j和k為節點i的朋友。最終得出節點的感測半徑$2r$內的所有朋友節點。重疊度根據節點i與朋友j和k的感測重疊面積比(ρ_i)，範圍$0 \le \rho_i \le 1$，如公式(4)。

$$\rho_i = \frac{1}{A_i} \cup_{j,k \in F_i} A_i \cap A_{j,k} \quad (4)$$

其中A_i是節點i感測範圍，$j, k \in F_i$屬於節點i的朋友，$A_i \cap A_{j,k}$是節點i與朋友j和k的重疊面積。

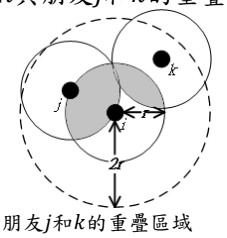


圖 3、節點i與它的朋友j和k的重疊區域

2.2. 叢集內多跳

DSBCA(**B**alanced **C**lustering **A**lgorithm with **D**istributed **S**elf-Organization)[14]，目的是產生更均衡的叢集能量消耗。透過節點的連接密度、剩餘電能和當選叢集頭時間為權重值，擁有愈高權重值的節點則優先成為叢集頭。叢集的形成考慮節點密度和節點與基地台的距離，計算叢集半徑(k)。如果兩個叢集具有相同的連接密度時，則離基地台較遠的叢集，具有較大的叢集半徑；如果兩個叢集距基地台相同時，較高連接密度的叢集，具有較小的叢集半徑，產生更平衡的電能消耗叢集結構。

- 節點u的連接密度：表示連接節點與鄰居節點的比例。

$$D_k(u) = \frac{|(t,v) \in E / t, v \in N_k(u) \cup \{u\}|}{|N_k(u)|} \quad (8)$$

其中$|N_k(u)|$是節點u的k跳鄰居數目。

- 節點u到基地台的距離D：當節點u到基地台的距離較短，其叢集範圍較小，反之較大。

上述演算法有各自的特性，我們採用某些特性，延長網路壽命。例如，以叢集為基礎的路由協定、

節點感測範圍的覆蓋情況、叢集大小的規劃、叢集內的多跳傳輸方式..等。

3. 提出的方法

我們提出的方法分成叢集形成演算法、路徑建置演算法以及重新路由演算法三個部分。

3.1. 叢集形成演算法

我們採用叢集式的協定，與現存方法有所不同，叢集過程只在網路初始時進行，目的是降低叢集重新組態的開銷。此外叢集頭將根據節點的重疊度、密度和剩餘電能來選擇，因重疊度高的節點將可以降低其死亡後對資料蒐集的影響，密度佳的節點可以形成較完整的叢集，剩餘電能則可避免過於頻繁地更換叢集頭以及能有更多電能傳遞資料。最後根據叢集頭的密度和基地台的距離，形成適當的叢集範圍，達成降低電能消耗的目標。

定義 1:每個節點皆有一個有限的感測範圍$(r、2r)$和傳輸範圍d，如圖 4 所示，節點i的感測區域範圍$(r、2r)$是為了方便計算感測重疊度；節點i的傳輸範圍$d = k \times 2r$，k是調整傳輸範圍的參數$(k = 1,2,...)$。

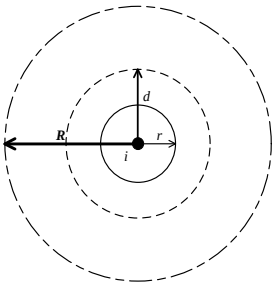


圖 4、節點的感測區域(r)、傳輸距離(d)和叢集範圍(R)

定義 2：節點i的感測範圍$(2r)$內的節點，稱為節點i的朋友節點。

定義 3：節點i的叢集範圍(R)內的節點，稱為節點i的鄰居節點。

3.2. 叢集頭的選擇

叢集頭選擇依據 4 個部份：(1)重疊度ρ，(2)密度C，(3)剩餘電能E，(4)權重W。

(1) 重疊度ρ：在初始狀態，每個節點廣播一個訊息，尋找周邊的朋友節點。接著利用公式(11)計算節點的重疊度，ρ_i值介於0 與 1。重疊面積愈大，代表該節點與朋友節點的相關度愈高，死亡後的所造成的影響將會愈低。

$$\rho_i = \frac{A_i \cap A_{F_i}}{A_i} \quad (11)$$

其中A_i為節點i感測面積。A_{F_i}為節點i的朋友節點感測面積。

(2) 密度C：是節點叢集範圍內密集程度，用於平衡節點間的電能損耗。公式(12)為感測節點的密度計算。

$$C(i) = \frac{|F(i)|}{|N(i)|} \quad (12)$$

其中$|F(i)|$是感測節點i的朋友節點數。$|N(i)|$為感測節點i的鄰居節點數。如果密度明顯小於 0.25，或密度明顯大於 0.25 的感測節點擔任叢集頭，可能造成叢集能量消耗不均勻，因此我們選擇密度趨近於 0.25，使得叢集可以有更均勻的能量消耗。

(3) 剩餘電能E：為選擇叢集頭的重要參數，可達到降低叢集頭的更換次數，公式(13)為該節點剩餘電能的百分比。

$$E(i) = \frac{E_{residual}(i)}{E_{max}(i)} \quad (13)$$

其中$E_{residual}(i)$為節點i的剩餘能量，$E_{max}(i)$為節點i的初始能量。

(4) 權重值W：推選出最高權重值的節點擔任叢集頭，代表此感測節點擁有較高的重疊度、適合的密度和較多的剩餘電能，使產生的叢集有較均勻的能量消耗和較長的生命週期。其中權重值包含感測節點的重疊度ρ、密度C與剩餘電能E，如公式(14)。

$$W(i) = \rho_i \times \alpha - |C(i) - 0.25| \times \beta + E(i) \times \gamma \quad (14)$$

其中α, β, γ為自定義的影響因素，介於 0.1~1 之間。ρ_i為節點i的重疊度。設計這個式子的目的是找出更合適的節點擔任叢集頭，在實驗中驗證α, β, γ，找出最佳值使叢集的生命週期最大化。

3.3. 決定叢集範圍

基地台距離$D(i)$：如公式(15)，為避免形成熱點，叢集頭距離基地台愈近，形成叢集範圍愈小，當叢集頭距離基地台愈遠時，形成的叢集範為愈大。

$$4. \quad D(i) = \frac{d_i}{d_{max}} \quad (15)$$

其中d_{max}為距離基地台最遠的節點，d_i為節點i到基地台的距離。

密度$C(i)$:當節點密度愈高時，造成叢集頭的電能消耗加重，最終影響叢集頭之間電能/壽命不均衡。因此我們調整密度高的區域，縮小叢集範圍。叢集的範圍形成後，可能使得有些節點被孤立，這些孤立節點將加入離它最近的叢集。

如果叢集範圍過大，則會加重叢集頭的負擔，使得叢集頭的電能消耗速度加快，如果叢集的範圍過小，則會形成過多的叢集，使得整個網路的壽命下降，加上熱點的問題，因此適當叢集範圍是很重要的。本文叢集大小是使用以下 2 種參數：基地台距離D，密度C，如公式(16)。

$$R_i = \left\lfloor \varphi \times \frac{D(i)}{C(i)} \right\rfloor \quad (16)$$

其中φ感測器具體應用參數，在實驗中我們將φ預設為 $2r$。叢集範圍改變後，可能會使得某些叢集範圍過小或過大的情況，將限制叢集範圍R_i的最小的範圍不可小於感測範圍 r。

節點的佈署方式通常為隨機分佈，可能會使某些叢集頭誕生於較偏遠的地區，造成傳輸電能負擔，因此我們設定叢集頭與基地台的距離限制，以防止叢集頭太過於遠離基地台。以下是導出選擇叢集頭的距離：

首先，將叢集頭的叢集範圍(R_i)給定一個擔任

叢集頭的最低門檻值(T)，如公式(17)：

$$R_i = \left\lfloor \varphi \times \frac{D(i)}{C(i)} \right\rfloor \geq T \quad (17)$$

接著，經由移項過後，可以得知感測節點i必須小於等於公式(18)，才有資格成為叢集頭。

$$d_i \leq \left\lfloor \left(1 - T \times \frac{C(i)}{\varphi}\right) d_{max} \right\rfloor \quad (18)$$

如果感測節點的W值較高，但距離不符合公式(18)，我們仍然不選擇它為叢集頭，因為它可能位於網路中較偏遠的地區。

Algorithm 1. Cluster formation algorithm

Initialization

1. Receive $CLUSTER\_FORM$ form the BS
2. find friends and neighbors
3. calculate $W(i)$ according to eq.(14)
4. set a timer t for the node, and then do **State Determination**

State Determination

1. **if** $t \geq 0$ **then**
2. bid for the head //競選叢集頭
3. **if** $(W_i = W_k (i < k,\ k \in N_i))$ && (Cluster head distance limitations to eq.(18)) **then**
4. become the head, calculate $R(i)$ according to eq.(15), according $R(i)$ range broadcast $HEAD$ message to neighbors, and then **exit**
5. **else if** receiver $HEAD$ from neighbor j **then**
6. become member of node j ,and then **exit**
7. **else**//孤立節點
8. find near head j, join member of node j ,and then **exit**
9. **end if**
10. **end if**

4. 路徑建置演算法

在找出叢集頭與叢集範圍後，將考慮叢集的路徑建置，我們分為叢集內與叢集間兩部分。

4.1. 叢集內路徑建置

叢集內的路徑採用自我穩定演算法，自我穩定演算法不須要系統初始化的特性，而且允許動態加入或退出，以及容錯優點，因此本文參考 Dijkstra 演算法的單源最短路徑(Single Source Shortest Paths)法，建立叢集頭與成員節點之間的最短路徑。連通圖$G = (V, E)$表示一個無線感測網路，其中V代表網路中的感測節點集合，$\forall i \in V$；E代表是鏈路集合，每條路徑$\{i, j\} \in E$表示感測節點i和j的雙向鏈接。變數定義如下：

1. 每一個節點都有一個變數d，以$d(i)$紀錄節點i最短路徑的值。
2. 每一個叢集都會有一個M集合，表示為叢集頭與成員組成的集合。
3. 每一個叢集都會有一個S集合，記錄已求出最短路徑節點所組成的集合。

演算法由規則 Rule0 和 Rule1 構成，以下我們將分成叢集頭與成員節點兩種情況介紹：

1. 叢集頭：定義源節點r為叢集頭，尋找叢集頭到每一個成員節點的最短路徑。
2. 成員節點：除了叢集頭以外的節點皆屬於該叢集頭的成員節點。

自我穩定演算法的規則如下：

[R0]**if** i = r & $S = \{\} \wedge d(r) \neq 0$ **then** $S = \{i\} \wedge d(r) = 0$

[R1]**if** i ≠ r ∧ $S \neq M \wedge d(i) \neq \min_{j \in N(i)} (d(j) + w(i,j))$ **then** $S = \{i\} \wedge d(i) = \min_{j \in N(i)} (d(j) + w(i,j))$ **if** $\left(d(i) = \min_{j \in N(i)} (d(j) + w(i,j))\right) == \left(d(i) = \min_{k \in N(i)} (d(k) + w(i,k))\right) \wedge \boldsymbol{E(j) > E(k)}$ **then** $S = \{i\} \wedge d(i) = \min_{j \in N(i)} (d(j) + w(i,j))$

4.2. 路徑建置(叢集間)

叢集間的訊息傳遞使用 Borůvka's 演算法，找出叢集頭通往基地台的最短路徑。首先假設無線感測網路為連通圖 G，每個頂點代表每個叢集頭，每個邊則是根據廣播的訊號強弱，判斷出叢集頭之間的距離。演算法的步驟如下：

1. 啟動與輸入連通圖 G，每個節點表示為叢集頭，其每個邊的權重表示與其它叢集頭之間的距離。
2. 初始化一棵樹 T。
3. 尋找每個叢集頭之間的最短路徑並加入 T 中。
4. 透過連接樹的邊，形成更大的樹。
5. 重複步驟 3-4，直到所有的叢集頭都加入 T 中。

假設叢集頭 a、b、c、d、e、f和 BS 為頂點。在初始的階段，叢集頭選擇完成後，將會廣播訊息告知周邊的叢集頭，並且根據訊號的強弱來推算出叢集之間的距離，也就是圖上的每個邊；每個叢集頭根據步驟 3 找出 $T_1 = \{BS, a, b, c\}$和 $T_2 = \{d, e, f\}$，2 棵樹；根據步驟 4，找出T_1和T_2樹中最短可連接的邊{b,e}，形成更大的樹$T = T_1 + T_2$;當所有的叢集頭皆在T中，則輸出最小生成樹T。

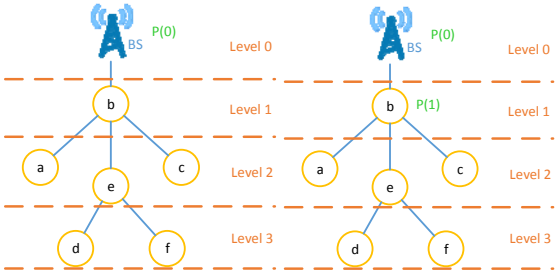


圖 5、(a)樹狀節點編號 1 與 (b)樹狀節點編號 2

建立一棵樹之後，我們將最小生成樹視為一層一層的樹狀結構進行節點編號。假定基地台(BS)為第 0 層並且為樹根節點，並將父親節點定義為自己，並且給予編號為$P(0)$，樹狀關係表示為$P(0) \rightarrow P(0)$，如圖 5(a)。接著第 0 層的樹根節點將尋找第

1 層的樹葉節點，因此 BS 為節點 b 的父親節點，並且給予編號$P(1)$，樹狀關係表示為$P(0) \rightarrow P(1)$，如圖 5(b)。第 1 層的樹葉節點將尋找第 2 層的樹葉節點，因此節點b為節點a,e,c的父親節點，並且給予編號$P(2),P(3),P(4)$，樹狀關係表示為$P(1) \rightarrow P(2),P(1) \rightarrow P(3),P(1) \rightarrow P(4)$，如圖 6(a)。第 2 層的樹葉節點將尋找第 3 層的樹葉節點，因此節點e為節點d,f的父親節點，並且給與編號$P(5),P(6)$，樹狀關係表示為$P(3) \rightarrow P(5),P(3) \rightarrow P(6)$，如圖 6(b)。

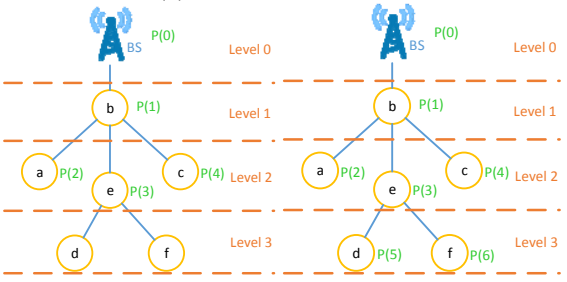


圖 6、(a)樹狀節點編號 3 與 (b)樹狀節點編號 4

4.3. 重新路由演算法

叢集演算法只在初始時執行一次，而每當叢集頭電能即將耗盡，將尋找節點替代，並且僅針對斷開的節點重新連接。如果叢集頭的剩餘能量小於預定之門檻值r_{th}，表示該叢集頭的能量即將耗盡。因此觸發重新路由演算法，讓目前的叢集頭從鄰居中找到新的叢集頭來替代自己。

重新路由演算法包含：叢集頭更換和路徑重新連接。當叢集頭i的能量低於門檻值時，將會廣播 HELP訊息告知叢集內的所有感測節點，並重新選擇最高權重值W的感測節點擔任新的叢集頭。新叢集頭誕生後將廣播 HEAD 訊息，告知叢集內的節點，並啟用路徑建置演算法規劃路徑。

Algorithm 3. Rerouting algorithm

Head Replacement executed by exhausted cluster head i

1. broadcast HELP message to neighbors
2. recalculate W according to eq.(14), without considering the exhausted node i
3. select the best node, as new cluster head, broadcast $HEAD$ message to neighbors, and **then** do **路徑建置演算法**

//從斷開的節點i執行重新連接

Path Reconnection executed by an exhausted node j

1. broadcast DEATH message to connected node of the node j
2. **if** receive response(s) from node j **then**
3. recalculate W according to eq.(14) without considering the exhausted node j
4. select the best node, denoted by i, as connection head
5. inform disconnect node to connect node i //通知斷開的節點連接節點 i
6. **end if**

當叢集頭i的電量低於預定的門檻值$R_e(i) < E(i) \times r_{th}$，將啟動 Head Replacement 演算法，並啟動路徑建置演算法，如 Head Replacement 1~3 行。

若節點j因能量耗盡即將消失於之前，將會發出一個DEATH訊息給與它連接的感測節點，感測節點接收到 DEATH 訊息後，並重新計算權重值W且不包含能量耗盡的節點j，選擇擁有最高的權重值W來擔任連接頭，最後通知斷開的感測節點連向連接頭，如 Path Reconnection 1~6 行。

5. 實驗結果

實驗結果是模擬叢集相關的方法 FBR、DSBCA 和我們提出的 CEMST 比較性能優劣，並分析這三種方法的網路壽命。採用 MATLAB 設計無線感測網路之模擬環境，我們模擬 100×100 平方公尺的網路面積，節點數目：200、300、400，且基地台是固定在(120,50)的座標，所有節點在初始階段會被分配一個唯一的識別號，初始能量為 0.5J，每回合需發送 200bits 的封包給基地台，傳輸的能量消耗使用[14]無線電電能消耗模型。參數值如表 1 所示，並使用以下假設。

1. 基地台位於感測區域外，且位置固定。
2. 感測器隨機佈署在感測區域內，且不具移動性。
3. 所有的感測器為同質，且能量均為有限。
4. 感測器偵測環境的能量消耗忽略不計。
5. 感測器都能與其他感測器或基地台直接溝通。
6. 感測器都不會進入睡眠的狀態。
7. 感測器在初始狀態將會被分配唯一的識別號。
8. 感測器收集一次資料且送回基地台視為一回合。

表 1、環境變數及模擬參數

參數項目	參數值
電能消耗模型	$Transmit:$ $E_{Tx}(l,d) = \begin{cases} lE_{elec} + l\varepsilon_{fs}d^2, & d < d_0 \\ lE_{elec} + l\varepsilon_{mp}d^4, & d \geq d_0 \end{cases}$ $Receive:$ $E_{Rx}(l) = lE_{elec}$
傳送、接收電能消耗E_{elec}	*50 nJ/bit*
放大器電能消耗ε_{fs} $(d < d_0)$	*10 pJ/bit//m²*
放大器電能消耗ε_{mp} $(d \geq d_0)$	*0.0013 pJ/bit/m⁴*
資料壓縮率	*0.7*
能量門檻比	*0.3*
數據聚集消耗	*5nJ/bit*

5.1. 叢集結果

FBR 的叢集頭選擇依據節點重疊度; 擁有高度重疊的節點，優先成為叢集頭，如圖 7、8、9。

DSBCA 的叢集頭選擇是依據感測節點的連接密度和剩餘電能，擁有高連接密度或愈多剩餘電能的節點優先成為叢集頭，如圖 10、11、12。

CEMST 的叢集頭選擇是依據感測節點的重疊度、密度和剩餘電能計算出權重值，並選擇高權重值的感測節點擔任叢集頭。叢集內的路徑是運用 Dijkstra's 演算法，叢集間的路徑則是 MST 演算法，縮短節點間的傳輸路徑，達成提高網路壽命的目標，如圖 13、14、15 所示。

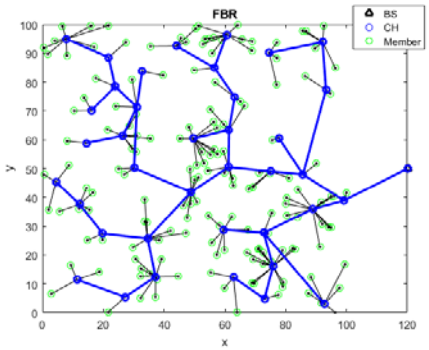


圖 7、FBR 模擬圖(1)(*Network Size =100m × 100m, Number of sensor nodes = 200*)

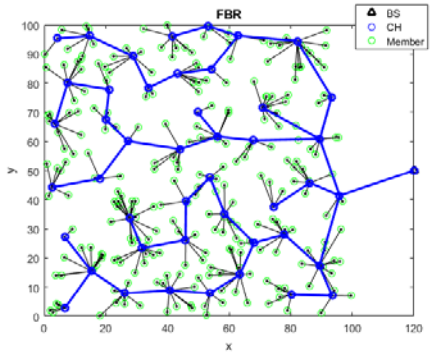


圖 8、FBR 模擬圖(2)(Network Size =100m × 100m, Number of sensor nodes = 300)

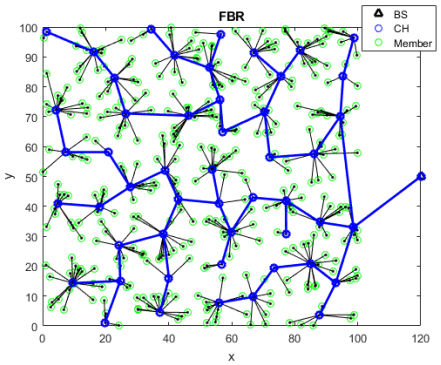


圖 9、FBR 模擬圖(3) (Network Size =100m × 100m, Number of sensor nodes = 400)

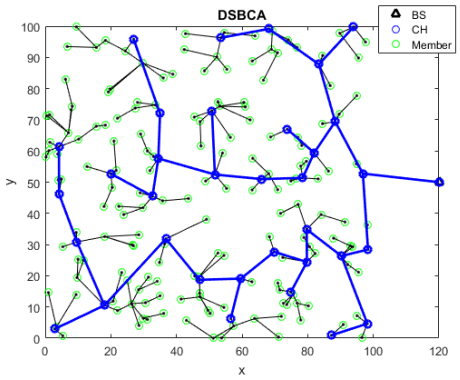


圖 10、DSBCA 模擬圖(1) (*Network Size =100m × 100m, Number of sensor nodes = 200*)

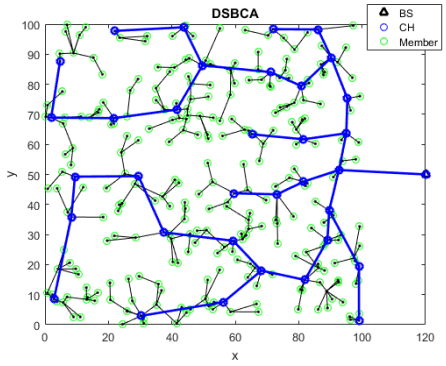


圖 11、DSBCA 模擬圖(2) (Network Size =100m × 100m, Number of sensor nodes = 300)

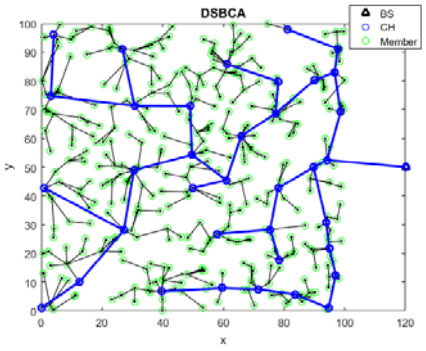


圖 12、DSBCA 模擬圖(3) (Network Size =100m × 100m, Number of sensor nodes = 400)

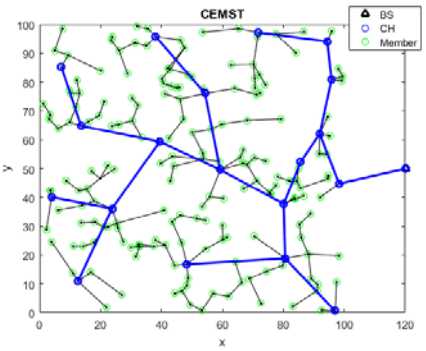


圖 13、CEMST 模擬圖(1) (Network Size =100m × 100m, Number of sensor nodes = 200)

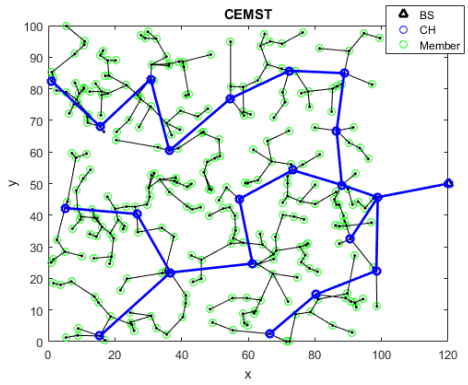


圖 14、CEMST 模擬圖(2) (Network Size =100m × 100m, Number of sensor nodes = 300)

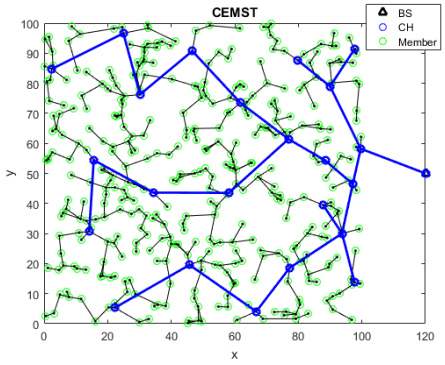


圖 15、CEMST 模擬圖(3) (Network Size =100m × 100m, Number of sensor nodes = 400)

5.2. 網路壽命

圖 16、17、18 是 200 個感測器佈署於 100m × 100m 區域的網路壽命，圖 16 是 CEMST(α=0.1，β=0.4)、DSBCA 和 FBR 演算法的節點存活時間，圖 17 是 CEMST(α=0.2，β=0.5)、DSBCA 和 FBR 演算法的節點存活時間，圖 18 是 CEMST(α=0.4，β=0.2)、DSBCA 和 FBR 演算法的節點存活時間。可以看出 16 在 20%個節點死亡之前，DSBCA 演算法優於 FBR 演算法，可能是 DSBCA 演算法在選擇叢集頭是考慮節點的連接密度和剩餘電能，暫時的延長節點壽命，但是可能增加部份節點的負擔，使得在 20%個節點死亡時，DSBCA 與 FBR 演算法的回合數是相近的，但在 20%個節點死亡之後，FBR 演算法則優於 DSBCA 演算法，FBR 演算法在選擇叢集頭的參數是考慮節點的重疊度，使得在部份節點死亡時，仍有較長的存活時間，而我們提出的 CEMST 演算在選擇叢集頭是考慮節點的重疊度、密度和剩餘電能，實驗結果顯示，CEMST 擁有更長的網路壽命。

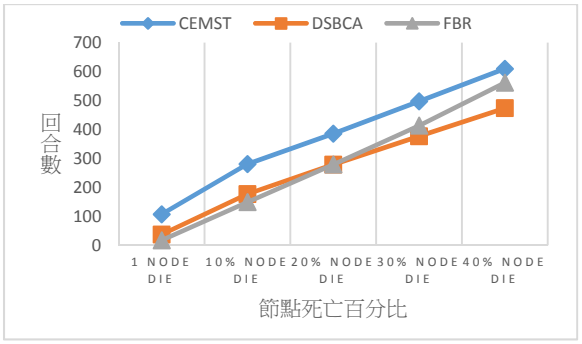


圖 16、CEMST(α=0.1, β=0.4)、DSBCA 和 FBR (Network Size =100m × 100m, Number of sensor nodes = 200)

6. 結論

在無線感測網路中，能源的消耗是決定網路存活時間的最大因素，因此需要一個良好的節能演算法延長網路壽命。本文採用叢集式方法來降低網路的能量消耗，但是叢集頭的選擇與其他方法不同。CEMST 根據節點的重疊度、密度與剩餘電能，推選出性能較佳的節點擔任叢集頭，降低並平衡叢集的能量消耗。在傳輸路徑方面，我們建立叢集內與叢集間路徑，叢集內的路徑是參考 Dijkstra's 最短路徑演算法，叢集外的路徑則使用 Borůvka's 最小生成樹演算法，縮短叢集間的路徑，降低叢集頭傳輸的能量消耗，達成延長網路壽命的目的。模擬結果顯示， CEMST 演算法在網路壽命上優於其他方法。

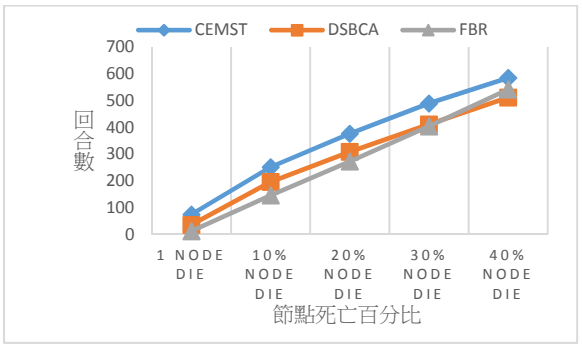


圖 17、CEMST(α=0.1, β=0.4)、DSBCA 和 FBR (Network Size =100m × 100m, Number of sensor nodes = 300)

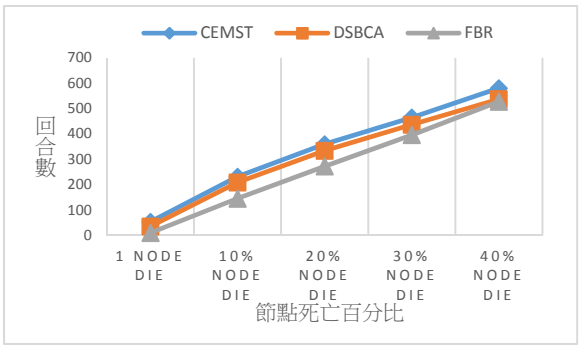


圖 18、CEMST(α=0.1, β=0.4)、DSBCA 和 FBR (Network Size =100m × 100m, Number of sensor nodes = 400)

誌謝

本研究感謝科技部計畫(編號 MOST 106-2221-E-025 -003-)補助支持，特此誌謝。

參考資料

[1] S. Tilak, N. B. Abu-Ghazaleh, and W. Heinzelman, "A taxonomy of wireless micro-sensor network models," *Mobile Comput. Commun. Rev.*, vol. 6, no. 2, pp. 28–36, 2002.

[2] Smart Dust Project, http://robotics.eecs.berkeley.edu/~pister/SmartDust/

[3] 無線感測器網路 (Wireless Sensor Networks), http://dblab.csie.ncku.edu.tw/sensor/

[4] Ye, W., Heidemann, J., & Estrin, D. (2002). An energy-efficient MAC protocol for wireless sensor networks. In INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE (Vol. 3, pp. 1567-1576). IEEE.

[5] King, V. (1997). A simpler minimum spanning tree verification algorithm. Algorithmica, 18(2), 263-270.

[6] Yong, Z., & Pei, Q. (2012). A energy-efficient clustering routing algorithm based on distance and residual energy for wireless sensor networks. Procedia Engineering, 29, 1882-1888.

[7] Mhatre, V., & Rosenberg, C. (2004). Design guidelines for wireless sensor networks: communication, clustering and aggregation. Ad Hoc Networks, 2(1), 45-63.

[8] Heinzelman, W. R., Chandrakasan, A., & Balakrishnan, H. (2000, January). Energy-efficient communication protocol for wireless microsensor networks. InSystem sciences, 2000. Proceedings of the 33rd annual Hawaii international conference on (pp. 10-pp). IEEE.

[9] Pahlavan, K., & Levesque, A. H. (2005). Wireless information networks (Vol. 93). John Wiley & Sons.

[10] Younis, O., & Fahmy, S. (2004). HEED: a hybrid, energy-efficient, distributed clustering approach for ad hoc sensor networks. Mobile Computing, IEEE Transactions on, 3(4), 366-379.

[11] Chen, G., Li, C., Ye, M., & Wu, J. (2009). An unequal cluster-based routing protocol in wireless sensor networks. Wireless Networks, 15(2), 193-207.

[12] Abdulla, A. E., Nishiyama, H., & Kato, N. (2012). Extending the lifetime of wireless sensor networks: A hybrid routing algorithm. Computer Communications, 35(9), 1056-1063.

[13] Tao, Y., Zhang, Y., & Ji, Y. (2013). Flow-balanced routing for multi-hop clustered wireless sensor networks. Ad hoc networks, 11(1), 541-554.

[14] Liao, Y., Qi, H., & Li, W. (2013). Load-balanced clustering algorithm with distributed self-organization for wireless sensor networks. Sensors Journal, IEEE, 13(5), 1498-1506.

[15] Heinzelman, W. B., Chandrakasan, A. P., & Balakrishnan, H. (2002). An application-specific protocol architecture for wireless microsensor networks.Wireless Communications, IEEE Transactions on, 1(4), 660-670.

Fast Algorithms for the Concatenated Longest Common Subsequence Problem with the Linear-space S-table*

Bi-Shiang Lin[a], Kuo-Tsung Tseng[b], Chang-Biau Yang[a†]and Kuo-Si Huang[c]
[a]Department of Computer Science and Engineering
National Sun Yat-sen University, Kaohsiung, Taiwan
[†]cbyang@cse.nsysu.edu.tw
[b]Department of Shipping and Transportation Management
National Kaohsiung University of Science and Technology, Kaohsiung, Taiwan
[c]Department of Information Management
National Kaohsiung University of Science and Technology, Kaohsiung, Taiwan

Abstract

Given two sequences A and B of lengths m and n, respectively, the consecutive suffix alignment *(CSA) problem is to compute the* longest common subsequence *(LCS) between A and each suffix of B. A two-dimensional S-table is constructed for solving the CSA problem. The linear-space S-table consists of the first row of the S-table and the changes between every two consecutive rows. Suppose that $A = A^{(1)}A^{(2)}$ (concatenation of two substrings), and we are given the S-table of $A^{(2)}$ and B, and the alignment result of $A^{(1)}$ and B. The* concatenated LCS *(CoLCS) problem is to find the alignment result of A and B. By using the linear-space S-table, instead of the 2-D S-table, we first propose an $O(n \log n)$-time algorithm to solve the CoLCS problem. Then, we propose a more efficient algorithm for the CoLCS problem, in $O(n)$ time, with the technique of set find and union.*

1 Introduction

The *longest common subsequence* (LCS) problem [2, 6, 8, 10, 12, 13, 15, 20, 22, 30] is a fundamental method for estimating the similarity between sequences. The LCS problem has been extensively studied for several decades since 1970. The LCS problem can be solved in $O(mn)$ time [13] by the dynamic programming approach, where m and n denote the lengths of the two input sequences, respectively. Lots of variant LCS problems were proposed, such as the *merged longest common subsequence* problem [14, 23, 29], which considers the LCS with the merged sequence, and the *constrained LCS* [4, 5, 9, 24, 27, 28], which computes the LCS with the constrained sequence.

The *consecutive suffix alignment* (CSA) problem is one of the variant LCS problems. Given two sequences A and B, the CSA problem is to compute the LCS between A and each suffix of B [16], where a suffix of a string means a substring starting at a certain position and ending at the last position. The *S-table* can be used to solve the CSA problem. The CSA problem can be used in various applications, such as the common substring alignment problem [18, 19], cyclic string comparison between two strings or between A and each suffix of B [17, 21, 25]. In 2003, Landau *et al.* [18] proposed a linear time algorithm with the given S-table to solve the common substring alignment problem.

In 2004, Landau *et al.* [16] proposed two algorithms to solve the CSA problem. One solves the problem in $O(nl)$ time and space with constant alphabets, and the other solves the problem in $O(nl+n|\log \Sigma|)$ time and $O(n)$ space, where $|\Sigma|$, l denote the alphabet size and the length of LCS, respectively. In 2005, Alves *et al.* [3] proposed another algorithm with $O(mn)$ time and $O(n)$ space for the CSA problem. In addition, Alves *et al.* [3] proposed the linear-space S-table, which consists of the first row of the S-table and the changes between every two consecutive rows.

Let $A = A^{(1)}A^{(2)}$ (concatenation of two substrings). And we already have the S-table of $A^{(2)}$ and B, and the alignment result of $A^{(1)}$ and B.

*This research work was partially supported by the Ministry of Science and Technology of Taiwan under contract MOST 104-2221-E-110-018-MY3.
†Corresponding author.

The *concatenated LCS* (CoLCS) problem is defined to calculate the alignment result of A and B. In this paper, we proposes two algorithms in $O(n \log n)$ and $O(n)$ time for solving the CoLCS problem with the linear-space S-table, instead of the 2-D S-table.

The organization of this paper is given as follows. Section 2 introduces the preliminary knowledge of the LCS and CSA problems, and the S-table. Next, our algorithms for the CoLCS problem are proposed in Section 3. Finally, the conclusions are given in Section 4.

2 Preliminaries

A sequence of characters is denoted as an upper-case letter, such as A or B. Taking sequence A as an example, the notations used in this paper are listed below.

- $A = a_1 a_2 \cdots a_m$.
- $|A|$: the length of sequence A.
- a_i: the ith character or element of A.
- $i..j$: an index range from position i to j.
- $A_{i..j}$: the substring of A from index i to j. Note that $A_{i..j} = \emptyset$ if $i > j$.

A subsequence of A is obtained by deleting an arbitrary number of characters (not necessarily consecutive) in A. For example, $A =$ `tctgatggt`, the subsequences of A may be `tctgatggt`, `catt`, `ctga`, `tatgt`, `a`, and so on. The longest common subsequence problem is defined as follows.

Definition 1. (LCS) *Given two sequences A and B with lengths m and n, respectively, the longest common subsequence (LCS) problem is to find the common subsequence between A and B with the maximal length.*

For example, suppose $A =$ `cggattctgt` and $B =$ `tctgatggt`. The LCS of A and B, denoted as $LCS(A, B)$, is `cgatgt` with length 6. The LCS problem can be solved by the grid directed acyclic graph (GDAG) [19] as shown in Figure 1.

Definition 2. ($P_G(i,j)$) *For $0 \le i \le m$ and $0 \le j \le n$, $P_G(i,j)$ is the value of the highest weight path from $G(0,0)$ to $G(i,j)$.*

With the GDAG, the length of LCS is equal to $P_G(m,n)$. The LCS problem can be solved through the dynamic programming (DP) approach in $O(mn)$ time by Equation 1 [30].

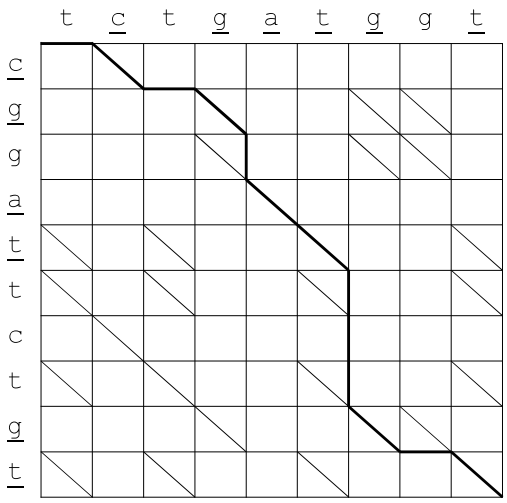


Figure 1: The grid directed acyclic graph (GDAG) for solving the LCS problem with $A =$ `cggattctgt` and $B =$ `tctgatggt`. Here, the path formed with thick lines is the LCS solution (`cgatgt`). Note that diagonal edges with weight 0 are not shown.

$P_G(i,j) =$

$$\max \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0, \\ P_G(i-1, j-1) + 1 & \text{if } a_i = b_j, \\ \max \begin{cases} P_G(i-1, j) \\ P_G(i, j-1) \end{cases} & \text{if } a_i \neq b_j. \end{cases} \tag{1}$$

2.1 The Consecutive Suffix Alignment Problem and the S-table

Definition 3. (consecutive suffix alignment) *Given two sequences A and B, the consecutive suffix alignment (CSA) problem is to compute the alignment of A and each suffix of B.*

With the above DP approach for the LCS problem, $|LCS(A, B_{1..j})|$ can be computed in $O(mn)$ time for all $1 \le j \le n$. The naïve method for the CSA problem with the DP approach requires $O(mn^2)$ time by computing each $|LCS(A, B_{i..j})|$, for $0 \le i \le j \le n$. However, it is inefficient. In the GDAG, the CSA problem can be transformed to finding the maximal weight path from $G(0,i)$ to $G(m,j)$ for $0 \le i \le j \le n$.

Definition 4. ($C_G(i,j)$) *For $0 \le i \le j \le n$, $C_G(i,j)$ is the maximal weight of all the paths from $G(0,i)$ to $G(m,j)$.*

Table 1: The matrix C_G with $A =$ `ttct` and $B =$ `tctgatggt`.

i \ j	0	1	2	3	4	5	6	7	8	9
0	0	1	2	3	3	3	3	3	3	3
1	0	0	1	2	2	2	2	2	2	3
2	0	0	0	1	1	1	2	2	2	3
3	0	0	0	0	0	0	1	1	1	2
4	0	0	0	0	0	0	1	1	1	2
5	0	0	0	0	0	0	1	1	1	2
6	0	0	0	0	0	0	0	0	0	1
7	0	0	0	0	0	0	0	0	0	1
8	0	0	0	0	0	0	0	0	0	1
9	0	0	0	0	0	0	0	0	0	0

Table 2: The S-table S of $A =$ `ttct` and $B =$ `tctgatggt`, where the starting index means of the position of B, and the column of D means that the number is the first occurrence in the row.

		Length				
		0	1	2	3	D
Starting index	0	0	1	2	3	
	1	1	2	3	$\underline{9}$	$\underline{9}$
	2	2	3	$\underline{6}$	9	$\underline{6}$
	3	3	6	9	∞	∞
	4	$\underline{4}$	6	9	∞	$\underline{4}$
	5	$\underline{5}$	6	9	∞	$\underline{5}$
	6	6	9	∞	∞	∞
	7	$\underline{7}$	9	∞	∞	$\underline{7}$
	8	$\underline{8}$	9	∞	∞	$\underline{8}$
	9	9	∞	∞	∞	∞

With Definition 4, it is clear that $C_G(i,j) = |LCS(A, B_{i+1..j})|$. Table 1 shows an example of C_G with $A =$ `ttct` and $B =$ `tctgatggt`.

Some properties in C_G are listed as follows.

- For each row in C_G, the value starts from 0.
- For each row in C_G, the values from left to right are nondecreasing.
- For each row in C_G, the difference between two consecutive values is either 0 or 1.

With the above properties, C_G can be represented by Table 2, denoted as S.

Definition 5. [3] (S-table) *For $0 \le i \le n$, $S_{i,0} = i$. For $0 \le i \le n$ and $0 < j \le L$, where L is the maximal value in C_G, $S_{i,j}$ is the minimum of k for $C_G(i,k) = j$. If no such k exists, $S_{i,j} = \infty$.*

The first element in $S_{i,*}$ (row i of S) is $S_{i,0} = i$, and each remaining element in $S_{i,*}$ records the index of the column which is the leftmost of each number appears in row i of C_G. For example, in row 3 of C_G, the leftmost 1 appears at the column 6, so $S_{3,1} = 6$. Alves *et al.* proposed and proved the following property of S-table [3].

Theorem 1. [3] *For $0 \le i < n$ in S,*

1. *Exactly one element of $S_{i,*}$ does not appear in $S_{i+1,*}$, which is $S_{i,0} = i$.*
2. *At most one element with a finite value in $S_{i+1,*}$ does not appear in $S_{i,*}$.*

Definition 6. [3] *(D) For $0 < i \le n$, d_i records the element which appears in $S_{i,*}$ but not in $S_{i-1,*}$. If there is no such new element, we set $d_i = \infty$.*

An example of D is shown in the rightmost column of Table 2. The S-table can be constructed from g the $S_{0,*}$ and D. Therefore, the solution of the CSA problem can be represented with the S-table, or $S_{0,*}$ and D [3]. In the following, the *linear-space S-table* means the first row of the S-table ($S_{0,*}$) and D.

2.2 Solving the Concatenated LCS Problem with the S-table

In this subsection, we use an example to explain how to solve the LCS problem with multiple common substrings by the S-table [18, 19]. Suppose we are given two strings $A =$ `cggattctgt` and $B =$ `tctgatggt`, where A is formed by concatenating three substrings $A^{(1)} =$ `cgga`, $A^{(2)} =$ `ttct` and $A^{(3)} =$ `gt`. In other situations, $A^{(r)}$ may repeat several times, but not consecutively, to form a longer sequence A. Note the S-table of $A^{(2)}$ and B has been already established in Table 2. Figure 2 shows the GDAG of A and B, which is composed of three subgraphs, corresponding to $A^{(1)}$, $A^{(2)}$ and $A^{(3)}$, respectively. The alignment result of the first subgraph can be viewed as the input of the second subgraph, and the alignment result of the second subgraph can be viewed as the input of the third subgraph.

Let $G^{(r)}$ denote subgraph r, whose input and output are denoted as $I^{(r)}$ and $O^{(r)}$, respectively. In addition, let $S^{(r)}$ denote the S-table of $A^{(r)}$ and B. The goal is to get the output $O^{(r)}$ with the input $I^{(r)}$ and S-table $S^{(r)}$. The following DP formula can be easily obtained [18, 19].

$$O_j^{(r)} = \max\{I_i^{(r)} + C_{G^{(r)}}(i,j)\}, \text{ for } 0 \le i \le j. \quad (2)$$

For example, $O_3^{(2)} = 3 = \max\{0+3, 0+2, 1+1, 1+0\}$. The value of $O_3^{(2)}$ comes from the input

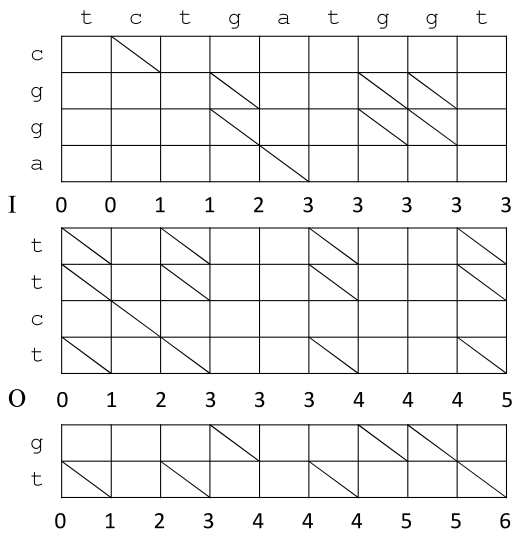


Figure 2: The GDAG composed of three subgraphs with $A^{(1)} = \texttt{cgga}$, $A^{(2)} = \texttt{ttct}$, $A^{(3)} = \texttt{gt}$ and $B = \texttt{tctgatggt}$.

$I^{(2)}$ and $G^{(2)}$. The case from $I_3^{(2)}$ can be ignored since $I_2^{(2)} = I_3^{(2)} = 1$ and $C_{G^{(2)}}(2,3) \geq C_{G^{(2)}}(3,3)$. Similarly, $I_1^{(2)}$ can be ignored since $I_0^{(2)} = I_1^{(2)} = 0$. As another example, $O_6^{(2)} = 4 = \max\{0+3, 0+2, 1+2, 1+1, 2+1, 3+1, 3+0\}$. Thus, only the leftmost position of $I^{(r)}$ with value k is needed to compute $O^{(r)}$. Let PI_k denote the smallest index in $I^{(r)}$ with value k, and PO_i denote the smallest index in $O^{(r)}$ with value i. In this example, $PI = \langle 0, 2, 4, 5\rangle$ and $PO = \langle 0, 1, 2, 3, 6, 9\rangle$. Now, $O^{(r)}$ can be represented as PO in Equation 3 [18].

$$PO_i = \min\{j | k + C_{G^{(r)}}(PI_k, j) = i, \quad 0 \leq k \leq i \text{ and } 0 \leq j \leq n.\} \tag{3}$$

With the S-table $S^{(r)}$, Equation 3 can be transformed into Equation 4.

$$PO_i = \min_{0 \leq k \leq i} \{S^{(r)}_{PI_k, i-k}\}, \text{ if } S^{(r)}_{PI_k, i-k} \text{ exists.} \tag{4}$$

For example, the smallest column index of value 4 in $O^{(2)}$, denoted by PO_4, is obtained by $\min\{S^{(2)}_{PI_0,4}, S^{(2)}_{PI_1,3}, S^{(2)}_{PI_2,2}, S^{(2)}_{PI_3,1}, S^{(2)}_{PI_4,0}\}$ $= \min\{S^{(2)}_{0,4}, S^{(2)}_{2,3}, S^{(2)}_{4,2}, S^{(2)}_{5,1}, S^{(2)}_{\infty,0}\} = \min\{-, 9, 9, 6, -\} = 6$. It means that $|LCS(A^{(1)}, B_{1..2})| + |LCS(A^{(2)}, B_{3..9})| = 1 + 3 = 4$, $|LCS(A^{(1)}, B_{1..4})| + |LCS(A^{(2)}, B_{5..9})| = 2 + 2 = 4$, and $|LCS(A^{(1)}, B_{1..5})| + |LCS(A^{(2)}, B_{6..6})| = 3 + 1 = 4$. And 6 is the leftmost index of B to get LCS length 4.

Definition 7. *(M)* $M_{k,i} = S_{PI_k, i-k}$, *for* $0 \leq k \leq |PI| - 1$ *and* $k \leq i \leq k + L$ *if such* $S_{PI_k,i-k}$ *exists.*

With the definition of matrix M, $PO_i = \min_{0 \leq k \leq |PI|-1}\{M_{k,i}\}$, for $0 \leq i \leq L$. The matrix M is shown in Table 3. The computation of PO is equivalent to finding the minimum of each column in M.

Table 3: The matrix M, where the row index means that of M, and each number in the bottom row is the column minimum.

M		Length 0	1	2	3	4	5
Row index	0	0	1	2	3		
	1		2	3	6	9	
	2			4	6	9	
	3				5	6	9
Minimum		0	1	2	3	6	9

To find the column minimum, the brute-force method needs $O(nl)$ time to examine all the numbers, where l denotes the length of $LCS(A^{(r-1)} + A^{(r)}, B)$. Note that the symbol + means the concatenation strings $A^{(r-1)}$ and $A^{(r)}$. The matrix M has been proved to be a *totally monotone matrix* [18]. Therefore, a recursive algorithm, named SMAWK and proposed by Aggarwal *et al.* [1], can find the column minimum of a totally monotone matrix in $O(l)$ time. With the S-table $S^{(r)}$ and the input $I^{(r)}$, the alignment of $G^{(r)}$ can be computed in $O(l)$ time, instead of the original DP approach in $O(mn)$ time.

In summary, given two substrings $A^{(1)}$ and $A^{(2)}$ and one string B with the S-table $S^{(2)}$ of $A^{(2)}$ and B, the *concatenated LCS* (CoLCS) problem is to find the LCS length of $A^{(1)} + A^{(2)}$ and B. It can be solved in $O(l)$ time [18].

3 Our Algorithms for the Concatenated LCS Problem

In this section, we propose two new algorithms for solving the CoLCS problem in $O(n \log n)$ and $O(n)$ time with the linear-space S-table: $S_{0,*}$ and D, instead of using the whole S-table.

3.1 The Alignment with the Linear-Space S-table

For easy explanation, we denote the infinity symbol ∞ mentioned in S-table and D as ∞_1, ∞_2, $\cdots$, and so on. Therefore, Table 2 is modified and shown in Table 4.

Table 4: The modified S-table and D with $A =$ `ttct` and $B =$ `tctgatggt`, where the starting index means of the position of B, and the value in column D means that the number is the first occurrence in the row.

S-table	Starting index	Length 0	1	2	3	D
	0	0	1	2	3	
	1	1	2	3	$\underline{9}$	$\underline{9}$
	2	2	3	$\underline{6}$	9	$\underline{6}$
	3	3	6	9	$\underline{\infty_1}$	$\underline{\infty_1}$
	4	$\underline{4}$	6	9	∞_1	$\underline{4}$
	5	$\underline{5}$	6	9	∞_1	$\underline{5}$
	6	6	9	∞_1	$\underline{\infty_2}$	$\underline{\infty_2}$
	7	$\underline{7}$	9	∞_1	∞_2	$\underline{7}$
	8	$\underline{8}$	9	∞_1	∞_2	$\underline{8}$
	9	9	∞_1	∞_2	$\underline{\infty_3}$	$\underline{\infty_3}$

The modified computation matrix M is shown in Table 5(a). Clearly, the same result is obtained if only the finite values are considered when the column minimums in M are computed. The finite values are considered as the output. The minimum of column 6 is ∞_1, so we can ignore it. The output of Table 5(a) is identical to Table 3.

Property 1. *Once a number k appears in $S_{i,*}$, k must appear in $S_{j,*}$ for $i \le j \le k$.*

Definition 8. *Let $C_{k,j}$ denote the minimum of $M_{0..k,j}$, for $0 \le k \le |PI| - 1$ and $0 \le j \le k + L$. And, let h_k denote the maximum of $M_{k,*} \setminus C_{k-1,*}$ ($M_{k,*}$ with excluding $C_{k-1,*}$), where the symbol $\setminus$ denotes the set difference operation.*

For example, the matrix C is shown in Table 5(b). And, $h_1 = 9$, $h_2 = \infty_1$ and $h_3 = 6$. With the above definition, the alignment result $PO = C_{|PI|-1,*} = C_{3,*} = \langle 0, 1, 2, 3, 6, 9, \infty_1 \rangle$. We present a property of two consecutive $C_{k-1,*}$ and $C_{k,*}$ as follows.

Theorem 2. *$C_{k-1,*} \cup \{h_k\} = C_{k,*}$, for $1 \le k \le |PI| - 1$.*

Proof. Let j be the smallest index for $C_{k-1,j} > M_{k,j}$. We can divide $C_{k,*}$ into two parts by index j as follows.

1. $0 \le i < j$. In this case, $C_{k-1,i} \le M_{k,i}$. Thus, $C_{k,i} = \min\{C_{k-1,i}, M_{k,i}\} = C_{k-1,i}$.
2. $j \le i \le k + L$. Because $C_{k-1,j} > M_{k,j}$, we have $C_{k,j} = \min\{C_{k-1,j}, M_{k,j}\} = M_{k,j}$. The value of $C_{k-1,i}$ comes from one number in rows $PI_0, PI_1, \cdots, PI_{k-1}$ of S. Because $C_{k-1,i} \ge PI_k$, with Property 1, $C_{k-1,i}$ appears in $M_{k,*}$ after $M_{k,j}$ for all i. Therefore, $C_{k,i+1} = C_{k-1,i}$.

Thus, $C_{k-1,*} \cup \{h_k\} = C_{k,*}$, where $h_k = M_{k,j}$. □

Table 5: The modified matrix M and C, where $PI = \langle 0, 2, 4, 5 \rangle$. (a) The matrix M, where each number in the bottom is the column minimum. (b) The matrix C.

(a)

M	Row index	Length 0	1	2	3	4	5	6
	0	0	1	2	3			
	1		2	3	6	9		
	2			4	6	9	∞_1	
	3				5	6	9	∞_1
Minimum		0	1	2	3	6	9	∞_1

(b)

C	Row index	Length 0	1	2	3	4	5	6
	0	0	1	2	3			
	1	0	1	2	3	9		
	2	0	1	2	3	9	∞_1	
	3	0	1	2	3	6	9	∞_1

For example in Table 5, $C_{1,*} = \{0, 1, 2, 3, 9\} = \{0, 1, 2, 3\} \cup \{9\} = C_{0,*} \cup \{9\}$, where $C_{0,*} = S_{0,*}$. $C_{2,*} = \{0, 1, 2, 3, 9\} \cup \{\infty_1\}$ and $C_{3,*} = C_{2,*} \cup \{6\}$, where 6 is the maximum of $M_{3,*} \setminus C_{2,*}$.

With the above properties and $PO = C_{|PI|-1,*}$, we can compute PO from $C_{0,*}$ sequentially where $C_{0,*} = S_{0,*}$. The alignment result PO consists of $S_{0,*}$ and h_k for $1 \le k \le |PI| - 1$. Since $S_{0,*}$ has already been given, we focus on finding h_k. We first propose an $O(n \log n)$-time algorithm, and then a linear time algorithm.

3.2 An $O(n \log n)$-time Algorithm

We first find the value of h_k in a sequential method with $k = 1$ to $|PI| - 1$ sequentially.

Lemma 1. *$S_{i,*}$ consists of the L largest numbers in $S_{0,*} \cup D_{1..i}$, where $D_{i..j}$ denotes $\{d_i, d_{i+1}, \cdots, d_j\}$.*

Proof. Each element in $S_{i,*}$ is greater than or equal to i. With Theorem 1, the smallest number of $S_{i-1,*}$ does not appear in $S_{i,*}$. Then, with $|S_{i,*}| = L$, the elements of $S_{i,*}$ are the L largest numbers in $S_{0,*} \cup D_{1..i}$. □

For example in Table 4, $S_{2,*}$ consists of the four largest numbers in $\{0, 1, 2, 3\} \cup \{9, 6\}$.

Theorem 3. *$h_k = \max(D_{1..PI_k} \setminus H_{1..k-1})$, for $1 \le k \le |PI| - 1$.*

Proof. By Definition 7, $M_{k,i} = S_{PI_k,i-k}$. With ignoring the detailed column index and applying the set concept, we have $M_{k,*} = S_{PI_k,*}$. By Definition 8 and Lemma 1, h_k is the maximum of $(S_{0,*} \cup D_{1..PI_k}) \setminus C_{k-1,*}$. Since $C_{0,*} = S_{0,*} \subseteq C_{k-1,*}$, we have that h_k is the maximum of $D_{1..PI_k} \setminus C_{k-1,*}$. By Theorem 2, $C_{k-1,*} \cup \{h_k\} = C_{k,*}$, so $C_{k-1,*} = C_{0,*} \cup \{h_1\} \cup \{h_2\} \cup \cdots \cup \{h_{k-1}\}$. We get

$$\begin{aligned} h_k &= \max(D_{1..PI_k} \setminus \{h_1, h_2, \cdots, h_{k-1}\}) \\ &= \max(D_{1..PI_k} \setminus H_{1..k-1}). \end{aligned} \tag{5}$$

□

By Theorem 3, we can use a sequential method to find h_k by querying the range maximum of D, and remove h_k from D after finding. Take Tables 4 and 5 as an example, where $PI = \langle 0, 2, 4, 5\rangle$ and $M_{0,*} = \langle 0, 1, 2, 3\rangle$. The sequential process is described as follows.

(1) $PI_1 = 2$, so we find h_1 in $\max(D_{1..2}) = 9$, and remove 9.

(2) $PI_2 = 4$ and $\max(D_{1..4}) = \infty_1$, so $h_2 = \infty_1$ and we remove ∞_1.

(3) $PI_3 = 5$ and $\max(D_{1..5}) = 6$, so $h_3 = 6$. Therefore, $PO = C_{3,*} = \langle 0, 1, 2, 3, 6, 9, \infty_1\rangle$.

The range maximum query and single point update (removal) of D with the segment tree structure requires $O(\log n)$ time for each operation [7] . Thus, the problem for finding the LCS of $A^{(r-1)} + A^{(r)}$ and B needs $\mathrm{O}(|PI| \log n) = \mathrm{O}(n \log n)$ time, when PI, $S^{(r)}_{0,*}$ (row 0 of S-table of $A^{(r)}$ and B) and $D^{(r)}$ are given. The algorithm is presented in Algorithm 1.

Algorithm 1 An $\mathrm{O}(n \log n)$-time algorithm

Input: PI, $S_{0,*}$ and D
Output: PO

```
1: PO = S_{0,*} // insert each of S_{0,*} into PO
2: for k = 1 to |PI| − 1 do
3:     i = max(D_{1..PI_k}) // i is the index of the range maximum
4:     insert d_i into PO // h_k = d_i
5:     d_i = −∞ // remove d_i from D
6: return PO
```

3.3 An O(n)-time Algorithm

In Section 3.2, we compute h_k with the range maximum of $D_{1..PI_k}$, for $1 \le k \le |PI| - 1$, and remove h_k after finding. Now we focus on whether the number d_i will become the value of one h_k or not.

Definition 9. *$nextPI(d_i)$ is the smallest PI_k such that $i \le PI_k$.*

The $nextPI$ of Table 4 is shown in Table 6, where $PI = \langle 0, 2, 4, 5\rangle$. For example, $nextPI(d_1) = nextPI(9) = 2$ means that the smallest PI_k satisfying $1 \le PI_k$ is 2.

Table 6: An example of $nextPI$. If $nextPI(d_i)$ does not exist, we keep it empty. $PI = \langle 0, 2, 4, 5\rangle$ is underlined in column i.

i	d_i	$nextPI(d_i)$
1	9	2
<u>2</u>	6	2
3	∞_1	4
<u>4</u>	4	4
<u>5</u>	5	5
6	∞_2	
7	7	
8	8	
9	∞_3	

With the preprocessing of $nextPI$, we explain how to compute h_k for $1 \le k \le |PI| - 1$. We check the numbers in $D = \{d_1, d_2, \cdots, d_n\}$ with the decreasing order of the d_i value. If $nextPI(d_i)$ is empty, we ignore it. The computation process is demonstrated as follows.

(1) $nextPI(\infty_3)$ and $nextPI(\infty_2)$ are empty, so we ignore them.

(2) $nextPI(\infty_1) = 4 = PI_2$. ∞_1 appears in the S-table after row $PI_1 = 2$. So ∞_1 should be the new member from $C_{1,*}$ to $C_{2,*}$. In other words, $h_2 = \max(D_{1..PI_2}) = \max(D_{1..4}) = \infty_1$, because we check the numbers of D in decreasing order.

(3) $nextPI(9) = 2 = PI_1$. So $h_1 = 9$.

(4) $nextPI(8)$ and $nextPI(7)$ are empty, so we ignore them.

(5) $nextPI(6) = 2$. We find $PI_1 = 2$, and h_1 has been already determined, so we check next of PI_1. Again, we find $PI_2 = 4$, and h_2 has been already determined, so we check PI_3. Thus, we have $h_3 = 6$.

(6) We finally get $h_1 = 9$, $h_2 = \infty_1$ and $h_3 = 6$, and $PO = S_{0,*} \cup H_{1..3} = \langle 0, 1, 2, 3, 6, 9, \infty_1\rangle$.

Since we check the numbers in D from the largest to the smallest, by Theorem 2, the above process can correctly find which h_k should be of the value d_i.

When we examine d_i, we use the *union-find* data structure [11, 26] to check whether PI_k and

h_k have been determined or not. If PI_k and h_k have been determined, we have to try the next, PI_{k+1} and h_{k+1}. The operations in the union-find data structure are listed as follows.

- $make(x, C)$: Create a new set named C containing exactly x.
- $find(x)$: Find the name of the set containing x.
- $union(x, y, C)$: Unite the set containing x and the set containing y into a new set named C.

In the union-find data structure, each number in PI except PI_0 is initially in a unique set, implemented by $make(PI_k, k)$ for $1 \le k \le |PI| - 1$. We also use $make(\infty, |PI|)$ to set the boundary. Our algorithm for finding PO is presented in Algorithm 2, where D is sorted in decreasing order.

Algorithm 2 An O(n)-time algorithm

Input: PI, $S_{0,*}$, D and $nextPI$, where D is sorted in decreasing
Output: PO

```
PO = S_{0,*} // insert each of S_{0,*} into PO
for k = 1 to |PI| − 1 do
    make(PI_k, k)
make(∞, |PI|) // set the boundary
for d_i ∈ D from the largest to the smallest number do // decreasing order, achieved by bucket sort
    if nextPI(d_i) exists and find(nextPI(d_i)) ≠ |PI| then
        set k = find(nextPI(d_i))
        insert d_i into PO // h_k = d_i
        union(PI_k, PI_{k+1}, find(PI_{k+1}))
return PO
```

Figure 3 shows an example of the union-find process, with Table 4 and $PI = \langle 0, 2, 4, 5\rangle$. In this case, $|PI| = 4$ is the boundary number. We start from checking the largest number in D, which is ∞_3. The detailed steps are shown as follows.

1. $nextPI(\infty_3)$ and $nextPI(\infty_2)$ are empty, so skip them.
2. $nextPI(\infty_1) = PI_2 = 4$, and $k = find(4) = 2 \neq |PI| = 4$, so we have $h_2 = \infty_1$ and $union(PI_2, PI_3, find(PI_3)) = union(PI_2, PI_3, 3)$. In this situation, h_2 with PI_2 has been determined. If h_2 is desired to be set next time, $union(PI_2, PI_3, 3)$ guarantees to set h_3 with PI_3, instead of h_2. In other words, when either h_2 or h_3 may be set next time, we always set h_3.
3. $nextPI(9) = PI_1 = 2$, and $k = find(2) = 1 \neq |PI| = 4$, so we have $h_1 = 9$ and $union(PI_1, PI_2, find(PI_2)) = union(PI_1, PI_2, 3)$. After $union(PI_1, PI_2, 3)$ is performed, if one of h_1, h_2 and h_3 is desired to be set next time, we always set h_3.
4. $nextPI(8)$ and $nextPI(7)$ are empty, so skip them.
5. $nextPI(6) = 4$ and $k = find(4) = 3 \neq |PI| = 4$. So $h_3 = 6$, and $union(PI_3, PI_4, find(PI_4)) = union(PI_3, PI_4, 4)$.
6. The algorithm finishes after $H_{1..3}$ are found. If we check the next number $d_i = 5$, $nextPI(5) = PI_3 = 5$, and $find(5) = |PI| = 4$. $d_i = 5$ cannot be the value of any h_k.

Finally, the output is $\langle 0, 1, 2, 3, 6, 9, \infty_1\rangle$, where the elements $\langle 0, 1, 2, 3\rangle$ come from $S_{0,*}$.

For the general union-find problem, the time required for each operation of union or find is $O(\beta)$, where the lower bound of β was proved to be functional inverse of Ackermann's function [26]. The union-find structure we use is a single path tree, and we only unite two consecutive sets. With the definition of static tree set union, the time complexity of each operation is reduced to O(1) [11]. Our algorithm needs O(n) operations of find and

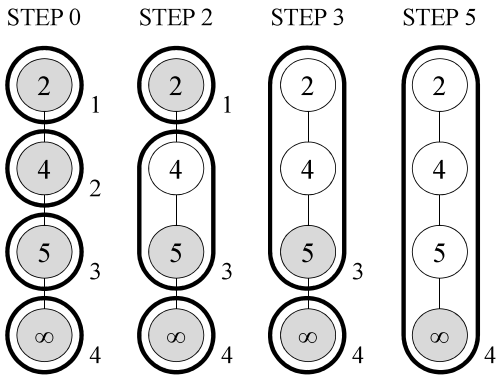


Figure 3: An example of the union-find process. Each thin circle is an element, and each bold circle is a set. The number beside each set is the name of the set.

union, so the time complexity is $O(n)$. The alignment result can be computed in linear time, when the linear-space S-table is given. In addition to get PO with increasing order, we can collect the elements h_k, and apply the bucket sort on these elements with an array of size n. It needs $O(n)$ time. In summary, the concatenated LCS problem with the linear-space S-table can be solved in linear time.

4 Conclusion

In the previous studies of the S-table, the whole S-table of quadratic space is needed for further applications. Due to the growth of data size, to reduce the required space is an important issue. This paper considers the linear-space S-table, which consists of the first row of the S-table and the changes between every two consecutive rows. New algorithms are proposed to solve the concatenated LCS problem in $O(n \log n)$ and $O(n)$ time with given the linear-space S-table, instead of the whole S-table reconstruction.

References

[1] A. Aggarwal, M. Klawe, S. Moran, P. Shor, and R. Wilber, "Geometric applications of a matrix searching algorithm," *Proceedings of the Second Annual Symposium on Computational Geometry*, New York, USA, pp. 285–292, ACM, 1986.

[2] L. Allison and T. I. Dix, "A bit-string longest-common-subsequence algorithm," *Information Processing Letters*, Vol. 23, No. 5, pp. 305–310, 1986.

[3] C. E. R. Alves, E. N. Cáceres, and S. W. Song, "An all-substrings common subsequence algorithm," *Electronic Notes in Discrete Mathematics*, Vol. 19, pp. 133–139, 2005.

[4] H.-Y. Ann, C.-B. Yang, and C.-T. Tseng, "Efficient polynomial-time algorithms for the constrained lcs problem with strings exclusion," *Journal of Combinatorial Optimization*, Vol. 28, No. 4, pp. 800–813, Nov. 2014.

[5] H.-Y. Ann, C.-B. Yang, C.-T. Tseng, and C.-Y. Hor, "Fast algorithms for computing the constrained lcs of run-length encoded strings," *Theoretical Computer Science*, Vol. 432, pp. 1–9, May 2012.

[6] A. Apostolico, "String editing and longest common subsequences," *Handbook of Formal Languages*, pp. 361–398, Springer, 1997.

[7] J. L. Bentley, "Algorithms for klee's rectangle problems." Technical Report, Computer Science Department, Carnegie Mellon University, 1977.

[8] L. Bergroth, H. Hakonen, and T. Raita, "A survey of longest common subsequence algorithms," *Proceedings of Seventh International Symposium on String Processing and Information Retrieval*, A Coruña, Spain, pp. 39–48, IEEE, 2000.

[9] F. Y. Chin, A. De Santis, A. L. Ferrara, N. Ho, and S. Kim, "A simple algorithm for the constrained sequence problems," *Information Processing Letters*, Vol. 90, No. 4, pp. 175–179, 2004.

[10] M. Crochemore, C. S. Iliopoulos, Y. J. Pinzon, and J. F. Reid, "A fast and practical bit-vector algorithm for the longest common subsequence problem," *Information Processing Letters*, Vol. 80, No. 6, pp. 279–285, 2001.

[11] H. N. Gabow and R. E. Tarjan, "A linear-time algorithm for a special case of disjoint set union," *Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing*, New York, USA, pp. 246–251, ACM, 1983.

[12] D. S. Hirschberg, "A linear space algorithm for computing maximal common subsequences," *Communications of the ACM*, Vol. 18, No. 6, pp. 341–343, 1975.

[13] D. S. Hirschberg, "Algorithms for the longest common subsequence problem," *Journal of the ACM*, Vol. 24, No. 4, pp. 664–675, 1977.

[14] K.-S. Huang, C.-B. Yang, K.-T. Tseng, H.-Y. Ann, and Y.-H. Peng, "Algorithms for the merged-LCS problem and its variant with block constraint," *Proc. of the 23rd Workshop on Combinatorial Mathematics and Computation Theory*, Chang-Hua, Taiwan, pp. 232–239, 2006.

[15] J. W. Hunt and T. G. Szymanski, "A fast algorithm for computing longest common subsequences," *Communications of the ACM*, Vol. 20, No. 5, pp. 350–353, 1977.

[16] G. M. Landau, E. Myers, and M. Ziv-Ukelson, "Two algorithms for LCS consecutive suffix alignment," *Annual Symposium on Combinatorial Pattern Matching*, Istanbul, Turkey, pp. 173–193, Springer, 2004.

[17] G. M. Landau, E. W. Myers, and J. P. Schmidt, "Incremental string comparison,"

SIAM Journal on Computing, Vol. 27, No. 2, pp. 557–582, 1998.

[18] G. M. Landau, B. Schieber, and M. Ziv-Ukelson, "Sparse LCS common substring alignment," *Annual Symposium on Combinatorial Pattern Matching*, Michoacn, Mexico, pp. 225–236, Springer, 2003.

[19] G. M. Landau and M. Ziv-Ukelson, "On the common substring alignment problem," *Journal of Algorithms*, Vol. 41, No. 2, pp. 338–359, 2001.

[20] M. Lu and H. Lin, "Parallel algorithms for the longest common subsequence problem," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 5, No. 8, pp. 835–848, 1994.

[21] M. Maes, "On a cyclic string-to-string correction problem," *Information Processing Letters*, Vol. 35, No. 2, pp. 73–78, 1990.

[22] S. B. Needleman and C. D. Wunsch, "A general method applicable to the search for similarities in the amino acid sequence of two proteins," *Journal of Molecular Biology*, Vol. 48, No. 3, pp. 443–453, 1970.

[23] Y.-H. Peng, C.-B. Yang, K.-S. Huang, C.-T. Tseng, and C.-Y. Hor, "Efficient sparse dynamic programming for the merged LCS problem with block constraints," *International Journal of Innovative Computing, Information and Control*, Vol. 6, No. 4, pp. 1935–1947, 2010.

[24] Y.-H. Peng, C.-B. Yang, K.-T. Tseng, and K.-S. Huang, "An algorithm and applications to sequence alignment with weighted constraints," *International Journal of Foundations of Computer Science*, Vol. 21, No. 1, pp. 51–59, Feb. 2010.

[25] J. P. Schmidt, "All highest scoring paths in weighted grid graphs and their application to finding all approximate repeats in strings," *SIAM Journal on Computing*, Vol. 27, No. 4, pp. 972–992, 1998.

[26] R. E. Tarjan, "Efficiency of a good but not linear set union algorithm," *Journal of the ACM*, Vol. 22, No. 2, pp. 215–225, 1975.

[27] Y.-T. Tsai, "The constrained longest common subsequence problem," *Information Processing Letters*, Vol. 88, No. 4, pp. 173–176, 2003.

[28] C.-T. Tseng, C.-B. Yang, and H.-Y. Ann, "Efficient algorithms for the longest common subsequence problem with sequential substring constraints," *Journal of Complexity*, Vol. 29, No. 1, pp. 44–52, Feb. 2013.

[29] K.-T. Tseng, D.-S. Chan, and C.-B. Yang, "An efficient merged longest common subsequence algorithm for similar sequences," *Proceedings of the 20th World Multi-Conference on Systemics, Cybernetics and Informatics*, Vol. I, Orlando, Florida, USA, pp. 93–98, July 2016.

[30] R. A. Wagner and M. J. Fischer, "The string-to-string correction problem," *Journal of the ACM*, Vol. 21, No. 1, pp. 168–173, 1974.

Algorithms for Rotating Sector Graphs

Chang Wu Yu
Dept. of Computer Science and Information Engineering
Chung Hua University
james.cwyu@gmail.com

***Abstract*—Wireless sensor networks raise a number of interesting and undiscovered algorithmic issues, but traditional techniques are not sufficient to solve these problems in the right way. This is specifically due to constrained energy and computation capability, nondeterministic sensor failures, channel impairments, node mobility, hostile and distrusted environments, and even external attackers. A sector is obtained by taking a portion of a disk with central angle $\theta \leq \pi$ radians. A sector graph $G=(V, r, \theta)$ consists of equal-sized sectors (with sector degree θ and radius r) placed in a two-dimensional space R^2 and directed edge set $E=\{[i, j] \mid$ sector i contains the center of sector j where i, j are in vertex set $V\}$. Given a sector graph, if it is not connected, we try to rotate some sectors properly so that the resulting sector graph becomes connected by applying the proposed algorithms. We also compute the probability of rotating a disconnected sector graph into a connected one where $\theta \leq$.**

Keywords- Random sector graphs, wireless sensor networks, algorithms.

I. INTRODUCTION

Wireless sensor networks raise a number of interesting and undiscovered algorithmic issues, but traditional graph techniques are not sufficient to solve these problems in the right way. This is specifically due to constrained energy and computation capability, nondeterministic sensor failures, channel impairments, node mobility, hostile and distrusted environments, and even external attackers. In all these issues, wireless sensor networks exhibit substantial vulnerability when compared to other networks. It is challenging to design a robust wireless sensor network by devising novel algorithms or developing new graph theories whilst introducing minimal communication overhead and energy consumption.

However, the algorithmic and theoretical issues in the wireless sensor networks were not fully explored. The main focus of this article is devoted to quick understanding of the algorithms and theories which are developed to build up a robust wireless sensor network.

Applications of random geometric graphs [10] and random graphs [2] include classification, spatial statistics, epidemiology, astrophysics and wireless communications networks with omni-directional antenna [16]. However, random geometric graphs and random graphs cannot be used to represent wireless ad hoc networks equipped with directional antenna properly, which has various advantages over omni-directional antenna [9]. As a result, in this paper, we define a new graph, called *random sector graphs* as follows. A *sector* is obtained by taking a portion of a disk with central angle $\theta \leq \pi$ radians. A *sector graph* $G=(V, r, \theta)$ consists of equal-sized sectors (with sector degree θ and radius r) placed in a two-dimensional space R^2 and directed edge set $E=\{[i, j] \mid$ sector i contains the center of sector j where i, j are in vertex set $V\}$. Let $X_n=\{x_1, x_2, \ldots, x_n\}$ be a set of independently and uniformly distributed random sectors. Here, $\Psi(X_n, r, \theta, A)$ is used to denote the *random sector graph* (RSG) of n sectors on X_n with central angle θ and radius r and placed in an area A. RSGs consider sector graphs on random sector configurations.

An RSG $\Psi(X_n, r, \theta, A)$ is appropriate for modeling an wireless ad hoc sensor network consisting of n mobile devices with a directional antenna that are independently and uniformly distributed randomly in an area A. The transmission radius of each equipped antenna is r unit length, and its coverage is often limited and can be modeled by a circular sector in the plane. Specifically, an RSG $\Psi(X_n, r, \theta, A)$ consists of n same size circular sectors with sector degree θ, which uniformly randomly distribute in space A. If a circular sector i covers the center of another circular sector j, there exists an arc (directed edge) $[i, j]$, which also indicates transceiver i can communicate directly with transceiver j. Figure 1 is such network with its associated RSG.

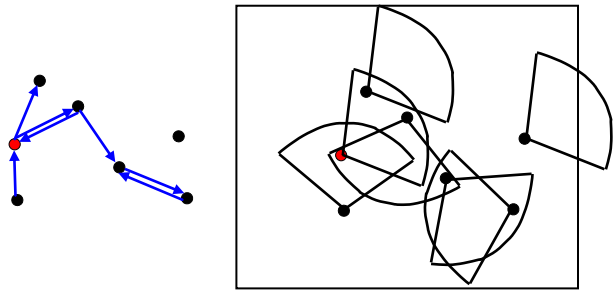


Figure 1. A wireless ad hoc network (equipped with directional antenna) and its associated RSG

Figure 1 displays an RSG and its representing network. In the example, area A is a rectangle used to model the deployed area such as a meeting room. Area A, however, can be a circle, or any other shape, and even infinite space. Note that torus convention is adopted here to remove border effects such that the deployed area appears to be homogeneous at any point [1], [8].

According to the above definitions, we have that RSGs are a natural generalization of random geometric graphs. That is, a random geometric graph is a special case of a RSG when θ =360. Also random scaled sector graphs are a superset of RSGs.

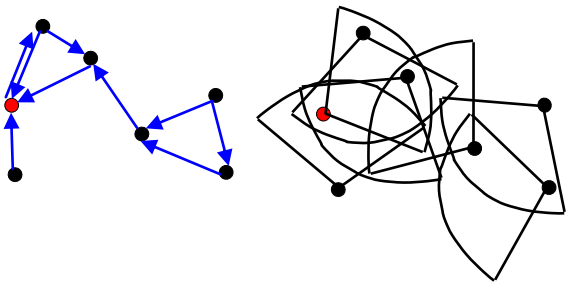


Figure 2. A wireless ad hoc network (equipped with directional antenna) and its associated RSG after proper rotations on some specific sectors.

Usually, we need a sink device to collect all detected useful information in a given wireless sensor network. That implies there exists a directed tree with a root for the sink in the corresponding sector graph. However, given an arbitrary sector graph, it is not necessary to be connected. Similarly, the probability of a random sector graph to be connected is not always high.

Fortunately, suppose that each sensor is equipped with a rotation mechanism, we are able to adjust the direction of the associated antenna for selected sensors (sectors) to proper position. In this work, we find that the resulting sensor network may contain a desired routing tree with a high probability.

When a given sector graph is not connected, this work aims to design an algorithm to rotate some sectors properly so that the resulting sector graph becomes connected. We also compute the probability of rotating a disconnected sector graph into a connected one. To the best of our knowledge, no previous work mentioned similar related results.

The rest of the paper is organized as follows. Section II introduces definitions and notations. Section III then briefly surveys pertinent literature. Next, Section IV analyzes the probability of a connected random sector graph, followed with an algorithm for rotating disconnected sector graphs into connected one in Section V. Section VI concludes this work.

II. Definitions and Notations

The *subgraph probability* of a labeled subgraph $G=(V, E)$ in $\Psi(X_n, r, \theta, A)$ is defined formally as follows. Let $\Omega=\{G_1, G_2, \ldots, G_w\}$ represent every possible labeled simple graphs of $\Psi(X_n, r, \theta, A)$, where $X_n=\{x_1, x_2, \ldots, x_n\}$ and $w=2^{\binom{n}{2}}$. For each labeled graph $G_k=(V_k, E_k)$ with $V_k=\{1, 2, \ldots, n\}$ in Ω, we have $E_k \subseteq V_k \times V_k$, where $[i, j] \in E_k$ and $1 \le k \le 2^{\binom{n}{2}}$. Given a subgraph $G_x=(V_x, E_x)$ where $V_x \subseteq \{1, 2, \ldots, n\}$ and $E_x \subseteq V_x \times V_x$, the *subgraph probability* of G_x in $\Psi(X_n, r, A)$, denoted by $\Pr(G_x)$, is summing up the probabilities of all label graphs in Ω whose induced subgraphs by V_x are identical to G_x. Specifically, we have $\Pr(G_x)=\sum_{\forall G \in \Omega \text{ and } G_{V_x}=G_x} \Pr(G)$.

III. Related Work in RSGs

We summary related results as follows. A book and several papers written by Penrose [10]-[13] provide and explain the theory of random geometric graphs (RGGs). Graph problems considered in the book include subgraph and component counts, vertex degrees, cliques and colorings, minimum degree, the largest component, partitioning problems, and connectivity and the number of components.

For n points uniformly randomly distributed on a unit cube in $d \ge 2$ dimensions, Penrose [13] showed that the resulting geometric random graph G is k-connected and G has minimum degree k at the same time when $n \to \infty$. In [3], [4], Díaz et al. discussed many layout problems including minimum linear arrangement, cutwidth, sum cut, vertex separation, edge bisection, and vertex bisection in random geometric graphs. In [5], Díaz et al. considered the clique or chromatic number of random geometric graphs and their connectivity.

Some results of RGGs can be applied to the connectivity problem of ad hoc networks. In [14], Santi and Blough discussed the connectivity problem of random geometric graphs $\Psi(X_n, r, A)$, where A is a d-dimensional region with the same length size. In [1], Bettstetter investigated two fundamental characteristics of wireless networks: its minimum node degree and its k-connectivity. In [6], Dousse et al. obtained analytical expressions of the probability of connectivity in the one dimension case.

In [7], Gupta and Kumar have shown that if $r=\sqrt{\frac{\log n + c(n)}{\pi n}}$, then the resulting network is connected with high probability if and only if $c(n) \to \infty$. In [17], Xue and Kumar have shown that each node should be connected to $\Theta(\log n)$ nearest neighbors in order that the overall network is connected.

Yen and Yu have analyzed link probability, expected node degree, and expected coverage of MANETs [19]. In [18], Yang has obtained the limits of the number of subgraphs of a specified type which appear in a random graph. In [20], Yu has proposed the first paradigm for exactly computing subgraph probability of RGGs.

IV. Computing the Probability of a Random Sector Graph

In the section, at first, a novel paradigm for exactly computing subgraph probability of RSGs with sector degree less than 60 is proposed. For simplicity, we always assume that A is sufficiently large to properly contain a circle with radius r in a $\Psi(X_n, r, \theta, A)$ throughout the paper. By applying this paradigm, we maybe have a chance to estimate the probability of connectivity of given a random sector graph.

First, a graph drawing convention used in [20], which is helpful for describing the proposed paradigm, is given. A *solid line* denotes an arc of G; a *broken line* denotes a possible arc between them; two vertices without a line denote a *non-edge* of G. A *class graph* $G=(V, A_S, A_B)$ consists of a vertex set V and two disjoint arc sets E_S and E_B, where E_S (E_B) denotes a set of solid-line arcs (broken-line arcs) joining two vertices of V. A *complete class graph* is a class graph whose vertices are pair-wise adjacent with two arc one of which is either a solid line or a broken line. Here we introduce two additional graphs $\alpha(G)$ and $\beta(G)$ from any

class graph $G = (V, E_S, E_B)$ such that $\alpha(G) = (V, E_S)$ and $\beta(G) = (V, E_S \cup E_B)$.

Some operators and notation of class graphs used in this work are defined similarly to that in [20]. The *union* of two class graphs G_a and G_b, denoted G_a+G_b, is the set whose elements are exactly the graphs in either G_a or G_b. The *difference* of two class graphs G_a and G_b, denoted $G_a–G_b$, is the set containing exactly those elements in G_a that are not in G_b. When G is a class graph, Pr(G) denotes the probability of the occurrence of $G_x \in G$ in $\Psi(X_n, r, \theta, A)$. If every element in G_a is also in G_b, we have $G_a \subseteq G_b$. Evidently, if $G_a \subseteq G_b$ then $\Pr(G_a) \le \Pr(G_b)$, and if G_a is isomorphic to G_b then $\Pr(G_a)=\Pr(G_b)$. The union and difference of class graphs can be represented by the graph drawing convention in Figure 2.

Note that the class graphs discussed here are directed graphs, which are different from that in [20]. Actually, the class graphs with their graph drawings are more complicated than that in [20].

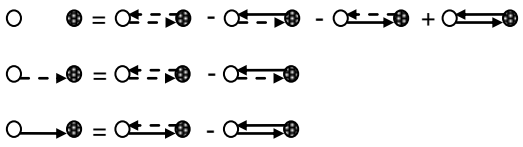
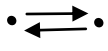
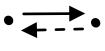
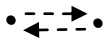
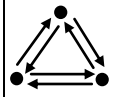


Figure 3. Graph drawing conventions and derivations used in RSGs

The probabilities of the following subgraphs in a $\Psi(X_n, r, \theta, A)$ can be computed by manipulating elementary geometric techniques. Due to the page limit of this paper, all derivations of the following results are skipped here, and Table 1 summaries the results briefly.

Table 1. List of derived probabilities of $\Psi(X_n, r, \theta \le 60, A)$

Notation	G_1	G_2	G_3
G			
$Pr(G)$	$\frac{\theta^2 r^2}{4\pi\lvert A\rvert}$	$\frac{\theta r^2}{2\lvert A\rvert}$	1

Notation	G_4	G_5	G_6	G_7
G				
$Pr(G)$	0	$\frac{r^4\theta^3\int_0^{\theta}\int_0^{\theta}\frac{\sin(\theta_a)*\sin(\theta_b)}{\sin(\pi-\theta_a-\theta_b)}d\theta_b d\theta_a}{16\lvert A\rvert^2\pi^2}$	$\frac{r^4\theta^4}{16\pi^2\lvert A\rvert^2}$	$\frac{r^4\theta^4}{16\lvert A\rvert^2\pi^2}$

Notation	G_8	G_9	G_{10}	G_{11}
G				
$Pr(G)$	$\frac{r^4\theta^2\int_0^{\theta}\int_0^{\theta}\frac{\sin(\theta_a)*\sin(\theta_b)}{\sin(\pi-\theta_a-\theta_b)}d\theta_b d\theta_a}{8\lvert A\rvert^2\pi}$	$\frac{r^4\theta^4}{16\lvert A\rvert^2\pi^2}$	$\frac{r^4\theta^3}{8\pi\lvert A\rvert^2}$	$\frac{r^4\theta^3}{8\pi\lvert A\rvert^2}$

Notation	G_{12}	G_{13}	G_{14}	G_{15}
G				
$Pr(G)$	$\frac{\theta^2 r^2}{4\pi\lvert A\rvert}$	$\frac{r^4\theta^3}{24\lvert A\rvert^2\pi}$	$\frac{r^4\theta^3}{8\lvert A\rvert^2}$	$\frac{r^4\theta^2}{4\lvert A\rvert^2}$

Notation	G_{16}	G_{17}	G_{18}	G_{19}
G				
$Pr(G)$	$\frac{r^4\theta^2}{4\lvert A\rvert^2}$	$\frac{r^4\theta^2}{4\lvert A\rvert^2}$	$\frac{r^2\theta}{2\lvert A\rvert}$	1

Given a subgraph $G=(V, E)$, the paradigm, similar to that shown in [20], computes its probability Pr(G) of a RSG by exploiting the following three steps:

(1) *Preprocessing step*: First, generate all complete class graph set CG_n with the same labeled vertex set where $n=|V|$. Find out all the equivalence sets such that the underline graph of each class graph is isomorphic to each other in same set. Next, compute the subgraph probability Pr(x) for each equivalence set x. Moreover, one element from each distinct equivalence set is selected to form a *basis* $\{G_1, G_2, \ldots, G_k\}$ of CG_n. For example, for $|V|=2$, $\{G_1, G_2, G_3\}$ in Table 1 forms a basis of CG_2; moreover, for $|V|=3$, $\{G_4, G_5, \ldots, G_{19}\}$ in Table 1 forms a basis of CG_3.

(2) *Decomposing step*: Decompose G into a linear combination of the selected *basis* of CG_n: $c_1G_1+c_2G_2+\ldots+c_kG_k$ by repeatedly applying the graph derivations in Figure 2 for each pair of vertices in class graph G. An example of the graph decomposition is shown in Figure 3. Note that the graph is decomposed into a linear combination of $\{G_4, G_5, G_6, G_7, G_8, G_{10}\}$.

(3) *Manipulating step*: Compute $\Pr(G) = c_1\Pr(G_1) + c_2\Pr(G_2) +\ldots+ c_k\Pr(G_k)$, where Pr($G_i$) is obtained in the preprocessing step, for $1 \le i \le k$. Accordingly, the probability of the derived graph in Figure 4 is equal to $\Pr(G_{10})- \Pr(G_6)-\Pr(G_8)+3\Pr(G_5)-\Pr(G_7)-\Pr(G_4)$.

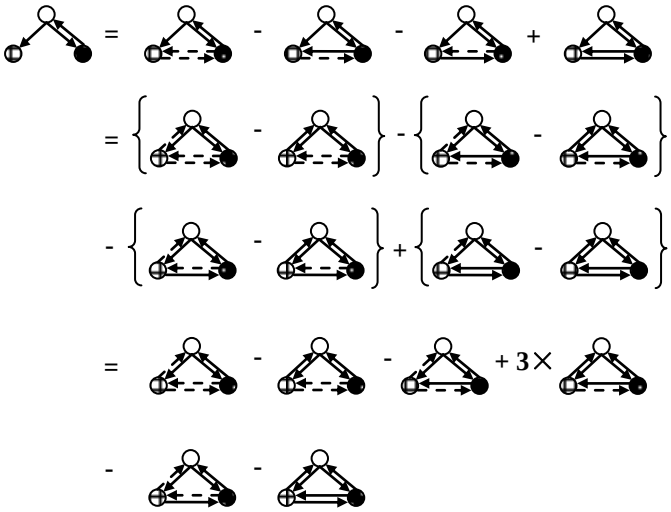


Figure 4. Decompose a graph G into a linear combination of the selected basis.

For example, G_{15} is a complete class graph and its $\alpha(G)$ is a directed tree and $\beta(G)$ is a complete graph. Please note that for this kind of graphs, its probabilities (shown below) can be easily computed by applying the similar method shown in [20].

Theorem 1: Given a complete class graph $G = (V, E_S, E_B)$, if $\alpha(G)$ is a directed tree then $\Pr(G) = (\theta r^2/2|A|)^{|V|-1}$.

V. Algorithms on Random Sector Graphs

In this section, we design an algorithm so that we can follow to select and to rotate the directions of antennas of some sensors to obtain a connected sensor networks.
The proposed rotating algorithm is described as follows:

Step 1: Construct a geometry graph $G=(V, E)$ by replacing each sector with a circle with the same radius in the given sector graph.
Step 2: Assign each edge e in E of G with the value of cost to rotate the associated antenna.
Step 3: Find out the minimum cost spanning tree T in the weighted graph G.
Step 4: Rotate every selected sensor according to the resulting tree created in Step 3.
Step 5: Construct a routing aggregation tree for the given wireless sensor network.

After following the above algorithm, we can show that with high probability the resulting graph is connected. Moreover, the time complexity of the above algorithm can be easily obtained.

For a given vertex as a sink in a wireless sensor network, if we can count the number of different spanning trees in the network, with the help of Theorem 1, we may sum up their probabilities and obtain the probability of a random sector graph with a directed spanning tree. Please also note that all the graphs mentioned in this paper are labeled graphs.

Moreover, we also can compute the probability of the corresponding random geometric graph created by Step 1 in the proposed rotation algorithm by applying the similar results in [20]. Finally, we obtain the probability of the resulting sector graph being connected improved by applying the proposed rotation algorithm.

VI. Conclusions

Given a sector graph, if it is not connected, in this work, we have designed an algorithm to rotate some sectors properly so that the resulting sector graph becomes connected by applying the proposed algorithms. We also computed the probability of rotating a disconnected sector graph into a connected one. More applications in on topology of ad hoc networks with directional antenna are also a challenging work in the future.

.

References

[1] Christian Bettstetter, "On the minimum node degree and connectivity of a wireless multi-hop network," *MobiHoc*, 2002, pp. 80–91.

[2] B. Bollobas, *Random Graphs*, London: Academic Press, 1985.

[3] J. Díaz, M. D. Penrose, J. Petit, and M. Serna, "Convergence theorems for some layout measures on random lattice and random geometric graphs," *Combinatorics, Probability, and Computing*, no. 6, pp. 489–511, 2000.

[4] J. Díaz, M. D. Penrose, J. Petit, and M. Serna, "Approximating layout problems on random geometric graphs," *J. Algorithms*, vol. 39, pp. 78–116, 2001.

[5] J. Díaz, J. Petit, and M. Serna, "Random geometric problems on $[0, 1]^2$," *Lecture Notes in Computer Science*, vol. 1518, Springer-Verlag, New York/Berlin, 1998.

[6] Dousse, P. Thiran, and M. Hasler, "Connectivity in ad-hoc and hybrid networks," *INFOCOM*, 2002.

[7] P. Gupta and P. R. Kumar, "Critical power for asymptotic connectivity in wireless networks," *Stochastic Analysis, Control, Optimization and Applications*, pp. 547–566, 1998.

[8] Peter Hall, *Introduction to the Theory of Coverage Process*, John Wiley and Sons, New York, 1988.

[9] Martin Horneffer and Dieter Plassmann, "Directed antennas in the mobile broadband system," *INFOCOM*, 1996, pp. 704–712.

[10] Mathew D. Penrose, *Random Geometric Graphs*, Oxford University Press, 2003.

[11] M. D. Penrose, "A strong law for the longest edge of the minimal spanning tree," *The Annals of Probability*, vol. 27, no. 1, pp. 246–260,1999.

[12] M. D. Penrose, "The longest edge of the random minimal spanningtree," *The Annals of Applied Probability*, vol. 7, no. 2, pp. 340–361,1997.

[13] M. D. Penrose, "On k-connectivity for a geometric random graph," *Random structures and Algorithms*, vol. 15, no. 2, pp. 145–164, 1999.

[14] Paolo Santi and Douglas M. Blough, "The critical transmitting range for connectivity in sparse wireless ad hoc networks," *IEEE Trans. Mobile Comput.*, vol. 2, no. 1, pp. 25–39, 2003.

[15] D. Shukla, L. Chandran-Wadia, and S. Iyer, "Mitigating the exposed node problem in IEEE 802.11 ad hoc networks,"

International Conference on Computer Communications and Networks, 2003, pp. 157–162.

[16] F. Tobagi and L. Kleinrock, "Packet switching in radio channels, Part II-The hidden terminal problem in carrier sense multiple access and the busy tone solution," *IEEE Trans. Commun.*, vol. COM-23, no. 12, pp. 1417–1433, 1975.

[17] F. Xue and P. R. Kumar, "The number of neighbors needed for connectivity of wireless networks," *Wireless Networks*, vol. 10, pp. 169–181, 2004.

[18] K. J. Yang, *On the Number of Subgraphs of a Random Graph in [0, 1]*^{d},Unpublished D. Phil. thesis, Department of Statistics and Actuarial Science, University of Iowa, 1995.

[19] L.-H. Yen and Chang Wu Yu, "Link probability, network coverage, and related properties of wireless ad hoc networks," *The 1st IEEE International Conference on Mobile Ad-hoc and Sensor Systems*, 2004, pp. 525–527.

[20] Chang Wu Yu, "Computing Subgraph Probability of Random Geometric Graphs with Applications in Quantitative Analysis of Ad Hoc Networks," *IEEE Journal on Selected Areas in Communications*, vol. 27, no. 7, pp.1056–1065, 2009.

The Integer Domination Number of Circulant Graphs

Xuan-Yun Lin and Hung-Lin Fu
Department of Applied Mathematics
Chiao Tung University
Hsin-Chu 300, Taiwan

Kuo-Ching Huang*
Department of Financial and Computational Mathematics
Providence University
Taichung 43301, Taiwan

Abstract

Let $G = (V, E)$ be a graph. For integer $k \geq 1$, a function $f : V \to \{0, 1, 2, 3, \ldots\}$ is a $\{k\}$-dominating function of G if for every vertex $v \in V$, $f(v) + \sum_{uv \in E} f(u) \geq k$. The weight of f is $\sum_{v \in E} f(v)$. The $\{k\}$-domination number, denoted by $\gamma_{\{k\}}(G)$, of G is the minimum weight of a $\{k\}$-dominating function. Clearly, when $k = 1$, a $\{k\}$-dominating problem is exactly the dominating problem, that is, $\gamma_{\{1\}}(G)$=$\gamma(G)$.

A circulant graph $C(n; a_1, a_2, \ldots, a_m)$ is a simple graph with the vertex set $V = \{v_i = i \mid i = 0, 1, 2, \ldots, n-1\}$ and the edge set $E = \{v_i v_j \mid i - j \equiv a_t \pmod{n}, t = 1, 2, \ldots, m\}$. In this talk, we completely determine the $\{k\}$-domination number of circulant graphs $C(2n; 1, n)$, $C(n; 1, 2)$ and $C(n; 1, 2, 3)$.

Keywords: *domination number, $\{k\}$-domination number, circulant graph*

1 Introduction

Nowadays, almost everyone has a smart phone and can use the internet through Wi-Fi. In order to make the Wi-Fi service more efficient and save more budgets, the problems arise naturely: How to use the least Wi-Fi stations and how to arrange the Wi-Fi stations to serve as many people as possible. The problem of Wi-Fi stations can be solved by considerig the $\{k\}$-dominating problem. We trasform each area to a vertex and its neighboring areas, adjacent vertices, are connected by edge. Building some Wi-Fi stations in each area such that the total number of Wi-Fi stations in each area and its neighboring areas is at least k. The $\{k\}$-dominating number is the minimum number of the total Wi-Fi stations.

Let $G = (V, E)$ be a graph. For integer $k \geq 1$, a function $f : V \to \{0, 1, 2, 3, \ldots\}$ is a $\{k\}$-dominating function of G if for every vertex $v \in V$, $f(v) + \sum_{uv \in E} f(u) \geq k$.

The weight of f is $\sum_{v\in E} f(v)$. The $\{k\}$-domination number, denoted by $\gamma_{\{k\}}(G)$, of G is the minimum weight of a $\{k\}$-dominating function. The concept of $\{k\}$-dominating problem was introduced by Domke, Hedetniemi, Laskar and Fricke [2]. In 1998, Haynes, Hedetniemi and Slater [3] gave an upper bound on the $\{k\}$-domination number. For Cartesian product of graphs, Bresar, Henning and Klavzar [1] gave the upper bound in 2006 and Hou and Lu [4] gave the lower bound in terms of packing in 2009. Kuan [5] gave an lower bound and obtained the exact values of the complete graph K_n, the star S_n, the path P_n and $P_2\Box P_n$ on the $\{k\}$-dominating number.

A circulant graph $C(n; a_1, a_2, \ldots, a_m)$ is a graph with the vertex set $V = \{v_i = i \mid i = 0, 1, 2, \ldots, n-1\}$ and the edge set $E = \{v_iv_j \mid i - j \equiv a_t \pmod n, t = 1, 2, \ldots, m\}$. In this talk, we completely determine the $\{k\}$-domination number of circulant graphs $C(2n; 1, n)$, $C(n; 1, 2)$ and $C(n; 1, 2, 3)$.

2 The $\{k\}$-Domination Number $\gamma_{\{k\}}(C(2n, 1, n))$

In [5], Kuan obtained the lower bound of graphs on the $\{k\}$-domination number.

Lemma 1 *For any graph G, $\gamma_{\{k\}}(G) \geq \lceil \frac{k|G|}{\Delta+1} \rceil$.*

Since the circulant graph $C(2n; 1, n)$ is 3-regular, $\gamma_{\{k\}}(C(2n, 1, n)) \geq \lceil \frac{2nk}{3+1} \rceil = \lceil \frac{nk}{2} \rceil$. The following lemma is essential for finding the upper bounds for $C(2n; 1, n)$.

Lemma 2 *Let $S = \{0, i(n-2), i(i+2) \mid 1 \leq i \leq \frac{n-r}{4}\}$, where $r \equiv n \pmod 4$. Then the followings hold.*

1. $|S| = 1 + 2 \cdot \frac{n-r}{4} = \frac{n+2-r}{2}$.
2. *If $n \equiv 0 \pmod 4$, then S is a domination set.*
3. *If $n \equiv 1 \pmod 4$, then S is an independent domination set.*
4. *If $n \equiv 2 \pmod 4$, then S is a packing and $\bigcup_{v\in S} N[v] = V(C(2n; 1, n))$.*
5. *If $n \equiv 3 \pmod 4$, then S is a packing such that*

$$T \setminus \bigcup_{v\in S} N[v] = \{\frac{n-3}{4}(n-2) + n - 1, \frac{n-3}{4}(n+2) + n + 1\}$$

and $S \cup T$ is a dominating set.

In what follows, we determine $\gamma_{\{k\}}(C(2n,1,n))$.

Lemma 3 *Suppose* $n \equiv 0 \pmod 4$. *Then*

1. $\gamma_{\{k\}}(C(2n,1,n)) = \frac{nk}{2}$ *if* $k \equiv 0 \pmod 2$.
2. $\gamma_{\{k\}}(C(2n,1,n)) = \frac{nk}{2} + 1$ *if* $k \equiv 1 \pmod 2$.

Lemma 4 *Suppose* $n \equiv 1 \pmod 4$. *Then*

1. $\gamma_{\{k\}}(C(2n,1,n)) = \lceil \frac{nk}{2} \rceil$ *if* $k \equiv 0,1,3 \pmod 4$.
2. $\gamma_{\{k\}}(C(2n,1,n)) = \frac{nk}{2} + 1$ *if* $k \equiv 2 \pmod 4$.

Lemma 5 *Suppose* $n \equiv 2 \pmod 4$. *Then* $\gamma_{\{k\}}(C(2n,1,n)) = \frac{nk}{2}$.

Lemma 6 *Suppose* $n \equiv 3 \pmod 4$. *Then*

1. $\gamma_{\{k\}}(C(2n,1,n)) = \lceil \frac{nk}{2} \rceil$ *if* $k \equiv 0,1,3 \pmod 4$.
2. $\gamma_{\{k\}}(C(2n,1,n)) = \frac{nk}{2} + 1$ *if* $k \equiv 2 \pmod 4$.

3 The $\{k\}$-Domination Numbers $\gamma_{\{k\}}(C(2n,1,2))$ and $\gamma_{\{k\}}(C(2n,1,2,3))$

We also completely determine $\gamma_{\{k\}}(C(2n,1,2))$ and $\gamma_{\{k\}}(C(2n,1,2,3))$.

Lemma 7 $\gamma_{\{k\}}(C(2n,1,2)) = \lceil \frac{nk}{5} \rceil$.

Lemma 8 $\gamma_{\{k\}}(C(2n,1,2,3)) = \lceil \frac{nk}{7} \rceil$.

References

[1] B. Bresar, M. A. Henning and S. Klavzar, On integer domination in graphs and Vizing-like problem. Taiwan J. Math. 10(2006), 1317-1328.

[2] G. Domke, S. T. Hedetniemi, R. C. Laskar and G. Fricke, Relationships between integer and fractional parameters of graphs, in: Graph Theory, Combinatorics and Applications, Vol. 2, John Wiley & Sons, Inc.(1991), 371-287.

[3] T. W. Haynes, S. T. Hedetniemi and P. J. Slater, Domination in Graphs: Advanced Topics, Marcel Deliker, New York (1998).

[4] X. Hou and Y. Lu, On the $\{k\}$-dominating number of Cartesian products of graphs. Discrete Math. 309(2009), 3413-3419.

[5] Y. T. Kuan, A study of integer domination number. Master thesis, National Chiao Tung University, 2017.

BigBigTree: reconstruct the phylogenetic trees of large orthologous sequences

Han-Lung Tsai, Chih-Chuan Chang, Jia-Ming Chang

Department of Computer Science

National Chengchi University, 11605 Taipei City, Taiwan

{103703019, 103703003, jmchang}@nccu.edu.tw

Abstract

A phylogenetic tree is a branching diagram base on the similarities of creatures in morphology, structure, physiology, genetics, ecology and genetic sequence. It shows a inferred evolutionary history among species. To deal with the massive genetic data, we propose a faster and more accurate approach, BigBigTree, reconstructing phylogenetic trees by divide and concatenate. The source code and docker of BigBigTree are available in https://github.com/jmchanglab/bigbigtree and changlabtw/bigbigtree, where the later allows users one-click installation.

1 Introduction

1.1 Research motivation

Phylogenetic tree, a categorization facility that classifies creatures by their genetic similarities, can facilely find the common ancestors of every species, such as Figure 1, each node represents the nearest common ancestor of each branch, and the edge lengths in phylogenetic trees may be interpreted as time estimates.

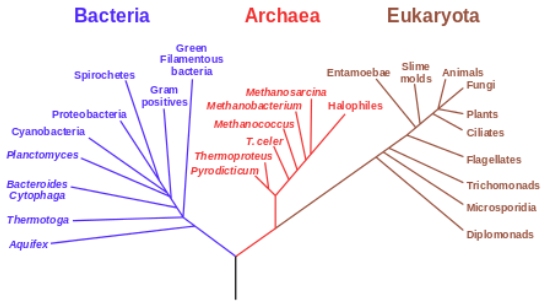


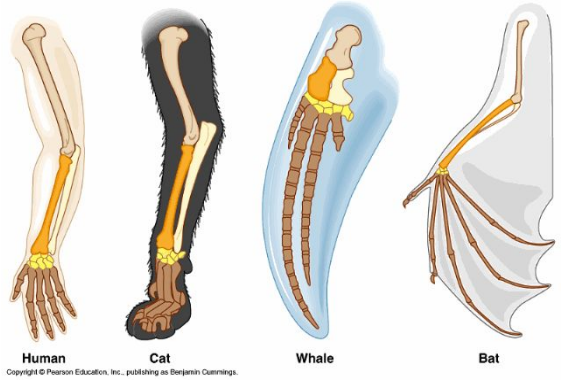
Figure 1：Phylogenetic Tree of Life (Extracted from [1]).

Building phylogenetic tree needs to confirm relevance between species, since the evolution is a extremely time-consuming process that we can not confirm it through observation or experiment directly, instead, we only proof it by collateral evidences, that means all of phylogenetic trees are hypotheses, building phylogenetic trees by different models and methods may produce various results. The major of biologists use two types of methods as the basis to confirm the similarities of species.

First is morphology, the characterizations of species fall into homology and analogy. homology means creatures have resemble body structures, but evolve into different appearances and abilities that depend on their living environment (Figure 2.a); Analogy, exactly the opposite, having similar appearances or abilities but evolve from different body structures , which means analogy may not have genetic relationship, only the result of convergent evolution (Figure 2.b), therefore, if we want to use the similar characterizations between fossils and living creatures to confirm their relevance, we must

base on the homology to ensure the close and distant relationships of species.

(a)



(b)

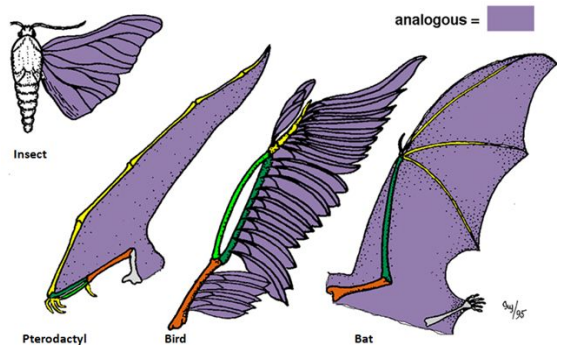


Figure 2：(a) Homology and (b) Analogy (Extracted from [2]).

The other one is molecular biology, using DNA sequenceing which confirms relevance between creatures by the order of four bases—adenine, guanine, cytosine, and thymine—in a strand of DNA.

A phylogenetic tree not only shows the close and distant relationships of species but also can handily classify each genotype and find the needed parts that can use in genetic modification and identification.

There are many ways building a phylogenetic tree, but most of them could build trees in a short time only when the input genetic data is small, when face to huge amounts of data, how to optimize the process of building a phylogenetic tree became an issue. To build a phylogenetic tree accurately and rapidly,basically we have two popular types of methods: Maximum Parsimony (MP) and Maximum Likelihood (ML).

MP is using the degree of variation between genetic sequences to confirm the close and distant relationships of species, that is, building a phylogenetic tree by changing the least in the evolution, this way may misclassify different creatures into similar species cause by the convergent evolution, therefore, this way usually uses in the situation that the relationships of species are close; MP is the way that using statistical data to estimate the stochastic model.

Thanks to the development of Next-Generation Sequencing, we can get lots of genetic sequences easily, means that it is capable to build a phylogenetic tree by genetic sequences. Compare to MP that is easy to understand but has more deviation, ML is more accurate but needs enormous calculations. Our research hopes to combine the advantages of MP and ML, using Divide and Concatenate algorithm which clusters genetic sequences by their homology, disposing them separately and then merge together, this algorithm not only keeps the complicated genetic informations completely but also can build a phylogenetic tree accurately and quickly.

1.2 Related works

S. Mirarab *et. al.* presented that although there are several multispecies coalescent models, but all of them have disadvantages, BUCKy-pop has high time complexity even if it can use in unrooted tree; BEST and *BEAST can build gene trees and species trees by sequence alignment simultaneously, but when data is big enough will lead those methods incalculable [3]. Another thesis also presented conflicts may occur if we use different allele to build different phylogenetic trees, to correct these conflicts we need to align several allele to increase the accuracy of phylogenetic trees, which make the data that it need more enormous [4]. Thus, the problem in front of the development of genetic technique is to reduce time complexity, Accurate Species TRee ALgorithm (ASTRAL) in the thesis is a way limit searching space by abandoning the less grade side to make time complexity in a polynomial time; in the other way, we reduce time complexity by dividing and merging data.

P. Vachaspati *et. al.* presented that many methods use ILS (Incomplete Lineage Sorting) in

building species trees, but only ASTRAL-2 and NJst can remain accuracy in a large data level, so he redesign NJst to improve the compatibility of data and can combine with different distance-based tree estimative methods, called ASTRID, has similar accuracy and even better efficiency than ASTRAL-2 [5]. The thesis also mentioned that INternode Distances is important in building phylogenetic trees, calculate the period of time in the evolution process to analyze relationship, and make the correspondent time be the proportion of the distance between nodes in phylogenetic trees, which can make the results more accurate.

2 Methods

2.1 Our Idea

In the course of meiosis or RNA replication, it will occur a phenomenon called *gene duplication*, a duplication of the gene region. Gene duplications are an essential source of genetic novelty that can lead to evolutionary innovation. Duplication creates genetic redundancy, where the extra copy of the gene is often free from selective pressure, that is, if one copy of the gene experiences a mutation affecting its original function, other copy can serve as a 'spare part' and continue to function correctly. A mutation will have no deleterious effects to its host organism. Thus, duplicate genes accumulate mutations faster than a functional single-copy gene, that is recognized as an evolutionary manifestation.

According to the Figure 3, when the genome region of an ancient species occur gene duplication, it produces α-gene and β-gene in descendant species. Afterward with speciation event, the species gradually evolve to three species frag, chick and mouse. The α-genes of three species are ‘orthologous genes’. The α-gene and β-gene of the same specie are ‘paralogous genes’.

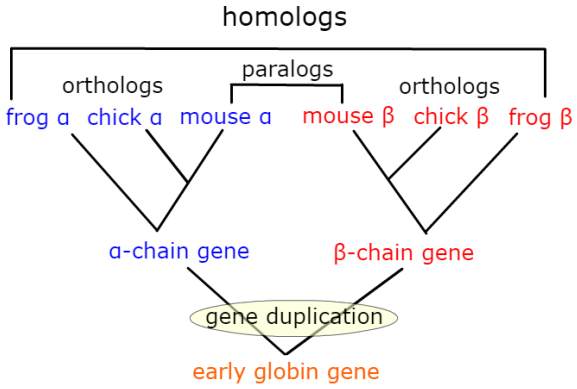


Figure 3：Difference between orthology and paralogy (Extracted from [6]).

Orthologous genes are more similar than paralogous genes because the later had mutated before speciation compared to the former had mutated after speciation. So, the orthologous genes between different species are more similar than paralogous genes of the same species. Our approach mainly takes advantage of this feature. First, sequences are clustered into groups (i.e., orthologous genes) which are used to build individual trees. Then, the hierarchical clustering of the those groups is determined by the concatenated orthologous alignment. The final phylogenetic tree is constructed by merging individual orthologous tree with the hierarchical clustering.

2.2 Algorithm Design

The algorithm is divided into six steps: 1) input data, 2) use BLAST to compare sequences; 3) cluster by hcluster, 4) align each cluster, 5) and then concatenate each orthologs among species alignment into on single string, 6) finally build a mother tree base on the result of previous step and merge subtrees of each clusters and the mother tree together. The overall flowchart is shown in Figure 4. The detail of each step is explained as the following:

Step1：Input

Sequences with species annotation in FASTA format. For example, there are m species and n orthology gene families, in total, $m*n$ sequences.

Step2：Sequence comparison

We compare *m***n* sequences by Basic Local Alignment Search Tool (BLAST). BLAST is a program that have been used widely to align primary structure of biological sequences in analysing bioinformatics, which can let researchers find target sequences or similar ones, using heuristic algorithm to search and have quite speed and accuracy [7].

Step3：DIVIDE - Cluster ortholog

According to the result of BLAST, we get similarity between sequences. Then, we apply a tool, hcluster, to cluster sequences into cross-species orthologues and transfer them into FASTA format [8]. FASTA is a text format used in recording nucleic acid or peptide sequences, any nucleic acid and amino acid present as single alphabet code so that we can easily analyze sequences with scripting language such as Python, Ruby, and Perl.

Step4：Cluster alignment

Instead of generating a big alignment of all sequences, we generate alignment for each cluster - ortholog cluster ($cluster_1$, $culster_2$, …, $cluster_n$) and species cluster (spe_1, spe_2, …, spe_m).

Step5：CONCATENATE - Orthology concatenation

We concatenate each orthologs among species alignment (spe_1, spe_2, …, spe_m) into one single string. Therefore, we will have the sequence alignment of paralogous strings which not only provides more information for phylogenetic reconstruction but also reduces time complexity in building evolutionary tree.

Step6：Build tree & Merge tree

Users can choose either TreeBeST [9] or PhyML [10] to build phylogenetic trees of orthologous sequences alignment (i.e., $cluster_n$) and the concatenate long-string alignment. A final phylogenetic tree will be constructed by replacing the leaf node of the concatenate tree with corresponding orthologous trees. Compared with a traditional way, building tree directly based on all sequences, Divide and Concatenate approach utilizes more information of the concatenated strings to complete phylogenetic tree accurately with reduced time complexity.

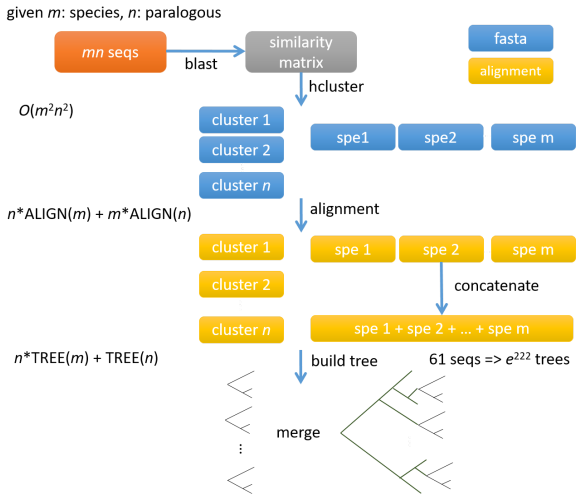


Figure 4：The flow chart of our algorithm.

2.3 Implementation

We reimplement whole pipeline by Nextflow [11] instead of Python [12]. Nextflow simplifies the implementation and the deployment of complex parallel and reactive workflows, which is used in our project to make the pipeline more quickly and more efficiently. It is possible to execute locally by cloning the repository from github or downloading docker container. Both are available in https://github.com/jmchanglab/bigbigtree and changlabtw/bigbigtree, respectively (Figure 5). We will reconstruct a new web service for biologists without installation.

```
N E X T F L O W  ~  version 0.25.5
Launching `bigbigtree.nf` [big_swartz] - revision: 1cda8
R N A T O Y   P I P E L I N E
============================
aa: example/Or_aa.fasta
speciesTree : /home/cs/Bigbigtree-final/bigbigtree-maste
nn : example/Or_nn_v2.fasta
cluster_dir: res_dic/cluster


[warm up] executor > local
[37/e3678a] Submitted process > step1_1cluster
[41/41c43a] Submitted process > step0_check_fasta_diff
[28/dca162] Submitted process > step1_2dif

[19/1975e1] Submitted process > step2_cluster_to_fasta_n
[04/08a2ac] Submitted process > step2_cluster_to_fasta_a
[fb/9a3eb8] Submitted process > step3_1_alignment_aa (1)
[c1/6774e5] Submitted process > step3_1_alignment_aa (2)
[38/08a2e3] Submitted process > step3_1_alignment_aa (3)
[66/f41729] Submitted process > step3_1_alignment_aa (4)
[72/f96f65] Submitted process > step3_1_alignment_aa (5)
[38/c70277] Submitted process > step3_1_alignment_aa (6)
[9b/6eb054] Submitted process > step3_1_alignment_aa (8)
```

Figure 5：The snapshot of execution of BigBiogTree by Nextflow.

2.4 Evaluation

2.4.1 Real biology data

For testing ability of BigBigTree, we collected data sets with two evolutionary scenarios, which are summarized in Table1.

- Same family size among all species: The data is token from JGI web server [13].
- Different family size among all species: Olfactory receptor among 12 *Drosophila* species.

Table 1. The summary of biology data.

Data Set type	# of gene families	# of species	# of sequences
Same family size			
Cladiomy	5	3	15
microspor	29	8	232
Different family size			
Olfactory receptor	~60	12	858

2.5 Comparison

We compared BigBigTree against RAxML (Randomized Axelerated Maximum Likelihood), which is a program for sequential and parallel Maximum Likelihood based inference of large phylogenetic trees [14]. The evaluation of BigBigTree is conducted in a desktop with 2.60 GHz, 2 cores CPU and 2 GB memory. Evaluation of RAxML is conducted in CIPRES Science Gateway web service [15]. Considering that the performance using RAxML in version Pthread at the local side was not good enough to build phylogenetic trees, we used RAxML-HPC2 on XSEDE in CIPRES Science Gateway instead, which had better performance than local side. The input data RAxML needed was alignment data, so we used MAFFT to align raw data into alignment data to build phylogenetic trees with RAxML and then compared the results to BigBigTree without including alignment time and bootstrap time. See the result below.

3 Result

The running time comparison is summarized in Table 2. When the input sequences are in a small amount, BigBigTree has about half execution time than RAxML; from microspor and Olfactory receptor two dataset, we can find out that in microspor dataset, BigBigTree has about 12 times faster than RAxML, and in Olfactory receptor dataset, BigBigTree has about 5 times faster than RAxML, and therefore we know that the difference in speed rate has nothing to do with the total number of sequences, we believe it relates to the number of sequences in the cluster after BigBigTree clustered the sequences, because the less number of sequences are in the cluster after clustered, the less length of sequences will be after alignment, and the program can get a better performance. Besides running time, we evaluated the quality of the tree by calculating its maximum likelihood score by PhyML package with topology constrain. Currently, only Cladiomy result is available, where BigBigTree gets little better performance than RAxML, -8643.62 versus -8643.61.

Table 2. Running time(in secs)/maximum likelihood analysis.

Data Set type	BigBigTree*	RAxML*
Cladiomy	13/-8643.62	27/-8643.61
microspor	257	3402
Olfactory receptor	1748	9873

*running of BigBigTree = alignment + tree building, RAxML = only tree building

4 Discussion

We get quite promising results in terms of running time against RAxML thanks our divide and concatenate approach. After brief visualizing tree topology, BigBigTree come out reasonable phylogenetics (Figure 6). We will further evaluate the

quality of topology based on log likelihood. Besides real biology data, simulation data by *SimPhy* will be included into evaluation such that we are able to compare outputs again ground truths based on Robinson Foulds distance metric [16].

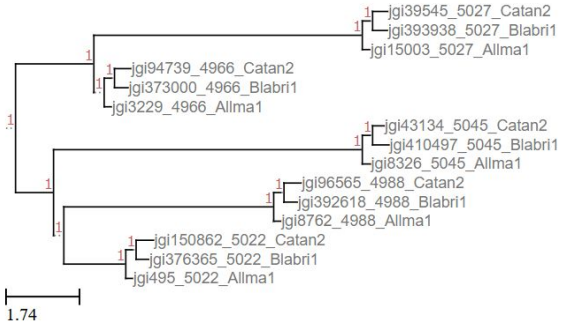


Figure 6 : The snapshot of Cladiomy tree by BigBigTree. The topology is drawed by TreeViewer on ETE 3 [17].

5 Acknowledgement

This work was supported by the *College Student Participation in Research Projects*. The project number of MOST is 106-2813-C-004-037-E.

6 References

[1] Phylogenetic tree - https://en.wikipedia.org/wiki/Phylogenetic_tree

[2] HOMOLOGY - http://www.bio.miami.edu/dana/106/106F05_4.html

[3] Mirarab, S. et al. 2014. ASTRAL: genome-scale coalescent-based species tree estimation. *Bioinformatics* 30, 17 (Sep. 2014), i541–8.

[4] Mirarab, S. and Warnow, T. 2015. ASTRAL-II: coalescent-based species tree estimation with many hundreds of taxa and thousands of genes. *Bioinformatics*. 31, 12 (Jun. 2015), i44–52.

[5] Vachaspati, P. and Warnow, T. 2015. ASTRID: Accurate Species TRees from Internode Distances. *BMC genomics*. 16 Suppl 10, (Jan. 2015), S3.

[6] Difference of Orthology and Paralogy -http://petang.cgu.edu.tw/Bioinfomatics/MANUALS/NCBIblast/Orthology.html

[7] BLAST - https://blast.ncbi.nlm.nih.gov/Blast.cgi

[8] hcluster - https://pypi.python.org/pypi/hcluster

[9] TreeBest - http://treesoft.sourceforge.net/treebest.shtml

[10] Guindon S., Dufayard J.F., Lefort V., Anisimova M., Hordijk W., Gascuel O. New Algorithms and Methods to Estimate Maximum-Likelihood Phylogenies: Assessing the Performance of PhyML 3.0. *Systematic Biology*, 59(3):307-21, 2010.

[11] Di Tommaso, P., Chatzou, M., Floden, E. W., Barja, P. P., Palumbo, E., & Notredame, C. (2017). Nextflow enables reproducible computational workflows. *Nature Biotechnology*, 35(4), 316–319.

[12] Jia-Ming Chang, Matthieu Muffato, Paolo Di Tommaso, Jean-Francois Taly, Javier Herrero, Cedric Notredame, A divide and concatenate strategy for the phylogenetic reconstruction of large orthologous datasets, *SMBE* 2012, poster.

[13] JGI Clusters - https://genome.jgi.doe.gov/clm/run/microsporidia-2017-01.1750;sjugmT?organismsGroup=microsporidia

[14] Alexandros Stamatakis; RAxML version 8: a tool for phylogenetic analysis and post-analysis of large phylogenies, *Bioinformatics*, 30(9):1312–1313 (May 2014).

[15] Miller, M.A., Pfeiffer, W., and Schwartz, T. (2010) "Creating the CIPRES Science Gateway for inference of large phylogenetic trees" in *Proceedings of the Gateway Computing Environments Workshop* (GCE), 14 Nov. 2010, New Orleans, LA pp 1 - 8.

[16] Robinson DF, Foulds LR Math Biosci 1981, Comparison of phylogenetic trees

[17] ETE 3: Reconstruction, analysis and visualization of phylogenomic data. Jaime Huerta-Cepas, Francois Serra and Peer Bork. *Mol Biol Evol*, 33(6):1635-1638 (2016)

適應性演算法診斷大量故障點於超立方體 *

張烜瀚，吳冠頡，賴寶蓮，蔡正雄
國立東華大學資訊工程系
{810221003, 610221112, baolein, chtsai}@gms.ndhu.edu.tw

摘要

要如何即時、有效的在分散式系統中，將故障的處理器正確診斷出來是一項非常重要的議題。 在一個多處理器組成的分散式系統中，沒有人可以確保所有的處理器都能正常工作，隨時都有可能發生固障並且我們無法知道會有多少個處理器發生故障的情況。 為了保持系統的可靠性，我們必需要找出故障的處理器並且代換成正常的處理器。 在這樣的情況下，本篇論文以超立方體網路為模擬對象，使用*MM*模型，提出了適應性演算法，並且我們也推導出一個故障點範圍值 $T(N)$。 只要超立方體網路架構上故障處理器的數量不超過 $T(N)$ 個，我們能判斷出所有的故障處理器。 此外，我們也針對故障處理器數量超過 $T(N)$ 時的情況做了實驗模擬分析，來證明我們所提出的演算法診斷出故障處理器的能力是值得被探討的。

關鍵字: 錯誤診斷(Fault diagnosis)、 t-可診斷(t-diagnosable)、 可診斷性(diagnosability)、 比較模型(Comparison model)、 超立方體(Hypercube)。

1 簡介

隨著科技快速的發展，伴隨而來的是需要發展運算能力更強、更快速的電腦運算系統。 由於高階伺服器及超級電腦造價昂貴，許多大型運算紛紛採用分散式運算系統來提升運算能力。 因為藉由網路連結大量平價電腦的分散式運算系統或許能夠以更少的成本達到相同的效益，而由許多電腦所架構的運算系統裡，我們無法保證所有的電腦都能正常運作，隨時可能都會有電腦故障。 為了保證運算系統的可靠性，我們必需找出故障的電腦並且代換成能正常運作的電腦。 因此**錯誤診斷(fault diagnosis)**的問題日趨重要，如何能即時、正確、有效率的將故障的電腦找出來將是一項非常重要的議題。 找出故障電腦的方式，稱為**診斷演算法(diagnosis algorithm)**。 在診斷演算法中，要找出故障的處理器，需要使用正常的處理器來進行診斷，在許多的文獻中，已經提出了幾種不同的模型來進行 [2, 4, 5, 8, 13]。

Nakajima [10] 在 1981 年時提出了適應性診斷。 適應性診斷主要是藉由前面已知的測結結果，來選擇下一次要測試的對象，而並不是一次性的藉由症狀來全面進行診斷結果，這樣可以減少測試的次數並且提高在診斷中測試的效率，所以降低測試的總次數是適應性診斷的一個主要目標。

比較模型(Comparison model)是由Maeng和Malek所提出，所以又被稱為MM模型[11, 12]。 在此模型中，每個處理器都可以直接對相鄰的處理器進行通信，並且透過傳送一組相同的訊號到已經配對好的一對相鄰處理器，再比較他們回傳的訊號來進行判斷是否有處理器故障的情形。

超立方體(Hypercube)是一個很常被使用的連結網路圖形結構；超立方體具有良好的特性，像是結構簡單、容易擴充和良好的容錯能力等等。 關於超立方體性質的相關研究，可以參考 [3, 6, 7, 9, 15]。

在這篇論文中，我們首先在超立方體網路架構上，推導出一個故障點範圍值 $T(N)$，接著我們提出了一個在MM模型下，找出 $T(N)$ 的適應性診斷演算法。

本篇論文裡，第二節將介紹所使用的名詞以及一些相關的特性和定義； 第三節首先說明本篇論文所推導出來的故障點範圍值，接著說明我們所提出來的適應性演算法和所包含的相關演算法； 第四節我們展示了實驗模擬結果； 第五節提出了我們的結論。

2 名詞與主要特性

在多處理器分散式網路系統的研究中，我們通常使用圖形 $G = (V, E)$ 來表示整個多處理器分散式網路系統。 圖 G 中每個節點 $v \in V$ 表示多處理器分散式系統中的一個處理器，V 為所有處理器的集合。 每個處理器之間通訊的連線，我們通常用邊來表示 在圖 G 中，我們假設$u, v \in V$, 如果 u 和 v 相連，我們使用 $(u, v) \in E$ 表示多處理器分散式系統中的連線。 我們通常用 $|V|$ 表示圖

*This work was supported in part by the Ministry of Science and Technology of the Republic of China under Contract MOST 106-2221-E-259-006.

G 中點的數量，以 $|E|$ 表示圖 G 中邊的數量。 在此篇論文中，圖形的術語和多處理器分散式系統網絡可以互換使用，其他圖形術語和符號，我們建議參考[1, 16]。

路徑(path)是一連串相鄰的點。 **迴圈(cycle)**是一個至少有三個點的路徑，且第一個點與最後一個點相同。 在圖 G 中如果存在一條路徑而且經過所有的點恰好一次，我們稱為**漢彌爾頓路徑(Hmailtonian path)**，同樣地，若存在一個迴圈而且經過所有的點恰好一次，則稱為**漢彌爾頓迴圈(Hamiltonain cycle)**。

一個 n 維度的超立方體，簡稱為 Q_n 是一個很常被使用到的連結網路之一。 在 Q_n 中的每個點可以用一個 n 位元的二進制字串來表示。 如果在 Q_n 中有兩個點只有一個位元的相異，則會有一條邊相連這兩個點，我們稱這兩個點相鄰。 若這兩個點有兩個位元以上的相異，則沒有邊相連，此兩點不相鄰。 一個 n 維度超立方體的 Q_n 包括兩個不相交的 $n-1$ 維度的超立方體，我們分別使用 Q^0_{n-1} 和 Q^1_{n-1} 表示。 在兩個 $n-1$ 維度的超立方體中， 會有一條邊連接在 Q^0_{n-1} 的點 $v = 0v_{n-2}v_{n-3}\cdots v_1v_0$ 和 Q^1_{n-1} 的點 $u = 1u_{n-2}u_{n-3}\cdots u_1u_0$ 之間，且 $v_i = u_i$，$0 \le i \le n-2$。

識別圖形中所有壞點的過程，稱為圖形的**診斷(diagnosis)**。 如果一個圖形在最多故障 t 個點的情況下，所有的故障點都可以被正確識別出來，這個圖被稱為 t-**可診斷(t-diagnosable)**。圖形 G 中可以保證被偵測出來的最大故障點數量，稱為圖形 G 的**可診斷性(diagnosability)**。

在MM模型下，將相同的測試訊號傳送到已經配對好的點 u 和點 v ，並比較所回傳的訊號來進行診斷是否有點故障。 比較是由分別和點 u 以及點 v 相鄰的點 w 來進行。 為了接下來方便說明，本篇論文將好的點以 0 表示，故障的點以 1 表示，比較回傳訊號的結果也使用 0 和 1 來表示。 首先 $(w; u, v)$ 表示點 w 可以對點 u 和點 v 做測試，測試結果記作 $\gamma(w; u, v)$ ，其中點 w 稱為**測試者(tester)**，點 u 和點 v 稱為**被測試者(tested)**。 在本篇論文中，如果點 w 測試點 u 和點 v 這兩個點的測試結果相同，則 $\gamma(w; u, v) = 0$; 若測試結果不相同，則 $\gamma(w; u, v) = 1$。 如果測試者點 w 本身就故障，則無論測試結果是 0 或是 1 ，都不可靠。

因此，若點 w 是一個無故障點且 $\gamma(w; u, v) = 0$，我們可以診斷出點 u 和點 v 都是無故障點。假設點 w 是一個無故障點且 $\gamma(w; u, v) = 1$, 則可以確認至少點 u 和點 v 其中一個是故障點。 一個多處理器運算系統中，所有比較結果的集合，可以用來分析故障點，我們將這集合稱為**症狀(syndrome)**，記為 γ。 圖1列出在MM模型中，點 w 對點 v 和點 u 所有可能的測試結果。

為了保持MM模型結果一致，我們使用以下假設，參考[14]： (1).故障的處理器永遠都是故障的， (2).故障的處理器所回傳的訊號是不正確的， (3).故障的處理器所比較出來的測試結果是不可靠的， (4).兩個故障處理器不會回傳相同的訊息。

測試者 (w)	被測試者(u)	被測試者(v)	測試結果 $\gamma(w; u, v)$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0 / 1
1	0	1	0 / 1
1	1	0	0 / 1
1	1	1	0 / 1

表 1: MM模型所有可能的測試結果

接著我們介紹 $\gamma^0, \gamma^1, \gamma^{grey}$ 三種概念。 假設有三個連續的點，分別是: v_{i-1}，v_i 和 v_{i+1}，其中 v_i 分別是 v_{i-1}，v_{i+1} 的鄰居且 v_i 比較兩個鄰居回傳的結果。 點 v_i 被稱為 γ^0 (或是 γ^1)，如果 $\gamma(v_i; v_{i-1}, v_{i+1}) = 0$ (或是 $\gamma(v_i; v_{i-1}, v_{i+1}) = 1$)。在路徑 P 中，一條子路徑 P' 被稱為 γ^0 (或被稱為 γ^1)，如果這條子路徑是一條具有連續 γ^0 (或被稱為 γ^1) 的最大路徑。 一般而言，γ^0 子路徑和 γ^1 子路徑是交錯在路徑 P 中 (見圖1)。

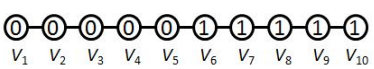


圖 1: $\langle v_1, v_2, v_3, v_4, v_5\rangle$ 是 γ^0 子路徑。 $\langle v_6, v_7, v_8, v_9, v_{10}\rangle$ 是 γ^1 子路徑。

輔助定理1. [7] 假設一條最多有 t 個故障點的路徑 P 上有一條長度是 l 的 γ^0 子路徑 P' 則以下性質成立: (1) $l \ge t+1$, P' 中的所有點以及兩個分別和 P' 中兩個端點相鄰的 γ^1 點都是無故障的。 (2) $l \le t$, 若非 P' 中的所有點皆為故障，則 p' 中的所有點以及和 P' 兩個端點相鄰的 γ^1 點都是無故障的。

我們將輔助定理1中的第二點擴展成推論1。

推論1. 假設 $P = \langle v_{m-2}, v_{m-1}, v_m, v_{m+1}, \ldots, v_{q-1}, v_q, v_{q+1}, v_{q+2}\rangle$ 是一條路徑以及 $P' = \langle v_m, v_{m+1}, \ldots, v_{q-1}, v_q\rangle$ 是一條 γ^0 子路徑。 如果 P' 中的所有點以及兩個 γ^1 鄰居 v_{m-1} 和 v_{q+1} 都沒有故障，則 v_{m-2} 和 v_{q+2} 都是故障點。

輔助定理2. [6] 假設迴圈上有一條長度為 k 的 γ^1 子路徑；當 $k \ge 3$ 時，這條 γ^1 子路徑至少會有 $\lceil k/3\rceil$ 個故障點。

當 $k \ge 2$ ，一條路徑 $P = \langle v_0, v_1, \ldots, v_{2k}\rangle$ 被稱為 γ^{grey} 路徑，如果點 v_{2i} 是 γ^0 ，點 v_{2+j} 是 γ^1 ，其中 $0 \le i \le k$ 並且 $0 \le j \le k-1$。(見圖2)

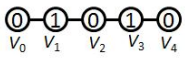


圖 2: γ^{grey} 路徑.

輔助定理3. [7] 假設迴圈上有一條長度為 $2k+1$ 的 γ^{grey} 子路徑，當 $k \geq 2$ 時，這條 γ^{grey} 子路徑至少會有 $\lceil k/2 \rceil$ 個故障點。

根據輔助定理2和輔助定理3，我們可以從 γ^1 和 γ^{grey} 子路徑中預估最少故障點的數量。

關於所需要的測試回合數，我們引用輔助定理4。

輔助定理4. [7] 對於一個迴圈而言，如果迴圈的點數是三的倍數時，只需要三個回合即可找出所有的測試結果。如果不是三的倍數則最少需要四個回合。

在 Q_n 中，漢彌爾頓迴圈總共有 2^n 個點，並不是三的倍數。 由輔助定理4，我們可以知道需要對此漢彌爾頓迴圈進行四個回合的測試，才可以找出所有的症狀。 依靠症狀進行，可以將所有點判斷為無故障，有故障或未知。 如果存在未知點，則需要再多進行兩回合測試，才能替未知點產生新的症狀，並進一步判斷。

3 主要結果

在這篇論文中，我們在 Q_n 上推導出故障點範圍值 $T(N)$ 並提出主要的 **演算法 1 AdaDag(Q_n)**，以及模擬實驗的結果。 我們主要的演算法 **AdaDag(Q_n)** 是藉由在漢彌爾頓迴圈上的症狀 γ ，來判斷 Q_n 中每個點的狀態。 其中，我們使用Algorithm 6 將漢彌爾頓圈分割成一些不重疊的 γ^0 子路徑。 我們使用這些 γ^0 子路徑來診斷每個點的狀態，其中最長的 γ^0 subpath 將在第一次被使用。

值得注意的是，一個故障點最多產生連續三個 γ^1 的點。 最少會有 $N-3t$ 個 γ^0 的點在漢彌爾頓迴圈中。 當有越來越多條 γ^0 子路徑時，最長的 γ^0 子路徑會越短。

根據鴿籠原理，我們有如下的輔助定理5。

輔助定理5. 若有 t 個故障點在 Q_n 中必存在一條最少 $\lceil \frac{N-3t}{t} \rceil$ 個點的 γ^0 子路徑。

根據輔助定理5.我們令 $T(N) = max\{t|\lceil (N-3t)/t \rceil > t\}$ 為我們的最大故障點範圍值。表格1說明了維度和故障點範圍值 $T(N)$ 的關係；我們發現維度越大的 Q_n ，$T(N)$ 會比原本傳統的診斷所偵測到的故障點高出許多。

在本篇論文中，診斷故障點主要是利用 γ^0 子路徑來幫助我們進行診斷。 在開始進行演算法時，會先設定一個初始條件，來幫助我們找到第一條 γ^0 子路徑來進行診斷。 我們可以從產生出來的症狀中，尋找 γ^0 、 γ^1 、 γ^{grey} 三種子路徑。 根據前面的輔助定理2和輔助定理3，可以先將 γ^1 和 γ^{grey} 子路徑中包含的最少故障點數計算出來。 然後我們將 $T(N)$ 減去由 γ^1 和 γ^{grey} 子路徑所估計出來的故障點數，當作找出第一條 γ^0 子路徑的初始條件。

$2^n = N$	Diagnosabilitiy	$T(N)$
$2^6 = 64$	6	6
$2^7 = 128$	7	9
$2^8 = 256$	8	14
$2^9 = 512$	9	21
$2^{10} = 1024$	10	30
$2^{11} = 2048$	11	43
$2^{12} = 4096$	12	62

表 2: 故障點範圍值 $T(N)$ 於不同維度的 Q_n。

在介紹主要的 **AdaDiag(Q_n)** 中各個詳細子程序之前，我們先說明一些符號的定義。 假設 f_1 和 f_g 分別是使用 γ^1 以及 γ^{grey} 子路徑所估計出來的故障點數。 所有 γ^0 子路徑的集合，我們設為 S_{γ^0}。

3.1 在Q_n中建構漢彌爾頓迴圈及產生症狀

在這一小節中，我們將介紹兩個演算法，分別是 **演算法 2 Hamiltonian(n)** 以及 **演算法 3 FourRoundTest**。 首先，我們執行 **演算法 2 Hamiltonian(n)** 在 Q_n 中產生一條漢彌爾頓迴圈。 建構漢彌爾頓迴圈的方式是使用兩個分別在 Q^0_{n-1} 和 Q^1_{n-1} 中相對應的漢彌爾頓路徑相連成一個在 Q_n 中的漢彌爾頓迴圈。

接著，我們利用這一條漢彌爾頓迴圈，執行 **演算法 3 FourRoundTest** 來產生測試結果。 因為 Q_n 中的總點數是 2^n，所以建構出來在 Q_n 中的漢彌爾頓迴圈的點數並不是三的倍數，根據 **輔助定理 4**，需要執行四個回合才能產生每一點的症狀。(見圖3)

3.2 分割漢彌爾頓迴圈演算法

在我們使用 **演算法 3 FourRoundTest** 得到症狀後，我們繼續找出所有需要的 γ^0 子路徑來幫助我們進行診斷。 我們提出了 **演算法 4 CyclePartition** 來分割漢彌爾頓迴圈成為不重疊的 γ^0 子路徑，並存入集合 S_{γ^0}。

圖4表示，執行 **演算法 4 CyclePartition**後，分割成四條 γ^0 子路徑。

3.3 點著色演算法

在使用 **演算法 4 CyclePartition** 後，我們得到漢彌爾頓迴圈上所有的 γ^0 子路徑。 接著，

Algorithm 1 AdaDiag(Q_n)

Input : Q_n 且 $n \geq 5$, $f_1 = 0, f_g = 0$.
Output : 所有點的狀態(無故障點、故障點、未知點)。

Step 1 : 依據故障點範圍值 $T(N)$ ，產生隨機分佈的故障點在 Q_n 中。
Step 2 : 執行 **演算法 2 Hamiltonian(n)** 在Q_n 中建立漢彌爾頓迴圈。
Step 3 : 在漢彌爾頓迴圈中執行 **演算法 3 FourRoundTest** ，產生相對應的症狀。
Step 4 : 執行 **演算法 4 CyclePartition** 將漢彌爾頓迴圈分割成不重疊的 γ^0 子路徑，並將所有的 γ^0 子路徑存入集合 S_{γ^0} 中。
Step 5 : 對 S_{γ^0} 中所有的 γ^0 依長度由大到小進行排序。接著，將長度是 1 的 γ^0 子路徑從 S_{γ^0} 中刪除。
Step 6 : 將 $T(N)$ 減去 f_1 以及 f_g ，並將此值設為我們 **演算法 5 NodeColor** 中找出第一條 γ^0 子路徑的初始條件。
Step 7 : while($S_{\gamma^0} \neq \emptyset$){
執行**演算法 5 NodeColor**。首先，使用從 **Step 6** 所得到的初始條件，在 S_{γ^0} 中找出第一條 γ^0 子路徑，來診斷出無故障點、故障點、未知點。接著，刪除使用過的 γ^0 子路徑，並依序在 S_{γ^0} 中找到下一條子路徑進行診斷。
}
Step 8 : 如果有存在未知點，則執行下一個步驟；否則，執行 **Step 11**。
Step 9 : 執行 **演算法 6 Match** 將所有未知點和無故障點進行配對，並診斷。如果還是存在未知點，則執行下一個步驟；否則，執行 **Step 11**。
Step 10 : 執行 **演算法 7 NodeColor1** 對目前還存在的未知點進行診斷。
Step 11 : 完成診斷並且輸出所有點的狀態。演算法結束。

Algorithm 2 Hamiltonian(n)

Input : A positive integer n
Output : A Hamiltonian cycle(HC) of a Q_n

1: **if** $(n = 1)$ **then**
2: $P = (0, 1)$
3: **else**
4: P =Hamiltonian$(n-1)$
5: Let $P_0 = \langle 0v_0, 0v_1, ..., 0v_{2^{n-1}-1} \rangle$ and $P_1 = \langle 1v_{2^{n-1}-1}, ..., 1v_1, 1v_0 \rangle$
6: Let $HC = (P_0, P_1, 0v_0)$
7: {Comment : $(0v_0, 1v_0) \in E(Q_n)$ and $(0v_{2^{n-1}-1}, 1v_{2^{n-1}-1}) \in E(Q_n)$}
8: **end if**
9: output HC

Algorithm 3 FourRoundTest

Input : Indices of nodes of Hamiltonian cycle(HC)
Output : Syndrome (γ) of all nodes

1: **round 1** : test $(4i; 4i-1, 4i+1)$ and create $\gamma(4i; 4i-1, 4i+1)$ for $0 \leq i \leq 2^{n-2}-1$.
2: **round 2** : test $(4i+1; 4i, 4i+2)$ and create $\gamma(4i+1; 4i, 4i+2)$ for $0 \leq i \leq 2^{n-2}-1$.
3: **round 3** : test $(4i+2; 4i+1, 4i+3)$ and create $\gamma(4i+2; 4i+1, 4i+3)$ for $0 \leq i \leq 2^{n-2}-1$.
4: **round 4** : test $(4i+3; 4i+2, 4i+4)$ and create $\gamma(4i+3; 4i+2, 4i+4)$ for $0 \leq i \leq 2^{n-2}-1$.

我們從 S_{γ^0} 中，將這些 γ^0 子路徑的長度由大到小的排序，並且將長度是 1 的 γ^0 子路徑從 S_{γ^0} 刪除。 然後，我們將 $T(N) - f_1 - f_g$ 當做初始條件，來找出符合我們需求條件的 γ^0 子路徑。我們根據 **輔助定理 4** 和 **輔助定理 5** 提出 **演算法 5 NodeColor** ，應用適合的 γ^0 子路徑來診斷每個點的狀態。 執行 **演算法 5 NodeColor** 時，我們將每個點診斷後的狀態使用三個顏色來表示。 如果是故障點填成黑色，無故障點填成白色，若有無法知道狀態的點，我們填成灰色來表示。 注意，在**演算法 4 CyclePartition**中，$P = \langle v_{m-2}, v_{m-1}, v_m, v_{m+1}, \ldots, v_{q-1}, v_q, v_{q+1}, v_{q+2} \rangle$ 是一條路徑以及 $P' = \langle v_m, v_{m+1}, \ldots, v_{q-1}, v_q \rangle$ 是一條 γ^0 子路徑。 我們將所有填成黑色的故障點，存入集合 S_{f_b}，將所有填成灰色的未知點，存入集合 S_{f_g}。

由圖5中，我們可以知道目前診斷出四個黑色的故障點以及二十七個無故障點，還有一個我們目前並不確定狀態的灰色點。

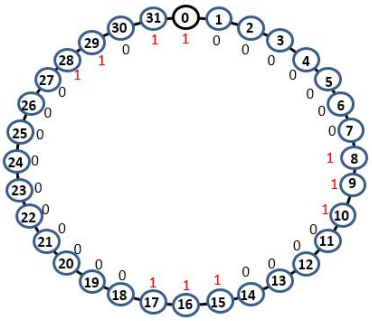


圖 3: 使用 **演算法 2 Hamiltonian(n)** 和 **演算法 3 FourRoundTest** 在 Q_n 中所產生的漢彌爾頓迴圈以及症狀的範例。

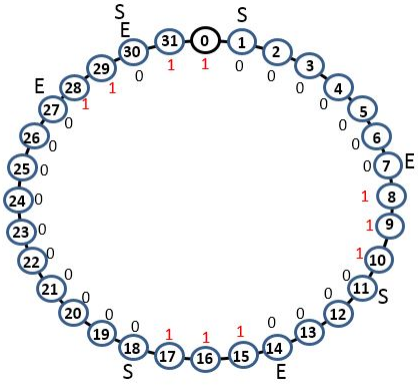


圖 4: 四條 γ^0 子路徑，分別是: $\langle 1,2,3,4,5,6,7\rangle$，$\langle 11,12,13,14\rangle$，$\langle 18,19,20,21,22,23,24,25,26,27\rangle$，$\langle 30\rangle$。

3.4 配對演算法

這一節我們將介紹針對無法確定狀態的灰色點進行配對，並確定其狀態的演算法。 我們提出 **演算法 6 Match** 對所有灰色點依MM模型的規則進行配對，由於我們在找尋可用無故障點鄰居時，可能發生同時使用到同一個鄰居的情況，所以我們將對無故障點鄰居較少的點做優先選擇並進行配對，以避免有未知點無法找到鄰居的情況。 本篇論文採用MM模型，所以我們在配對時會為未知點配對兩個無故障點鄰居，三個一組進行MM模型的診斷。

使用 **演算法 6 Match** 進行配對以及診斷後，如果還存在未知點，我們將會採用 **演算法 7 NodeColor1** 演算法，來診斷剩餘未知點的狀態。

Algorithm 4 CyclePartition

Input : Syndrome (γ) of HC
Output : S_{γ^0} : The set of all γ^0 subpaths sorted by length.

1: **Step 1** : Choose a γ^0 node following a γ^1 node in right direction as the **Start node**. {Comment : The **Start node** is denoted by **S**.}
2: **Step 2** : Proceed inspecting the following node in right direction.
If the current node is γ^0, then repeat Step 2.
If the current node is γ^1, then the previous node is marked **End node**.
Then calculate the number of nodes between **S** and **E**, save the γ^0 subpath into S_{γ^0}. Go to Step 3. {Comment : The **End node** is denoted by **E**.}
3: **Step 3** : Proceed inspecting the following node in right direction.
If the current node is γ^1, then repeat Step 3.
If the current node is γ^0 and unmarked, then mark it **S** and go to Step 2;
otherwise, the algorithm terminates.

4 實驗模擬結果

本節將對於模擬結果進行討論，我們用來運算模擬資料的電腦硬體設備分別是，處理器Intel Core i3-3770 CPU 3.4 GHz、 記憶體16GB、 作業系統 windows 7 64-bit 並且使用C語言將相關演算法撰寫成程式。 假設故障點可以完全被診斷出來，而且沒有其他誤判的情形，我們可以視為診斷成功。 診斷成功的成功率，我們使用 **成功率 = 診斷成功次數 / 總測試次數)**來計算。 我們主要針對故障點範圍值 $T(N)$ 的數量和隨機分佈的情況來討論診斷的成功率。 我們模擬最大到 4 倍的 $T(N)$ ，需要注意的是，在 Q_5 中，故障點達到 4 倍 $T(N)$ 時，已經超過總點數的一半，所以我們不納入模擬結果中。

表格3的模擬結果是經過 1000 次的測試，故障點的分佈情況為隨機分佈，分別以 $T(N)$ 到 $4T(N)$ 個故障點數進行模擬實驗。

我們從表格3模擬結果中，可以觀察到當故障點有 $T(N)$ 個時都可以完全的診斷出來。 而在 $2T(N)$ 中，Q_5、Q_6、Q_7，Q_8，Q_9 因為維度比較低，總點數相對較少，在使用配對演算法時不一定可以配對成功，所以成功率有降低，而在 Q_{10}、Q_{11}、Q_{12} 中因為維度較高，總點數相對較多，使用配對演算法比較容易配對成功，所以診斷的成功率也相對較高。 同樣的道理，在 $3T(N)$ 和 $4T(N)$ 時，因為故障點數提高了許多，所以 Q_5、Q_6、Q_7、Q_8 有可能出現無法配對的情況而

Algorithm 5 NodeColor

Input : S_{γ^0} : The set of all γ^0 subpaths sorted by length, $f_b = 0$.
Output : White nodes, black nodes and grey nodes.

1: **while** $S_{\gamma^0} \neq \emptyset$ **do**
2: **if** $|P'| > T(N) - f_1 - f_g - f_b$ **then**
3: We color the v_{m-2} and v_{q+2} with black, the v_{m-1}, and the v_{q+1} and all nodes in P' with white.
4: **if** $v_{m-2} \in S_{f_b}$ or $v_{q+2} \in S_{f_b}$ **then**
5: $f_b = f_b + 1$ and $S_{f_b} \cup \{v_{q+2}\}$; otherwise, $f_b = f_b + 2$ and $S_{f_b} \cup \{v_{m-2}\} \cup \{v_{q+2}\}$.
6: **end if**
7: **if** v_{m-2} in other γ^0 subpath (P_1) **then**
8: We colour all nodes of P_1 with black, $S_{\gamma^0} - \{P_1\}$ and $S_{f_b} \cup P_1$.
9: $f_b = f_b + |P_1| - 1$.
10: **end if**
11: **if** v_{q+2} in other γ^0 subpath (P_2) **then**
12: We colour all nodes of P_2 with black, $S_{\gamma^0} - \{P_2\}$ and $S_{f_b} \cup P_2$.
13: $f_b = f_b + |P_2| - 1$.
14: **end if**
15: **else**
16: We colour all nodes of P' with grey, $S_{\gamma^0} - \{P'\}$ and $S_{f_g} \cup P'$.
17: **end if**
18: **end while**

導致無法成功的診斷。 而從表格3中，我們也可以觀察到，在 Q_{10}、Q_{11}、Q_{12} 中，當故障點數提高到 $3T(N)$ 和 $4T(N)$ 時， 診斷的成功率還是非常好。

5 結論

在本篇論文中，我們推導出故障點範圍值 $T(N)$ 表示可以診斷的故障點數量界限並且提出演算法來幫助我們進行故障點數量的診斷。 當我們進行實驗模擬後，可以發現在 $2T(N)$ 個故障點數下，診斷的成功率還是相當高，而且隨著維度增加，超立方體的總點數成長速度遠大於故障點的成長速度，所以成功率也會逐漸提高。 另外我們也可以從實驗模擬結果中得知，當超立方體在維度低的時侯，$T(N)$ 的數量並不會超過原本超立方體所能診斷出來的故障點數量太多，但隨著維度升高，例如維度 10 的時侯， $T(N)$) 的值已經達到 30 個故障點數，高出原本故障點數的 3 倍，這可以說明本篇論文所提出來的演算法在診斷大量故障點時具有一定的實用性。

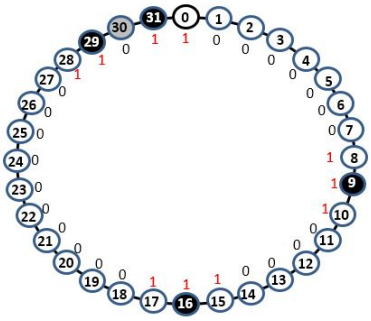


圖 5: 執行**演算法 4 NodeColor** 將各點狀態分別以三種顏色表示。

Algorithm 6 Match

Input : S_{f_g}: 灰點(未知點)的集合。
Output : 點的狀態，白點，黑點與灰點。

Step 1 :將 S_{f_g} 中的每個點，依照好鄰居數量，由小到大排序.
Step 2 : 選擇好鄰居數量最少的點 u 進行配對。
Step 3 : 對每一個和 u 相鄰的無故障點，進行好鄰居數量的計算; 並依照此數量，選出好鄰居數量最少的無故障點 w 和 u 進行配對。
Step 4 : 選擇和 w 相鄰的無故障點 v，來和 w 進行配對，將 u, w and v 配對成一組。
Step 5 : 對u, w and v這一組配對好的點，使用MM模型進行診斷。
Step 6 : 如果 u 無故障，則將 u 填成白色。如果 u 故障，則填成黑色。將 u 從S_{f_g} 中刪除。
Step 7 : 從 S_{f_g} 中，選擇需要配對的下一個點，重覆 **Step 2**; 如果 S_{f_g} 中的所有點都選擇過，演算法結束。

References

[1] J. A. Bondy and U. S. R. Murty, Graph theory with applications, North Holland, New York, 1980.

[2] N. W. Chang and S. Y. Hsieh, "Conditional diagnosability of augmented cubes under the PMC model," *IEEE Transactions on Dependable and Secure Computing,* vol. 9, no. 1, pp.46-60, 2012.

[3] F. Harary, J. P. Hayes, and H. J. Wu. "A survey of the theory of the hypercube graphs," *Computer Mathematics with Applications,* vol. 15, pp.277-289, 1988.

[4] W. S. Hong and S. Y. Hsieh, "Strong diagnosability and conditional diagnosability of aug-

Algorithm 7 NodeColor

Input : 經過演算法 6 Match 診斷後，所有 γ^0 子路徑的所有點集合 N_{γ^0}。
Output : 點的狀態，白點與黑點

1: **while** $N_{\gamma^0} > 0$ **do**
2: 從 N_{γ^0} 中取出點做診斷。
3: **if** 點的狀態是無故障 **then**
4: 整段 γ^0 子路徑 P 上點的狀態及點 V_{m-1}, V_{n+1} 設為無故障，且V_{m-2}, V_{n+2} 設為故障。
5: **if** 設為故障的點 V_{m-2}, V_{n+2} ，位於另一條 γ^0 子路徑 P' **then**
6: 將 P' 上所有點的狀態設為故障。
7: **end if**
8: **else**
9: 整段γ^0 子路徑 P 上所有點的狀態設為故障。.
10: **end if**
11: $N_{\gamma^0} = N_{\gamma^0} - 1$
12: **end while**

維度n	5	6	7	8	9	10	11	12
$T(N)$	5	6	9	14	21	30	43	62
成功率	100%	100%	100%	100%	100%	100%	100%	100%
$2T(N)$	10	12	18	28	42	60	86	124
成功率	58%	61%	78%	89%	95%	100%	100%	100%
$3T(N)$	15	18	27	42	63	90	129	186
成功率	0%	0%	6%	21%	85%	100%	100%	100%
$4T(N)$	20	24	36	56	84	120	172	248
成功率	N/A	0%	0%	0%	2%	93%	100%	100%

表 3: 實驗模擬結果。

mented cubes under the comparison diagnosis model," *IEEE Transactions on Reliability,* vol. 61, no. 1, pp.140-148, 2012.

[5] C.P. Chang, P. L. Lai, J. J. M. Tan, L. H. Hsu, "Diagnosabiltiy of t-connected networks and product networks under the comparison diagnosis model," *IEEE Transactions on Computers*, vol. 53, no. 12, pp. 1582-1590, 2004.

[6] P. L. Lai, "A systematic algorithm for identifying faults on hypercube-like networks under the comparison model," *IEEE Transactions on Reliability*, vol. 61, no. 2, pp. 452-459, 2012.

[7] P. L. Lai, "Adaptive system-level diagnosis for hypercube multiprocessors using a comparison model," *Information Sciences*, 252: 118-131, 2013.

[8] A. Kavianpour and K. H. Kim, "A comparative evaluation of four basic system-level diagnosis strategies for hypercubes," *IEEE Trans. Rel.*, vol. 41, no. 1, pp. 26-37, Mar. 1992.

[9] E. Kranakis and A. Pelc, "Better adaptive diagnosis of hypercubes," *IEEE Transactions on Computers*, vol. 49, no. 10, pp. 1013-1020, 2000.

[10] K. Nakajima, "A new approach to system diagnosis," *Proceedings of the 19th Allerton Conference on Communications, Control, and Computing*, pp. 697-709, 1981.

[11] J. Maeng and M. Malek, "A comparison connection assignment for self-diagnosis of multiprocessors systems," *Proceedings of the 11th International Syposium on Fault-Tolerant Computing*, pp. 173-175, 1981.

[12] M. Malek, "A comparison connection assignment for diagnosis of multiprocessor systems," *Proceedings of the 7th International Symposium on Computer Architecture*, pp. 31-35, 1980.

[13] F. P. Preparata, G. Metze, and R. T. Chien, "On the connection assignment problem of daiganosable systems," *IEEE Trans. Electronic Computers,* vol. EC-16(6), pp.848-854, 1967.

[14] A. Sengupta, A. Dahbura, "On self-diagnosable multiprocessor systems: diagnosis by the comparison approach," *IEEE Transactions on Computers,* vol. 41(11), pp.1386-1396, 1992.

[15] K. C. Wu, P. L. Lai, "A study of adaptive diagnosis for large faults in hypercubes under the comparison model," *Thesis of National Dong Hwa University Department of Computer Science and Information Engineering*, 2016.

[16] J. Xu, "Topological Structure and Analysis of Interconnection Networks," Kluwer: Dordrecht, 2001.

Can Honeycomb Tori Configure Cellular MIHP Parallelism? - for analyzing interference and supporting cipher coding

Li-Yen Hsu
Head Architect, HSU Studio
liyenhsu@ms19.hinet.net

Abstract

To evolve contemporary knowledge economy, cellular information transmission is getting more importance. The mutually Hamiltonian property (MIHP) of cellular honeycomb tori (HT) is studied, used for parallel analyzing interference and supporting cipher coding to offer privacy. It can be expected that the honeycomb tori, HT(m) m≥2, have fully MIHP parallelism.

1 Introduction

Contemporary sensor-information networking may hardly be seen yet, besides means of passive crime prevention through environmental design, “prototyping” [1], more holistic (everywhere or concerning availability) [2,3,4], and reliable (all-time) networking, is becoming critical to facilitate profiling events in real time, to protect and serve travelers and residents, and to prevent incidents, counter disasters, environmental challenges and other wicked problems ([1], i.e. problems that are difficult to anticipate).

The “2016 Taoyuan bus fire” being referred to in the public media, including Wikipedia, and the new London embassy [4] are evidenced that the planning of holistic environmental control to counter intentional terrorist attacks is needed. Pervasiveness is the quality of spreading widely or being present throughout an area or a group of people. Pervasive computing is an emerging development [3]. In terms of prototyping systematic availability, the methodology of this research can also be described as design or task based, and holistic pervasiveness is the quality aimed.

Well enhancing dependable capabilities, i.e. ARM, or “availability, reliability and maintainability” [5,6], to strategically create resource utilization [7] and trust development is the foundation of successful placemaking [8-10]. Effective, efficient capabilities on prevention or mitigation of wicked acts is worthwhile to be considered, through prototyping and with integrative and public welfare oriented minds, before disasters really happen [1].

Facilities concerning attacks through advanced technologies, e.g. through the unmanned aerial vehicle (drone), which can hardly be detected by the roadside surveillance due to it being small and probably very far, are getting more concerns [11(p.215),12]. Such drones are being considered to be legally utilized, e.g. for logistic application; however, they should be detectable, manageable along each lane of the path. In areas off roadside surveillance, more networking studies are considered needed e.g. the network honeycomb torus is introduced for cellular communication applications.

2 Methodology

Communication/information networks are usually illustrated by graphs in which nodes represent processors and edges represent links between processors. It is noted that mathematically, scalable performance is beneficial in building up a network prototype; the scalability is also important for establishing a communication/information sensor- node platform to flexibly support offering the availability for dealing different environment conditions; the mathematical Hamiltonian order helps guarantee maintenance justifiably done (without loss, and with rational efficiency). This paper proposes an approach on the reliability in establishing communication/information networks for managing, serving areas which require quite significant amount of sensor-nodes for reliable communication/ information acquiring, serving and managing.

Let G = (V, E) be a graph if V is a finite set and E is a subset of {(a, b) | (a, b) is an unordered pair of V}. A path is delimited by (x_0, x_1, x2, … x_{n-1}). A path is called a Hamiltonian path if its nodes are distinct and span V. A cycle is a path of at least three nodes such that the first node is the same as the last node. A cycle is called Hamiltonian cycle or Hamiltonian if its nodes are distinct except for the first node and the last node, and if they span V.

A bipartite graph G = (V, E) is a graph such that V = A ∪B and E is a subset of {(a, b) | a ∈ A and b ∈ B}; if G □ F remains Hamiltonian for any F = {a, b} with a ∈ A and b ∈ B, then G is 1p-Hamiltonian. A graph G is 1-edge Hamiltonian

if G □ e is Hamiltonian for any e ∈ E; moreover, if there is a Hamiltonian path between any pair of nodes {c, d} with c ∈ A and d ∈ B, then the bipartite graph G is Hamiltonian laceable. It is noted that laceability is used for concerning the connectivity to keep extended areas being integrated, or vice versa, an area can be managed hierarchically yet effectively.

Assume that m and n are positive integers, where n is even and m≥2. The honeycomb hexagonal mesh HM(n) is the graph with the node set $\{(x_1, x_2, x_3) \mid -n + 1 \le x_1, x_2, x_3 \le n \text{ and } 1 \le x1 + x2 + x3 \le 2\}$. Two nodes (x^1_1, x^1_2, x^1_3) and (x^2_1, x^2_2, x^2_3) are adjacent if and only $|x^1_1 - x^2_1| + |x^1_2 - x^2_2| + |x^1_3 - x^2_3| = 1$. The honeycomb (hexagonal) torus HT(n) is the graph with the same node set as HM(n). The edge set is the union of E(HM(n)) and the wraparound edge set $\{(i, n - i + 1, 1 - n), (i - n, 1 - i, n) \mid 1 \le i \le n\} \cup \{(1 - n, i, n - i + 1), (n, i - n, 1 - i) \mid 1 \le i \le n\} \cup \{(i, 1 - n, n - i + 1), (i - n, n, 1 - i) \mid 1 \le i \le n\}$. Assume d be any integer such that (m-d) is even. The generalized honeycomb torus [13], GHT(m, n, d) is the graph with the node set {(i, j)| 0≤ i< m, 0≤j<n} such that (i, j) and (k, l) are adjacent if they satisfy one of the following conditions: (1) i=k and j=l±1(mod n); (2) j=l and k=i-1 if i+j is even; and (3) i=0, k=m-1, and l=j+d(**mod** n) if j is even. GHT(m, n, n/2) is 1-edge Hamiltonian if n≥4; 1p-Hamiltonian if n≥6 or m=2, n≥4 [14] and Hamiltonian laceable [15], besides, GHT(m, 6m, 3m) is isomorphic to honeycomb torus, HT(m) (**Fig. 1**).

The number of links connecting a node is called the degree, and networks regularly having smaller degree are economic in general [16]. Two Hamiltonian paths, $P_1 = (u_1, u_2, \ldots, u_n(G))$ and $P_2 = (v_1, v_2, \ldots, v_n(G))$ of G from u to v are independent if $u = u_1 = v_1$, $v = u_n(G) = v_n(G)$, and $u_i \neq v_i$ for every $1 < i < n(G)$. A set of Hamiltonian paths, $\{P_1, P_2, \cdots, P_k\}$, of G from u to v, are mutually independent if any two distinct paths in the set are independent from u to v [12,17]. It is noted that at least two "mutually independent Hamiltonian paths" (MIHP) can be considered for parallel, pact wireless information transmission, diagnosing, and offering additional ciphered information, which is considered important for offering real-time private information to logistic consigner [18]. (**Fig. 2**)

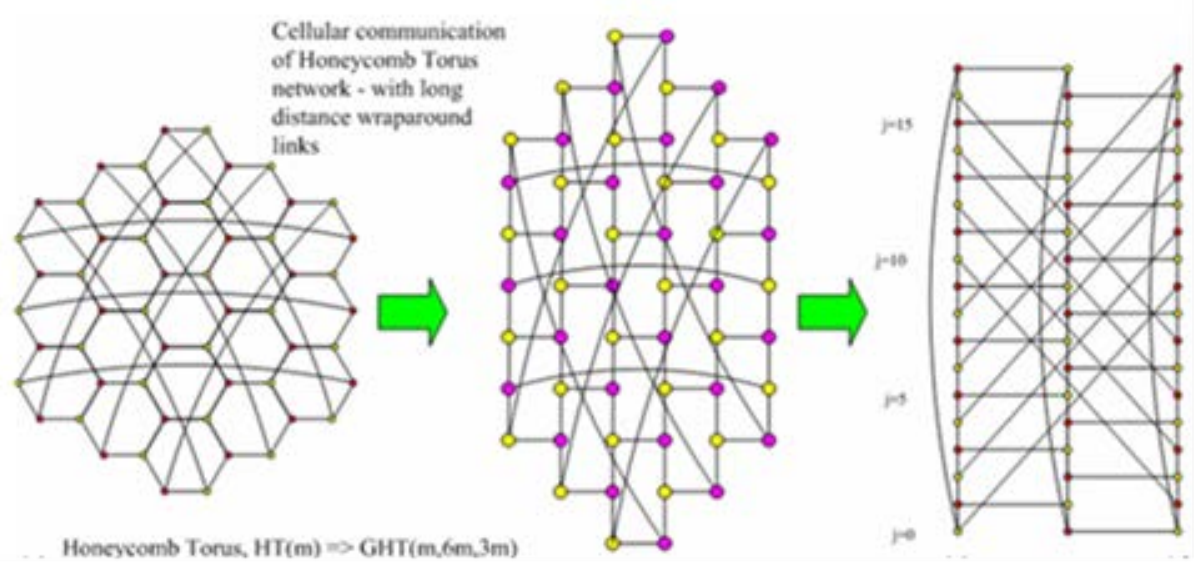


Figure 1. Graph generation of the generalized honeycomb torus from the honeycomb torus.

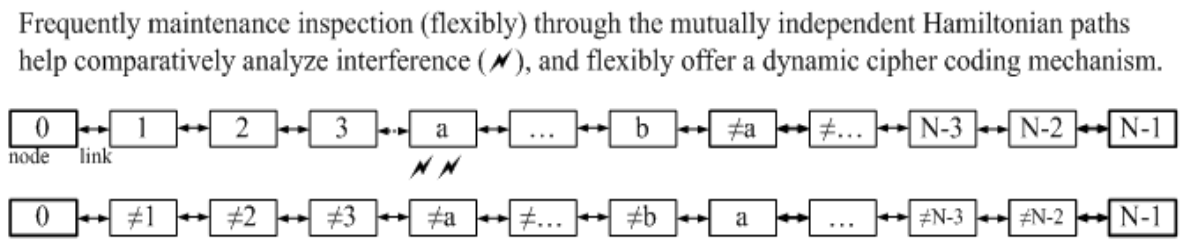


Figure 2. Diagnose interference and assign cipher codes via MIHP.

3 Results

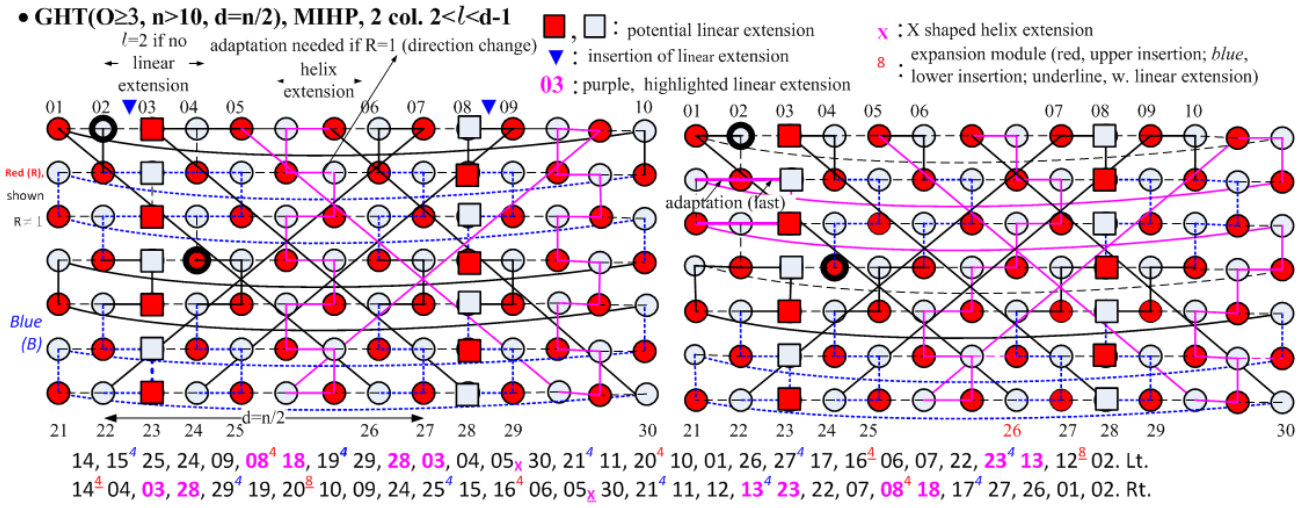


Figure 3. Exemplary case [GHT(odd, n>10, d=n /2), v

Note: the linear extension need be consistent with the definition of the GHT -i.e., the requirement, d=n/2. "l" can be vertically measured from the left end-node, based with the horizontally right link, to another end-node.

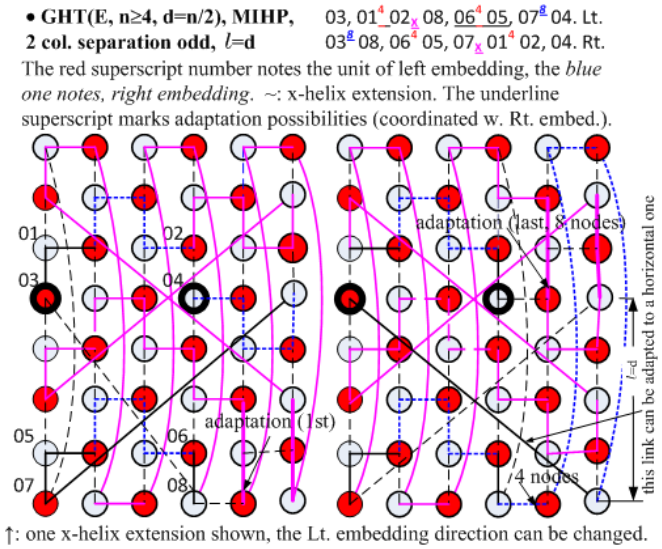


Figure 4. Exemplary case [GHT(even, n≥4, d=n /2), two end-nodes disconnected w. vertical separation d].

Proof (Fig. 3). The superscripts, colored red or blue, show node insertion quantity of each embedding in the upper and lower inserted rows, respectively. Two embedding modes are given, as B and R according to colors. Lt. 22 keep lagging if R>1 and is not conflicted if R=0. If R=1, two nodes' "underline marked" superscripts (in pair "16, 06" and "12, 02" of Lt. pattern, or in pair "20, 10" and "14, 04" of Rt. pattern, whose insertion can have no conflicts due to consistent direction and enough separation) need be exchanged. Comprehensively, nodes "15, 25", 24, 09, "08, 18", 01, 26, "27, 17" keep lagging; Lt. "19, 29" (being able to be supported from linear extension similarly aligned in

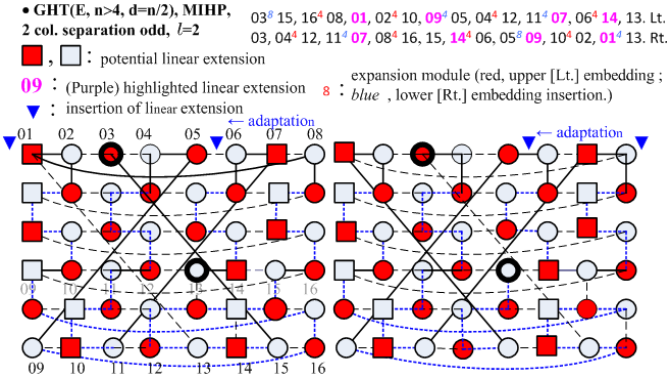


Figure 5. Exemplary case [GHT(even, n≥4, d=n /2), two end-node's vertical separation l=2].

Proof (Fig. 4). The left (Lt.) embedding quantity has only one location and is 8 nodes for each embedding if the x-helix extension is not applied. When the x-helix extension is not applied, the embedding quantity of 8 nodes can be located at the first or the last x-helix extension according to different Hamiltonian paths. All embeddings are considered having consistently same direction. On Lt. pattern, "01, 02" keeps lagging; 08, "06, 05", 07 can avoid conflict because their corresponding separations can not be divided by four. Similarly, x-helix extension can avoid conflict. The adaptation in the last or the first step of Rt. embedding can help maintain the lead or lagging order of each element in x-helix extension. The Lt embedding in Fig.3 can be omitted because its direction can be either upward or downward.

Proof (Fig. 5). By symmetry, assume Lt. embed. ≥ Rt. embed. The content within Lt. "03, 15" can keep lagging to the same nodes within Rt. "05, 09" and "01, 13"; similarly, Lt.15, "16, 08",_ 01, "02, 10", "09, 05". keep lagging. Lt. "04, 12", "11, 07", "06, 14" keep lead.

Proof (Fig. 6). By symmetry, assume Rt. embed. ≥ Lt. embed. The (purple) X shaped helix extension does not be considered in the beginning. Lt. "02⁴ 14" is well separated from Rt. "4⁴ 02" due to Rt. "4⁴ 02". Lt. "13, **24**, 09, 10, 08, 20" can keep lagging. Lt. "22, 21" can keep lead. Lt. "23⁸ 05", 07, "06⁴ 18", "17⁴ 11" can avoid conflicts with Rt. "23⁴

05", **07**, "06$^{\underline{4}}$ 18", "17$^{\underline{8}}$ 11" due to the divisibility by four with the start separation of two. Lt. "23$^{\underline{8}}$ 05" keep lagging to the counter part of Rt. "13^{8} **07**". After Lt. "23$^{\underline{8}}$ 05", Lt. "04^{4} 16, 15" keep lead. Lt. "15^{4} 09" keep lagging to the counter part of Rt. "13^{8} **07**". The counter part of Lt. "10$^{\underline{8}}$" to Rt. "**12**$^{\underline{4}}$ **24**" keep lead, but that to Rt. "08^{8} **20**" keep lagging. Lt. "20^{4} 08" also keep lagging to counter part of Rt. "08^{8} **20**". Rt. "**20**" can have the linear extension separated from that of Lt. "**19**". Those in both Rt. "17$^{\underline{8}}$ 11" and Lt. "01^{8} **19**" are kept lead/lagging; the other contents of Lt. "01^{8} **19**" are in Rt. beginning "03^{4} 21".

Then, consider X shaped helix extension. The node coding can be moved and easily be caused confusing, so such coding need be noted with reasonable but flexible viewpoints. The Rt. X shaped helix extension along with affixed linear extensions is initiated after the Lt. one. The helix extensions are regular a cyclical, we may check the situation of one X-helix extension and then guarantee further extensions without conflicts. It is noted that such movements are along specific linear alignment, and cannot affect lead/lagging or divisibility. Therefore, previous discussions cannot be affected.

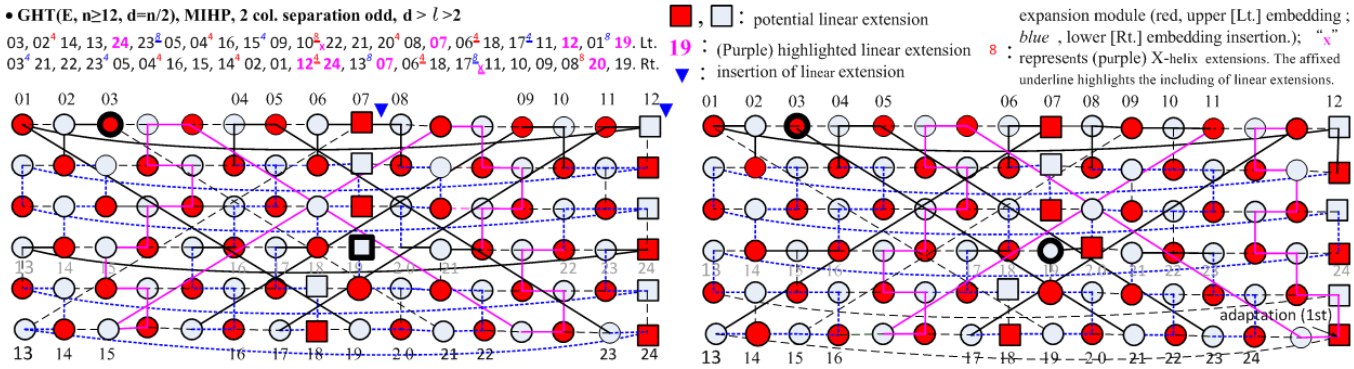


Figure 6. Exemplary case [GHT (even, n>8, d=n /2), two end-node's vertical separation d> l>2].

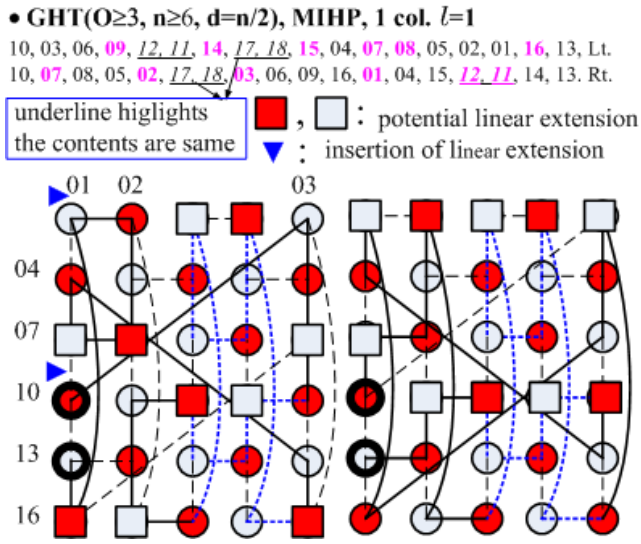


Figure 7. Exemplary case [GHT(odd, n≥6, d=n /2), two end-nodes connected in the same column].

Proof (Fig. 7). Purple digits highlight linear extensions; the linear extension in Rt. 09, 14 or 08 is located and kept lagging in Lt. 12, 01 or 11; the linear extension in Rt. 15 or 16 is located and kept lead in Lt. 02 or 03. Lt. nodes 03, 06, 09, 12, 11, 14, 15, 04 can keep lead. Lt. nodes 17, 18, 07, 08, 05, 02, 01, 16 can keep lagging.

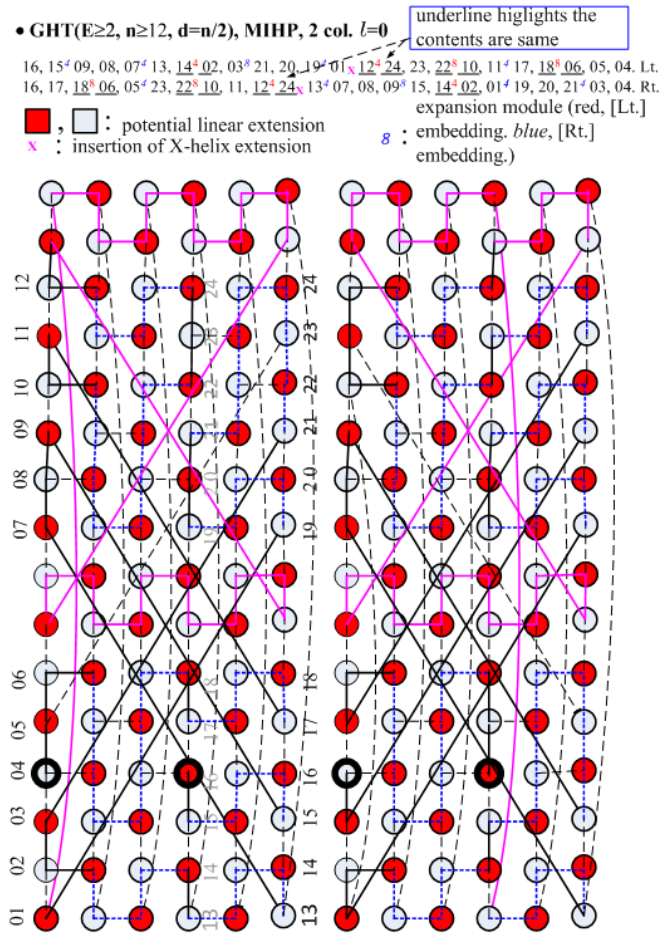


Figure 8. Exemplary case [GHT(even, n≥12, d=n /2), two end-nodes aligned in a row with odd separation].

Proof (Fig. 8). By symmetry, assume Lt. embeddings ≥ Rt. embeddings. Rt. "13, 07", are not in the same sequence as Lt. "07, 13"; yet they do not collide due to the existing of the buffer which is not less than "07, 13" in Lt. pattern. Similarly, Rt. "09, 15", "21, 03" and "01, 19" do not collide with counter parts in Lt. "15, 09", "03, 21", "19, 01". Lt. "11, 17" contains same nodes, also in the same sequence, in Rt. "09, 15" whose potential separations can not be divided by four. Similarly, Rt. "05, 23" contains same nodes, also in the same sequence, in Lt. "03, 21" whose potential separations can not be divided by four. Lt. 08, 20 keep lagging. Lt. nodes 23, 05 keep lead. Rhythmic X-helix extensions with different start locations cannot cause collisions.

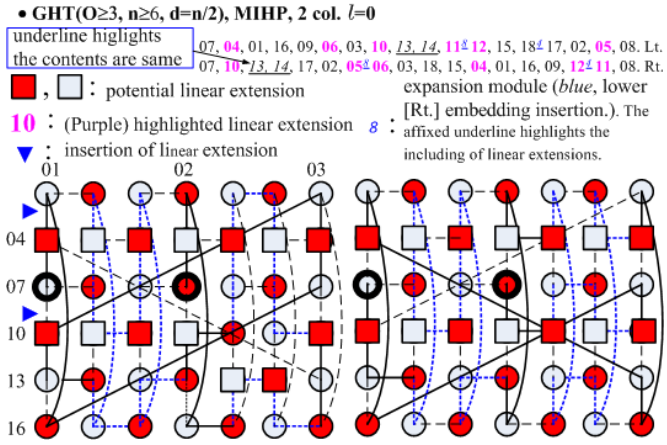


Figure 9. Exemplary case [GHT(odd, n≥6, d=n /2), two end-nodes aligned in a row with odd separation].

Proof (Fig. 9). Separation is designed between end nodes; underlined 13, 14 affix same embedding possibilities in both Rt. and Lt. patterns without conflicts. The embedding in both Lt. "11, 12" and Rt. "05, 06" are in the same sequential order; the former keep lead. The embedding in both Lt. "11, 12" and Rt. "12, 11" are not in the same sequential order; however, with the aid of Rt. "05, 06" the latter keep lead. The embedding in both Lt. "18, 17" and Rt. "12, 11" are in the same sequential order; the latter keep lead. Lt. nodes **04**, 01, 16, 09, **06**, 03, **11**, **12** can keep lagging. Lt. nodes **10**, 13, 14, 15, 18, 17, 02, **05** can keep lead.

Proof (Fig. 10). Lt. nodes "**05**, 31" have same composition as that in Rt; besides, they and nodes 38, 28 can have no conflict with those in Rt. because their original separation distance is two yet the increasing module is four; i.e., dis-divisibility (**mod** 4≠0). Lt. nodes **15**, "16, 06", 21, **11**, 12, 13, 14 keep lag. Lt. nodes 04, "03, 37", 36, **26** can keep leading. The linear extension affixed with Lt. 36, 32, and 22 being same and having same extension direction to that affixed with Rt. 35, they have no conflict due to dis-divisibility (**mod** 4≠0). The linear extension affixed with Lt. 32 and 22 being coordinated with adjacent nodes (i.e., 31 and 21) and having same extension direction to that affixed with Rt. 31, and 21 respectively, they have dis-divisibility (**mod** 4≠0) to avoid conflicts. The inner embedded nodes in either Lt. "14, 04" or Rt. "12, 02" will have no conflicts between Lt. ("14, 04" or X-helix extension) and Rt. ("12, 02" or X-helix extension) due to well separation (keeping lag or leading). Lt. 27, 17, "18, 08", 07, 33 have no conflict, yet the original leading will be changed to keeping lag if X-helix is added. Lt. 23, 24, "35, 01" can keep leading.

Whether X-helix extension exists cannot affect above relations. Both patters' X-helix extensions have same direction and start with enough separation even though the start position being different.

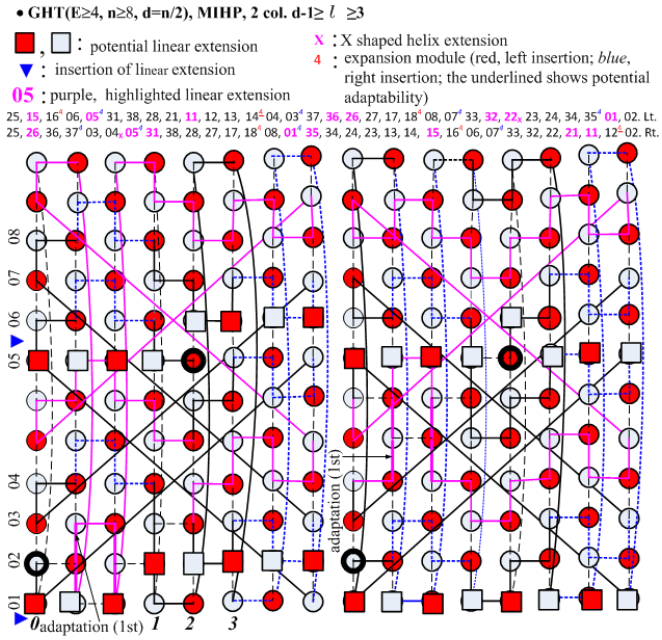


Figure 10. Exemplary case [GHT(even, n≥8, d=n /2), two end-node's vertical separation d> l>1].

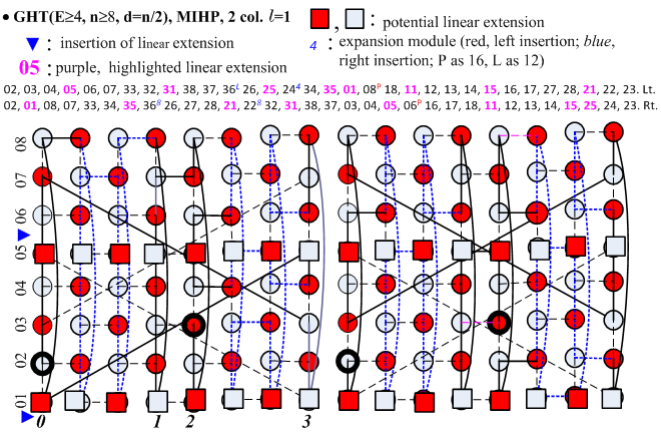


Figure 11. Exemplary case [GHT(even, n≥8, d=n /2), two end-node's vertical separation l=1].

Proof (Fig. 11). The right insertion on Fig. 11 is direction upward for the even ones and downward for the odd ones; specifically, the expansion module of "07, 33" is L (or12) and "35, 01" is 4 for the first Rt insertion; moreover, Lt nodes 03, 04, **05**, 06, 32, **31**, 38, 37, **25**, 24, 18, **11**, 12, 13, 14, **15**, can keep lag, and Lt nodes 07, 33, 36, 26, 34, **35**, **01**, 08, 16, 17, 27, 28, **21**, 22 can keep leading. By alternating

downard-upward Rt. insertion, Lt. “07, 33” can be conflict free with Rt. “37, 03”; strictly, they are fully separated. That is consdering conflicts beteeen Lt “35, 01” and Rt. “37, 03”, and conflicts between Lt “07, 33” and Rt “07, 33”; because inserting nodes’ directions are consistent, conflicts can be prevented. Similarly, Lt. “08, 18” and Rt. “06, 16” have same contents yet conflicts can avoided.

4 Discussion

In GHT formation, MIHP can probably not be found in some cases, whose end nodes are “adjacent” and vertical dimension is small (i.e., n is 4 or 8). The honeycomb (hexagonal) torus HT(2) is isomorphic to GHT(2, 12, 6). Especially from Fig. 8 (n=12) □whose end-nodes are adjacent in GHT, that the honeycomb torus, HT(m) and m≥2, has fully MIHP performance can be expected.

5 Conclusion

Contemporary economy significantly counts on globalization activities. Openness or incorporation participation in knowledge based economy is getting more attention. Just-in-time economy can push the strategy of using computational transportation facilities, which naturally can benefit tourism and business if security-information services can be synergistically acknowledged.

Just as mankind use two eyes for seeing, two ears for hearing, two nostrils for smell, to well sense and communicate the changing environments and related information, plural surveillance-information networking is suggested, especially in the era that wireless telecommunications are naturally prevailed.

The honeycomb (hexagonal) torus HT(m) is isomorphic to GHT(m, 6m, 3m), m≥2. It can be expected that honeycomb tori, HT(m) have fully MIHP performance.

References

[1] T. Fisher. Public values and the integrative mind: how multiple scores can collaborate the city building. P*AR, 2014, 76(4): 457-464*.

[2] L.-Y. Hsu. Building Taiwan aeropolis of airports through infrastructuring care and integrity. *JCEA, 2017, 11(6): 608-615.*

[3] D, Saha, A. Mkheryjee, and S. Bandyopadhyay. Networking infrastructure for pervasive computing: enabling technologies and systems. *New York: Springer, 2012; p.2, p.52.*

[4] C. Slessor. The new London embassy. *J. AIA, 2018, Feb. p.70.*

[5] D. Daley. Design for Reliability: Developing assets that meet the needs of owners. *New York: Industrial Press, 2011; p.19, p.203.*

[6] J. Knezevic. Systems maintainability- analysis, engineering and management. *London: Chapman Hall, 1997; p.11, p.221*.

[7] C. Hill, M. Schilling, and G. Jones. Strategic management: theory & cases: An Integrated Approach. *New York: Cengage, 2017; p.46, p.84.*

[8] S. Robbins and M. Coulter. Management. *Upper Saddle River, NJ: Prentice Hall, 2012; p.473.*

[9] L. Schneekloth and R. Shibley Placemaking: The art and practice of building communities. *Hoboken, NJ: Wiley, 1995; pp.109-120, p.229.*

[10] C. Tilly Cities, states, and trust networks: chapter 1 of cities and states in world history. In Contention and trust in cities and states, M, Hanagan and C. Tilly (Eds.). *New York: Springer, 2011, pp.3-7.*

[11] L. Hoppe and L. Awsede. Design of control of UAV objects. In Advanced technologies for intelligent systems of national border security; A. Nawrat, K. and A. SimekSwieriak (eds.). *New York: Springer, 2013, pp.211-220.*

[12] L.-Y. Hsu. Ubiquitous integrity via network integration and parallelism - sustaining pedestrian / bike urbanism, *Algorithms, 2013, 6(3): 459-470.*

[13] H.-J. Cho, and L.-Y. Hsu, “Generalized honeycomb torus.” *INFORM PROCESS LETT, 86(4), 2003, pp. 185-190.*

[14] L.-Y. Hsu, F.-I. Ling, S.-S. Kao, H.-J. Cho, Ring embedding in faulty generalized honeycomb tori – GHT(m,n,n/2). *INT J COMPUT MATH, 87(15), 2010, pp. 344-3358.*

[15] L.-Y. Hsu, T.-Y. Lin, and S.-S. Kao, “The Hamiltonian laceability of some generalized honeycomb tori”, *AIP Conference Proceedings – International Electronic Conference on Computer Science 2007, 1060(1), 2008, pp.302 ~306.*

[16] I. Stojmenovic, “Honeycomb Networks: Topological properties and communication algorithms. *IEEE T PARALL DISTR 8 (10), 1997, pp.1036-42.*

[17] Y.-H. Teng, J. Tan, T.-Y. Ho, and L.-H. Hsu, On Mutually independent Hamiltonian paths. *APPL MATH LETT 19(4), 2006, pp. 345-50.*

[18] L.-Y. Hsu. Parallelism enhancing cellular communication - on knowledge city evolving (accepted). IEEE *2018 WCICA Changsha, China, July.*

適用於 GPU 上的快速機密分享

蔡穎榛、徐熊健
Department of Computer Science and Information Engineering,
Ming Chuan University, Taiwan
alonstilllove@gmail.com, sjshyu@mail.mcu.edu.tw

Abstract

(k, n) 門檻式機密分享機制是將機密 s 編碼成 n 個部分(稱為分享機密)—分給 n 位機密分享參與者—任 k 位參與者可利用其擁有的分享機密求得 s，而任何少於 k 個分享機密皆無法求得機密。完善的機密分享機制除可提供完美安全性，還可容許高達 n–k 個分享機密意外損毀。在多媒體資料普遍使用之際，資料的大小隨而增加，機密分享的效率即成為實際應用上的關鍵訴求。我們在循序的 CPU 和平行的 GPU 平台上分別實作 Kurihara 等人提出的機密分享機制。實驗結果顯示：相較 CPU，GPU 可達到令人滿意的加速比。

1. 簡介

機密分享 (secret sharing) 是一種機密保護的方法，由機密分享的參與人一同保管機密，只有在滿足某些條件的時候，參與人方可共同將機密解出。將機密 s 加密分成 n 份 (shadows，或稱之為分享機密)，分給 n 位參與者，當任 k 位參與者一同使用自身的分享機密，即可還原原始機密 s，少於 k 位，則無法還原機密，稱之為 (k, n) 門檻式機密分享機制 (threshold secret sharing scheme, TSSS)。

Shamir [1] 於 1979 年利用多項式的特性，設計了門檻式機密分享機制—其簡單且容易執行，啟發了許多後續的研究。Kurihara 等人 [2] 認為 Shamir 的機制需要較多的計算來實現加密與解密的演算法，於是在 2008 年提出另一個運用 XOR (eXclusive-OR) 的門檻式機密分享機制，以期減少計算的時間，使其比 Shamir 的機制要來得快速。由 [2] 提出的實驗結果可知：在循序執行平台（個人電腦： Intel Core 2 Duo E6600 (2.4 GHz) 處理器、 2.0 GB 記憶體、Windows XP SP2 作業系統、Microsoft Visual C++ .NET 2003 撰寫編譯程式）上，以 4.5 MB 資料測試，其機制在 (k, n) = (3, 11) （或 (3, 59)）的加密 / 解密計算可比 Shamir 的機制快上 5 / 45 倍（或 6 / 16 倍）。此外，他們也提到若 n_p 變很大，Shamir 的方法可能會快過他們。

GPGPU (general-purpose computing on graphics processing units) [3]—本文簡稱 GPU—因其高效能的計算能力，已然成為近年許多應用中解決耗時問題的實用方案。吾人可在 CUDA (compute unified device architecture) [4] 環境下，使用高階程式語言（如 C、Java 等）撰寫編譯，使單張顯示卡（或多張卡）中，包含高達數千個核心 (cores) 的 GPU 同時間執行大量平行運算。

我們在本研究中將 Kurihara 等人所提出的門檻式機密分享機制分別於個人電腦的循序（以 CPU 計算為主）與平行（以 GPU 計算為主）執行平台上實作，調較其性能，探討其高速計算的可能性。

本文後續部份架構如下：第 2 節介紹 Kurihara 等人所設計的門檻機制；第 3 節敘述此機制的編碼/解碼循序演算法；第 4 節討論適

合於 GPU 上執行的平行編碼/解碼演算法；第 5 節陳列我們實作的實驗結果，並對其做分析比較；第 6 節為本文結語。

2. Kurihara 機密分享機制

Kurihara 等人所提出的機密分享機制，機密 $s \in \{0,1\}^{d(n_p-1)}$，分成 $s_1, s_2, \ldots, s_{n_p-1}$，共 n_p-1 個 d 位元 (bit) 的片段，其中 n_p 為質數，$n_p \geq n$，$d > 0$。$s_0 = 0^d$，$s_0 \oplus a = a$，隨機產生 $(k-1)n_p-1$ 個 $\{0,1\}^d$ 的亂數：$r_0^0, \ldots, r_{n_p-2}^0, r_0^1, \ldots, r_{n_p-1}^1, r_0^{k-2}, \ldots, r_{n_p-1}^{k-2}$。分享機密片段會與所產生的亂數做運算如下：

$$w_{(i,j)} = \left\{ \bigoplus_{h=0}^{k-2} r_{h\cdot i+j}^{h} \right\} \oplus s_{j-i} \qquad (1)$$

其中 $0 \leq i \leq n-1$，$0 \leq j \leq n_p-2$；$w_{(i,j)}$ 大小皆為 d 位元。之後將各 $w_{(i,j)}$ 針對特定 i 進行合併：

$$w_i = w_{(i,0)} \| \ldots \| w_{(i,n_p-2)} \qquad (2)$$

再分給第 i 位參與者 $(0 \leq i \leq n-1)$。

還原時 k 位參與者將其擁有的 $w_{t_0}, \ldots, w_{t_{k-1}}$ 分享機密各自分成 (n_p-1) 個 d 位元片段，將其組成一個 $k(n_p-1)$ 維度的二元向量 (binary vector) **w**：

$$\mathbf{w} = (w_{(t_0,0)}, w_{(t_0,1)}, \ldots, w_{(t_0,n_p-2)}, \ldots, w_{(t_1,0)}, w_{(t_1,1)}, \ldots, w_{(t_1,n_p-2)}, \ldots, w_{(t_{k-1},0)}, w_{(t_{k-1},1)}, \ldots, w_{(t_{k-1},n_p-2)})^{\mathrm{T}} \qquad (3)$$

而後藉由函式 MAT 建立一個 $k(n_p-1)\times k(n_p-1)$ 二元矩陣 **M**（請見文後函式 MAT 的演算法），最後計算 **M**·**w**，就能還原原始機密 s，而任何少於 k 位的參與者，無法將機密解出。

3. 循序演算法

輸入的資料，在電腦上被視為二元資料 (binary data)，資料的處理，可使用 unsigned char、unsigned short int、unsigned int、unsigned long long int 等四種資料型別 (data type) 實作，即 d 可分別設定為 8、16、32、64 位元。Kurihara [2] 所提之演算法，敘述如下，E1 為加密演算法。

Encoding
Input: $s \in \{0,1\}^m$ where $m = d(n_p-1)$
Output: $(w_0, w_1, \ldots, w_{n-1})$

E1
1. $s_0 = 0^d$, $s = s_1 \| s_2 \| \ldots \| s_{n_p-1}$
2. for (each i, $0 \leq i \leq k-2$)
   { for (each j, $0 \leq j \leq n_p-1$)
     { $r_j^i = \text{random}(\{0,1\}^d)$
     }
   } // discard $r_{n_p-1}^i$
3. for (each i, $0 \leq i \leq n-1$)
   { for (each j, $0 \leq j \leq n_p-2$)
     { $w_{(i,j)} = (\bigoplus_{h=0}^{k-2} r_{h\times i+j}^i) \oplus s_{j-i}$
     }
     $w_i = w_{(i,0)} \| w_{(i,1)} \| \ldots \| w_{(i,n_p-2)}$
   }
4. return $(w_0, w_1, \ldots, w_{n-1})$

解密時，則採用 D1 的演算法，M 則是由函式 MAT 產生。

Decoding
Input: $(w_{t_0}, w_{t_1}, \ldots, w_{t_{k-1}})$
Output: s

D1
1. for (each i, $0 \leq i \leq k-1$)
   $w_{(t_i,0)} \| w_{(t_i,1)} \| \ldots \| w_{(t_i,n_p-2)} = w_{t_i}$
2. $\mathbf{w} = (w_{(t_0,0)}, w_{(t_0,1)}, \ldots, w_{(t_0,n_p-2)}, \ldots, w_{(t_1,0)}, w_{(t_1,1)}, \ldots, w_{(t_1,n_p-2)}, \ldots, w_{(t_{k-1},0)}, w_{(t_{k-1},1)}, \ldots, w_{(t_{k-1},n_p-2)})^{\mathrm{T}}$
3. $\mathbf{M} = \text{MAT}(t_0, t_1, \ldots, t_{k-1})$
4. $(s_1, s_2, \ldots, s_{n_p-1})^T = \mathbf{M} \cdot \mathbf{w}$
5. $s = s_1 \| s_2 \| \ldots \| s_{n_p-1}$
6. return s

在 D1 中，第 4 步驟為 **M** 與 **w** 矩陣相乘，由於此兩個矩陣都是 $\{0,1\}$ 矩陣，因此為 0 的項可以不看，只需做為 1 的，計算次數即可減少，D2 為採用此方式之演算法。

D2
1-3. Same as D1
4. // $(s_1, s_2, \ldots, s_{n_p-1})^{\mathrm{T}} = \mathbf{M} \cdot \mathbf{w}$
   for (each r, $0 \leq r \leq n_p-2$)
   { for (each c, $0 \leq c < k(n_p-1)$
     { if (M[r][c] = 1)
       $s_r = s_r \oplus \mathbf{w}_c$
     }
   }
5. $s = s_1 \| s_2 \| \ldots \| s_{n_p-1}$
6. return s

M1 為函式 MAT 的演算法，先建立一個 $k(n_p-1)\times(kn_p-2)$ 的二元矩陣 G，再與單位矩陣

$\mathbf{I}_{k(n_p-1)}$ 合併成 $[\mathrm{G} \quad \mathbf{I}_{k(n_p-1)}]$，第 3 步的 $FG()$ 函式代表對 $[\mathrm{G} \quad \mathbf{I}_{k(n_p-1)}]$ 做向前高斯消去 (forward Gaussian elimination)，得到階梯形矩陣 (row echelon form) $[\bar{\mathbf{G}} \quad \mathbf{J}]$，而 $[\bar{\mathbf{G}} \quad \mathbf{J}]$ 可分為六個區塊，如式 (4)：

$$[\bar{\mathbf{G}} \quad \mathbf{J}] = \begin{bmatrix} \mathbf{G}_2 & \mathbf{G}_1 & \mathbf{J}_1 \\ \varnothing & \mathbf{G}_0 & \mathbf{J}_0 \end{bmatrix} \tag{4}$$

只需右下兩個區塊 $\mathbf{G}_0$ 與 $\mathbf{J}_0$，表示為 $[\mathbf{G}_0 \ \mathbf{J}_0]$，接著對 $[\mathbf{G}_0 \quad \mathbf{J}_0]$ 進行 $BG()$ 函式的處理，也就是向後取代 (backward substitution)，將左邊變為單位矩陣 $\mathbf{I}_{n_p-1}$，右側即為所需的 $\mathbf{M}$ 矩陣。

MAT
Input: $t_0, t_1, t_2, \ldots, t_{k-1}$
Output: $\mathbf{M}$

M1
1. for (each i, $0 \le i \le k-1$)
   { for (each j, $0 \le j \le n_p - 2$)
   { $\mathbf{v}_{(t_i, j)} = VEC(t_i, j)$
   }
   }
2. $\mathrm{G} = (\mathbf{v}_{(t_0, 0)}, \ldots, \mathbf{v}_{(t_{k-1}, n_p-2)})^{\mathrm{T}}$
3. $\begin{bmatrix} \mathbf{G}_2 & \mathbf{G}_1 & \mathbf{J}_1 \\ \varnothing & \mathbf{G}_0 & \mathbf{J}_0 \end{bmatrix} \leftarrow FG([\mathrm{G} \quad \mathbf{I}_{k(n_p-1)}]) = [\bar{\mathbf{G}} \quad \mathbf{J}]$
4. $[\mathbf{I}_{n_p-1} \quad \mathbf{M}] = BG([\mathbf{G}_0 \quad \mathbf{J}_0])$
5. return M

---

4. 平行演算法

為了利用 GPU 中多核 (cores) 的平行能力，我們將資料 (data) 分割成顆粒般的小片段，每份顆粒片段都能被 GPU 的執行序 (thread) 處理。一個 N 位元組 (bytes)，字組大小 (word size) = l 位元組的二元資料 (binary data) D，以及能同時運算的執行序最大數量 $maxT$，$q = N/l$，而一次可做 n_p-1 個位元組，$\lambda = \lceil q/(n_p-1) \rceil$，由於 λ 可能會大於 $maxT$，所以我們將 λ 個片段 (segments) 分成 $\eta = \lceil \lambda/maxT \rceil$ 個區塊 (regions) $e_1, e_2, \ldots, e_\eta$，每一個區塊包含 $maxT$ 個片段，如果 λ 無法被 $maxT$ 整除，最後一塊區塊則包含 (λ mod $maxT$) 個片段。$maxT$ 份片段分配給 GPU 中的核心同時運算，稱為一個運行 (*run*)，所有的工作會在 $\lceil \lambda/maxT \rceil$ 個運行後完成。

在加密的階段，每一份片段需要 $(k-1)n_p-1$ 個亂數，一次運行中，我們需要 $((k-1)n_p-1)maxT$ 個亂數，總共則是需要 $\lambda((k-1)n_p-1)$ 個亂數，這個隨機產生亂數的任務，可由 GPU 平行運算產生。

解密時，每個片段都需一個 $\mathbf{w}$ 矩陣，總共則需要 λ 個，此步驟，也可用 GPU 平行產生。

Parallel Encoding
Input: k, n, secret D, n_p
Output: n shares: $Y = \{Y_1, Y_2, \ldots, Y_n\}$

$PE(k, n, D, n_p, N, l, maxT)$
1. $q = N/(n_p-1)$
2. $\lambda = \lceil q/l \rceil$
3. $\eta = \lceil \lambda/maxT \rceil$
4. Allocate memory of Y_j for $1 \le j \le n$ in CPU and GPU
5. for (each region e_i, $1 \le i \le \eta$)
   { __paralleldo<<$maxT$>>__
   $\{y_{1i}, y_{2i}, \ldots, y_{ni}\} = Encode(k, n, e_i, n_p)$
   copy $\{y_{1i}, y_{2i}, \ldots, y_{ni}\}$ back to CPU
   __parallelend<<$maxT$>>__
   }
6. for (each participant i, $1 \le i \le n$)
   $Y_i = y_{i1} \cup y_{i2} \cup \ldots \cup y_{i\eta}$
7. return $Y = \{Y_1, Y_2, \ldots, Y_n\}$

__paralleldo<<$maxT$>>__ 與 __parallelend<<$maxT$>>__之間的區域，在 GPU 中有 $maxT$ 個執行序同時間運算，請注意，實際上 GPU 上的核心數取決於硬體設備，而 $maxT$ 只是概念上的執行序數量，在我們測試平台上，$maxT$ 選定為 65535×1024。演算法 PE 中的 $Encode$，即為 E1，而 PD 演算法的 $Decode$，則是從 D1 與 D2 之間，選出最有效率的演算法。

所有片段所需的 M 矩陣，都是一樣的，因此，函式 MAT 只需做一次即可。

Parallel Decoding
Input: $\mathcal{T} = \{Y_{i_1}, Y_{i_2}, \ldots, Y_{i_t}\}$, $P = \{i_1, i_2, \ldots, i_t\}$
Output: secret D (if $t \ge k$), or random file (otherwise)

$PD(t, n_p, \mathcal{T}, P, N, l, maxT)$
1. $q = N/(n_p-1)$
2. $\lambda = \lceil q/l \rceil$
3. $\eta = \lceil \lambda/maxT \rceil$
4. MAT(t, P)

5. Allocate memory of D in CPU and GPU
6. for (each region e_i, $1 \le i \le \eta$)
   { __paralleldo<<<$maxT$>>>__
   $\{d_1', d_2', \ldots, d_t'\} = Decode(maxT, t, n_p, \mathcal{T}, M, e_i)$
   copy $\{d_1', d_2', \ldots, d_t'\}$ back to CPU
   __parallelend<<<$maxT$>>>__
   }
7. return $\mathcal{D} = d_1' \cup d_2' \cup \ldots \cup d_t'$ // $\mathcal{D} = D$, if $t \ge k$

5. 實驗結果

我們分別在 CPU 和兩種 GPU 平台測試上述的循序與平行演算法，CPU 循序平台是使用 Windows 7 作業系統、i7-4790 (3.6 GHz) 處理器與 8 GB 記憶體的個人電腦 (personal computer)，於 Borland C++ Builder 上撰寫編譯。GPU 平行平台 1 則是在相同的個人電腦上，使用 GTX 760 顯卡，其擁有 1152 個核心與 2 GB 記憶體；GPU 平台 2 則使用 Titan X 顯卡，其具有 3072 個核心與 12 GB 記憶體；平行 CUDA 程式皆以 Visual Studio 2015 撰寫。

以下實驗數據，皆使用 15.9MB 的測試資料，單位為秒。令 SE1 (SD2) 表示在 CPU 平台上執行 E1 (D2) 演算法的加（解）密的運算時間；表 1 是在設定不同 (k, n) 和 $d = 8$、16、32 和 64 情況下 SE1 和 SD2 的時間比較。由表 1 可知，當使用 64 位元時 ($d = 64$) 所需花費的加密或解密時間皆為最少—加（解）密比上 8 位元 ($d = 8$) 快了 2-5 (7-8) 倍。

令 $PE1_{760}$ ($PD2_{760}$) 表示 GTX 760 顯示卡執行 E1 (D2) 的運算時間；表 2 即為 $PE1_{760}$ 與 $PD2_{760}$ 在不同 (k, n) 和 $d = 8$、16、32 和 64 情況下的時間比較。以 (6, 31) 加密為例，採用 $d = 64$ 比 $d = 8$ 快了約 4.8 倍，解密的話，則是 $d = 8$ 比 $d = 64$ 慢了 7.8 倍左右。

令 $PE1_X$ ($PD2_X$) 表示 Titan X 顯示卡執行 E1 (D2) 的運算時間；表 3 陳列了 $PE1_X$ 與 $PD2_X$ 在不同 (k, n) 和 $d = 8$、16、32 和 64 情況下的時間。由表 3 可知，加密時，使用 $d = 16$ 在此平台上是最快的，我們認為 Titan X 在 $d = 16$ 時的資料傳遞、產生亂數…等的效能較其它 d 值為佳，因此總加密時間比 $d = 64$ 快。至於解密的表現，則以 $d = 64$ 最快。

表 4 是從表 1、2、3 中選出個別表現最好的數據做比較，在循序平台及 GTX 760 平台，我們都是選擇 $d = 64$ 的數據，而 Titan X 雖然加密時間是 $d = 16$ 較快，但考量實用上解密所需時間大都比加密時間來得重要，因此我們選擇解密時間較快 $d = 64$ 的數據。

表 1、d 不同時，循序演算法時間比較

(k, n)	Encoding SE1				Decoding SD2			
	8 bits	16 bits	32 bits	64 bits	8 bits	16 bits	32 bits	64 bits
(2, 5)	1.22	0.77	0.52	0.39	0.47	0.22	0.11	0.06
(2, 11)	2.23	1.26	0.73	0.55	0.89	0.44	0.22	0.12
(2, 23)	4.31	2.26	1.25	0.81	1.98	0.91	0.47	0.25
(2, 31)	5.65	2.93	1.61	1.05	2.57	1.28	0.66	0.34
(2, 43)	7.82	4.04	2.17	1.15	3.65	1.84	0.94	0.50
(4, 5)	2.51	1.68	1.22	1.03	0.87	0.44	0.22	0.11
(4, 11)	4.32	2.51	1.53	1.17	1.73	0.86	0.44	0.23
(4, 23)	8.05	4.35	2.45	1.72	3.49	1.78	0.87	0.47
(4, 31)	10.47	5.59	3.09	2.09	4.96	2.45	1.25	0.67
(4, 43)	14.40	7.53	4.09	2.67	7.13	3.57	1.78	0.94
(6, 11)	6.32	3.65	2.34	1.84	2.48	1.23	0.61	0.34
(6, 23)	11.69	6.35	3.62	2.64	6.30	3.28	1.61	0.84
(6, 31)	15.18	8.11	4.52	3.14	11.51	5.85	2.92	1.47
(6, 43)	20.86	10.91	5.94	4.09	18.44	9.52	4.81	2.43
(8, 11)	8.44	4.90	3.15	2.51	3.43	1.70	0.86	0.47
(8, 23)	15.26	8.32	4.79	3.48	11.70	6.15	3.11	1.56
(8, 31)	19.83	10.61	5.98	4.17	18.49	9.44	4.67	2.43
(8, 43)	27.10	14.32	7.85	5.27	30.31	15.30	7.89	3.93

表 2、使用 GTX760 顯示卡平行時間比較

(k, n)	Encoding $P\text{E}1_{760}$				Decoding $P\text{D}2_{760}$			
	8 bits	16 bits	32 bits	64 bits	8 bits	16 bits	32 bits	64 bits
(2, 5)	1.22	0.77	0.52	0.39	0.47	0.22	0.11	0.06
(2, 11)	2.23	1.26	0.73	0.55	0.89	0.44	0.22	0.12
(2, 23)	4.31	2.26	1.25	0.81	1.98	0.91	0.47	0.25
(2, 31)	5.65	2.93	1.61	1.05	2.57	1.28	0.66	0.34
(2, 43)	7.82	4.04	2.17	1.15	3.65	1.84	0.94	0.50
(4, 5)	2.51	1.68	1.22	1.03	0.87	0.44	0.22	0.11
(4, 11)	4.32	2.51	1.53	1.17	1.73	0.86	0.44	0.23
(4, 23)	8.05	4.35	2.45	1.72	3.49	1.78	0.87	0.47
(4, 31)	10.47	5.59	3.09	2.09	4.96	2.45	1.25	0.67
(4, 43)	14.40	7.53	4.09	2.67	7.13	3.57	1.78	0.94
(6, 11)	6.32	3.65	2.34	1.84	2.48	1.23	0.61	0.34
(6, 23)	11.69	6.35	3.62	2.64	6.30	3.28	1.61	0.84
(6, 31)	15.18	8.11	4.52	3.14	11.51	5.85	2.92	1.47
(6, 43)	20.86	10.91	5.94	4.09	18.44	9.52	4.81	2.43
(8, 11)	8.44	4.90	3.15	2.51	3.43	1.70	0.86	0.47
(8, 23)	15.26	8.32	4.79	3.48	11.70	6.15	3.11	1.56
(8, 31)	19.83	10.61	5.98	4.17	18.49	9.44	4.67	2.43
(8, 43)	27.10	14.32	7.85	5.27	30.31	15.30	7.89	3.93

由表 4 得知，當 k 固定時，不管是 CPU 或是 GPU，加密時間都會隨著 n 的增長而變大，因為越大的 n 對固定的 k 會導致更多的加密計算。在這些實驗中，對於不同的 (k, n)，$P\text{E}1_{760}$ 與 $P\text{E}1_{\text{X}}$ 分別比 $S\text{E}1$ 快了約 2-6 以及 2-15 倍。以 $(k, n) = (8, 11)$ 為例，$S\text{E}1$ 比 $P\text{E}1_{760}$ 慢 6.14 倍，比 $P\text{E}1_{\text{X}}$ 則是慢了 14.89 倍；此外 $P\text{E}1_{\text{X}}$ 比 $P\text{E}1_{760}$ 快了約 2.4 倍。至於解密的部分，我們將參數 t 設為 k，當 k 固定時，n 變大，解密時間也會增加；而當 n 固定時，解密時間也是會隨著 k 增長而增加。在解密的實驗中，不同 (k, n) 情況下，$P\text{D}2_{760}$ 大約比 $S\text{D}2$ 快 2-9 倍，而 $P\text{D}2_{\text{X}}$ 則是比 $S\text{D}2$ 快了約 4-42 倍。以 $(k, n) = (6, 43)$ 來說，$S\text{D}2$ 比 $P\text{D}2_{760}$ 慢 7.33 倍，比 $P\text{D}2_{\text{X}}$ 則是慢了 37.03 倍，而 $P\text{D}2_{\text{X}}$ 比 $P\text{D}2_{760}$ 快了約 5.1 倍。

6. 結語

我們的演算法在平行平台上比在循序上效能更高，平行計算的成果具有吸引力與重要性—代表在實際應用中，機密分享是可行的。

表 3、使用 Titan X 顯示卡平行時間比較

(k, n)	Encoding $P\text{E}1_{\text{X}}$				Decoding $P\text{D}2_{\text{X}}$			
	8 bits	16 bits	32 bits	64 bits	8 bits	16 bits	32 bits	64 bits
(2, 5)	0.13	0.06	0.10	0.11	0.02	0.02	0.01	0.01
(2, 11)	0.19	0.09	0.15	0.16	0.10	0.06	0.04	0.02
(2, 23)	0.39	0.19	0.28	0.21	0.34	0.20	0.10	0.06
(2, 31)	0.56	0.25	0.39	0.40	0.29	0.18	0.08	0.06
(2, 43)	0.87	0.38	0.57	0.59	0.27	0.14	0.08	0.04
(4, 5)	0.14	0.07	0.10	0.11	0.04	0.03	0.03	0.02
(4, 11)	0.20	0.09	0.15	0.17	0.17	0.11	0.06	0.04
(4, 23)	0.38	0.19	0.28	0.30	0.34	0.17	0.11	0.06
(4, 31)	0.56	0.25	0.39	0.41	0.37	0.18	0.12	0.07
(4, 43)	0.87	0.39	0.58	0.59	0.28	0.18	0.10	0.06
(6, 11)	0.24	0.11	0.15	0.17	0.25	0.14	0.08	0.05
(6, 23)	0.39	0.19	0.28	0.30	0.39	0.23	0.13	0.08
(6, 31)	0.59	0.26	0.39	0.41	0.37	0.23	0.14	0.08
(6, 43)	0.89	0.38	0.58	0.59	0.30	0.18	0.12	0.07
(8, 11)	0.24	0.11	0.15	0.17	0.35	0.20	0.12	0.07
(8, 23)	0.48	0.20	0.29	0.30	0.38	0.23	0.15	0.09
(8, 31)	0.64	0.23	0.39	0.41	0.39	0.22	0.16	0.09
(8, 43)	0.93	0.39	0.58	0.59	0.60	0.34	0.22	0.09

表 4、循序與平行演算法時間與比較

(k, n)	Encoding (64 bits)					Decoding (64 bits)				
	S(E1)	$PE1_{760}$	$PE1_X$	S(E4)/$PE4_{760}$	S(E4)/$PE4_X$	S(D2)	$PD2_{760}$	$PD2_X$	S(D2)/$PD2_{760}$	S(D2)/$PD2_X$
(2, 5)	0.39	0.21	0.11	1.90	3.42	0.06	0.04	0.01	1.75	4.20
(2, 11)	0.55	0.26	0.16	2.08	3.31	0.12	0.04	0.02	3.26	5.06
(2, 23)	0.81	0.50	0.21	1.62	3.94	0.25	0.05	0.06	5.56	4.47
(2, 31)	1.05	0.65	0.40	1.60	2.58	0.34	0.06	0.06	5.36	6.10
(2, 43)	1.15	0.96	0.59	1.19	1.95	0.50	0.09	0.04	5.31	12.09
(4, 5)	1.03	0.21	0.11	4.90	9.17	0.11	0.07	0.02	1.68	4.68
(4, 11)	1.17	0.28	0.17	4.25	6.98	0.23	0.07	0.04	3.39	5.71
(4, 23)	1.72	0.51	0.30	3.38	5.76	0.47	0.11	0.06	4.11	7.38
(4, 31)	2.09	0.66	0.41	3.18	5.13	0.67	0.16	0.07	4.16	9.88
(4, 43)	2.67	0.97	0.59	2.75	4.51	0.94	0.22	0.06	4.22	14.47
(6, 11)	1.84	0.30	0.17	6.14	10.99	0.34	0.10	0.05	3.44	6.38
(6, 23)	2.64	0.53	0.30	4.98	8.83	0.84	0.17	0.08	4.87	11.02
(6, 31)	3.14	0.74	0.41	4.23	7.67	1.47	0.25	0.08	5.94	18.23
(6, 43)	4.09	1.05	0.59	3.91	6.90	2.43	0.33	0.07	7.33	37.03
(8, 11)	2.51	0.41	0.17	6.14	14.89	0.47	0.13	0.07	3.52	6.36
(8, 23)	3.48	0.62	0.30	5.66	11.59	1.56	0.22	0.09	7.09	17.90
(8, 31)	4.17	0.78	0.41	5.31	10.13	2.43	0.32	0.09	7.70	28.50
(8, 43)	5.27	1.06	0.59	4.99	8.88	3.93	0.44	0.09	8.89	41.52

以個人電腦而言，我們循序平台上的 i7 CPU 和平行平台上的 Titan X 顯示卡，屬於規格稍高的產品，但其成本與電腦叢集 (PC clusters)、甚至於超級電腦 (supercomputers) 相比，會低許多；就連 GTX 760 此較舊的顯示卡（與 Titan X 相比）都可達成優於 i7 CPU 的效能。活用 Titan X 或 GTX 760 皆可在成本效益考量下，實現機密分享的平行計算。

因硬體設備的不同，不同的參數，也將會有不同的結果，未來將進行更多的實驗，針對 CUDA 中所需的 blocknum、threadnum 此兩個參數，觀察實驗參數間的關係，讓加速效率更為優良。

References

[1] Shamir, A., “How to share a secret”, *Communications of the ACM*, vol.22, no.11, 612—613, *1979*.

[2] J. Kurihara, S. Kiyomoto, K. Fukushima, and T. Tanaka, On a Fast (k, n)-Threshold Secret Sharing Scheme, *IEICE TRANS. FUNDAMENTALS*, VOL.E91–A, *2008*.

[3] Nvidia, Cuda GPUs, https://developer.nvidia.com/cuda-gpus

[4] Nvidia, Cuda Zone, https://developer.nvidia.com/cuda-zone

A Survey on the Algorithms of the Edit Distance Problem, the Genome Rearrangement Problem and Related Variants*

Shian-Liang Lin[a], Chiou-Ting Tseng[b] and Chang-Biau Yang[a†]

[a]Department of Computer Science and Engineering
National Sun Yat-sen University, Kaohsiung, Taiwan
[b]Air Navigation and Weather Services, Civil Aeronautics Administration
Ministry of Transportation and Communications, Taiwan

Abstract

The edit distance problem has been studied for several decades. Given sequences (strings) A and B with length m and n, respectively, $m \leq n$, the edit distance problem is to find the minimum cost of operations required to transform A into B. According to different models of cost functions, operations and input sequences, the problem has several variants. The edit distance on run-length encoding strings and cyclic strings are the variants on the input aspect. The edit distance considering consecutive insertions and deletions is a variant on the cost function. The block edit problem is a variant on the operation aspect. Besides, the genome rearrangement problem can also be viewed as a variant, whose operations include inversions, reversals and transpositions. In this paper, we survey some algorithms for the edit distance problem, its variants and the genome rearrangement problem.

1 Introduction

The sequence similarity has been studied for several decades and many algorithms have been developed for various applications. For example, in biological area, proteins or genomes can be represented by a sequence, and the similarities of sequences can be viewed as the relations between proteins or genomes. In 1970, Needleman and Wunsh [49] first proposed the concept of *sequence similarity* computation of two amino acid sequences, and they presented a primitive algorithm with $O(m^2n)$ time for solving the problem.

Following the same concept, in 1974, Seller [52, 53] presented an improved algorithm with $O(mn)$ time. In the same year, Wangner and Fisher [62] defined a simple version of the *edit distance* problem, including three operations: character *insertion*, *deletion* and *replacement*. They used the *dynamic programming* (DP) approach to solve the problem with $O(mn)$ time, where m and n denote the lengths of two input strings (sequences). Based on the DP approach, many algorithms and variants of this problem have been proposed later. Lowrance and Wagner [42] added a new operation, character exchange, to this problem. Their algorithm is still of $O(mn)$ time. Masek and Peterson [45] proposed an $O(n^2/\log n)$-time algorithm with the four Russians' technique.

With mapping to the *shortest edit script* (SES) problem on the edit graph, equivalent to the *longest common subsequence* (LCS) problem, the diagonal method with $O(nd)$ time was proposed by Myers [48] and $O(np)$ time algorithm by Wu *et al.* [68], where d denotes the edit distance of the two input sequences and the value of p is about a half of d. On the other hand, the variants of the edit distance problem have been studied, such as edit distance for *cyclic strings* [43, 44], edit distance for *run length encoded* (RLE) strings [4, 6, 12, 35, 39] and *block edit distance* [3, 21, 41, 47, 54, 55, 60].

The edit distance originally defined by Wangner and Fisher [62] only considers the cost on single character operations. For example, the cost of two consecutive deletions is twice of a single deletion. The edit distances with considering consecutive insertions/deletions have also been studied by several researchers [20, 24, 46, 57, 64, 65].

Furthermore, in the genome rearrangement problem, the operations are performed on a seg-

*This research work was partially supported by the Ministry of Science and Technology of Taiwan under contract MOST 104-2221-E-110-018-MY3.

†Corresponding author (Chang-Biau Yang). E-mail: cbyang@cse.nsysu.edu.tw .

ment of sequence (substrings), including reversal (reversing the substring), inverse (reversing the substring, and then substituting each character by its complement in DNA), transposition (exchanging two consecutive substrings). Since the problem with overlapping operations is NP-hard, some approximation algorithms were proposed [7, 17, 33, 63]. Then, with the non-overlapping restriction on operations, some polynomial-time algorithms have also been designed [30, 51, 59].

In this paper, we survey several papers discussing the edit distance problem and the genome rearrangement problem. We use some simple examples to explain the key points or main ideas in these algorithms. Besides, we discuss the evolution of these algorithms, and analyze the difference of these algorithms.

The rest of this paper is organized as follows. In Section 2, we introduce the background knowledge and list the time complexities of the algorithms for a summary of the surveyed papers. In Section 3, we survey some algorithms of the edit distance problem and its variants. In Section 4, we survey some genome rearrangement algorithms with operations performed on a substring. In Section 5, we give the conclusion of this paper.

2 Preliminaries

2.1 Longest Common Subsequence

Given two sequences (strings) $A = a_1a_2a_3\cdots a_m$ and $B = b_1b_2b_3\cdots b_n$, the *longest common subsequence* (LCS) problem (of two sequences) is that of finding the longest common part of A and B by deleting zero or more characters from A and B. For example, suppose that we are given $A =$ `acaagc` and $B =$`atcagtc`. The the answer of the LCS is `acagc`, whose length is 5.

The LCS problem was first presented by Needleman and Wunsch in 1970 [49]. Their purpose is to solve the alignment of biological sequences, composed of DNA, RNA or amino acids of proteins. They proposed a brute-force method to solve the alignment problem with $O(mn(m+n))$ time.

The well-known *dynamic programming* (DP) formula for solving the LCS problem was proposed by Hirschberg in 1975. He rewrote the DP method for the edit distance problem, proposed by Wagner and Fischer [62], in Equation 1 [29], where $M[i,j]$ denotes the LCS length of $A_{1..i}$ and $B_{1..j}$. Here,

	-	a	t	c	a	g	t	c
-	0	0	0	0	0	0	0	0
a	0	1	1	1	1	1	1	1
c	0	1	1	2	2	2	2	2
a	0	1	1	2	3	3	3	3
a	0	1	1	2	3	3	3	3
g	0	1	1	2	3	4	4	4
c	0	1	1	2	3	4	4	5

Figure 1: The DP lattice for LCS with $A =$ `acaagc` and $B =$`atcagtc`, where the LCS answer is `acagc`.

$A_{i..j}$ denotes the substring of A from position indices i to j.

$$M[i,j] = \max \begin{cases} 0 & \text{if } i=0 \text{ or } j=0; \\ M[i-1,j-1]+1 & \text{if } a_i = b_j; \\ M[i-1,j] & \text{if } a_i \neq b_j; \\ M[i,j-1] & \end{cases} \tag{1}$$

For example, the DP lattice for LCS with $A =$ `acaagc` and $B =$`atcagtc` is shown in Figure 1. Obviously, the time complexity is $O(mn)$.

2.2 Edit Distance

The *edit distance* problem is to find a series of edit operations with the minimum cost to transform sequence (string) A into sequence B. It was first defined by Wagner and Fischer in 1974 [62]. The edit operations include character insertion, character deletion and character replacement, with cost $INS(b_j)$, $DEL(a_i)$ and $REP(a_i, b_j)$, respectively. Let $M_{wf}[i,j]$ denote the minimum cost to transform $A_{1..i}$ into $B_{1..j}$. The DP formula proposed by Wagner and Fischer [62] is presented in Equation 2.

$$M_{wf}[i,j] = \min \begin{cases} \text{if } a_i = b_j: & \\ \quad M_{wf}[i-1,j-1] & \\ \text{if } a_i \neq b_j: & \\ \quad M_{wf}[i-1,j-1] + REP(a_i,b_j) & \\ \quad M_{wf}[i-1,j] + DEL(a_i) & \\ \quad M_{wf}[i,j-1] + INS(b_j) & \end{cases} \tag{2}$$

The time complexity of the above edit distance algorithm is $O(mn)$. An example of the process for calculating the edit distance is shown in the DP lattice of Figure 2, with the cost for each character insertion, deletion and replacement being 1, 1 and 2, respectively. The LCS length can be got by Equation 3 with this cost assignment, where L denotes the LCS length and d denotes the edit distance. In the example, the LCS length $5 = \frac{6+7-3}{2}$.

$$L = \frac{m+n-d}{2}. \tag{3}$$

	-	a	t	c	a	g	t	c
-	0	1	2	3	4	5	6	7
a	1	0	1	2	3	4	5	6
c	2	1	2	1	2	3	4	5
a	3	2	3	2	1	2	3	4
a	4	3	4	3	2	3	4	5
g	5	4	5	4	3	2	3	4
c	6	5	6	5	4	3	4	3

Figure 2: The DP lattice for calculating the edit distance of $A =$ `acaagc` and $B =$`atcagtc` with cost functions $DEL(a_i) = 1$, $INS(b_j) = 1$ and $REP(a_i, b_j) = 2$

	-	a	t	c	a	g	t	c
-	0	1	2	3	4	5	6	7
a	1	0	1	2	3	4	5	6
c	2	1	1	1	2	3	4	5
a	3	2	2	2	1	2	3	4
a	4	3	3	3	2	2	3	4
g	5	4	4	4	3	2	3	4
c	6	5	5	5	4	3	3	3

Figure 3: The DP lattice for calculating the edit distance of $A =$ `acaagc` and $B =$`atcagtc` with cost functions $DEL(a_i) = 1$, $INS(b_j) = 1$ and $REP(a_i, b_j) = 1$.

One may define variously allowed edit operations and various cost of each operation. For example, suppose the cost for each character insertion, deletion and replacement is defined to be 1, 1 and 1, respectively. Then, the DP lattice for calculating the edit distance with the same example is shown in Figure 3.

In 1974, Sellers [52, 53] also proposed an algorithm for solving the edit distance problem in $O(mn)$ time. His algorithm is based on the concept given by Needleman and Wunsch [49]. Furthermore, Sellers gave a formal definition of *evolutionary distance*, in which the cost of each operation on different characters may be different.

Later on, the edit distance problem with character operations was extended to allow the block operation [3, 21, 41, 47, 54, 55, 60], including block move, block copy, block deletion and block reversal.

2.3 Alignment

Given two sequences (strings) A and B, the alignment is a way for presenting how to transform A into B. For example, given $A =$ `acaagc` and $B =$ `atcagtc`, one of the possible alignments is shown as follows.

$$\begin{array}{ll} A\colon & \texttt{a}\ -\ \texttt{c}\ \texttt{a}\ \texttt{a}\ \texttt{g}\ -\ \texttt{c} \\ B\colon & \texttt{a}\ \texttt{t}\ \texttt{c}\ \texttt{a}\ -\ \texttt{g}\ \texttt{t}\ \texttt{c} \end{array}$$

As one can see, the above alignment corresponds to Figures 1 and 2. Eevery two corresponding positions of A and B form an aligned pair, such as $(a_1, b_1) =$ (`a, a`), $(-, b_2) =$ (`-, t`) and $(a_4, -) =$ (`a, -`). Here, each minus sign $-$ represents a gap in an alignment. In an aligned pair $(-, b_j)$, the gap in A represents that character b_j is inserted into A at the position. Similarly, the gap in $(a_i, -)$ represents the deletion of character a_i. The edit distance of each alignment can be easily calculated. Note that different alignments may have the same edit distance. In some biology applications, the gaps is hoped to appear consecutively, which means the minimization of biological mutations. Some variations with linear, concave, convex or general cost functions for finding the best alignment were also studied [20, 24, 46, 64, 65].

2.4 Run-length Encoding

Run-length encoding (RLE) is a simple method to compress data in a lossless way. For a string, RLE compresses the data according to the symbol and the counts of consecutive appearances. For example, $A =$`aaabbbcc` can be represented as $A = a^3b^3c^2$. In an RLE string, each substring formed by an identical symbol is defined as a *run*, such as a^3, b^3 and c^2 in A. The edit distance problem on RLE strings is a variant of the edit distance problem. Several algorithms for solving this problem usually apply DP based on runs, instead of individual symbols [3, 4, 6, 12, 35, 39].

2.5 Summary of the Surveyed Results

We list the time complexities of the algorithms for solving the edit distance problem and its variants in Table 1. The keyword field in the table contains the key points or techniques for describing the algorithms or problems abstractly. Most algorithms are based on the DP lattice and their time complexities $O(mn)$ depend deeply on the size of the DP lattice.

As shown in Table 1, when the character insertions or deletions are consecutive, the algorithms are still efficient. Some variants can be seen in the table, including cyclic strings, RLE strings, and block edit operations.

Table 1: The algorithms for the edit distance problem. Notations: R_i: cost of each insertion or deletion is 1, each replacement is i; $|S|$: number of candidates considered in each DP cell; s: number of alternations in the mixed cost function of concave or convex; $\alpha(\ )$: inverse Ackermann function; m_a, n_b: numbers of runs in strings A and B, respectively; p_1: number of elements on the bottom boundary of a matched block.

Year	Author(s)	Time complexity	Keywords
Traditional edit distance (INS,DEL,REP)			
1974	Wagner and Fischer [62]	$O(mn)$	DP
1974	Sellers [52, 53]	$O(mn)$	DP
1975	Lowrance and Wagner [42]	$O(mn)$	DP, interchange
1980	Masek and Peterson [45]	$O(n^2/\log n)$	DP, Four Russians
1986	Myers [48]	$O(nd)$	shortest edit script, R_2
1990	Wu [68]	$O(np)$	shortest edit script, R_2
2002	Jiang *et al.* [31]	$O(mn^3)$	DP, arc, RNA structure
Consecutive Insertions/Deletions			
1976	Waterman and Smith [65]	$O(mn^2)$	DP
1981	Smith and Waterman [57]	$O(mn^2)$	DP, local alignment
1982	Gotoh [24]	$O(mn)$	DP, linear cost function
1984	Waterman [64]	$O(mn\|S\|), \|S\| \leq n$	DP, concave cost function
1988	Miller and Myers [46]	$O(mn \log n)$	DP, concave cost function, curve line, binary search
1990	Eppstein [20]	$O(n^2 \cdot \alpha(n/s))$	DP, concave, convex and mixed functions, monotone
Edit distance for cyclic strings			
1990	Maes [43]	$O(mn \log m)$	DP, divide and conquer
2000	Marzal and Barrachina [44]	$O(mn \log m)$	branch and bound, R_1
Edit distance for RLE strings			
1993	Bunke and Csirik [12, 13]	$O(n_b m + m_a n)$	subdivision, DP, R_2
2002	Arbell *et al.* [6]	$O(n_b m + m_a n)$	subdivision, DP, R_1
2007	Liu *et al.* [39]	$O(\min(n_b m, m_a n))$	subdivision, DP, R_1
2008	Ann *et al.* [4]	$O(m_a n_b + p_1)$	range min/max query R_2, LCS
Block edit distance			
2010	Ann *et al.* [3]	$O(mn)$, $O(mn \log n)$, $O(mn^2)$	block edit, cost measure, non-overlapping, DP, suffix tree

Table 2 shows the algorithms for genome rearrangement. Since the general genome rearrangement problems with overlapping operations are NP-hard, some restrictions are made, such as non-overlapping operations.

3 Edit Distance

3.1 Algorithm by Lowrance and Wagner

In 1975, Lowrance and Wagner [42] proposed an extension to the edit distance problem with one more operation, interchange, which exchanges two adjacent elements in the same sequence. Under some specific cost assignments, the problem can still be solved in $O(mn)$ time. In the same year, Wagner [61] further analyzed the complexities of other cost assignments. His conclusion is that some are NP-Complete, while some are solvable with polynomial-time algorithms. Furthermore, in 1992, Schoniger and Waterman [51] presented an extension to the edit distance problem with additional operation, inversion (inverting a substring), which can be viewed as a generalization of the interchange operation.

Table 2: The algorithms for genome rearrangement.

Year	Author(s)	Time complexity	Keywords
1992	Schoniger and Waterman [51]	$O(n^6)$	inversion, non-overlapping
1993	Keceioglu and Sankoff [33]	2-approximation $O(n^2)$	reversal, permutation, overlapping, break-point graph, NP-hard
1996	Bafna *et al.* [33]	$\frac{7}{4}$-approximation $O(n^2)$	reversal, permutation, overlapping, break-point graph, NP-hard
2000	Walter *et al.* [63]	$\frac{9}{4}$-approximation $O(n^2)$	transposition, permutation, overlapping, break-point graph, NP-hard
2016	Ta *et al.* [59]	$O(n^3)$	inversion, transposition, non-overlapping, mutation fragment
2017	Hsu *et al.* [30]	$O(n^2)$	inversion, transposition, non-overlapping, repetition, run

3.2 Time Bounds by Wong and Chandra

In 1976, Wong and Chandra [67] proved the bounds on the time complexity for the edit distance problem. Suppose that the cost of each character insertion, deletion and replacement is denoted INS, DEL and REP, respectively. The only decision is equal or unequal between two symbols from A or B. Let $v = REP/(INS + DEL)$. The lower bound of the number of required comparisons is $m(n-m)+vm^2-1/v+1$. When $v = 1$, which is equivalent to the LCS problem, the lower bound becomes mn. This is the same as the result of Aho *et al.* [2]. When $INS = DEL = REP$ and $v = 0.5$, the lower bound is $mn - m^2/2 - 1$. The upper bound for the number of comparisons is $mn - \lfloor m(1-v)\rfloor \times \lfloor m(1-v)+1\rfloor$. When $v = 1$, which is equivalent to the LCS problem, the upper bound becomes mn. When $INS = DEL = REP$ and $v = 0.5$, the upper bound is $mn-(m^2-4m)/4$.

As a result, when $INS = DEL = REP$, it may be solved with a more efficient algorithm than that with $INS = DEL = 1$ and $REP = 2$ (equivalent to LCS).

3.3 Four Russians' Technique by Masek and Paterson

In 1980, Masek and Paterson [45] proposed an improved algorithm for solving the edit distance problem. The algorithm applies the four Russians' technique to split the lattice matrix into several $k \times k$ submatrices and to store some of the *step* values to speed up the computation. The *step*, computed according to Theorem 1, means the difference between two adjacent cells in the same row or column.

Theorem 1. *[45] Let M_{mp} be the edit lattice matrix. The step (the difference between two adjacent entries) can be computed by*

$$M_{mp}[i,j] - M_{mp}[i-1,j] = min \begin{cases} REP(a_i,b_j) \\ \quad -(M_{mp}[i-1,j] - M_{mp}[i-1,j-1]), \\ DEL(a_i), \\ INS(b_j) \\ \quad +(M_{mp}[i,j-1] - M_{mp}[i-1,j-1]) \\ \quad -(M_{mp}[i-1,j] - M_{mp}[i-1,j-1]), \end{cases}$$

$$M_{mp}[i,j] - M_{mp}[i,j-1] = min \begin{cases} REP(a_i,b_j) \\ \quad -(M_{mp}[i,j-1] - M_{mp}[i-1,j-1]), \\ DEL(a_i) \\ \quad +(M_{mp}[i-1,j] - M_{mp}[i-1,j-1]) \\ \quad -(M_{mp}[i,j-1] - M_{mp}[i-1,j-1]), \\ INS(b_j), \end{cases}$$

where INS, DEL and REP denote the cost of each character insertion, deletion and replacement, respectively.

For example, consider $A =$ `acaagc` and $B =$ `atcagtc` with $k = 4$. Figure 4 shows the concept of computation process. Only the cells on the lower boundary and the right boundary of each $k \times k$ submatrix are calculated. And the values of one submatrix boundary can be obtained from its left submatrix and upper submatrix by the table lookup scheme with the two corresponding substrings as the searching index.

To reduce the amount of precomputed lookup tables, the step concept (Theorem 1) is applied to build the step table Sp, as shown in Figure 5. The left of each cell records $M_{mp}[i,j] - M_{mp}[i-1,j]$ and the right records $M_{mp}[i,j] - M_{mp}[i,j-1]$. For example, $Sp[1,4]$ stores the value $M_{mp}[1,4] - M_{mp}[0,4] = -1$, and $Sp[4,2]$ stores $M_{mp}[4,2] - M_{mp}[4,1] = 0$. The edit distance of each cell on

	-	a	t	c	a	g	t	c	ϕ
-	0	1	2	3	4	5	6	7	8
a	1				3				7
c	2				2				6
a	3				1				5
a	4	3	4	3	2	3	4	5	6
g	5				3				5
c	6				4				4
ϕ	7				5				3
ϕ	8	7	8	7	6	5	6	5	4

Figure 4: An example of edit matrix M_{mp} for computing the edit distance, where $A =$ `acaagc`, $B =$ `atcagtc`, $k = 4$ and ϕ denotes the dummy character to make m and n be multiples of k. Here, $DEL(a_i) = 1$, $INS(b_j) = 1$ and $REP(a_i, b_j) = 2$.

the rightmost column and the bottom row can be reconstruced with the step table.

For a submatrix, given the two substrings of length k with steps in the leftmost column and top row, the algorithm builds the resulting steps of the rightmost column and the bottom row. All possible $k \times k$ submatrices can be precomputed and the resulting steps are stored. As a result, after $O(|\Sigma|^k k^2 \log k)$-time preprocessing, the algorithm needs only $O((m/k) \times (n/k) \times (\log n))$ time to split the $m \times n$ edit matrix into mn/k^2 submatrices and needs $O(k + \log n)$ time to fetch the precomputed steps to compute the edit distance. In addition to the preprocessing time $O(|\Sigma|^k k^2 \log k)$, the algorithm requires $O(mn/\log n)$ time when $k = \lfloor \log n \rfloor$, and requires $O(n)$ time when $k > m$.

3.4 The Diagonal Method by Myers

In 1986, Myers [48] proposed a *diagonal method* for solving the edit distance problem with time complexity $O(nd)$, where d denotes the edit distance between the two input sequences and $m \leq n$. Here, the costs of each insertion, deletion and replacement are assumed to be 1, 1, and 2, respectively. It is very efficient if the distance is very small, that is, the two input sequences are very similar.

The main concept is to calculate the furthest contours of distance $0, 1, 2, \cdots, d$, sequentially. On each diagonal line k, consisting of all cells (i, j) in the DP lattice with $k = j - i$, the furthest cell achieving distance d', $0 \leq d' \leq d$, is maintained.

An example of the calculated lattice is shown in Table 3. For round 0, the cells with distance 0 are computed. For round 1, the cells of distance 0 are extended to compute distance 1. The furthest contour with distance 1 consists of $(3, 4)$ on diagonal line 1 and $(4, 3)$ on diagonal line -1. For round 2, only $(3, 4)$ and $(4, 3)$ are extended to compute distance 2. Some cells with distance 2 are not calculated, such as $(2, 0), (3, 1)$ and $(4, 2)$. For round 3, when diagonal -1 is to be extended, there are two possible starting cells, $(6, 5)$ from $(6, 4)$ on diagonal -2, or $(5, 4)$ from $(4, 4)$ on diagonal 0. $(6, 5)$ is selected as the starting cell, since $(6, 5)$ is further than $(5, 4)$.

Table 3: The calculation lattice for Myers' algorithm with $A =$ `cadaadaccb` and $B =$`cdaddcabccbd`. Here, the numbers with underlines are the furthest (lowest right) cells on each diagonal k with the same distance and the cells with Italic and bold are really traced in the algorithm.

		0	1	2	3	4	5	6	7	8	9	10	11	12
		-	*c*	*d*	*a*	*d*	*d*	*c*	*a*	*b*	*c*	*c*	*b*	*d*
0	-	***0***	1	2	3	4	5							
1	*c*	1	***0***	***1***	2	3	4	5						
2	*a*	2	***1***	2	***1***	2	3	4	5					
3	*d*	3	2	***1***	2	***1***	***2***	***3***	4	5				
4	*a*	4	3	2	***1***	***2***	***3***	4	***3***	***4***	***5***			
5	*a*	5	4	3	***2***	3	4	5	***4***	***5***				
6	*d*	6	5	4	3	***2***	***3***	***4***	5	6				
7	*a*		6	5	4	***3***	***4***	5	***4***	***5***	6			
8	*c*			6	5	***4***	5	***4***	***5***	6	***5***	6		
9	*c*				6	***5***	6	***5***	***6***		6	***5***	6	
10	*b*					***6***		***6***		***6***		***6***	***5***	***6***

Since the minimum distance on diagonal k is $|k|$, the range of diagonals required to be searched is $\{-d, -d+1, \cdots, d-1, d\}$. When the calculation process touches cell (m, n), the algorithm terminates and the edit distance is obtained.

3.5 The Diagonal Method by Wu *et al.*

In 1990, Wu *et al.* [68] proposed another diagonal method with time complexity $O(np)$, where p is the number of deletions in the edit operations. Their algorithm is nearly twice as fast as Myers' algorithm [48].

Table 4 shows the p values in the calculation lattice of Wu *et al.* with the same inputs of Table 3. Let $\triangle$ denote the diagonal passing through cell (m, n), where $\triangle = n - m$ and it is assumed $m \leq n$. Wu *et al.* found the following equality

$$d = \triangle + 2p. \tag{4}$$

	-	a	t	c	a	g	t	c	ϕ
-	-,-	-,1	-,1	-,1	-,1	-,1	-,1	-,1	-,1
a	1,-				-1,-				-1,-
c	1,-				-1,-				-1,-
a	1,-				-1,-				-1,-
a	1,-	-,-1	-,1	-,-1	1,-1	-,1	-,1	-,1	1,1
g	1,-				1,-				-1,-
c	1,-				1,-				-1,-
ϕ	1,-				1,-				-1,-
ϕ	1,-	-,-1	-,1	-,-1	1,-1	-,-1	-,1	-,-1	1,-1

Figure 5: The step table Sp, where in each cell, the left is $M_{mp}[i,j] - M_{mp}[i-1,j]$ and the right is $M_{mp}[i,j] - M_{mp}[i,j-1]$. The symbol '$-$' means that it is not calculated.

Table 4: The p values (numbers of deletions) for the algorithm of Wu *et al.* with $A =$ `cadaadaccb` and $B =$ `cdaddcabccbd`. Here, the numbers with underline are the furthest (lowest right) cells on each diagonal with the same p value and the cells with Italic and bold are really traced in the algorithm.

		0	1	2	3	4	5	6	7	8	9	10	11	12
		-	*c*	*d*	*a*	*d*	*d*	*c*	*a*	*b*	*c*	*c*	*b*	*d*
0	-	***0***	0	0	1	2								
1	*c*	1	***0***	***0***	0	1	2							
2	*a*	2	***1***	1	***0***	0	1	2						
3	*d*		2	***1***	1	***0***	***0***	***1***	2					
4	*a*			2	***1***	***1***	***1***	1	***1***	***2***				
5	*a*				***2***	2	2	2	***1***	***2***				
6	*d*					***2***	***2***	***2***	2	2				
7	*a*								***2***	***2***	2			
8	*c*										***2***	2		
9	*c*											***2***	2	
10	*b*												***2***	***2***

For example, in Table 4, $p = 2$ and $\triangle = n - m = 12 - 10 = 2$. In Table 3, $d = 6$. Therfore, $d = \triangle + 2p = 6 = 2 + 2 \times 2$. In other words, d can be calculated from p.

When the algorithm is executed, only diagonals $\{-p, -p+1, \cdots, \triangle, \triangle+1, \cdots, \triangle+p\}$ (totally $\triangle + 2p+1$ diagonals) are searched, instead of $\{-d, -d+1, \cdots, d-1, d\}$ (totally $2d+1$ diagonals). Thus, the required time of Wu *et al.* is about a half of of Myers asymptotically.

The main spirit of the algorithm of Wu *et al.* is to count only character deletions (p values), not character insertions. The DP formula for the algorithm of Wu *et al.* can be rewritten as follows when (i,j) is on diagonal $k = j - i \leq \triangle - 1$.

$$M_{wu}[i,j] = \min \begin{cases} M_{wu}[i-1,j-1] & \text{if } a_i = b_j, \\ M_{wu}[i,j-1], & \text{// insertion} \\ M_{wu}[i-1,j]+1. & \text{// deletion} \end{cases} \tag{5}$$

When (i,j) is on diagonal $k = j - i \geq \triangle$, the DP formula is rewritten as follows.

$$M_{wu}[i,j] = \min \begin{cases} M_{wu}[i-1,j-1] & \text{if } a_i = b_j, \\ M_{wu}[i,j-1]+1, & \text{// insertion,} \\ \quad \text{// but need one more deletion} \\ M_{wu}[i-1,j]. & \text{// deletion,} \\ \quad \text{// number of deletion has} \\ \quad \text{// been counted on insertion} \end{cases} \tag{6}$$

Applying the concept of Myers' algorithm, the algorithm incrementally test the p value by finding the furthest reaching cell on diagonal k. For round p, diagonals $-p, -p+1, \cdots, \triangle - 1$ are first updated sequentially. Then diagonals $\triangle + p, \triangle + p - 1, \cdots, \triangle$ are updated sequentially. The algorithm stops when cell (m,n) is reached.

See Table 4. For example, in round $p = 2$, the starting cell on diagonal -2 is $(5,3)$. Then, it can be extended to $(6,4)$ on the same diagonal. However it cannot be extended any more. Thus, it is directed to $(6,5)$ on diagonal -1, and then to $(6,6)$, $(7,7)$ on diagonal 0. Finally, it reaches the furthest cell $(10,11)$ on diagonal 1.

3.6 Consecutive Insertions and Deletions by Waterman *et al.*

In 1976, Waterman *et al.* [65] presented a more general form for edit distance. They not only considered the number of used operations but also the state of alignment. For example, given $A =$ `aacc` and $B =$ `ac`, the edit distance of the alignment of $A =$ `aacc` to $B =$ `-ac-` is different from that of $A =$ `aacc` to $B =$ `a--c`. The former alignment

	-	a	c
-	0	1	2
a	1	0	1
a	2	1	1
c	3	2	1
c	4	3	2

(a)

	-	a	c
-	0	1	1.1
a	1	0	1
a	1.1	1	1
c	2.1	1.1	1
c	2.2	2.1	1.1

(b)

Figure 6: An example of the DP lattice M_{wa} for the edit distance with two different cost functions, where $A =$ `aacc` and $B =$ `ac`. (a) The lattice that the cost of each single insertion, deletion or replacement is 1. (b) The lattice that the cost of each single insertion, deletion or replacement is 1, and the cost of each double-insertion or double-deletion is 1.1.

contains two deletions of length 1, while the latter involves only a deletion of length 2.

They considered the length of consecutive insertions/deletions as a factor of cost. For example, see Figures 6. In the first cost function, the cost of each single insertion, deletion or replacement is 1; in the second cost function, the cost of each single insertion, deletion or replacement is 1, and the cost of each double-insertion (two consecutive insertions) or each double-deletion (two consecutive deletions) is 1.1.

Different cost functions of insertions/deletions with different lengths may be more accurate to get the desired alignment. If we want to get more consecutive insertions/deletions, we can decrease the cost per insertion/deletion as the length increases. The DP formula for considering consecutive insertions/deletions is given in Equation 7 [65], where $\delta(k)$ is the cost of a consecutive insertion/deletion of length k.

$$M_{wa}[i,j] = \min \begin{cases} M_{wa}[i-1,j-1] + REP(a_i,b_j) \\ Ins[i,j] \\ Del[i,j], \end{cases}$$

where

$$Ins[i,j] = \min_{1\le k\le j}\{M_{wa}[i,j-k] + \delta(k)\},$$
$$Del[i,j] = \min_{1\le k\le i}\{M_{wa}[i-k,j] + \delta(k)\}. \tag{7}$$

The time complexity of the above algorithm is $O(mn^2)$, since each cell should considers $O(n)$ cases of consecutive insertions and consecutive deletions, and the size of the DP lattice is $O(mn)$.

3.7 Relation between Edit Distance and Similarity by Smith *et al.*

In 1981, Smith *et al.* [56–58] formally defined the equation for the similarity and distance of two given sequences. An alignment Λ can be represented by recording the aligned pairs. For example, suppose $A = a_1a_2\ldots a_8$ and $B = b_1b_2\ldots b_6$. An alignment $\Lambda = \{(a_1,b_1),(a_2,b_4),(a_6,b_5),(a_7,b_6)\}$ mean that $A_{3..5}$, a_8 and $B_{2..3}$ are unmatched. The unmatched substrings are called *gaps* in an alignment. The unmatched substrings of A are called deletions and the unmatched substrings of B are called insertions.

Let $s(a_i,b_j)$ be the score of aligning a_i with b_j, $REP(a_i,b_j)$ be the distance between a_i and b_j, $\delta(k)$ be the cost for a consecutive insertion/deletion of length k. The total score of an alignment Λ is the sum of $s(a_i,b_j)$, $(a_i,b_j)\in\Lambda$, minus the sum of the gap penalties. The similarity between two given sequences is the maximum score among all possible alignments. On the other hand, the distance measure of an alignment was proposed by Sellers [52] and generalized by Waterman *et al.* [65]. The cost of an alignment Λ is the sum of $REP(a_i,b_j)$, $(a_i,b_j)\in\Lambda$, plus the sum of the gap penalties. The distance between two given sequences is the minimum cost among all possible alignments.

Smith *et al.* analyzed the relation of the distance measured by Sellers [53] and the similarity measured by Needleman and Wunsch [49]. Let $\delta_{Sellers}(k)$ and $\delta_{Needleman}(k)$ be the gap penalty with length k by Sellers and Needleman *et al.*, respectively. If we set $\delta_{Sellers}(k) = \delta_{Needleman}(k) + k/2$, the two measurements become equivalent. They also proposed an algorithm for the *maximum similarity segment*, also called *local alignment*. Instead of finding the similarity of the two whole given sequences (strings), this problem tries to find the maximum similarity pair of substrings in the two given sequences. Their algorithm is given in Equation 8 [57].

$$M_{sw1}[i,0] = 0$$
$$M_{sw1}[0,j] = 0$$
$$M_{sw1}[i,j] = \max \begin{cases} M_{sw1}[i-1,j-1] + s(a_i,b_j), \\ \max_{1\le k<i}\{M_{sw1}[i-k,j] - \delta(k)\}, \\ \max_{1\le k<j}\{M_{sw1}[i,j-k] - \delta(k)\}, \\ 0. \end{cases} \tag{8}$$

3.8 Consecutive Insertions and Deletions with the Linear Cost Function by Gotoh

In 1982, Gotoh [24] presented an algorithm for calculating the edit distance in $O(mn)$ time with a linear cost function $\delta(k) = \alpha + \beta k$, where $\alpha, \beta \geq 0$, for a consecutive insertion/deletion of length k. Observing the original DP formula of Waterman *et al.* [65] in Equation 7 with $\delta(k)$, Gotoh presented a more efficient way to calculate the cost of consecutive insertions as follows.

$$\begin{aligned} Ins[i,j] &= \min_{1\leq k\leq j}\{M_{wa}[i,j-k]+\delta(k)\} \\ &= \min\{M_{wa}[i,j-1]+\delta(1), \\ &\quad \min_{2\leq k\leq j}\{M_{wa}[i,j-k]+\delta(k)\}\} \\ &= \min\{M_{wa}[i,j-1]+\delta(1), \\ &\quad \min_{1\leq k'\leq j-1}\{M_{wa}[i,j-1-k']+\delta(k')\}+\beta\} \\ &= \min\{M_{wa}[i,j-1]+\delta(1), Ins[i,j-1] \\ &\quad +\beta\}. \end{aligned} \tag{9}$$

In other words, each $Ins[i,j]$ needs only to check two possible candidates, $M_{wa}[i,j-1]+\delta(1)$ and $Ins[i,j-1]+\beta$, instead of the original $\min_{1\leq k\leq j}\{M_{wa}[i,j-k]+\delta(k)\}$. The formula for the deletion case can be derived similarly.

Based upon the above observation, Gotoh gave a DP formula, shown in Equation 10 [24], for the linear cost function. As a result, with the linear cost function for consecutive insertions and consecutive deletions, the time complexity can be reduced to $O(mn)$.

$$M_{go}[i,j] = \min \begin{cases} M_{go}[i-1,j-1]+REP(a_i,b_j) \\ Ins[i,j] \\ Del[i,j], \end{cases}$$

where

$$\begin{aligned} Ins[i,j] &= \min\{M_{go}[i,j-1]+\delta(1), Ins[i,j-1]+\beta\}, \\ Del[i,j] &= \min\{M_{go}[i-1,j]+\delta(1), Del[i-1,j]+\beta\}. \end{aligned} \tag{10}$$

3.9 Consecutive Insertions and Deletions with the Concave Cost Function by Waterman

In 1984, Waterman presented an algorithm for edit distance by considering consecutive insertions and deletions with the concave cost function [64]. A *concave* function, such as $\delta(k) = \alpha + \beta\log(k), \alpha, \beta > 0$, satisfies the general inequality

$$\delta(\ell_a+\ell_b) \leq \delta(\ell_a)+\delta(\ell_b), \quad \ell_a,\ell_b \geq 1, \tag{11}$$

or equivalently,

$$\delta((1-\alpha)x+\alpha y) \geq (1-\alpha)\delta(x)+\alpha\delta(y), \quad 0\leq\alpha\leq 1. \tag{12}$$

The inequality in Equation 11 shows that the cost of a consecutive insertion/deletion with length $\ell_a+\ell_b$ is less than or equal to that of two consecutive insertions/deletions with lengths ℓ_a and ℓ_b. By combining the two inequalities in Equation 11 with $\delta(\ell_a+\ell_b+\ell_c)$ and $\delta(\ell_a+\ell_c)$, Waterman got the inequality

$$\delta(\ell_a+\ell_b+\ell_c)-\delta(\ell_a+\ell_c) \leq \delta(\ell_a+\ell_b)-\delta(\ell_a), \quad \ell_a,\ell_b,\ell_c \geq 1. \tag{13}$$

Equation 13 is the key of Waterman's algorithm. First, Waterman analyzed the insertion case of the general formula in Equation 7

$$Ins[i,j] = \min_{0\leq k<j}\{M_{wa}[i,k]+\delta(j-k)\}. \tag{14}$$

Assume $Ins[i,j] = M_{wa}[i,l]+\delta(j-l)$ for some $0\leq l<j$ has the minimum value, we have

$$M_{wa}[i,l]+\delta(j-l) \leq M_{wa}[i,k]+\delta(j-k), \quad 0\leq k<j. \tag{15}$$

If $l \leq k$, Equation 13 can be applied by letting $\ell_a = j-k, \ell_b = 1, \ell_c = k-l$, we get

$$\delta(j-l+1)-\delta(j-l) \leq \delta(j-k+1)-\delta(j-k), \quad 0<l\leq k<j. \tag{16}$$

Combining Equations 15 with 16, we have

$$M_{wa}[i,l]+\delta(j-l+1) \leq M_{wa}[i,k]+\delta(j-k+1), \quad 0<l\leq k<j. \tag{17}$$

Thus,

$$\begin{aligned} &Ins[i,j+1] = \min\{M_{wa}[i,j]+\delta(1), \\ &\min_{0\leq k'\leq l}\{M_{wa}[i,k']+\delta(j+1-k')\} \end{aligned} \tag{18}$$

In other words, $M_{wa}[i,k], k>l$, can be ignored since it is dominated by $M_{wa}[i,l]$ when computing $Ins[i,j+1]$. When $Ins[i,j+2], Ins[i,j+3], \cdots, Ins[i,n]$ are computed, it is still true that $M_{wa}[i,k] \leq M_{wa}[i,l]$ for $k>l$. This concept is illustrated in Figure 7.

Waterman constructed the candidate set $S_{Ins}(i) = \{l | Ins[i,l+1] = M_{wa}[i,l]+\delta(1)\}$ to record all positions of $M_{wa}[i,l]$ used in row i. The deletion case can be derived similarly. The DP formula is shown in Equation 19 [64] .

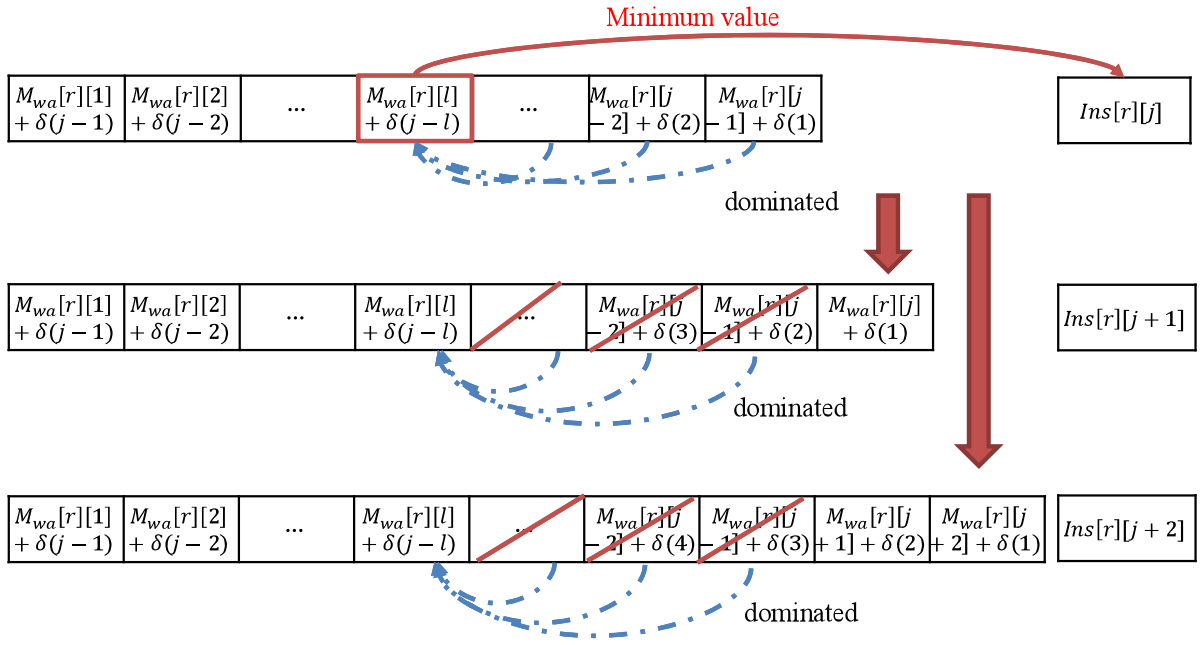


Figure 7: An illustration of computing $Ins[r,j]$ with the concave function in Waterman's algorithm. Assume that $M_{wa}[r,l]+\delta(j-l)$ is the value selected for computing $Ins[r,j]$. When computing $Ins[r,j+1]$ and $Ins[r,j+2]$, we have $M_{wa}[r,l] \leq M_{wa}[r,k]$ for $l < k \leq j$. This situation can be extended to the computation of $Ins[i,n]$.

$$M_{wa}[i,j] = \min \begin{cases} M_{wa}[i-1,j-1] + REP(a_i,b_j) \\ Ins[i,j] \\ Del[i,j], \end{cases}$$

where

$$\begin{aligned} &Ins[i,j] = \min\{M_{wa}[i,j-k] + \delta(j-k), \\ &k \in S_{Ins}(i)\}, \\ &Del[i,j] = \min\{M_{wa}[i-k,j] + \delta(j-k), \\ &k \in S_{Del}(j)\}. \end{aligned} \tag{19}$$

The time complexity depends on the sizes of $S_{Ins}(i)$ and $S_{Del}(j)$. Waterman conjectured that this size does not grow faster than $\log n$. In summary, Waterman's algorithm reduces the number of candidates for consecutive insertions/deletions in each cell of the DP lattice for concave cost functions. The time complexity is $O(|S(i)|mn)$, where $|S(i)|$ is the maximum size of all candidate sets $S_{Ins}(i)$ and $S_{Del}(j)$, and it was conjectured that $|S(i)| = O(\log n)$ [64].

3.10 Consecutive Insertions and Deletions with the Concave Cost Function by Miller and Myers

Based on the candidate set of Waterman [64], in 1988, Miller and Myers [46] presented two edit distance algorithms for consecutive insertions/deletions with the concave cost function. The size of candidate set $S_{Ins}(i)$ and $S_{Del}(j)$ in Waterman's algorithm [64] may become larger and larger, when calculating $Ins[i,j]$ and $Del[i,j]$ in the same row or column. In other words, once a candidate cell (i,j) is put into the candidate set $S_{Ins}(i)$ or $S_{Del}(j)$, the candidate will never be eliminated.

The candidate set $S_{Ins}(i)$ can be arranged into a decreasing list $S_{Ins}(i,x)$, where i is the row index, x is the index of the candidate list. For example, suppose $S_{Ins}(7) = \langle 2,4,5\rangle$. Then, $S_{Ins}(7,1) = 2$, $S_{Ins}(7,2) = 4$ and $S_{Ins}(7,3) = 5$. Note that $S_{Ins}(i,1)$ is always the minimum candidate of $S_{Ins}(i)$ according to the property presented by Waterman [64], as shown in Figure 7.

Here, we present the case of consecutive insertions to illustrate their algorithm, because the deletion case is similar to the insertion case. Their algorithm eliminates the dominated candidates in the candidate list by the idea of *p-curve*. The p-curve consists of all values calculated from $M_{mm}[i,p]$ in row i, as shown in Figure 8.

The example in Figure 8 shows that all possible candidates for calculating $Ins[i,3]$ are the in-

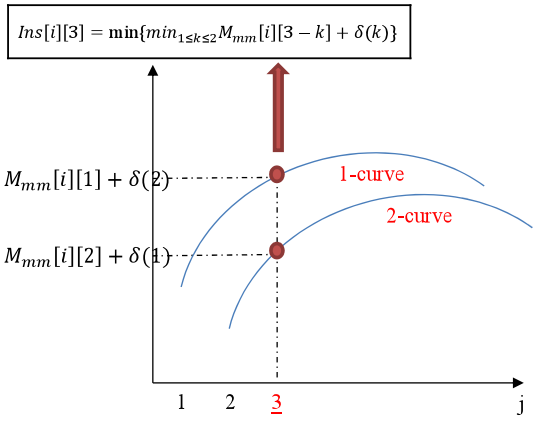


Figure 8: An example of p-curve used to calculate $Ins[i,3]$. The p-curve consists of $M_{mm}[i,p] + \delta(k), 1 \le k \le n-p$. The candidates for calculating $Ins[i,3]$ will choose one from 1-curve and one from 2-curve.

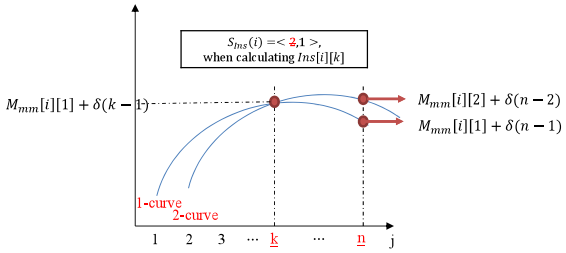


Figure 9: An example for eliminating the candidates of p-curve. $S_{Ins}(r) =< 2,1 >$. Suppose that $M_{mm}[i,1]$ is the minimum candidate for calculating $Ins[i,k]$. In other words, $M_{mm}[i,2] + \delta(n-2) \ge M_{mm}[i,1] + \delta(n-1)$. So the value from $M_{mm}[i,1]$ (1-curve) should be smaller than the value from $M_{mm}[i,2]$ (2-curve) when calculating $Ins[i,j], k < j \le n$.

tersections between the p-curves and the vertical line $j = 3$. The candidate list records the p-curves which will be used in the future. These curves have two properties. First, all curves have the same shape, since they are of the same concave function. Second, the number of intersections between two curves is at most 1.

Figure 9 illustrates how to eliminate these dominated candidates. Once the minimum candidate for calculating $Ins[i,k]$ is determined to be $M_{mm}[i,p]$, we can eliminate the candidates $M_{mm}[i,q]$, where $M_{mm}[i,q] + \delta(n-q) \ge M_{mm}[i,p] + \delta(n-p)$. As shown in Figure 9, 1-curve is always lower than 2-curve after vertical line k. Thus, 2-curve is dominated and it can be eliminated after $Ins[i,k]$ has been calculated.

Furthermore, the algorithm records the lifetime of each candidate in the candidate list. To calculate the lifetime, each candidate needs to find the nearest intersection with other curves in the future. Since the binary search is invoked to find the intersection of two curves, the time required for the candidate list is $O(\log n)$ in each cell. Thus, the total time complexity of their algorithm is $O(mn \log n)$.

3.11 Consecutive Insertions and Deletions with the Mixed Convex and Concave Cost Functions by Eppstein

In 1990, Eppstein [20] considered consecutive insertions/deletions with a mixed convex and concave cost function, composed of interleaving convex and concave segments. For example, a mixed cost function $\delta(k)$ can be split into several segments with index $c_i, 1 \le i \le s$, where $\delta_1(k)$ is convex (concave in his paper) when $0 < k \le c_1$, $\delta_2(k)$ is concave (convex in his paper) when $c_1 < k \le c_2$, $\delta_3(k)$ is convex (concave in his paper) when $c_2 < k \le c_3$ and so on.

A concave function satisfies the *quadrangle inequality*,

$$w(i,j) + w(i',j') \ge w(i',j) + w(i,j'), \quad i \le i' \le j \le j'. \tag{20}$$

In the above inequality, $w(i,j) = \delta(j-i)$ denotes the cost of a consecutive insertion/deletion with length $j-i$. Similarly, a convex function satisfies the inverse quadrangle inequality by replacing $\ge$ with $\le$ in Equation 20.

It should be noted that there is some inconsistency in the definition of a *concave* function in the previous studies. The definition of a concave function by Waterman [64] follows the standard mathematical definition, that is, $f(x+y) \le f(x)+f(y)$. Miller and Myers [46] also follows this definition. However, starting from Yao [69], she interchanged the definitions of a concave function and a convex function (She called the inequality in Equation 20 a *convex quadrangle inequality.*). Then, some subsequent studies follow the definition of Yao, such as Wilber [66], Galil and Giancarlo [22], Klawe and Kleitman [36], Eppstein [20].

Eppstein's algorithm utilizes Wilber's algorithm [66] to deal with the convex (concave in the original paper) segments, and uses Klawe and Kleitman's algorithm [36] to deal with the concave

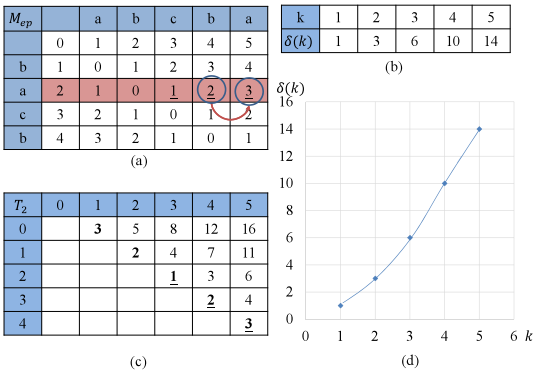


Figure 10: An example for computing the insertion case with the convex cost function by Wilber's algorithm [66]. (a) The DP lattice M_{ep}. (b) A convex cost function $\delta(k)$, where k is the length of a consecutive insertion. (c) The matrix T_2 for calculating $M_{ep}[2][j]$. (d) The curve of $\delta(k)$ in (b).

(convex in the original paper) segments. Wilber's algorithm [66] can find the minimum of each column of an $n \times n$ matrix in $O(n)$ time when the cells in the matrix satisfy the convex quadrangle inequality (convex segment).

An example of the insertion case is shown in Figure 10. $\delta(k)$ shown in Figure 10 (b) is a convex segment, whose visual curve is shown in Figure 10 (d).

In Figure 10 (c), each column j of T_2 records all possible costs of consecutive insertions at $M_{ep}[2][j]$. For example, $M_{ep}[2,5]$ is obtained from $T_2[4,5] = M_{ep}[2,4] + \delta(5-4) = 2+1$, which is the minimum of column 5 of T_2. It means that the distance at $M_{ep}[2,5]$ is formed by $M_{ep}[2,4]$ plus one consecutive insertion of length 1, i.e. $\delta(1)$. Another possible source for $M_{ep}[2,5]$ is $= M_{ep}[2,3] + \delta(5-3) = 1+3 = 4$, which means $M_{ep}[2,3]$ plus one consecutive insertion of length 2, i.e. $\delta(2)$, but it is not the minimum.

T_2 is a matrix satisfying the convex quadrangle inequality, such as $T_2[1][3] + T_2[2][5] \leq T_2[2][3] + T_2[1][5]$. Since T_2 satisfies the convex quadrangle inequality, Wilber's algorithm calculates the minimum of column j, for all j, in $O(n)$ time. That is, the calculation of $M_{ep}[2][j]$ can be done in $O(n)$ time.

Klawe and Kleitman's algorithm can calculate the minimum of all candidates in the concave case with a similar way as Wilber's algorithm in $O(n\alpha(n))$ time, where $\alpha(\ )$ is the inverse Ackermann function, because both of their algorithms use the algorithm presented by Aggarwal *et al.* [1].

Eppstein's algorithm is shown in Equation 21 [20], whose time complexity is $O(n^2 s\alpha(n/s))$.

$$M_{ep}[i,j] = \min \begin{cases} M_{ep}[i-1,j-1] + REP(a_i, b_j) \\ Ins[i,j] \\ Del[i,j], \end{cases}$$

where

$$\begin{aligned} Ins[i,j] &= \min_{1 \leq p \leq s} Ins_p[i,j], \\ Ins_p[i,j] &= \min\{M_{ep}[i,j-k] + \delta_p(j-k), 1 \leq k \leq j\}, \\ Del[i,j] &= \min_{1 \leq p \leq s} \{Del_p[i,j]\}, \\ Del_p[i,j] &= \min\{M_{ep}[i-k,j] + \delta_p(i-k), 1 \leq k \leq i\}. \end{aligned} \tag{21}$$

In 1989, Galil and Giancarlo [22] presented two algorithms for consecutive insertions/deletions with the concave and convex cost functions. Both of their algorithms run in $O(nm \log n)$ or $O(n^2)$ time when the cost function satisfies the *closest zero property*. Their algorithm for the concave function is similar to the algorithm presented by Miller and Myers [46].

3.12 Cyclic Strings by Maes

In 1990, Maes [43] proposed the *cyclic string-to-string correction problem*, a variant of the edit distance problem, which requires to transform the rotations of A into B. A *k-rotation* of A is to remove its prefix with length k and to concatenate the removed prefix to the end. For example, `tatgagatca` is a 4-rotation of $A =$ `atcatatgag`.

The naïve method is to find the edit distance of all k-rotations, $0 \leq k \leq m-1$, of A versus B by applying the algorithm of Wagner and Fischer [62] m times, whose total time complexity is $O(m^2 n)$.

The algorithm of Maes constructs the edit lattice of AA , the concatenation of A with itself, versus B. Maes found that the minimal cost edit paths between AA and B in the lattice may not cross, but may intersect. So the search area is limited, instead of mn cells. The algorithm is shown in Algorithm 1. As the search area in each loop in line 6 is a subdivision of the area between the minimal cost path from $(0,0)$ to (m,n) and the minimal cost path from $(m,0)$ to $(2m,n)$ (inclusive), the total time required for each loop in line 3 is sum up to $O(mn)$. So the total time complexity is $O(mn \log m)$ and the required space is $O(mn)$.

3.13 Cyclic Strings with Divide-and-conquer by Marzal and Barrachina

In 2000, Marzal and Barrachina [44] proposed an improved algorithm over the algorithm of Maes

Algorithm 1 The algorithm for the cyclic string-to-string correction problem by Maes [43].

```
q = ⌈log m⌉
Find the minimal cost paths from (0,0) to (m,n) and from (m,0) to (2m,n)
for i = q − 1 down to 0 do
    j = 2^i
    while j < m do
        Find the minimal cost path from (j,0) to (j+m,n) between the path from (j−2^i,0) to (j−2^i+m,n) and the path from (min(j+2^i,m),0) to (min(j+2^i,m)+m,n)
        j = j + 2^{i+1}
    end while
end for
```

[43] and the algorithm of Gregor and Thomason [25]. Their algorithm is based on the divide and conquer strategy of Maes and applies a branch and bound strategy inspired from Gregor and Thomason [25]. The cost function is restricted that the cost of each insertion, deletion or replacement is 1.

The algorithm of Marzal and Barrachina calculates all minimal edit distance paths for $\{M_{mb}[0,0]$ to $M_{mb}[m,n]$, $M_{mb}[1,0]$ to $M_{mb}[m+1,n]$, $\cdots$, $M_{mb}[m-1,0]$ to $M_{mb}[2m-1,n]$, $M_{mb}[m,0]$ to $M_{mb}[2m,n]\}$, where M_{mb} denotes the DP lattice of AA and B. The algorithm utilizes the lower bound $g(i,j)$ of all edit paths starting between $M_{mb}[i,0]$ and $M_{mb}[j,0]$ to eliminate the calculation of some edit paths.

Let $\sigma^k(A)$ denote the k-rotation of A. The edit distance $d(\sigma^k(A),B)$ is the minimum edit path from $M_{mb}[k,0]$ to $M_{mb}[k+m,n]$. For example, suppose $m=16$. The algorithm calculates $d_{min}=\min\{d(\sigma^0(A),B),\ d(\sigma^8(A),B)$ and $d(\sigma^{16}(A),B)\}$, which divides the interval $[0,16]$ into $[0,8]$ and $[8,16]$. Recursively apply to $[0,8]$ if $g(0,8)<d_{min}$ and $[8,16]$ if $g(8,16)<d_{min}$. No edit distance $d(\sigma^k(A),B)$, $k\in[i,j]$, will be the minimum edit distance, if the lower bound $g(i,j)\geq d_{min}$, where d_{min} means the minimum edit distance of all computed $d(\sigma^k(A),B)$ before. The lower bound formulas are shown in Theorems 2 and 3. The time complexity of their algorithm is $O(mn\log m)$, which is the same as Maes' algorithm.

Theorem 2. *[44] The lower bound of $d(\sigma^k(A),B)$ for all $k\in[i,j]$, $0\leq i<j\leq m$, is given by*

$$g_1(i,j)=\max(0,\lceil\tfrac{d(\sigma^i(A),B)+d(\sigma^j(A),B)}{2}\rceil-(j-i))\leq d(\sigma^k(A),B).$$

Theorem 3. *[44] The lower bound of $d(\sigma^k(A),B)$ for all $k\in[i,j]$, $0\leq i<j\leq m$ is given by*

$$g_2(i,j)=\max(g_1\{i,j\},\min_{i<k'<j}\rho_{k'}(A,B))\leq d(\sigma^k(A),B),$$

where $\rho_{k'}(A,B)=\min_{0\leq q\leq n}(\rho(A_{1\cdots k'},B_{q+1\cdots n))+(\rho(A_{k'+1\cdots m},B_{1\cdots q}))$, $\rho(A,B)=\max(m,n)-|\Sigma|$ and $|\Sigma|$ is the alphabet set of A and B.

3.14 Run-length Encoding Strings by Bunke and Csirik

In 1993, Bunke and Csirik [12] proposed an algorithm for calculating the edit distance on *run-length encoding* (RLE) strings (sequences). Suppose that two RLE strings A and B are of lengths m_a and n_b, respectively, and the lengths of the extracted plain text are m and n, respectively. For example, if $A=$ aaaacccccbb, then the RLE string is encoded as $A=\mathtt{a}^4\mathtt{c}^5\mathtt{b}^2$, $m_a=3$ and $m=11$.

A pair of matched or unmatched symbols in the RLE format represents a block in the plain text. The DP lattice is divided into two kinds of blocks. A *dark block* corresponds to a matched pair and a *light block* corresponds to an unmatched pair. In each block, only the cells on the right and bottom boundaries need to be calculated. Each cell can be calculated in constant time. Thus, the total time complexity is $O(n_bm+m_an)$ if the lengths of all runs in both sequences are the same.

In 1995, Bunke and Csirik proposed an improved algorithm [13] for the same problem. The problem is restricted on the cost function that the costs of each insertion, deletion and replacement are 1, 1, and 2, respectively. There is no restriction on the length of each run.

Table 5 shows an example for the block division of the DP lattice. As examples, the dark block $D_{1,1}$ corresponds to the matched pair of $\mathtt{a}^3$ in A and $\mathtt{a}^2$ in B, the light block $D_{2,1}$ corresponds to the unmatched pair of $\mathtt{c}^3$ in A and the run $\mathtt{a}^2$ in B. Only the cells on the bottom row and rightmost column of each block needs to be calculated. An example is shown in Table 6. By Lemmas 1 and 2, the computation time of each cell is constant. Thus, the total time complexity is $O(n_bm+m_an)$.

Lemma 1. *[13] Let x be a symbol, and A and B be two strings. Then $d(Ax^k,Bx^k)=d(A,B)$ for any $k\geq 0$.*

Lemma 2. *[13] Let x and y, $x\neq y$ be two symbols, and A and B be two strings. Then $d(Ax^k,By^h)=min\{d(Ax^k,B)+h,d(A,By^h)+k\}$ for any $k,h\geq 0$.*

Table 5: The block division of the DP lattice for RLE strings with $A = \mathtt{a}^3\mathtt{c}^3\mathtt{b}^2$ and $B = \mathtt{a}^2\mathtt{c}^2$ by Bunke and Csirik [13].

	-	a a	c c
-	$D_{0,0}$	$D_{0,1}$	$D_{0,2}$
a a a	$D_{1,0}$	$D_{1,1}$	$D_{1,2}$
c c c	$D_{2,0}$	$D_{2,1}$	$D_{2,2}$
b b	$D_{3,0}$	$D_{3,1}$	$D_{3,2}$

Table 6: The DP lattice for $A = \mathtt{a}^3\mathtt{c}^3\mathtt{b}^2$ and $B = \mathtt{a}^2\mathtt{c}^2$ by Bunke and Csirik [13].

	-	a	a	c	c
-	0	1	2	3	4
a	1		1		3
a	2		0		2
a	3	2	1	2	3
c	4		2		4
c	5		3		5
c	6	5	4	5	6
b	7		5		7
b	8	7	6	7	8

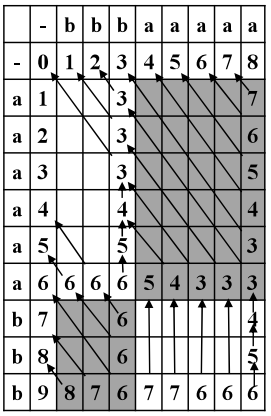


Figure 11: The DP lattice for RLE strings $A = \mathtt{a}^6\mathtt{b}^3$ and $B = \mathtt{b}^3\mathtt{a}^5$ by Arbell *et al.* [6], where the arrow lines mean where the values of the cells come from (not unique).

3.15 Run-length Encoding Strings by Arbell *et al.*

In 2002, Arbell *et al.* [6] proposed an algorithm for calculating the edit distance of two *run-length encoding* RLE strings (sequences). In their algorithm, on the cost of each insertion, deletion or substitution is assumed to be 1.

The DP lattice of two RLE strings is divided into several blocks, as shown in the example of Figure 11. In a dark block, the value of each cell $M_{ar}[i,j]$ is equal to $M_{ar}[i-1,j-1]$. In a light block, the top row and the leftmost column are separated into three zones (zone I, zone II, zone III) according to the position of the calculated cell and the form of block (horizontal block or vertical block), as shown in the example of Figure 12.

The algorithm chooses the minimum of each cell from two possible candidates, one from zone I and the other from zone II, because the value from zone III must be larger. For a horizontal block

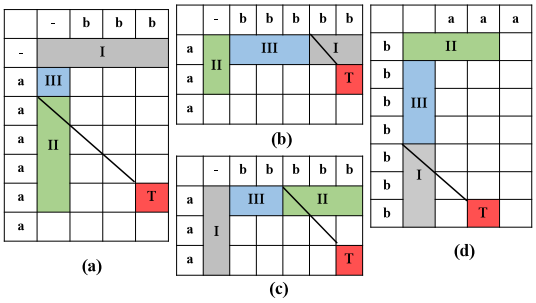


Figure 12: The zones in a light block of the DP lattice for RLE strings by Arbell *et al.* [6]. (a) An example for the rightmost column. (b) Another example for the rightmost column. (c) An example for the bottom row. (d) Another example for the bottom row.

Table 7: An example for computing the edit distance between an RLE string and an uncompressed string by Liu *et al.* [39], where $A = \mathtt{a}^6\mathtt{b}^3$ and $B = \mathtt{bbbaaaaa}$.

	-	b	b	b	a	a	a	a	a
-	0	1	2	3	4	5	6	7	8
a^6	6	6	6	6	5	5	5	5	5
b^3	9	8	7	8	8	9	9	8	8

, the calculation of $M_{ar}[i,j]$ in the rightmost column considers only two positions: $M_{ar}[i-1,j]$ and $M_{ar}[i_{top}, j_{diagonal}]$ (the intersection point of zone I and the diagonal line through $M_{ar}[i,j]$) in zone I. Other cases can be done similarly.

The calculation of each cell in both dark and light blocks can be done in $O(1)$ time, and the number of calculated cells is $O(n_b m + m_a n)$. Thus, the total time complexity of the algorithm is $O(n_b m + m_a n)$.

3.16 A Run-length Encoding String and an Uncompressed String by Liu *et al.*

In 2007, Liu *et al.* [39] proposed an algorithm for the edit distance between an RLE string (sequence) A and an uncompressed string (sequence) B. Here, the cost of each insertion, deletion or replacement is 1.

In the DP lattice $D_{li}[i,j]$, the index i represents A_i, the ith run in A. An example in shown in Table 7. Note that $D_{li}[i,j]$ is different from $M_{li}[i,j]$. For example, $D_{li}[1,1] = M_{li}[6,1]$ is the edit distance of the first run of A ($\mathtt{a}^6$) and $B = \mathtt{b}$, while $M_{li}[1,1]$ is the edit distance of $A = \mathtt{a}$ and $B = \mathtt{b}$. The main idea is to check how the value of $D_{li}[i,j]$ is calculated from $D_{li}[i,j-1]$ with Lemma 3.

Lemma 3. *[60]* $M_{li}[i,j] - M_{li}[i,j-1] \in \{-1, 0, 1\}$.

The DP formula for calculating $D_{li}[i,j]$ is shown as follows [39].

$$D_{li}[i,j] = \min_{0 \le u \le j} \{D_{li}[i-1,u] + \varepsilon D_{li}[A_i, B_{u+1\cdots j}]\}. \tag{22}$$

Table 8: The tables used in the algorithm of Liu *et al.* [39], where $T_{A^-}[T_A[u]] = u$, $T_{B^-}[T_B[u]] = u$ and $T_{C^-}[T_C[u]] = u$.

Name	Value	Restriction
$T_A[u]$	$D_{li}[i-1,u]+$ $B_{1\cdots u}(\tau)$	$\max\{j-\lvert A_i\rvert, 0\}$ $\le u \le j$
$T_B[u]$	$D_{li}[i-1,u]+$ $B_{1\cdots u}(\tau) - u$	$0 \le u \le j - \lvert A_i\rvert$ and $B_{1\cdots u}(\tau) \le \lvert A_i\rvert$
$T_C[u]$	$D_{li}[i-1,u] - u$	$0 \le u \le j$ and $\lvert A_i\rvert \le B_{1\cdots u}(\tau)$
$T_{A^-}[v_1]$	$\max\{u \mid T_A[u] = v_1$ for $0 \le u \le j\}$	$-n \le v_1 -$ $T_A[0] \le 2n$
$T_{B^-}[v_2]$	$\max\{u \mid T_B[u] = v_2$ for $0 \le u \le j - \lvert A_i\rvert\}$	$-2n \le v_2 -$ $T_B[0] \le n$
$T_{C^-}[v_3]$	$\min\{u \mid T_C[u] = v_3$ for $0 \le u \le j\}$	$-2n \le v_3 -$ $T_C[0] \le 0$

$$\begin{aligned} \varepsilon D_{li}[A_i, B_{u\cdots j}] &= max\{|A_i|, j-u\} - \\ &\quad min\{|A_i|, B_{u+1\cdots j}(\tau)\} \\ &= \begin{cases} |A_i| - B_{u+1\cdots j}(\tau) \\ \quad \text{if } |A_i| \ge j-u, \\ j-u-B_{u+1\cdots j}(\tau) \\ \quad \text{if } B_{u+1\cdots j}(\tau) \le |A_i| \le j-u, \\ j-u-|A_i| \\ \quad \text{if } |A_i| \le B_{u+1\cdots j}(\tau). \end{cases} \end{aligned} \tag{23}$$

Combing Lemma 3 with Equation 22, $D_{li}[i,j]$ comes from three possible candidates $D_{li}[i,j-1]-1$, $D_{li}[i,j-1]$ or $D_{li}[i,j-1]+1$. Accordingly, their algorithm uses three tables T_A, T_B, and T_C to store the value of left-hand side of the three equations in Equation 22. To make their algorithm more efficient, they also build three inverted tables T_{A^-}, T_{B^-} and T_{C^-} for T_A, T_B, and T_C. The definition and restriction of these tables are shown in Table 8. The time complexity of the algorithm is $O(m_a n)$.

3.17 LCS of Run-length Encoding Strings by Liu *et al.* and Ann *et al.*

In 2008, Liu *et al.* also proposed an algorithm for calculating the LCS between an RLE string and an uncompressed string [40]. The time complexity is $O(m_a n)$.

As mentioned in Section 2.2, the LCS problem is equivalent to calculate the edit distance with costs of insertion, deletion and replacement being 1, 1, and 2, respectively. The conversion formula is $L = \frac{m+n-d}{2}$, as shown in Equation 3.

In 2008, Ann *et al.* [4] presented an algorithm

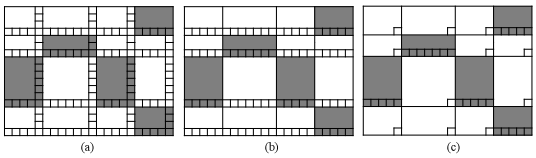


Figure 13: The comparison of three algorithms for calculating the LCS of RLE strings by Ann *et al.* [4], where only small cells are required to be calculated. (a) $O(n_b m + m_a n)$ by Bunke and Csirik [13]. (b) $O(m_a n)$ by Liu *et al.* [40]. (c) $O(n_b m_a + p_1)$ by Ann *et al.* [4], where p_1 denotes the number of cells in the bottom boundaries of all black blocks.

for calculating the LCS of two RLE strings [4] with $O(n_b m_a + \min\{p_1, p_2\})$ time, where p_1 and p_2 denote the number of cells in the bottom and right boundaries of all black blocks. They also illustrated the comparison of the time complexities of three algorithms, as shown in Figure 13.

In 2012, Ann *et al.* [5] presented an algorithm for computing the constrained LCS of RLE strings with $O(m_a n_b r + r \times \min\{q_1, q_2\} + q_3)$, where r denotes the length of constrained sequence P, q_1 and q_2 denote the numbers of cells in the south and east faces of cuboids blocks on the first layer of the DP lattice, and q_3 denotes the number of face cells of black cuboids of the DP lattice. Note that a black cuboid corresponds to a strong match, meaning that three runs in A, B and P have the same symbol.

3.18 The Block Edit Problem by Ann *et al.*

In 2010, Ann *et al.* [3] proposed an algorithm for the block edit problem. The block edit problem not only involves the character-edit operations, but also the block-edit operations. Three problems $P(EIS, C)$, $P(EI, L)$ and $P(EI, N)$ are presented with some restrictions that the block operations cannot overlap and they should be performed from left to right on the parts which have not been edited before.

These edit problems are formulated as $P(o, c)$, where o is the block copy operation and c is the cost measurement. The definition of allowed operations and cost measurement are given in Table 9. An example of the operations is shown in Figure 14. For the problem $P(EIS, C)$, the block operations include block deletion, internal block copy, external block copy and shift block copy, whose costs are constant.

Table 9: The allowed block operations and cost measurements by Ann *et al.* [3], with the example shown in Figure 14.

Block Operations		
Name	**Description**	**Example**
External Copy (E)	Copy a substring of X and insert it into a valid position of the active part of W_i.	$W_1 \to W_2$
Internal Copy (I)	Copy a substring of the inactive part of W_i and insert it into a valid position of the active part of W_i.	$W_4 \to Y$
Shift Copy (S)	Copy a shifted string, which is a substring of X or a substring of the inactive part of W_i, and insert it into a valid position of the active part of W_i.	$W_3 \to W_4$ $W_2 \to W_3$
Deletion	Delete a valid substring from the active part of W_i.	$X \to W_1$
Cost Measurement		
Name	**Description**	
Constant cost (C)	Costs of all block operations with different lengths are the same.	
Linear cost (L)	Cost of one block operation is $p_s + ip_e$, where p_s and p_e are constants and i is the length of the copied or deleted substring.	
Nested cost(N)	Cost of block deletions is constant. Cost of a block copy is $p_{copy} + d_{ch}(s_1, s_2)$ where s_1 is the copied string, s_2 is the string after editing, and $d_{ch}(s_1, s_2)$ is the edit distance from s_1 to s_2 with character edit operations.	

The $P(EIS, C)$ problem is solved with a DP formula as shown in Figure 15. The straightforward DP algorithm requires $O(nm^2(n + m)|\Sigma|)$ time. By some processing techniques with $O(n + m^2)$ time, the time complexity can be reduced to $O(mn)$.

For more details in $P(EIS, C)$, the traditional edit distance $d_1(X_i, Y_j)$ with character edit operations can be calculated in $O(1)$ time. Block deletion distance $d_2(X_i, Y_j)$ can also be calculated in $O(1)$ time by preserving the current minimum value of $\{d(X_{k-1}, Y_j) | 0 \le k \le i\}$ in each iteration. To compute the costs of the external block copy d_3, internal block copy d_4, external block shifted copy d_5 and internal block shifted copy d_6, a preprocess for building a suffix tree and a minimum query structure is needed.

$P(EI, L)$ and $P(EI, N)$ problems can be solved by the DP algorithm with a similar strategy except

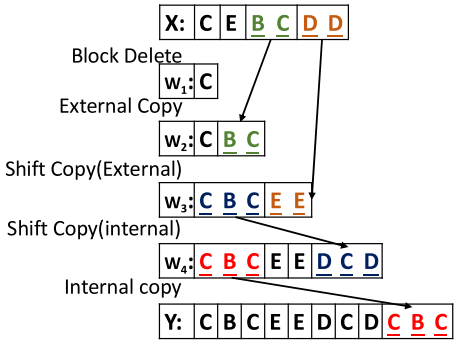


Figure 14: An example of the block edit operation by Ann *et al.* [3].

the cost calculation. As a result, $P(EI, L)$ can be solved in $O(mn \log m)$ time with $O(n + m^2)$ preprocessing time and $P(EI, N)$ can be solved in $O(nm^2)$ time with $O((n + m)m^2)$ preprocessing time.

In 2014, Peng and Yang [50] applied the incremental suffix maximum query with the set union and find technique to improve the algorithm for $P(EI, L)$, whose time complexity is further reduced to $O(nm + m^2)$.

4 Genome Rearrangement

The genome rearrangement involves a group of block edit operations, originally used to compare the genomes of different species. The operations include reversal, transposition, translocation, block move and duplication [3, 7–11, 14–19, 26–28, 32, 34, 37, 47, 54, 55, 63]. Since the problems involving overlapping operations are NP-hard, some restrictions on the operations were proposed, such as non-overlapping operations. With such restrictions, the problems become solvable with polynomial time.

4.1 Overlapping Reversals on Permutations by Kececioglu and Sankoff

In the *reversal distance on permutation* problem, the *reversal* operation is used to transform permutation π into permutation γ [33], and the operated intervals may overlap. To simplify this problem, we can regard the target permutation γ as a sorted sequence from 1 to n. A reversal $\rho(i, j)$ reverses the order of π in the interval $[i, j]$. Let π' be the result after applying a reversal $\rho(i, j)$ on

$$
\begin{aligned}
d(X_i, Y_j) &= \begin{cases} \infty, & \text{if } i < 0 \text{ or } j < 0 \\ 0, & \text{if } i = j = 0 \\ \min \left\{ \begin{array}{l} d_1(X_i, Y_j), \\ d_2(X_i, Y_j), \\ d_3(X_i, Y_j), \\ d_4(X_i, Y_j), \\ d_5(X_i, Y_j), \\ d_6(X_i, Y_j) \end{array} \right\}, & \text{otherwise} \end{cases} \\
d_1(X_i, Y_j) &= \begin{cases} d(X_{i-1}, Y_{j-1}), & \text{if } X_i = Y_j \\ \min \left\{ \begin{array}{l} d(X_i, Y_{j-1}), \\ d(X_{i-1}, Y_j) \end{array} \right\} + 1, & \text{if } X_i \neq Y_j \end{cases} \\
d_2(X_i, Y_j) &= \min\{d(X_{k-1}, Y_j) + p_{delete} | i \leq k \leq i\} \\
d_3(X_i, Y_j) &= \min\{d(X_i, Y_{k-1}) + p_{copy} | \\
&\qquad Y_{k \cdots j} \text{ is a substring of } X\} \\
d_4(X_i, Y_j) &= \min\{d(X_i, Y_{k-1}) + p_{copy} | \\
&\qquad Y_{k \cdots j} \text{ is a substring of } Y_{k-1}\} \\
d_5(X_i, Y_j) &= \min\{d(X_i, Y_{k-1}) + p_{copy} | \\
&\qquad Y_{k \cdots j} \text{ is a shifted substring of } X\} \\
d_6(X_i, Y_j) &= \min\{d(X_i, Y_{k-1}) + p_{copy} | \\
&\qquad Y_{k \cdots j} \text{ is a shifted substring of } Y_{k-1}\}
\end{aligned} \tag{24}
$$

Figure 15: The DP formula for solving $P(EIS, C)$ by Ann *et al.* [3].

a sequence π. The formal definition is given as follows.

$$
\pi'_t = \begin{cases} \pi_{i+j-t} & \text{if } \ i \leq t \leq j, \\ \pi_t & \text{otherwise.} \end{cases} \tag{25}
$$

For example, if $\pi = 4213$, $\pi \cdot \rho(1, 3) = 1243$. An example for transforming from π to γ is shown in Figure 16.

The straightforward method is to apply one reversal to bring one element into its correct place. So the reversal distance is at most $n-1$. In a permutation π, a break point is a pair of neighboring positions (i, j) such means that $|\pi_{i+1} - \pi_i| \neq 1$. In other words, π_i and π_{i+1} are not consecutively increasing or decreasing. For example, suppose that $\pi = 4213$. Then, $(1, 2)$, and $(3, 4)$ are break points. Whenever there exists a breakpoint in π, π is not sorted completely.

In 1993, Kececioglu and Sankoff [33, 34] presented a greedy approximation algorithm based on the concept of breakpoints. They apply a reversal operation between two breakpoints which can eliminate at least one breakpoint until there is no breakpoint in π. An example of their algorithm is shown in Figure 16. Their greedy algorithm is of 2-approximation, whose time complexity is $O(n^2)$ and space complexity is $O(n)$.

Kececioglu and Sankoff [33, 34] also presented a branch and bound method with linear programming for getting the exact solution. A reversal can

π:	[0	[4]	2]	[1	[3]	5]
$\Downarrow \rho_1(1,3)$						
π':	0	1	[2	4]	[3	5]
$\Downarrow \rho_2(3,4)$						
γ:	0	1	2	3	4	5

Figure 16: An example of the greedy algorithm for calculating the reversal distance on a permutation. The first and last elements are the pseudo boundaries 0 and $n+1$. $[\pi_i, \pi_{i+1}]$ means a breakpoint between positions i and $i+1$ in π.

remove at most two breakpoints. A series of i reversals can remove at most $i+1$ breakpoints, since only the ith reversal can removes two breakpoints. For example, given π=42315, applying a reversal $\rho(2,3)$ and then applying a reversal $\rho(1,4)$ on π can remove three breakpoints.

Kececioglu and Sankoff [33, 34] showed that a lower bound of the required reversals is $\lceil \frac{2b(\pi)}{3} - \frac{b_2(\pi)}{3} \rceil$, where $b_2(\pi)$ is the number of reversals which can remove two breakpoints at the same time and $b(\pi)$ is the number of breakpoints in π. To find a dynamic upper bound, they performed the greedy algorithm which always tries to apply a reversal to remove two breakpoints one time. Then, the exact algorithm is a branch and bound approach by eliminating some paths in the search tree with the upper bounds and lower bounds. The required time and space are $O(tL(n,n))$ and $O(n^2)$, respectively, where t is the size of the branch and bound search tree and $L(n,n)$ is the time required for solving a linear programming with n variables and n constraints.

4.2 Lower Bounds for Overlapping Reversals on Permutations by Bafna and Pevzner

Bafna and Pevzner [7] proved a tighter lower bound $\frac{2b(\pi)}{3} - \frac{c_4(\pi)}{3}$, where c_4 is the number of 4-cycles (a cycle with 4 edges) in the *breakpoint graph* [33]. They also proposed some efficient algorithms based on the breakpoint graph.

In the breakpoint graph $G(\pi)$ of a permutation π, each breakpoint contributes two edges, a solid eage and a dashed edge, as an example shown in Figure 17. A *solid edge* connects vertices π_i and π_j if $|i-j|=1$ and $|\pi_i - \pi_j| > 1$. A *dashed edge* connects π_i and π_j if $|\pi_i - \pi_j| = 1$ and $|i-j| > 1$. For example, there is a solid edge between $\pi_3 = 1$ and $\pi_4 = 4$ and a dashed edge between $\pi_5 = 6$ and $\pi_{10} = 7$.

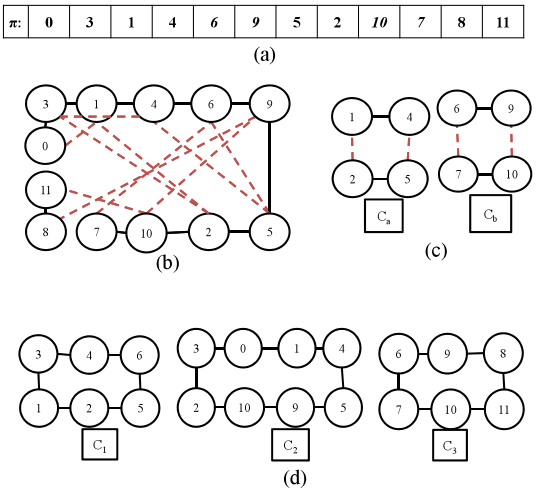


Figure 17: An example of the breakpoint graph. (a) A permutation π = 03146952(10)78(11), where 0 and 11 are dummies used as bounaries. (b) The breakpoint graph $G(\pi)$ with $c(\pi)=3$ and $b(\pi)=10$. (c) Crossing cycles C_a and C_b in $G(\pi)$. (d) A maximal cycle decomposition of $G(\pi)$.

In $G(\pi)$, the solid edge and dashed edge must be interleaved to form a cycle, such as C_a and C_b in Figure 17. Let $c(\pi)$ denote the number of cycles in the maximal cycle decomposition of $G(\pi)$. A tighter lower bound of the reversal distance $d(\pi)$ is given in Theorem 4.

Theorem 4. *[7] For any permutation π,*

$$d(\pi) \geq b(\pi) - c(\pi) \geq$$
$$b(\pi) - c_4(\pi) - (c(\pi) - c_4(\pi)) = \frac{2b(\pi)}{3} - \frac{c_4(\pi)}{3}.$$

When all the $c(\pi)$ are c_4, we have $c(\pi) = \frac{b(\pi)}{2}$. Clearly, $d(\pi) \geq \frac{b(\pi)}{2}$. A property for c_4 is described in Lemma 4. Two cycles are *crossing* means that their solid edge are interleaved. For example, in Figure 17, solid edges (1, 4), (5, 2) in C_a are interleaved with solid edges (6, 9), (10, 7) in C_b.

Lemma 4. *[7] Suppose that a 4-cycle C has no reversal which can remove at least one breakpoint in C. If C has a crossing cycle C', doing a reversal on C' will make C have a reversal removing at least one breakpoint.*

x-reversal, $x \in \{-2,-1,0,1,2\}$, is a reversal which removes x breakpoints if $x \geq 0$, or adds x breakpoints if otherwise. With such an observation, Algorithm 2 sorts a signed permutation by at most $b(\pi) - \frac{c_4(\pi)}{2}$ steps with $\frac{3}{2}$ approximation ratio. Furthermore, with the properties of c_4, they

presented another algorithm with $b(\pi)-\frac{c_4(\pi)}{4}$ steps and $\frac{7}{4}$ approximation ratio. Overall, the time complexities of these algorithms are $O(n^2)$.

Algorithm 2 Sorting a signed permutation π [7], where $Reversal(\pi)$ is presented in [33].

```
1: while π contains a breakpoints do
2:     if π has no decreasing strips then
3:         if any 4-cycle C remains in G(π) then
4:             Find a cycle C' which crosses C.
5:             Do a 0-reversal on C'.
6:             Do a 2-reversal on the 4-cycle C.
7:         else
8:             Do a 0-reversal on an arbitrary cycle.
9:         end if
10:    else
11:        ρ = Reversal(π)
12:        π = π · ρ
13:    end if
14: end while
```

4.3 An Approximation Algorithm for Overlapping Transpositions on Permutations by Walter *et al.*

The *transposition distance* is defined to be the minimum number of transposition operations to transform an input sequence π into the target sequence γ [63]. A *transposition* $\tau(i,j,k)$ exchanges two substrings $\pi_{i..j-1}$ and $\pi_{j..k-1}$, $1 \le i < j < k \le n+1$. For example, Figure 18 illustrates the transposition operations. Let π' be the resulting sequence after applying a transposition $\tau(i,j,k)$ on a sequence π. The formal definition is given by Walter *et al.* as follows [63].

$$\pi'_t = \begin{cases} \pi_{t+j-i} & \text{if } i \le t < i+k-j, \\ \pi_{t-k+j} & \text{if } i+k-j \le t < k, \\ \pi_t & \text{otherwise.} \end{cases} \quad (26)$$

Walter *et al.* [63] presented an approximation algorithm with approximation ratio 2.25 for computing the transposition distance. Here, we still regard target sequence γ as a sorted sequence from 1 to n. The algorithm is also based on the breakpoint graph. We add two dummy elements $\pi_0 = 0$ to the front and $\pi_{n+1} = n+1$ to the rear of π. A breakpoint for transposition on permutation π exists between two adjacent elements π_{i-1} and π_i, when $\pi_i - \pi_{i-1} \neq 1$, $1 \le i \le n+1$. For example, given $\pi = \texttt{41235687}$ as shown in Figure 18, $(\texttt{4},\texttt{1})$, $(\texttt{3},\texttt{5})$, $(\texttt{6},\texttt{8})$ and $(\texttt{8},\texttt{7})$ are the breakpoints. Note that $(\texttt{8},\texttt{7})$ is not a breakpoint for calculating the reversal distance.

π	0	4	1	2	3	5	6	8	7	9
$\Downarrow \tau(1,2,5)$										
π'	0	1	2	3	4	5	6	8	7	9
$\Downarrow \tau(7,8,9)$										
γ	0	1	2	3	4	5	6	7	8	9

Figure 18: An example of transforming permutation $\pi = \texttt{41235687}$ into $\gamma = \texttt{12345678}$ by transpositions. The first and last elements are the dummy boundaries $\pi_0 = 0$ and $\pi_9 = 9$.

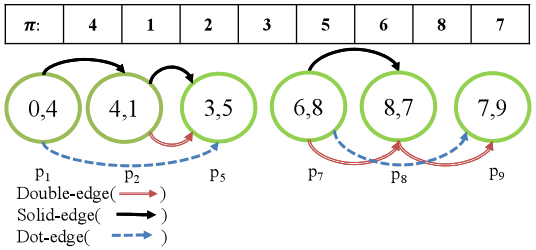


Figure 19: An example of the transposition breakpoint graph $D(\pi)$ for $\pi = \texttt{41235687}$.

In the transposition breakpoint graph $D(\pi)$, node p_i corresponds to a breakpoint between π_{i-1} and π_i. There are three kinds of edges, a solid-edge from p_i to p_j when $\pi_j - \pi_{i-1} = 1$, $i < j$ and $\pi_j \neq n+1$; a double-edge from p_i to p_j when $\pi_j - \pi_{i-1} = 1$, $i < j$ and $\pi_{i-1} \neq 0$; a dot-edge from p_i to p_j when $\pi_i - \pi_{j-1} = 1$ and $i < j$. An example of the transposition breakpoint graph is shown in Figure 19.

An x-transposition, $x \in \{-3,-2,-1,0,1,2,3\}$ removes x breakpoints if $x \ge 0$, or adds x breakpoints if otherwise. Note, to keep the consistency of this paper, we have reversed the definition from the original one defined by Walter *et al.* [63]. In $D(\pi)$, if there are $p_i \to p_j$, $p_j \Rightarrow p_k$ and $p_i \dashrightarrow p_k$, then there exists a 3-transposition. As the result, algorithm 3 is presented to find the transposition distance of a permutation. The approximation ratio of the algorithm is 2.25, provided by Lemma 5.

Lemma 5. *[63] Given a permutation π and its transposition breakpoint graph $D(\pi)$ with more than 5 nodes, if there is no 2-transposition nor 3-transposition, it is possible to remove at least four nodes in three transpositions.*

Algorithm 3 The approximation algorithm for finding the transposition distance of a permutation π by Walter *et al.* [63].

```
1: Construct the transposition breakpoint graph D(π)
2: i := 0
3: while |V| ≠ 0 do
4:     i := i + 1
5:     if there is a 3-transposition then
6:         τ_i := 3-transposition
7:     else
8:         if there is a 2-transposition then
9:             τ_i := 2-transposition
10:        else there is a 1-transposition
11:            τ_i := 1-transposition
12:        end if
13:    end if
14:    π := π · τ_i
15:    Construct D(π)
16: end while
17: return i, (τ_1, τ_2, ··· , τ_i)
```

4.4 Upper and Lower Bounds for Reversals and Transpositions on Binary Strings by Christie and Irving

Christie and Irving [17] proved the upper and lower bounds for the reversal distance and the transposition distance between two binary strings with the concept of breakpoints. They also proved that the decision version of the reversal distance problem on two binary strings is NP-Hard, by transforming from the 3-partition problem, which is NP-complete [23].

The reversal breakpoint they defined for the reversal distance of two binary strings S and T is different from two permutations. In their new definition, a breakpoint exists in a length-2 common substring κ of S and T when the number of occurrences of κ in S is greater than the number of occurrences of κ in T. For example, if S has two substrings "00" and T has one, one of "00" in S has to be broken when transforming from S into T, which means a breakpoint in S. Furthermore, substrings "01" and "10" are counted together since reversals can convert "01" into "10" and vice versa. The number of breakpoints $b_r(S,T)$ of two binary strings S and T can be calculated by Equation 27, which calculates the occurrence difference of all length-2 substrings in S and T.

$$b_r(S,T) = \sum_{\alpha \le a < b \le \omega} \delta(f_{ab}(S) + f_{ba}(S) - f_{ab}(T) - f_{ba}(T)) + \sum_{0 \le a \le 1} \delta(f_{aa}(S) - f_{aa}(T)),$$

where $\delta(x) = \max\{x, 0\}$, $f_{ab}(S)$ is the number of substring "ab" in S and, α and ω are the dummy elements added to the front and the rear of S and T. (27)

Consequently, the lower bound of the reversal distance, provided in Theorem 5, can be obtained, because a reversal can remove at most two breakpoints.

Theorem 5. *[17] Given two binary strings S and T, let $d_r(S,T)$ be the reversal distance. Then,*

$$\lceil b_r(S,T)/2 \rceil \le d_r(S,T) \le \lfloor n/2 \rfloor.$$

For the transposition distance $d_t(S,T)$ of two binary strings, the number of occurrences of "01" and "10" have to be counted separately. Then the proof technique of the reversal distance can be applied to the transposition distance similarly. The lower bound and upper bound of $d_t(S,T)$ can be obtained with the number of transposition breakpoints $b_t(S,T)$.

Theorem 6. *[17] Given two binary strings S and T, we have*

$$\lceil b_t(S,T)/3 \rceil \le d_t(S,T) \le \lfloor n/2 \rfloor.$$

4.5 Non-overlapping Inversions by Schoniger and Waterman

Because the problem with overlapping reversals and transpositions is NP-hard, in 1992, Schoniger and Waterman [51] made some restrictions on the operations of the sequence alignment problem. Their operations include character insertion, deletion and replacement, and non-overlapping inversions. They presented a DP algorithm for the sequence alignment on DNA sequence with $O(n^6)$ time [51].

An inversion $\theta(i,j)$ on a sequence A is to reverse the substring $A_{i..j}$ and to replace each element of $A_{i..j}$ with its complement. Note that a DNA sequence consists of four characters a, t, c and g. Besides, a is complementary to t with each other, and c is complementary to g with each other. For example, suppose $A =$ `taccgtca`. Then, $B = A \cdot \theta(4,7) =$ `tacgacga`. Let $A = a_1 a_2 \cdots a_n$

Table 10: An example for the algorithm of Schoniger and Waterman [51] with $A =$ `taccgtca` and $B =$ `gatcacgga`.

	-	g	a	t	c	a	c	g	g	a
-	0	0	0	0	0	0	0	0	0	0
t	0	0	0	1	0	0	0	0	0	0
a	0	0	1	0	0	1	0	0	0	1
c	0	0	0	0	1	0	2	1	0	0
c	0	0	0	0	1	0	1	2	1	0
g	0	1	0	0	0	0	0	2	3	2
t	0	0	1	1	0	0	0	1	3	3
c	0	0	1	0	2	0	1	0	2	2
a	0	0	1	1	1	3	2	1	3	3

and $B = b_1 b_2 \cdots b_n$ be the resulting sequence of $\theta(i, j)$ on A. The formal transformation is given as follows.

$$b_t = \begin{cases} \overline{a_{i+j-t}} & \text{if } i \le t \le j, \\ a_t & \text{otherwise.} \end{cases} \tag{28}$$

The operation of non-overlapping inversions means that the intervals of two inversions cannot overlap. For example, $\theta(3, 5)$ and $\theta(5, 7)$ have an overlap on a_5. They defined $Z(g, h; i, j)$ to represent the local alignment score of $A_{g..i}$ and $B_{h..j}$ after applying an inversion $\theta(h, j)$ on B. They defined two matching functions $d_1(a_i, b_j)$ and $d_2(a_i, b_j)$, and two gap functions $w_1(k) = \alpha_1 + \beta_1 k$ and $w_2(k) = \alpha_2 + \beta_2 k$, where k is the length of consecutive gaps. The cost function $w_1(k)$(insertion and deletion) and $d_1(a_i, b_j)$(replacement) are used to calculate $Z(g, h; i, j)$(local alignment), and $w_2(k)$ and $d_2(a_i, b_j)$ are used for the entire algorithm. The cost of one inversion is γ. Their algorithm calculates mn cells, each of which takes $O(m^2 n^2)$ time to find the best inversion of $B_{h..j}$ to align with $A_{g..i}$. Thus, the total time complexity of their algorithm is $O(n^6)$ when $m = n$.

An example is shown in Table 10, where $\gamma = -1$, $w_1(k) = w_2(k) = 0 - k$, and $d_1(a_i, b_j) = d_2(a_i, b_j) = 1$ if $a_i = b_j$, otherwise -2. Let $M_{sw2}[i, j]$ denote the best alignment score of $A_{1..i}$ and $B_{1..j}$. $M_{sw2}[8, 8]$ involves a score from an inversion of $A_{3..8} =$ `ccgtca` versus $B_{3..8} =$ `tcacgg`, In more detail, $M_{sw2}[8, 8] = M_{sw2}[2, 2] + Z(3, 3; 8, 8) + \gamma = 1 + 3 - 1 = 3$, where $Z(3, 3; 8, 8)$ is the alignment score of $A_{3..8} =$ `ccgtca` and $\overline{B_{3..8}} =$ `ccgtga`.

4.6 Non-overlapping Inversions and Transpositions by Ta *et al.*

The operations involved in the problem solved by Schoniger and Waterman [51] are character insertions, deletions, replacements and non-overlapping inversions. Since the algorithm of Schoniger and Waterman [51] needs $O(n^6)$ time, in 2016, Ta *et al.* [59] made more restrictions on the operations, including only non-overlapping inversions and non-overlapping transpositions, but without character insertions, deletions and replacements. In this situation, the two input strings should of the same length, that is $|A| = |B|$. They presented an algorithm for solving the *non-overlapping inversion and transposition distance problem* with $O(n^3)$ time and $O(n^2)$ space [59].

For the non-overlapping meaning here, for example, if an inversion $\theta(3, 5)$ (applied on interval [3,5]) is applied to sequence A, no operations can be applied on a part of interval [3,5] of A, such as inversion $\theta(1, 3)$ or transposition $\tau(5, 7, 10)$ (swapping $A_{5..6}$ and $A_{7..9}$).

Their algorithm uses two *mutation tables* M_{ta1} and M_{ta2} as tools to calculate all possible inversions and transpositions, respectively. $M_{ta1}[i, j] = \overline{a_j}$ and $M_{ta2}[i, j] = a_j$. Table 11 shows an example of the two mutation tables. As one can see, an inversion $\theta(1, 3)$ on $A_{1..3}$ is equal to $B_{1..3} =$ `gat`. $M_{ta1}[1, 3]$, $M_{ta1}[2, 2]$ and $M_{ta1}[3, 1]$ form a slash line whose content is equal to $B_{1..3}$. An inversion $\theta(5, 6)$ on $A_{5..6}$ is equal to $B_{5..6} =$ `ct`. $\tau(5, 7, 10)$ on $A_{5..9}$ (swapping $A_{5..6}$ and $A_{7..9}$) is equal to $B_{5..9} =$ `ctagg`. $M_{ta2}[7, 5]$, $M_{ta2}[8, 6]$, and $M_{ta2}[9, 7]$ forms a backslash line whose content is equal to $B_{5..7} =$ `agg`, $M_{ta2}[5, 8]$ and $M_{ta2}[6, 9]$ forms a backslash line whose content is equal to $B_{8..9} =$ `ct`. Note that a_4 is not covered by any mutation operation because $a_4 =$ `t` $=$ `b`$_4$.

With the mutation table M_{ta1}, whether the inversion of a substring $A_{i..j}$, for all $1 \le i \le j \le n$ is equal to $B_{i..j}$ can be checked in $O(n^2)$ time. However, to check all possible transpositions with the mutation table M_{ta2} needs $O(n^3)$ time, since there are $O(n)$ pairs of slash lines may form a transposition on the same substring. Thus, the total time complexity is $O(n^3)$ and space complexity is $O(n^2)$.

In 2017, Hsu [30] proposed a more efficient algorithm to solve the same problem as Ta *et al.* solved. His algorithm is based on the *run structure* of a string, proposed by Kolpakov and Kucherov [38]. The time complexiy of his algorithm is re-

Table 11: An example of mutation tables M_{ta1} and M_{ta2}, where $A =$ `atctaggct` and $B =$ `gatactagg`.

$M_{ta1}[i,j]$

	1	2	3	4	5	6	7	8	9
1	t	a	**g**	a	t	c	c	g	a
2	t	**a**	g	a	t	c	c	g	a
3	**t**	a	g	a	t	c	c	g	a
4	t	a	g	a	t	c	c	g	a
5	t	a	g	a	t	**c**	c	g	a
6	t	a	g	a	**t**	c	c	g	a
7	t	a	g	a	t	c	c	g	a
8	t	a	g	a	t	c	c	g	a
9	t	a	g	a	t	c	c	g	a

(a)

$M_{ta2}[i,j]$

	1	2	3	4	5	6	7	8	9
1	a	t	c	t	a	g	g	c	t
2	a	t	c	t	a	g	g	c	t
3	a	t	c	t	a	g	g	c	t
4	a	t	c	t	a	g	g	c	t
5	a	t	c	t	a	g	g	**c**	t
6	a	t	c	t	a	g	g	c	**t**
7	a	t	c	t	**a**	g	g	c	t
8	a	t	c	t	a	**g**	g	c	t
9	a	t	c	t	a	g	**g**	c	t

(b)

duced to $O(n^2)$.

5 Conclusions and Future Work

The traditional edit distance problem is to find the minimum edit distance between two input sequences (strings) with the edit operations, including character insertions, deletions and replacements. We use the traditional edit distance problem as the baseline to compare with other variants in three aspects: input, allowed operations and output. First, from the input aspect, the edit distance for cyclic strings is different because one of the input strings is viewed as a set of strings by rotation. The edit distance of RLE strings can be seen as a special case that the same characters are usually consecutive. Thus, the edit distance for RLE strings and genome rearrangement are the same as the traditional problem in the input aspect.

Second, from the allowed operations aspect, the block edit distance and genome rearrangement problems are different from the traditional edit distance problem as operations can be performed on substrings in the former problems. However, the operations on cyclic strings are the same as the traditional problem.

Third, from the output aspect, all of these problems desire to find the minimum cost required for transforming a string into another string. Thus, these problems are the same in the output aspect.

With these three aspects, we give the comparison of the variants to the traditional edit distance problem, as shown in Table 12. Furthermore, the characteristics of these variants lead them to be useful in different applications. For example, the edit distance for cyclic strings can be applied to pattern recognition, because the string which represents patterns like a polygon can be viewed as the same after rotation. The edit distance for RLE strings can be applied to compare the compressed information directly. The block edit distance can simulate human editing behaviors to compare two documents. And the block edit distance can also be used to compare DNA sequences.

The genome rearrangement problem simulates the mutation of genomes and finds the relations between genomes. With the same idea, we can use different operations to simulate human behaviors or possible operations to transform or destroy the data and then to recover the original information. For example, we can simulate how humans modify articles by string copies and deletions. We could also find out how the data in some special formation, such as images and compressed files, were destroyed in transmission.

We hope that this paper is useful for those who study or do not study in this field. Especially, it is helpful to understand the key points of these algorithms. In the future, we will survey the related longest common subsequence (LCS) problem and its variants.

References

[1] A. Aggarwal, M. M. Klawe, S. Moran, P. Shor, and R. Wilber, "Geometric applications of a matrix-searching algorithm," *Algorithmica*, Vol. 2, No. 1, pp. 195–208, 1987.

[2] A. V. Aho, D. S. Hirschberg, and J. D. Ullman, "Bounds on the complexity of the longest common subsequence problem,"

Table 12: The variants compared to the traditional edit distance problem in three aspects.

Problem	Input	Allowed operations	Output
Edit distance for cyclic strings	different	same	same
Edit distance for RLE strings	same	same	same
Block edit distance	same	different	same
Genome rearrangement	same	different	same

Journal of the ACM, Vol. 23, No. 1, pp. 1–12, 1976.

[3] H. Y. Ann, C. B. Yang, Y. H. Peng, and B. C. Liaw, "Efficient algorithms for the block edit problems," *Information and Computation*, Vol. 208(3), pp. 221–229, 2010.

[4] H. Y. Ann, C. B. Yang, C.-T. Tseng, and C. Y. Hor, "A fast and simple algorithm for computing the longest common subsequence of run-length encoded strings," *Information Processing Letters*, Vol. 108, pp. 360–364, 2008.

[5] H. Y. Ann, C. B. Yang, C.-T. Tseng, and C.-Y. Hor, "Fast algorithms for computing the constrained lcs of run-length encoded strings," *Theoretical Computer Science*, Vol. 432, pp. 1–9, 2012.

[6] O. Arbell, G. M. Landau, and J. S. B. Mitchell, "Edit distance of run-length encoded strings," *Information Processing Letters*, Vol. 83, No. 6, pp. 307–314, 2002.

[7] V. Bafna and P. A. Pevzner, "Genome rearrangements and sorting by reversals," *SIAM Journal of Computing*, Vol. 25, No. 2, pp. 172–289, 1996.

[8] V. Bafna and P. A. Pevzner, "Sorting by transpositions," *SIAM Journal on Discrete Mathematics*, Vol. 11, No. 2, pp. 224–240, 1998.

[9] A. Bergeron, J. Mixtacki, and J. Stoye, "Reversal distance without hurdles and fortresses," *Proceedings of 15th Annual Combinatorial Pattern Matching Symposium*, Vol. 3109, Istanbul, Turkey, pp. 389–399, 2004.

[10] P. Berman and S. Hannenhalli, "Fast sorting by reversal," *Proceedings of the 7th Annual Symposium on Combinatorial Pattern Matching*, London, UK, pp. 168–185, 1996.

[11] P. Berman, S. Hannenhalli, and M. Karpinski, "1.375-approximation algorithm for sorting by reversals," *Proceedings of the 10th Annual European Symposium on Algorithms*, Vol. 2461, Rome, Italy, pp. 200–210, 2002.

[12] H. Bunke and J. Csirik, "An algorithm for matching run-length coded strings," *Computing*, Vol. 50, No. 4, pp. 297–314, 1993.

[13] H. Bunke and J. Csirik, "An improved algorithm for computing the edit distance of run-length coded strings," *Information Processing Letters*, Vol. 54, No. 2, pp. 93–96, 1995.

[14] A. Caprara, "Sorting by reversals is difficult," *Proceedings of the First International Conference on Computational Molecular Biology*, NM, USA, pp. 75–83, 1997.

[15] A. Caprara, "Sorting permutations by reversals and eulerian cycle decompositions," *SIAM Journal of Discrete Mathematics*, Vol. 12, No. 1, pp. 91–100, 1999.

[16] D. A. Christie, "A 3/2-approximation algorithm for sorting by reversals," *Proceeding of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms*, Pennsylvania, USA, pp. 244–252, 1998.

[17] D. A. Christie and R. W. Irving, "Sorting strings by reversals and by transpositions," *SIAM Journal on Discrete Mathematics*, Vol. 14, No. 2, pp. 193–206, 2001.

[18] G. Cormode and S. Muthukrishnan, "The string edit distance matching problem with moves," *Proceedings of the 13th annual ACM-SIAM symposium on Discrete algorithms*, San Francisco, USA, pp. 667–676, 2002.

[19] N. EI-Mabrouk, "Reconstructing an ancestral genome using minimum segments duplications and reversals," *Journal of Computer and System Sciences*, Vol. 65, No. 3, pp. 442–464, 2002.

[20] D. Eppstein, "Sequence comparison with mixed convex and concave cost," *Journal of Algorithms*, Vol. 11, pp. 85–101, 1990.

[21] F. Ergun, S. Muthukrishnan, and C. Sahinalp, "Comparing sequences with segment rearrangements," *Proceedings of the 23rd Foundations of Software Technology and Theoretical Computer Science*, Mumbai, India, pp. 183–194, 2003.

[22] Z. Galil and R. Giancarlo, "Speeding up dynamic programming with application to

molecular biology," *Theoretical Computer Science*, Vol. 64, pp. 107–118, 19889.

[23] M. R. Garey and D. S. Johnson, "Complexity results for multiprocessor scheduling under resource constraints," *SIAM Journal on Computing*, Vol. 4, pp. 397–411, 1975.

[24] O. Gotoh, "An improved algorithm for matching biological sequences," *Journal of Molecular Biology*, Vol. 162, No. 3, pp. 705–708, 1982.

[25] J. Gregor and M. G. Thomason, "Dynamic programming alignment of sequences representing cyclic patterns," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 15, No. 2, pp. 129–135, 1993.

[26] S. Hannenhalli, "Polynomial-time algorithm for computing translocation distance between genomes," *Proceedings of the 6th Annual Symposium on Combinatorial Pattern Matching*, Espoo, Finland, pp. 162–176, 1996.

[27] S. Hannenhalli and P. A. Pevzner, "Transforming cabbage into turnip: polynomial algorithm for sorting signed permutations by reversals," *Proceedings of the 27th Annual Symposium on Theory of Computing*, New York, USA, pp. 178–189, 1995.

[28] T. Hartman and R. Shamir, "A simpler 1.5-approximation algorithm for sorting by transpositions," *Proceedings of the 14th Annual Symposium on Combinatorial Pattern Matching*, Morelia, Mexico, pp. 156–169, 2003.

[29] D. S. Hirschberg, "A linear space algorithm for computing maximal common subsequence," *Communications of the ACM*, Vol. 24, pp. 664–675, 1975.

[30] T.-C. Hsu, *An Algorithm for Computing the Distance of the Non-overlapping Inversion and Transposition*. National Sun Yat-sen University, Master's Thesis, Kaohusing, Taiwan, 2017.

[31] T. Jiang, G. Lin, B. Ma, and K. Zhang, "A general edit distance between RNA structures," *Journal of Computational Biology*, Vol. 9, No. 2, pp. 371–388, 2002.

[32] H. Kaplan and N. Shafrir, "The greedy algorithm for edit distance with moves," *Information Processing Letters*, Vol. 97, No. 1, pp. 23–27, 2006.

[33] J. Kececioglu and D. Sankoff, "Exact and approximation algorithms for the inversion distance between two permutations," *Proceedings of the 4th Annual Symposium on Combinatorial Pattern Matching*, Vol. 684, Berlin, Germany, pp. 87–105, 1993.

[34] J. Kececioglu and D. Sankoff, "Exact and approximation algorithms for sorting by reversals, with application to genome rearrangement," *Algorithmica*, Vol. 13, No. 1-2, pp. 180–210, 1995.

[35] J. W. Kim, A. Amir, G. M. Landau, and K. Park, "Computing similarity of run-length encoded strings with affine gap penalty," *Theoretical Computer Science*, Vol. 395, No. 2-3, pp. 268–282, 2008.

[36] M. M. Klawe and D. J. Kleitman, "An almost linear time algorithm for generalized matrix searching," *SIAM Journal on Discrete Mathematics*, Vol. 3, pp. 81–97, 1990.

[37] P. Kolman, "Approximating reversal distance for strings with bounded number of duplicates in linear time," *Proceedings of 30th International Symposium on Mathematical Foundations of Computer Science*, Gdansk, Poland, pp. 580–590, 2005.

[38] R. Kolpakov and G. Kucherov, "Finding maximal repetitions in a word in linear time," *Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, New York, USA, pp. 596–604, IEEE Computer Society, 1999.

[39] J. J. Liu, G. S. Huang, Y. L. Wang, and R. C. T. Lee, "Edit distance for a run-length-encoded string and an uncompressed string," *Information Processing Letters*, Vol. 105, No. 1, pp. 12–16, 2007.

[40] J. J. Liu, Y. L. Wang, and R. C. T. Lee, "Finding a longest common subsequence between a run-length-encoded string and an uncompressed string," *Journal of Complexity*, Vol. 24, No. 2, pp. 173–184, 2008.

[41] D. Lopresti and A. Tomkins, "Block edit models for approximate string matching," *Theoretical Computer Science*, Vol. 181, pp. 159–179, 1997.

[42] R. Lowrance and R. A. Wagner, "An extension of the string-to-string correction problem," *Journal of the ACM*, Vol. 22, No. 2, pp. 177–188, 1975.

[43] M. Maes, "On a cyclic string-to-string correction problem," *Information Processing Letters*, Vol. 35, pp. 73–78, 1990.

[44] A. Marzal and S. Barrachina, "Speeding up the computation of the edit distance for cyclic strings," *In Proceeding of the 15th International Conference on Pattern Recognition*, Barcelona, Spain, pp. 895–898, 2000.

[45] W. J. Masek and M. S. Paterson, "A faster algorithm computing string edit distances,"

Journal of Computer and System Sciences, Vol. 20, pp. 18–31, 1980.

[46] W. Miller and E. W. Myers, "Sequence comparison with concave weighting functions," *Bulletin of Mathematical Biology*, Vol. 50, No. 2, pp. 97–120, 1988.

[47] S. Muthukrishnan and S. C. Sahinalp, "Approximate nearest neighbors and sequence comparison with block operations," *Proceedings of the 32nd Symposium on the Theory of Computing*, Portland, USA, pp. 416–424, 2000.

[48] E. W. Myers, "An $O(ND)$ difference algorithm and its variations," *Algorithmica*, No. 1, pp. 251–266, 1986.

[49] D. B. Needleman and C. D. Wunsch, "A general method applicable to the search for similarities in the amino acid sequence of two proteins," *Journal of Molecular Biology*, Vol. 48, No. 3, pp. 443–453, 1970.

[50] Y. H. Peng and C. B. Yang, "Finding the gapped longest common subsequence by incremental suffix maximum queries," *Information and Computation*, Vol. 237, pp. 95–100, 2014.

[51] M. Schoniger and M. S. Waterman, "A local algorithm for DNA sequence alignment with inversions," *Bulletin of Mathematical Biology*, Vol. 54, No. 4, pp. 521–536, 1992.

[52] P. H. Sellers, "An algorithm for the distance between two finite sequences," *Journal of Combinatorial Theory*, Vol. 16, pp. 253–258, 1974.

[53] P. H. Sellers, "On the theory and computation of evolutionary distances," *SIAM Journal on Applied Mathematics*, Vol. 26, No. 4, pp. 787–793, 1974.

[54] D. Shapira and J. A. Storer, "Large edit distance with multiple block operations," *Proceedings of 10th International Symposium, String Processing and Information Retrieval*, Vol. 2857, Manaus, Brazil, pp. 369–377, 2003.

[55] D. Shapira and J. A. Storer, "Edit distance with move operations," *Journal of Discrete Algorithms*, Vol. 5, No. 2, pp. 380–392, 2007.

[56] T. F. Smith and M. S. Waterman, "Comparison of biosequences," *Advances in Applied Mathematics*, Vol. 2, pp. 482–489, 1981.

[57] T. F. Smith and M. S. Waterman, "Identification of common molecular subsequences," *Journal of Molecular Biology*, Vol. 147, No. 1, pp. 195–197, 1981.

[58] T. F. Smith, M. S. Waterman, and W. M. Fitch, "Comparative biosequence metrics," *Journal of Molecular Mubon*, Vol. 18, pp. 38–46, 1981.

[59] T. T. Ta, C. Y. Lin, and C. L. Lu, "An efficient algorithm for computing non-overlapping inversion and transposition distance," *Information Processing Letters*, Vol. 116, pp. 744–749, 2016.

[60] E. Ukkonen, "Algorithms for approximate string matching," *Information and Control*, Vol. 64, pp. 100–118, 1985.

[61] R. A. Wagner, "On the complexity of the extended string-to-string correction problem," *Proceedings of Seventh Annual ACM Symposium on Theory of Computing*, New York, USA, pp. 218–223, 1975.

[62] R. A. Wagner and M. J. Fischer, "The string-to-string correction problem," *Journal of the ACM*, Vol. 21, No. 1, pp. 168–173, 1974.

[63] M. E. M. T. Walter, Z. Dias, and J. Meidanis, "A new approach for approximating the transposition distance," *Proceedings of the Seventh International Symposium on String Processing Information Retrieval*, Corunna, Spain, pp. 199–208, 2000.

[64] M. S. Waterman, "Efficient sequence alignment algorithms," *Journal of Theoretical Biology*, Vol. 108, pp. 333–337, 1984.

[65] M. S. Waterman, T. F. Smith, and W. Beyer, "Some biological sequence metrics," *Advances in Mathematics*, Vol. 20, No. 3, pp. 367–387, 1976.

[66] R. Wilber, "The concave least-weight subsequence problem revisited," *Journal of Algorithms*, Vol. 9, pp. 418–425, 1988.

[67] C. K. Wong and A. K. Chandra, "Bounds of the string editing problem," *Journal of the ACM*, Vol. 23, pp. 13–16, 1976.

[68] S. Wu, U. Manber, G. Myers, and W. Miller, "An O(NP) sequence comparison algorithm," *Information Processing Letters*, Vol. 35, pp. 317–323, 1990.

[69] F. F. Yao, "Speed-up in dynamic programming," *SIAM Journal on Algebraic Discrete Methods*, Vol. 3, pp. 532–540, 1982.

使用基因演算法解 *p*-中心位點問題

Using Genetic Algorithm to solve the *p*-centdian problem

Tsai Chueh Wang[1], Yen Hung Chen[2], Kuan Wen Wang[3]

Department of Computer Science,
University of Taipei, Taiwan
peter771109@gmail.com[1] , yhchen@utaipei.edu.tw[2], das781016@gmail.com[3]

摘要

給定一個無向圖$G(V,E,l)$，其中 l 是一個非負數函數表示該圖中每一個邊的長度，和一個正整數p，$0<p<|V|$，對於一個p個節點的集合V'，$|V'|=p$，$d_G(v,V')$為節點v與集合V'內最近節點的距離。在一個圖中，V'的離心率$£_C(V')$定義為每個節點 v 離 V' 的距離中最遠的數值(即，$£_C(V')=max_{v\in V} d_G(v,V')$)。$V'$的中位距離則定義為每個節點$v$離$V'$的距離值的總和(即，$£_M(V')=\sum_{v\in V} d_G(v,V')$)。$p$-中心位點問題($p$-centdian problem)定義為在圖$G$中找一個$p$個節點的集合$P$，使得加總$£_C(P)$和$£_M(P)$要最小。$p$-中心位點問題不難證明為NP-hard。本研究為此問題設計一個基因演算法(Genetic algorithm)來加快其執行時間以及透過TSPLibrary(TSP-Lib)中的資料，其節點個數為22~152範圍間，來模擬測試此技術的倍率及時間。實驗結果顯示基因演算法所得的解與最佳解的倍率大約為1.35至1.6之間，最高到3.36倍。在執行效率上，我們的基因演算法所用之時間平均為1.34至1.57秒（s）。

1 緒論

在組合最佳化問題中，*瓶頸問題*(*bottleneck problems*，或稱為 *min-max problems*)及*加總問題*(*min-sum* problems) 是兩種常用的*準則(criteria)*在許多通訊網路的設計(communication network design)、作業研究中企業或工廠上的倉儲(warehouse)及配送中心等服務規劃(service planning)、交通運輸網路上的緊急設施規劃(emergency facility planning)、工作排程(scheduling)、資源管理(resource management)、容量規劃(capacity planning)等應用上 [1]。瓶頸問題的顯著特性是會存在一個最差的(worst case)值或條件(稱 bottleneck)在整個(網路或企業)環境中，我們的目的是希望找到最差的情況(bottleneck)要最小。一個非常著名的瓶頸問題即為*中心點問題*(*center problem*)。中心點問題最早可追朔到在 1869 年 Jordan [2]提出圖論上的中心點問題：在圖中找到一個節點(中心點)，使得圖中剩下來的節點可透過最短路徑連到此中心點最遠的距離要最小(即為 *1-center problem*)。1965 年 Hakimi [3] 擴展 1-center problem 到 *p-center problem*。給定一個無向圖 $G(V,E,l)$，其中 l 是一個非負數函數表示該圖中每一個邊的長度和一個正整數 p，$0<p<|V|$，對於一個 p 個節點的集合 V'，$|V'|=p$，$d_G(v,V')$為節點v與集合V'內最近節點的距離。如果節點 v 為 V'內的一個節點，則 $d_G(v,V')=0$。在一個圖中，V'的離心率$£_C(V')$定義為每個節點 v 離 V'的距離中最遠的數值(即，$£_C(V')=max_{v\in V} d_G(v,V')$)。*p-中心點問題*(*p-center* (*pC*) *problem*)定義為在圖 G 中找到一個 p 個節點的集合 P，使得 P 的離心率要最小 [3]。*中位點問題*(*median problem*)類似於中心點問題，但要求的準則為最小加總(min-sum)。中位點問題是希望在圖中找到一個節點(中位點)，使得圖中每個剩下的節點可透過最短路徑連到此中位點的距離加總要最小(即為 *1-median problem*)。Hakimi [3] 同樣於 1965 年將問題擴展到 *p-median problem*。給定一個無向圖 $G(V,E,l)$，其中 l 是一個非負數函數表示該圖中每一個邊的長度和一個正整數 p，$0<p<|V|$，對於一個 p 個節點的集合 V''，V''的中位距離則定義為每個節點 v 離 V''的距離值的總和(即，$£_M(V'')=\sum_{v\in V} d_G(v,V'')$)。*p-中位點問題*(*p-median* (*pM*) *problem*)定義為在圖 G 中找到一個 p 個節點的集合 P，使得 P 的中位距離要最小 [3]。*pC* 及 *pM* 問題有可應用在設備配置(facility location) [3-18]與社交網路(social network)上 [19]。Halpern [20-21]提出了一種方式要同時滿足最大傳輸(延遲)時間及整體傳輸(延遲)時間的一種*雙準則(bicriteria)*問題：*中心位點*(*centdian*)問題。Hooker 等人 [22] 擴展 *centdian* 問題到 *p-centdian problem*。給定一個無向圖 $G(V,E,l)$，其中 l 是一個非負數函數表示該圖中每一個邊的長度，和一個正整數 p，$0<p<|V|$，*p-中心位點問題*(*p-centdian* (*pD*) *problem)*定義為在圖 G 中找一個 p 個節點的集合 P(稱為 p-中心位點)，使得加總$£_C(P)+£_M(P)$要最小 [22]。因為 *pC* 及 *pM* 問題為 NP-hard，不難證明 *pD* 問題也是 NP-hard 就算輸入的圖形是 metric graphs（完全圖並滿足三角不等式）下。在圖形為 metric graphs 下，Tamir 等人 [23]認為(只用一句話帶過)透過 Bartal [24,25]提出的在解 *p*M 問題的 randomized 演算法下，會有一個期望值為 $O(log\ |V|\ log\ log\ |V|)$的近似演算法解在 *pD* 問題上。

Kalcsics, Nickel [26]則是提出了一個 $O(log\ |V|\ log\ log\ |V|)$的近似演算法解 *p*-facility ordered median problems（此問題不等同於我們提出的 *pM* 及 *pD* 問題）。因此目前並無任何 deterministic approximation algorithm 的論文被提出對於 *pD* 問題。Brito 和 Perez [27]設計了一個 $O(|V|^{p+2}|E|^p)$的正確演算法來找出 generalized *pD* 問題得最佳解。

針對 *pD* 問題，我們在去年發表了如何使用整數規劃和線性規劃鬆弛來解決這個問題[32]，在本論文則提出使用基因演算法來找到 *pD* 問題的最佳解。接著我們再跟線性規劃鬆弛(Linear programming relaxation)求得的 *p*D 問題的合法解及整數規劃產生的最佳解進行比較，觀察所得到的解與最佳解之間的關係(我們設計的方法產生的合法解的數值除以整數規劃產生的最佳解的數值)。我們的四個方法將使用 TSPLibrary(TSP-Lib) [28]資料進行模擬。

本論文第二節我們將描述我們的基因演算法來解決 *pD* 問題。第三節利用我使用一些 TSP-Lib 資料對第二節的演算法進行模擬(simulation)測試並觀察我們設計的演算法的效率以及正確性。*pD* 問題的未來研究方向則在第四節呈現。

2 基因演算法對於 *p*-centdian 問題

本節中，我們設計一個基因演算法來找出問題合法解。我們令 *n* 為圖 *G* 中的節點個數|*V*|。*P* 為 *p*D 問題最佳解(*p* 個節點的集合)。對於 *p*D 問題的一個合法解 P_f，如果節點 *v* 連結到離 P_f 內距離最近的節點 *u*，我們稱節點 *v* 被 *u* 管(*assign*)。

底下我們敘述我們的基因演算法的設計過程。

基因演算法由 Holland 所提出[31]，其中三項操作方法分別為交配(cross-over)、反轉(inversion)和突變(mutation)，經過各界多年持續研究與應用，在方法的細部調整與加入新步驟強化讓基因演算法能有更完整的架構，以下我們說明如何設計 *p*D 問題的基因演算法：

基因組：

從一個族群的一組個體開始，每一個個體都是待解決問題的一個候選解。個體以一組參數(節點個數 n)為特徵，這些特徵被稱做基因，串連這些基因就可以組成染色體(問題的解)。

單個個體的基因組我們使用二進制編碼，一個二進制串代表一條染色體串。我們給圖 *G* 中的節點個數|*V*|加以編號，隨機挑選合法解 P_f 的節點，如此便完成一組基因組。對於所有的 $i, 1\le j\le n$，變數 y_j 對應到圖中的每個節點 *j*，如果節點 *j* 是在合法解 P_f 內，則對應的 y_j 設為 *1*，否則為 *0*。

適應函數設計(fitness funtion)：

適應函數的目的在評估該個體對環境的適應度（與其它個體競爭的能力）。每一個體都有適應度評分，個體被選中進行繁殖的可能性取決於其適應度評分。適應度評分越高代表該基因組越能適應環境，藉此能得到最佳解 *P*，所以適應函數的設計要符合求解問題本身的要求。

pD 問題的適應函數：

$$F=£_C(P)+£_M(P)=\Sigma_{i=1\ to\ n}\Sigma_{j=1\ to\ n}\ l_{ij}x_{ij}+z\text{------(18)}$$

(1) 對於所有的 $i, j, 1\le i\le n, 1\le j\le n$，變數 l_{ij} 為一個 2 維矩陣紀錄節點 *i* 到節點 *j* 的最短路徑，l_{ij} 計算可透過 Floyd-Warshall Algorithm 解 all-pair shortest path problem 得到此矩陣的數值 [29]。

(2) 對於所有的 $i, j, 1\le i\le n, 1\le j\le n$，變數 x_{ij} 為一個 2 維矩陣，如果節點 *i* 是被節點 *j* 管(assign)，則 x_{ij} 設為 *1*，否則為 *0*。

(3) 變數 *z*，為一個合法解 P_f 的離心率 $£_C(P_f)$

選擇(selection)：

進行選擇運算是為了提高更能適應環境的子代機率，並使它們將基因傳到下一代中。基於其適應度評分，我們選擇多對較優良的個體（父母）。適應度高的個體更易被選中繁殖，即將較優父母的基因傳遞到下一代。

在每一代的進化中，我們選擇將適應度評分較低的 30%的個體移除。

交配(cross-over)：

交配是基因演算法中最重要的階段，將父母的兩組染色體的基因進行切段、互換、重組等，藉由此方式來產生新的子代。

交配方法有單點法(single-point crossover)、雙點法(two-point crossover)、均勻雜交法(uniform crossover)等，而我們使用了均勻雜交法(uniform crossover)。

突變(mutation)：

突變是為了避免族群裡的染色體過於相近而陷入局部最佳解，因此適當的突變函數能夠跳出交配的局限，使演化出來的新子代更有多樣性。

突變的方式有單點突變法(single-point mutation)、倒置法(inversion)等，在本研究中我們使用單點突變法(single-point mutation)，並設定有 10%的機率，隨機挑選基因組內部任兩點進行交換以增加演化的多樣性。

底下為執行步驟：

輸入: 一個無向完全圖 $G=(V,E,l)$，l 是一個非負數函數(需滿足三角不等式)表示該圖中每一個

邊的長度，及一個數值 p（$p \geq 1$）。設定族群個數 N，迭代次數 T，突變機率 M。

輸出: p-中心位點 P，$|P|=p$。

Step 1: 在每一個個體基因組中使用隨機分配選出合法解 P_f 的節點 p，然後生成大小為 N 的族群。

Step 2: 計算每個個體的適應度評分 F，再使用快速排序法進行排序。

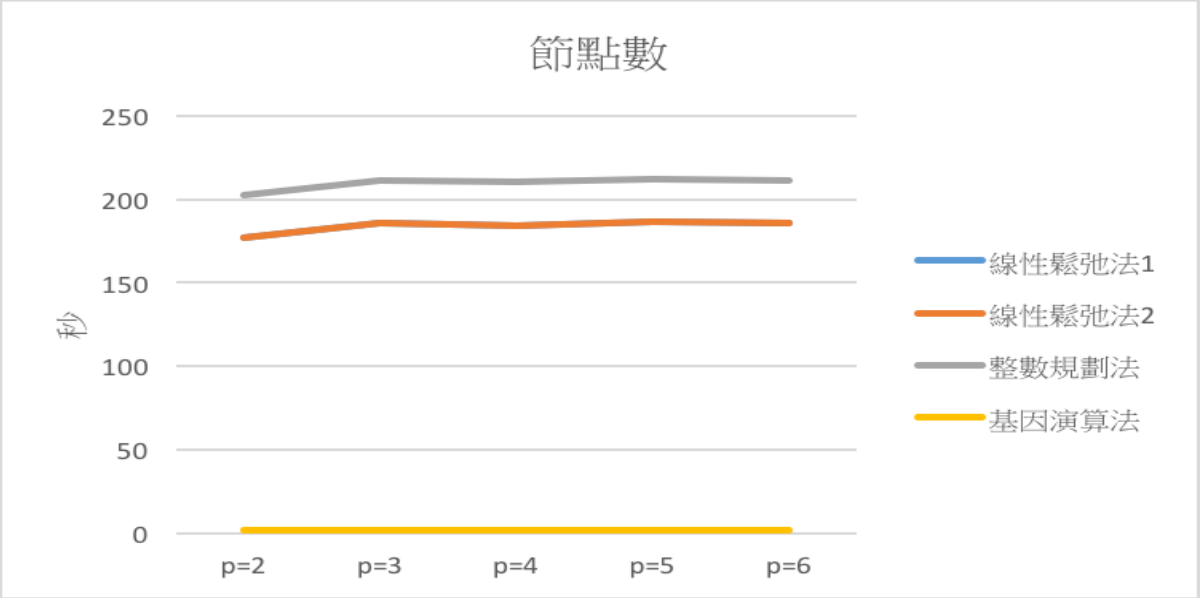
Step 3: 進行族群選擇運算，將排序後適應度評分較低的 30%刪除。

Step 4: 進行交配和突變，產生新的子代到族群個數為 N。

Step 5: 當族群中的個體不產生與前一代差異較大的後代，停止迭代。

Step 6: Return P。

圖 1：輸入為 TSP-Lib 內的完全圖，節點數在 22~152 個範圍間的情形，線性鬆弛演算法 1、線性鬆弛演算法 2(線性鬆弛演算法 1、2 時間相同，故重疊為一條線)、基因演算法與整數規劃演算法之耗時比較（單位：秒）。每個 p 點的值是分別執行 25 組資料後的平均值。



3 實測驗證

本節中，我們將針對第二節所設計的演算法來進行實際的程式模擬。我們藉由基因演算法得到 pD 問題的合法解 P 和對應的最佳合法解的值 $F=£_C(P)+£_M(P)$，將基因演算法的合法解的數值除以整數規劃的最佳解的數值來找出這兩個演算法的倍率。實驗模擬採用 Eclipse 撰寫 Python 程式，作業系統平臺為 64 位元 OS X Yosemite version 10.10.5。處理器使用 Intel Core i5，時脈 2.9GHz，記憶體為 16GB。實驗數據採用 TSPLibrary(TSP-Lib) [28]進行模擬。並用二維陣列儲存無向完全圖之節點及其權重。我們模擬的在 TSP-Lib 選取 25 組資料，因為 TSP-Lib 內的每個圖的節點數大多不一樣，所以這 25 組測試資料我們選擇的圖為節點個數在 22~152 的範圍間。演算法為第二節所提到的基因演算法。正確演算法是利用整數規劃演算法得到的最佳解[32]。p 的設定範圍為 2 到 6 個節點之間，結果為取 25 組測資執行後 $£_C(P)+£_M(P)$ 的平均值。

為了說明我們的演算法所產生的解與最佳解之間並不會相距太遠，表 1.分別列出了線性鬆弛演算法 1、線性鬆弛演算法 2 與基因演算法所產生出的解與最佳解之間的倍率(我們演算法所得到的合法解的數值除以最佳解的數值)。() 內為其標準差。線性鬆弛演算法 1 的倍率幾乎接近 1。透過線性鬆弛演算法 2 在節點數為 2~6 時得到的近似率平均值最高到 1.35，顯示出線性鬆弛演算法 1 有較好的效率(選出線性規劃解中的前 p 大的節點)；而基因演算法在節點數為 2~6 時得到的近似率平均值則更高，達到 1.59。所以線性鬆弛演算法 1 跟最佳解非常接近。在最佳解的結果方面，線性鬆弛演算法 2 會因為 p 節點個數增加而導致最佳解倍率增加，但 p 對線性鬆弛演算法 1 和基因演算法的影響較小，

大致是維持在一定倍率內。而線性鬆弛演算法 1 在整體近似倍率上也能提供比線性鬆弛演算法 2 和基因演算法更接近最佳解。雖然我們未能以數學證明我們的解距離最佳解的倍率，然而實驗數據顯示我們的解不失為一個好的結果。

p	2	3	4	5	6
倍率(線性鬆弛演算法 1)	1.0027 (0.006)	1.0063 (0.031)	1.0052 (0.015)	1.0075 (0.006)	1.0038 (0.004)
倍率(線性鬆弛演算法 2)	1.0343 (0.165)	1.0999 (0.191)	1.2117 (0.348)	1.2524 (0.343)	1.3522 (0.5149)
倍率(基因演算法)	1.4885 (0.288)	1.3584 (0.489)	1.3517 (0.483)	1.4876 (0.602)	1.5890 (0.724)

表 1. pD 問題的倍率模擬結果(找出最佳解分別與線性鬆弛演算法 1、線性鬆弛演算法 2 及基因演算法的解間的差異)，針對輸入為在 TSP-Lib 內的完全圖，節點數在 22~152 的範圍時情形，()內為其標準差。倍率為我們演算法所得到的合法解的數值除以最佳解的數值。每個數據的值是執行 25 組資料後的平均值

在時間分析上，圖 1.可以觀察到線性鬆弛演算法 1、線性鬆弛演算法 2、基因演算法與整數規劃演算法在執行時的效率。橫座標是 p，縱座標是執行的時間(秒)。Note：線性鬆弛演算法 1、2 時間相同，因為我們的這兩個演算法是都透過解線性規劃求解後，對前 y_j 的數值排序，$1 \le j \le n$，差別只在演算法 1 是取出前 p 大的數值而演算法 2 是取出前 p 小的數值。因此這兩個演算法的時間會是一樣。在 p=2 時，整數規劃演算法在執行上平均需花 202.4502 秒、線性鬆弛演算法 1 及 2 的平均時間為 176.6884 秒、基因演算法的平均時間為 1.3497 秒。在 p=6 時，整數規劃演算法平均需花 211.3155 秒、線性鬆弛演算法 1 及 2 平均需花 185.4813 毫秒、基因演算法的平均時間為 1.5714 秒。實驗結果顯示，p 設定為 2 到 6 個節點時，基因演算法的執行時間是最快的，而線性鬆弛演算法 1 和 2 的執行時間是相同的(差在取點方式不同)，但跟整數規劃演算法相比，線性鬆弛演算法 1 和 2 的執行時間較快。線性鬆弛演算法 1 與線性鬆弛演算法 2 的時間平均落在 175 至 187 秒之間。雖然基因演算法的時間遠快於線性鬆弛演算法 1，但在最佳合法解的值上，線性鬆弛演算法 1 的倍率幾乎接近 1，所以各有利弊。

4 未來研究方向

本研究主要是分析及設計基因演算法針對 p-中心位點問題(p-centdian problem)。我們設計了第一個基因演算法，並透過基因演算法找到問題的合法解。實驗顯示我們的基因演算法倍率約為1.35至1.60之間，最高到3.36倍。離最佳解平均倍率在1.6以下，而且執行的時間比整數規劃和線性鬆弛演算法1、2找最佳解的演算法快。未來我們希望能針對基因演算法在選擇、交配和突變三個步驟中，嘗試不同的方法，近一步提升最佳合法解與最佳解的倍率。

Acknowledgements

This work was supported in part by the Ministry of Science and Technology, Taiwan, under Contract MOST 105-2221-E-845-002 。 Corresponding author: Yen Hung Chen。

Reference

[1] D.S. Hochbaum, A. Pathria, Generalized p-center problems: complexity results and approximation algorithms. *European Journal of Operational Research 100, 594–607, 1997.*

[2] C. Jordan, Sur les assemblages des lignes. *Journal für die reine und angewandte Mathematik 70, 185-190, 1869.*

[3] S.L. Hakimi, Optimal distribution of switching centers in a communication network and some related theoretic graph theoretic problems. *Operations Research 13, 462–475, 1965.*

[4] B. Ben-Moshe, A. Dvir, M. Segal, A. Tamir, Centdian computation for sensor networks. In Proceedings of *the 7th Annual Conference Theory and Applications of Models of Computation, LNCS 6108, 187–198, 2010.*

[5] B. Ben-Moshe, A. Dvir, M. Segal, A. Tamir, Centdian computation in cactus graphs. *Journal of Graph Algorithms and Applications 16, 199–224, 2012.*

[6] G.G. Cornuéjols, G.L. Nemhauser, L.A. Wolsey, The uncapacitated facility location problem. In P.Mirchandani and R. Francis, editors, *Discrete Location Theory, John Wiley and Sons, Inc., New York, 119–171, 1990.*

[7] M.S. Daskin, Network and Discrete Location: Models Algorithms and Applications. Wiley, New York, *1995*.

[8] M.S. Daskin, What you should know about location modeling. *Naval Research Logistics 55, 283–294, 2008*.

[9] Z. Drezner, H.W. Hamacher, Facility Location Applications and Theory. 2nd edition, Springer, *2004*.

[10] D.S. Hochbaum, D.B. Shmoys, A unified approach to approximation algorithms for bottleneck problems. *Journal of the ACM 33, 533–550, 1986*.

[11] D.S. Hochbaum, A. Pathria, Generalized *p*-center problems: complexity results and approximation algorithms. *European Journal of Operational Research 100, 594–607 , 1997*.

[12] O. Kariv, S.L. Hakimi, An algorithmic approach to network location problems I: the *p*-centers. *SIAM Journal of Applied Mathematics 37, 514–538, 1979*.

[13] O. Kariv, S.L. Hakimi, An algorithmic approach to network location problems II: the *p*-medians. *SIAM Journal of Applied Mathematics 37, 539–560, 1979*.

[14] J. Mihelič, B. Robič, Facility location and covering problems. In Proceedings of *the 7th International Multiconference Information Society, vol. D–Theoretical Computer Science, Ljubljana, Slovenia, 2004*.

[15] P.B. Mirchandani, R.L. Francis, Discrete Location Theory. Wiley, New York, *1990*.

[16] C.S. Revelle, H.A. Eiselt, Location analysis: A synthesis and survey. *European Journal of Operational Research 165, 1–19, 2005*.

[17] C.S. Revelle, H.A. Eiselt, M.S. Daskin, A bibliography for some fundamental problem categories in discrete location science. *European Journal of Operational Research 184, 817–848, 2008*.

[18] B.C. Tansel, R.L. Francis, T.J. Lowe, Location on networks: a survey. Part I: the *p*-center and *p*-median problems. *Management Science 29, 482–497, 1983*.

[19] A. Chaudhury, P. Basuchowdhuri, S. Majumder, Spread of information in a social network using influential nodes. *Advances in Knowledge Discovery and Data Mining, LNCS 7302, 121–132, 2012*.

[20] J. Halpern, The location of a center-median convex combination on an undirected tree. *Journal of Regional Science 16, 237–245*, 1976.

[21] J. Halpern, Finding minimal center-median convex combination (cent-dian) of a graph. *Management Science 24, 535–544, 1978.*

[22] J.N. Hooker, R.S. Garfinkel, C.K. Chen, Finite dominating sets for network location problems. *Operation Research 39, 100–118, 1991*.

[23] A. Tamir, D. Perez-Brito, J.A. Moreno-Perez, A polynomial algorithm for the *p*-centdian problem on a tree. *Networks 32, 255–262, 1998*.

[24] Y. Bartal, Probabilistic approximation of metric spaces and its algorithmic applications. In Proceedings of *the 37th Annual IEEE Symposium on Foundations of Computer Science, 184–193, 1996*.

[25] Y. Bartal, On approximating arbitrary metrics by tree metrics. In Proceedings of *the 30th Annual ACM Symposium on Theory of Computing, 161–168 , 1998*.

[26] J. Kalcsics, S. Nickel, J. Puerto, Multifacility ordered median problems on networks: a further analysis. *Networks 41, 1-12, 2003.*

[27] D.P. Brito, J.A.M. Perez, The generalized p-centdian on network. *Top 8, 265-285, 2000*.

[28] G. Reinelt, TSPLIB—A traveling salesman problem library. *INFORMS Journal on Computing 3, 376–384, 1991*.

[29] T. H. Cormen, C.E. Leiserson. R.L. Rivest, C. Stein, Introduction to Algorithms. Third Edition. The MIT Press. Cambridge, Massachusetts, *2009*.

[30] N. Karmarkar. A new polynomial time algorithm for linear programming. *Combinatorica 4, 373–395, 1984*.

[31] J.H. Holland, Adaptation in Natural and Artificial Systems, Ann Arbor, The University of Michigan Press, 1975.

[32] Tsai Chueh Wang, Yen Hung Chen, Yu Xiang Zhu, Using linear programming relaxation to solve the p-centdian problem, University of Taipei, 2017.

A Study on Modeling and Classification of the Student Opinion Survey with Word2vec

Chih-Yang Huang
Department of Computer Science,
University of Taipei
user@hcy.idv.tw

Yen-Hung Chen
Department of Computer Science,
University of Taipei
yhchen@utaipei.edu.tw

Mei-Ching Ho
Department of English Instruction,
University of Taipei
mch0532@gmail.com

ABSTRACT

The Student Opinion Survey corpus is appealing to researchers because it represents a rich temporal record of impression of professors and courses. This paper attempts to conduct collection and analysis for the comments from the website Rate My Professor (http://www.ratemyprofessors.com) and builds the model to quantify the level of other comments, which means each comment is labeled scores from 1.0 to 5.0. We try to figure out the differences of positive comments and negative comments. For example, if the comments with some positive words are probable 5 scores, and on the other hand, the comments with negative words are about 1 score. Building the model to explore the terms used by the students is important to predict and distinguish the use of words.

CCS CONCEPTS

•The Establishment of the Corpus Database → Preprocessor → Modeling and Statistics → the Student Opinion Survey Analysis System

KEYWORDS

document classification, text-mining, word2vector, corpus modeling

1 INTRODUCTION

In a mere eighteen years, the university students in the USA try to share their comments and their idea about school through the Internet with the approach of rapid infrastructure of the communication. Base on the convenience of spreading information on the Internet, John Swapceinski founded the website called RateMyProfessors.com (RMP) to allow college and university students to assign ratings to professors and campuses of American, Canadian and United Kingdom institutions [1]. The website was originally launched as TeacherRatings.com and converted to RateMyProfessors in 2001. Nowadays, RateMyProfessors.com is the largest online destination for professor ratings. The site has 8,000+ schools, 1.7 million professors and over 19 million ratings. We collect the data from the website and insert the comments into database. In addition, we design the website and API for researchers to access this corpus database. Not only collecting data to preserve the corpus, we also extract some data to shape the dataset to a model, which means it can predict the rating of other incoming comments from users. In order to predict the most fit rating, we design a formula to calculate the similarity of the incoming comment between other comments, which enhances the precision of similarity of the comment. It would fit the dataset more accurately. After the modeling, we will design the student opinion survey analysis system for school to figure out the comments from students. With that system, it is easy for Academic Affairs Office to figure out the scenario of Teacher-student relationship and the quality of courses.

This paper is organized in four sections. The first section will introduce the collection of the corpus dataset. In the next two sections would introduce the corpus training model and the formula applied for achieving the accuracy of the predictions. Finally, the demonstration of the Student Opinion Survey Analysis System and the conclusion will be given.

2 EXPERIMENTAL AND COMPUTATIONAL DETAILS

2.1 The Establishment of the Corpus Database

We choose the website RateMyProfessors.com as our dataset. In order to collect the data in consistency, we use relational database to implement based on the relational model of data, as proposed by E. F. Codd in 1970 [2]. We first assume the website data is presented by html, thus collecting the data by Python and the famous package Beautiful Soup by installing it via pip. However, the website presentation is not as we think. We peep into the source code of that website. Finally, finding the website presentation of the comments data developed by Ajax is extremely important. We figure out another code designed by Node.js and simulate the user clicking the “load more” button. After clicking all buttons and conditions by the code of Node.js via headless explorer simulation, we catch the attributions as well as comments. Consequently, we output them as JSON （JavaScript Object Notation） files and insert them into MySQL database simultaneously. In prevention of data loss and instability, we choose Google Cloud Platform [3] as our first hosting server. Nevertheless, owing to the price of operation and maintenance, we change the host to Kinghood Technology CO. LTD [4].

2.2 The Corpus Training Model with Word2vec

2.2.1 Introduction of Word2vec. Language modeling is used in speech recognition, machine translation, part-of-speech tagging, parsing, handwriting recognition, information retrieval and other applications [5]. Word2vec is based on a statistical language model, which is a probability distribution over sequences of words. Given such a sequence, say of length m, it assigns a probability $P(w_1,w_{2,\ldots\ldots}w_m)$ to the whole sequence. It tries to maximize classification of a word based on another word in the same sentence. More precisely, it uses each current word as an input to a log-linear classifier with continuous projection layer, and predicts words within a certain range before and after the current word [7]. Having a way to estimate the relative likelihood of different phrases is useful in many natural language processing applications, especially ones that generate text as an output. All in all, we use that concept to estimate the words between words and the sentences between sentences. This paper is based on the complete theory and technology proposed by Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean on arXiv [6]. According to that paper, we use their implemented source code of continuous bag-of-words (CBOW) to apply on our dataset.

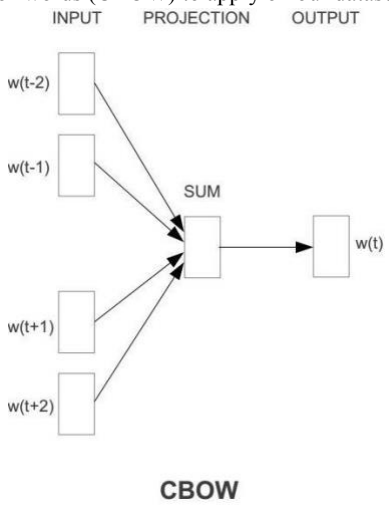


Figure 1: The concept of CBOW. It illustrates that the vector considers the relationship between words and words.

2.2.2 The correctness of using Word2vec. In the field of machine learning, the precision is the indicator that the most people are concerned about. In the case of Word2Vec, if the dataset is imbalanced, in other words, one kind of dataset occupies the most weight, it probably exists a bias to lean on the high weight of the dataset [7]. However, in our case, there are comments and labeled score bound. Thus, we just want to use the similarity analysis to distinguish the differences of each comment and find the labeled score. When getting the labeled score of the most similar comment or the rank 10 similar comments, we could deal with that labeled scores and point out the possible score of the inputted comment from the user. The formula of getting the possible score will be mentioned on the section 2.3.

2.2.3 Data choosing and modeling. For the purpose of training data in effectiveness, we choose the 20 universities from the top 200 universities at random.

Table 1: List of the chosen universities

Number	Name
1.	Johns Hopkins University
2.	Missouri University of Science and Technology
3.	San Francisco State University
4.	George Mason University
5.	Florida Agricultural and Mechanical University
6.	University of Southern California
7.	Regent University
8.	University of Washington
9.	Benedictine University
10.	Virginia Tech
11.	Georgia Institute of Technology
12.	Nova Southeastern University
13.	Biola University
14.	University of Alabama Huntsville
15.	California State University Fresno
16.	Pepperdine University
17.	Temple University
18.	University of Wisconsin - Milwaukee
19.	Boise State University
20.	University of Illinois Urbana Champaign Law School

We choose the comments from the above universities, setting the time from 2014/01/01 to 2017/12/31. On the ground of separating positive and negative comments as well as preparing for the adjustment of accuracy, we divide the comments of score from 1.0 to 2.0 and from 4.0 to 5.0 into two groups. So, there are three groups to train three models. One is the group of scores from 1.0 to 2.0 in the 20 universities. Another is the group of scores from 4.0 to 5.0. And the other is the mix of the two groups. We use three datasets to train three models by ignoring 5 noise words, the maximum 3 distances between the current and predicted word and neglecting all words with total frequency lower than 1 word for each sentence. The next section will explain the use of three models and adjustment of accuracy.

2.3 The Formula of Adjusting Accuracy

In order to get the scores of the comment more precisely, we design a formula for counting the probable scores. If a user input a comment, we will use the model of all comments to get the top 10 similar comments, acquiring each score of that 10 comments. Calculating the average of the scores from top 10, it is the main score for that inputted comment. Suppose that the user inputs an unknown comment. After inputting that into the mix model, we will get the top n scores $S_0, S_1, S_2, \ldots, S_n$ bounded with the most n similar comments. Assume the main score for the unknown comment is N. Thus, the first step for the predicted main scores is

$$\mathrm{N} = \frac{\sum_k^n S_k}{n} \quad (1)$$

After gaining the main scores, we need to adjust it based on itself. We put N into the lower scores (1.0 to 2.0 scores) model, getting the absolute cosine value of the first similarity called *L*. Also, we put N into the higher scores (4.0 to 5.0 scores) model, getting the absolute cosine value of the first similarity called *H*. We design the second step for the adjustment. We classify N into three conditions, one is N <= 2.0 called X_0, another is 2.0 < N <= 3.9 called X_1, the other is N >= 4.0 called X_2. If the N in the range of X_1, we express the adjustment formula as below.

$$\mathrm{N} = \alpha\mathrm{N} + (1-\alpha)(\mathrm{L}-\mathrm{H}) \quad (2)$$

α is the coefficient to control the weight between the unadjusted main score and the adjustment value. In this case (L-H) means the model of lower scores is more important that the model of higher scores.

On the other hand, there is also the second step for N in the range of X_3.

$$\mathrm{N} = \alpha\mathrm{N} + (1-\alpha)(\mathrm{H}-\mathrm{L}) \quad (3)$$

The meaning of α, H, L, N is the same as above. In this case (H-L) means the model of higher scores is more important that the model of lower scores.

In addition, we need to take N in the range of X_2 into consideration.

$$\mathrm{N} = \alpha\mathrm{N} + (1-\alpha)\frac{(\mathrm{H}+\mathrm{L})}{2} \quad (4)$$

In the case X_2, it means N is in the middle, which implies the comment is neutral. So, the adjustment of the main scores is required to modify in the smooth way. In brief, we separate N into 3 cases, and each of them is manipulated by (2) (3) (4). In addition, we will take some examples in the next section to illustrate our prediction models and adjustment to be sensible.

3 RESULTS AND DISCUSSION

3.1 Standard Positive Comments

We consider the unknown comment "I used to love math and I was straight A student before her class, She is good but she will take away all your confidence and she wont take it easy on students. I love challenge but she is more than that. You are going to have Quiz every day and with only one misstake she will take away many points, how ever she give many extra credit.No Eassy A", labeled as 4 scores, putting it into three models. After calculation, we get the data in the below table.

Table 2: The Data of the positive comment

The Content:	I used to love math and I was straight A student before her class, She is good but she will take away all your confidence and she wont take it easy on students. I love challenge but she is more than that. You are going to have Quiz every day and with only one misstake she will take away many points, how ever she give many extra credit.No Eassy A
The Labeled Scores:	4.0
The Main Scores:	4.5
α:	0.9
The Lower Scores Model:	0.43594351410865784
The Higher Scores Model:	0.45307379961013794
After Adjusting Scores:	4.05171

According the table 2, the result of the adjusted scores is almost close to the labeled scores, which means the models and adjustments are authentic.

3.2 Standard Negative Comments

We take the unknown comment "Worst class I have ever taken and biggest waste of money. Completely not helpful at all. Even after going to him several times with questions, he either never responded or it was months later!", labeled as 4 scores, for example. We put it into three models. After calculation, we get the data in the below table.

Table 3: The Data of the negative comment

The Content:	Worst class I have ever taken and biggest waste of money. Completely not helpful at all. Even after going to him several times with questions, he either never responded or it was months later!
The Labeled Scores:	1.0
The Main Scores:	4.2
α:	0.3
The Lower Scores Model:	0.636993765830993
The Higher Scores Model:	0.5076644420623779
After Adjusting Scores:	1.350529992

According the table 3, the result of the adjusted scores is not as close as to the labeled scores. In this case, $\boldsymbol{\alpha}$ is vital for the adjustment.

4 CONCLUSIONS

We have described how the student opinion survey corpus database was enhanced and refined for this study, and how to extract data from the website Rate My Professor, leading to model and classify for that corpus. In summary, we have performed both an experimental and theoretical study of the algorithm Word2Vec. In addition, we also design a formula to decrease the divergence between the predicted scores and actual scores. We attempt to minimize the noise originating from the natural language. Based on this corpus and concept, we would implement a system for academic institutions to predict and evaluate the teaching of one professor approximately. It would provide a roughly basic appearance of a professor to urge the positive relationships between teaching and learning. We believe that our research will help the other researchers to recognize and know the thinking from students. We also expect that this paper will bring other corpus analysis based on NLP applications.

ACKNOWLEDGMENTS

Corresponding author: Chih-Yang Huang.

REFERENCES

[1] https://en.wikipedia.org/wiki/RateMyProfessors.com
[2] Codd, E. F. (1970). A relational model of data for large shared data banks. Communications of the ACM, 13(6), 377-387.
[3] https://cloud.google.com
[4] http://www.kinghood.com
[5] https://en.wikipedia.org/wiki/Language_model
[6] Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.
[7] 蕭文峰, & 楊宗恩. (2016). 以 Word2vec 深度神經網路語言建構英文課程之摘要主題模型. TANET2016 臺灣網際網路研討會, 634-639.

A One-to-Many Parallel Routing Algorithm on Generalized Honeycomb Tori *

Shyue–Ming Tang[1] and Jou–Ming Chang[2]
[1]Department of Psychology and Social Work, National Defense University, Taipei, Taiwan.
[2]Institute of Information and Decision Sciences, National Taipei University of Business, Taipei, Taiwan.
tang1119@gmail.com, spade@ntub.edu.tw

Abstract

A one-to-many routing algorithm has applications in fault-tolerant broadcasting and secure message distribution in networks. Two spanning trees of a network are independent *if they are rooted at the same node r, and for every other node $v \neq r$, the two paths from r to v, one path in each tree, are internally disjoint. It is obvious that constructing multiple independent spanning trees (ISTs for short) rooted at one node can guarantee a one-to-many routing from the node.*

A generalized honeycomb torus (GHT for short) is constructed by adding wraparound edges on a honeycomb mesh. A GHT is 3-regular and node-transitive. Without loss of generality, the algorithm can choose any node as the root of the ISTs. In this paper, we proposed an algorithm to construct three ISTs based on the decision of individual node in a given GHT. As a result, the algorithm can parallelize the construction of the ISTs efficiently.

Keyword: interconnection networks, generalized honeycomb torus, fault-tolerant broadcasting, independent spanning trees, one-to-many parallel routing, internally disjoint paths.

1 Introduction

For $a \leqslant b$, let $[a, b] = \{a, a+1, \ldots, b\}$ be the set of consecutive integers from a to b. Based on the definition in [3], a generalized honeycomb torus (GHT for short), denoted by H(m,n,d), consists of $N(= m \times n)$ nodes and $3N/2$ edges, where $m \geqslant 2$, $n \geqslant 4$ is even, and $d \in [1, n-1]$ is odd if m is odd and $d \in [0, n-2]$ is even otherwise. For $i \in [0, m-1]$ and $j \in [0, n-1]$, a node $x = (i, j)$ of a GHT has three neighbors, denoted by three related skips from x: $\langle U \rangle$, $\langle D \rangle$ and $\langle R/L \rangle$ (the latter depends on $i+j$ being odd/even), where $\langle U \rangle$ and $\langle D \rangle$ stand for neighbors $(i, j+1)$ and $(i, j-1)$, respectively, $\langle R \rangle$ stands for either neighbor $(i+1, j)$ if $i < m-1$ or $(0, j-d)$ if $i = m-1$, and $\langle L \rangle$ stands for either neighbor $(i-1, j)$ if $i > 0$ or $(m-1, j+d)$ if $i = 0$. Notice that both $j \pm 1$ and $j \pm d$ take modulo n. For example, H(5,8,3) and H(6,8,4) are shown in Fig. 1(a) and 1(b), respectively.

In 1997, Stojmenović [16] proposed several variations of honeycomb tori. Cho and Hsu [3] then proved that all honeycomb tori can be characterized in a unified way. Many research results on GHT which is recognized as an alternative architecture of two-dimensional torus are proposed in the passed two decades [2, 6, 7, 8, 12, 13, 15, 20, 21]. In particular, some researchers contributed their efforts to the topic of ring embedding in a faulty GHT [2, 6, 8, 13].

Two different paths connecting two nodes in a network is said to be *internally disjoint* if they have no common node except two end nodes. The *one-to-many parallel routing* of a network is to construct internally disjoint paths from one given node to other nodes. According to the routing, one node can send copies of a message along different internally disjoint paths to achieve fault-tolerant broadcasting. Besides, a message can be separated into k parts and send them to other nodes through k internally disjoint paths to ensure secure message distribution.

Two spanning trees of a graph are *independent* if they are rooted at the same node r, and for every other node $v(\neq r)$, two different paths from v to r, one path in each tree, are internally disjoint. A set of spanning trees of a graph is said to be independent if they are pairwise independent. In 1989, Zehavi and Itai [25] conjecture that, for any node r in a k-connected graph G, there exist k independent spanning trees (ISTs for short) of

*This research is supported by the Ministry of Science and Techonology of Taiwan under the Grant MOST104–2221–E–141–002-MY3.

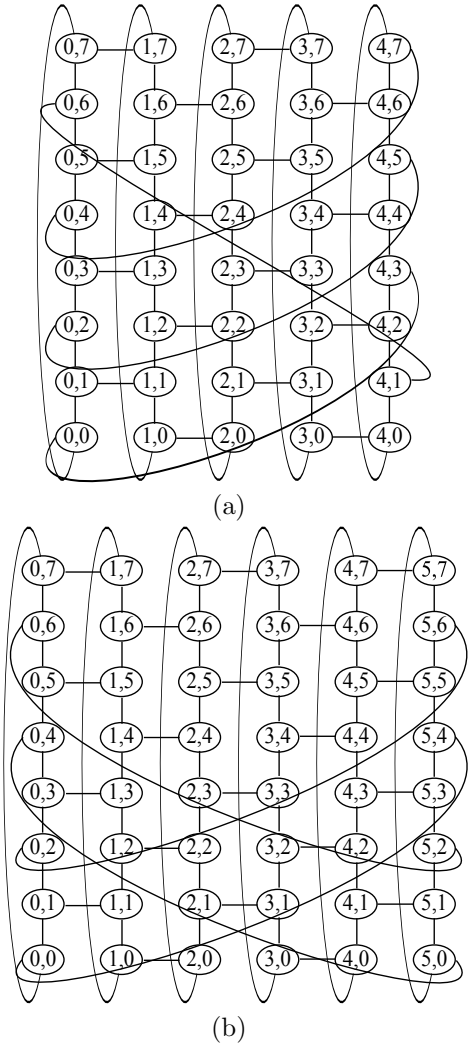


Figure 1: Two GHT examples, (a) H(5,8,3) and (b) H(6,8,4).

G rooted at r. Although the conjecture has been proved to be affirmative for k-connected graphs with $k \leqslant 4$ (see [9] for $k = 2$, [4, 25] for $k = 3$, and [5] for $k = 4$), it is still open for $k > 4$. Since the construction of ISTs guarantees the one-to-many routing in a network, lots of research results are presented for solving the IST problem in special graph classes, especially in interconnection networks [10, 11, 14, 17, 18, 19, 23, 22, 24].

In this paper, we propose an algorithm for constructing three ISTs rooted at an arbitrary node of a GHT. Particularly, the proposed algorithm is designed for every individual node, based only on the label of a node, and thus make the construction parallelized. Although the IST problem of general 3-connected graphs was solved in linear time by Cheriyan and Maheshwari [4], the proposed algorithm still has its contribution on parallelized implementation.

The remaining part of this paper is organized as follows. Sect. 2 gives essential notations of the algorithm. Sect. 3 presents the algorithm. Sect. 4 proves the correctness of the algorithm. The last section contains our concluding remarks.

2 Notations

To explicitly represent the adjacency of nodes in a GHT H(m,n,d), we say that node (i, j) takes a skip to reach one of three neighbors. If $i + j$ is odd, the three skips are $\langle U\rangle$ (up), $\langle D\rangle$ (down) and $\langle R\rangle$ (right), while if $i + j$ is even, the three skips are $\langle U\rangle$ (up), $\langle D\rangle$ (down) and $\langle L\rangle$ (left). The skip representation is helpful to express our construction algorithm.

Since a GHT is node-transitive [1], without loss of generality, we only need to consider all ISTs rooted at node (0,0). Because of the requirement of internally disjoint paths in ISTs, it is obvious that the root has only one child in each of the trees. If Λ is the skip taken by the only child for reaching the root, the tree is denoted by T_Λ. Hence, for the ISTs of a GHT, the root-reaching skips of every tree must be distinct. In Fig. 2(a), 2(b) and 2(c), for example, each IST of H(5,8,3) is named after the unique skip taken by the child to reach the root.

Furthermore, for every non-root node, the parent-reaching skips are also distinct in three ISTs. Accordingly, our construction algorithm is simply to determine the skips taken in different trees for every non-root node in parallel. We adopt the notation $x \xrightarrow{\Lambda} y$ to mean that x takes the skip Λ to reach its parent y in a tree. Also, for a node $x = (i, j)$ in a tree T_Λ, the unique path from x to the root $(0, 0)$ is denoted by $P_\Lambda[i, j]$.

For example, in Fig. 2, we consider node $x = (4, 7)$, and we have the following three paths, one path in each tree, from x to the root (0,0):
$P_{\langle U\rangle}[4,7] : (4,7) \xrightarrow{\langle D\rangle} (4,6) \xrightarrow{\langle L\rangle} (3,6) \xrightarrow{\langle U\rangle} (3,7) \xrightarrow{\langle L\rangle} (2,7) \xrightarrow{\langle D\rangle} (2,6) \xrightarrow{\langle L\rangle} (1,6) \xrightarrow{\langle U\rangle} (1,7) \xrightarrow{\langle L\rangle} (0,7) \xrightarrow{\langle U\rangle} (0,0)$;
$P_{\langle D\rangle}[4,7] : (4,7) \xrightarrow{\langle R\rangle} (0,4) \xrightarrow{\langle D\rangle} (0,3) \xrightarrow{\langle D\rangle} (0,2) \xrightarrow{\langle D\rangle} (0,1) \xrightarrow{\langle D\rangle} (0,0)$;
$P_{\langle R\rangle}[4,7] : (4,7) \xrightarrow{\langle U\rangle} (4,0) \xrightarrow{\langle U\rangle} (4,1) \xrightarrow{\langle U\rangle} (4,2) \xrightarrow{\langle U\rangle} (4,3) \xrightarrow{\langle R\rangle} (0,0)$.

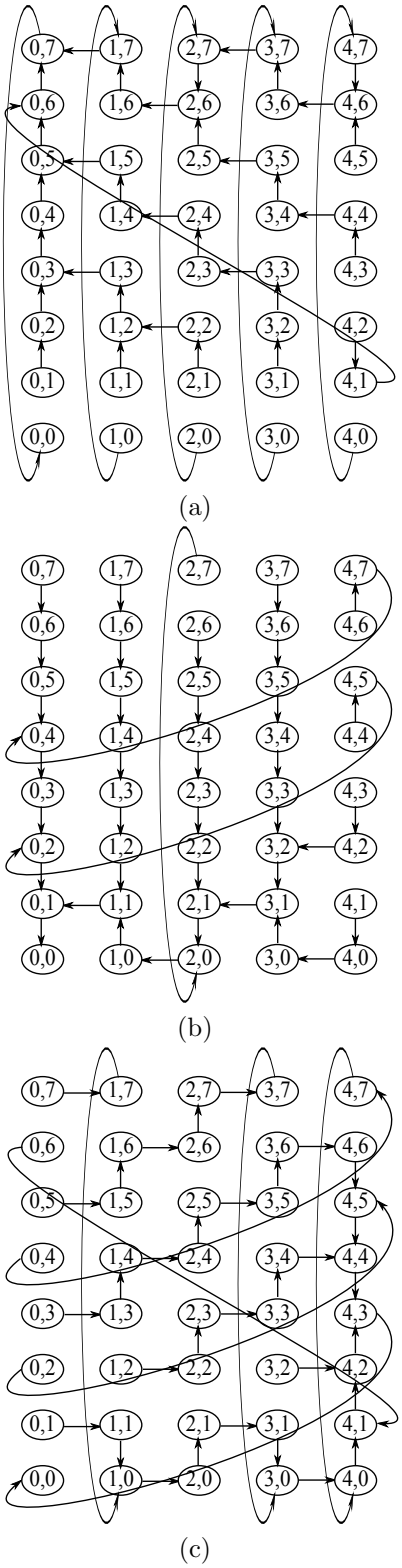


Figure 2: Three ISTs rooted at node (0,0) on H(5,8,3), (a) $T_{\langle U\rangle}$, (b) $T_{\langle D\rangle}$ and (c) $T_{\langle R\rangle}$.

It is easy to check that these paths are internally disjoint. For notational convenience, sometimes we write $P_{\langle U\rangle}[4,7] = (\langle D\rangle\langle L\rangle\langle U\rangle\langle L\rangle)^2\langle U\rangle$, $P_{\langle D\rangle}[4,7] = \langle R\rangle\langle D\rangle^4$ and $P_{\langle R\rangle}[4,7] = \langle U\rangle^4\langle R\rangle$ instead, where Λ^t denotes that a skip (or a skip sequence) Λ consecutively occurs t times for $t > 1$.

3 Parallel construction of independent spanning trees

The following algorithm constructs three ISTs rooted at node (0,0) in GHT(m,n,d), where $m \geqslant 2$ and $n \geqslant 4$ (n is even). In this algorithm, called Algorithm Parent-Determining, for every non-root node (i,j) the skip reaching its parent in every tree is determined only according to the node label i and j. For the sake of brevity, two procedures Parent-R and Parent-L are used to assign the parent-reaching skips of the node (i,j) for $i+j$ being odd and even, respectively. That is, the three parameters of Parent-R (resp. Parent-L) assign sequentially the skips for a node to get its parents in $T_{\langle U\rangle}$, $T_{\langle D\rangle}$ and $T_{\langle R\rangle}$ (resp. $T_{\langle U\rangle}$, $T_{\langle D\rangle}$ and $T_{\langle L\rangle}$).

The proposed algorithm looks very complicated due to the processing of some special cases. In case of $m = 2$ and $d = 0$, the parent-reaching skips of nodes $(m-1,j)$ $(0 \leqslant j \leqslant n-1)$ are different from other cases. Further, in all other cases, the parent-reaching skips of nodes $(m-1,0)$ (at bottom right corner) and $(m-1,n-1)$ (at top right corner) also varies according to different m and d. We summarized the GHTs to six types which are listed in Table 1. Fortunately, Algorithm Parent-Determining can solve the IST problem of GHTs for all of the six types.

Let us explain with an example. Table 2 shows the parent-reaching skips of all nodes in H(5,8,3) by using Algorithm Parent-Determining. In Table 2, all the nodes with odd $i+j$ which apply procedure Parent-R are put on the left side; while nodes with even $i+j$ which apply procedure Parent-L are put on the right side.

4 Correctness Proof

To show the correctness of Algorithm Parent-Determining, we have to prove firstly that the output of the algorithm are spanning trees of the input GHT network, or equivalently, every node has a unique path to connect the root. Secondly, we should prove that for every non-root node (i,j),

Algorithm 1: PARENT-DETERMINING

```
case i = 0 : do        // the left-most column
    if j is odd then PARENT-R(⟨U⟩,⟨D⟩,⟨R⟩) ;
    else PARENT-L(⟨U⟩,⟨D⟩,⟨L⟩) ;
case i ∈ [1, m − 2] and m ⩾ 3 : do        // the middle columns
    case j = 0 : do
        if i + j is odd then PARENT-R(⟨D⟩,⟨U⟩,⟨R⟩) ;
        else PARENT-L(⟨D⟩,⟨L⟩,⟨U⟩) ;
    case j = 1 : do
        if i + j is odd then PARENT-R(⟨U⟩,⟨D⟩,⟨R⟩) ;
        else PARENT-L(⟨U⟩,⟨L⟩,⟨D⟩) ;
    case j ∈ [2, n − 2] : do
        if i + j is odd then PARENT-R(⟨U⟩,⟨D⟩,⟨R⟩) ;
        else PARENT-L(⟨L⟩,⟨D⟩,⟨U⟩) ;
    case j = n − 1 : do
        if i is odd then PARENT-R(⟨D⟩,⟨U⟩,⟨R⟩) ;
        else PARENT-L(⟨L⟩,⟨D⟩,⟨U⟩) ;
case i = m − 1 : do        // the right-most column
    case j = 0 : do
        case d = 0 : do PARENT-R(⟨D⟩,⟨U⟩,⟨R⟩) ;
        case d = n − 1 : do PARENT-L(⟨U⟩,⟨L⟩,⟨D⟩) ;
        case d = n − 2 : do PARENT-R(⟨R⟩,⟨U⟩,⟨D⟩) ;
        case d ∉ {0, n − 1, n − 2} : do
            if i+j is odd then PARENT-L(⟨D⟩,⟨L⟩,⟨U⟩) ;
            else PARENT-R(⟨R⟩,⟨D⟩,⟨U⟩) ;
    case j ∈ [1, n − 2] : do
        case m = 2 and d = 0 : do PARENT-R(⟨U⟩,⟨R⟩,⟨D⟩) ;
        case m ≠ 2 or d ≠ 0 : do
            if i + j is odd then
                case j > d : do PARENT-R(⟨U⟩,⟨R⟩,⟨D⟩) ;
                case j = d : do PARENT-R(⟨U⟩,⟨D⟩,⟨R⟩) ;
                case j < d : do PARENT-R(⟨R⟩,⟨D⟩,⟨U⟩) ;
            else
                if j > d then PARENT-L(⟨L⟩,⟨U⟩,⟨D⟩) ;
                else PARENT-L(⟨D⟩,⟨L⟩,⟨U⟩) ;
    case j = n − 1 : do
        case d = 0 : do PARENT-L(⟨L⟩,⟨D⟩,⟨U⟩) ;
        case d = n − 1 : do PARENT-R(⟨D⟩,⟨U⟩,⟨R⟩) ;
        case d = n − 2 : do PARENT-L(⟨L⟩,⟨U⟩,⟨D⟩) ;
        case d ∉ {0, n − 1, n − 2} : do
            if i+j is odd then PARENT-R(⟨D⟩,⟨R⟩,⟨U⟩) ;
            else PARENT-L(⟨L⟩,⟨D⟩,⟨U⟩) ;
```

Table 1: Summary of six different types of GHTs

type	m	d	example
1	$m \geqslant 3$, odd	$1 \leqslant d \leqslant n-3$, odd	GHT(5,8,3)
2	$m \geqslant 3$, odd	$d = n-1$, odd	GHT(5,8,7)
3	$m \geqslant 2$, even	$2 \leqslant d \leqslant n-4$, even	GHT(6,8,4)
4	$m \geqslant 2$, even	$d = n-2$, even	GHT(6,8,6)
5	$m \geqslant 4$, even	$d = 0$	GHT(6,8,0)
6	$m = 2$	$d = 0$	GHT(2,8,0)

Table 2: The parent skips of node (i, j) in three ISTs on H(5,8,3)

node	$T_{[U]}$	$T_{[D]}$	$T_{[R]}$
(0,1)	$\langle U\rangle$	$\langle D\rangle$	$\langle R\rangle$
(0,3)	$\langle U\rangle$	$\langle D\rangle$	$\langle R\rangle$
(0,5)	$\langle U\rangle$	$\langle D\rangle$	$\langle R\rangle$
(0,7)	$\langle U\rangle$	$\langle D\rangle$	$\langle R\rangle$
(1,0)	$\langle D\rangle$	$\langle U\rangle$	$\langle R\rangle$
(1,2)	$\langle U\rangle$	$\langle D\rangle$	$\langle R\rangle$
(1,4)	$\langle U\rangle$	$\langle D\rangle$	$\langle R\rangle$
(1,6)	$\langle U\rangle$	$\langle D\rangle$	$\langle R\rangle$
(2,1)	$\langle U\rangle$	$\langle D\rangle$	$\langle R\rangle$
(2,3)	$\langle U\rangle$	$\langle D\rangle$	$\langle R\rangle$
(2,5)	$\langle U\rangle$	$\langle D\rangle$	$\langle R\rangle$
(2,7)	$\langle D\rangle$	$\langle U\rangle$	$\langle R\rangle$
(3,0)	$\langle D\rangle$	$\langle U\rangle$	$\langle R\rangle$
(3,2)	$\langle U\rangle$	$\langle D\rangle$	$\langle R\rangle$
(3,4)	$\langle U\rangle$	$\langle D\rangle$	$\langle R\rangle$
(3,6)	$\langle U\rangle$	$\langle D\rangle$	$\langle R\rangle$
(4,1)	$\langle R\rangle$	$\langle D\rangle$	$\langle U\rangle$
(4,3)	$\langle U\rangle$	$\langle D\rangle$	$\langle R\rangle$
(4,5)	$\langle U\rangle$	$\langle R\rangle$	$\langle D\rangle$
(4,7)	$\langle D\rangle$	$\langle R\rangle$	$\langle U\rangle$

node	$T_{[U]}$	$T_{[D]}$	$T_{[R]}$
(0,2)	$\langle U\rangle$	$\langle D\rangle$	$\langle L\rangle$
(0,4)	$\langle U\rangle$	$\langle D\rangle$	$\langle L\rangle$
(0,6)	$\langle U\rangle$	$\langle D\rangle$	$\langle L\rangle$
(1,1)	$\langle U\rangle$	$\langle L\rangle$	$\langle D\rangle$
(1,3)	$\langle L\rangle$	$\langle D\rangle$	$\langle U\rangle$
(1,5)	$\langle L\rangle$	$\langle D\rangle$	$\langle U\rangle$
(1,7)	$\langle L\rangle$	$\langle D\rangle$	$\langle U\rangle$
(2,0)	$\langle D\rangle$	$\langle L\rangle$	$\langle U\rangle$
(2,2)	$\langle L\rangle$	$\langle D\rangle$	$\langle U\rangle$
(2,4)	$\langle L\rangle$	$\langle D\rangle$	$\langle U\rangle$
(2,6)	$\langle L\rangle$	$\langle D\rangle$	$\langle U\rangle$
(3,1)	$\langle U\rangle$	$\langle L\rangle$	$\langle D\rangle$
(3,3)	$\langle L\rangle$	$\langle D\rangle$	$\langle U\rangle$
(3,5)	$\langle L\rangle$	$\langle D\rangle$	$\langle U\rangle$
(3,7)	$\langle L\rangle$	$\langle D\rangle$	$\langle U\rangle$
(4,0)	$\langle D\rangle$	$\langle L\rangle$	$\langle U\rangle$
(4,2)	$\langle D\rangle$	$\langle L\rangle$	$\langle U\rangle$
(4,4)	$\langle L\rangle$	$\langle U\rangle$	$\langle D\rangle$
(4,6)	$\langle L\rangle$	$\langle U\rangle$	$\langle D\rangle$

three paths from (i, j) to the root (0,0) in different spanning trees must be internally disjoint. In the following proof, we will be focusing on the type 1 GHT networks for concise sake.

Lemma 1. *Algorithm* PARENT-DETERMINING *can generate three spanning trees rooted at node (0,0) in H(m,n,d).*

Proof : In H(m,n,d), node (i, j) can take one path to reach the root. We consider $P_{\langle U\rangle}[i, j]$, $P_{\langle D\rangle}[i, j]$ and $P_{\langle R\rangle}[i, j]$ separately, which are shown in Table 3, 4 and 5, respectively.

In any case, a skip sequence from (i, j) to (0,0) in the output spanning subgraph forms a unique path. □

The following lemma shows the independency of the output spanning trees.

Table 3: Analysis of the path $P_{\langle U\rangle}[i,j]$

case	skip sequence	length
$i=0$	$\langle U\rangle^{n-j}$	$n-j$
$i=m-1$, odd j		
(1) $1\le j\le d-2$	$\langle R\rangle\langle U\rangle^{d-j}$	$d-j+1$
(2) $d\le j\le n-3$	$\langle U\rangle(\langle L\rangle\langle U\rangle)^{n-j-2}(\langle L\rangle\langle D\rangle\langle L\rangle\langle U\rangle)^{(m-n+j)/2}\langle L\rangle\langle U\rangle$	$2m-1$
(3) $j=n-1$	$\langle D\rangle\langle L\rangle(\langle U\rangle\langle L\rangle\langle D\rangle\langle L\rangle)^{(m-3)/2}\langle U\rangle\langle L\rangle\langle U\rangle$	$2m-1$
$i=m-1$, even j		
(1) $j=0$	$\langle D\rangle^2\langle L\rangle(\langle U\rangle\langle L\rangle\langle D\rangle\langle L\rangle)^{(m-3)/2}\langle U\rangle\langle L\rangle\langle U\rangle$	$2m$
(2) $2\le j\le d-1$	$\langle D\rangle\langle R\rangle\langle U\rangle^{d-j+1}$	$d-j+3$
(3) $d+1\le j\le n-2$	$(\langle L\rangle\langle U\rangle)^{n-j-1}(\langle L\rangle\langle D\rangle\langle L\rangle\langle U\rangle)^{(m-n+j-1)/2}\langle L\rangle\langle U\rangle$	$2m-2$
$1\le i\le m-2$, odd i		
(1) $j=0$	$\langle D\rangle(\langle L\rangle\langle D\rangle\langle L\rangle\langle U\rangle)^{(i-1)/2}\langle L\rangle\langle U\rangle$	$2i+1$
(2) $j=1$	$\langle U\rangle(\langle U\rangle\langle L\rangle)^i\langle U\rangle^{n-i-j-1}$	$i-j+n$
(3) $2\le j\le n-2$, even j	$(\langle U\rangle\langle L\rangle)^i\langle U\rangle^{n-i-j}$	$i-j+n$
(4) $3\le j\le n-1$, odd j	$(\langle L\rangle\langle U\rangle)^i\langle U\rangle^{n-i-j}$	$i-j+n$
$1\le i\le m-2$, even i		
(1) $j=0$	$\langle D\rangle(\langle D\rangle\langle L\rangle\langle U\rangle\langle L\rangle)^{i/2}\langle U\rangle$	$2i+1$
(2) $1\le j\le n-1$, odd j	$(\langle U\rangle\langle L\rangle)^i\langle U\rangle^{n-i-j}$	$i-j+n$
(3) $2\le j\le n-2$, even j	$(\langle L\rangle\langle U\rangle)^i\langle U\rangle^{n-i-j}$	$i-j+n$

Table 4: Analysis of the path $P_{\langle D\rangle}[i,j]$

case	skip sequence	length
$i=0$	$\langle D\rangle^j$	j
$i=m-1$, odd j		
(1) $j=1$	$\langle D\rangle(\langle L\rangle\langle U\rangle\langle L\rangle\langle D\rangle)^{(m-1)/2}$	$2m-1$
(2) $3\le j\le d$	$\langle D\rangle\langle L\rangle\langle D\rangle^{j-2}\langle L\rangle\langle D\rangle(\langle L\rangle\langle U\rangle\langle L\rangle\langle D\rangle)^{(m-3)/2}$	$j+2m-4$
(3) $d+2\le j\le n-1$	$\langle R\rangle\langle D\rangle^{j-d}$	$j-d+1$
$i=m-1$, even j		
(1) $j=0$	$(\langle L\rangle\langle U\rangle\langle L\rangle\langle D\rangle)^{(m-1)/2}$	$2m-2$
(2) $2\le j\le d-1$	$\langle L\rangle\langle D\rangle^{j-1}\langle L\rangle\langle D\rangle(\langle L\rangle\langle U\rangle\langle L\rangle\langle D\rangle)^{(m-3)/2}$	$j+2m-4$
(3) $d+1\le j\le n-2$	$\langle U\rangle\langle R\rangle\langle D\rangle^{j-d+1}$	$j-d+3$
$1\le i\le m-2$, odd i		
(1) $j=0$	$(\langle U\rangle\langle L\rangle\langle D\rangle\langle L\rangle)^{(i-1)/2}\langle U\rangle\langle L\rangle\langle D\rangle$	$2i+1$
(2) $j=1$	$(\langle L\rangle\langle D\rangle\langle L\rangle\langle U\rangle)^{(i-1)/2}\langle L\rangle\langle D\rangle$	$2i$
(3) $2\le j\le n-2$	$\langle D\rangle^{j-1}(\langle L\rangle\langle D\rangle\langle L\rangle\langle U\rangle)^{(i-1)/2}\langle L\rangle\langle D\rangle$	$j+2i-1$
(4) $j=n-1$	$\langle D\rangle^{j-1}(\langle L\rangle\langle D\rangle\langle L\rangle\langle U\rangle)^{(i-1)/2}\langle L\rangle\langle D\rangle$	$2i+j-1$
$1\le i\le m-2$, even i		
(1) $j=0$	$(\langle L\rangle\langle U\rangle\langle L\rangle\langle D\rangle)^{(i-1)/2}$	$2i$
(2) $j=1$	$(\langle D\rangle\langle L\rangle\langle U\rangle\langle L\rangle)^{i/2}\langle D\rangle$	$2i+1$
(3) $2\le j\le n-2$	$\langle D\rangle^j(\langle L\rangle\langle U\rangle\langle L\rangle\langle D\rangle)^{i/2}$	$j+2i$
(4) $j=n-1$	$\langle U\rangle(\langle L\rangle\langle U\rangle\langle L\rangle\langle D\rangle)^{i/2}$	$2i+1$

Table 5: Analysis of the path $P_{\langle R\rangle}[i,j]$

case	skip sequence	length
$i=0$, odd j		
(1) $j=1$	$\langle R\rangle\langle D\rangle(\langle R\rangle\langle U\rangle\langle R\rangle\langle D\rangle)^{(m-3)/2}\langle R\rangle\langle U\rangle^d\langle R\rangle$	$2m+d-2$
(2) $3\le j\le n-1$		
(2a) $j+m-2>d$	$(\langle R\rangle\langle U\rangle)^{m-2}\langle R\rangle\langle D\rangle^{j+m-2-d}\langle R\rangle$	$j+3m-d-4$
(2b) $j+m-2<d$	$(\langle R\rangle\langle U\rangle)^{m-2}\langle R\rangle\langle D\rangle^{d-j-m+2}\langle R\rangle$	$m+d-j$
$i=0$, even j		
(1) $2\le j\le n-d-3$	$\langle L\rangle\langle D\rangle^j\langle R\rangle$	$j+2$
(2) $n-d-1\le j\le n-2$	$\langle L\rangle\langle U\rangle^{n-j}\langle R\rangle$	$n-j+2$
$i=m-1$		
(1) $j<d$	$\langle U\rangle^{d-j}\langle R\rangle$	$d-j+1$
(2) $j=d$	$\langle R\rangle$	1
(3) $j>d$	$\langle D\rangle^{j-d}\langle R\rangle$	$j-d+1$
$1\le i\le m-2$, odd i		
(1) $j=0$	$(\langle R\rangle\langle U\rangle\langle R\rangle\langle D\rangle)^{(m-i-2)/2}\langle R\rangle\langle U\rangle^d$	$2m-2i-3+d$
(2) $j=1$	$(\langle D\rangle\langle R\rangle\langle U\rangle\langle R\rangle)^{(m-i-2)/2}\langle D\rangle\langle R\rangle\langle U\rangle^d$	$2m-2i-2+d$
(3) $j=n-1$	$\langle U\rangle(\langle R\rangle\langle L\rangle\langle R\rangle\langle D\rangle)^{(m-i-2)/2}\langle R\rangle\langle U\rangle^d$	$2m-2i+d-2$
$1\le i\le m-2$, even i		
(1) $j=0$	$(\langle U\rangle\langle R\rangle\langle D\rangle\langle R\rangle)^{(m-i-1)/2}\langle U\rangle^d$	$2m-2i-2+d$
(2) $j=1$	$(\langle R\rangle\langle D\rangle\langle R\rangle\langle U\rangle)^{(m-i-1)/2}\langle U\rangle^{d-1}$	$2m-2i-3+d$
(3) $j=n-1$	$\langle R\rangle\langle U\rangle(\langle R\rangle\langle L\rangle\langle R\rangle\langle D\rangle)^{(m-i-3)/2}\langle R\rangle\langle U\rangle^d$	$2m-2i+d-3$
$2\le i\le m-2$, odd $i+j$ $(j\ne 0,1,n-1)$		
(1) $2\le m-i-2+j<d$	$(\langle R\rangle\langle U\rangle)^{m-i-2}\langle R\rangle\langle U\rangle^{d-m+i}\langle R\rangle$	$m-i+d$
(2) $d<m-i-2+j\le n-2$	$(\langle R\rangle\langle U\rangle)^{m-i-2}\langle R\rangle\langle D\rangle^{m-i-d}\langle R\rangle$	$3m-3i-d-2$
(3) $n-2<m-i-2+j$	$(\langle R\rangle\langle U\rangle)^{m-n-i+j}\langle U\rangle^d$	$2m-2n-2i$ $+2j+d$
$2\le i\le m-2$, even $i+j$ $(j\ne 0,1,n-1)$		
(1) $2\le m-i-2+j<d$	$(\langle U\rangle\langle R\rangle)^{m-i-1}\langle U\rangle^{d-m+i+2-j}\langle R\rangle$	$m-i+d$
(2) $d<m-i-2+j\le n-2$	$(\langle U\rangle\langle R\rangle)^{m-i-1}\langle D\rangle^{m-i-2+j-d}\langle R\rangle$	$3m-3i-d-2$
(3) $n-2<m-i-2+j$	$(\langle U\rangle\langle R\rangle)^{m-n-i+j}\langle R\rangle\langle U\rangle^d$	$2m-2n-2i$ $+2j+d+1$

Lemma 2. *Three paths from a non-root node (i,j) to the root in the output spanning trees on H(m,n,d) are internally disjoint.*

Proof : Let (u,v) be a non-root ancestor of node (i,j) in one spanning tree. We consider the following six cases:
Case 1: $i=0$. In $P_{\langle U\rangle}[i,j]$, $u=0$ and $v>j$. In $P_{\langle D\rangle}[i,j]$, $u=0$ and $v<j$. In $P_{\langle R\rangle}[i,j]$, $u>0$. Thus the three paths are internally disjoint.
Case 2: $i=m-1$ and $j=0$. In $P_{\langle U\rangle}[i,j]$, $v=n-1$ or $n-2$. In $P_{\langle D\rangle}[i,j]$, v=0 or 1 and $u<m-1$. In $P_{\langle R\rangle}[i,j]$, $v\le d\le n-3$ and $u=m-1$. Thus the three paths are internally disjoint.
Case 3: $i=m-1$ and $1\le j\le n-2$. There are five subcases:
Case 3.1: $j=d$. In $P_{\langle U\rangle}[i,j]$, $0<u+v\ge i+j$. In $P_{\langle D\rangle}[i,j]$, $0<u+v<i+j$. In $P_{\langle R\rangle}[i,j]$, $u+v=0$.
Case 3.2: $j>d$ and $i+j$ is odd. In $P_{\langle U\rangle}[i,j]$, $0<u\le m-1$ and $v>j$. In $P_{\langle D\rangle}[i,j]$, $u=0$ and $0<u+v<i+j$. In $P_{\langle R\rangle}[i,j]$, $u=m-1$ and $v<j$.
Case 3.3: $j<d$ and $i+j$ is odd. In $P_{\langle U\rangle}[i,j]$, $u=0$. In $P_{\langle D\rangle}[i,j]$, $u+v<i+j$. In $P_{\langle R\rangle}[i,j]$, $u+v<i+j$.
Case 3.4: $j>d$ and $i+j$ is even. In $P_{\langle U\rangle}[i,j]$, $u<m-1$ and $v\ge j$. In $P_{\langle D\rangle}[i,j]$, either $u=0$ and $v<j$ or $u=m-1$ and $v=j+1$. In $P_{\langle R\rangle}[i,j]$, $u=m-1$ and $v<j$.
Case 3.5: $j<d$ and $i+j$ is even. In $P_{\langle U\rangle}[i,j]$, either $u=0$ and $v>j$ or $u=m-1$ and $v=j-1$. In $P_{\langle D\rangle}[i,j]$, $u<m-1$ and $v\le j$. In $P_{\langle R\rangle}[i,j]$, $u=m-1$ and $v>j$.
In all of the five cases, the three paths are internally disjoint.
Case 4: $2\le i\le m-2$ and $j\le 1$. In $P_{\langle U\rangle}[i,j]$, $u\le i$ and either $v=n-1$ or $n-2$. In $P_{\langle D\rangle}[i,j]$, $u\le i$ and either v=0 or 1. In $P_{\langle R\rangle}[i,j]$, $u\ge i$. The three paths are internally disjoint.
Case 5: $2\le i\le m-2$ and $2\le j\le n-3$. In $P_{\langle U\rangle}[i,j]$, $u\le i$ and $v\ge j$. In $P_{\langle D\rangle}[i,j]$, $u\le i$ and $j>v$. In $P_{\langle R\rangle}[i,j]$, $u\ge i$ and $j\le v$. The three paths must be internally disjoint.
Case 6: $2\le i\le m-2$ and $j\ge n-2$. In $P_{\langle U\rangle}[i,j]$, $u\le i$ and $j=n-2$ or $n-1$. In $P_{\langle D\rangle}[i,j]$, $u\le i$ and $v<j$. In $P_{\langle R\rangle}[i,j]$, $u\ge i$ and $v\ge j$. The three paths are internally disjoint.

It turns out that every node can route three internally disjoint paths to the root node in the network. □

According to Lemmas 1 and 2, we give the following theorem.

Theorem 3. *For a single node, Algorithm* Parent_Determine *can be used to determine its parents in three IST on a GHT network in $O(1)$ time.*

5 Concluding Remarks

In this paper, a parallel algorithm is proposed to construct three ISTs on a GHT network. Based on the algorithm, each non-root node can determine its parents in different ISTs in constant time. The algorithm is easy to implement and has contribution in the one-to-many parallel routing of GHT networks. Our future work is to design parallel construction algorithms for other classes of node-transitive interconnection networks.

References

[1] B. Alspach and M. Dean, "Honeycomb toroidal graphs are Cayley graphs," Inform. Process. Lett., vol.109, pp.705–708, 2009.

[2] H.-J. Cho and L.-Y. Hsu, "Ring embedding in faulty honeycomb rectangular torus," Int. J. Comput. Math., vol.84, pp.277–284, 2002.

[3] H.-J. Cho and L.-Y. Hsu, "Generalized honeycomb tours," Inform. Process. Lett., vol.86, pp.185–190, 2003.

[4] J. Cheriyan and S.N. Maheshwari, "Finding nonseparating induced cycles and independent spanning trees in 3-connected graphs," J. Algorithms, vol.9, pp.507–537, 1988.

[5] S. Curran, O. Lee, and X. Yu, "Finding four independent trees," SIAM J. Comput., vol.35, pp.1023–1058, 2006.

[6] Q. Dong, X. Yang and J. Zhao, "Embedding a fault-free Hamiltonian cycle in a class of faulty generalized honeycomb tori," Comput. Elec. Eng., vol.35, pp.942–950, 2009.

[7] Q. Dong, Q. Zhao and Y. An, "The Hamiltonicity of generalized honeycomb torus networks," Inform. Process. Lett., vol.115, pp.104–111, 2015.

[8] L.-Y. Hsu, F.-I Ling, S.-S. Kao and H.-J. Cho, "Ring embedding in faulty generalized honeycomb torus GHT(m, n, $n/2$)," Int. J. Comput. Math., vol.87, pp.3344–3358, 2010.

[9] A. Itai and M. Rodeh, "The multi-tree approach to reliability in distributed networks," Inform. Comput., vol.79, pp.43–59, 1988.

[10] Y. Iwasaki, Y. Kajiwara, K. Obokata, and Y. Igarashi, "Independent spanning trees of chordal rings," Inform. Process. Lett., vol.69, pp.155–160, 1999.

[11] J.-C. Lin, J.-S. Yang, C.-C. Hsu and J.-M. Chang, "Independent spanning trees vs. edge-disjoint spanning trees in locally twisted cubes," Inform. Process. Lett., vol.110, pp.414–419, 2010.

[12] G.M. Megson, X. Yang and X. Liu, "Honeycomb tori are Hamiltonian," Inform. Process. Lett., vol.72, pp.99–103, 1999.

[13] G.M. Megson, X. Liu and X. Yang, "Fault-tolerant ring embedding in a honeycomb torus with nodes failures," Parallel Process. Lett., vol.9, pp.551–561, 1999.

[14] A.A. Rescigno, "Vertex-disjoint spanning trees of the star network with applications to fault-tolerance and security," Inform. Sci., vol.137, pp.259–276, 2001.

[15] Y.-K. Shih, Y.-C. Wu, S.-S. Kao and J. J.-M. Tan, "Vertex-bipancyclicity of the generalized honeycomb tori," Comput. Math. Appl., vol.56, pp.2848–2860, 2008.

[16] I. Stojmenović, "Honeycomb networks: topological properties and communication algorithms," IEEE Trans. Parallel Distrib. Syst., vol.8, pp.1036–1042, 1997.

[17] S.-M. Tang, Y.-L. Wang, and Y.-H. Leu, "Optimal independent spanning trees on hypercubes," J. Inform. Sci. Eng., vol.20, pp.143–155, 2004.

[18] S.-M. Tang, J.-S. Yang, Y.-L. Wang, and J.-M. Chang, "Independent spanning trees on multidimensional torus networks," IEEE Trans. Comput., vol.59, pp.93–102, 2010.

[19] S.-M. Tang, J.-S. Yang, J.-M. Chang and Y.-L. Wang, "A one-to-many parallel routing algorithm on a generalized recursive circulant graph," Proc. 31st Workshop on Combinatorial Mathematics and Computation Theory, Taipei, pp.29–36, 2014.

[20] X. Yang, D. J. Evans, H. Lai and G.M. Megson, "Generalized honeycomb torus is Hamiltonian," Inform. Process. Lett., vol.92, pp.31–37, 2004.

[21] X. Yang, G.M. Megson, Y. Tang and D.J. Evans, "Diameter of parallelogramic honeycomb torus," Comput. Math. Appl., vol.50, pp.1477–1486, 2005.

[22] J.-S. Yang, H.-C. Chan, and J.-M. Chang, "Broadcasting secure messages via optimal independent spanning trees in folded hypercubes," Discrete Appl. Math., vol.159, pp.1254–1263, 2011.

[23] J.-S. Yang and J.-M. Chang, "Independent spanning trees on folded hyper-stars," Networks, vol.56, pp.272–281, 2010.

[24] J.-S. Yang, S.-M. Tang, J.-M. Chang, and Y.-L. Wang, "Parallel construction of optimal independent spanning trees on hypercubes," Parallel Comput., vol.33, pp.73–79, 2007.

[25] A. Zehavi and A. Itai, "Three tree-paths," J. Graph Theory, vol.13, pp.175–188, 1989.

On weighted perfect target set selection

Lo Ming-Che and Chang Ching-Lueh
Department of Computer Science and Engineering
Yuan Ze University, Taoyuan, Taiwan
s1056017@mail.yzu.edu.tw (Lo Ming-Che)
clchang@saturn.yzu.edu.tw (Ching-Lueh Chang)

Abstract

Consider the following graph process on a vertex-weighted undirected graph $G=(V,E)$. Initially, a set of vertices are open. Whenever the total weight of a vertex v's open neighbors exceeds that of v's closed neighbors, v will be opened. The whole process continues until no more vertices can be opened. We show how to open $\lceil |V|/2 \rceil$ vertices so that all vertices are eventually open. Our proofs modify those of Khoshkhah et al. [2].

1 Introduction

All graphs in this paper are simple, vertex-weighted and undirected. Each vertex of a graph $G=(V,E)$ can be open or closed. All weights are positive. We denote the set of neighbors of $v \in V$ in G by $N_G(v)$.

At the beginning, we open some vertices of G. Whenever the total weight of a vertex v's open neighbors exceeds that of v's closed neighbors, v will be opened. The whole process continues until no more vertices can be opened. A perfect target set refers to a set of vertices whose opening will open all vertices at the end[1]. We show how to open $\lceil |V|/2 \rceil$ vertices so that all vertices are eventually open. Our proofs modify those of Khoshkhah et al.[2].

2 Opening the vertices

Theorem 1. Given any graph $G=(V,E)$ with positive vertex weights, a perfect target set of size at most $\lceil |V|/2 \rceil$ can be found in polynomial time.

Proof. Run the DFS algorithm on G to get a DFS tree T. For all $v \in V$, denote by $\mathrm{depth}(v)$ the depth of v in T and let T_v be the subtree of T rooted at v. Paint the leaves of T black or white arbitrarily. Inductively, having colored the vertices deeper in T than a vertex $v \in V$, color v white if the black vertices contribute more than half of the total weight of the neighbors (in G) of v in T_v and black otherwise.

Claim 1. For each $v \in V$, opening the following vertices will open v at the end:

- The vertices shallower than v in T.
- All black vertices.
- The root of T.

Proof of Claim 1.

- Case 1. v is black or is the root of T.

  Clearly, v is directly opened.

- Case 2. v is white and is not the root of T.

  Let
  $\alpha = \{u \in N_G(v) | \mathrm{depth}(u) < \mathrm{depth}(v)\}$,
  $\beta = N_G(v) \setminus \alpha$.
  The vertices in α are already opened because they are shallower in T than v. α contains at least one vertex because v has a parent in T. So the total weight of the vertices in α is positive. Because all vertices in $N_G(v)$ that are no shallower than v in T must be in T_v, β is the set of neighbors (in G) of v in T_v. So by construction, the total weight of the black neighbors (in G) of v in T_v is at least that of the white neighbors (in G) of v in T_v. In summary, the total weight of v's open neighbors exceeds that of v's closed neighbors. Fig. 1 illustrates this case.

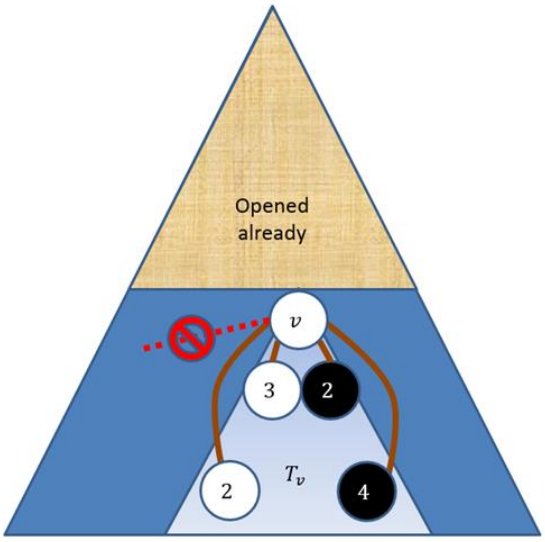


Figure 1. A simple illustration.

Figure 1. In Case 2, the vertices shallower than v in T are already open. All neighbors (in G) of v must be in T_v because a DFS tree cannot have a cross edge, i.e., the red dashed edge cannot exist. In this illustration, v has four neighbors (in G) deeper than v in T (note that those four neighbors are connected to v either by a tree edge or a back edge, but not by a cross edge). By our method of coloring, v is colored white because at least half of the total weight of v's neighbors (in G) deeper than v are contributed by black vertices—In this figure, the white neighbors of v deeper than v have a total weight of $3 + 2 = 5$, and the black counterpart has a greater total weight of $2 + 4 = 6$. Now in Case 2, at least half of the total weight of v's neighbors (in G) deeper than v are contributed by black vertices, and all of v's neighbors (in G) shallower than v are already open; hence v would be opened according to our graph process.

Let B be the set of non-root black vertices and W be the set of non-root white vertices. According to Claim 1, opening the root and all black vertices will open all vertices at the end. By symmetry, opening the root and all white vertices will open all vertices at the end. So there exist perfect target sets of sizes at most $1 + |B|$ and at most $1 + |W|$. Because $B \cup W$ is the set of non-root vertices, $|B| + |W| = |V| - 1$, implying

$$\min\{1 + |B|, 1 + |W|\} \leq \left\lceil \frac{|V|}{2} \right\rceil.$$

■

Acknowledgment

The authors are supported in part by the Ministry of Science and Technology of Taiwan under grant 106-2221-E-155-013-.

References

[1] Eyal Ackerman, Oren Ben-Zwi and Guy Wolfovitz, Combinatorial model and bounds for target set selection, *Theoretical Computer Science* 411 (2010) 4017–4022.

[2] Kaveh Khoshkhah, Hossein Soltani and Manouchehr Zaker, On dynamic monopolies of graphs: The average and strict majority thresholds, *Discrete Optimization* 9 (2012) 77–83.

基於正三角形鑲嵌之韋伯點近似解

Finding the Weber Point Based on Triangle Tiling

詹景裕
國立臺南藝術大學
校長室
gejan@mail.ntpu.edu.tw

施念齊
國立臺北大學
電機工程系
boy10166@gmail.com

Kevin Fung
國立清華大學
工業工程與
工程管理系
s106034402@m106.nthu.edu.tw

雒超民
美國底特律大學
電機工程系
luoch@udmercy.edu

摘要

本文以Alfred Weber所提出工業區位 (Industrial Location)及韋伯點 (Weber Points) 觀念為基礎，從歷年計算幾何學 (Computational Geometry)領域中的相關研究可以發現，對於韋伯理論的區位設置方面，尚未得知以多項式時間能夠求出韋伯點位置之演算法。歷年來學者對於尋找韋伯點的嘗試多以趨近為主，且演算過程中需花費不少的計算時間。

基於上述情況，本研究以啟發式(Heuristics)思維探討時間複雜度為$O(n)$之韋伯點近似解，並參考歐幾里得距離 (Euclidean Distance)與曼哈頓距離 (Manhattan Distance)兩者的特性，以正三角形鑲嵌 (Triangle Tiling)的方式著手，將平均值與中位數兩種起始三角形尋找方式納入演算法流程，並比較兩者所得到的近似點於歐幾里得距離 (Euclidean Distance)中的韋伯距離數據，其執行過程中也包含比較鄰近三角形重心計算出的韋伯距離。我們並於論文中整理實驗結果的比較資料，在來源點數量 (Source Points) |S|=3情況中透過本論文之演算法所得到的韋伯距離數據，與費馬點 (Fermat Point)所得到的韋伯距離數據僅相差百分之二以內。當來源點數量持續增加時，本演算法所找出的韋伯點趨近解為區域最佳解 (Local Optimum)之近似數據。

關鍵字：啟發法、正三角形鑲嵌、韋伯點、計算幾何學

一、 緒論

在工業的發展與工廠位置的選擇上，可以透過工業區位 (Industrial Location)的思考將生產過程的原物料、製造完成產品之運輸成本降到最低[1]。所謂的工業區位可以從地理位置設置開始考量，透過工廠位置的區位選擇來節省成本，過去曾提出工業區位 (Industrial Location)相關研究之學者包括 Fetter [2]以商品運送時，市場區域的距離遠近來探討區位設定對價格的影響，並於研究中透過 A、B 兩個市場的設定嘗試畫出各區域的價格預測；Hotelling [3]則在顧客相同的前提下，以工業區位觀念來描述價格和區位設定之間的關聯性；Von Thunen [4]以單獨的城市環境、各農夫將商品直接送到城市之基礎，探討農夫租用土地價格與市場距離之間的關係；Chang *et al.* [5]則認為找出可以滿足各地商品購買需求的最低成本或最高利潤之廠房位置，是在經濟學上相當重要的課題，並且根據其對經典韋伯理論的定義，本研究在歐幾里得空間(Euclidean Space)中，給定

$$S = \{s_i(x_i, y_i) | 0 \le i \le n-1\}$$

表示由下列點所組成的點集合：

$$\{x_1, x_2, x_3, \dots, x_n\}$$

我們也以下列方程式表示對於來源點的韋伯距離總和，並將其最小化：

$$min \sum_{i=1}^{n} |d^W - s_i|$$

能夠符合此條件的，我們稱之為韋伯點。

我們透過上述學者之研究歸納出兩項要點：運輸(Transportation)路徑的長度、各個來源點的權重 (Weight)，並將啟發式設計 (Heuristics) [6]納入本研究的演算法設計思考。由於德國學者 Alfred Weber (韋伯) [7]理論提出以運輸成本 (Transportation Cost)的誘因吸引企業至成本最小的地方設廠[8]，因此對於區位的決策上，傾向於將工廠設置在最低運輸成本的位置，來達到節省成本的效果[9]，並可將各點之人口數量 (Population) [10][11]以權重方式表現在數值資料中；本研究同時將歐幾里得距離 (Euclidean Distance) [12]及曼哈頓距離 (Manhattan Distance，也稱為 Taxicab Metric) [13][14]的觀念，與算術平均數 (Arithmetic Mean)及中位數 (Median)一起納入啟發式 (Heuristics)韋伯點近似解演算法的設計思考。所謂的算術平均數[15]是將 n 筆資料加總後除以總個數 n，可以由下列方程式表示：

$$\frac{\sum_{i=1}^{n} x_i}{n}$$

中位數則是將 n 筆資料排序後區分為前、後兩部分[16]，若 n 為奇數，排序最中間的資料即為中位數；若 n 為偶數，則將排序中間的兩筆資料相加後取其算術平均數作為中位數。而歐幾里得距離是指在歐幾里得空間中兩點的直線距離，若在一個歐幾里得空間中的點集合為 $S = \{s_i(x_i, y_i) | 0 \le i \le n-1\}$，則任意兩點 $s_i(x_i, y_i)$ 及 $s_{i+1}(x_{i+1}, y_{i+1})$的歐幾里得距離$D_i$可以表示為：

$$D_i = \left\{ \sqrt{(y_{i+1} - y_i)^2 + (x_{i+1} - x_i)^2} \,\middle|\, 0 \le i \le n-1 \right\}$$

而$s_i(x_i, y_i)$及$s_{i+1}(x_{i+1}, y_{i+1})$兩點間的曼哈頓距離表示為：

$$|x_i - x_{i+1}| + |y_i - y_{i+1}|$$

本研究以中位數 (Median)與數據統計中常使用的算術平均數 (Arithmetic Mean)兩種 Heuristics 一起納入本演算法的設計考量，並在第四章將這兩種方向所得到的韋伯近似點以歐幾里得距離 (Euclidean Distance)數值做比較。本論文提出之演算法並結合正三角形鑲嵌(Triangle Tiling)之佈建方式，以 $O(n)$時間複雜度提供韋伯點近似解，實際步驟將於後續章節做詳細說明。

業界的工作團隊選擇設廠位置時，必須考量工廠與各點間之運輸(Transportation)路徑[17][18]，並考慮各點之權重 (Weight)參數；回顧 Drezner [19]對於 Weber 理論的敘述，可以透過其作為出發點，撰寫本論文對於韋伯點的定義，延伸前述韋伯點條件之方程式，可以加入權重參數，我們將權重以p_i表示：

$$min \sum_{i=1}^{n} (|d^W - s_i| * p_i)$$

然而若透過先前學者提出之 Weiszfeld Procedure 以迭代方式求解上述方程式時，根據 Boltyanski *et al.*[20]對於 Weiszfeld 方法的分析，若以數學觀點 (Mathematical Aspect)繼續進行計算，其演算過程會有 stuck 之情形，須再透過其他方式設法解決；因此，本研究以電機資訊領域的離散觀點 (Discretization Aspect)為出發點，以趨近(Approximation)的方式設計演算法，提供擁有實際工業區位設置需求的使用者以軟體方式得到韋伯點趨近解，並從 Durocher *et al.*[21]可以知道，點座標通常以實際位置離散化 (Discretization)至鄰近的網格座標來代表，也就是說每個點的實際位置是由其鄰近的網格點來趨近(approximated)；且根據 Jones [22]對於韋伯相關研究的看法，歷年來尚未有單純以數學分析方式確實解決韋伯區位設置之研究，Bajaj [23]也提到在 S 中的來源點數量$n \geq 5$時，尚未能直接解出以韋伯定義所列出之多項式 (not solvable)，且目前並未有演算法可以透過算術運算(Arithmetic Operations)實際找出方程式的根 (Roots)，Rosen [24]並提到目前尚未得知多項式時間 (Polynomial Time)之韋伯演算法，過去對於韋伯點的尋找嘗試也以 Approximation 為主要目標[25][26]。Anderegg [27]在 2003 年的文獻中也提到目前沒有任何有限 (finite)的演算法可以解出真正韋伯點的實際位置；同時考量 Stanimirović *et al.* [28]表示韋伯在區位理論相關文獻中是最受重視之模型，因此本研究選擇以 Weber 所提出的工業區位模型為基礎，找到韋伯點近似解以提供設廠業者之位置指引，達成以最小韋伯距離節省運輸成本之效果。且我們將於演算法執行過程中，以正三角形均勻分布於矩框內，並將鄰近正三角形重心與來源點 (Source Points)計算出的韋伯距離比較來完成演算法的運算過程。時間複雜度方面，回顧 Jan 等人在 2017 年所提出之韋伯點趨近方式[29]，當時透過洋蔥趨近韋伯點的時間複雜度為 $O(n^{1.5})$。在本研究中，我們希望能透過啟發法 (Heuristics)的方式，先以探索 Local Optimum 為首要目標，進而提供未來學者對於 Global Optimum 能有更進一步的研究[30]。時間複雜度方面，我們也以設計 $O(n)$時間複雜度的演算法作為思考方向，並透過幾何方式設計的正三角形鑲嵌產生近似點。

本文以啟發法 (Heuristics)的角度，透過分析中位數(Median)與算術平均數 (Arithmetic Mean)兩種 Heuristics 為出發點的韋伯點近似數據，提供使用者以正三角形鑲嵌演算法近似出的趨近解。我們於執行過程中將韋伯點所在區域以正三角形均勻分布，並將鄰近正三角形重心與來源點集合計算出的韋伯距離比較，透過正三角形鑲嵌之方式進行運算。回顧 Jan 等人在 2017 年所提出以洋蔥之韋伯點趨近方式，主要目標為透過洋蔥之演算法設計嘗試找出韋伯點，該篇研究的洋蔥演算法時間複雜度為 $O(n^{1.5})$；而本研究之正三角形鑲嵌演算法，可以提供以離散觀點所設計的新版韋伯點趨近方式，並且經過各步驟的效能分析，是以時間複雜度 $O(n)$完成韋伯點近似解之計算。

二、 相關文獻

2.1 韋伯理論相關文獻

本研究以Alfred Weber的工業區位設置概念為基礎，根據Drezner *et al.*[31]所提到的情境範例，可以將韋伯工業區位課題設想成：我們需要尋找一個地點來建置倉庫，由這個地點運送貨品到位於(x_i, y_i)的顧客手中，若權重 (Weight)表示為p_i，以$d_i(x_i, y_i, x_k^c, y_k^c) = \sqrt{(x_k^c - x_i)^2 + (y_k^c - y_i)^2}$表示$(x_k^c, y_k^c)$和$(x_i, y_i)$的歐幾里得距離，並根據Cooper[32]對於韋伯理論的詮釋，第一章所述含有權重 (Weight)之韋伯點定義可以稱為廣義韋伯問題 (Generalized Weber Problem)：

$$min \left[\sum_{i=1}^{n} (|d^W - s_i| * p_i) \right]$$

而另一種定義方式為透過Dual Formulation，考量到前述的兩種定義中，廣泛採用的是前述的第一種定義，本演算法的韋伯距離D^W及廣義韋伯距離D^P皆以該定義作為設定之基礎，將在第三章詳細說明。在本文的第一章曾提到所謂啟發法 (Heuristics)，根據Pearl [33]的說明，啟發法為透過可獲得且能夠讀取的資料，來設計實際解決方案之分析動作。我們將上述動作與計算幾何學的觀念結合，以中位數與算術平均數搭配正三角形均勻分布的演算法執行方式進行實驗，並將鄰近正三角形重心與來源點 (Source Points)計算出的韋伯距離比較，希望透過以正三角形鑲嵌之方式得到韋伯點的近似解。

2.2 鑲嵌方式相關文獻

本演算法部分的設計發想源自於以鑲嵌 (Tiling)方式[34]將歐基里得平面 (Euclidean Plane) [35]中包含來源點 (Source Points)的二維空間透過幾何形狀充填，以達到佈滿該區域的目標。須將區域佈滿的主要因素是在演算法後續韋伯點趨近過程中，可以透過所使用的幾何形狀，在每輪的演算法運算流程裡嘗試尋找該輪運算當中，近似點可能存在的三角形區域；在本論文中，我們將這個近似點可能存在的區域稱為潛在三角形T^{Pot}。並且，考量幾何形狀的選擇時，以幾何多邊形鑲嵌的方向為出發點，設計規律的填充方式與演算法步驟結合。參考相關學者對於相同形狀之幾何鑲嵌相關研究[36][37]，可以透過該幾何圖形邊的數量，歸納出以下數種方式：正三角形鑲嵌、正方形鑲嵌、正六邊形鑲嵌，據此可延伸出更多邊長之鑲嵌方式。

表1 本研究演算法之變數代號及其解釋

變數代號	名詞解釋
w	韋伯
S	來源點集合；$S=\{s_i(x_i,y_i)\|0\le i\le n-1\}$
H	輔助點集合；$H=\{h_i(x_i,y_i)\|0\le i\le n-1\}$
$A^{\square}$	最小矩框面積
A^{Δ}	鑲嵌之正三角形面積
D_i	點(x_k^c,y_k^c)與來源點$s_i(x_i,y_i)$之距離公式；$D_i=\left\{\sqrt{(x_k^c-x_i)^2+(y_k^c-y_i)^2}\middle\|0\le k\le n-1,0\le i\le n-1\right\}$
C	重心集合；$C=\{c_k(x_k^c,y_k^c)\|0\le k\le n-1\}$
D^W	韋伯距離；$\sum_{i=1}^{n}\|d^W-s_i\|$
D^P	廣義韋伯距離；$\sum_{i=1}^{n}(\|d^W-s_i\|\ *\ p_i)$, p_i 為各個來源點的人口數，$D^P=D^W\ \forall\ p_i=1$
$D_{i,j}^W(k)$	韋伯距離集合； $D_{i,j}^W(k)=\left\{d_{i,j}^W(k)\middle\|2\le i\le n-1,\ j=3,0\le k\le K-1\right\}$ i為來源點， j重心，k剖分次數，K為常數依ε而定
$D_{i,j}^P(k)$	廣義韋伯距離集合； $D_{i,j}^P(k)=\left\{d_{i,j}^P(k)\middle\|2\le i\le n-1,\ j=3,0\le k\le K-1\right\}$ i為來源點， j重心，k剖分次數，K為常數依ε而定
$Diff$	誤差值計算；$Diff_{i,j}^{(k)}=\left\|d_{i,j}^W(k)-d_{i,j}^W(k+1)\right\|\le\varepsilon$
T^{Can}	鄰近的候選三角形
T^{Pot}	韋伯點可能存在的潛在三角形
x^{max},y^{max} x^{min},y^{min} x^{avg},y^{avg} x^{med},y^{med}	來源點集合中找出x,y座標值最大的點，記錄其x,y座標值 來源點集合中找出x,y座標值最小的點，記錄其x,y座標值 來源點集合中所有點之x,y座標平均值 來源點集合中所有點之x,y座標中位數
T	k次正三角形剖分過程中所產生之子三角形集合； $T=\{t(k)\|0\le k\le K-1\}$

本提案的演算法設計考量到韋伯點之趨近目標、剖分邊之數量較小與剖分過程之規律性質，採用每次剖分均可產生相同形狀之正三角形鑲嵌，以離散觀點提供韋伯點的趨近解。

三、 演算法與圖解

3.1 演算法參數及其說明

本節開始介紹透過正三角形鑲嵌之韋伯點近似解尋找方法，在演算法的介紹過程中將會使用到以代號表示的參數，整理如表1所示。

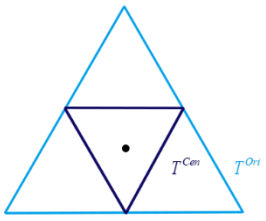


圖 4 將正三角形剖分為四個小正三角形

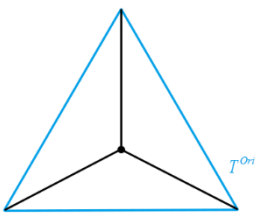


圖 5 將正三角形剖分為三個小三角形

3.2 演算法執行步驟

在本節的內容中，將介紹以最小矩框、正三角形鑲嵌設計的韋伯點趨近演算法。首先透過來源點決定最小矩框，並計算正三角形之面積大小；接著嘗試尋找韋伯點的潛在三角形，並與鄰居三角形比較廣義韋伯距離，若原潛在三角形之廣義韋伯距離最小，則進行三角形剖分，以產生符合使用者ε定義之廣義韋伯點趨近解。我們在演算法步驟的設計中，也考量到透過連結正三角形各邊中點，將原三角形剖分成四個小正三角形時，原正三角形T^{Ori}之重心會與所剖分出的四個小正三角形中的T^{Cen}之重心重疊，如圖4範例所示。因此，當演算法的執行過程中遇到上述情況，我們將連結原正三角形T^{Ori}的重心與三個頂點，從潛在三角形中剖分出三個小三角形，這三個小正三角形各自的重心將與原正三角形T^{Ori}的重心為不同點，如圖5範例所示。

Function 1:　　最小矩框
Begin
Step 1　　由S中找出x,y座標值最大與最小點，分別記錄為x^{max},x^{min},y^{max}及y^{min}

Step 2　以線段連接(x^{max}, y^{max})、(x^{max}, y^{min})、(x^{min}, y^{max})及(x^{min}, y^{min})四個點，得到面積為$A^{□}$的最小矩框

End

Function 2:　以正三角形均勻剖分最小矩框

Begin

Step 1　計算鑲嵌之正三角形面積 $(\frac{A^{□}}{|S|})=A^{\Delta}$

Step 2　將最小矩框之全部區域以面積 A^{Δ} 的正三角形均勻剖分

End

Function 3:　正三角形剖分

Begin

連結正三角形$t(k)$各邊中點，將原三角形剖分成四個小正三角形

End

Function 4:　韋伯趨近解

Begin

Step 1　計算潛在三角形T^{Pot}重心的廣義韋伯距離D^P，並與鄰近之三個三角形重心的廣義韋伯距離比較，取其最小者

Step 2　若Step 1求出之廣義韋伯距離D^P最小者，其所在之三角形為原先的潛在三角形T^{Pot}，

則進行Step 3；

否則$T^{Pot} \leftarrow T^{Can}$（將潛在三角形$T^{Pot}$設為$T^{Can}$），回到Step 1

Step 3　呼叫**Function 3** 對潛在三角形進行剖分，並將原先潛在三角形求出之廣義韋伯距離與剖分出來的三角形中得到的最小廣義韋伯距離比較，若$D^P_{i,j}(k)$與$D^P_{i,j}(k+1)$的誤差值$Diff \leq \varepsilon$且$Diff \neq 0$即停止；若$D^P_{i,j}(k)$與$D^P_{i,j}(k+1)$誤差值$Diff = 0$，則連結潛在三角形的重心與三個頂點，從潛在三角形中剖分出三個小三角形，並將原先潛在三角形求出之廣義韋伯距離與剖分出來三角形中得到的最小廣義韋伯距離比較，當$D^P_{i,j}(k)$與$D^P_{i,j}(k+1)$的誤差值$Diff \leq \varepsilon$即停止；否則將$D^P_{i,j}(k+1)$所在的三角形設為新的潛在三角形T^{Pot}，回到Step 1

End

演算法：基於正三角形鑲嵌之韋伯點近似解

Begin

Step 1　初始化在歐基里德空間的所有來源點

Step 2　呼叫**Function 1**找出最小矩框

Step 3　進行正三角形鑲嵌，並找出韋伯點可能存在的正三角形範圍

Step 3.1　呼叫**Function 2**將最小矩框以相同大小正三角形均勻剖分

Step 3.2　包括算術平均數 (Arithmetic Mean)以及中位數 (Median)兩種Heuristics，每次選擇其中一種執行：

Case 1. 算術平均數法：將S中所有點之x,y座標值加總，並除以S的總個數來計算x^{avg}, y^{avg}，以此點所在的正三角形作為韋伯點可能存在的潛在三角形區域T^{Pot}

Case 2. 中位數法：從S中所有點之x,y座標值中，根據S的總個數找出中位數點所在的座標值x^{med}, y^{med}，以此點所在的正三角形作為韋伯點可能存在的潛在三角形區域T^{Pot}

Step 4　找出韋伯點之近似解

呼叫**Function 4**外移或剖分子三角形，並找出誤差值$Diff$小於或等於ε的趨近解

End

3.3　演算法實際執行範例

本節以正三角形鑲嵌之方式，以演算法所敘述之順序，提供演算法於程式執行時的圖片範例，來源點 (Source Points)的產生方式包含隨機產生與使用者定義兩種方式：

第一種為隨機產生來源點之x,y座標值及相對應的Population數值，並由使用者設定ε，具體包含以下五個步驟：隨機產生來源點、透過座標值產生最小矩框、輸入（佈滿）正三角形鑲嵌、顯示韋伯點之潛在三角形、以及完成韋伯點之近似；若使用者選擇隨機產生的選項，將會由程式透過檔案輸出的方式，將第一步驟所產生的來源點記錄並儲存在檔案中。

第二種為使用者提供來源點數量、x,y座標值及相對應的Population數值，並透過檔案方式記錄，由程式讀入所記錄的資料，所使用的紀錄格式與前段所述之儲存格式相同。

以下為將x,y,p範圍以電腦螢幕顯示的1000*1000 像素 (Pixels)為考量，皆設定為大於等於0，小於等於999的正整數，因此最小顯示座標值為 (0,0)，最大顯示座標值為 (999,999)；以隨機產生250個來源點時，各步驟的執行範例，包括隨機產生來源點、產生最小矩框、輸入正三角形鑲嵌、顯示潛在三角形，並完成韋伯點的近似。

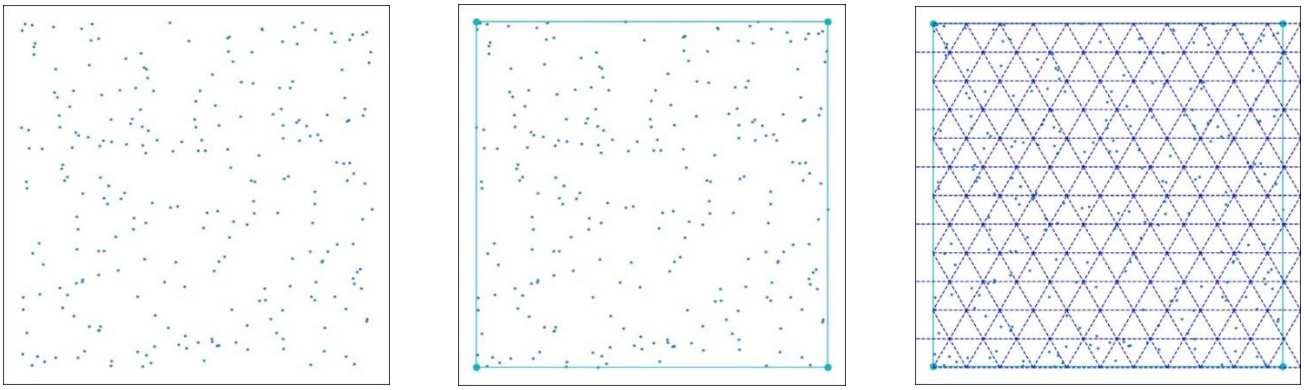
四、 實驗數據與演算法效能分析

本研究以正三角形鑲嵌作為主要思考方向，透過下列以啟發法 (Heuristics)所設計的兩種起始 (Initial)潛在三角形尋找方法作為演算法的開端：算術平均數(Arithmetic Mean)及中位數 (Median)。本章將以演算法所敘述的方法順序，詳細解說各步驟的時間複雜度，並將各項時間複雜度進行整理，計算透過本演算法進行韋伯點近似所需的總時間複雜度，且透過實作的運算時間統計，可以得知統計結果與總時間複雜度的計算相符。我們同時將上述兩種潛在三角形尋找方法，將多種來源點數量所產生近似韋伯點之韋伯距離資料作比較，提供給後續相關研究人員參考。

4.1計算時間複雜度

本節內容以演算法所敘述之順序，詳細解說各步驟的預期時間複雜度，同時將各項時間複雜度進行整理，整合透過本演算法進行韋伯點近似所需的總時間複雜度。透過實作的運算時間統計，整合後的數值符合總時間複雜度的計算結果。

步驟一與步驟二的運算對象為小於或等於n個來源點，並由S中找出x座標值最大點x^{max}及最小點x^{min}，與y座標值最大點y^{max}及最小點y^{min}；而步驟二將(x^{max}, y^{max})、(x^{max}, y^{min})、(x^{min}, y^{max})及(x^{min}, y^{min})四個點



a.初始化歐幾里得空間的所有來源點　b.透過座標值產生最小矩框　c.輸入（佈滿）正三角形鑲嵌

d.顯示韋伯點之潛在三角形　e.顯示剖分過程　f.完成韋伯點近似

圖6 演算法運作範例圖

以連接線段方式形成最小矩框，兩步驟的總時間複雜度為$O(n)$。

在演算法的步驟三中，主要目標為進行正三角形鑲嵌及找出韋伯點可能存在的潛在三角形，包括呼叫Function 2將最小矩框佈滿三角形，以及透過S中的n個點之x座標值計算x^{avg}或x^{med}，y座標值計算y^{avg}或y^{med}，所需之時間複雜度皆為$O(n)$。

步驟四中以呼叫Function 4近似出韋伯點之趨近解，包含下列三項動作：第一項，對S中的n個點計算廣義韋伯距離；第二項，從鄰居中找出演算法該輪過程中近似點可能存在的三角形範圍，我們將這個範圍稱為潛在三角形T^{Pot}；第三項，將原先的潛在三角形T^{Pot}剖分k次之動作；其中第一項動作是將S中n個點的Euclidean Distance乘上相對應的Weight數值後加總；第二項動作為比較前一輪所找到近似點之韋伯距離，與該點所在三角形共邊的鄰近三個三角形重心之韋伯距離；第三項動作是根據每輪演算法當中在前述第二項動作的比較結果，若比較數據$D_{i,j}^{P}(k)$與$D_{i,j}^{P}(k+1)$的誤差值$Diff \leq \varepsilon$即停止運算並在螢幕上顯示近似點，否則若該近似點的韋伯距離比較結果最小，即在該近似點所在三角形內進行剖分；由於ε為使用者於演算法開始執行前所設定的常數 (Constant)，對於整個演算法運作過程所進行的剖分次數k，是根據每輪執行過程的比較結果是否符合使用者所設定之誤差值條件$Diff \leq \varepsilon$而定，當演算法滿足誤差值條件時，我們可以得知剖分次數k為一有限數 (Finite Number)；綜合以上幾項分析的內容，我們可以知道本步驟的時間複雜度為小於或等於$O(n)$。

將前述四個步驟的時間複雜度整合，我們可以得知，以本研究提出的正三角形鑲嵌演算法求出韋伯點近似解，所需的時間複雜度為 $O(n)+O(n)+O(n)=O(n)$。

4.2執行時間統計

本研究透過正三角形鑲嵌之演算法設計進行韋伯點的近似，關於程式執行的測試平台方面，作業系統為Windows 10，其中央處理器為Intel ® Core ™ i5-4210M CPU，運作時脈為2.60 GHz，搭配已安裝之16 GB隨機存取記憶體。本節測試平台的軟硬體相關資訊整理於表3。

本文之正三角形鑲嵌演算法以Windows 64位元系統環境之Python 3.6實作，在執行效能的實驗中，我們以500至1500個來源點做為實驗範圍，以秒 (Second)為單位，統計執行出韋伯點結果之時間數據（如表4），每項時間結果數據皆為實驗之平均數值，所測量到的數據包含演算法本身實際運算之時間以及透過Python圖形介面函式庫(Library)繪製結果於螢幕上之時間。本節根據表4之演算法執行時間資料繪製折線圖（如圖7），可以觀察其成長趨勢發現演算法執行時間與前一節預估之時間複雜度$O(n)$相符合。

表4 演算法及圖形顯示之執行時間統計

隨機來源點數量	演算法執行總時間 (秒)	圖形顯示繪製總時間 (秒)
500	0.257	1.495
750	0.362	2.187
1000	0.434	3.179
1250	0.516	4.431
1500	0.603	6.039

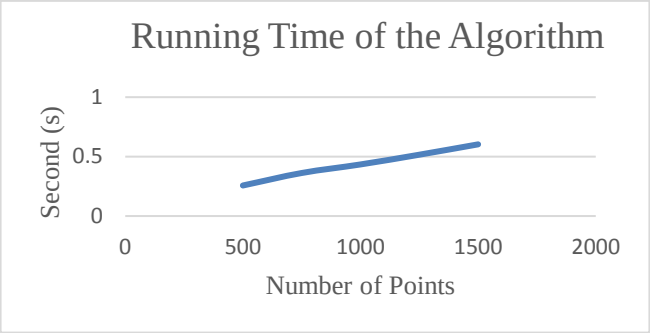


圖7 演算法執行時間折線圖

4.3韋伯距離數據比較

我們在本節中將算術平均數 (Arithmetic Mean)以及中位數 (Median)兩種透過啟發法 (Heuristics)技巧所設計的方向，各自所得到的近似最小韋伯距離數據在不同的來源點範圍中，以相同的來源點數量及相同權重的執行結果範例進行比較，其比較結果呈現於表5中，其中隨機來源點數量、以算術平均數作為起始點三角形尋找方式所得到的韋伯距離結果、以中位數作為起始點三角形尋找方式所得到的韋伯距離結果、以及韋伯距離數據比較，分別以$|S|$、D^{avg}、D^{med}、$D^{avg}-D^{med}$表示。

我們同時考量來源點數量$|S|=3$的情況時，韋伯點(Weber Point)的定義為找到一個點使該點到這三個點的距離總和最小，而費馬點 (Fermat Point)的定義為該點對於三角形三個點的距離總和最小[38]；費馬點的尋找方式如圖8所示，是將二維平面上的三個點P_1、P_2、P_3以線段$\overline{P_1P_2}$、$\overline{P_2P_3}$、$\overline{P_1P_3}$連接成一個三角形，以上述線段所在的邊，各自向外做出正三角形$\Delta P_1P_2P_4$、$\Delta P_1P_3P_5$、$\Delta P_2P_3P_6$，並將$\overline{P_1P_6}$、$\overline{P_2P_5}$、$\overline{P_3P_4}$三個線段連接，其交點即為費馬點；本節參考上述兩種定義，提供本演算法以算術平均數(Arithmetic Mean)及中位數 (Median) 所產生近似點計算出的韋伯距離，與費馬點計算出之韋伯距離實驗數據比較，比較結果呈現於表6中，並將每組資料對應的三角形呈現於圖9；其中三角形以T^i表示，i為來源點的編號；而透過算術平均數作為起始三角形尋找方式、以及中位數作為起始三角形尋找方式時，經由本演算法所產生的近似點計算出之韋伯距離，以D^{avg}及D^{med}表示；對應的費馬點計算出之韋伯距離，則以D^{fer}表示。

並且，我們希望從二維空間中不同位置的點所計算出之韋伯距離數據，以統計圖的方式觀察其分布情形，圖10為透過三維空間表面 (3-D Surface) 方式，透過10000個來源點在Normal Distribution時所計算出之韋伯距離，進行繪製而成的範例統計圖形，x,y座標值對應原二維空間中各點x,y所表示之位置，z座標值則表示該點所對應的韋伯距離數據，其顯示出的形狀與凹面 (Concave)相似；對於該圖形進行觀察可以發現，在這10000個來源點所計算出的韋伯距離數值當中，全局最小 (Global Minimum)的數值應位在整個凹面之中央區域；而本演算法的兩種起始三角形尋找方式，包含算術平均數法與中位數法，應可透過找出更為接近全局最小數值位置的起始點，來節省整個運算過程所需的時間。若將圖10與圖11所示之以其它數據繪製的範例圖形比較，可以發現在圖10的形狀中，全局最小數值的所在區域較容易透過凹面的形狀觀察出來。若從來源點統計出的韋伯距離數據圖形與圖11相似，則透過演算法所找出的近似點之韋伯距離可能為區域最小 (Local Minimum)，而不一定是全局最小 (Global Minimum)。

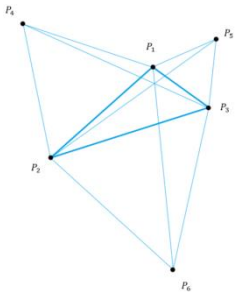
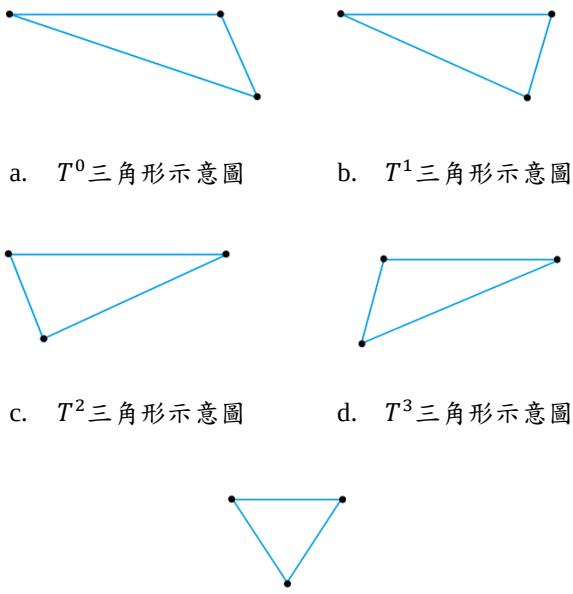


圖8 來源點數量$|S|=3$情況時之費馬點示意圖



a. T^0三角形示意圖　b. T^1三角形示意圖

c. T^2三角形示意圖　d. T^3三角形示意圖

e. T^4三角形示意圖

圖9 來源點數量$|S|=3$情況時之三角形範例示意圖

表5　算術平均數與中位數之韋伯距離數據比較

$\|S\|$	D^{avg}	D^{med}	$D^{avg} - D^{med}$
3	487.542	487.263	0.279
9	3321.549	3321.549	0
500	198456.609	198537.755	-81.146
750	281628.436	281628.436	0
1000	384689.613	384689.613	0

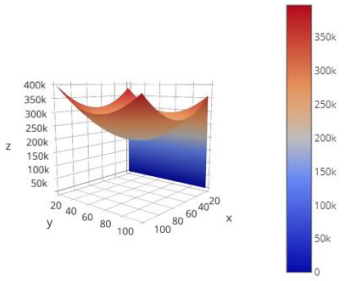


圖10 三維空間表面 (3-D Surface)韋伯距離數據範例統計圖

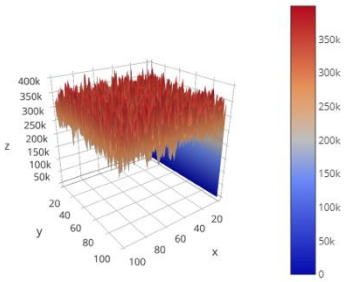


圖11 三維空間表面 (3-D Surface)其它數據範例統計圖

五、 結論

本研究所設計的演算法，於執行過程中將矩框所在區域以正三角形均勻分布，並比較鄰近正三角形重心的韋伯距離數據，透過正三角形鑲嵌之方式進行韋伯點近似點的尋找。由於韋伯理論的區位設置方面，尚未得知以多項式時間能夠求出韋伯點位置之演算法，且歷年來學者對於尋找韋伯點的嘗試多以趨近為主。因此，我們以啟發式設計 (Heuristics) 作為出發點，並參考歐幾里得距離 (Euclidean Distance)與曼哈頓距離 (Manhattan Distance)兩者的特性，以正三角形鑲嵌 (Triangle Tiling)的方式著手，以算術平均數與中位數兩種起始三角形尋找方式設計演算法，並比較兩者所得到的近似點之韋伯距離以歐幾里得距離 (Euclidean Distance)計算之結果。

本論文提出之正三角形鑲嵌演算法，透過前一章內容所敘述的效能分析可以得知，本方法是以時間複雜度 $O(n)$完成韋伯點近似解之計算。我們並在論文中提供算術平均數與中位數作為起始三角形尋找方式時，以演算法所產生之近似點與費馬點兩者韋伯距離的比較資料。在來源點數量 (Source Points) $|S| = 3$情況中透過本論文之演算法所得到的韋伯距離數據，與費馬點 (Fermat Point)所得到的韋伯距離數據僅相差百分之二以內。當來源點數量持續增加時，本演算法所找出的韋伯點趨近解為區域最佳解 (Local Optimum)之近似數據。

參考文獻

[1] Z. Drezner and A. Goldman, "On the set of optimal points to the Weber problem," *Transportation science*, vol. 25, no. 1, pp. 3-8, 1991.

[2] F. A. Fetter, "The economic law of market areas," *The Quarterly Journal of Economics*, vol. 38, no. 3, pp. 520-529, 1924.

[3] H. Hotelling, "Stability in competition," *The economic journal*, vol. 39, no. 153, pp. 41-57, 1929.

[4] J. H. v. Thünen, P. Hall, and J. H. v. Thünen, "Von Thunen's isolated state," 1966.

[5] K.-Y. Chang, C.-M. Su, G. Jan, and C. Chen, "An efficient method for single-facility location and path-connecting problems in a cell map," *International Journal of Geographical Information Science*, vol. 27, no. 10, pp. 2060-2076, 2013.

[6] A. Tversky and D. Kahneman, "Judgment under uncertainty: Heuristics and biases," *science,* vol. 185, no. 4157, pp. 1124-1131, 1974.

[7] A. Weber, *Ueber den standort der industrien*. Рипол Классик, 1909.

[8] E. Carrizosa and A. M. Rodríguez-Chía, "Weber problems with alternative transportation systems," *European Journal of Operational Research*, vol. 97, no. 1, pp. 87-93, 1997.

[9] A. Weber and C. J. Friedrich, "Alfred Weber's theory of the location of industries," 1929.

[10] Z. Drezner and J. Guyse, "Application of decision analysis techniques to the Weber facility location problem," *European Journal of Operational Research*, vol. 116, no. 1, pp. 69-79, 1999.

[11] E. Carrizosa, E. Conde, M. Muñoz-Marquez, and J. Puerto, "The generalized Weber problem with expected distances," *RAIRO-Operations Research*, vol. 29, no. 1, pp. 35-57, 1995.

[12] J. Dattorro, *Convex optimization & Euclidean distance geometry*. Lulu. com, 2010.
[13] V. Perlibakas, "Distance measures for PCA-based face recognition," *Pattern Recognition Letters*, vol. 25, no. 6, pp. 711-724, 2004.
[14] E. F. Krause, *Taxicab geometry: An adventure in non-Euclidean geometry*. Courier Corporation, 1975.
[15] N. Aidara, "Introduction to probability and statistics," 2018.
[16] S. Dean and B. Illowsky, "Descriptive statistics: skewness and the mean, median, and mode," *Connexions website http://www.connexions.org. Accessed,* vol. 3, 2017.
[17] E. V. Weiszfeld, "Sur le point pour lequel la somme des distances de n points donnes est minimum", *Tohoku Mathematical Journal*, Vol. 43, pp. 355-386, 1937.
[18] I. Hong and A. T. Murray, "Efficient measurement of continuous space shortest distance around barriers", *International Journal of Geographical Information Science*, Vol. 27, No. 12, pp. 2302-2318, 2013.
[19] Z. Drezner, "A note on the Weber location problem," *Annals of Operations Research*, vol. 40, no. 1, pp. 153-161, 1992.
[20] V. Boltyanski, H. Martini, and V. Soltan, *Geometric methods and optimization problems*. Springer Science & Business Media, 2013.
[21] S. Durocher and D. Kirkpatrick, "The projection median of a set of points," *Computational Geometry*, vol. 42, no. 5, pp. 364-375, 2009.
[22] E. Jones, "The 3-vertex single source Weber location problem," *International Journal of Mathematical Education in Science and Technology*, vol. 19, no. 5, pp. 671-679, 1988.
[23] C. Bajaj, "The algebraic degree of geometric optimization problems," *Discrete & Computational Geometry*, vol. 3, no. 1, pp. 177-191, 1988.
[24] K. H. Rosen, *Handbook of discrete and combinatorial mathematics*. CRC press, 1999.
[25] R. Chandrasekaran and A. Tamir, "Algebraic optimization: the Fermat-Weber location problem," *Mathematical Programming*, vol. 46, no. 1, pp. 219-224, 1990.
[26] P. Indyk, "Sublinear time algorithms for metric space problems," in *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, 1999, pp. 428-434: ACM.
[27] L. Anderegg, M. Cieliebak, and G. Prencipe, "The weber point can be found in linear time for points in biangular configuration," ed: Università di Pisa, 2003.
[28] P. S. Stanimirović, M. S. Ćirić, L. A. Kazakovtsev, and I. A. Osinuga, "Single-facility Weber location problem based on the lift metric," *Facta universitatis-series: Mathematics and Informatics*, vol. 27, no. 2, pp. 175-190, 2012.
[29] G. E. Jan, K. Y. Chang, W. J. Chou and Y. C. Lin, "The Approximation of Weber Point," *Proceedings of National Computer Symposium*, #7376, National Dong Hwa University, Taiwan, Dec. 2017.
[30] S. Lim and J. Zhu, "Integrated data envelopment analysis: Global vs. local optimum," *European Journal of Operational Research*, vol. 229, no. 1, pp. 276-278, 2013.
[31] L. Cooper, "An extension of the generalized Weber problem," *Journal of Regional Science*, vol. 8, no. 2, pp. 181-197, 1968.
[32] J. Pearl, "Heuristics: intelligent search strategies for computer problem solving," 1984.
[33] B. Grünbaum and G. C. Shephard, *Tilings and patterns*. Freeman, 1987.
[34] P. Mattila, *Geometry of sets and measures in Euclidean spaces: fractals and rectifiability.* Cambridge university press, 1999.
[35] H. Steinhaus, *Mathematical snapshots*. Courier Corporation, 2012.
[36] P. S. Stevens and C. P. Stevens, *Handbook of regular patterns: an introduction to symmetry in two dimensions.* MIT Press Cambridge, MA, 1981.
[37] P. Spain, "The Fermat point of a triangle," *Mathematics Magazine*, vol. 69, no. 2, pp. 131-133, 1996.

二次曲面上韋伯問題之趨近解

Approximation of Weber Point on Quadratic Surfaces

詹景裕
國立臺南藝術大學
校長室
gejan@mail.ntpu.edu.tw

雒超民
美國底特律大學
電機工程系
luoch@udmercy.edu

Kevin Fung
國立清華大學
工業工程與
工程管理系
s106034402@m106.nthu.edu.tw

張啟隱
國立臺灣海洋大學
商船學系
b0170@mail.ntou.edu.tw

摘要

在空間經濟學以及作業研究領域中，韋伯問題一直是近百年來相當經典的問題。例如於歐幾里得空間中的找出某一點設為賣場，使得此賣場與各相鄰城鎮的距離總和最小，此點即是韋伯點(Weber Point)。

本文以啟發式(Heuristics)思維提出如下二次曲面上韋伯問題之趨近解。其主要方法為以在三角網格中使用Ahuja-Dijkstra和脊點方式來產生一距離成本累加表，並透過該累加表上的各組距離算出總距離累加值來求出二次曲面上韋伯點的趨近解可能坐落區位。所需的總時間複雜度為$O(n^2)$，其中n為來源點的數目。

關鍵字：*三角剖分*、*韋伯點*、*基點*

一、 緒論

歐氏幾何上的韋伯問題是一探討區域設置的問題，一直以來都受到相當程度的討論，但是不能運用於高低起伏的地形，例如：若運用在太空計畫、月球探險之二次曲面地形上時，考量基地(源點)及補給站(韋伯點)間的總距離最短和各基地人口數量，此研究能達到使各基地及補給站間的總距離最小及降低運輸時間之目標。

目前已有學者應用 Dijkstra 及 Voronoi 的方法，提出 1-Center、1-Median 在二次曲面上韋伯趨近解的解答[1]。此次研究結合了在二次曲面上使用脊點(ridge point) [2]、Ahuja-Dijkstra 演算法[3]、二次曲面上的最短路徑演算法[4]及成本累加表[5]等方法來找到二次曲面上韋伯問題之趨近解，期望透過此方式能有效降低運算過程中所花費之時間，並使其在使用者接受之精確度內得到結果。本演算法所找出的韋伯點趨近值為區域最佳解 (Local Optimum)之近似數據。

二、 基本概念及背景

之前我們以網格圖、洪泛理論及距離累加觀念，發展出網格圖上韋伯點之趨近解，並可變換每個原點之權重，其時間複雜度為$O(lN)$，其中l為來源點的數目，N為網格圖中網格的數目[6]。

最近我們以洋蔥方法找出韋伯潛在區，而後再透過三角剖分演算法(Delaunay Triangulation)求得趨近解，得以在期望的時間內完成搜尋，總時間複雜度為$O(n^{1.5})$，其中n為來源點的數目[7]。

以上述想法及演算法來計算在二次曲面上尋找韋伯點的最佳趨近解。本研究提出以在三角網格中使用 Ahuja-Dijkstra 和脊點方式來產生一距離成本累加表，並透過該累加表上的各組距離算出總距離累加值來求出二次曲面上韋伯點的趨近解可能坐落區位，而在此研究中我們則選擇使用 Delaunay Triangulation [8]的演算法，於三角網格上透過 Ahuja-Dijkstra 演算法和曲面上的脊點(Ridge point)來找出通過源點的最短路徑並找到韋伯點的趨近解在透過二次曲面上的最短路徑演算法求出韋伯點及各源點間最短路徑。

二次曲面是一三元二次函數方程式在三維座標系(x, y, z)的圖形總稱，在 n 維的超曲面，其定義為二次方程式解的軌跡，在座標上，定義則為代數的方程式。二次曲面較歐氏幾何運用面廣，能運用在崎嶇不平的區域上，相較歐氏幾何只能用在二維平面上，其透過曲面模型能更有效的去使用在多方面的應用模擬上也能較符合實際環境。

脊線是由脊點所組成的曲線，因此脊點(Ridge Point)一定是為脊線上的某一點，如兩相鄰區域分別各有一點且兩點相連，形成一穿過脊線的線段，在垂直於此線段並通過脊線上的某點時，則此在脊線上的點則為脊點。兩相鄰曲面上的點之最短路徑只要有通過上述方式所找到的脊點皆會為兩點之最短路徑。

Dijkstra 演算法是以圖上某一點為起始點，計算從此點

出發並經由相連且未被選擇之節點裡，選擇離出發點距離最短之節點新增至路徑中，並藉由更新結點計算到達其他節點之累加距離值，直到所有節點都含括在內為止，並得出到達其他點之最短路徑，執行時需要考慮到 n 個節點，因此其時間複雜度為$O(n^2)$。

Ahuja-Dijkstra 演算法之特性與 Dijkstra 相似，但不同處在於 Ahuja-Dijkstra 能使 Dijkstra 的複雜度從$O(n^2)$降為$O(n)$，在執行速度上相對比原先快速許多。

三、 演算法及其圖解

以二次曲面上的 Ahuja-Dijkstra、脊點、二次曲面上的最短路徑演算法方法來解決尋找韋伯點問題，在此將以圖 1 中解釋。圖 1 (a)先將實際地形轉換為三角網格圖，圖 1 (b)則輸入源點且在沒有源點的三角形上設立重心點，圖 1 (c)透過 ridge point 演算法連接相鄰源點或重心點三角形構成一連結圖，圖 1 (d)利用 Ahuja-Dijkstra 來得出各點到其他點的最短距離值，圖 1(e)所有點計算圖 1 (d)所得之各組最短距離累加值，圖 1(f)顯示累加值最小距離的三角形為韋伯點可能坐落的三角形，接著圖 1 (g)對韋伯點潛在三角形做三角剖分並設置三個重心點，圖 1 (h)透過二次曲面上的最短路徑演算法計算圖 1 (g)圖中所設置的重心點到其他點的距離值，圖 1 (i)計算出子三角形上的三個重心點，並選擇四個累加距離值中數值最小的點，圖 1 (j)顯示距離累加值最小的三角形，圖 1 (k)則是顯示重複圖 1 (h) –圖 1 (j)步驟計算子三角內重心點的距離累加值，並從中取距離累加值最小點，直至符合使用者定義時停止，圖 1 (l)顯示最終符合使用者定義之誤差值的韋伯點趨近解。

3.1 演算法

Function 1:產生韋伯點區位連結圖

Step 1：輸入源點並於沒有源點的三角形內設置重心點。

Step 2：找出相鄰三角形內源點或重心點連結後所產生之稜點。

Step 3：計算所有相鄰三角形內重心點及源點間的距離並形成連結圖。

End

Function 2: 找到韋伯點可能座落區位

Step 1：透過 Ahuja-Dijkstra 演算法計算出連結圖中各點到其他點的最短距離值。

Step 2：所有點計算 Step 1 所得出的各組距離累加值，於連結圖中取其距離累加值最小的點，則此三角形為韋伯點潛在三角形。

End

Function 3：三角剖分

Step 1：將韋伯點潛在區位三角形做三角剖分。

Step 2：剖分完成在子三角型輸入重心點。

End

Function 4: 韋伯趨近解

Step 1：韋伯點潛在三角形呼叫二次曲面上的最短路徑演算法和脊點演算法求出 Function 3 所得子三角

型內重心點的距離累加值。

Step 2：並將原先潛在子三角形內四個重心點的距離累加值做比較，若$D_{i,j}^{W}(k) - D_{i,j}^{W}(k+1) \leq$ ε則停止，否則將$D_{i,j}^{W}(k+1)$所在的三角形設為新的潛在三角形，回到 Step 1。

End

Algorithm:二次曲面之韋伯點設置區位演算法()初始化 轉換地形為二次曲面圖。

Step 1：初始化二次曲面圖中各來源點及重心點。

Step 2：呼叫 Function 1 生成最短路徑連結圖。

Step 3：找出韋伯點可能座落範圍。

呼叫 Function 2 找到距離累加值最小之三角形

Step 4：韋伯點之趨近解。

呼叫 Function 4 找出趨近解。

END {二次曲面之韋伯點設置區位演算法()}

3.2 演算法圖解

本演算法個步驟地詳細圖解如圖一所示。

四、 演算法效能分析

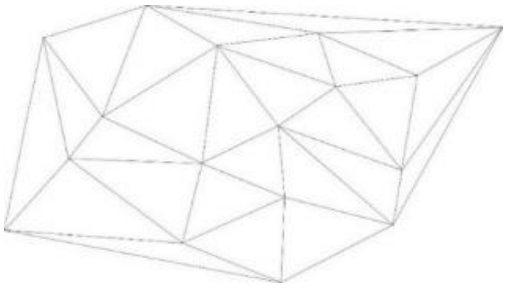
本文的總時間複雜度為$O(n^2)$。

研究透過 Ahja-Dijkstra 配合脊點來降低在二次曲面中找尋韋伯點之時間複雜度。此節主要以演算法所執行步驟之順序，詳加描述各步驟預期的時間複雜度，並統整各項演算法之時間複雜度，預期本演算法在進行二次曲面上韋伯點趨近解所需的總時間複雜度。

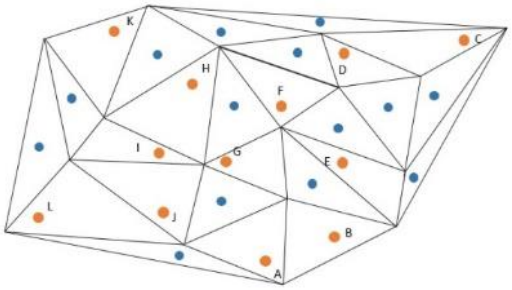
在步驟一和二中先把實際地形轉換為二次曲面，透過呼叫 Function 1 形成路徑圖時，須考慮到的對象為 n 個源點及 m 個重心點，因 n 與 m 皆為一常數，因此兩步驟時間複雜度為$O(n)$。

接續前兩步驟，在演算法中第三步驟的目標為找到韋伯點可能坐落的潛在三角形區位，呼叫 Function 2 透過時間複雜度為$O(n)$的 Ahuja-Dijkstra 得出所有點的距離值，之後累加每個點所得之各組距離值，得出各點的總距離累加值後，從中取總距離累加值最小者，此點所在之三角形即為韋伯點可能坐落三角形。找到可能座落之三角形後加入重心點，在使用時間複雜度為$O(n)$的二次曲面上的最短路徑演算法配合脊點演算法計算新的三個重心點的最短距離累加值並與一開始所找到的距離累加值最小點做比較，此步驟的時間複雜度為$O(n^2)$。

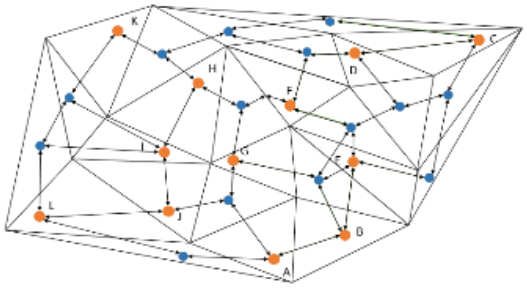
步驟四中呼叫 Function 3、Function 4 得出韋伯點趨近



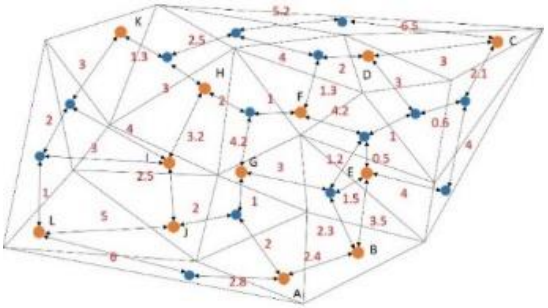
a. 將實際地形轉換為三角網格



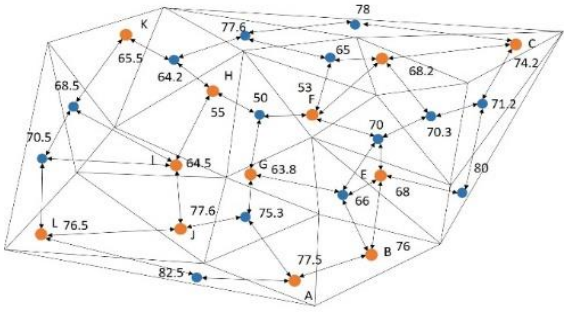
b 輸入源點及在沒有源點的三角形上設立重心點



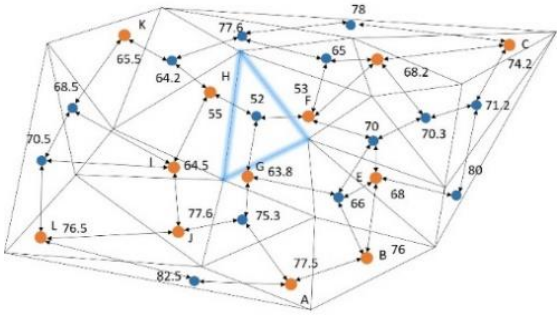
c. 透過 ridge point 產生一路徑連結圖並得到每個邊的距離



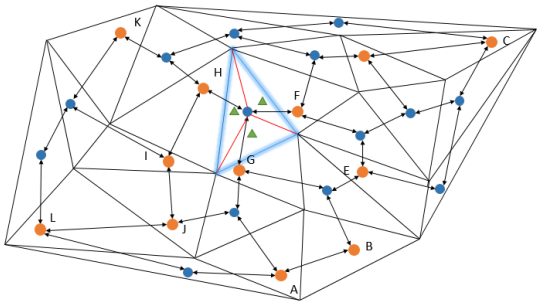
d. 透過 Ahuja-Dijkstra 得出各點到其他點之最短距離值



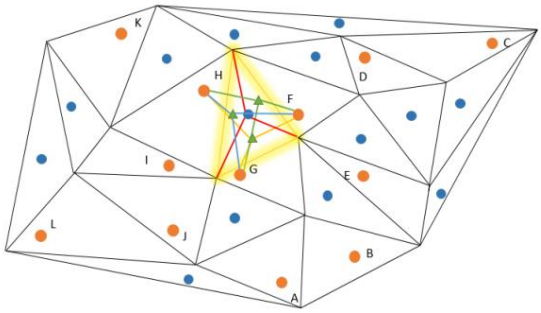
e 每個三角形的點計算(d)所得出各組最短距離之累加值



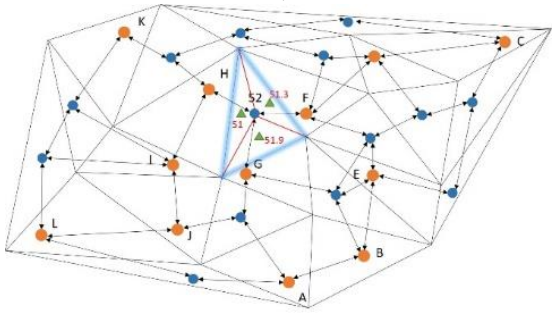
f 找出累加值最小距離者，即為韋伯點可能坐落的三角形



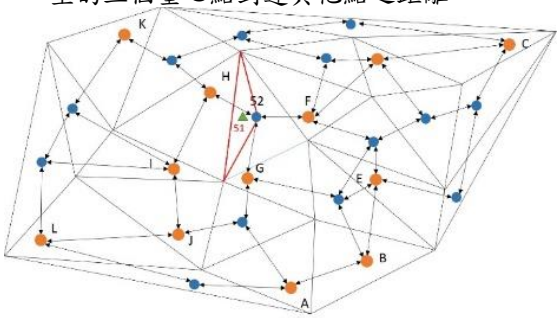
(g) 對此三角型做三角剖分找出三個新的重心點



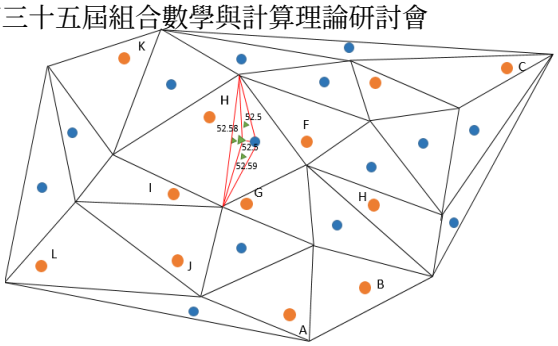
(h) 透過 Near- shortest path 和脊點演算法計算新產生的三個重心點到達其他點之距離



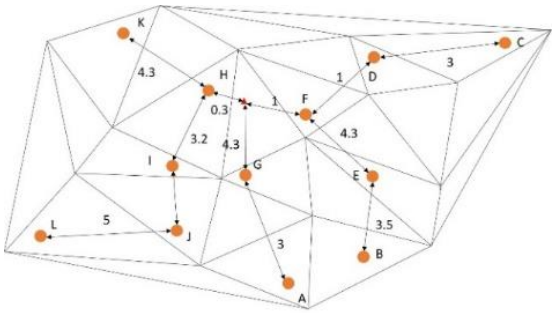
(i) 計算出各重心點的累加距離值



(j) 找到距離累加值為最小的三角形



(k) 重複三角剖分並重複(h)步驟計算子三角形內重心點累加距離值，並取其最小者。



(l) 重複上述步驟直到 k 和 k+1 次的韋伯距離誤差值小於使用者所設之誤差值 ε值，完成韋伯點趨近解。

圖 1 演算法細部圖解

解之近似值，包含對韋伯點可能座落區位之重心點和源點 k 次，k 的次數必依照使用者所設定之誤差值ε而定，而ε為常數，而三角形剖分次數 k 為常數，因此三角剖分之時間複雜度為$O(n)$，即可知此次步驟複雜度為$O(n)$。

將前述四項步驟的時間複雜度做統整後，可得知本研究所提出的從二次曲面上求出韋伯點趨近解，所需的總時間複雜度為$O(n)+O(n^2)+O(n)=O(n^2)$。

五、 結論

大部分韋伯問題研究都在歐氏幾何平面上求解，本文透過二次曲面來模擬地形能考慮到較多地形變因，較一般平面模擬更貼近實際應用。如在太空計畫、月球探索等或在已知地形卻沒有道路系統的二次曲面地形時，透過此演算法能使設置補給站到達各基地之總距離最短，也能大幅降低運輸時間。

參考文獻:

[1] M. Fort, and J. A. Sellarès, “Computing generalized higher-order Voronoi diagrams on triangulated surfaces,” *Applied Mathematics and Computation*, 215(1), pp. 235-250, 2009.

[2] J. R. Sack and J. Urrutia, “Handbook of Computational Geometry,” *Elsevier*, pp. 370, 1999.

[3] K. M. Ravindra, K. Ahuja, J. B. Orlin, and R. E. Tarjan, "Faster algorithms for the shortest path problem," *Journal of ACM*, vol. 37, pp. 213-223, 1990.

[4] Chi-Chia Sun, Gene Eu Jan, Shao-Wei Leu, Kai-Chien Yang and Yi-Chun Chen, “Near-Shortest Path-Planning on a Quadratic Surface with *O*(*n* log *n*) Time,” *IEEE Sensors Journal*, Vol. 15, No. 11, pp. 6079- 6080, Nov. 2015.

[5] Gene Eu Jan, Ki-Yin Chang, Su Gao and Ian Parberry, “A 4-Geometry Maze Router and Its Application on Multi-terminal Nets,” *ACM Transactions on Design Automation of Electronic Systems*, Vol. 10, No. 1, pp. 116-135, Jan. 2005.

[6] Ki-Yin Chang, Chien-Ming Su, Gene Eu Jan and P.-C. Chen, “An Efficient Method for Single Facility Location and Path Connecting Problems in Cell Map,” *International Journal of Geographical Information Science*, Volume 27, Issue 10, pp. 2060-2076, Oct. 2013.

[7] Gene Eu Jan, Ki-Yin Chang, Wei-Ju Chou, and Yang Chen Lin, “The Approximation of Weber Point,” *Proceedings of National Computer Symposium*, National Dong Hwa University, Taiwan, December 2017.

[8] L. Paul Chew, “Constrained Delaunay triangulations,” *Algorithmica*, 4.1-4: 97-108, 1989.

系統晶片封裝之重分佈線路繞線(線路重佈)

Redistribution Layer Routing on System in Package

詹景裕
國立臺南藝術大學
校長室
gejan@mail.ntpu.edu.tw

陳松懋
國立臺北大學
電機工程系
c9942024@gmail.com

雒超民
美國底特律大學
電機工程系
luoch@udmercy.edu

摘要

在IC Design的領域上，系統晶片封裝SiP (System in Package) 是由系統單晶片SoC (System on Chip) 衍生的技術，因SoC開發時間較長與開發難度越來越高，致使SiP的蓬勃發展，而SiP不管是在開發時間抑或是難度上都優於SoC，而無論是覆晶封裝、晶圓級封裝抑或是系統晶片封裝等，繞線都是不可或缺的一環，而在繞線的部分，又區分全域繞線(Global routing) 與細部繞線(Detailed routing)，因此繞線的策略就顯得尤為重要，而本文則是基於Two Terminal Nets Routing Algorithm改進提出一應用於SiP的重分佈層(RDL)的演算法，主要透過HGMR演算法快速計算線路屬性，利用線路的屬性與設立優先權數值來快速排序，以此為基準將每組線路繞線，並在遇到線路繞死問題時，透過優先權機制將其解決，完成繞線。本文提出之演算法可將每組線路連接，並縮短線路總長度。本演算法的時間複雜度為O(pN)，其中p為組對數，N為網格總數且為Higher Geometry Maze Router Algorithm每次計算的時間複雜度。

關鍵字：Printed circuit board, Redistribution Layer, Very Large Scale Integration, System in Package, Flip Chip.

一、 緒論

在超大型積體電路(VLSI)以及印刷電路板(Printed Circuit Board)的實體設計(Physical Design)過程中，共可以區分為五大主要步驟：分割(Partitioning)、平面規劃與擺置 (Floor Planning & Placement)、繞線(Routing)、壓縮(Compaction)以及最後的驗證(Extraction & Verification)。

而不管是在積體電路設計(Integrated circuit design)中，或是EDA P&R的應用上，繞線除了不可或缺也是尤為重要的一環，如圖 1。

首先，在設計多層的超大型積體電路以及印刷電路板的應用上，良好的繞線策略不但可以減少繞線所需的長度，更可以大幅減少層數的需求，以達到節省繞線與層數成本。繞線策略應用方面除了PCB與VLSI 的繞線問題[12][13]，也常應用於解決機器人避碰路徑搜尋問題[6][7]、電子海圖顯示資訊系統(Electronic Chart Display Information Systems, ECDIS)的路徑搜尋問題[1]、道路地圖的繞行問題[4]，電腦遊戲人工智慧[10]等。

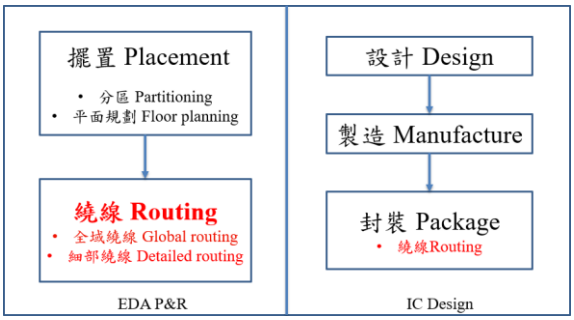
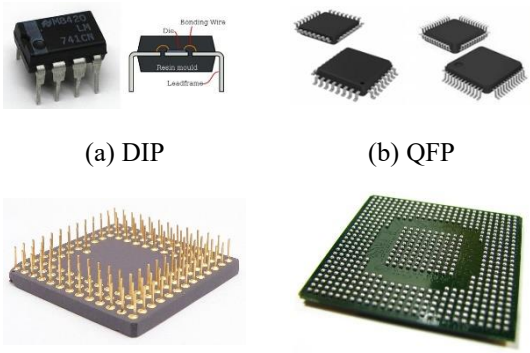


圖 1 EDA 與 IC 的共通性

而近年來，晶片已從早期的低密度、大體積、散熱性差，發展到如今的高密度、小體積、高可靠度與散熱性高。

從最早期的雙排封裝(Dual Inline Package, DIP)、四側引角封裝(Quad Flat Package, QFP)到後來的針格陣列封裝(Pin Grid Array, PGA)與衍伸出球格陣列(Ball Grid Array, BGA)，如圖 2 所示。

封裝技術也由耗時較長的打線封裝(Wire Bonding)技術，到後來發展出可快速完成的覆晶封裝(Flip Chip)技術，如圖 3，該技術除了可以滿足高密度封裝外，相較於打線封裝耗費的時間也更短。



(a) DIP (b) QFP

(c) PGA (d) BGA

圖 2 各類封裝晶片示意圖

除了覆晶封裝外，也發展出如晶圓級封裝(Wafer Level Package, WLP)、系統晶片封裝(System in Package, SiP)等技術；而 WLP 的特點則是在生產的晶圓上直接進行封裝測試，最後切割成單顆IC，相較於一般封裝流程節省了許多步驟，縮短製程時間，如圖 4 所示。

晶片系統(System on Chip, SoC)製程技術是是一種將電腦或其他電子系統整合到單一晶片的積體電路。其特性為低成本、低功耗及高效能。

而 SiP 則是基於 SoC 發展出的一種封裝技術，其特性是開發難度遠低於 SoC，且 SiP 具有整合彈性，可有效縮減電路載板面積。

而在晶片生產時，若需更改或優化線路時，勢必要重新開發，其成本將倍增，故而發展出線路重佈層(Redistribution Layer, RDL)技術，其方法是在原有的基板上，鋪上一層絕緣層，並在絕緣層上重拉一層金屬線(metal line)，最後鋪上一層 EP 封膠並開新的接腳孔(Pad Opening)，達到改變線路位置的目的，如圖 5，其特性為改變原有線路 I/O 設計，也可取代部分 IC 線路設計，加速開發時程，降低開發成本且增加電路附加價值。因此在繞線層(IC Bump 連接到 I/O Pad 間)的繞線策略就尤為重要，如圖 3 的 Substrate 與圖 5 (d)的紅色部分，而本文主軸則是提出一個可應用於 SiP 的 RDL 繞線演算法。

在第二節中介紹本文所使用的方法與其他演算法之相關文獻。於第三節中介紹演算法流程。而在第四節將分析演算法的時間複雜度。最後第五節則為本文結論。

二、 相關文獻

於先前已闡述了繞線對於該領域的重要性，在本節中將介紹相關的繞線演算法，在 2008 年由Fang等人提出應用於Flip Chips的RDL繞線演算法[2]，該演算法先透過網路流量公式(Network Flow Formulation)中最大流量最小成本公式(Minimum Cost Maximum Flow)計算全域繞線，再透過利用有效的配置繞線點(Passing Point Assignment)概念，並決定繞線順序(Net Ordering Determination)，最後利用迷宮繞線(Maze Routing)來完成線路繞線，該演算法可確實完成每組線路的連接，並有效縮短線路長度。

而Fang等人在2009年基於改進先前的演算法提出一應用於Flip Chip的RDL繞線演算法[3]，該演算法基於DT (Delaunay Triangulation) 與 Voronoi Diagram將電路剖分並建立網絡，在透過的網路流公式(Network Flow Formulation)來計算需要預先分配或自由分配的Bump Pad，然後完成繞線，最後再透過Network Flow Formulation計算並優化繞線地圖，該演算法可保證每組線路確實連接，且線路長度較先前提出的繞線演算法[2] 短，

在 2016 年，Lin 等人提出一應用於InFO WLCSP的RDL的繞線演算法[11]，首先透過為每一個晶片建立同心圓模組來處理Bump預先分配與自由分配問題並將線路分層，而後透過扇入扇出將線路整理避免線路卡死，最後透過網路流公式將線路優化並平均分層，完成繞線。該演算法可在保證線路確實連接的同時，處理多層的電路分配。

而本文演算法則是基於改進Two Terminal Nets Routing Algorithm [9]，核心為HGMR (Higher Geometry Maze Router Algorithm)演算法[5] ，透過該演算法快速計算每組組對屬性(長度、交叉數)，再利用基底排序法將每組組對快速排序，完成演算法，可在確實連接每一組對的同時，縮短線路總長度。

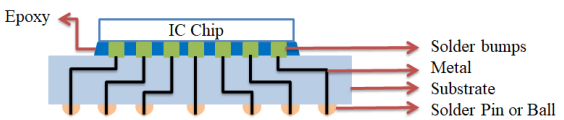


圖 3 覆晶封裝側視圖

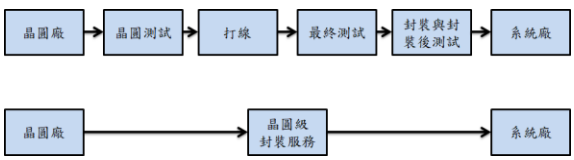


圖 4 封裝流程圖，上為傳統封裝，下為 WLP

(a) 基板 (b) 鋪上 PI 絕緣 (c) 鋪上金屬布線 (d) EP 封膠並開孔

圖 5 RDL 流程

三、 演算法與圖解

在本節中將介紹演算法的概述與定義使用的變數。

3.1 線路重佈

Function : 提高繞線優先權
當第k組組對無法完成繞線時，將其$pri(k)$的提升至$Int(k)$的組對之前。

演算法: 線路重佈

Step 1: 初始化
Step 2: 計算線路屬性
2.1 輸入組對與欲改變位置
將每一組組對k的起始點與終點等參數輸入。
2.2 計算長度
利用HGMR的最短路徑演算法，計算每一組組對並紀錄其節點與長度$Len(k)$
2.3 計算交叉數
計算線路與所有組對的交叉數並紀錄$Int(k)$。

表 1 變數表

變數	型態	說明
p	整數	欲繞線的組對總數。
k	整數	索引值，代表第k組組對，$1 \le k \le p$
$v_{i,j}$	二為陣列	vertices，儲存$\sqrt{n} \times \sqrt{n}$的電路布局圖的節點狀態，介於$\{0, 1, 2, \ldots, p\}$間。其中0時表示該節點為自由節點，不為0時表示為第k組組對。其中(i, j)為二維矩陣之索引值，$0 \le i \le N-1$，$0 \le j \le N-1$。
AP	二為陣列	Array of Pairs，輔助計算用的資料結構，為$p \times 5$的矩陣，儲存每組組對的k，Pri，Int，Len與$Rank$值。
$Pri(k)$	整數	Priority，優先權，用來決定繞線順序的參數之一，其值越大則優先權越低，$1 \le k \le p$。
$Int(k)$	整數	Intersection，第k組組對與其他線路相互交叉的數目，$1 \le k \le p$。
$Len(k)$	浮點數	Length，第k組線路的路徑長度，$1 \le k \le p$。
$Rank(k)$	整數	Rank，第k組線路繞線次序，$1 \le k \le p$。

2.4 設定優先值

加入優先值$Pri(k)$參數，用於提高組對的繞線順序。如VDD或GND等，需要優先連結的線路。

Step 3: 將所有組對繞徑

3.1 排序

利用基底排序法將每一組組對依優先值、交叉數與長度排序。

3.2 連結線路

利用HGMR依排序將連結每一組組對並避開障礙物。

3.3 提高線路優先權

連接組對時，若有組對繞死，則Call Function計算繞死的組對，並回到Step 3.2 重新計算。

END {線路重佈}

3.2 演算法圖解

本演算法步驟如圖6，在圖6 (a)為線路初始化；利用HGMR計算每一組對，如圖6 (b)；透過基底排序法將組對依屬性排序，如圖6 (c-d)；利用HGMR依排序重新繞線，組對繞死，如圖6 (e)；提高繞死的組對順序，如圖6 (f-g)；最後圖6 (h)為完成演算法之結果。

四、 效能分析

關於本文之時間複雜度，可分為兩個部分來說明，第一個部分為Step 2中，計算每一組對的長度、交叉數與設定優先值，共有p組組對，透過HGMR演算法來計算，故在此部分時間複雜度需要$O(pN)$，其中p為組對數，N為HGMR每次計算的時間複雜度。

在第二個部分中，step 3.1 利用基底排序法來確立繞線順序，需要花費$O(p \log p)$；step 3.2中，一樣透過HGMR來連線，其耗費時間為$O(N)$，需計算p組線路，故此步驟耗費時間同為$O(pN)$；最後，在step 3.3中，提高繞線順序，因有p組線路，固時間複雜度為$O(p)$。總結所有步驟，本演算法的時間複雜度為$O(pN)$。

五、 結論

本文基於改進Two Terminal Nets Routing Algorithm提出一個應用於SiP的RDL繞線方法，與相關文獻中提到的演算法的方法不同，是利用網格化與HGMR來連接組對，可確實連接每組組對，並縮短線路總長度。且演算法時間複雜度為$O(pN)$，可快速計算並解決繞線問題。

在將來的研究，有幾個方向可加強或改進。在演算法繞線的部分，可以增加一些新的機制，如限制每組線路的平均長度 (避免訊號衰減)，符合實際情況。或尋求新的應用，如3D立體的RDL等。

參考文獻

[1] K. Y. Chang, G. E. Jan and I. Parberry, “A Method for Searching Optimal Routes with Collision Avoidance on Raster Charts,” *Journal of Navigation*, Vol. 56, No. 3, pp. 371~384, 2003.

[2] J. W. Fang and Y. W. Chang, “Area I/O flip-chip routing algorithm for chip-package co-design,” *Proc. of ICCAD*, pp. 518–522, 2008.

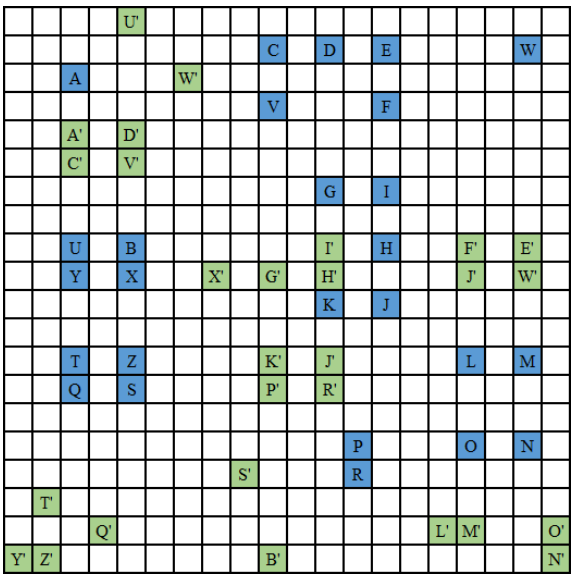
[3] J. W. Fang, Martin D. F. Wong, and Y. W. Chang, “Flip-Chip Routing with Unified Area-I/O Pad Assignments for Package-Board Co-Design,” *Proc. of the 46th Annual Design Automation Conference,* July, 2009.

[4] J. Fawcett and P. Robinson, “Adaptive Routing for Road Traffic,” *IEEE Computer Graph*, vol. 20, no. 3, pp. 46-53, June 2000.

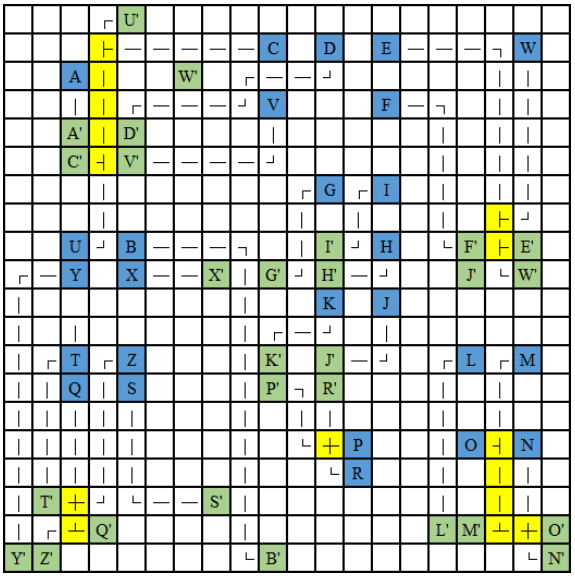
[5] G. E. Jan, K. Y. Chang, S. Gao, and I. Parberry, "A 4-Geometry Maze Router and Its Application on Multi-terminal Nets," *ACM Transactions on Design Automation of Electronic Systems*, vol. 10, pp. 116-135, 2005.

[6] G. E. Jan, K. Y. Chang, and I. Parberry, “Optimal Path Planning for Mobile Robot Navigation,” *IEEE/ASME Transactions on Mechatronics*, Vol. 14, No. 9, pp. 925- 936, Aug. 2008.

[7] G. E. Jan, C. C. Sun, W. C. Tsai and T. H. Lin, “An O(n log n) shortest path algorithm based on Delaunay triangulation,”



(a) 初始化，藍色節點為組對起始點(Bump)，綠色為終點(I/O Pad)



(b)利用HGMR計算組對屬性，虛線部分為組對之路徑，黃色部分為組對路徑的重疊處

IEEE/ASME Transactions on Mechatronics, Vol. 19, No. 2, pp. 660- 666, April 2014.

[8] D. C. Jhou, "Realization of Multiple Steiner Trees Routing," *National Taipei University Master Thesis*, 2011.

[9] F. C. Kuo, "Two-Terminal Nets Routing Algorithms on Printed Circuit Boards," *National Taipei University Master Thesis*, 2009.

[10] M. V. Lent, "Game Smarts," *IEEE Computer Society*, vol. 40, Issue 4, pp. 99-101, Apr. 2007.

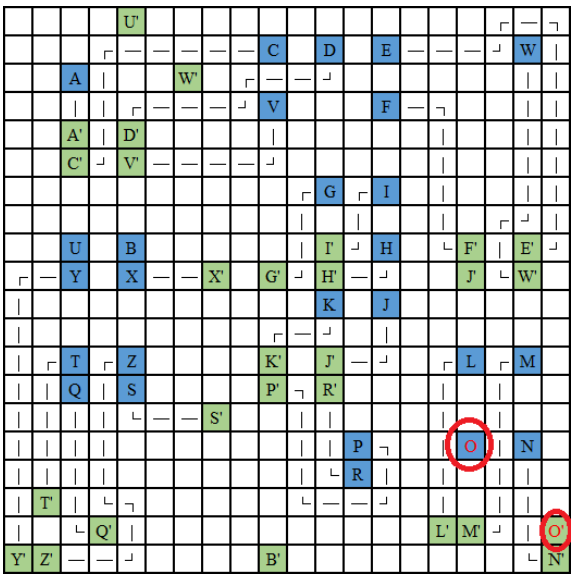
[11] B. Q. Lin, T. C. Lin, and Y. W. Chang. "Redistribution Layer Routing for Integrated Fan-out Wafer-level Chip-scale Packages." *Proceedings of the 35th International Conference on Computer-Aided Design*, Nov. 2016.

Rank	pair	Pri(k) (priority)	Int(k) (intersection)	Len(k) (length)
-	A	1000	-	1
-	B	1000	-	15
-	C	1000	-	10
-	D	1000	-	9
-	E	1000	W(1)	11
-	F	1000	-	7
-	G	1000	-	4
-	H	1000	-	2
-	I	1000	-	3
-	J	1000	-	3
-	K	1000	-	3
-	L	1000	-	6
-	M	1000	O(1)	7
-	N	1000	O(1)	4
-	O	1000	M, N(1)	5
-	P	1000	R(1)	4
-	Q	1000	Z(1)	5
-	R	1000	P(1)	3
-	S	1000	-	3
-	T	1000	-	5
-	U	1000	C(1)	9
-	V	1000	-	6
-	W	1000	E(1)	9
-	X	1000	-	2
-	Y	1000	-	11
-	Z	1000	Q(1)	9

(b) 排序前的組對屬性

Rank	pair	Pri(k) (priority)	Int(k) (intersection)	Len(k) (length)
1	A	1000	-	1
2	H	1000	-	2
3	X	1000	-	2
4	I	1000	-	3
5	J	1000	-	3
6	K	1000	-	3
7	S	1000	-	3
8	G	1000	-	4
9	T	1000	-	5
10	L	1000	-	6
11	V	1000	-	6
12	F	1000	-	7
13	D	1000	-	9
14	C	1000	-	10
15	Y	1000	-	11
16	B	1000	-	15
17	R	1000	P(1)	3
18	N	1000	O(1)	4
19	P	1000	R(1)	4
20	Q	1000	Z(1)	5
21	M	1000	O(1)	7
22	U	1000	C(1)	9
23	W	1000	E(1)	9
24	Z	1000	Q(1)	9
25	E	1000	W(1)	11
26	O	1000	M, N(2)	5

(d)利用基底排序法計算後的組對排序



(e)依排序繞線時，遭遇組對繞死情況，如紅圈處

Rank	pair	Pri(k) (priority)	Int(k) (intersection)	Len(k) (length)
1	A	1000	-	1
2	H	1000	-	2
3	X	1000	-	2
4	I	1000	-	3
5	J	1000	-	3
6	K	1000	-	3
7	S	1000	-	3
8	G	1000	-	4
9	T	1000	-	5
10	L	1000	-	6
11	V	1000	-	6
12	F	1000	-	7
13	D	1000	-	9
14	C	1000	-	10
15	Y	1000	-	11
16	B	1000	-	15
17	R	1000	P(1)	3
18	N	1000	O(1)	4
19	P	1000	R(1)	4
20	Q	1000	Z(1)	5
21	M	1000	O(1)	7
22	U	1000	C(1)	9
23	W	1000	E(1)	9
24	Z	1000	Q(1)	9
25	E	1000	W(1)	11
26	O	1000	M, N(2)	5

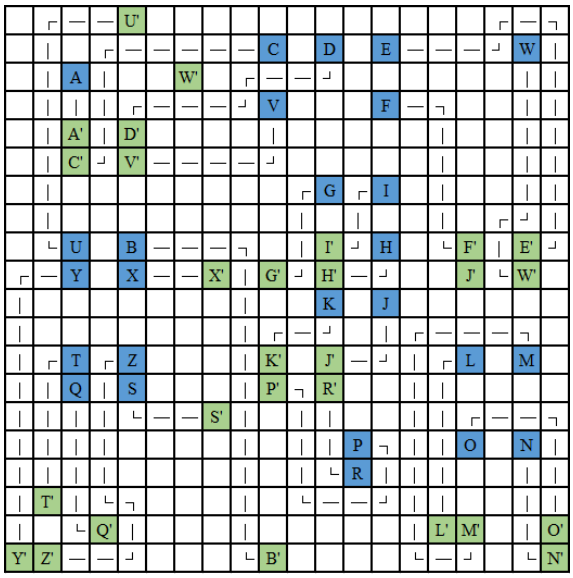
(f) 繞死的組對，如紅色處

[12] Y. L. Lin, Y. C. Hsu and F. S. Tsai, “Hybrid Routing,” *IEEE Trans. on CAD*, pp. 151-157, Feb. 1990.

[13] K. Mikami and K. Tabuchi, “A Computer Program for Optimal Routing of Printed Circuit Connections,” *IFIPS Proc.*, vol. H-47, pp. 1475-1478, 1968.

Rank	pair	Pri(k) (priority)	Int(k) (intersection)	Len(k) (length)
1	A	999	-	1
2	H	999	-	2
3	X	999	-	2
4	I	999	-	3
5	J	999	-	3
6	K	999	-	3
7	S	999	-	3
8	G	999	-	4
9	T	999	-	5
10	L	999	-	6
11	V	999	-	6
12	F	999	-	7
13	D	999	-	9
14	C	999	-	10
15	Y	999	-	11
16	B	999	-	15
17	R	999	P(1)	3
18	O	999	M, N(2)	5
19	N	1000	O(1)	4
20	P	1000	R(1)	4
21	Q	1000	Z(1)	5
22	M	1000	O(1)	7
23	U	1000	C(1)	9
24	W	1000	E(1)	9
25	Z	1000	Q(1)	9
26	E	1000	W(1)	11

(g) 提高繞死組對之優先權並重新繞線



(h) 完成繞線

圖6 線路重佈演算法示意圖

印刷電路板之多組對節點連結演算法及層數最佳化

詹景裕
國立臺南藝術大學
校長室
gejan@mail.ntpu.edu.tw

陳松懋
國立臺北大學
電機工程系
c9942024@gmail.com

黃彥文
國立海洋大學
電機工程系
a5130470@hotmail.com

雒超民
美國底特律大學
電機工程系
luoch@udmercy.edu

摘要

之前我們提出一個快速的單層多對節點連結演算法，本文擴充此文由單層成為多層多組對節點演算法，其時間複雜度為$O(p^2N)$，其中p，N各為在平面自由空間中，組對以及網格的總數。本演算法主要方法使用Higher Geometry Maze Router快速將各對線路連結，藉由線路連結之交錯點數及長度排出優先序，經過初步分層與優化總層數，來確定最佳層數後優化總線長。

關鍵字：多層多對節點連結、印刷電路板繞線、Higher Geometry Maze Router

一、 緒論

印刷電路版(Printed Circuit Board, PCB)技術在現代人的生活中，佔有著不可缺少的地位，不論是先進的 3C 產品或是一般的生活家電，內部的各個電子元件都必定有著 PCB 的連接。依照 PCB 的製作方式，主要分為三大類：單面板、雙面板、多層板。其製成成本的高低，便是以 PCB 的層數和使用的金屬線為主要的考量。如何將總層數降低，以及將所使用的金屬線總長度減少一直是各個研究想要追求的目標 [6]。在設計多層的超大型積體電路以及印刷電路板的應用上，良好的繞線策略不但可以減少繞線所需的長度，更可以大幅減少層數的需求，以達到節省繞線與層數成本。

現今常見的印刷電路板設計軟體如 Protel [8]，Dip Trace [3]。以上提出的設計軟體均為半自動化繞徑，因繞線策略不夠優良，造成所需層數過多與線路過長的問題。此外，多層板的繞線多以半人工的方式，利用軟體助配合人工拉線方式輔以來完成。但其使用上，在配置之前必需先決定好所要的PCB的層數才能進行繞線，但是層數多寡的決定，往往需要長期的人力培訓及豐富的實作經驗才能做出較適當的決策。

關於在印刷電路板上多對節點連結問題有許多演算法已被提出 [1]，[4]，[7]，本文在繞線時以電腦計算方式，迅速且自動化的決定 PCB 所需層數及最佳化層數，並在各層之間取得金屬線總長度較短且平衡的演算法。

二、 自動繞線優勢

本節將人工繞線及自動繞線之優缺點依人工繞線之人事成本、繞線時的出錯機率、光罩成本、繞線層數優化、線路複雜度極大情況下做比較如表1所示。

三、 演算法與圖解

本文演算法中所使用的變數符號之定義，列於表2方便索引及參考。

本演算法擴充並改進過去我們提出的單層多對節點連結演算法[9]，首先透過Higher Geometry Maze Router

表1 人工繞線與自動繞線之比較表

繞線方式 / 比較	人工繞線	自動繞線器
人工繞線之人事成本	高	極低
繞線時的出錯機率	有	無
光罩成本	高	低
繞線層數優化	無	有
線路複雜度極大	無法以人工解決	必須仰賴自動繞線器
※目前日月光聘用50人來處理SiG繞線作業		

表2 變數表

變數	說明
p	在平面自由空間中，組對的總數。
k	索引值，代表第k組組對，$1 \le k < p$
$v_{i,j}$	vertices，儲存m × n 電路佈局圖內節 點目前的狀態，其值介於$\{0, 1, 2, \ldots, p\}$之間。其中值為 0 時表示該節點為 自由節點，其中值不為 0 時表示該節點為欲連結的 k 線路對其中之一點。 其中(i, j)為二維陣列之索引值，$0 \le i \le m$，$0 \le j \le m$
ASP	Array of Sorting Pairs，輔助計算用的 資料結構，為一 p×5 的矩陣，內容儲存各組對的 k、*Pri*、*Int*、*Len* 以 及 *Rank* 值。
LP	Lowest Priority，定義最低的優先權 值，設為一個最大的正整數值
Pri(k)	Priority，優先權，在這邊指的是線路作傳繞演算法的順序，其值越大者優先權越低，$1 \le k \le p$。
Int(k)	Intersection，線路相互交錯數，第 k 組線路與其他線路互相交錯的數目，$1 \le k \le p$。
Len(k)	Length，原始長度，第 k 組組對線路的路徑長度，$1 \le k \le p$。
Rank(k)	Rank，線路連結順序，$1 \le k \le p$。

(HGMR) [5]演算法求得其每一組組對線路的路徑長度、路徑位置，以及各組線路之間交錯的節點數目，將每條線路加入優先值，然後進行初步的分層與層數的優化，最後計算出最佳化層數和總線長。

3.1多層多對節點連結演算法

演算法:多層多對節點連結

BEGIN

Step 1 將所有組對相連並排序

Step 1.1 將每一組對以HGMR演算法兩兩相連，取得每一組對的最短路徑並將路徑節點紀錄在$v_{i,j}$，忽略路徑間的交叉，並將每一組對的路徑長度*Len*(*k*)與交叉數*Int*(*k*)記錄到*ASP*中。

Step 1.2 利用Radix sort [2]將所有組隊依交叉數與長度來排序，並加入優先值*Pri*(*k*) (初始均為1000)初始化*ASP*。

Step 2 將所有組對初步分層

Step 2.1 將交岔的組對分層，由交叉數最多的組對開始依序分層，每當新插入的組對與該層中既有組有路徑交岔，則向下新增一層並將組對插入該層。

Step 2.2 將沒有交岔的組對插入第一層中

Step 3 層數最佳化

Step 3.1 Call Function 計算總層數內的組對

Step 3.2 總層數<2則演算法結束；若總層數≥2 且下半層數(lower Layers) 內剩餘的組對(pairs)為零，則回到Step 3.1再次計算。

Step 3.3 若總層數≥2且下半層數(lower Layers) 內剩餘的組對>0，則Call Function計算下半層組對，重複此步驟直到總層數≤2為止。

END

Function：Optimization of number of the layers

BEGIN

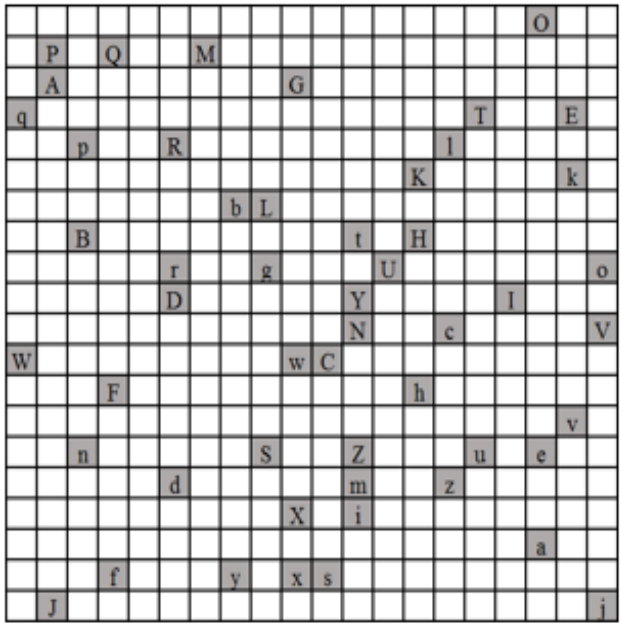
Step 1： 將總層數(number of Layers)分割為上下兩等份，若總層數(number of Layers)= 1時，Function結束。

Step 2 ： 將下半部(lower Layers)內的每一組組對(pairs)插入上半部(upper Layers)的每一層(Layer) ，計算並挑選增加長度最短的層(Layer)插入，直到下半層數內的組對無法再插入上半層或下半層中的組對均插入上半層為止。

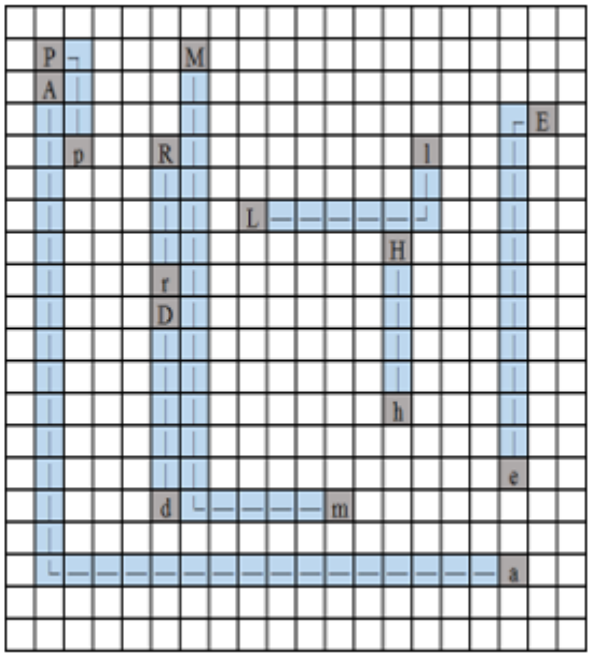
END

3.2 演算法範例

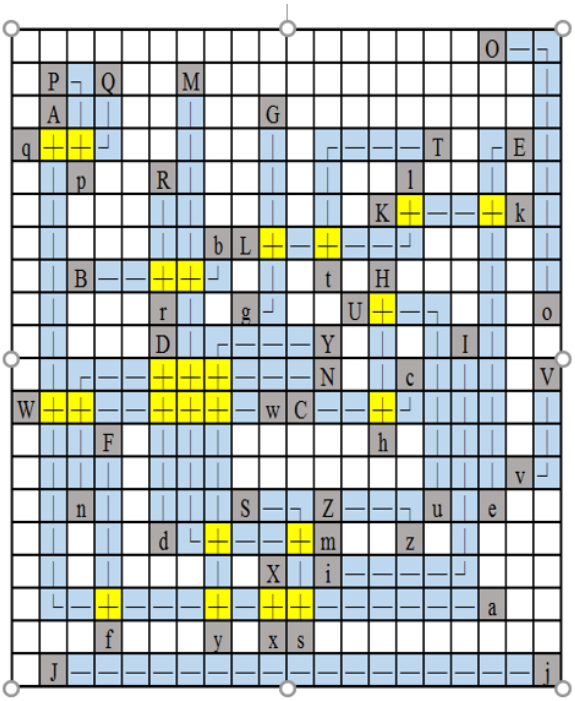
將所有組對相連並排序如圖1.所示，將參數記錄到*ASP*中如表3(a)所示，利用Radix sort將所有組隊依交叉數與長度來排序如表3(b)所示，將所有組對初步分層如圖2所示，最後經由最佳化層數得到最終結果如圖3所示。



(a) 初始狀態



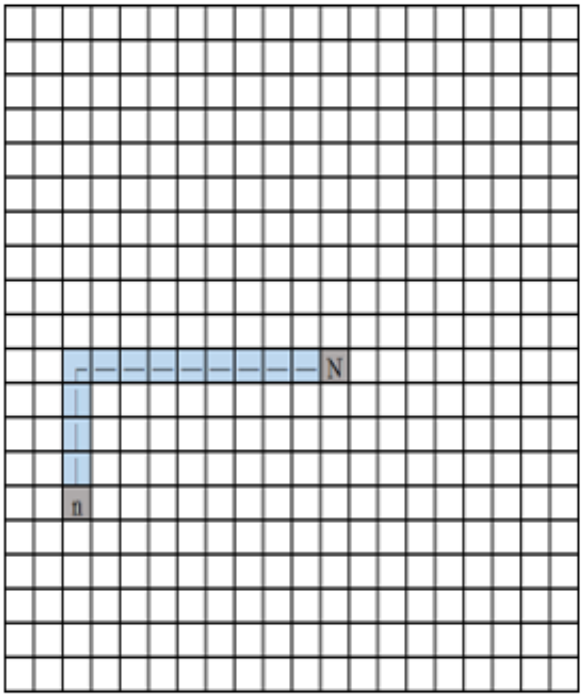
(a) 第一層



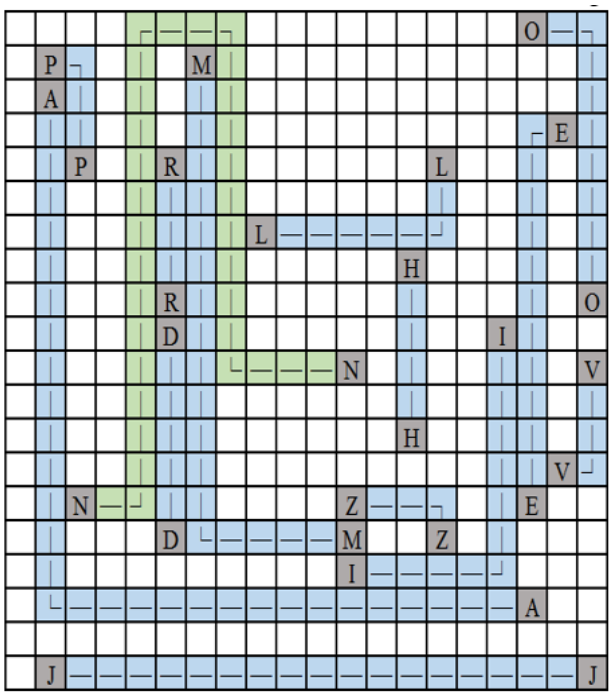
(b) 組對相連

(b) 第二層

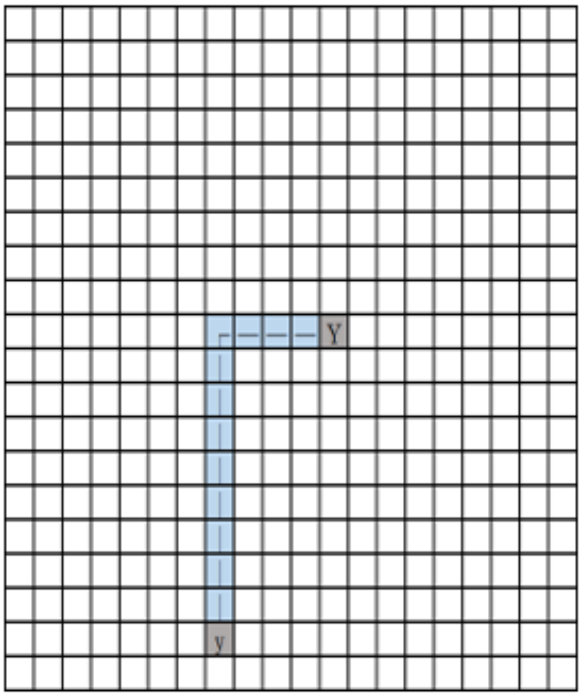
圖1 連接所有組對



(c) 第三層

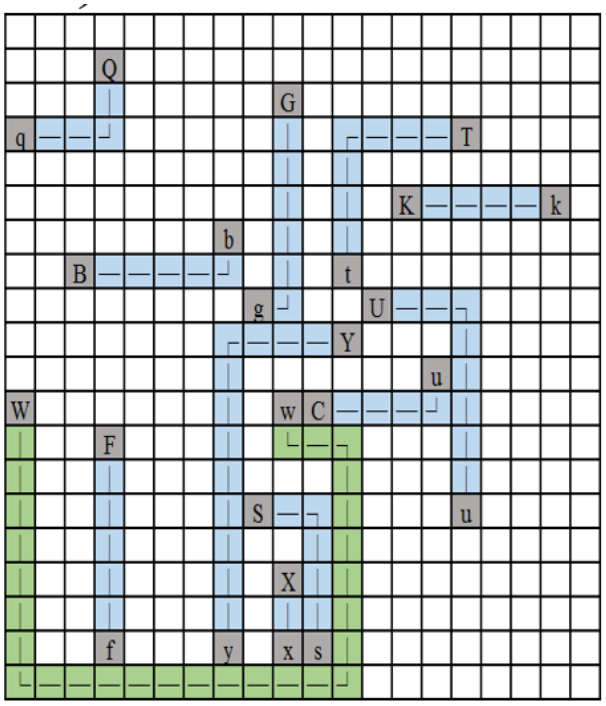


(a) 優化後第一層



(d) 第四層

圖2 初步分層為四層



(b) 優化後第二層

圖3 最佳化分層為兩層

表3 ASP數值

(a) *ASP*初始化後的數值

Rank	pair	Pri(k) (priority)	Int(k) (intersection)	Len(k) (length)
-	A	1000	F, Q, S, W, X, Y(6)	31
-	B	1000	M, R(2)	6
-	C	1000	H(1)	5
-	D	1000	N, W(2)	6
-	E	1000	K(1)	12
-	F	1000	A(1)	6
-	G	1000	L(1)	7
-	H	1000	C, U(2)	5
-	I	1000	-	12
-	J	1000	-	18
-	K	1000	E, L(2)	5
-	L	1000	G, K, T(3)	8
-	M	1000	B, N, S, W, Y(5)	19
-	N	1000	D, G, M, W(4)	13
-	O	1000	-	10
-	P	1000	Q(1)	4
-	Q	1000	P(1)	5
-	R	1000	B(1)	4
-	S	1000	A, M(2)	6
-	T	1000	L(1)	8
-	U	1000	H(1)	9
-	V	1000	-	4
-	W	1000	A, D, M, N, Y(5)	9
-	X	1000	A(1)	2
-	Y	1000	A, M, N, W(4)	13
-	Z	1000	-	4

(b) *ASP*經過排序的數值

Rank	pair	Pri(k) (priority)	Int(k) (intersection)	Len(k) (length)
1	A	1000	F, Q, S, W, X, Y(6)	31
2	W	1000	A, D, M, N, Y(5)	9
3	M	1000	B, N, S, W, Y(5)	19
4	N	1000	D, G, M, W(4)	13
5	Y	1000	A, M, N, W(4)	13
6	L	1000	G, K, T(3)	8
7	S	1000	A, M(2)	6
8	D	1000	N, W(2)	6
9	B	1000	M, R(2)	6
10	K	1000	E, L(2)	5
11	H	1000	C, U(2)	5
12	X	1000	A(1)	2
13	P	1000	Q(1)	4
14	R	1000	B(1)	4
15	C	1000	H(1)	5
16	Q	1000	P(1)	5
17	F	1000	A(1)	6
18	G	1000	L(1)	7
19	T	1000	L(1)	8
20	U	1000	H(1)	9
21	E	1000	K(1)	12
22	V	1000	-	4
23	Z	1000	-	4
24	O	1000	-	10
25	I	1000	-	12
26	J	1000	-	18

四、 效能分析

本文演算法之時間複雜度，共分為三個部分，第一個部分為Step 1.1中，計算每組線路的長度、交叉數與設定優先值，共有k組線路，此部分透過HGMR演算法來計算故其時間複雜度需為O(pN)，p為線路組數，N為網格的總數且為HGMR演算法時間複雜度。在第二個部分中，Step 1.2中使用Radix sort來做優先級的排序，需要花費$O(p \log p)$；Step 2 初步分層中，一樣透過HGMR來連線，其耗費時間為$O(N)$，需計算p組線路，故此步驟耗費時間同為$O(pN)$，step 3所需時間複雜度為$O(p^2N)$，綜觀Step 1-3時間複雜度，得知本演算法時間複雜度的最大需求在Step 3，故本演算法的時間複雜度$O(p^2N)$。

五、 結論

在多層多對節點問題中，本文提出一個試誤型的演算法，藉由線路的屬性:例如連結組對線路的長度、組對與其他線路之間的交錯數來作為排序的依據，再以HGMR演算法作為繞線機制，並在最後試著降低印刷電路板多餘的層數與總線長。而未來的研究方向，可以尋求一個更快的演算法當作繞線機制或是更有效率的排序機制，來降低總金屬線的長度與總層數。

參考文獻

[1] M. Burstein and R. Pelavin, “Hierarchical channel router,” *Proc., of 20th ACM/IEEE Design Automation Conf.*, pp. 519-597, 1983

[2] T. H. Cormen, and C. E. Leiserson, *Introduction to Algorithms*, The MIT Press 1990.

[3] Dip Trace website, https://diptrace.com/

[4] D. Dolev, K. Karplus, A. Siege, A. Strong, and J. D. Ullman, “Optimal wiring between rectangle,” *Proc. of 13th Annual ACM Symposium*, pp. 312-317, May 1987.

[5] G. E. Jan, K.-Y. Chang, S. Gao, and I. Parberry, “A 4-Geometry Maze Router and Its Application on Multi-

terminal Nets," *ACM Transactions on Design Automation of Electronic Systems*, vol. 10, pp. 116-135, 2005.

[6] C. E. Leiserson and F. M. Maley, "Algorithms for routing and testing routability of planar VLSI layouts," 1985, pp. 69-78F.

[7] C. E. Leiserson and R. Y. Pinter, "Optimal placement for river routing," *SIAM Journal of Computing*, 12, No. 3:447- 462, August 1983.

[8] Protel website, https://www.altium.com/solution/protel-pcb

[9] 詹景裕, 劉萬榮, 郭芳誠, 陳永源, "單層多對節點連結演算法," 2007 *年全國計算機會議* ,亞洲大學, 2007

樂器聲音之音高判別以及音準分析

羅峻旗, 胡育誠
靜宜大學資訊工程學系, 靜宜大學資訊管理學系
{cclo,ychu}@pu.edu.tw

摘要

本論文的內容主要是提出一個樂器聲音之音高判別以及音準分析之方法，並實作於電腦進行測試。系統製作的方法是將樂器所發出的聲音，經過傅立葉頻譜分析之後，得到各組成頻率的震幅所構成的向量，經由找出這個向量的各個峰值的所在位置，我們可以判別樂器所發出的聲音的位階。同時，藉由分析這些峰值所對應的頻率，我們也可以進一步評判樂器聲音音高之準確度。本論文以薩克斯風吹奏出的聲音爲測試對象，藉由提出的方法，判別此聲音的音高，同時也可分析這個聲音中的基頻以及泛音的準確度。

關鍵詞： 音高判別，傅利葉分析，音準分析，薩克斯風。

1 說明

語音辨識在現代已經是一個非常成熟而且應用非常廣泛的技術。只要透過 Apple 或者是 Google 的語音辨識系統，我們的各項裝置都可以輕易的判別出我們的語音所傳達的內容。然而聲音辨識的應用應該不只侷限於語音辨識。如果可以透過簡易的設備辨識生活周遭的各種聲音，這也可能會爲我們帶來許多便利的應用。 例如如果我們透過設備可以辨認出廚房中開水燒開時笛音壺鳴叫的聲音，並且即時的通知不在現場的我們， 這將可能減少一些意外災害的發生。又例如如果我們可以透過設備辨識出來嬰幼兒哭泣的聲音或者咳嗽的聲音並即時通報給父母和照護者，如此可以讓嬰幼兒照護的工作更加的周全，減少意外事故的發生。

以上所述這些聲音類別辨識能力對於我們的日常生活而言，都可以帶來非常大的幫助和益處。然而 Apple 或者 Google 的語音辨識系統卻不提供語音以外的聲音辨識功能。所以在我們過去的研究當中，我們便自行研究開發出一個嬰幼兒照護系統 [1]，透過辨識嬰幼兒的聲音，我們的系統可以將判別出來的聲音種類(哭聲、咳嗽聲、笑聲)傳達給父母親或者照顧者知道，也提供一些介面讓位在遠端的父母親或照護者與嬰幼兒做各式的互動（傳送父母語音、播放嬰兒歌、觀看嬰幼兒視頻）。

在目前各界積極發展的物聯網技術與應用中，聲音事件辨識也是可以帶來多樣新型的應用。例如當廚房中開水燒開的當時，人們可能因爲疏忽而不在現場，如果我們有個聲音識別的系統能夠辨識出當時的狀況，並透過網路將狀況傳達給當事人，這也可以幫助我們防止許多意外災害的發生。類似的應用還會有許多種，例如半夜小偷開門進入屋內的聲音，居家老人跌倒或咳嗽的聲音，屋外下雨的聲音等等。

在最近的科技部計劃中，我們也設計並開發出這樣的一個物聯網聲音事件辨識系統，來協助達成以上的目標 [2]。儘管如此，有關聲音特徵辨識的相關技術與應用，應該還有許多值得研究與開發可能空間。

本論文的目的，是將聲音辨識的能力擴展到樂器聲音的音高辨識。如果我們可以藉著聲音辨識系統來判斷樂器所發出的聲音音高，我們便可以即時了解樂器演奏的內容，甚至可以進而將這些內容轉換成樂譜，另外我們也可以依此判別樂器音高的準確性。而本論文將以薩克斯風這個樂器作爲例子，發展一個音高辨識以及聲音音高準確度判別的方法。台中市后里區裡區是一個製作薩克斯風的重鎮，其製作薩克斯風也已經有長久的時間，然而面對國際的競爭，后里薩克斯風的製作技術仍然必須持續的改良，在改良的過程以及改良後的產品，也需要一個科學化以及客觀的測量技術與系統來判斷其樂器聲音的精準度，才能得到國際音樂人員的認可。所以本論文的另一個目的就是在辨別薩克斯風的音高的同時也提供一個分析其聲音精準度的方法，期望其對國內樂器的製作技術的提升與發展有所助益。

2 相關文獻

讓電腦對音高有辨識的能力是長久以來大家有興趣的領域。也有許多研究提出他的演算法以求達到這樣的目標。一般而言，我們對於這類演算法的要求是要能夠即時的將樂器聲音的音高偵測出來。根據 [6] 的描述，聲音高度判別的演算法可以分爲兩類，一種是以時域（time domain）

的角度來判斷聲音的音高，另一種則是以頻域（frequency domain）的角度來分析。

從時域的角度來分析的第一種演算法是過 0 偵測演算法，這個方法的原理就是計算聲音訊號通過 0 的次數，這個次數應該至少會是基頻的兩倍。這個方法的缺點是會受到雜訊以及泛音的干擾。後來就有 Tadokoro [7] 等人對這類演算法提出改進措施，方法就是對聲音訊採取調適性濾波。另一種偵測的方式就是用自相關的技巧 [4]。這個方法的原理是當我們把一個訊號挪移一個週期後其與原來訊號的相關性會最大，所以我們就可以用這個道理來判斷一個聲音訊號的週期。第三個方法爲平均震幅差距函數的運用，這個方法與自我相關的方法類似，只是在運算上他不是用乘法而是用加法。他的道理是當我們把一個訊號挪移一個週期之後再跟原來的訊號相減所得到的平均值會最小，這個特性也提供給我們一個判斷訊號週期或頻率的方法。 Cuadra [5] 等人更進一步把自相關以及平均震幅差距函數的作法結合起來。

從頻率領域這個角度出發的演算法也有許多。第一個方法就是透過將各個諧波乘上各自對應的係數，就可以疊合成一個只包含基頻一個波峰的頻譜。而最通用的方法就是將聲音透過離散傅立葉轉換得到聲音的頻譜資訊再透過計算頻譜中波峰之間的距離，來得到這個訊號的基頻。

本論文藉助 Sphinx-4 [3] 這個開放原始碼的系統來達成聲音訊號輸入以及快速傅立葉轉換動作。Sphinx-4 這個系統的架構主要包含三個部分：（1）FrontEnd，(2) Decoder 以及(3) Linguist . FrontEnd 可以處理語音訊號的各個模組包括讀進各種格式的語音檔、直接從麥克風讀入語音資料、預加強（preemphasis）、取漢明視窗（Hamming Windowing）、快速傅利葉轉換（FFT）、離散餘弦轉換（DCT）、線性預估編碼等（LPC ,Linear Predictive Encoding）、梅爾倒頻譜轉換係數（MFCC ,mel-cepstral frequency coefficient）等等。

Sphinx-4 這個開放原始碼的聲音處理套件，讓我們可以在較短的時間內製作並完成聲音音高判別的系統。Sphinx-4 是一個完全由 Java 程式碼所撰寫出來的語音辨識系統，所以他可以跨不同的作業系統與機器平台，不僅如此，他已經把許多聲音辨識所需要的組件開發成，所以對於後續的研究，我們只需要再嵌入自己的模組或是修改既有的模組，就可以調整此語音處理辨識系統的功能。

3 樂器音高的判別與音準分析

本論文是透過樂器聲音頻譜的分析來判別樂器所發出來的音高，而大家也都理解樂器所發出的聲音音高與其聲音基頻有著非常直接的關係，這個關係還有一個特點，就是對於人類而言，音高的感受與基頻是成對數性的。例如中央 C 這個音高的頻率是低八度 C 頻率的兩倍，而高八度 C 的頻率又是中央 C 頻率的兩倍，以此類推。在音樂中的所謂的「八度音」，代表的是某個音階與其高音音階的距離，如中央音的 Do 到高音的 Do 。我們在樂器中使用的十二平均律就是將這八度音分成十二個半音。

由於上述人類對聲音的感受，這十二個半音的頻率是依等比關係分佈，並不是以等差關係分佈的。基於這些理由，就有公定的標準把樂器音階中所有半音的頻率分別對應到一個編號 p：

$$p = 69 + 12 \times \log_2\left(\frac{f}{440}\right)$$

反過來，我們也可以由一個半音的編號得到其基頻 f：

$$f = 440 \times 2^{(p-69)/12}$$

這個編號規則是以 69 號音也就是中央 A 爲基準，其頻率爲 440 Hz。由此公式可知，樂器中相鄰兩個半音之間的頻率比爲 $\sqrt[12]{2} \approx 1.05946$。 由這個公式，我們可以推算出中央 A 前後共三個八度音的頻率如表 1。

本論文是以薩克斯風的聲音作爲爲我們判別和分析以及分析的目標。因爲樂器的聲音音高與其聲音的基頻有直接關係，要分析樂器聲音的音高，就必須對樂器的聲音進行頻譜分析。而頻譜分析的方法一般都是將聲音的波形資料通過傅利葉轉換得出頻譜。本論文的做法一樣是藉由電腦程式進行離散傅立葉轉換，再依據轉換過後所得到的頻譜來判斷聲音的音高。

首先我們透過電腦的麥克風來接收樂器所發出的聲音，而在進行離散傅立葉轉換之前，我們先將接收到的聲音訊號進行數位化的取樣工作， 取樣頻率是 44.1 KHz，每個樣本都以 16 位元做編碼。我們將數位化後的聲音訊號切割成百分之一秒的區塊，然後對每一個區塊進行離散傅立葉轉換，轉換的過程我們將 44.1 KHz 頻譜切割成 2048 個頻率， 每個頻率之間的間隔是 21.553 Hz。樂器的聲音經過傅立葉頻譜轉換之後， 我們會得到一個包含 1024 個頻率振幅的向量，根據這個向量，我們會分析其各個頻譜震幅峰值的位置所在，然後再根據這些峰值所對應的頻率，來分析聲音的基頻，以及所衍生出來的泛音。

分析頻譜震幅峰值的所在位置是一個相較之下比較麻煩的問題。我們以分析薩克斯風的聲音所得到的頻譜圖(圖 1)爲例，如果我們把峰值定義成比左右相鄰的兩個頻率震幅都大的頻率爲峰值所在，那麼我們所分析出來的峰值間將會如圖 1 所示。這樣的結果將讓我們難以適當的分析出聲音的音高爲何。如果以較爲人工的方式來判斷(請參考圖 2)，我們可以發現，當樂器的音高比較低的時候，頻譜圖的波峰間隔比較小，其峰值出現的

表 1: 中央 A 前後三個八度音的頻率

	A	♭B	B	C	C♯	D	♭E	E	F	C♯	G	G♯
O4	220	233	246	261	277	294	311	330	349	370	392	415
O5	440	466	494	523	554	587	622	659	698	740	784	831
O6	880	932	988	1047	1109	1175	1245	1319	1397	1480	1568	1661

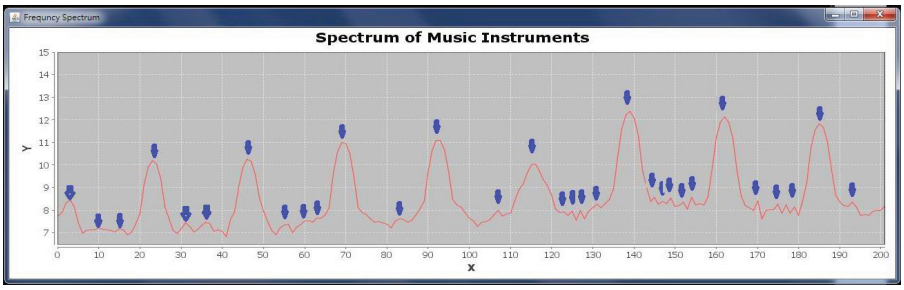
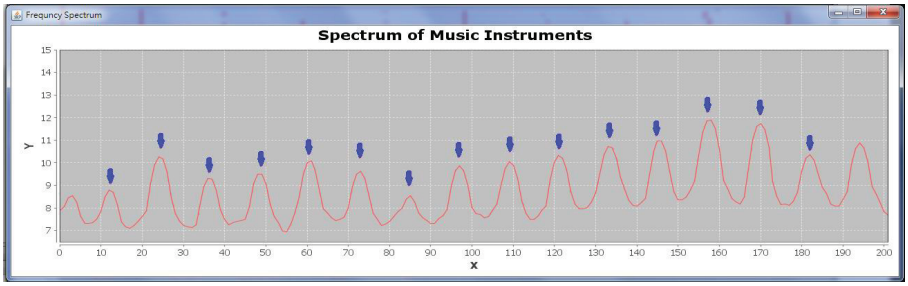
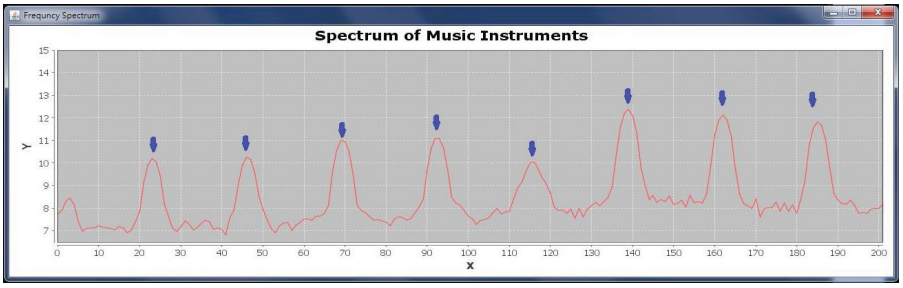


圖 1: 不恰當的峰值位置分析



(a)



(b)

圖 2: 樂器低音頻譜以及高音頻譜峰值

位置分佈比較規律。但是在高音的頻譜圖中，主要的峰值之間的間隔會變大，在這些間隔中也可能出現一些比較不顯著且不規則的小峰值出來，這些不規則的小峰值也會嚴重的影響我們對音高的判斷。所以在分析樂器聲音的時候，我們希望能判斷出這些比較明顯的峰值位置而忽略相對震幅比較小較不明顯的峰值位置。我們使用的演算法大致如下：

1. 對於頻譜圖中由低頻到高頻的每個頻率 f_i 而言$(0 \leq i < 1024)$，設定其鄰近頻率的範圍。如果是頻譜最低頻的部分(頻譜中靠左邊的頻率，第一個峰值尚未判斷出來的時候)，我們設定其鄰近頻率爲 1 到 $2f_i - 1$ 這個區間的頻率。如果第一個峰值已經出現，且其峰值所在頻率爲 f_b，則我們設定 f_i 的鄰近頻率範圍爲 $f_i - f_b + 1$ 到 $f_i + f_b - 1$。

2. 對於 f_i，判斷自己震幅是否爲鄰近頻率震幅中的最大值，如果是，則設定 f_i 爲峰值的所在$(p_i = 1)$，否則設定 $p_i = 0$。如果 f_i 是第一個出現的峰值所在，則我們暫時假設這個頻率 f_i 就是樂器聲音的基頻，此時我們設定基頻頻率 $f_b = f_i$。

3. 重複回到步驟 1 直到所有頻譜頻率都做出使否爲峰值所在的判斷。

以上的演算法可以大致分爲兩個階段第一個階段是找出聲音的基頻所在。如果某個頻率是基頻，那麼頻譜圖裡面所出現的峰值應該會落在基頻以及基頻的倍數上面。所以在基頻的兩倍範圍裡面應該不會出現第二個峰值。所以我們會用兩倍頻率以內範圍來判斷是否爲峰值。因爲如此，在第一個峰值未出現前，我們就以兩倍頻率爲判斷是否爲峰值的範圍。當第一個峰值出現之後，我們先假設其爲樂器聲音的基頻所在，則未來峰值出現的位置應該是在基頻的倍數上，所以我們判斷峰值的範圍必須規範在距離一個基頻的範圍以內，免得較高的峰值把相鄰的峰值給覆蓋掉，而遺失了該有的峰值資訊。 使用以上的演算法，我們可以準確的推算出頻譜中相對震幅較高的那幾個峰值頻率。

得到這些峰值所對應的頻率之後，下一個階段要進行的動作是判斷這些頻率的相對關係。判斷的規則如下：

1. 我們假設第一個峰值的頻率是樂器聲音的基頻，這時其他峰值所對應的頻率屬於這個音高的泛音，其頻率會是這個基頻的倍數。如果眞是如此，我們便可以這個基頻來判定這個聲音的音高。

2. 如果其他峰值所對應的頻率都不是非常接近這個基頻的倍數，我們就不將這個聲音判定成樂器的一個正確音高。

3. 如果其他峰值所對應的頻率大致都接近這個基頻的倍數，但仍然存在些許的偏離，我們仍然會將此聲音判定爲樂器的某一個音高，但是系統也會分析這個聲音的精準度。

綜合以上的幾個規則，我們在音高判斷上設計了以下的一個評分公式：

$$e(i) = \frac{\sum_{p_k=1} d(f_k, f_i)}{N}$$

其中，$e(i)$ 代表我們把 f_i 當作基頻時，與目前頻譜圖矛盾的程度，所以如果此數值越小，則 f_i 是基頻的可能性就越高， $d(f_k, f_i)$ 表示 f_k 偏離 f_i 倍數的程度。也就是說，如果 f_k 越接近 f_i 的倍數的話，其函數值就會越小，N 則表示頻譜圖中峰值的個數。 而對於 $d(f_k, f_i)$ 這個函數的值，我們是以以下方式來計算：

$$d(f_k, f_i) = \frac{\frac{f_k}{2} - \left| f_k - \left(\lfloor \frac{f_k}{f_i} \rfloor \times f_i \right) - \frac{f_k}{2} \right|}{\frac{f_k}{2}}$$

這個函數的回傳值會介於 0 跟 1 之間。如果 f_k 恰好是 f_i 的倍數的話，那他的回傳值將會是 0，如果偏離 f_i 的倍數，則這個函數會接近 1。

對於樂器聲音頻譜圖上的所有頻率 f_i 而言，我們透過以上評分公式去計算各自的 $e(i)$ 值，如果所有 $e(i)$ 值都大於一個門檻值 Δ (我們設定爲 0.1)，則我們就會假設這個聲音不是樂器的正確聲音，否則我們會選擇 $e(i)$ 值最小的頻率當作樂器聲音的基頻，並由這個基頻推算樂器聲音的音高。

至於音高的準確性我們就以這個誤差值來代表。由於系統所使用的傅立葉分析爲離散的傅立葉分析，所以頻譜圖上的頻率並不是連續的，也因此頻率的數值在經過數位化之後，本身就會造成少許的誤差，而這個誤差值會被包含在我們所計算出來的數值裡面，所以我們顯現出來的音高誤差值，會有些微的偏移，但幅度很小。

4 實驗結果

有關樂器聲音音高辨識以及製度分析的結果我們用圖 3 來說明。我們的系統在進行聲音音高辨識的時候會出現三個視窗，位於上方的視窗是用以表示我們所分析出來的峰值頻率所在位置，左下圖是樂器聲音的頻譜圖，而右下方的視窗裡面則顯示我們的系統所辨識出來的音高以及這個聲音的準確度誤差值。

由圖中可以看出我們的系統可以正確的判斷各個頻率的峰值所在，而系統藉由這些峰值也可以分辨出正確的音高。以這個例子而言這個聲音的音準偏誤率也是很低的。在對薩克斯風做音高判別以及音準分析時，我們大部分的聲音音高判斷

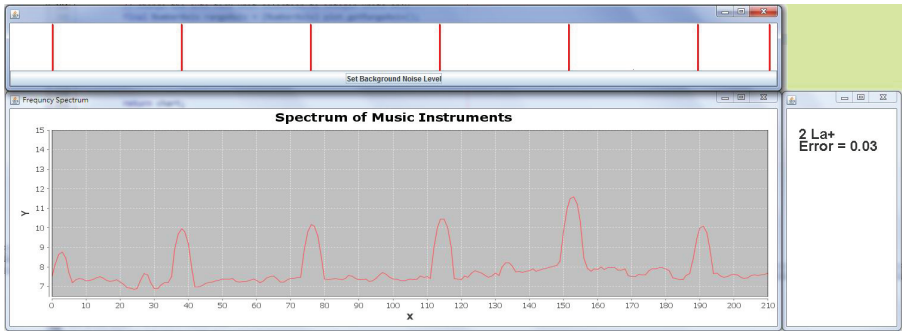


圖 3: 樂器聲音音高的判斷與準度分析結果

結果都是正確的，只有在樂器的聲音音量不穩定或者是聲音音高的轉變非常快速時，會有少許誤判的情況發生，在這些情況之下要維持準確的音高和音準判斷，技術上比較具有挑戰性，這也可以是未來我們要繼續努力克服的方向。

5 結論與未來研究方向

對於一次只會發出單一音高聲音的樂器而言，我們所提出的方法可以即時的分辨出其音高，並且計算出這個聲音音高的誤差值，這對於樂器製作的公司而言會是一個幫助其提升樂器品質的有效工具，使他們能快速地了解其所生產的樂器是否合乎標準，這對於提高這些公司的競爭力也會是一個有效的輔助工具。而對於是可以同時發出多重音高的樂器而言，本論文所提出的方法還不足以做出多重音高的判斷，這也應該是我們未來研究可以努力的方向。

參考文獻

[1] 陳竺儀, 羅峻旗, and 周嬋. 嬰幼兒照護系統之聲音判別技術. In *2015*第二十屆行動計算研討會, Aug. 2015.

[2] 秦伯瑋, 周家麒, and 羅峻旗. 物聯網中聲音識別漸進式學習演算法研究. In *TANET 2017* 臺灣學術網路研討會, Oct. 2017.

[3] Sphinx-4: A speech recognizer written entirely in the java programming language.

[4] Alain De Cheveigné and Hideki Kawahara. Yin, a fundamental frequency estimator for speech and music. *The Journal of the Acoustical Society of America*, 111(4):1917–1930, 2002.

[5] Patricio de la Cuadra, Aaron S. Master, and Craig Stuart Sapp. Efficient pitch detection techniques for interactive music. In *ICMC*, 2001.

[6] Alexandre Savard. Overview of homophonic pitch detection algorithms. 2006.

[7] Yoshiaki Tadokoro, Wataru Matsumoto, and Michiru Yamaguchi. Pitch detection of musical sounds using adaptive comb filters controlled by time delay. In *Multimedia and Expo, 2002. ICME'02. Proceedings. 2002 IEEE International Conference on*, volume 1, pages 109–112. IEEE, 2002.