氣泡排序圖上寬路徑路由問題及寬直徑問題研究

謝育平
銘傳大學資訊工程研究所
arping@gmail.com

王子銘
銘傳大學資訊工程研究所
tmjwang@gmail.com

摘要

在平行計算與分散式系統的研究中，可靠度與效能是相當重要的關鍵。近年來，凱萊圖(Cayley Graph)與寬直徑(Wide Diameter)在常被用來研究該些系統的連結網路(Interconnection Network)狀態。本文研究凱萊圖中的一個分支，氣泡排序圖；研究圖上的寬路徑及寬直徑，提供建立連結網路的選擇與參考。

關鍵字：凱萊圖、氣泡排序圖、寬路徑、寬距離、寬直徑

1. 前言與研究目的

討論網路，可以從兩個方面來看待，一個是協同生成的規則性較弱的網路，此類以網際網路(Internet)為代表，另一類為精心設計，規則性較強的網路，此類以平行計算或分散式系統地連結網路(Interconnection Network)為代表。網際網路係區域網路進行互聯而成，為了能連結眾多的網路型態，很多網路性質是無法假設，但在網際網路的骨幹或單一分散式系統中，建立該系統時就可以依照採用的連結網路設計來假設很多的網路性質，擁有較好的性質就可以讓整系統達到實用性。在討論的網路性質中，主要以可靠度(Reliability)與效能(Efficiency)為重要的議題。

網路的可靠度，主要探討節點失效後，整個網路的訊息傳送是否還能進行，即所謂的容錯性(Fault Tolerance)[11][13]，在網路性質上主要探討網路的連通性(Connectivity)來探討；而網路的效能則是討論兩個節點間傳遞訊息的速度，這速度包含單路徑傳輸或多路徑傳輸，也需要考量網路流量現況，在網路性質上主要探討網路的直徑(單路徑最遠距離)與寬直徑(多路徑最遠距離)。

在網路性質中，從一個開始節點出發，往一個目的節點傳送資料，在單一路徑時，我們考慮整個網路直徑，用來代表整個網路任兩個節點傳送資料時，最遠需要經過幾個節點。在考慮多路徑時，我們會從一個開始節點出發，同時發展數條路徑，往同一個目的節點傳送資料。此時為了保證資料能順利傳送，沒有發生擁擠的狀況，假設這些路徑是節點互斥的路徑(開始節點與目的節點除外)，另外為了達到越快的傳輸，可以同時發展盡量多的路徑。對一個連通性(Connectivity)為k的圖來說，我們可以同時發展k-1條路徑，並探討這k-1條路徑的長度。重新定義寬路徑(container)、寬距離、最短寬路徑、寬直徑等概念。

假設G(V,E)為一個無重複邊的無向圖，V為該圖的節點集合，$E \subset V \times V$為該圖的邊集合；假設G(V,E)是無向圖，所以有$\forall_{x\in V}(x,x) \notin E$，$\forall_{x,y\in V}(x,y) \in E \Rightarrow (y,x) \in E$。對於圖G中的兩點$a,b \in V$，由a連結到b的路徑為一個點序列$p[v_0, v_1, v_2, \dots, v_k]$，其中$a = v_0$，$b = v_k$，$\forall_{0\le i\le k} v_i \in V$，且$\forall_{1\le i\le k}(v_{i-1}, v_i) \in E$。路徑p=的長度則定義為L(p)=k。定義$P(a,b) = \{p|p$為一個由a連結到b的路徑$\}$為所有從a到b所有路徑的集合。a、b兩點的距離$d(a,b) = \min\{L(p)|p \in P(a,b)\}$則定義為所有從a到b的最短路徑的長度。圖G的直徑$d(G) = \max\{d(x,y)|x,y \in V\}$則定義為距離最遠的兩個點的距離。

一張圖G為非連通圖定義為存在G中的兩個節點a,b使得不存在從a連結到b的路徑$P(a,b) = \phi$。一張圖G的連通性(Connectivity) κ(G)定義為最少需要移除多少節點才能使圖G成為非連通圖或是單一節點圖。Menger's定理(1927)[12]提到一個連通性為κ(G)(=w)的圖G上，對任兩點a,b，從a到b必存在w條節點互斥(a、b例外)的路徑。對於一個路徑$p=[v_0, v_1, v_2, \dots, v_k]$來說，可以將之分為三個，開頭部分$H(p) = \{v_0\}$，結尾部分$T(p) = \{v_k\}$，與中間部分$I(p) = \{v_1, v_2, v_3, \dots, v_{k-1}\}$，將這三個部分再延伸4個集合$HIT(p) = H(p) \cup I(p) \cup T(p)$ 、 $HI(p) = H(p) \cup I(p)$ 、$HT(p) = H(p) \cup T(p)$、$IT(p) = I(p) \cup T(p)$。兩個路徑p與q是中間節點互斥是指$I(p) \cap I(q) = \phi$。

對於一張連通圖G中的兩點$a,b \in V$，開始考慮由a連結到b的多個互斥路徑，定義從a到b的w寬路徑(Container)[5][7][14]為w條從a到b的中間節點互斥路徑$c = \{p_1, p_2, p_3, \dots, p_w\}$，其中$\forall_{1\le i\le w} H(p_i) = \{a\}$，$\forall_{1\le i\le w} H(p_i) = \{b\}$ ，$\forall_{1\le i<j\le w} I(p_i) \cap I(p_j) = \phi$。對於寬路徑$c = \{p_1, p_2, p_3, \dots, p_w\}$，我們定義其寬度(width)為$W(c) = |c| = w$，長度為$L(c) = \max\{L(p)|\ p \in c\}$。定義$C_w(a,b) = \{c|c$為從a到b的w寬路徑$\}$為所有從a到b的w寬路徑所成集合。從a到b的w寬距離$d_w(a,b) = \min\{L(c)|c \in C_w(a,b)\}$，而對一張圖$G = (V,E)$來說，G的w寬直徑$d_w(G) = \max\{d_w(a,b)|a,b \in V\}$則定義為在G中w寬距離最遠的兩點的w寬距離。寬路徑、寬長度、寬距離、寬直徑，可視為路徑、長度、距離、直徑的延伸，這是因為當w=1時，寬路徑、寬長度、寬距離、寬直徑的定義都與路徑、長度、距離、直徑是協調一致。進一步來說，對於一個連通性為k的圖G來說，$d_1(G) \le d_2(G) \le d_3(G) \le \cdots \le d_k(G)$。

凱萊圖$G = Cayley(S,\Sigma) = G(V,E)$[4][5][7][8][10]由一個乘法群(S,×)及該群中的一子集$\Sigma \subset S$建構而得，其中乘法群中的元素為凱萊圖的節點($V = S$)，而凱萊圖的邊E則為$\{(s, s\times\sigma)|s \in S, \sigma \in \Sigma\}$，此處稱Σ為該凱萊圖的產生集(Generating Set)。如果$\Sigma = \Sigma^{-1} = \{\sigma^{-1}|\sigma \in \Sigma\}$，則所創造的凱萊圖將會是個無向圖。

給定一圖G(V,E)，圖上兩個點$x,y \in V$，兩點間的w-寬路徑(Container)定義為兩點間w條互斥的路徑集合，其長度定義為當中最長路徑的長度；兩點間的w-寬距離$d_w(x,y)$定義為兩點間最短w-寬路徑的長度；而該圖的w-寬直徑$d_w(G)$則定義為該圖中任兩點的w-寬距離的最大值。

而在圖論上，超立方體（hypercubes）、交叉立方體（crossed cubes）、雙扭立方體（twisted cubes）、局部雙扭立方體（locally twisted cubes）、廣義雙扭立方體（generalized twisted cubes）、莫氏立方體（Mobius cubes）、星圖（star graphs）、氣泡排序圖（bubble-sort graphs）、煎餅圖（pancake graphs）和交替群圖(alternating group graph)等都是連結網路設計主要被探討的模型設計。Chi-Hsiang Yeh and Emmanouel A. Varvarigos (1998)[18]

便針對著名的網路拓樸整理出其圖論特性，特別是以節點數N作為比較的基準，整理如表1：值得觀察的一件事是，這些圖的節點鏈結度、直徑相對於節點數N都是非常小的，顯示這些圖大量節點，低鏈結，低距離(直徑)的好特性。

考慮n元素的對稱群(Symmetric Group)。假設$Z_n = \{1,2,3,\dots,n\}$，一個Z_n上的重排函數$f: Z_n \to Z_n$為一個一對一且映成函數。令$S_n = \{f | f: Z_n \to Z_n$，f為一對一且映成函數$\}$為$Z_n$上的重排函數集合；考慮函數結合運算子$f \circ g(x) = f(g(x))$，則$(S_n, \circ)$為一個群；考慮$P_n = \{f_k | k = 1,2,3\dots,n-1, f_k(k) = k+1, f_k(k+1) = k, \forall_{x \neq k, x \neq k+1} f_k(x) = x\}$為$Z_n$上兩鄰兩數交換的重排函數，有重排本身有其迴圈表達式(cycle form)，例如P_n中的f_k，其迴圈表達式為$(k, k+1)$，所以$P_n = \{(1,2),(2,3),(3,4),\dots,(n-1,n)\} \subset S_n$為氣泡排序法上相鄰元素互換的動作。而最後氣泡排序圖[1][9][14][15][17]被定義為凱萊圖$B_n = Cayley(S_n, P_n)$。另外與氣泡排序圖非常相關的兩個凱萊圖為星狀圖(star graph)[2][3]與置換圖(transposition graph)[6][16]，其最主要的差別就是星狀圖的生成集為$\{(1,2),(1,3),(1,4),\dots,(1,n)\}$，而置換圖的生成集為$\{(i,j) | 1 \leq i,j \leq n, i \neq j\}$

氣泡排序圖B_n以n元素的重排為節點，兩個重排如果是只有在兩個相鄰的位置上不同，則這兩的節點有邊線相連，反之則無。觀察圖1，以n=4為例，4元素的重排共有4!=24個，組成B_4的24個節點，任兩個節點是否有邊相連，則觀察其是否只有在兩個相鄰的位置上不同，例如 4321與3421有邊相鄰是因為其只有在第1與第2位置上不一樣。因為其判斷是否有邊是依照是否只有在兩個相鄰的位置上不同，其與氣泡排序演算法上只交換相鄰的兩個元素狀況類似，故取名氣泡排序圖。

氣泡排序圖B_n計有n!個節點，每個節點有(n-1)個邊，整個網路共有n!(n-1)/2個邊，其直徑為n(n-1)/2，其連通度是(n-1)，可以探索同時發展(n-1)條互斥路徑傳送訊息。氣泡排序圖是屬於低度連結的網路，以該圖節點數N=n!來衡量，其鏈結數(n-1)與直徑n(n-1)/2都是非常小的，再加上氣泡排序圖對任何兩點來說都是同構的，對任何兩邊來說也是同構的。

觀察【圖 1.1】，在B_4中，1243與1234的距離為1，但是如果是要從1243同時發展3條互斥路徑到1234，則有以下的解答，[1243→1234]、[1243→2143→2134→1234]、[1243→1423→1432→1342→1324→1234]，此三條路徑構成1243到1234的寬度為3的寬路徑，該寬路徑長度以其中最長一條路徑的長度為主，即該寬路徑的長度為5，而且對於所有寬度為3的寬路徑，5是最短的長度了，所以該組寬路徑事實上就是該兩點的最短寬度為3的寬路徑，我們定義該兩點的3-寬距離為5。圖1中在節點右下方的數字為該節點到1234的3-寬距離。又因為氣泡排序圖是節點同構，所以在計算兩點的距離時，可以將其中一點依同構函數對應至單位元素1234，例如計算2431與2314的距離時，可以將兩個重排同時與重排 3124進行函數合成，可得1423與1234。所以在計算氣泡排序圖的寬直徑時，可以計算所有點到單位重排的寬距離，取其最大者即為寬直徑。所以氣泡排序圖B_4的3-寬直徑為8，也就是同時要發三條互斥路徑的要求下，2143與1234是氣泡排序圖B_4中3-寬距離最遠的兩個節點，其3-寬距離為8。

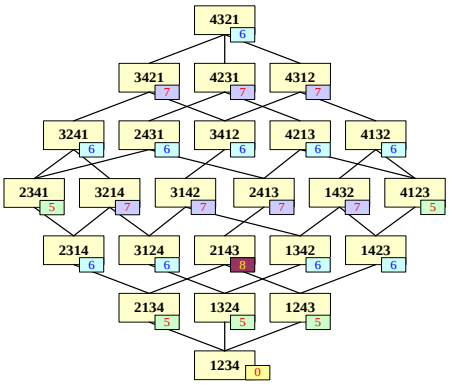

圖 1.1：氣泡排序圖 B_4

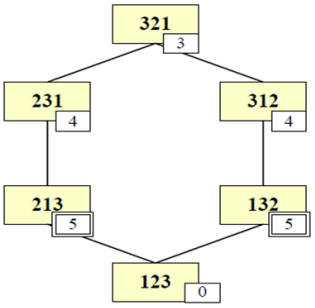

圖 1.2：氣泡排序圖 B_3

表 1.1：網路拓樸圖論特性

圖型	節點數	節點鏈結數	直徑
超立方體 Hypercube	N	$\log_2 N$	$\log_2 N$
星狀圖 Star graph	N	$\frac{\log_2 N}{\log_2 \log_2 N} + O\left(\frac{\log N}{\log \log N}\right)$	$\frac{1.5 \log_2 N}{\log_2 \log_2 N} + O\left(\frac{\log N}{\log \log N}\right)$
D 維網 D-dimensional mesh	N	2d	$\theta(N^{\frac{1}{d}})$
De Bruijn 圖	N	4	$\log_2 N$
立方體連通圖 Cube-connected cycle graph	N	3	$2.5 \log_2 N - O(\log \log N)$
星狀連通圖 star-connected cycle graph	N	3	$\theta\left(\frac{\log^2 N}{(\log \log N)^2}\right)$
Balance Macro Star (1,n)	N	$2\sqrt{\frac{\log_2 N}{\log_2 \log_2 N}} + O\left(\sqrt{\frac{\log N}{\log \log N}}\right)$	$\frac{2.5 \log_2 N}{\log_2 \log_2 N} + O\left(\frac{\log N}{\log \log N}\right)$

一個n元素的重排函數f，可以使用序列來表達，即詳列f(1),f(2),f(3),…,f(n)；圖1是氣泡排序圖B_4即其連線狀況。其中每一個節點代表一個重排函數，圖上函數序列表達式省略了逗號，4321表示f(1)=4, f(2)=3, f(3)=2, f(4)=1。又因為氣泡排序圖是節點同構，所以在計算兩點的距離時，可以將其中一點依同構函數對應至單位元素，也就是對於兩個重排f與g，如果$(f,g) \in E$，則存在一個k使得$g = f \circ (k, k+1)$，則假設h為g的左逆函數，也就是$h \circ g(x) = h\big(g(x)\big) = id(x)$，其中id是單位函數，也就是$\forall_x id(x) = x$，則$id = h \circ g = h \circ f \circ (k, k+1)$，所以計算f與g的距離，可以轉為計算id與$h \circ f$的距離，所以如果要計算氣泡排序圖的直徑或寬直徑，可以將考慮所有點到原點(id)的距離或寬距離，其最大值即為直徑與寬直徑。所以圖1中，每個方框上面的數字代表該節點與單位函數1234節點的3-寬距離，而氣泡排序圖B_4的寬直徑為8，可由2143與1234撐起。而圖2則是氣泡排序圖B_3，在B_3中，我們探討每一節點同時發兩條路徑到節點123，每一個節點右下方數字表示其到單位函數(id)的2-寬距離，所以B_3的寬直徑為5。

我們臆測$D(B_n) = d(B_n) + 1$，不但是縮小範圍，而且是確切希望$D(B_n) = d(B_n) + 1$，對於$D(B_2)$、$D(B_3)$、$D(B_4)$不滿足臆測，我們認為是因為網路拓樸太小所造成的特例，因為點數少，多路徑繞行時難免會互搶節點，但是氣泡排序圖的節點數成階乘式成長，而路徑寬度成線性成長，所以互搶節點的狀況應該是越來越少。為了這項臆測，在研究開始前我們在實驗上與理論上，都預先進行了一些研究工作，實驗上我們預先計算到$D(B_8)$，發現自B_5以後都符合我們的臆測，見【表1.2】。

表 1.2：氣泡排序圖 B_n 上的直徑與(n-1)寬直徑

網路	節點	鍊結	總邊	$d(B_n)$	$D(B_n)$	符合臆測
B_2	2	1	1	1	1	否
B_3	6	2	6	3	5	否
B_4	24	3	36	6	8	否
B_5	120	4	240	10	11	是
B_6	720	5	1,800	15	16	是
B_7	5,040	6	15,120	21	22	是
B_8	40,320	7	141,120	28	29	是

此次對於氣泡排序圖的寬直徑問題，本研究欲完成以下幾項目標：從實驗上，透過更好的演算法，更多的機器進行平行運算，計算更多的$D(B_n)$的寬直徑，觀察其是否支撐我們的臆測，期待計算至$D(B_{11})$或更高維度；第四章開始介紹本研究的各種演算法的發展過程，第五章列出實驗結果及結論。

2. 文獻探討

Yasuto Suzuki & Keiichi Kaneko(2008)[14]將氣泡排序圖、k元n維環面、超立方體等圖，依其分支度、節點、直徑…等特性，整理如【表 2.1】其中基礎參數係由各網路拓樸生成時的自然參數，其中對稱性係指節點同構，也就是從一節點看待整個網路，可以與從另一任意節點看待整個網路同構，而自我內嵌性係指該系列網路拓樸中，小型網路拓樸是否內嵌於大型網路拓樸中；例如氣泡排序圖B_3就會內嵌於氣泡排序圖B_4中，而氣泡排序圖B_4會內嵌於氣泡排序圖B_5中。

氣泡排序圖B_n的(n-1)-寬路徑路由問題，係指對於氣泡排序圖上的任兩個點，該如何從其中一點出發，同時連接(n-1)條互斥路徑(頭尾除外)，到達另一節點。

在這問題上，學者關心幾個特性，(1)該如何尋找寬路徑，(2)整個網路拓樸的最大(n-1)-寬距離((n-1)-寬直徑)會是多少?(3)前兩個議題的時間複雜度會是多少?

表 2.1：氣泡排序圖 B_n 上(n-1)寬路徑問題的進展

		K. Kaneko、Y. Suzuki(2008)	K. Kaneko、Y. Suzuki (2008)
理論證明	最大(n-1)-寬路徑長度	$O(n^3)$	$O(n^2)$
	時間複雜度	$O(n^4)$	$O(n^4)$
實驗觀察	平均(n-1)-寬路徑長度	$O(n^{2.94})$	$O(n^{1.95})$
	時間複雜度	$O(n^{3.9})$	$O(n^{3.5})$

表 1.3：著名網路拓樸的圖論性質

圖形	節點數	節點鏈結數	直徑	方向性	對稱性	自我內嵌性
氣泡排序圖	n!	$n-1$	$\frac{n(n-1)}{2}$	無向圖	有	有
k 元 n 維環面	k^n	4	$n \times \lfloor \frac{k}{2} \rfloor$	無向圖	有	無
超立方體	2^n	n	n	無向圖	有	有
燒餅圖	n!	$n-1$	$\leq \lfloor \frac{5(n-1)}{3} \rfloor$	無向圖	有	有
星狀圖	n!	$n-1$	$\lfloor \frac{3(n-1)}{2} \rfloor$	無向圖	有	有
旋轉圖	n!	$n-1$	$n-1$	有向圖	有	有
De-Bruijn 圖	n^k	n	k	無向圖	無	無
Kautz 圖	$n^k + n^{k-1}$	n	k	無向圖	無	無

如【表 2.1 】所述，Keiichi Kaneko與 Yasuto Suzuki (2004)[9]對氣泡排序圖發表一個寬路徑尋找方式，並理論上證明氣泡排序圖B_n的(n-1)-寬直徑在$O(n^3)$中，其尋找單一條(n-1)-寬路徑的時間在$O(n^4)$；而在實驗上觀察氣泡排序圖B_n的平均(n-1)-寬路徑長度在$O(n^{2.94})$中，其尋找單一條(n-1)-寬路徑的時間在$O(n^{3.9})$。四年後，Yasuto Suzuki與 Keiichi Kaneko (2008)[14]針對氣泡排序圖再次發表一個寬路徑演算法，並理論上證明氣泡排序圖B_n的(n-1)-寬直徑小於$n(n+1)/2$，在$O(n^2)$中，其尋找單一條(n-1)-寬路徑的時間在$O(n^4)$；而在實驗上觀察氣泡排序圖B_n的平均(n-1)-寬路徑長度在$O(n^{1.95})$中，其尋找單一條(n-1)-寬路徑的時間在$O(n^{3.5})$。

給定一張圖G，我們使用$d(G)$表示圖G的直徑，使用$d_w(G)$表示圖G的w-寬直徑，也就是同時發w條情況下，圖G中的最大的最短w-寬路徑。對於氣泡排序圖B_n來說，我們關心(n-1)-寬直徑，所以我們特別使用$D(B_n) = d_{n-1}(B_n)$來表達氣泡排序圖B_n的(n-1)-寬直徑。本研究初步探索，實驗發現幾個氣泡排序圖B_n的寬直徑在$n \geq 5$後，只比直徑多1，也就是$D(B_n) = d(B_n) + 1$，數據顯示如【表1.24 】。

Yasuto Suzuki與 Keiichi Kaneko (2008)[14]是將$D(B_n)$從$O(n^3)$降成$O(n^2)$，事實上是降到$n(n+1)/2$，也就是$D(B_n) \leq d(B_n) + n$，但是經過我們做了幾個實驗，我們臆測$D(B_n) = d(B_n) + 1$，不但是縮小範圍，而且是確切希望$D(B_n) = d(B_n) + 1$，對於$D(B_2)$、$D(B_3)$、$D(B_4)$不滿足臆測，我們認為是因為網路拓樸太小所造成的特例，因為點數少，多路徑繞行時難免會互搶節點，但是氣泡排序圖的節點數成階乘式成長，而路徑寬度成線性成長，所以互搶節點的狀況應該是越來越少。為了這項臆測，我們在實驗上與理論上，都預先進行了一些研究工作，在研究開始前實驗上我們已經計算$D(B_n)$到n=8，發現從n=5以後，都是符合我們的臆測的。

對於氣泡排序圖的$B_n = Cayley(S_n, P_n)$，我們猜測$\forall_{n\geq 5} d_{n-1}(B_n) = d_1(B_n) + 1$。Y. Suzuki與 K. Kaneko (2008)[14] 已經有$\forall_n d_{n-1}(B_n) \leq \frac{n(n+1)}{2} = d_1(B_n) + n$，並在$O(n^4)$解出寬路徑問題，並得到長在$d_1(B_n) + n$以內的寬路徑，進而得到寬直徑在$d_1(B_n) + n$以內。此處，我們企圖更上層樓，我們希望能夠證明$\forall_{n\geq 5} d_{n-1}(B_n) = d_1(B_n) + 1$，並在$O(n^3)$解出寬路徑問題，並得到長在$d_1(B_n) + 1$以內的寬路徑，進而得到對於$n \geq 5$寬直徑恰等於$d_1(B_n) + 1$以內。

對於氣泡排序圖$B_n (n \geq 5)$，假設$Z_n = \{1,2,3,\dots,n\}$，考量S_n上的每一個重排f，重排 f 有好幾種表達方式，一個是函數表達$f: Z_n \to Z_n$，一個是迴圈表達式，此處採用序列表達式$< f(1), f(2), f(3), \dots, f(n) >$，而一個序列表達式$< s_1, s_2, s_3, \dots, s_k >$代表一個函數$f: Z_k \to \mathcal{N}$，$f(1) = s_1$，$f(2) = s_2$，$f(3) = s_3$，、、、、$f(k) = s_k$，此處特別注意$< s_1, s_2, s_3, \dots s_k >$代表的函數為一個$Z_k$到自然數集合$\mathcal{N}$的函數，而非要求$Z_k$到$Z_k$的函數，例如$< 2,4,3,7 >$代表函數$f: Z_4 \to N$，$f(1) = 2$、$f(2) = 4$、$f(3) = 3$、$f(4) = 7$。

表 2.2：Suzuki 對節點的五個分類

Case I	$d_n = n$
Case II	$d_n \neq n-1, d_{n-1} = n$
Case III	$d_n, d_{n-1} \neq n$
Case IV	$S^{n-1} = d$
Case V	$d_n = n-1, d_{n-1} = n, s^{(n-1)} \neq d$

Suzuki針對氣泡排序問題，客製化了一套演算法，依照節點$d =< d_1, d_2, d_3, \dots, d_n >$的性質，將所有節點分成如【表 2.2】的5個分類，再依照各分類逐一突破。而其寬路徑尋找演算法大致如【表 2.3】。其中$s^{(n-1)} =< 1,2,3, \dots, n-2, n, n-1 >$。

表 2.3：Suzuki 演算法

步驟	輸入：起始節點（$\mathbf{s = id} =< \mathbf{1,2,3,\dots,n} >$）與目標節點（$\mathbf{d} =< \mathbf{d_1, d_2, d_3, \dots, d_n} >$） 輸出：(n-1)-寬路徑
步驟一	分析起始節點與目標節點，若$\mathbf{d_n = n}$，則跳至步驟二，若$\mathbf{d_n \neq n-1, d_{n-1} = n}$則跳至步驟三，若$\mathbf{d_n, d_{n-1} \neq n}$則跳至步驟四，若$\mathbf{S^{n-1} = d}$則跳至步驟五，若$\mathbf{d_n = n-1, d_{n-1} = n, s^{(n-1)} \neq d}$則跳至步驟六。
步驟二	使用遞迴獲得$\mathbf{n-2}$條從 S 到 d 的最短路徑，最後一條先使用最短路徑演算法先到$\mathbf{a} = (\mathbf{1,2,\dots, d_{n-1}-1, d_{n-1}+1, \dots n-2, n, n-1, d_{n-1}})$，再從節點 a 使用最短路徑演算法到$\mathbf{d_{n-1}}$接著再到 d，並跳至步驟七。
步驟三 步驟四 步驟五 步驟六	在 CaseII、CaseIII、CaseIV、CaseV 中，Suzuki 在 s 與 d 之間建立兩組中繼節點，首先從 s 建立最短路徑到$\mathbf{c_i^n}$其中$\mathbf{c_i} = (\mathbf{1,2,\dots i-1, i+1, \dots n-1, i, n})$接著再使用最短路徑演算法到$\mathbf{b_i^n}$其中$\mathbf{b_i} = (\mathbf{d_1, d_2, \dots d_{l_i-1}, d_{l_i+1}, \dots, d_{n-1}, i, d_n})$，接著再從$\mathbf{b_i}$到 d，並跳至步驟七。詳細請見[14]。
步驟七	印出路徑。

Suzuki在理論上提供了尋找$d_1(B_n) + n$長度的寬路徑演算法，但是我們的企圖不僅於此，我們希望能證出主臆測$\forall_{n\geq 5} d_{n-1}(B_n) = d_1(B_n) + 1$。

表 2.4：Yasuto Susuki 兩篇研究結果及本研究目標

氣泡排序圖B_n中	K. Kaneko 與 Y. Suzuki (2004)	Y. Suzuki 與 K. Kaneko (2008)	本研究目標
任兩點的(n-1)-寬路徑長度	$O(n^3)$	$\leq d_1(B_n) + n = O(n^2)$	$\leq d_1(B_n) + 1 = O(n^2)$
(n-1)-寬路徑尋找時間	$O(n^4)$	$O(n^4)$	$O(n^3)$
(n-1)-寬直徑	$O(n^3)$	$\leq d_1(B_n) + n = O(n^2)$	$d_1(B_n) + 1$
(n-1)-寬直徑尋找時間	$O(n^4)$	$O(n^4)$	$O(1)$

3. 研究方法

本研究主要著重在如何解決氣泡排序圖的寬路徑及寬直徑問題。本研究有兩個目標：(1)發展演算法解決氣泡排序圖B_n的(n-1)寬路徑路由問題並使用程式計算$D(B_9)$、$D(B_{10})$及更多的數據來支撐主臆測$\forall_{n\geq 5} D(B_n) = d(B_n)+1$。(2)從理論上來證明主臆測，使本研究及主臆測更加完備。

對於寬路徑問題，我們提出四種演算法試圖解決寬路徑的路由問題。(1)重複呼叫法、(2)最短路徑指引多路徑同時發展演算法、(3) 秩指引多路徑同時發展演算法、(4)有限長度秩指引多路徑同時發展演算法。

最短路徑演算法

對於寬路徑問題，一個最直覺的演算法就是使用最短路徑演算法(如dijkstra演算法)先尋找一條最短路徑，固定下來，然後在途中移除該路徑的中間節點，再執行一次尋找第二條最短路徑，並避免掉已經經過的節點，再固定下來，如此執行(n-1)次即可取得共(n-1)條互斥的路徑所組成寬路徑。

演算法 1：
最短路徑演算法(Ver.1) 重複呼叫法

	輸入：給定一整數 n 及氣泡排序圖$\mathbf{B_n}$中的一個目的節點 f。 輸出：一組內部互斥的從單位節點(id)往目的節點 f 的(n-1)條路徑。
步驟 1	令$\mathbf{G(V,E)} = \mathbf{B_n}$ 為初始圖。 令$\mathbf{c} = \emptyset$用來記載寬路徑。
步驟 2	以最短路徑演算法在圖$\mathbf{G(V,E)}$上找出 f 至原點(id)的路徑$\mathbf{p} = [\mathbf{f}, \mathbf{r_1}, \mathbf{r_2}, \dots, \mathbf{r_m}, \mathbf{id}]$，令$\mathbf{c} = \mathbf{c} \cup \{\mathbf{p}\}$，令$\mathbf{G} = (\mathbf{V'}, \mathbf{E'})$，其中$\mathbf{V'} = \mathbf{V} \backslash \{\mathbf{r_1}, \mathbf{r_2}, \dots, \mathbf{r_m}\}$，$\mathbf{E'} = \mathbf{E} \cap (\mathbf{V'} \times \mathbf{V'})$，即在 G 中摘除$\{\mathbf{r_1}, \mathbf{r_2}, \dots, \mathbf{r_m}\}$後剩餘的網路。
步驟 3	若已尋得 n-1 條路徑，即$\mathbf{|c| = n-1}$，則結束演算法並輸出所有已得路徑 c。
步驟 4	重複步驟 2。

使用演算法1計算完所有節點後，發現實驗結果各節點路徑之直徑易超過 d(n)+1，且因本演算法一次發展完一條最短路徑，容易造成路徑數越接近n-1時長度越容易超過d(n)+1。例如在【圖1.1】中，[2431→1234]時，最短路徑演算法(Ver.1)會找出三條長度各為4、4、8的路徑，但我們手動優化計算的寬路徑內的路徑長度應為4、6、6。

我們經修正後發展出Ver.2。Ver.2與Ver.1不同之處在於Ver.2同時發展n-1條路徑，循序步進且自此開始引入隨機挑選的概念，既可快速找出互斥之路徑也可避免Ver.1無法另尋路徑的問題。以下介紹演算法2之演算步驟。

演算法 2
最短路徑指引多路徑同時發展演算法

步驟 1	以最短路徑演算法找出所有節點至原點(id)的距離。令 d(g)為 g 與原點(id)的距離。
步驟 2	從目標節點 f 出發，從相鄰節點$\{\mathbf{f_1}, \mathbf{f_2}, \dots, \mathbf{f_{n-1}}\}$發展出 n-1 條路徑$\mathbf{c} = \{\mathbf{p_1}, \mathbf{p_2}, \mathbf{p_3}, \dots, \mathbf{p_{n-1}}\}$，其中$\mathbf{p_i} = [\mathbf{id}, \mathbf{f_i}]$。令$\mathbf{U} = \{\mathbf{f}, \mathbf{f_1}, \mathbf{f_2}, \dots, \mathbf{f_{n-1}}\}$用來記載(n-1)路徑所經過的節點。
步驟 3	從 c 中的選出未到達原點之最短路徑 p，設 g 為路徑$\mathbf{p} = [\mathbf{f}, \mathbf{r_1}, \mathbf{r_2}, \dots, \mathbf{r_m}, \mathbf{g}]$的最後一個節點，考慮 g 的 n-1 個鄰居$\mathbf{gs} = \{\mathbf{g_1}, \mathbf{g_2}, \mathbf{g_3}, \dots \mathbf{g_{n-1}}\}$，另$\mathbf{gs'} = \mathbf{gs} \backslash \mathbf{U}$為出現在 gs 中但不出現在 U 中的節點。在$\mathbf{gs'}$中隨機挑選一節點$\mathbf{g'}$。我們將 g'連結到路徑 p 的後方，即重新令$\mathbf{p} = [\mathbf{f}, \mathbf{r_1}, \mathbf{r_2}, \dots, \mathbf{r_m}, \mathbf{g}, \mathbf{g'}]$。如果$\mathbf{g'} \neq \mathbf{id}$，我們將$\mathbf{g'}$加入 U 中以維護路徑的互斥性，即令$\mathbf{U} = \mathbf{U} \cup \{\mathbf{g'}\}$。
步驟 4	如果 c 中所有路徑皆已達原點，則輸出 c 並結束本演算法，否則重複步驟 3。

使用演算法結束2實驗後，發現在隨機挑選的過程中，無法得知候選節點與原點的最短路徑，再優先挑選之，因此我們繼續修訂演算法。

演算法3將記錄所有節點抵達id的距離，讓演算法在隨機挑選時可優先挑出與id最短距離的節點。意思是說，每一個節點可以記錄數條最短路徑的方向供挑選。在氣泡排序圖中，最短多路徑方向其實就是擁有較少逆序的鄰居方向，而一個重排逆序的個數稱為秩，故稱秩指引多路徑同時發展演算法。

演算法 3
秩指引多路徑同時發展演算法

步驟 1	以最短路徑演算法找出所有節點至原點(id)的距離。令 d(g)為 g 到原點(id)的距離。
步驟 2	從目標節點 f 出發，從相鄰節點$\{\mathbf{f_1}, \mathbf{f_2}, \dots, \mathbf{f_{n-1}}\}$發展出 n-1 條路徑$\mathbf{c} = \{\mathbf{p_1}, \mathbf{p_2}, \mathbf{p_3}, \dots, \mathbf{p_{n-1}}\}$，其中$\mathbf{p_i} = [\boldsymbol{id}, \boldsymbol{f_i}]$。令$\mathbf{U} = \{\mathbf{f}, \mathbf{f_1}, \mathbf{f_2}, \dots, \mathbf{f_{n-1}}\}$用來記載(n-1)路徑所經過的節點。
步驟 3	從 c 中的選出未到達原點之最短路徑 p，設 g 為路徑$\mathbf{p} = [\mathbf{f}, \mathbf{r_1}, \mathbf{r_2}, \dots, \mathbf{r_m}, \mathbf{g}]$的最後一個節點，考慮 g 的 n-1 個鄰居$\mathbf{gs} = \{\mathbf{g_1}, \mathbf{g_2}, \mathbf{g_3}, \dots \mathbf{g_{n-1}}\}$，另$\mathbf{gs'} = \mathbf{gs} \backslash \mathbf{U}$為出現在 gs 中但扣除出現在 U 中的節點。考慮$\mathbf{gs'}$中的節點，令$\mathbf{g'} = \mathbf{argmin}\{\mathbf{d(h)} | \mathbf{h} \in \mathbf{gs'}\}$為$\mathbf{gs'}$集合中距離 id 最近之節點(即秩最小者)，若最近之節點有複數個，則隨機挑選之。我們將 g'連結到路徑 p 的後方，即重新令$\mathbf{p} = [\mathbf{f}, \mathbf{r_1}, \mathbf{r_2}, \dots, \mathbf{r_m}, \mathbf{g}, \mathbf{g'}]$。如果$\mathbf{g'} \neq \mathbf{id}$，我們將$\mathbf{g'}$加入 U 中以維護路徑的互斥性，即令$\mathbf{U} = \mathbf{U} \cup \{\mathbf{g'}\}$。
步驟 4	如果 c 中所有路徑皆已達原點，則輸出 c 的內容並結束本演算法。
步驟 5	重複步驟 3。

對於寬直徑問題來說，從計算就是需要計算所有節點到原點的寬距離，為了因應龐大的節點計算量，以及節省時間增進效能，我們再次修訂演算法。

因為氣泡排序圖的寬直徑臆測為$\mathbf{d(B_n)+1}$，所以演算法4將直接放棄尋找路徑的過程中距離已經超過$\mathbf{d(B_n)+1}$的節點，待全部節點皆尋找完畢後，去除已完成(即其長度在$\mathbf{d(B_n)+1}$的範圍內)的節點後，將尚未完成的節點再尋找一次，直到所有節點的長度皆落在$\mathbf{d(B_n)+1}$內，加入本方法後將可使得寬直徑運算的耗費時間大幅降低。

演算法 4
有限長度秩指引多路徑同時發展法

步驟 1	以最短路徑演算法找出所有節點至原點(id)的距離。令 d(f)為 f 到原點(id)的距離(最短路徑的長度)。
步驟 2	從目標節點 f 出發，從相鄰節點$\{\mathbf{f_1}, \mathbf{f_2}, \dots, \mathbf{f_{n-1}}\}$發展出 n-1 條路徑$\mathbf{c} = \{\mathbf{p_1}, \mathbf{p_2}, \mathbf{p_3}, \dots, \mathbf{p_{n-1}}\}$，其中$\mathbf{p_i} = [\mathbf{id}, \mathbf{f_i}]$。 令$\mathbf{U} = \{\mathbf{f}, \mathbf{f_1}, \mathbf{f_2}, \dots, \mathbf{f_{n-1}}\}$用來記載(n-1)路徑所經過的節點。
步驟 3	從 c 中的選出未到達原點之最短路徑 p，設 g 為路徑$\mathbf{p} = [\mathbf{f}, \mathbf{r_1}, \mathbf{r_2}, \dots, \mathbf{r_m}, \mathbf{g}]$的最後一個節點，考慮 g 的 n-1 個鄰居$\mathbf{gs} = \{\mathbf{g_1}, \mathbf{g_2}, \mathbf{g_3}, \dots \mathbf{g_{n-1}}\}$，另$\mathbf{gs'} = \mathbf{gs} \backslash \mathbf{U}$為出現在 gs 中但扣除出現在 U 中的節點。考慮$\mathbf{gs'}$中的節點，令$\mathbf{g'} = \mathbf{argmin}\{\mathbf{d}(\mathbf{h}) | \mathbf{h} \in \mathbf{gs'}\}$為$\mathbf{gs'}$集合中距離 id 最近之節點，若最近之節點有複數個，則隨機挑選之。我們將 g'連結到路徑 p 的後方，即重新令$\mathbf{p} = [\mathbf{f}, \mathbf{r_1}, \mathbf{r_2}, \dots, \mathbf{r_m}, \mathbf{g}, \mathbf{g'}]$。如果$\mathbf{g'} \neq \mathbf{id}$，我們將$\mathbf{g'}$加入 U 中以維護路徑的互斥性，即令$\mathbf{U} = \mathbf{U} \cup \{\mathbf{g'}\}$。
步驟 4	若$\mathbf{p} = [\mathbf{f}, \mathbf{r_1}, \mathbf{r_2}, \dots, \mathbf{r_m}, \mathbf{g}, \mathbf{g'}]$之路徑長度已超過 d(n)+1 即告失敗，不再尋找。
步驟 5	如果 c 中所有路徑皆已達原點，則輸出 c 的內容並結束本演算法。
步驟 6	重複步驟 3。

氣泡排序圖的節點與邊成長相當快，節點個數以階乘方式成長，程式為了計算寬直徑，通常需要在記憶體中先建立圖形，使用稀疏圖的僅紀錄連結的方式建立氣泡排序圖仍需耗用大量記憶體。因為氣泡排序圖的寬直徑臆測為$\mathbf{d}$(B_n)+1，所以演算法4將直接放棄尋找路徑的過程中距離已經超過d(B_n)+$\mathbf{1}$的節點，待全部節點皆尋找完畢後，去除已完成(即其長度在d(B_n)+$\mathbf{1}$的範圍內)的節點後，將尚未完成的節點再尋找一次，直到所有節點的長度皆落在d(B_n)+1內，加入本方法後將可使得寬直徑運算的耗費時間大幅降低。【表 5.1】為最短路徑演算法的實驗統計結果，我們發現了出以下幾點問題：：

(1) **記憶體使用量龐大**
實驗終將因為記憶體硬體限制上的問題而無法繼續進行。

(1) **建圖費時**
雖然預先建圖可得到明顯的指引，降低接下來尋路的計算時間，但隨著B_n的成長，建圖過程將會愈來越費時。

(2) **節點使用比例會越來越低**
在我們的臆測中，在計算完$D(B_n)$後會得到$(n-1)$條路徑結果，而每條路徑結果應會有$D(B_n)$以內個節點，換句話說我們總共會使用到的節點數$\leq D(B_n) \times (n-1)$。以B_{10}為例，我們最後所求得的路徑節點數少於$\frac{46\times9}{3628800} \approx$ 萬分之一，相較於最短路徑演算法預先將每一個節點逐一計算建立起來的指引圖，其比例相差懸殊。

綜觀以上幾點問題，最短路徑演算法到此遇到了瓶頸，我們思考如何能夠不預先建立指引圖，就能夠計算寬路徑將是一個關鍵議題。

5. 結論

在撰寫本文時我們已經將主臆測（即$D(B_n) = d(B_n) + 1$）的計算實驗驗證到B_{10}，目前數據顯示都符合臆測。以下將詳細說明本研究接下來的目標與目前結論。

【表5.1】，我們可看出最短路徑演算法在記憶體使用量上隨著B_n成長，而最短距離演算法的記憶體使用量始終都在10Mb左右。最短路徑演算法在計算B_{11}時所預估的花費時間為27天。

本研究主要解決兩個問題，一個是氣泡排序圖B_n上的寬路徑問題，一個是氣泡排序圖B_n上的寬直徑問題，前者要給定B_n上兩節點，希望找出較短的(n-1)寬路徑連接兩點，後者是給定B_n，計算出B_n的(n-1)-寬直徑。這兩個問題都很基本、很重要。前者關於網路路由的問題，後者關於網路本身的圖論性質。我們的目標即在發展客製的演算法尋找較短的寬路徑，並且利用於實證該臆測$\forall_{n\geq5} d_{n-1}(B_n) = d_1(B_n) + 1$。

表 5.1：最短路徑演算法實驗數據

網路	節點數	邊數	記憶體用量	建圖耗時	計算耗時
B4	24	72	27Mb	0.06 秒	1.2 秒
B5	120	480	27Mb	0.11 秒	1.7 秒
B6	720	4,320	29Mb	0.94 秒	2.7 秒
B7	5,040	32,400	35Mb	8.23 秒	25 秒
B8	40,320	282,240	59Mb	147 秒	4.7 分
B9	362,880	2,903,040	327Mb	45 分	1.7 小時
B10	3,628,800	36,259,200	524Mb	15 小時	22 小時
B11	39,916,800	399,168,000	估 2GiB	12 天(估)	15 天(估)

6. 參考文獻

[1] Sheldon B. Akers, Balakrishnan Krishnamurthy: A Group-Theoretic Model for Symmetric Interconnection Networks. IEEE Trans. Computers 38(4): 555-566 (1989).

[2] Zi-Tsan Chou, Chiun-Chieh Hsu, Jang-Ping Sheu: Bubblesort star graphs: a new interconnection network. ICPADS 1996: 41-48 (1996).

[3] M. Dietzfolbingor, S. Maclhavapeddy and I. H. Sudborough: Three disjoint path paradigms in star networks, in Proc. of the 3rd IEEE Symposium on Parallel and Distributed Processing (IEEE CS Press, 1991) 400-406.

[4] Robert Elsässer, Thomas Sauerwald: Broadcasting vs. Mixing and Information Dissemination on Cayley Graphs. STACS 2007: 163-174 (2007).

[5] S. Gao and D. F. Hsu: Short conatiners in Cayley graphs, DIMACS Technical Report 2001-18, May 2001, 15 pages.

[6] Marie-Claude Heydemann, Nausica Marlin, Stéphane Pérennes: Rotational Cayley Graphs on Transposition Generated Groups. Electronic Notes in Discrete Mathematics, 177~180 (2000).

[7] D. F. Hsu: On container width and length in graphs, groups, and networks, in IEICE Trans. on Fundamentals of Electronics, Communications, and Computer Science, v.E77-A, NoA (1994) 668-680.

[8] Tatsuya Iwasaki, Keiichi Kaneko: A routing algorithm of pairwise disjoint paths in a burnt pancake graph. SoICT 2011: 62-66 (2011).

[9] Keiichi Kaneko, Yasuto Suzuki: Node-to-Node Internally Disjoint Paths Problem in Bubble-Sort Graphs. PRDC 2004: 173-182 (2004).

[10] S. Labshmivarahan, J. Jwo and S. K. Dhall: Symmetry in interconnection networks based on cayley graphs of permutation group: a survey, in Parallel Comput. 19 (1993) 361-407.

[11] Shahram Latifi, Pradip K. Srimani: Transposition networks as a class of fault-tolerant robust networks. Computers, IEEE Transactions on (February 1996), 45 (2), pg. 230-238 (1996).

[12] Karl Menger: Zur allgemeinen Kurventheorie. Fund. Math. 10: 96–115 (1927).

[13] E. Oh and J. Chen: Strong fault-tolerance: parallel routing in star networks with faults, in J. of interconnection Networks, vA, No.1 (2003) 113-126.

[14] Yasuto Suzuki, Keiichi Kaneko: The Container Problem in Bubble-Sort Graphs. IEICE Transactions 91-D(4): 1003-1009 (2008).

[15] Yasuto Suzuki, Keiichi Kaneko: An algorithm for disjoint paths in bubble-sort graphs. Systems and Computers in Japan 37(12): 27-32 (2006).

[16] Yasuto Suzuki, Keiichi Kaneko, Mario Nakamori: Node-Disjoint Paths Algorithm in a Transposition Graph. IEICE Transactions 89-D(10): 2600-2605 (2006).

[17] Chi-Hsiang Yeh, Behrooz Parhami, Emmanouel A. Varvarigos: The Recursive Grid Layout Scheme for VLSI Layout of Hierarchical Networks. IPPS/SPDP 1999: 441-

[18] Chi-Hsiang Yeh, Emmanouel A. Varvarigos: Macro-Star Networks: Efficient Low-Degree Alternatives to Star Graphs. IEEE Trans. Parallel Distrib. Syst. 9(10): 987-1003 (1998)

[19] N-Queen Problem on Integer Sequnce. https://oeis.org/search?q=A000170