行政院國家科學委員會補助專題研究計畫
成果報告

※※※※※※※※※※※※※※※※※※※※※※※※※※
※ ※
※ 廣義的 shuffle-exchange 網路之訊息傳送演算法之設計 ※
※ ※
※※※※※※※※※※※※※※※※※※※※※※※※※※

計畫類別：■個別型計畫 □整合型計畫

計畫編號：NSC97－2628－M－009－06－MY3

執行期間：97 年 8 月 1 日至 100 年 10 月 31 日

計畫主持人：陳秋媛

本成果報告包括以下應繳交之附件：

□赴國外出差或研習心得報告一份

(另繳) 赴大陸地區出差或研習心得報告一份

(另繳) 出席國際學術會議心得報告及發表之論文各一份

□國際合作研究計畫國外研究報告書一份

執行單位：國立交通大學應用數學系

中 華 民 國 101 年 1 月 20 日

行政院國家科學委員會專題研究計畫成果報告

國科會專題研究計畫成果報告撰寫格式說明

Preparation of NSC Project Reports

計畫編號：NSC 97-2628-M-009-006-MY3
執行期限：97 年 8 月 1 日至 100 年 10 月 31 日

主持人：陳秋媛　國立交通大學應用數學系
cychen@mail.nctu.edu.tw
計畫參與人員：林武雄、藍國元、邱鈺傑、
吳思賢、黃思綸、羅健峰、
蔡詩妤、何恭毅、王奕倫、袁智龍

一、中文摘要

多級式連接網路在平行和分散式系統中有相當多的應用，本計畫之目的即在於探討多級式連接網路中的訊息傳送問題，並以廣義的 shuffle-exchange 網路之訊息傳送演算法之設計為主。本計劃為三年期計劃，在不計入會議論文以及已投稿但尚在審查中的論文下，本計劃執行過程中共順利投稿並被接受四篇期刊論文，其中兩篇是關於「廣義的 shuffle-exchange 網路」，一篇是關於「混合的弦環式網路」，一篇是關於「獨立擴張樹之點猜測問題」。

關鍵詞：連接網路、訊息傳送、shuffle-exchange 網路、點獨立擴張樹、混合的弦環式網路、直徑

Abstract

Multistage interconnection networks (MINs) have many applications in parallel and distributed computing systems. The purpose of this project is to study routing problems that arise in MINs, especially, those that arise in generalized shuffle-exchange networks. This project is a three-year project. During the three years, we have published four journal papers. Two of them consider routing problems in generalized shuffle-exchange networks; one of them considers the mixed chordal ring networks; and one of them discusses the vertex conjecture of the independent spanning trees problem.

Keywords: interconnection network, message routing, shuffle-exchange network, independent spanning trees, mixed chordal ring network, diameter

二、緣由與目的

在過去幾年中，本人所做的研究以「環式網路」為主；然而，由於「連接網路」及「無線網路」在現今的研究及實際應用中，佔有重要地位，故想藉此計劃開始「連接網路」以及「無線網路」方面之研究。

三、結果與討論

在 1991 年，Padmanabham 提出了廣義的 shuffle-exchange 網路，其目的在於打破 shuffle-exchange 網路中、網路節點數必須是 k 的指數之限制（其中 k 表示網路中的 switching elements 均為 of size $k \times k$），Padmanabham 並給出有效率且簡明的訊息傳送演算法。在 2003 年， Chen、Liu、Qiu 等三人將廣義的 shuffle-exchange 網路再推廣為連線均為雙向的；因此雙向的廣義的 shuffle-exchange 網路中即有正向、及反向兩網路。

在過去這三年中，我們已經完成、順利投稿、並被接受四篇期刊論文：其中第一篇論文被 Discrete Mathematics, Algorithms and Applications 接受；第二篇論文被 Information Processing Letters 接受；第三篇及第四篇論文被 Theoretical Computer Science 接受。以上四篇論文之電子檔均附於報告最後。今簡要敘述四篇論文之結果如下。

論文一：Efficient routing algorithms for generalized shuffle-exchange networks
（論文電子檔附於後）

在此論文中，我們探討雙向的廣義的 shuffle-exchange 網路中的正向及反向網路之訊息傳送所使用的 routing tags 的關係，我們導出關係式，並得出快速的訊息傳送演算法。

論文二：Improved upper and lower bounds on the optimization of mixed chordal ring networks
（論文電子檔附於後）

Chen 等人在文獻[3]中提出「混合的弦環式網路」。他們並指出：「混合的弦環式網路」與「雙環式網路」的硬體花費相同，但是「混合的弦環式網路」的直徑（communication delay）遠小於「雙環式網路」的直徑。不幸的是，我們發現文獻[3]中關於「混合的弦環式網路的直徑」的上界的推導是有錯誤的。在此論文中，我們指出文獻[3]的證明中的錯誤、提出更佳之直徑上界之結果、大大改進文獻[3]的結果。

論文三：All-to-all personalized exchange in generalized shuffle-exchange networks
（論文電子檔附於後）

全體對全體私有訊息傳送（all-to-all personalized exchange）出現在許多平行與分散式處理系統之應用。在文獻[15]中，Yang 以及 Wang 運用拉丁方陣的技巧，針對了具有 unique-path 以及 self-routable 性質的多級式連接網路，提出了時間複雜度為 $O(N)$ 的最佳全體對全體私人化交換演算法。所有在文獻[15]中被討論到的網路（包括 shuffle-exchange 網路），皆滿足 $N = 2^n$（N 表示多級式網路的輸入及輸出端的個數，n 是多級式網路的階級數）。在廣義的 shuffle-exchange 網路中，$2^{n-1} < N \le 2^n$，不再要求 $N = 2^n$。由於廣義的 shuffle-exchange 網路不具有 unique-path 性質，因此無法使用 Yang 以及 Wang 的演算法。我們的論文的目的即在於探討廣義的 shuffle-exchange 網路的全體對全體私有訊息傳送。

論文四：Constructing independent spanning trees for locally twisted cubes
（論文電子檔附於後）

在網路中使用多棵獨立擴張樹對於資料廣播有相當多的好處，例如：可以提高容錯以及頻寬等；因此，在各種的網路結構上，建造多棵獨立擴張樹，一直以來都被廣泛地研究。Zehavi 和 Itai 在文獻[16]中，對於建造多棵獨立擴張樹提出了兩個猜測。「點猜測」闡述的是：在一個點連通度為 n 的圖上，能以圖中任一點為樹根，產生 n 棵點獨立擴張樹；「邊猜測」闡述的是：在一個邊連通度為 n 的圖上，能以圖中任一點為樹根，產生 n 棵邊獨立擴張樹。在文獻[9] 中，Khuller 和 Schieber 證明了點猜測能涵蓋邊猜測。局部扭轉超立方體是超立方體的變形。最近，Hsieh 和 Tu 在文獻[6]中，提出了一個能在 n 維局部扭轉超立方體上，建造以 0 為樹根的 n 棵邊獨立擴張樹的演算法。因為局部扭轉超立方體不具點對稱性質，Hsieh 和 Tu 所提出的演算法無法解決局部扭轉超立方體的邊猜測。在

這篇論文中，我們提出了一個可以在局部扭轉超立方體上，以任一點為樹根，建構 n 棵點獨立擴張樹的演算法；我們的演算法證明了局部扭轉超立方體符合點猜測，當然，也證明了局部扭轉超立方體符合邊猜測。

四、計劃成果自評

本計劃之執行成果與預期成果非常相符。

五、參考文獻

[1] B. W. Arden and H. Lee, Analysis of chordal ring network, IEEE Trans. Computer. 30 (1981) 291-295.

[2] L. Barriere, J. F\'abrega, E. Simo and M. Zaragora, Fault-tolerant routing in chordal ring networks, Networks 36 (2000) 180-190

[3] S. K. Chen, F. K. Hwang and Y. C. Liu, Some combinatorial properties of mixed chordal rings," J. Inter. Networks 4 (2003) 3-16.

[4] C. Y. Chen and J. K. Luo, An efficient tag-based routing algorithm for the backward network of a bidirectional general shuffle-exchange network, IEEE Commun. Lett. 10, no. 4 (2006) 296-298.

[5] Y. Cheng and F. K. Hwang, Diameters of weighted double loop networks, J. Algorithms 9 (1988) 401-410.

[6] S. Y. Hsieh and C. J. Tu, Constructing edge-disjoint spanning trees in locally twisted cubes, Theoretical Computer Science 410 (2009) 8-10.

[7] J. L. Hurink and T. Nieberg, Approximating minimum independent dominating sets in wireless networks, Information Processing Letters, 109, no. 2 (2008) 155-160.

[8] F. K. Hwang, A complementary survey on double-loop networks, Theoret. Comput. Sci. 263 (2001) 211-229.

[9] S. Khuller, B. Schieber, On Insependent spanning-trees, Information Processing Letters 42 (1992) 321-323.

[10] V. W. Liu, C. Y. Chen, and R. B. Chen, Optimal all-to-all personalized exchange in d-nary banyan multistage interconnection networks, Journal of Combinatorial Optimization, 14 (2007) 131-142.

[11] A. Massini, All-to-all personalized communication on multistage interconnection networks, Discrete Appl. Math. 128, no. 2 (2003) 435-446.

[12] K. Padmanabham, Design and analysis of even-sized binary shuffle-exchange networks for multiprocessors, IEEE Trans. Parallel Distrib. Syst. 2, no. 4 (1991) 385-397.

[13] C. S. Raghavendra and J. A. Sylvester, A survey of multi-connected loop topologies for local computer networks, Comput. Netw. ISDN Syst. 11 (1986) 29-42.

[14] Y. Tscha and K. H. Lee, Yet another result on multi-log N networks, IEEE Trans. Commu. 47 (1999) 1425–1431.

[15] Y. Yang, J. Wang, Optimal all-to-all personalized exchange in self-routable multistage networks, IEEE Trans. Parallel Distrib. Syst. 11, no. 3 (2000) 261-274.

[16] A. Zwhavi and A. Itai, Three tree-paths, Journal of Graph Theory 13 (1989) 175-188.

Discrete Mathematics, Algorithms and Applications
Vol. 1, No. 2 (2009) 267–281

World Scientific
www.worldscientific.com

EFFICIENT ROUTING ALGORITHMS FOR GENERALIZED SHUFFLE-EXCHANGE NETWORKS*

JAMES K. LAN, WELL Y. CHOU and CHIUYUAN CHEN†

Department of Applied Mathematics
National Chiao Tung University
Hsinchu 300, Taiwan

Accepted 13 February 2009

The shuffle-exchange network has been proposed as a popular architecture for multistage interconnection networks. In 1991, Padmanabhan introduced the generalized shuffle-exchange network (GSEN) and proposed an efficient routing algorithm. Later, Chen *et al.* further enhanced the GSEN with bidirectional links and proposed the bidirectional GSEN (BGSEN). A BGSEN consists of the forward and the backward network. Based on the idea of inversely using the control tag generated by Padmanabhan's algorithm, Chen *et al.* proposed a routing algorithm for the backward network. Recently, Chen and Lou also proposed a routing algorithm for the backward network. It should be noted, however, that Padmanabhan's algorithm is actually an explicit formula for computing the control tag for routing and takes only $O(1)$ time. Neither the algorithm of Chen *et al.* nor the algorithm of Chen and Lou provides an explicit formula for computing the control tag for routing and both algorithms take at least $\Omega(n)$ time, where $n+1$ is the number of stages in the BGSEN. This paper attempts to propose an explicit formula for computing the control tag for routing in the backward network. We will demonstrate how this formula greatly simplifies the computation process and how it leads to efficient routing algorithms. In particular, an $O(1)$-time one-to-one routing algorithm and an efficient routing-table construction algorithm have been proposed.

Keywords: Multistage interconnection network; routing; algorithm; shuffle-exchange network; omega network.

Mathematics Subject Classification 2000: 68M10, 68W40, 68M20

1. Introduction

Throughout this paper, N denotes the number of input (output) terminals of a given multistage interconnection network (MIN) and all the switching elements of an MIN are assumed to be of size $k \times k$. See Fig. 1 for an example. It is well-known that $k|N$. The *k-ary shuffle operation* on N terminals is the permutation π

*This research was partially supported by the National Science Council of the Republic of China under the grants grant NSC97-2628-M-009-006-MY3.
†The corresponding author, e-mail: cychen@mail.nctu.edu.tw

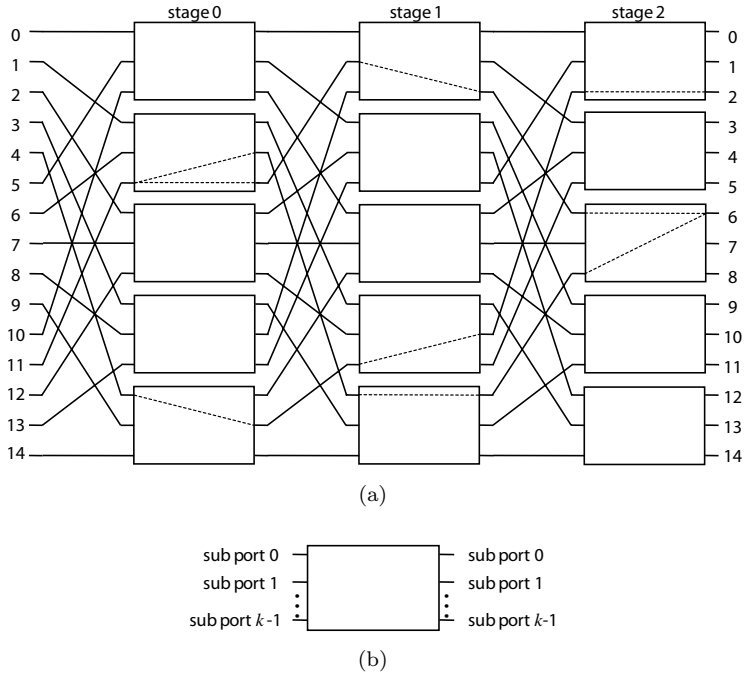


Fig. 1. (a) BGSEN(3,5,3). It is an MIN with $N = 15$ input (output) terminals and all the switching elements are of size 3×3. One (4,2)-path and two (11,6)-paths are also shown in this figure. (b) A $k \times k$ switching element and its sub ports.

defined by

$$\pi(i) = \left(ki + \left\lfloor \frac{ki}{N} \right\rfloor \right) \bmod N, \quad 0 \le i \le N-1.$$

The *shuffle-exchange network* is an MIN with $N = k^d$ input (output) terminals such that each stage consists of the k-ary shuffle on N terminals followed by N/k switching elements. The shuffle-exchange network has been proposed as a popular architecture for MINs; see [3, 4, 6, 7, 9]. When the number of stages is exactly d, the shuffle-exchange network is also called the *omega network*; see [7].

Padmanabhan [8] first generalized the shuffle-exchange network to allow $N \neq k^d$ and introduced the generalized shuffle-exchange network. More precisely, the *generalized shuffle-exchange network* (GSEN) is a network that has N input (output) terminals and has exactly $\lceil \log_k N \rceil$ stages such that each stage consists of the k-ary shuffle on N terminals followed by N/k switching elements; see Fig. 1(a) for an example. Note that Padmanabhan used the word "general" instead of "generalized." Also note that an MIN must have at least $\lceil \log_k N \rceil$ stages to ensure that every input terminal can get to every output terminal; Padmanabhan used this minimum number of stages to construct a GSEN.

In an MIN, a path from a source (of a routing request) to a destination can be described by a sequence of labels that label the successive links on this path. Such a sequence is called a *control tag* [8] or *tag* [1] or *path descriptor* [5]. The control tag may be used as a header for routing a message: each successive switching element uses the first element of the sequence to route the message, and then discards it. For example, in Fig. 1(a), input 4 can get to output 2 by using the control tag $14 = (\mathbf{1\ 1\ 2})_{\mathbf{3}}$, which means that the routing is via sub port 1 at stage 0, sub port 1 at stage 1, and sub port 2 at stage 2. In [8], Padmanabhan also proposed an elegant tag-based routing algorithm (call it P-algorithm for convenience) for the GSEN; this algorithm is actually an explicit formula for computing the control tag.

In [1], Chen *et al.* further enhanced the GSEN with bidirectional links. Their reason for enhancing it is that although unidirectional links are extensively used, bidirectional links can also be widely applicable as suggested in [3]. A bidirectional generalized shuffle-exchange network (BGSEN) can be divided into two dependent networks: the *forward network* and the *backward network*. The forward network is from the left-hand side of the network to the right-hand side. A routing in the forward network is from left to right. The control tags used in the forward network are called the *forward control tags*. The backward network is from the right-hand side of the network to the left-hand side. A routing in the backward network is from right to left. The control tags used in the backward network are called the *backward control tags*.

Since the forward network is the GSEN, Padmanabhan's elegant tag-based routing algorithm can thus be applied on it. As for the backward network, Chen *et al.* [1] proposed a tag-based routing algorithm (hereafter refers to as CLQ-algorithm) by using the forward control tag inversely. More precisely, CLQ-algorithm first runs P-algorithm to obtain the forward control tag. It then obtains the port sequence used by the forward control tag. Finally, it runs another procedure to convert the port sequence into the backward control tag. Recently, Chen and Lou [2] showed that the backward network has a wonderful property: for each destination i, there are two backward control tags associated with i such that every source j can get to i by using one of the two backward control tags. They also proposed an efficient algorithm (hereafter refers to as CL-algorithm) for computing the two associated backward control tags.

It should be noted that P-algorithm is actually an explicit formula for computing the control tag (see Sec. 3) and takes only $O(1)$ time. Therefore, there is an explicit formula for computing the forward control tag. There is, however, neither [1] nor [2] that provides an explicit formula for computing the backward control tag. In [1] and [2], an algorithm has to be executed for obtaining the backward control tag and the algorithm takes at least $\Omega(n)$ time, where $n+1$ is the number of stages in the network. In [1], Chen *et al.* mentioned that the computation of the backward control tag is still more complex than the forward control tag, and that they asked whether there is an easy algorithm or a direct relation between the forward control tag and the backward control tag.

The purpose of this paper is to provide an *affirmative* answer to the question of Chen *et al.* and to propose efficient routing algorithms for the BGSEN. In particular, we propose a formula for characterizing the relation between the forward control tag F and the backward control tag B. This formula will be termed the forward backward control tag (FBCT) formula. We will use FBCT formula to obtain an explicit formula for computing the backward control tag B. This explicit formula for B greatly simplifies the process of computing the backward control tag B and leads to an $O(1)$-time one-to-one routing algorithm for the backward network, the simplified version of CL-algorithm, and an efficient routing-table construction algorithm for the backward network.

This paper is organized as follows. Section 2 gives definitions and conventions that will be used throughout this paper; Sec. 3 briefly reviews P-algorithm, CLQ-algorithm and CL-algorithm; Sec. 4 contains our main result and its listed applications; Sec. 5 gives an efficient algorithm for constructing a routing table for the backward network; concluding remarks are given in the last section.

2. Definitions and Conventions

We will follow most of the notations and terminologies used in [1] and [8]. In particular, the number of stages is denoted by $n+1$ instead of n. BGSEN$(k, r, n+1)$ denotes a BGSEN in which the switching elements are aligned in $n+1$ stages, labelled $0, 1, \ldots, n$, with stage 0 to be the leftmost stage; each stage consists of r switching elements, labelled $0, 1, \ldots, r-1$; and each switching element is a $k \times k$ bidirectional crossbar. For example, the network in Fig. 1(a) is BGSEN(3,5,3). In BGSEN$(k, r, n+1)$, there are a total of $N = k \times r$ ports on each side of a stage, labelled $0, 1, \ldots, N-1$. The parameters k, r, and n satisfy the following equation: $\lceil \log_k (k \cdot r) \rceil = \lceil \log_k N \rceil = n+1$. Throughout this paper, let

$$N = k^n + M, \quad \text{with } 0 < M \leq k^{n+1} - k^n. \tag{2.1}$$

The following conventions are used in this paper. No matter the network considered is the forward or the backward network, stage 0 means the leftmost stage. The switching elements in a stage are considered cyclic; that is, the switching element labelled 0 is considered to be the successive switching element of the switching element labelled $r-1$. Also, terminal i is assumed on the left-hand side of the network and terminal j is assumed on the right-hand side of the network. An $(\overrightarrow{i,j})$-*request* ($(\overleftarrow{i,j})$-*request*) denotes a routing request or a request for sending a message from i to j (from j to i). An (i,j)-*path* denotes a path between i and j.

Note that an $(\overrightarrow{i,j})$-request or an $(\overleftarrow{i,j})$-request can be fulfilled by an (i,j)-path. Moreover, an (i,j)-path can be characterized by its *port sequence*, which is the sequence of ports $(R_{-1}, R_0, R_1, \ldots, R_n)$ passed by this path such that R_{-1} is defined to be $R_{-1} = i$ and R_ℓ is the port to the *right-hand* side of the switching element at stage ℓ on this path. Clearly, $R_n = j$. Another way to define the port sequence is to use the ports on the *left-hand* side of the switching element at each stage; but in this paper, we will use the former definition.

All the previous routing algorithms of GSENs and BGSENs are tag-based. A tag-based routing algorithm sets up a path from the source to the destination by using a control tag T. Each digit of the k-ary representation of T is used to control one switching element in the path. More precisely, suppose

$$T = (t_0 t_1 \cdots t_n)_{\mathbf{k}},$$

then digit t_ℓ controls the switching element at stage ℓ in the path.

In a BGSEN, there are two types of control tags: the forward control tag and the backward control tag. The former is comprised of the sub ports on the right-hand side of the switching elements on the routing path, while the latter is comprised of the sub ports on the left-hand side of the switching elements on the routing path. Let F denote a forward control tag and assume

$$F = (f_0 f_1 \cdots f_n)_{\mathbf{k}}.$$

Let B denote a backward control tag and assume

$$B = (b_0 b_1 \cdots b_n)_{\mathbf{k}}.$$

Clearly, F and B satisfy

$$F = f_0 k^n + f_1 k^{n-1} + \cdots + f_{n-1} k + f_n k^0 \tag{2.2}$$

and

$$B = b_0 k^n + b_1 k^{n-1} + \cdots + b_{n-1} k + b_n k^0. \tag{2.3}$$

F is used in the order $f_0\ f_1\ \cdots\ f_{n-1}\ f_n$ and B is used in the order $b_n\ b_{n-1}\ \cdots\ b_1\ b_0$. Note that

$$0 \le f_i, b_i < k \quad \text{for all } i, 0 \le i \le n, \text{ and } 0 \le F, B < k^{n+1}. \tag{2.4}$$

Take BGSEN(3,5,3) in Fig. 1(a) for an example. The $(\overrightarrow{4,2})$-request and also the $(\overleftarrow{4,2})$-request can be fulfilled by using the $(4,2)$-path shown in this figure. The port sequence of this $(4,2)$-path is $(4,\ 13,\ 10,\ 2)$. So the $(\overrightarrow{4,2})$-request can be routed by using the forward control tag $F = 14 = (\mathbf{1\ 1\ 2})_{\mathbf{3}}$ and the $(\overleftarrow{4,2})$-request can be routed by using the backward control tag $B = 8 = (\mathbf{0\ 2\ 2})_{\mathbf{3}}$.

3. Previous Routing Algorithms

We now briefly review previous routing algorithms of GSENs and BGSENs. P-algorithm was stated in a theorem in [8] and it provides an explicit formula for computing the control tag of a GSEN and also for computing the forward control tag of BGSEN$(k, r, n+1)$.

Theorem 3.1 [8]. *Any i, $0 \le i < N$, can set up a path to a j, $0 \le j < N$, by using the control tag*

$$F_1 = (j + k \cdot M \cdot i) \mod N. \tag{3.1}$$

In addition, other control tags (and paths) may be available, specified by

$$F_p = F_1 + (p-1)N \quad \text{if } F_p < k^{n+1},\ 1 < p \le k. \tag{3.2}$$

CLQ-algorithm is based on the idea of inversely using the forward control tag generated by P-algorithm. See the following.

CLQ-algorithm
Input: i on the left-hand side and j on the right-hand side of BGSEN$(k, r, n+1)$.
Output: A backward control tag B that can send a message from j to i.

1. Use P-algorithm to obtain a forward control tag $F = (f_0 f_1 \cdots f_n)_{\mathbf{k}}$ that can send a message from i to j.
2. Get the port sequence R_ℓ $(-1 \le \ell \le n)$ in the path based on F by:

$$R_\ell = \begin{cases} i & \text{if } \ell = -1, \\ (k \cdot R_{\ell-1}) \mod N + f_\ell & \text{if } 0 \le \ell \le n. \end{cases}$$

3. Use R_ℓ $(0 \le \ell \le n)$ to obtain the backward control tag $B = (b_0 b_1 \cdots b_n)_{\mathbf{k}}$ by:

$$b_\ell = \left\lfloor \frac{k \cdot R_{\ell-1}}{N} \right\rfloor \quad \text{for } 0 \le \ell \le n.$$

Recently, Chen and Lou [2] showed that the backward network of a BGSEN has a wonderful property: for each destination i, there are two backward control tags associated with i such that every source j can get to i by using one of the two tags. See the following.

CL-algorithm
Input: i on the left-hand side of BGSEN$(k, r, n+1)$.
Output: The value $v(i)$ and the two backward control tags $B = (b_0\ b_1\ \cdots\ b_n)_{\mathbf{k}}$ and $B' = (b'_0\ b'_1\ \cdots\ b'_n)_{\mathbf{k}}$ associated with i.

1. **for** $\ell = 0$ **to** n **do** $C_\ell = (i \cdot k^\ell) \mod r$ **end for**
2. $v(i) = C_n \cdot k$
3. **if** $(r - C_{n-1}) \cdot k \ge r$
   then
   $D_n = 1$; **for** $\ell = 0$ **to** $n-1$ **do** $D_\ell = 0$ **end for**;
   else
   for $\ell = 0$ **to** n **do if** $C_\ell + k^\ell > r$ **then** $D_\ell = 1$ **else** $D_\ell = 0$ **end if**
   end for
   end if
4. $b_0 = \left\lfloor \frac{i}{r} \right\rfloor$; **for** $\ell = 1$ **to** n **do** $b_\ell = \left\lfloor \frac{k \cdot C_{\ell-1}}{r} \right\rfloor$ **end for**
5. **for** $\ell = 0$ **to** n **do** $b'_\ell = (b_\ell + D_\ell) \mod k$ **end for**

Chen and Lou proved the following theorem.

Theorem 3.2 [2]. *If $v(i) \le j < N$, then j can get to i by using B; if $0 \le j < v(i)$, then j can get to i by using B'.*

4. The Main Result

In [1], Chen *et al.* pointed out that the computation of the backward control tag B is still more complex than the forward control tag F. They also asked whether there is an easy algorithm or a direct relation between F and B. In this section, we give an affirmative answer to this problem.

This section is organized as follows. Subsection 4.1 gives a formula for the relation between F and B; Subsection 4.2 gives applications of the formula.

4.1. *A formula for the relation between F and B*

Consider BGSEN$(k, r, n+1)$ and an (i, j)-path $\mathcal{P}$ in it. Let $(i = R_{-1}, R_0, R_1, \ldots, R_n = j)$ be the port sequence of $\mathcal{P}$. Let F be the forward control tag that fulfills an $\overrightarrow{(i, j)}$-request by using $\mathcal{P}$ and let B be the backward control tag that fulfills an $\overleftarrow{(i, j)}$-request by using $\mathcal{P}$. See Fig. 2 for an illustration.

Note that when sending a message from i to j along $\mathcal{P}$, the message enters a switching element at stage ℓ via sub port b_ℓ and leaves the switching element via sub port f_ℓ. Conversely, when sending a message from j to i along $\mathcal{P}$, the message enters a switching element at stage ℓ via sub port f_ℓ and leaves the switching element via sub port b_ℓ. The following result has been obtained in [1].

Lemma 4.1 [1]. *For* $0 \leq \ell \leq n$,

(a) $R_\ell = (k \cdot R_{\ell-1} + f_\ell) \mod N$.
(b) $b_\ell = \lfloor \frac{k \cdot R_{\ell-1}}{N} \rfloor = \lfloor \frac{R_{\ell-1}}{r} \rfloor$.

The formula stated in the following theorem is our main result. It characterizes the relation between F and B. We call this formula the *forward backward control tag (FBCT) formula*.

Theorem 4.2. ***(FBCT formula)***

$$k^{n+1} \cdot i + F = B \cdot N + j.$$

Proof. We first prove that

$$k \cdot R_{\ell-1} + f_\ell = b_\ell \cdot N + R_\ell, \quad 0 \leq \ell \leq n. \tag{4.1}$$

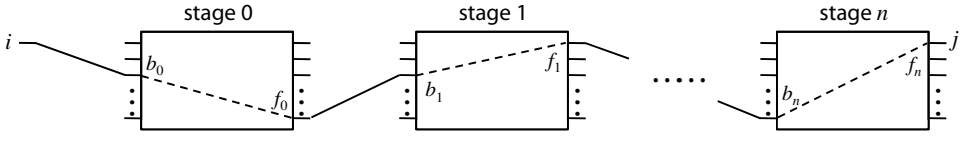


Fig. 2. An (i, j)-path and the corresponding sub ports.

By Lemma 4.1(a), R_ℓ is the remainder of $k \cdot R_{\ell-1} + f_\ell$ divided by N. Also, by (2.4) and Lemma 4.1(b),

$$\left\lfloor \frac{k \cdot R_{\ell-1} + f_\ell}{N} \right\rfloor = \left\lfloor \frac{R_{\ell-1} + \frac{f_\ell}{k}}{r} \right\rfloor = \left\lfloor \frac{R_{\ell-1}}{r} \right\rfloor = b_\ell.$$

This equation says that b_ℓ is the quotient of $k \cdot R_{\ell-1} + f_\ell$ divided by N. Hence we have (4.1). By (4.1), we have

$$\begin{aligned}
(k \cdot R_{-1} + f_0) \cdot k^n &= (b_0 \cdot N + R_0) \cdot k^n \\
(k \cdot R_0 + f_1) \cdot k^{n-1} &= (b_1 \cdot N + R_1) \cdot k^{n-1} \\
(k \cdot R_1 + f_2) \cdot k^{n-2} &= (b_2 \cdot N + R_2) \cdot k^{n-2} \\
\vdots \qquad & \qquad \vdots \\
(k \cdot R_{n-1} + f_n) \cdot k^0 &= (b_n \cdot N + R_n) \cdot k^0.
\end{aligned}$$

Since the summation of the left-hand sides of these equations equals to the summation of the right-hand sides of these equations, we have

$$k^{n+1} \cdot R_{-1} + \underbrace{f_0 \cdot k^n + \cdots + f_n \cdot k^0}_{\text{by (2.2), this is } F} = (\underbrace{b_0 \cdot k^n + \cdots + b_n \cdot k^0}_{\text{by (2.3), this is } B}) \cdot N + R_n.$$

Consequently, $k^{n+1} \cdot i + F = B \cdot N + j$. □

4.2. *Applications of FBCT formula*

In this subsection, we show various applications of FBCT formula. We first use it to obtain the destination j of a message sending from a source i when the forward control tag is F.

Corollary 4.3. *Given i and F, the destination j is given by*

$$j = (k^{n+1} \cdot i + F) \mod N.$$

Proof. This corollary follows from FBCT formula. □

Define

$$v(i) = k^{n+1} \cdot i \mod N.$$

Note that the definition of $v(i)$ is identical to the one used in CL-algorithm. The following corollary follows immediately from Corollary 4.3 and its proof is omitted.

Lemma 4.4. *Given i and $F = 0$, the destination j is $v(i)$.*

Lemma 4.4 indicates that i can get to $j = v(i)$ by *always* going to sub port 0 of the switching element at each stage. Recall that $0 \le F < k^{n+1}$ and $0 \le j < N$. In the remaining part of this paper, $F = 0$ is considered to be the successive F of $F = k^{n+1} - 1$, $j = 0$ is considered to be the successive j of $j = N - 1$, and

$j = N - 1$ is considered to be the preceding j of $j = 0$. Therefore, if $F = k^{n+1} - 1$, then $F + 1 = 0$; if $j = N - 1$, then $j + 1 = 0$; if $j = 0$, then $j - 1 = N - 1$. The following result follows immediately from Corollary 4.3 and its proof is omitted.

Lemma 4.5. *Given i, j increases by 1 whenever F increases by 1. Moreover, consecutive F's will make i get to consecutive j's.*

By Lemma 4.3, we already know which j will receive a message from i when the forward control tag is F. Conversely, how can j reply a message to i? The following corollary provides the answer.

Corollary 4.6.

$$B = \left\lfloor \frac{k^{n+1} \cdot i + F}{N} \right\rfloor.$$

Proof. This corollary follows from FBCT formula and the fact that $0 \leq j < N$. □

This corollary provides a direct conversion from F into its corresponding B. Consider BGSEN(3,5,3) in Fig. 1(a). Suppose $i = 4$ and $F = 14$. Then by Corollary 4.3, $j = 2$ will receive the message from i. On the contrary, by Corollary 4.6, $j = 2$ can reply a message to $i = 4$ by using $B = 8 = (\mathbf{0\ 2\ 2})_{\mathbf{3}}$.

We now use FBCT formula to obtain the destination i of a message sending, in the backward network, from a source j when the backward control tag is B.

Corollary 4.7. *Given j and B, the destination i is given by*

$$i = \left\lfloor \frac{B \cdot N + j}{k^{n+1}} \right\rfloor.$$

Proof. This follows from FBCT formula and the fact that $0 \leq F < k^{n+1}$. □

FBCT formula also suggests the following bounds for B.

Corollary 4.8. *For an $(\overleftarrow{i, j})$-request, B satisfies*

$$\frac{k^{n+1} \cdot i - j}{N} \leq B < \frac{k^{n+1} \cdot (i+1) - j}{N}. \tag{4.2}$$

Moreover, any integer B satisfying (4.2) can serve as a backward control tag for an $(\overleftarrow{i, j})$-request.

Proof. (4.2) follows from FBCT formula and the fact that $0 \leq F < k^{n+1}$. Suppose B is an integer satisfying (4.2). Then $i \leq \frac{B \cdot N + j}{k^{n+1}} < i + 1$. By Corollary 4.7, j can get to i by using B. □

The following theorem follows from Corollary 4.8 and its proof is omitted.

Theorem 4.9. *Any j, $0 \le j < N$, in the backward network of BGSEN$(k, r, n+1)$ can set up a path to i, $0 \le i < N$, by using the backward control tag*

$$B_1 = \left\lceil \frac{k^{n+1} \cdot i - j}{N} \right\rceil .$$

In addition, other backward control tags (and paths) may be available, specified by

$$B_p = B_1 + (p-1) \quad \text{if } B_p < \frac{k^{n+1} \cdot (i+1) - j}{N},\ 1 < p \le k.$$

Take the backward network in Fig. 1(a) as an example. By Theorem 4.9, $j = 6$ can get to $i = 11$ by using $B_1 = 20 = (\mathbf{2\ 0\ 2})_{\mathbf{3}}$ and $B_2 = B_1 + 1 = 21 = (\mathbf{2\ 1\ 0})_{\mathbf{3}}$; the two routing paths are depicted in Fig. 1(a). Do notice that Theorem 4.9 provides an explicit formula for computing the backward control tag B; see the following algorithm. The correctness of this algorithm follows from Theorem 4.9 and it takes only $O(1)$ time.

ONE-TO-ONE-ROUTING
Input: i on the left-hand side and j on the right-hand side of BGSEN$(k, r, n+1)$.
Output: A backward control tag B, which can be used to send a message from j to i.
1. $B = \left\lceil \frac{k^{n+1} \cdot i - j}{N} \right\rceil$.

Recall that CL-algorithm obtains the two backward control tags B and B' associated with i such that every j can get to i by using one of B and B'. We now propose a theorem, which is a special case of Theorem 5.2 in Sec. 5, that gives explicit formulae for computing B and B'.

Theorem 4.10. *Given an $(\overleftarrow{i, j})$-request,*

(a) *if $v(i) \le j < N$, then j can get to i by using $B = \left\lfloor \frac{k^{n+1} \cdot i}{N} \right\rfloor$;*
(b) *if $0 \le j < v(i)$, then j can get to i by using $B' = \left\lfloor \frac{k^{n+1} \cdot i}{N} \right\rfloor + 1$.*

Proof. Set $q = \left\lfloor \frac{k^{n+1} \cdot i}{N} \right\rfloor$ for easy writing. By the definition of $v(i)$, we have

$$k^{n+1} \cdot i - v(i) = q \cdot N.$$

We first prove (a). By Lemmas 4.4 and 4.5, setting $0 \le F < N - v(i)$ will make i get to j, where $v(i) \le j < N$. So

$$q \le \frac{k^{n+1} \cdot i}{N} \le \frac{k^{n+1} \cdot i + F}{N} < \frac{k^{n+1} \cdot i + N - v(i)}{N} = q + 1.$$

By Corollary 4.6, we have $B = q = \left\lfloor \frac{k^{n+1} \cdot i}{N} \right\rfloor$.

We now prove (b). Again, by Lemmas 4.4 and 4.5, setting $N - v(i) \leq F < N$ will make i get to j, where $0 \leq j < v(i)$. So

$$q + 1 = \frac{k^{n+1} \cdot i + N - v(i)}{N} \leq \frac{k^{n+1} \cdot i + F}{N} < \frac{k^{n+1} \cdot i}{N} + 1 \leq q + 2.$$

Again, by Corollary 4.6, we have $B' = q + 1 = \left\lfloor \frac{k^{n+1} \cdot i}{N} \right\rfloor + 1$. □

Theorem 4.10 provides explicit formulae for computing the two backward control tags B and B' associated with i; see the following algorithm.

COMPUTE-TWO-TAGS

Input: i on the left-hand side of BGSEN$(k, r, n+1)$.

Output: The value $v(i)$ and the two backward control tags B and B' associated with i such that every j can get to i by using one of them.

1. $v(i) = k^{n+1} \cdot i \mod N$
2. $B = \left\lfloor \frac{k^{n+1} \cdot i}{N} \right\rfloor$ and $B' = \left\lfloor \frac{k^{n+1} \cdot i}{N} \right\rfloor + 1$

The correctness of this algorithm follows from Theorem 4.10. It takes only $O(1)$ time and it greatly simplifies CL-algorithm.

5. Constructing Routing Tables for the Backward Network

In [2], Chen and Lou had used CL-algorithm to construct a routing table for the backward network. However, Chen and Lou's routing table provides *only one* backward control tag for an $(\overleftarrow{i, j})$-request. By Theorem 4.9, we know that an $(\overleftarrow{i, j})$-request in BGSEN$(k, r, n+1)$ may have up to k backward control tags. The purpose of this section is to construct a routing table so that *all* the backward control tags for an $(\overleftarrow{i, j})$-request are in this table. Such a routing table can be used for fault-tolerance.

For convenience, set

$$\omega = \left\lceil \frac{k^{n+1}}{N} \right\rceil \quad \text{and} \quad \mu = k^{n+1} \mod N.$$

Lemma 5.1. *For a fixed i, setting $F = 0, 1, \ldots, k^{n+1} - 1$ will make i get to k^{n+1} consecutive j's (not necessarily distinct) and these j's are*:

$$v(i), v(i) + 1, \ldots, v(i + 1) - 1.$$

Proof. This lemma follows from Lemma 4.4, Lemma 4.5 and the fact that $(v(i) + k^{n+1} - 1) \mod N = v(i + 1) - 1$. □

Again, take BGSEN(3,5,3) in Fig. 1(a) for an illustration. By Lemma 5.1, setting $F = 0, 1, \ldots, k^{n+1} - 1$ will make:

$i = 0$ get to these j's: $0, \ldots, 14, 0, \ldots, 11$;
$i = 1$ get to these j's: $12, \ldots, 14, 0, \ldots, 14, 0, \ldots, 8$;

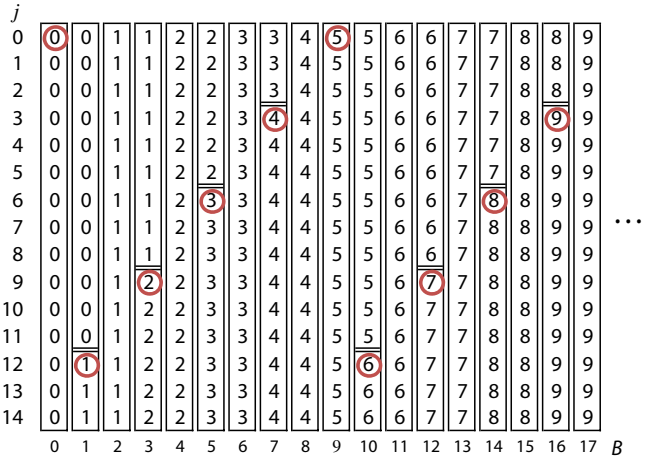


Fig. 3. Setting $F = 0, 1, \ldots, k^{n+1} - 1$ in the forward network of BGSEN(3,5,3) makes an i get to 27 consecutive j's; this figure illustrates the case of $i = 0, 1, \ldots, 9$. An i enclosed by a circle indicates that the corresponding j is $v(i)$. The bottom row indicates the backward control tag B that can make j get to i.

$i = 2$ get to these j's: $9, \ldots, 14, 0, \ldots, 14, 0, \ldots, 5$;
$i = 3$ get to these j's: $6, \ldots, 14, 0, \ldots, 14, 0, \ldots, 2$;
$i = 4$ get to these j's: $3, \ldots, 14, 0, \ldots, 14$;

and make $i = 5, 6, \ldots, 9$ get to the same pattern of j's (see Fig. 3 for an illustration), and make $i = 10, 11, \ldots, 14$ get to the same pattern of j's.

See the diagram in Fig. 3 for the following arguments. For a fixed i, by Corollary 4.6, every N consecutive F's would increase B by 1. Consequently, by Lemma 4.5, every N consecutive j's would increase B by 1. Since $v(0) = 0$, by Corollary 4.6, the first column in this diagram has $B = 0$, the second column has $B = 1$, the third column has $B = 2$, and so on. For each i, CL-algorithm considers only the first two columns (i.e., the first two B's) in this diagram. However, an i may have up to ω or $\omega + 1$ columns (i.e. up to ω or $\omega + 1$ B's) associated with it.

The following theorem can be proven by arguments used in the previous paragraph and we omit its proof.

Theorem 5.2. *Given an i,*

(a) *if $v(i)+\mu \leq N$, then there are ω backward control tags associated with i, specified by $B_1 = \left\lfloor \frac{k^{n+1} \cdot i}{N} \right\rfloor$ and $B_p = B_1 + (p-1),\ 1 < p \leq \omega$;*
(b) *if $v(i) + \mu > N$, then there are $\omega + 1$ backward control tags associated with i, specified by $B_1 = \left\lfloor \frac{k^{n+1} \cdot i}{N} \right\rfloor$ and $B_p = B_1 + (p-1),\ 1 < p \leq \omega + 1$.*

The correctness of the following algorithm follows from Theorem 5.2. This algorithm takes only $O(\omega)$ time (note that $1 \leq \omega \leq k$).

Table 1. The routing table for the backward network of the BGSEN shown in Fig. 1(a).

i	$v(i)$	B_1	B_2	B_3	i	$v(i)$	B_1	B_2	B_3
0	0	0	1	—	8	6	14	15	16
1	12	1	2	3	9	3	16	17	—
2	9	3	4	5	10	0	18	19	—
3	6	5	6	7	11	12	19	20	21
4	3	7	8	—	12	9	21	22	23
5	0	9	10	—	13	6	23	24	25
6	12	10	11	12	14	3	25	26	—
7	9	12	13	14					

COMPUTE-ALL-TAGS

Input: i on the left-hand side of BGSEN$(k, r, n+1)$.

Output: The value $v(i)$ and the ω or $\omega+1$ backward control tags $B_1, B_2, \ldots, B_\omega$ $(B_{\omega+1})$ associated with i.

1. $v(i) = k^{n+1} \cdot i \mod N$
2. $B_1 = \left\lfloor \frac{k^{n+1} \cdot i}{N} \right\rfloor$
3. **if** $v(i) + \mu \leq N$ **then** let $t = \omega$ **else** let $t = \omega + 1$
4. **for** $p = 2$ **to** t **do** $B_p = B_1 + (p-1)$ **end for**

Table 1 is a routing table generated by algorithm COMPUTE-ALL-TAGS. In this table, "—" means not available. The size of the routing table generated by algorithm COMPUTE-ALL-TAGS is only $O(N \times \omega)$ and is much smaller than that of a regular routing table, which is $N^2 \times k$.

The routing table generated by algorithm COMPUTE-ALL-TAGS can be used as follows.

Theorem 5.3. *An* $(\overleftarrow{i, j})$*-request can be fulfilled as follows.* (*See Fig.* 4.)

(a) *If* $v(i) + \mu < N$, *then* j *can get to* i *by using* $B_2, B_3, \ldots, B_\omega$ *if* $0 \leq j < v(i)$; $B_1, B_2, \ldots, B_\omega$ *if* $v(i) \leq j < v(i+1)$; $B_1, B_2, \ldots, B_{\omega-1}$ *if* $v(i+1) \leq j < N$.

(b) *If* $v(i) + \mu = N$, *then* j *can get to* i *by using* $B_2, B_3, \ldots, B_\omega$ *if* $0 \leq j < v(i)$; $B_1, B_2, \ldots, B_\omega$ *if* $v(i) \leq j < N$.

(c) *If* $v(i) + \mu > N$, *then* j *can get to* i *by using* $B_2, B_3, \ldots, B_{\omega+1}$ *if* $0 \leq j < v(i+1)$; $B_2, B_3, \ldots, B_\omega$ *if* $v(i+1) \leq j < v(i)$; $B_1, B_2, \ldots, B_\omega$ *if* $v(i) \leq j < N$.

This theorem can be proven by using the arguments used in obtaining Fig. 3 and Theorem 5.2; hence we omit the proof.

We now summarize the routing algorithms for the backward network of BGSEN$(k, r, n+1)$. In the following table, "algo." means algorithm, (*1) means if *only one* tag is needed for an $(\overleftarrow{i, j})$-request, (*2) means if *all* tags are needed for an $(\overleftarrow{i, j})$-request, and "—" means not available.

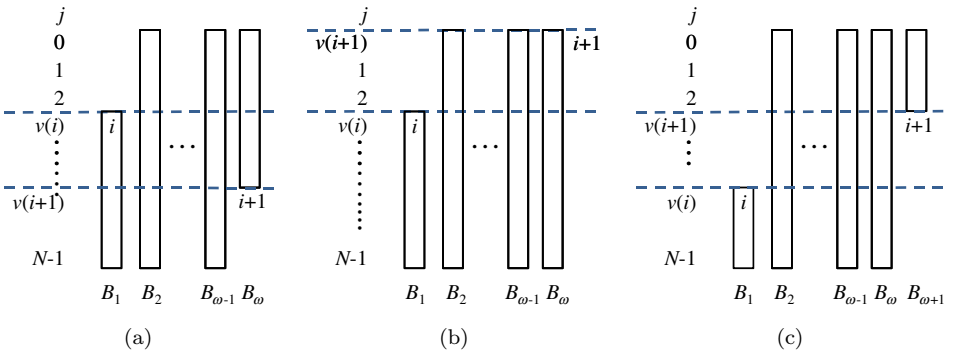


Fig. 4. An illustration of Theorem 5.3.

Table 2. Comparison of the routing algorithms for the bidirectional generalized shuffle-exchange network.

Time required to obtain	CLQ-algo.	CL-algo.	This paper (by which algorithm)
a tag for a j to get to an i	$O(n)$	$O(n)$	$O(1)$ (ONE-TO-ONE-ROUTING)
tags for all j's to get to an i	$O(Nn)$	$O(n)$	$O(1)$ (COMPUTE-TWO-TAGS)
a routing table (*1)	$O(N^2n)$	$O(Nn)$	$O(N)$ (COMPUTE-TWO-TAGS)
a routing table (*2)	$O(N^2kn)$	-	$O(N\omega)$ (COMPUTE-ALL-TAGS)

6. Concluding Remarks

This paper concerns about the routing problem for the bidirectional generalized shuffle-exchange network (BGSEN). Especially, we proposed the FBCT formula, which characterizes the relation between the forward and the backward control tags of a BGSEN. The FBCT formula is very useful and leads to several efficient routing algorithms. It leads to a simple $O(1)$-time one-to-one routing algorithm for the backward network of a BGSEN. For a destination i of the backward network of a BGSEN, the FBCT formula leads to a simple $O(1)$-time algorithm for generating the two backward control tags associated with i; this algorithm greatly simplifies Chen and Lou's algorithm [2]. The FBCT formula also leads to a very simple and efficient algorithm for constructing a routing table for the backward network of a BGSEN; such a routing table can be used for fault-tolerance.

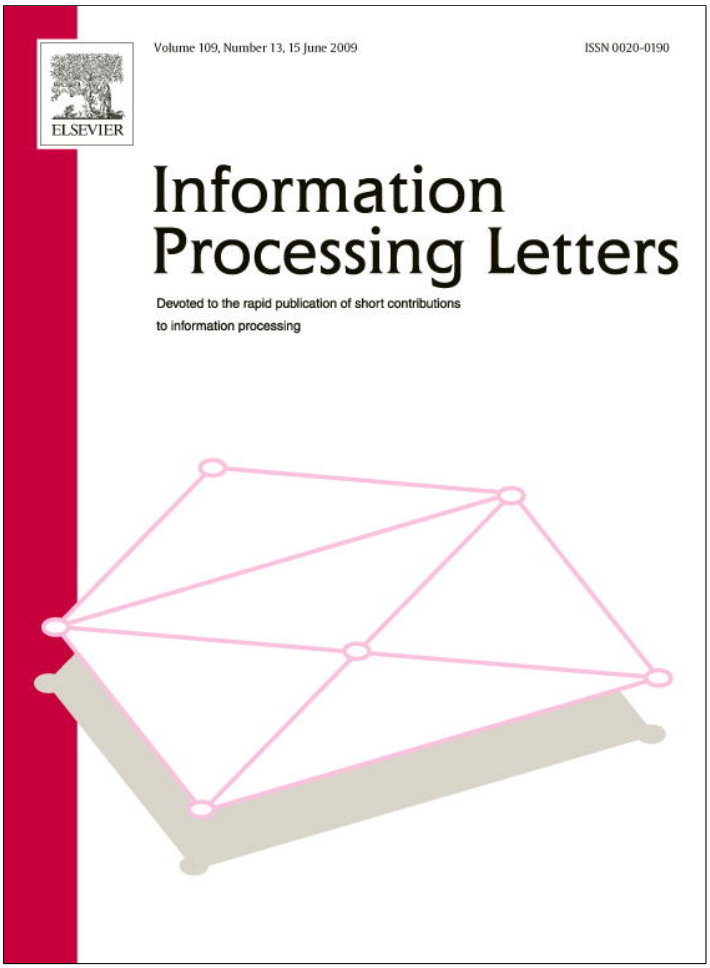
References

[1] Z. Chen, Z. Liu and Z. Qiu, Bidirectional shuffle-exchange network and tag-based routing algorithm, *IEEE Commun. Lett.* **7** (2003) 121–123.

[2] C. Chen and J. K. Lou, An efficient tag-based routing algorithm for the backward network of a bidirectional general shuffle-exchange network, *IEEE Commun. Lett.* **10** (2006) 296–298.

[3] M. Gerla, E. Leonardi, F. Neri and P. Palnati, Routing in the bidirectional shufflenet, *IEEE/ACM Trans. Netw.* **9** (2001) 91–103.

[4] F. K. Hwang, *The Mathematical Theory of Nonblocking Switching Networks*, Series on Applied Mathematics, Vol. 15 (2004), pp. 12–22.
[5] C. P. Kuruskal, A unified theory of interconnection network structure, *Theor. Comput. Sci.* **48** (1986) 75–94.
[6] D. H. Lawrie, Access and alignment of data in an array processor, *IEEE Trans. Comput.* **C-24** (1975) 1145–1155.
[7] S. C. Liew, On the stability if shuffle-exchange and bidirectional shuffle-exchange deflection networkA, *IEEE/ACM Trans. Netw.* **5** (1997) 87–94.
[8] K. Padmanabhan, Design and analysis of even-sized binary shuffle-exchange networks for multiprocessors, *IEEE Trans. Parallel Distrib. Syst.* **2** (1991) 385–397.
[9] R. Ramaswami, Multi-wavelength lightwave networks for computer communication, *IEEE Commun. Mag.* **31** (1993) 78–88.
[10] Y. Yang and J. Wang, Optimal all-to-all personalized exchange in a class of optical multistage networks, *IEEE Trans. Parallel Distrib. Syst.* **12** (2001) 567–582.



Information Processing Letters 109 (2009) 757–762

Contents lists available at ScienceDirect

Information Processing Letters

www.elsevier.com/locate/ipl

Improved upper and lower bounds on the optimization of mixed chordal ring networks ☆

James K. Lan [a], Victor W. Liu [b], Chiuyuan Chen [a,*]

[a] Department of Applied Mathematics, National Chiao Tung University, Hsinchu 300, Taiwan
[b] Department of Electrical & Computer Engineering, The State University of New York at Stony Brook, Stony Brook, NY 11794, USA

ARTICLE INFO

Article history:
Received 23 January 2009
Received in revised form 11 March 2009
Accepted 13 March 2009
Available online 21 March 2009
Communicated by A.A. Bertossi

Keywords:
Optimization
Diameter
Double-loop network
Mixed chordal ring network
Interconnection network
Parallel processing
Ring
Loop

ABSTRACT

Recently, Chen, Hwang and Liu [S.K. Chen, F.K. Hwang, Y.C. Liu, Some combinatorial properties of mixed chordal rings, J. Interconnection Networks 1 (2003) 3–16] introduced the mixed chordal ring network as a topology for interconnection networks. In particular, they showed that the amount of hardware and the network structure of the mixed chordal ring network are very comparable to the (directed) double-loop network, yet the mixed chordal ring network can achieve a better diameter than the double-loop network. More precisely, the mixed chordal ring network can achieve diameter about $\sqrt{2N}$ as compared to $\sqrt{3N}$ for the (directed) double-loop network, where N is the number of nodes in the network. One of the most important questions in interconnection networks is, for a given number of nodes, how to find an optimal network (a network with the smallest diameter) and give the construction of such a network. Chen et al. [S.K. Chen, F.K. Hwang, Y.C. Liu, Some combinatorial properties of mixed chordal rings, J. Interconnection Networks 1 (2003) 3–16] gave upper and lower bounds for such an optimization problem on the mixed chordal ring network. In this paper, we improve the upper and lower bounds as $2\lceil\sqrt{N/2}\rceil + 1$ and $\lceil\sqrt{2N} - 3/2\rceil$, respectively. In addition, we correct some deficient contexts in [S.K. Chen, F.K. Hwang, Y.C. Liu, Some combinatorial properties of mixed chordal rings, J. Interconnection Networks 1 (2003) 3–16].

1. Introduction

One of the most important issues in the design of parallel and distributed computing systems is the choice of an interconnection network suitable for a range of different applications. The *diameter* of a network, which is the maximum distance over all node-pairs, represents the maximum transmission delay between two stations. The *ring network* (i.e., the *single-loop network*) is one of the most frequently used topologies for interconnection networks. It has many attractive properties such as simplicity, extendibility, low degree, and ease of implementation. Although it has many attractive properties, it has poor reliability (any failure in an interface or communication link destroys the function of the network) and it has high transmission delay (its diameter equals to $N - 1$ if each link is directed, where N is the number of nodes). As a result, a lot of hybrid topologies utilizing the ring network as a basis for synthesizing richer interconnection schemes have been proposed to improve the reliability and reduce the transmission delay [3,4,6,14,17].

One example of the commonly used extensions for the ring network is the *multi-loop network*, which was first proposed by Wong and Coppersmith in [17] for organizing multi-module memory services. The most studied multi-loop network is possibly the double-loop network. The *double-loop network* $DL(N; a, b)$ is a digraph with N

☆ This research was partially supported by the National Science Council of the Republic of China under the grants grant NSC97-2628-M-009-006-MY3.

* Corresponding author.
E-mail addresses: drjamesblue@gmail.com (J.K. Lan), cychen@mail.nctu.edu.tw (C. Chen).

0020-0190/$ – see front matter
doi:10.1016/j.ipl.2009.03.017

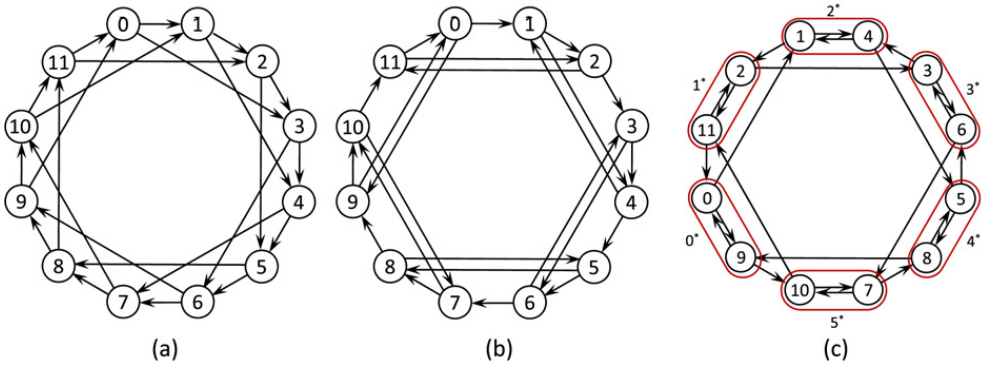


Fig. 1. Examples of the double-loop network and the mixed chordal ring network. (a) $DL(12;1,3)$. (b) $MCR(12;1,3)$. (c) Embed $MCR(12;1,3)$ into $DL(\frac{12}{2};\frac{1-3}{2},\frac{1+3}{2})$; i.e., $DL(6;5,2)$.

nodes $0,1,\ldots,N-1$ and $2N$ links (also called *steps*) (see Fig. 1(a)):

$$i \to (i+a) \bmod N, \quad i = 0,1,\ldots,N-1,$$

$$i \to (i+b) \bmod N, \quad i = 0,1,\ldots,N-1,$$

where $1 \leqslant a, b < N$, $a \neq b$, and $\gcd(N,a,b) = 1$. Doorn [9] had proven that:

Theorem 1.1. *(See [9].) $DL(N;a,b)$ is strongly 2-connected if and only if* $\gcd(N,a,b) = 1$.

Namely, any node or link failure will not disconnect the network. For a fixed N, let $D_{DL}(N)$ denote the optimal (i.e., smallest) diameter of all double-loop networks with N nodes. Many researchers tried to determine the exact value of $D_{DL}(N)$, but this is a difficult problem even when one of the two steps is 1 [4]; see also [1,7,8,10,12]. Therefore, researchers devoted their attention on finding bounds on $D_{DL}(N)$. A well-known lower bound on $D_{DL}(N)$ is as follows [17]:

$$D_{DL}(N) \geqslant \lceil \sqrt{3N} \rceil - 2. \tag{1}$$

For upper bounds on $D_{DL}(N)$, Hwang and Xu [13] managed to prove, using a heuristic method, that

$$D_{DL}(N) \leqslant \sqrt{3N} + 2(3N)^{1/4} + 5 \quad \text{for } N \geqslant 6348. \tag{2}$$

In [16], Rödseth further improved the above upper bound to be

$$D_{DL}(N) \leqslant \sqrt{3N} + (3N)^{1/4} + \frac{5}{2} \quad \text{for } N \geqslant 1200. \tag{3}$$

Another example of the commonly used extensions for the ring network is the *chordal ring network*; see [3] and [15]. Recently, Chen et al. [6] proposed the mixed chordal ring network as a topology of interconnection networks. The *mixed chordal ring network* $MCR(N;s,w)$, where N is even and both s and w are positive odd, is a digraph with N nodes $0,1,\ldots,N-1$ and $2N$ links of the following types (see Fig. 1(b)):

ring-links:

$$i \to (i+s) \bmod N, \quad i = 0,1,2,\ldots,N-1,$$

chordal-links:

$$i \to (i+w) \bmod N, \quad i = 1,3,5,\ldots,N-1,$$

chordal-links:

$$i \to (i-w) \bmod N, \quad i = 0,2,4,\ldots,N-2.$$

Let $d(N;s,w)$ denote the diameter of $MCR(N;s,w)$. For a fixed positive even integer N, let $D_{MCR}(N)$ denote the optimal (i.e., smallest) diameter of all mixed chordal ring networks with N nodes. It is obvious that each node in the mixed chordal ring network has two in-links and two out-links. Therefore, the mixed chordal ring network is very comparable in hardware to the well-known double-loop network; see [6]. Surprisingly, Theorems 1.2 and 1.3 show that the mixed chordal ring network can achieve a better diameter than the double-loop network (as compared to (1), (2) and (3)).

Theorem 1.2. *(See [6].)* $D_{MCR}(N) \geqslant \sqrt{2N} + o(N^{1/2})$.

Theorem 1.3. *(See [6].) There exists a choice of s and w such that the diameter of $MCR(N;s,w)$ is no larger than $\sqrt{2N}+3$. In other words, $D_{MCR}(N) \leqslant \sqrt{2N}+3$.*

Chen et al. [6] also proved:

Theorem 1.4. *(See [6].) $MCR(N;s,w)$ is strongly 2-connected if and only if* $\gcd(N,s,w) = 1$.

The proofs of Theorems 1.3 and 1.4 are based on the idea of embedding a mixed chordal ring network into a double-loop network (see Section 2 for details). Unfortunately, we find that this embedding is not always successful and the proofs of Theorems 1.3 and 1.4 are incomplete (see Section 4 for details). Thus whether $\sqrt{2N}+3$ is an upper bound on $D_{MCR}(N)$ and whether $\gcd(N,s,w) = 1$ guarantees the strongly 2-connectivity of $MCR(N;s,w)$ remain open. In this paper, we fill these voids by improving the upper and lower bounds on $D_{MCR}(N)$ and correcting the proof of Theorems 1.3 and 1.4. We summarize in Table 1.

This paper is organized as follows. Section 2 gives some preliminaries. Section 3 contains our main results. Section 4 gives a correct proof to Theorem 1.4. Section 5

Table 1

The bounds	In paper [6]	In this paper
Upper bound on $D_{MCR}(N)$	$\sqrt{2N}+3$	$2\lceil\sqrt{N/2}\rceil+1$
Lower bound on $D_{MCR}(N)$	$\sqrt{2N}+o(N^{\frac{1}{2}})$	$\lceil\sqrt{2N-3/2}\rceil$

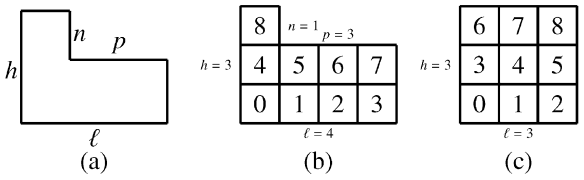


Fig. 2. The minimum distance diagram of double-loop networks. (a) The four parameters. (b) The L-shape of $DL(9;1,4)$. (c) The L-shape of $DL(9;1,3)$.

is for the concluding remarks; some open problems on the double-loop networks and the mixed chordal ring networks are also given here.

2. Preliminaries, assumptions, and embedding

Since the mixed chordal ring network is very related to the double-loop network, we first introduce some terminologies of the double-loop network. Given a $DL(N;a,b)$, a *minimum distance diagram* (MDD) is a diagram with node 0 in cell $(0,0)$ and node v in cell (i,j) if and only if $ia+jb\equiv v \pmod N$ and $i+j$ is the minimum among all (i',j') satisfying the congruence. In other words, a shortest path from 0 to v is through taking i a-links and j b-links (in any order). Note that, in cell (i,j), i (respectively, j) is the column (respectively, row) index. An MDD includes every node exactly once (in case of two shortest paths, the convention is to choose the cell with the smaller row index, i.e., the smaller j).

It had been proven that the MDD of $DL(N;a,b)$ is always an *L-shape* determined by four parameters ℓ, h, p, n [17]; see Fig. 2(a). These four parameters are the lengths of four of the six segments on the boundary of the L-shape. For example, $DL(9;1,4)$ has $\ell=4$, $h=3$, $p=3$, and $n=1$; see Fig. 2(b). An L-shape can degenerate into a rectangle as Fig. 2(c). Fiol et al. [11] observed that an L-shape always tessellates the plane regardless of the L-shape is degenerate or not. Many studies of the double-loop network are based on the L-shape [2,5,11]. One important function of the L-shape is that we can easily compute the diameter of its double-loop network by

$$\max\{\ell+h-p-2,\ \ell+h-n-2\}.$$

Throughout this paper, we will assume that $MCR(N;s,w)$ satisfies the following three conditions:

$$s\neq w,\qquad s+w\neq N,\quad \text{and}\quad \gcd(N,s,w)=1.$$

The reason is as follows. If $s=w$ or $s+w=N$, then $MCR(N;s,w)$ will contain multiple links between two nodes, which means a waste of the hardware. On the other hand, $MCR(N;s,w)$ is strongly connected if and only if $\gcd(N,s,w)=1$. Since we will only talk about strongly connected mixed chordal ring networks, we assume $\gcd(N,s,w)=1$.

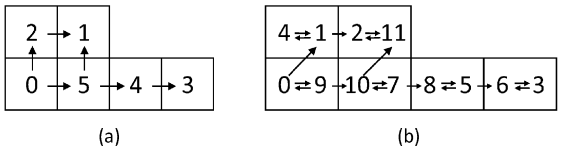


Fig. 3. The pseudo-MDD of a mixed chordal ring network. (a) The MDD of $DL(6;5,2)$. (b) The pseudo-MDD of $MCR(12;1,3)$.

Note that the double-loop network and the mixed chordal ring network are different network topologies: the former is vertex-transitive and the latter may or may not be vertex-transitive. For example, in $MCR(12;3,5)$, node 0 can reach any node within 4 moves, but it takes 5 moves for node 1 to reach node 8.

Chen, Hwang and Liu showed that the mixed chordal ring network $MCR(N;s,w)$ can be embedded into the double-loop network $DL(\frac{N}{2};\frac{s-w}{2},\frac{s+w}{2})$ by combining nodes $2k+1$ and $2k+1+w$ as *supernode* k^* for all $k=0,1,\ldots,N/2-1$ [6]. Note that, unless otherwise specified, $\frac{s-w}{2}$ means $(\frac{s-w}{2})$ mod $\frac{N}{2}$, $\frac{s+w}{2}$ means $(\frac{s+w}{2})$ mod $\frac{N}{2}$, all nodes of a mixed chordal ring network are taken modulo N, and all nodes of a double-loop network with $N/2$ nodes are taken modulo $N/2$.

Another way to embed the mixed chordal ring network $MCR(N;s,w)$ into the double-loop network $DL(\frac{N}{2};\frac{s-w}{2},\frac{s+w}{2})$ is to combine nodes $2k$ and $2k-w$ as supernode k^* for all $k=0,1,\ldots,N/2-1$. See Fig. 1(c) for an example. Such an embedding results in the same double-loop network $DL(\frac{N}{2};\frac{s-w}{2},\frac{s+w}{2})$ as the one used in [6] but is more natural since node 0 of $MCR(N;s,w)$ is in supernode 0^* of $DL(\frac{N}{2};\frac{s-w}{2},\frac{s+w}{2})$. Thus, throughout this paper, we use $DL(\frac{N}{2};\frac{s-w}{2},\frac{s+w}{2})$ to denote the embedding of combining nodes $2k$ and $2k-w$ as supernode k^*.

Since we can embed a mixed chordal ring network into a double-loop network, we can embed a mixed chordal ring network into the MDD of the corresponding double-loop network. More precisely, given $MCR(N;s,w)$, we replace each node k in the MDD of $DL(\frac{N}{2};\frac{s-w}{2},\frac{s+w}{2})$ with two nodes $2k$ and $2k-w$ in such a way that if k is in cell (i,j), then $2k$ and $2k-w$ are in cells $(2i,j)$ and $(2i+1,j)$, respectively. We call the resultant diagram the *pseudo-MDD* of $MCR(N;s,w)$. See Fig. 3 for an example.

The following lemma had been proven in [6] and it follows from the fact that each move in the MDD of $DL(\frac{N}{2};\frac{s-w}{2},\frac{s+w}{2})$ corresponds to either one or two moves in $MCR(N;s,w)$ (depending on which node in the supernode we start from).

Lemma 2.1. *(See [6].) Suppose* $DL(\frac{N}{2};\frac{s-w}{2},\frac{s+w}{2})$ *has* L-shape(ℓ,h,p,n), *then* $d(N;s,w)\leqslant 2\max\{\ell,h\}-1$.

3. Improved bounds on $D_{MCR}(N)$

This section is devoted to improve upper and lower bounds on $D_{MCR}(N)$. Given a $MCR(N;s,w)$, let n_k denote the number of additional nodes that node 0 can reach in k moves. Clearly, $n_0=0$, $n_1=2$ and $n_2=3$. Chen et al. [6] had proven that

$$n_k \leqslant n_{k-1} + 1 \quad \text{for } 2 \leqslant k \leqslant d(N; s, w). \tag{4}$$

In other words, for $k \geqslant 2$, the number of additional nodes that node 0 can reach at the kth move increases by at most 1. We now have the following result.

Theorem 3.1. *$D_{MCR}(N) \geqslant \lceil \sqrt{2N} - 3/2 \rceil$ and this bound is tight.*

Proof. By (4),

$$N \leqslant \sum_{k=0}^{d(N;s,w)} (k+1) = \frac{(d(N;s,w)+2)(d(N;s,w)+1)}{2}.$$

Therefore, $(d(N; s, w))^2 + 3d(N; s, w) + (2 - 2N) \geqslant 0$. Since $d(N; s, w)$ is positive, $d(N; s, w) \geqslant (\sqrt{8N+1} - 3)/2 > \sqrt{2N} - 3/2$. Since $d(N; s, w)$ is an integer, $d(N; s, w) \geqslant \lceil \sqrt{2N} - 3/2 \rceil$. This bound is tight since $d(8; 1, 3) = 3 \geqslant D_{MCR}(8) \geqslant \lceil \sqrt{2 \cdot 8} - 3/2 \rceil = 3$. □

We now obtain an upper bound on $D_{MCR}(N)$. The main idea used in obtaining the upper bound is, for each N, to choose s and w suitably so that the corresponding double-loop network $DL(\frac{N}{2}; \frac{s-w}{2}, \frac{s+w}{2})$ has an L-shape(ℓ, h, p, n) with ℓ and h being as small as possible and to apply Lemma 2.1.

Define $\hat{N}$ to be a function of N as follows:

$$\hat{N} = \left\lceil \sqrt{\frac{N}{2}} \right\rceil. \tag{5}$$

According to the parity of $\hat{N}$, define M as follows:

$$M = \begin{cases} \hat{N} & \text{if } \hat{N} \text{ is even,} \\ \hat{N} + 1 & \text{if otherwise.} \end{cases} \tag{6}$$

Lemma 3.2. *Suppose $N \neq 2(4t^2 + 2t - 1)$ for any positive integer t and let M be defined as in* (6)*. Then the* L-shape(ℓ, h, p, n) *of $DL(\frac{N}{2}; 1, M)$ satisfies $\ell \leqslant M$ and $h \leqslant M$.*

Proof. Consider $\mathbb{N} = \bigcup_{t=0}^{\infty} [4t^2 + 1, 4(t+1)^2]$. Then $\frac{N}{2} \in [4t^2 + 1, 4(t+1)^2]$ for some non-negative integer t. Thus $M = 2t + 2$. Consider the L-shape(ℓ, h, p, n) of $DL(\frac{N}{2}; 1, M)$. Since

$$M \cdot 1 \equiv 1 \cdot M \pmod{\frac{N}{2}},$$

cell $(M, 0)$ and cell $(0, 1)$ contain the same node. Since $M > 1$, cell $(M, 0)$ is outside the L-shape. Consequently, $\ell \leqslant M$. Now let $N_0(t) = [4t^2 + 1, 4t^2 + 2t - 2]$, $N_1(t) = [4t^2 + 2t - 1, 4t^2 + 4t]$, $N_2(t) = [4t^2 + 4t + 1, 4t^2 + 6t + 2]$, and $N_3(t) = [4t^2 + 6t + 3, 4t^2 + 8t + 4]$. Note that $N_0(0)$, $N_1(0)$, and $N_0(1)$ are empty. Then $\mathbb{N} = \bigcup_{t=0}^{\infty} (N_0(t) \cup N_1(t) \cup N_2(t) \cup N_3(t))$. Suppose $\frac{N}{2} \in N_k(t)$, where $0 \leqslant k \leqslant 3$. Define $N_k^*(t)$ to be the maximum integer in $N_k(t)$. Clearly, $N_k^*(t) = 4t^2 + 2t - 2 + (2t + 2)k$. Suppose $\frac{N}{2} = N_k^*(t) - j$ for some non-negative integer j. Then $0 \leqslant j \leqslant 2t - 3$ if $k = 0$ and $0 \leqslant j \leqslant 2t + 1$ if $1 \leqslant k \leqslant 3$. Again, consider the L-shape(ℓ, h, p, n) of $DL(\frac{N}{2}; 1, M)$. Since

$$\begin{aligned} j \cdot 1 &= N_k^*(t) - \frac{N}{2} \\ &= \left(4t^2 + 2t - 2 + (2t+2)k\right) - \frac{N}{2} \\ &\equiv (2t - 1 + k)(2t + 2) \pmod{\frac{N}{2}} \\ &= (2t - 1 + k)M \pmod{\frac{N}{2}}, \end{aligned}$$

cell $(j, 0)$ and cell $(0, 2t - 1 + k)$ contain the same node. Note that $j \leqslant 2t - 1 + k$ except when $k = 1$ and $j = 2t + 1$, that is, except when $\frac{N}{2} = 4t^2 + 2t - 1$. Hence if $N \neq 2(4t^2 + 2t - 1)$ for any positive integer t, then cell $(0, 2t - 1 + k)$ is outside the L-shape. Consequently, $h \leqslant 2t - 1 + k \leqslant 2t + 2 = M$. □

Lemma 3.3. *Suppose $N = 2(4t^2 + 2t - 1)$ for some positive integer t and let M be defined as in* (6)*. Then the* L-shape(ℓ, h, p, n) *of $DL(\frac{N}{2}; 2, M - 1)$ satisfies $\ell \leqslant M - 1$ and $h \leqslant M - 1$.*

Proof. Since $N = 2(4t^2 + 2t - 1)$ for some positive integer t, we have $M = 2t + 2$. Consider the L-shape(ℓ, h, p, n) of $DL(\frac{N}{2}; 2, M - 1)$. Since

$$(2t + 1) \cdot 2 \equiv 2 \cdot (2t + 1) \pmod{\frac{N}{2}},$$

cell $(2t + 1, 0)$ and cell $(0, 2)$ contain the same node. Since t is a positive integer, we have $2t + 1 > 2$. Thus cell $(2t + 1, 0)$ is outside the L-shape. Consequently, $\ell \leqslant 2t + 1 \leqslant M - 1$. Similarly, since

$$(t + 1) \cdot 2 \equiv (2t + 1)(2t + 1) \pmod{\frac{N}{2}},$$

cell $(t + 1, 0)$ and cell $(0, 2t + 1)$ contain the same node. Clearly, $2t + 1 > t + 1$ for $t > 0$; thus cell $(0, 2t + 1)$ is outside the L-shape. Thus $h \leqslant 2t + 1 \leqslant M - 1$. □

Lemma 3.4. *Let M be defined as in* (6)*. Then*:

1. *If $N \neq 2(4t^2 + 2t - 1)$ for any positive integer t, then $d(N; M + 1, M - 1) \leqslant 2M - 1$.*
2. *If $N = 2(4t^2 + 2t - 1)$ for some positive integer t, then $d(N; M + 1, M - 3) \leqslant 2M - 3$.*

Proof. Consider the first statement. It is not difficult to verify that both $M + 1$ and $M - 1$ are positive odd integers and $\gcd(N, M + 1, M - 1) = 1$. Thus $MCR(N; M + 1, M - 1)$ is a valid mixed chordal ring network. Since we can embed $MCR(N; M + 1, M - 1)$ into $DL(\frac{N}{2}; 1, M)$, this statement follows directly from Lemmas 2.1 and 3.2. The second statement can be proven similarly except that Lemma 3.2 is replaced with Lemma 3.3. □

Theorem 3.5. *Let $\hat{N}$ be defined as in* (5)*.*

1. *If $\hat{N}$ is even, then $D_{MCR}(N) \leqslant 2\lceil \sqrt{N/2} \rceil - 1$.*
2. *If $\hat{N}$ is odd and $N = 2(4t^2 + 2t - 1)$ for some positive integer t, then $D_{MCR}(N) \leqslant 2\lceil \sqrt{N/2} \rceil - 1$.*

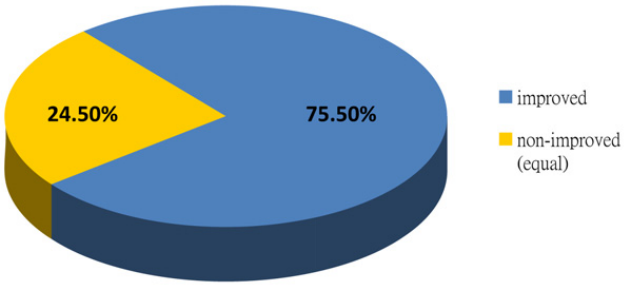


Fig. 4. The improved ratio of our upper bound as compared to the previous upper bound for $N = 6, 8, 10, \ldots, 10004$ (total 5000 N's).

3. *If $\hat{N}$ is odd and $N \neq 2(4t^2+2t-1)$ for any positive integer t, then $D_{MCR}(N) \leqslant 2\lceil\sqrt{N/2}\rceil + 1$.*

Moreover, these bounds are tight.

Proof. Note that if $N = 2(4t^2+2t-1)$ for some positive integer t, then $\hat{N}$ is odd. Thus if $\hat{N}$ is even, then $N \neq 2(4t^2 + 2t - 1)$ for any positive integer t; consequently, $M = \hat{N}$. If $\hat{N}$ is odd and $N = 2(4t^2 + 2t - 1)$ for some positive integer t, then $M = \hat{N} + 1$. If $\hat{N}$ is odd and $N \neq 2(4t^2 + 2t - 1)$ for any positive integer t, then $M = \hat{N} + 1$. Statements 1, 2 and 3 in this theorem now follow from Lemma 3.4. By the aid of a computer program, we obtain $D_{MCR}(20) = 7$, $D_{MCR}(38) = 9$ and $D_{MCR}(12) = 5$. Thus the bound in statement 1 is tight since $D_{MCR}(20) = 7$ and $2\lceil\sqrt{20/2}\rceil - 1 = 7$. The bound in statement 2 is tight since $D_{MCR}(38) = 9$ and $2\lceil\sqrt{38/2}\rceil - 1 = 9$. Similarly, the bound in statement 3 is tight since $D_{MCR}(12) = 5$ and $2\lceil\sqrt{12/2}\rceil - 1 = 5$. □

Remark 1. The previous upper bound on $D_{MCR}(N)$ is $\sqrt{2N} + 3$ [6]. Since $\sqrt{2N} + 3$ is served as an upper bound, we replace it with $\lfloor\sqrt{2N} + 3\rfloor$. The largest upper bound in Theorem 3.5 is $2\lceil\sqrt{N/2}\,\rceil + 1$ and it is always no larger than $\lfloor\sqrt{2N} + 3\rfloor$. To see how good our upper bound $2\lceil\sqrt{N/2}\rceil + 1$ is, we use a computer to obtain results for $N = 6, 8, 10, \ldots, 10004$. Among these 5000 N's, for 3775 (about 75.50%) out of them, our upper bound $2\lceil\sqrt{N/2}\rceil + 1$ improves the previous upper bound $\lfloor\sqrt{2N} + 3\rfloor$; see Fig. 4.

Remark 2. The upper bound $2\lceil\sqrt{N/2}\rceil - 1$ in Theorem 3.5 is no larger than the upper bound $\lceil\sqrt{2N}\rceil + 1$ in Theorem 3.5 and is very close to the lower bound $\lceil\sqrt{2N} - 3/2\rceil$ in Theorem 3.1. In the following, we show that there exist infinite number of N's such that the upper bound $2\lceil\sqrt{N/2}\rceil - 1$ matches the lower bound $\lceil\sqrt{2N} - 3/2\rceil$; in other words, we determine the exact value of $D_{MCR}(N)$ for these N's.

Theorem 3.6. *Suppose $N = 2(4t^2 - t + k)$ for some positive integers t and k, where $1 \leqslant k \leqslant t$. Then*

$$D_{MCR}(N) = 2\lceil\sqrt{N/2}\rceil - 1.$$

Moreover, $d(N; \lceil\sqrt{N/2}\rceil + 1, \lceil\sqrt{N/2}\rceil - 1) = D_{MCR}(N)$.

Proof. Suppose $N = 2(4t^2 - t + k)$ for some positive integer t and k, where $1 \leqslant k \leqslant t$. Then $2(4t^2 - 4t + 1) < N \leqslant 2 \cdot 4t^2$; therefore, $M = \hat{N} = \lceil\sqrt{N/2}\rceil = 2t$. By Lemma 3.4 and Theorem 3.5, $D_{MCR}(N) \leqslant d(N; \lceil\sqrt{N/2}\rceil + 1, \lceil\sqrt{N/2}\rceil - 1) \leqslant 2\lceil\sqrt{N/2}\rceil - 1$. Since $2(4t^2 - t + \frac{1}{4}) < N \leqslant 2(4t^2 + t + \frac{1}{4})$, we have $D_{MCR}(N) \geqslant \lceil\sqrt{2N} - 3/2\rceil = 4t - 1 = 2\lceil\sqrt{N/2}\rceil - 1$. We now have this theorem. □

The N's that satisfy Theorem 3.6 are: 8, 30, 32, 68, 70, 72, 122, ..., and so on. For $N = 6, 8, 10, \ldots, 10004$ (total 5000 N's), about 12.60% out of them satisfy Theorem 3.6 and their optimal diameter can be determined by Theorem 3.6.

4. Strongly connectivity of $MCR(N; s, w)$

We first indicate the problem in the proof of Theorem 1.3 in [6]. To obtain $D_{MCR}(38)$, Chen et al. [6] will use $MCR(38; 7, 5)$ and embed $MCR(38; 7, 5)$ into $DL(19; 1, 6)$. The L-shape of $DL(19; 1, 6)$ has $\ell = 5$ and $h = 7$, which has $h > N' = 6$ and violates

$$\ell \leqslant N' \quad \text{and} \quad h = N' \tag{7}$$

needed in the proof of $D_{MCR}(38) \leqslant \sqrt{2N} + 3$. In fact, we can construct infinite many N's that violates (7); see [6] for more details.

In Section 2, we have shown how to embed the mixed chordal ring network $MCR(N; s, w)$ into the double-loop network $DL(\frac{N}{2}; \frac{s-w}{2}, \frac{s+w}{2})$. However, this embedding sometimes fails. Take $MCR(10; 1, 5)$ as an example; its corresponding double-loop network is $DL(\frac{10}{2}; \frac{1-5}{2}, \frac{1+5}{2})$, i.e., $DL(5; 3, 3)$, which is clearly not a valid double-loop network, yet $MCR(10; 1, 5)$ is a valid mixed chordal ring network. In general, $MCR(2(2k+1); 1, 2k+1)$ is embedded into $DL(2k+1; k+1, k+1)$ but $DL(2k+1; k+1, k+1)$ is not a valid double-loop network. The idea used in [6] to prove Theorem 1.4 is to show that $MCR(N; s, w)$ is strongly 2-connected if and only if the corresponding double-loop network $DL(\frac{N}{2}; \frac{s-w}{2}, \frac{s+w}{2})$ is strongly 2-connected. We now correct the proof.

Lemma 4.1. *For $MCR(N; s, w)$,*

1. *if $w \neq \frac{N}{2}$, then $DL(\frac{N}{2}; \frac{s-w}{2}, \frac{s+w}{2})$ is a double-loop network;*
2. *if $w = \frac{N}{2}$, then $DL(\frac{N}{2}; \frac{s-w}{2}, \frac{s+w}{2})$ is not a double-loop network and $MCR(N; s, \frac{N}{2})$ is itself the double-loop network $DL(N; s, \frac{N}{2})$.*

Proof. $DL(\frac{N}{2}; \frac{s-w}{2}, \frac{s+w}{2})$ is not a valid double-loop network whenever $\frac{s-w}{2} \equiv 0 \pmod{\frac{N}{2}}$ or $\frac{s+w}{2} \equiv 0 \pmod{\frac{N}{2}}$ or $\frac{s-w}{2} \equiv \frac{s+w}{2} \pmod{\frac{N}{2}}$ or $\gcd(\frac{N}{2}, \frac{s-w}{2}, \frac{s+w}{2}) \neq 1$. Since we assume $s \neq w$ and $s + w \neq N$, it is impossible that $\frac{s-w}{2} \equiv 0 \pmod{\frac{N}{2}}$ or $\frac{s+w}{2} \equiv 0 \pmod{\frac{N}{2}}$. Also, $\frac{s-w}{2} \equiv \frac{s+w}{2} \pmod{\frac{N}{2}}$ if and only if $w = \frac{N}{2}$. In addition, we have assumed $\gcd(N, s, w) = 1$; therefore $\gcd(\frac{N}{2}, \frac{s-w}{2}, \frac{s+w}{2}) = 1$. Thus we have the first if-statement. When $w = \frac{N}{2}$, $\frac{N}{2} \equiv -\frac{N}{2} \pmod{N}$ occurs and the chordal-links of $MCR(N; s, w)$ become:

$$i \to \left(i + \frac{N}{2}\right) \bmod N, \quad i = 0, 1, \ldots, N-1.$$

Thus $MCR(N; s, \frac{N}{2})$ is itself the double-loop network $DL(N; s, \frac{N}{2})$ and we have the second if-statement. □

Lemma 4.1 shows that $DL(\frac{N}{2}; \frac{s-w}{2}, \frac{s+w}{2})$ is a valid embedding if and only if $w \neq \frac{N}{2}$. It was proven in [6] that

Lemma 4.2. *$MCR(N; s, w)$ is strongly connected if and only if* $\gcd(N, s, w) = 1$.

Now we give correct proof of Theorem 1.4.

Proof of Theorem 1.4. *Necessity.* This follows directly from Lemma 4.2.

Sufficiency. There are two cases.

Case 1: $w \neq \frac{N}{2}$. Then by Lemma 4.1, $DL(\frac{N}{2}; \frac{s-w}{2}, \frac{s+w}{2})$ is a double-loop network. Since $w \neq \frac{N}{2}$, $\frac{s-w}{2} \neq \frac{s+w}{2}$. Since $\gcd(N, s, w) = 1$, $\gcd(\frac{N}{2}, \frac{s-w}{2}, \frac{s+w}{2}) = 1$. Thus by Theorem 1.1, $DL(\frac{N}{2}; \frac{s-w}{2}, \frac{s+w}{2})$ is strongly 2-connected. Since the two nodes in each super-node can reach each other through the chordal-links between them, $MCR(N; s, w)$ is strongly 2-connected.

Case 2: $w = \frac{N}{2}$. By Lemma 4.1, $MCR(N; s, w)$ is itself the double-loop network $DL(N; s, w)$. Thus by Theorem 1.1 and by the assumption that $\gcd(N, s, w) = 1$, $MCR(N; s, w)$ is strongly 2-connected. □

5. Concluding remarks

In [6], Chen et al. proposed a new network topology called the mixed chordal ring network and discussed its combinatorial properties. They obtained the surprising result that the mixed chordal ring network is comparable in hardware to the well-known double-loop network and yet can achieve a better diameter than the double-loop network. In this paper, we have improved the upper and lower bounds on $D_{MCR}(N)$ (i.e., the optimal diameter of mixed chordal ring networks) as $2\lceil\sqrt{N/2}\rceil + 1$ and $\lceil\sqrt{2N} - 3/2\rceil$, respectively. We have also corrected some deficient contexts in [6].

For the double-loop network, determining the exact value of $D_{DL}(N)$ is a hard problem and even determining $\overline{D}_{DL}(N) = \min_b\{d_{DL}(N; 1, b)\}$, where $d_{DL}(N; 1, b)$ is the diameter of $DL(N; 1, b)$, is a hard problem, too [4]. By (1), (2) and (3), the gap between the upper and the lower bounds on $D_{DL}(N)$ increases by a factor of $(3N)^{1/4}$ and it seems that there is no closed form for $D_{DL}(N)$. For the mixed chordal ring network, we have successfully narrowed the gap between the upper and the lower bounds on $D_{MCR}(N)$ as $2\lceil\sqrt{N/2}\rceil + 1$ and $\lceil\sqrt{2N} - 3/2\rceil$. It has a great probability to determine $D_{MCR}(N)$ and therefore solve this optimization problem in the near future.

References

[1] F. Aguiló, M.A. Fiol, An efficient algorithm to find optimal double loop networks, Discrete Math. 138 (1995) 15–29.

[2] F. Aguiló, E. Simo, M. Zaragoza, Optimal double-loop networks with non-unit steps, Electron. J. Combin. 10 (1) (2003).

[3] B. Arden, H. Lee, Analysis of chordal ring network, IEEE Trans. Comput. 30 (4) (1981) 291–295.

[4] J.C. Bermond, F. Comellas, D.F. Hsu, Distributed loop computer-networks – a survey, J. Parallel Distrib. Comput. 24 (1) (1995) 2–10.

[5] C. Chen, J.K. Lan, W.-S. Tang, An efficient algorithm to find a double-loop network that realizes a given L-shape, Theoret. Comput. Sci. 359 (1–3) (2006) 69–76.

[6] S.K. Chen, F.K. Hwang, Y.C. Liu, Some combinatorial properties of mixed chordal rings, J. Interconnection Networks 1 (2003) 3–16.

[7] Y. Cheng, F.K. Hwang, Diameters of weighted double loop networks, J. Algorithms 9 (1988) 401–410.

[8] C.Y. Chou, D.J. Guan, K.L. Wang, A dynamic fault-tolerant message routing algorithm for double-loop networks, Inform. Process. Lett. 70 (6) (1999) 259–264.

[9] E.A. Doorn, Connectivity of circulant digraphs, J. Graph Theory 10 (1) (1986) 9–14.

[10] P. Esqué, F. Aguiló, M.A. Fiol, Double commutative-step digraphs with minimum diameters, Discrete Math. 114 (1993) 147–157.

[11] M.A. Fiol, J.L.A. Yebra, I. Alegre, M. Valero, A discrete optimization problem in local networks and data alignment, IEEE Trans. Comput. C-36 (6) (1987) 702–713.

[12] D.J. Guan, An optimal message routing algorithm for double-loop networks, Inform. Process. Lett. 65 (5) (1998) 255–260.

[13] F. Hwang, Y. Xu, Double loop networks with minimum delay, Discrete Math. 66 (1–2) (1987) 109–118.

[14] F.K. Hwang, A complementary survey on double-loop networks, Theoret. Comput. Sci. 263 (1–2) (2001) 211–229.

[15] F.K. Hwang, P.E. Wright, Survival reliability of some double-loop networks and chordal rings, IEEE Trans. Comput. 44 (12) (1995) 1468–1471.

[16] O.J. Rödseth, Weighted multi-connected loop networks, Discrete Math. 148 (1996) 161–173.

[17] C.K. Wong, D. Coppersmith, A combinatorial problem related to multimodule memory organizations, J. ACM 21 (3) (1974) 392–402.

Theoretical Computer Science 411 (2010) 1669–1684

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

All-to-all personalized exchange in generalized shuffle-exchange networks☆

Well Y. Chou, Chiuyuan Chen*

Department of Applied Mathematics, National Chiao Tung University, Hsinchu 300, Taiwan

ARTICLE INFO

Article history:
Received 8 May 2009
Accepted 25 October 2009
Communicated by D.-Z. Du

Keywords:
Multistage interconnection networks
Shuffle-exchange networks
Omega network
Parallel and distributed computing
All-to-all communication
All-to-all personalized exchange

ABSTRACT

An all-to-all communication algorithm is said to be optimal if it has the smallest communication delay. Previous all-to-all personalized exchange algorithms are mainly for hypercube, mesh, and torus. In Yang and Wang (2000) [13], Yang and Wang proved that a multistage interconnection network (MIN) is a better choice for implementing all-to-all personalized exchange and they proposed optimal all-to-all personalized exchange algorithms for MINs. In Massini (2003) [9], Massini proposed a new optimal algorithm for MINs, which is independent of the network topology. Do notice that the algorithms in [9] and [13] work only for MINs with the unique path property (meaning that there is a unique path between each pair of source and destination) and satisfying $N = 2^n$, in which N is the number of processors, 2 means all the switches are of size 2×2, and n is the number of stages. In Padmanabhan (1991) [10], Padmanabhan proposed the generalized shuffle-exchange network (GSEN), which is a generalization of the shuffle-exchange network. Since a GSEN does not have the unique path property, the algorithms in [9] and [13] cannot be used. The purpose of this paper is to consider the all-to-all personalized exchange problem in GSENs. An optimal algorithm and several bounds will be proposed.

1. Introduction

Processors in a parallel and distributed processing system often need to communicate with other processors. The communication among these processors could be *one-to-one*, *one-to-many*, or *all-to-all*. All-to-all communication can be further classified into *all-to-all broadcast* and *all-to-all personalized exchange*. In all-to-all broadcast, each processor sends the same message to all other processors; while in all-to-all personalized exchange, each processor sends a specific message to every other processor. All-to-all personalized exchange occurs in many important applications (for example, matrix transposition and fast Fourier transform (FFT)) in parallel and distributed computing. The all-to-all personalized exchange problem has been extensively studied for hypercubes, meshes, and tori; see [9,13] for details. Although the algorithm for a hypercube achieves optimal time complexity, a hypercube suffers from unbounded node degrees and therefore has poor scalability; on the other hand, although a mesh or torus has a constant node degree and better scalability, its algorithm has a higher time complexity. In [13], Yang and Wang had proven that a multistage interconnection network (MIN) is a better choice for implementing all-to-all personalized exchange due to its shorter communication delay and better scalability.

Given N processors $P_0, P_1, \ldots, P_{N-1}$, an $N \times N$ MIN can be used in communication among these processors as shown in Figs. 1 and 2, where $N \times N$ means this MIN has N inputs and N outputs. A column in a MIN is called a *stage* and the nodes stages of a MIN are called *switches* (or *switching elements* or *crossbars*). Throughout this paper, N denotes the number of

☆ This research was partially supported by the National Science Council of the Republic of China under grant NSC97-2628-M-009-006-MY3.
* Corresponding author. Tel.: +886 3 5731767.
E-mail addresses: well.am94g@nctu.edu.tw (W.Y. Chou), cychen@mail.nctu.edu.tw, cychen@cc.nctu.edu.tw (C. Chen).

0304-3975/$ – see front matter © 2010 Published by Elsevier B.V.
doi:10.1016/j.tcs.2009.10.026

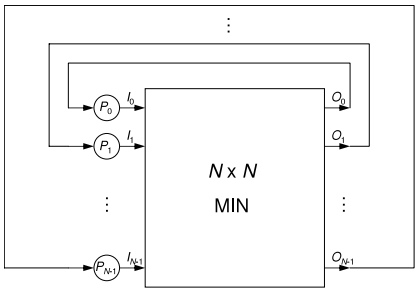


Fig. 1. Communications among processors using a MIN.

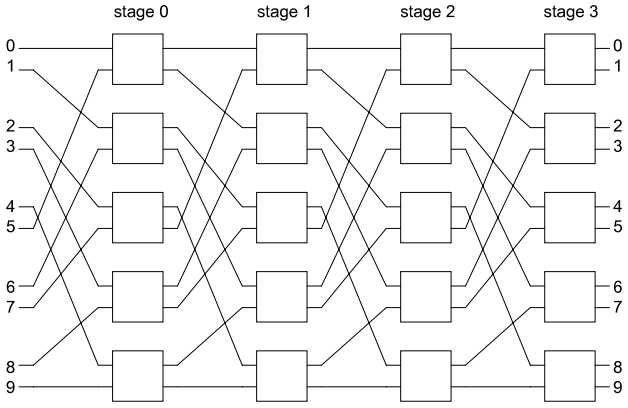


Fig. 2. A 10 × 10 MIN which is also a 10 × 10 GSEN.

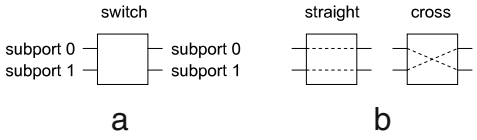


Fig. 3. (a) A 2 × 2 switch and its subports. (b) The two possible states of a 2 × 2 switch.

inputs (outputs) and n denotes the number of stages. Also, all the switches in a MIN are assumed to be of size 2×2. It is well known that a 2×2 switch has only two possible states: *straight* and *cross*, as shown in Fig. 3. A shuffle-exchange network (SEN) is also called an omega network (see [7]) and has been proposed as a popular architecture for MINs; see [3,6,10,12]. Since a SEN must satisfy $N = 2^n$, in [10], Padmanabhan generalized it to allow $N \neq 2^n$. More precisely, let N be an even integer. An $N \times N$ *generalized shuffle-exchange network* (GSEN) is a $\lceil \log_2 N \rceil$-stage $N \times N$ MIN such that each stage consists of the perfect shuffle on N terminals followed by $N/2$ switches. The N terminals in an $N \times N$ GSEN are numbered $0, 1, \ldots, N-1$ and the *perfect shuffle operation* on the N terminals is the permutation π defined by $\pi(i) = (2 \cdot i + \lfloor \frac{2 \cdot i}{N} \rfloor) \bmod N, \ \ 0 \leq i < N$. See Fig. 2 for an example. In [1,2], bidirectional GSENs are considered.

In the remaining discussion, unless otherwise specified, a MIN means an $N \times N$ MIN and a GSEN means an $N \times N$ GSEN. Do notice that we will follow the convention used in [1,2,10] that a GSEN has exactly $\lceil \log_2 N \rceil$ stages; $\lceil \log_2 N \rceil$ is the minimum number of stages to ensure that each input can get to each output. Based on this convention and for convenience, we will define

$$n = \lceil \log_2 N \rceil.$$

Clearly, for a GSEN, its N satisfies $2^{n-1} < N \leq 2^n$.

In this paper, an all-to-all communication algorithm is said to be *optimal* if it has the smallest communication delay. Now we review previous results. Yang and Wang [13] first considered the all-to-all personalized exchange problem for MINs. In particular, they proposed optimal all-to-all personalized exchange algorithms for a class of unique path, self-routable MINs; for example, baseline, omega, banyan networks, and the reverse networks of these networks. Note that a MIN is *unique path* if there is a unique path between each pair of source and destination and *self-routable* if the routing decision at a switch depends only on the addresses of the source and the destination. The algorithms in [13] can use the *stage control* technique (see [11]), which is a commonly used technique to reduce the cost of the network setting for all-to-all personalized exchange communication. Stage control means that the states of all the switches of a stage have to be identical. With stage control, a single control bit (0 for straight and 1 for cross), or in other words, one electronic driver circuit, can be used to control all the switches of a stage. Thus the number of expensive electronic driver circuits needed is significantly lower than that

of individual switch control. It was pointed out by Massini in [9] that the algorithms in [13] depend on network topologies and require pre-computation and memory allocation for Latin squares. In the same paper, Massini proposed a new optimal algorithm, which is independent on the network topology and does not require pre-computation or memory allocation for a Latin square. In [8], Liu et al. further generalized Massini's algorithm to MINs with $d \times d$ switches. See also [14].

When $N \neq 2^n$, it is possible to implement an $N \times N$ GSEN by using a $2^n \times 2^n$ SEN (recall that $2^{n-1} < N \leq 2^n$). For example, it is possible to implement a 514×514 GSEN by using a 1024×1024 SEN. A 514×514 GSEN uses 2570 switches while its corresponding 1024×1024 SEN uses 5120 switches; the former saves about 50% switches than the latter. To compare the hardware costs of a GSEN and a SEN, we calculate the numbers of switches used by an $N \times N$ GSEN and by its corresponding $2^n \times 2^n$ SEN for $N = 4, 6, 8, \ldots, 10002$. Among these 5000 N's,

- for 4175 (about 84%) out of them, a GSEN saves at least 10% switches than its corresponding SEN;
- for 3356 (about 67%) out of them, a GSEN saves at least 20% switches than its corresponding SEN;
- for 2537 (about 51%) out of them, a GSEN saves at least 30% switches than its corresponding SEN;
- for 1632 (about 33%) out of them, a GSEN saves at least 40% switches than its corresponding SEN.

Therefore a GSEN outperforms a SEN in hardware cost.

Do notice that although the algorithms in [9] and [13] are optimal, they work only for MINs that have the unique path property and satisfy $N = 2^n$. Since a GSEN is not a unique path MIN, the algorithms in [9] and [13] cannot be used. To our knowledge, no one has studied the all-to-all personalized exchange problem for MINs which do not have the unique path property and do not satisfy $N = 2^n$. The purpose of this paper is to consider the all-to-all personalized exchange problem for GSENs. In particular, we propose an optimal all-to-all personalized exchange algorithm for GSENs. This algorithm works for all N with $N \equiv 2 \pmod 4$. Let $\mathcal{R}(N)$ and $\mathcal{R}_{sc}(N)$ denote the minimum number of network configurations (defined in the next section) required to fulfill an all-to-all communication in a GSEN when the stage control technique is not assumed and assumed, respectively. Do notice that $\mathcal{R}(N)$ and $\mathcal{R}_{sc}(N)$ are closely related to the smallest communication delay. In particular, for a GSEN, the smallest communication delay of any all-to-all communication algorithm is $\theta(\mathcal{R}(N) + \log_2 N)$ and $\theta(\mathcal{R}_{sc}(N) + \log_2 N)$ when the stage control technique is not assumed and assumed, respectively. The optimal algorithms in [9] and [13] imply that $\mathcal{R}(2^n) = \mathcal{R}_{sc}(2^n) = 2^n$. In this paper, we will prove that, for $2^{n-1} < N \leq 2^n$, the followings hold:

- $N \leq \mathcal{R}(N) \leq \mathcal{R}_{sc}(N) \leq 2^n$;
- $\mathcal{R}_{sc}(N) = 2^n$;
- $\mathcal{R}(N) = N$ if $N \equiv 2 \pmod 4$;
- $\mathcal{R}(N) = 2^n$ if $k \geq 2, N \equiv 0 \pmod{2^k}, N \not\equiv 0 \pmod{2^{k+1}}$, and $2^{n-1} + 2^{n-k} \leq N \leq 2^n$;
- $\mathcal{R}(20) = 24$.

This paper is organized as follows: In Section 2, we give some preliminaries. In Section 3, we prove $N \leq \mathcal{R}(N) \leq \mathcal{R}_{sc}(N) = 2^n$. In Section 4, we propose an optimal all-to-all personalized exchange algorithm for GSENs with $N \equiv 2 \pmod 4$ and prove that $\mathcal{R}(N) = N$ if $N \equiv 2 \pmod 4$. In Section 5, we focus on GSENs with $N \equiv 0 \pmod 4$ and obtain several bounds. Some discussions and concluding remarks are given in the final section.

2. Preliminaries

In a GSEN, the switches are aligned in n stages: stage 0, stage 1, ..., stage $n-1$, with each stage consists of $N/2$ switches. The *network configuration* of a GSEN is defined by the states of its switches. Since a GSEN has $(N/2) \times n$ switches, its network configuration can be represented by an $(N/2) \times n$ matrix in which each entry is defined by the state of its corresponding switch. For example, the network configuration of the GSEN in Fig. 4(a) is shown in Fig. 4(b).

A *permutation* of a MIN is one-to-one mapping between the inputs and outputs. For a MIN, if there is a permutation that maps input i to output $p(i)$, where $p(i) \in \{0, 1, \ldots, N-1\}$ for $i = 0, 1, \ldots, N-1$, then we will use

$$\begin{pmatrix} 0 & 1 & \cdots & N-1 \\ p(0) & p(1) & \cdots & p(N-1) \end{pmatrix}$$

or simply

$$p(0)\, p(1)\, \cdots\, p(N-1)$$

to denote the permutation. Given the network configuration of a MIN, a permutation between the inputs and outputs can be obtained. For example, the network configuration shown in Fig. 4(a) maps input 0 to output 9, input 1 to output 7, input 2 to output 5, ..., and input 9 to output 0; thus this network configuration obtains the permutation 9 7 5 3 8 1 6 4 2 0.

The following conventions are used in the remaining part of this paper. Terminal i (j) is assumed on the left-hand (right-hand) side of the network and therefore is an input (output) processor. An (i, j)-*request* denotes a request for sending a message from i to j. An (i, j)-*path* denotes a path between i and j. Obviously, an (i, j)-request can be fulfilled by an (i, j)-path.

Consider an (i, j)-request and an (i, j)-path and see Fig. 5 for an illustration. An (i, j)-path P can be described by a sequence of labels that label the successive links on this path; a number whose binary representation corresponds to such a sequence is called a *control tag* or *tag* or *path descriptor* [1,2,4,10]. A control tag can be used as a header for routing a message: each successive switch uses the first element in the binary representation of the control tag to route the message, and then

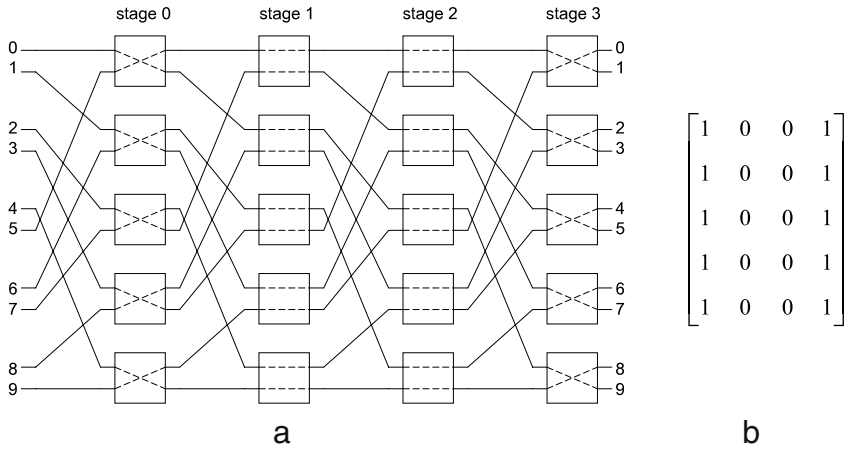


Fig. 4. (a) A 10 × 10 GSEN in which stage control is used. (b) The network configuration of the GSEN in (a).

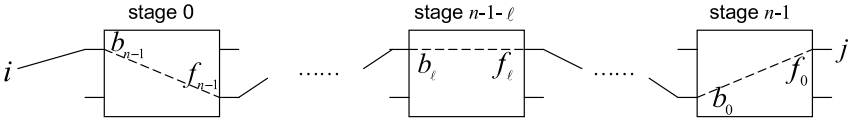


Fig. 5. An (i, j)-path P and the subports on P.

discards it. Take Fig. 4(a) for an example. Then $i = 2$ can get to $j = 5$ by using $13 = (1101)_2$, which means that the $(2, 5)$-request can be fulfilled by the path via subport 1 at stage 0, subport 1 at stage 1, subport 0 at stage 2, and subport 1 at stage 3. A routing algorithm is called *tag-based* if it uses a control tag to route a message. Most of the routing algorithms for MINs are tag-based, including those for GSENs. The routing algorithms proposed in this paper are also tag-based. Therefore, whenever a message is sent out, a control tag will be equipped with it.

Again, see Fig. 5. When a message is sent from i to j along P, the message enters a switch at stage $n-1-\ell$ via subport b_ℓ and leaves the switch via subport f_ℓ. On the other hand, when a message is sent from j to i along P, then the message enters a switch at stage $n-1-\ell$ via subport f_ℓ and leaves the switch via subport b_ℓ. The control tag

$$F = f_{n-1}2^{n-1} + f_{n-2}2^{n-2} + \cdots + f_0 2^0$$

is called a *forward control tag* for i to get to j. Most researchers simply called a forward control tag a control tag; here we add the word "forward" to specify that this control tag is used for sending a message in the forward direction, i.e., from the left-hand side of the GSEN to the right-hand side. Now let

$$B = b_{n-1}2^{n-1} + b_{n-2}2^{n-2} + \cdots + b_0 2^0.$$

B is called a *backward control tag* and it is used for sending a message in the backward direction (from j to i). Clearly, $0 \le F < 2^n$ and $0 \le B < 2^n$.

Suppose F is given. In this paper, $P(i, F)$ denotes the path started from i and using the forward control tag F. Also, $B(i, F)$ denotes the backward control tag obtained from the path $P(i, F)$. Let

$$\mathcal{B}_F = \{B(i, F) \mid i = 0, 1, \ldots, N-1\}.$$

In the remaining discussion, $\oplus$ denotes the bitwise XOR operation. As a reference,

$$0 \oplus 0 = 0, \quad 0 \oplus 1 = 1, \quad 1 \oplus 0 = 1, \quad 1 \oplus 1 = 0.$$

If $U = (u_{n-1}\, u_{n-2}\, \cdots\, u_0)_2$ and $V = (v_{n-1}\, v_{n-2}\, \cdots\, v_0)_2$, then we define

$$U \oplus V = (u_{n-1} \oplus v_{n-1}\ u_{n-2} \oplus v_{n-2}\ \cdots\ u_0 \oplus v_0)_2.$$

3. The proof of $N \le \mathcal{R}(N) \le \mathcal{R}_{sc}(N) = 2^n$

The purpose of this section is to prove that $N \le \mathcal{R}(N) \le \mathcal{R}_{sc}(N) = 2^n$. We first prove two lemmas.

Lemma 3.1. $N \le \mathcal{R}(N) \le \mathcal{R}_{sc}(N) \le 2^n$.

Proof. Given a network configuration, a permutation can be obtained. Thus a network configuration can be used to send N (personalized) messages simultaneously. The inequality $N \le \mathcal{R}(N)$ thus follows from that fact that N^2 messages have to be sent to fulfill all-to-all personalized exchange and each network configuration can send only N messages. The inequality $\mathcal{R}(N) \le \mathcal{R}_{sc}(N)$ is obvious. The inequality $\mathcal{R}_{sc}(N) \le 2^n$ follows from the fact that a GSEN has at most 2^n network configurations when the stage control technique is assumed. □

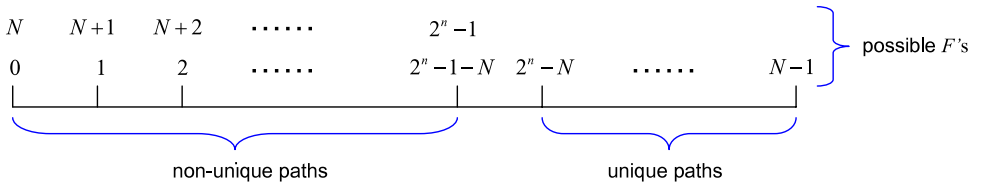


Fig. 6. Unique paths and multipaths.

In this paper, we call the process of transmitting all the messages to their next stage(s) a *round*. Thus in an n-stage MIN, it takes n rounds for a message to arrive its destination. In [13], Yang and Wang proved that the communication delay of all-to-all personalized exchange in a $(\log_2 N)$-stage MIN is $\Omega(N+\log_2 N)$. This is due to the fact that each of the N processors (say, processor j) has to receive N messages and it takes $\log_2 N$ rounds for the first message to arrive j and $N-1$ rounds for the remaining $N-1$ messages to arrive j. By similar arguments, we have the following lemma and its proof is omitted.

Lemma 3.2. *The communication delay of all-to-all communication in a GSEN is $\Omega(N+n)$, or equivalently, $\Omega(N+\log_2 N)$.*

Do notice that although $\Omega(N+n)=\Omega(N)$, we will still write $\Omega(N+n)$ instead of $\Omega(N)$ to emphasize that it takes n rounds for the first message to arrive its destination. In [5], Lan et al. considered GSENs with switches of size $d\times d$. By setting $d=2$, the following lemma can be obtained.

Lemma 3.3 (*[5]*). *Given i and F, the destination j of the path $P(i,F)$ is determined by*

$$j=(i\cdot 2^n+F) \mod N.$$

Moreover, the backward control tag B of the path $P(i,F)$ is given by

$$B=\left\lfloor \frac{i\cdot 2^n+F}{N} \right\rfloor.$$

When the stage control technique is assumed, the network configuration of a GSEN can be represented by a number as follows. Let c_ℓ denote the state, 0 for straight and 1 for cross, of all the switches at stage $n-1-\ell$. Then the network configuration of the GSEN can be represented by the number

$$C=c_{n-1}2^{n-1}+c_{n-2}2^{n-2}+\cdots+c_0 2^0$$

or by the binary number $(c_{n-1}\ c_{n-2}\ \cdots\ c_0)_2$. For example, the network configuration of the GSEN in Fig. 4(a) can be represented by 9 or by $(1001)_2$. Clearly, $0\le C<2^n$. Now we give the relation between F (a forward control tag), B (its corresponding backward control tag) and C (the network configuration).

Lemma 3.4. *When the stage control technique is assumed, F and B together uniquely determine the network configuration C and*

$$C=B\oplus F.$$

Proof. Consider stage $n-1-\ell$. Since the stage control technique is assumed, all switches in stage $n-1-\ell$ are of the same state. Let $C=c_{n-1}2^{n-1}+c_{n-2}2^{n-2}+\cdots+c_0 2^0$ be the network configuration and see Fig. 5. At stage $n-1-\ell$, a message enters subport b_ℓ and leaves subport f_ℓ. If $b_\ell=f_\ell$, then the state of the switch is straight; hence $c_\ell=0=b_\ell\oplus f_\ell$. If b_ℓ differs from f_ℓ (in this case, (b_ℓ,f_ℓ) is (0, 1) or (1, 0)), then the state of the switch is cross; hence $c_\ell=1=b_\ell\oplus f_\ell$. From the above, $C=B\oplus F$. □

We call a path a *unique path* if it is the unique path between its source and destination. The following lemma is important.

Lemma 3.5. *For all $0\le i<N$, path $P(i,F)$ is a unique path if and only if $2^n-N\le F<N$; in particular, $P(i,2^{n-1})$ and $P(i,2^{n-1}+1)$ are unique paths. (See Fig. 6 for illustration.)*

Proof. Let i and j be the source and destination of a message. Suppose there are two distinct paths $P(i,F_1), P(i,F_2)$ from i to j. Then, by Lemma 3.3, the difference between F_1 and F_2 is N. Without loss of generality, assume that $F_2-F_1=N$. Since $F_1\ge 0$, $F_2\ge N$ must hold. Since $F_2<2^n$, it follows that $F_1<2^n-N$. Thus $P(i,F)$ is a unique path if and only if $2^n-N\le F<N$. Since $2^n-N\le 2^{n-1}<N$, $P(i,2^{n-1})$ is a unique path. Since $2^n-N\le 2^{n-1}+1<N$, $P(i,2^{n-1}+1)$ is also a unique path. □

Lemma 3.6. $\mathcal{B}_{2^{n-1}}=\mathcal{B}_{2^{n-1}+1}$.

Proof. Let $0\le i<N$. Let $b_{n-1}f_{n-1}b_{n-2}f_{n-2}\cdots b_0 f_0$ be the sequence of subports passed by path $P(i,2^{n-1})$; see Fig. 5. Similarly, let $b'_{n-1}f'_{n-1}b'_{n-2}f'_{n-2}\cdots b'_0 f'_0$ be the sequence of subports passed by path $P(i,2^{n-1}+1)$. Since the binary representations of 2^{n-1} and $2^{n-1}+1$ differ only at their rightmost bits, $b_{n-1}f_{n-1}b_{n-2}f_{n-2}\cdots b_0 f_0$ and $b'_{n-1}f'_{n-1}b'_{n-2}f'_{n-2}\cdots b'_0 f'_0$ are identical except that $f_0\ne f'_0$. Hence $B(i,2^{n-1})=b_{n-1}b_{n-2}\cdots b_0=b'_{n-1}b'_{n-2}\cdots b'_0=B(i,2^{n-1}+1)$. Since $\mathcal{B}_{2^{n-1}}=\{B(i,2^{n-1})\mid i=0,1,\ldots,N-1\}$ and $\mathcal{B}_{2^{n-1}+1}=\{B(i,2^{n-1}+1)\mid i=0,1,\ldots,N-1\}$, we have this lemma. □

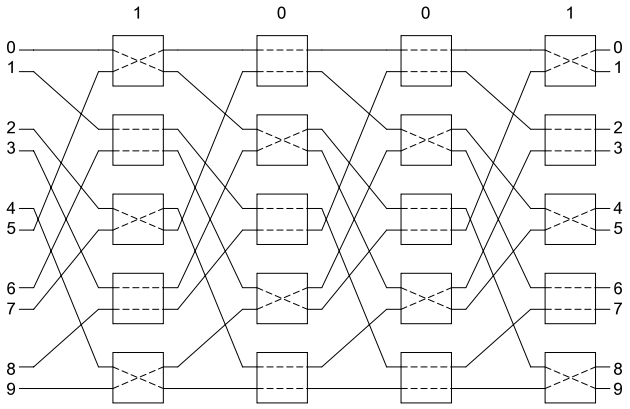


Fig. 7. Applying alternating stage control on a 10×10 GSEN; the shown network configuration is $A = 9 = (1001)_2$.

For convenience, if a number is in $\{0, 1, 2, \ldots, 2^n - 1\}$ but not in $\mathcal{B}_F$, then we call it a *hole* of $\mathcal{B}_F$. The following lemma shows that the elements of $\mathcal{B}_F$ are distributed *very uniformly* in $\{0, 1, 2, \ldots, 2^n - 1\}$.

Lemma 3.7. *For all $F \in \{0, 1, 2, \ldots, 2^n - 1\}$, $\mathcal{B}_F$ has no two consecutive holes.*

Proof. We will prove this lemma by showing that $B(0, F) \leq 1$, $B(N-1, F) \geq 2^n - 2$, and

$$B(i-1, F) + 1 \leq B(i, F) \leq B(i-1, F) + 2 \text{ for } i = 1, 2, \ldots, N-1.$$

Recall that $2^{n-1} < N \leq 2^n$ and $0 \leq F < 2^n$. By Lemma 3.3, $B(0, F) = \left\lfloor \frac{F}{N} \right\rfloor \leq 1$. Also, $B(N-1, F) = \left\lfloor \frac{(N-1)\cdot 2^n + F}{N} \right\rfloor \geq \left\lfloor \frac{(N-1)\cdot 2^n}{N} \right\rfloor \geq 2^n - 2$. Finally, consider i, where $1 \leq i \leq N-1$. By Lemma 3.3, $B(i-1, F) + 1 = \left\lfloor \frac{(i-1)\cdot 2^n + F}{N} \right\rfloor + 1 = \left\lfloor \frac{i\cdot 2^n + F}{N} - \frac{2^n}{N} \right\rfloor + 1 \leq \left\lfloor \frac{i\cdot 2^n + F}{N} \right\rfloor = B(i, F) = \left\lfloor \frac{(i-1)\cdot 2^n + F}{N} + \frac{2^n}{N} \right\rfloor \leq \left\lfloor \frac{(i-1)\cdot 2^n + F}{N} \right\rfloor + 2 = B(i-1, F) + 2$. □

Now we are ready to prove the main result of this section.

Theorem 3.8. $N \leq \mathcal{R}(N) \leq \mathcal{R}_{sc}(N) = 2^n$.

Proof. By Lemma 3.1, it suffices to prove that $\mathcal{R}_{sc}(N) \geq 2^n$. When the stage control technique is assumed, there are only 2^n possible network configurations: $0, 1, \ldots, 2^n - 1$. Thus to prove that $\mathcal{R}_{sc}(N) \geq 2^n$, it suffices to prove that each of the 2^n possible network configurations is required for every processor to receive N messages.

When the stage control technique is assumed, the network configuration C can be determined by an arbitrary path P set up by C. Moreover, if F and B are the forward and backward control tags used by P, then Lemma 3.4 tells us that $C = B \oplus F$. In the following, we will prove that for each C in $\{0, 1, \ldots, 2^n - 1\}$, at least one of the paths set up by C is a unique path and therefore C must be used in all-to-all personalized exchange. Suppose to the contrary there is a $\hat{C}$ in $\{0, 1, \ldots, 2^n - 1\}$ such that none of the paths set up by $\hat{C}$ is a unique path. Then consider $2^{n-1} \oplus \hat{C}$ and let $\hat{B} = 2^{n-1} \oplus \hat{C}$; consider $(2^{n-1} + 1) \oplus \hat{C}$ and let $\hat{B}' = (2^{n-1} + 1) \oplus \hat{C}$. We claim that $\hat{B} \notin \mathcal{B}_{2^{n-1}}$ and $\hat{B}' \notin \mathcal{B}_{2^{n-1}+1}$. Suppose this clam is not true. Then either $\hat{B} \in \mathcal{B}_{2^{n-1}}$ or $\hat{B}' \in \mathcal{B}_{2^{n-1}+1}$ or both. Suppose $\hat{B} \in \mathcal{B}_{2^{n-1}}$. Since $\hat{C} = \hat{B} \oplus 2^{n-1}$, by Lemma 3.5, $\hat{C}$ conducts a unique path, which contradicts with the assumption that none of the paths set up by $\hat{C}$ is a unique path. The case that $\hat{B}' \in \mathcal{B}_{2^{n-1}+1}$ can be proven similarly. Now we have the claim that $\hat{B} \notin \mathcal{B}_{2^{n-1}}$ and $\hat{B}' \notin \mathcal{B}_{2^{n-1}+1}$. By Lemma 3.6, $\mathcal{B}_{2^{n-1}} = \mathcal{B}_{2^{n-1}+1}$. Thus $\hat{B}' \notin \mathcal{B}_{2^{n-1}}$. Since $\hat{B}$ and $\hat{B}'$ differ by 1, they are two consecutive holes in $\mathcal{B}_{2^{n-1}}$; this contradicts with Lemma 3.7. Thus for each C in $\{0, 1, \ldots, 2^n - 1\}$, at least one of the paths set up by C is a unique path and therefore C must be used in all-to-all personalized exchange. So $\mathcal{R}_{sc}(N) \geq 2^n$. □

4. All-to-all personalized exchange of GSENs with $N \equiv 2 \pmod 4$

Throughout this section, unless other specified, subports 0 and 1 are the subports 0 and 1 on the right-hand side of a switch. We will propose an optimal all-to-all personalized exchange algorithm for GSENs with $N \equiv 2 \pmod 4$ and prove that $N = \mathcal{R}(N) < \mathcal{R}_{sc}(N) = 2^n$ if $N \equiv 2 \pmod 4$. We first introduce a variation of the stage control technique and we call it *alternating stage control*, meaning that the states of the switches of a stage *alternate between straight and cross*. See Fig. 7 for an illustration.

When alternating stage control is used, the network configuration of a GSEN can be represented by a number as follows. Let a_ℓ denote the states of the switches at stage $n-1-\ell$ such that

- $a_\ell = 0$ means the states are 0, 1, 0, 1, and so on;
- $a_\ell = 1$ means the states are 1, 0, 1, 0, and so on.

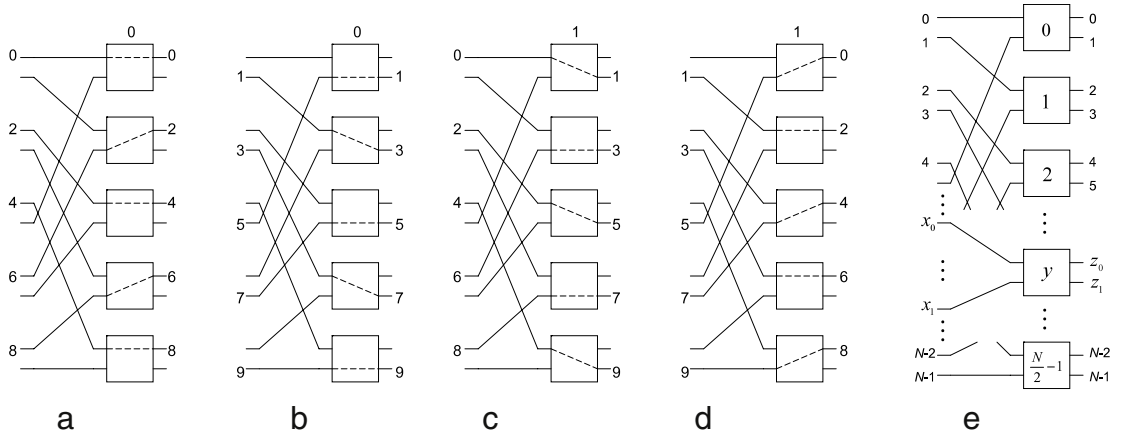


Fig. 8. (a) and (b): a stage in a 10×10 GSEN when $a_\ell = 0$. (c) and (d): a stage in a 10×10 GSEN when $a_\ell = 1$. (e) An illustration for the proof of Property $(*)$.

The network configuration of the GSEN can be represented by the number

$$A = a_{n-1}2^{n-1} + a_{n-2}2^{n-2} + \cdots + a_0 2^0$$

or $(a_{n-1}\ a_{n-2}\ \ldots\ a_1\ a_0)_2$ in the binary form. Clearly, $0 \le A < 2^n$. We will call A an *alternating configuration*. When $N \equiv 2 \pmod 4$ and alternating stage control is used, the N input terminals and N output terminals of stage $n-1-\ell$ have the following property.

Property $(*)$ (*see Fig. 8 (a)–(d) for an illustration*).

1. *If $a_\ell = 0$, then $Even \xrightarrow{0} Even$, $Odd \xrightarrow{1} Odd$. That is, every even-numbered input terminal is connected to an even-numbered output terminal via subport 0, and every odd-numbered input terminal is connected to an odd-numbered output terminal via subport 1.*
2. *If $a_\ell = 1$, then $Even \xrightarrow{1} Odd$, $Odd \xrightarrow{0} Even$. That is, every even-numbered input terminal is connected to an odd-numbered output terminal via subport 1, and every odd-numbered input terminal is connected to an even-numbered output terminal via subport 0.*

Proof. Consider an arbitrary stage of a GSEN and an arbitrary switch y of this stage; see Fig. 8(e) for an illustration. Suppose input terminals x_0 and x_1 are connected to subports 0 and 1 of switch y, respectively. By the definition of a GSEN, $x_0 = y$ and $x_1 = y + \frac{N}{2}$ hold. Note that $\frac{N}{2}$ is an odd number since $N \equiv 2 \pmod 4$. Since $\frac{N}{2}$ is odd, one of x_0 and x_1 is even and the other one is odd. Now consider the output terminals z_0 and z_1 of switch y. Then z_0 is even and z_1 is odd.

Suppose $a_\ell = 0$ and y is even. Then x_0 is even (by the fact that $x_0 = y$) and x_0 is connected to z_0 (due to the setting of a_ℓ). Thus every even-numbered input terminal is connected to an even-numbered output terminal via subport 0. Now suppose $a_\ell = 0$ and y is odd. Then x_0 is odd (by the fact that $x_0 = y$) and x_0 is connected to z_1 (due to the setting of a_ℓ). Thus every odd-numbered input terminal is connected to an odd-numbered output terminal via subport 1. The case of $a_\ell = 1$ can be proven similarly. □

Do notice that Property $(*)$ holds only when $N \equiv 2 \pmod 4$ holds. Now we give other properties of alternating stage control.

Lemma 4.1. *Suppose $N \equiv 2 \pmod 4$ and alternating stage control is used. Then the following statements hold:*

1. *The forward control tags of even-numbered inputs are identical.*
2. *The forward control tags of odd-numbered inputs are identical.*

Proof. Let $A = (a_{n-1}\ a_{n-2}\ \cdots\ a_0)_2$ be the alternating configuration used. By Property $(*)$, all the messages sent out from inputs 0, 2, 4, …, $N-2$ are via subports 0 of switches at stage $n-1-\ell$ if $a_\ell = 0$ and via subports 1 of switches at stage $n-1-\ell$ if $a_\ell = 1$. Thus statement 1 holds. By Property $(*)$, all the messages sent out from inputs 1, 3, 5, …, $N-1$ are via subports 1 of switches at stage $n-1-\ell$ if $a_\ell = 0$ and via subports 0 of switches at stage $n-1-\ell$ if $a_\ell = 1$. Thus we have statement 2. □

Theorem 4.2. *Suppose $N \equiv 2 \pmod 4$ and alternating stage control is used. Let A be a given alternating configuration, F be the forward control tag of any even-numbered input, and $\overline{F}$ be the forward control tag of any odd-numbered input. Then:*

(i) $F \oplus \overline{F} = (11\cdots1)_2$;
(ii) $A = F \oplus \left\lfloor \frac{F}{2} \right\rfloor$;
(iii) $F = A \oplus \left\lfloor \frac{A}{2} \right\rfloor \oplus \left\lfloor \frac{A}{2^2} \right\rfloor \oplus \cdots \oplus \left\lfloor \frac{A}{2^{n-1}} \right\rfloor$.

Proof. First consider (i). Let $F = (f_{n-1}\, f_{n-2}\, \cdots\, f_0)_2$ and $A = (a_{n-1}\, a_{n-2}\, \cdots\, a_0)_2$. By Property (∗), if messages from even-numbered inputs are via subport f_ℓ at stage $n-1-\ell$, then messages from inputs odd are via subport $1-f_\ell$ at stage $n-1-\ell$, ($\ell = n-1, n-2, \ldots, 0$). Thus $F \oplus \overline{F} = (11\cdots1)_2$. Now consider (ii). Clearly, $a_{n-1} = f_{n-1}$. For $\ell = n-2, n-3, \ldots, 0$, by Property (∗), we have:

- If $a_\ell = 0$, then $f_\ell = 0$ whenever $f_{\ell+1} = 0$ and $f_\ell = 1$ whenever $f_{\ell+1} = 1$.
- If $a_\ell = 1$, then $f_\ell = 0$ whenever $f_{\ell+1} = 1$ and $f_\ell = 1$ whenever $f_{\ell+1} = 0$.

Thus $a_\ell = f_\ell \oplus f_{\ell+1}$ for $\ell = n-2, n-3, \ldots, 0$. Therefore

$$\begin{aligned} A = (a_{n-1}\, a_{n-2}\, \ldots\, a_1\, a_0)_2 &= (f_{n-1}\ f_{n-2}\oplus f_{n-1}\ f_{n-3}\oplus f_{n-2}\ \ldots\ f_0\oplus f_1)_2 \\ &= (f_{n-1}\oplus 0\ f_{n-2}\oplus f_{n-1}\ f_{n-3}\oplus f_{n-2}\ \ldots\ f_0\oplus f_1)_2 \\ &= (f_{n-1}\ f_{n-2}\ f_{n-3}\ \ldots\ f_0)_2 \oplus (0\ f_{n-1}\ f_{n-2}\ \ldots\ f_1)_2 = F \oplus \left\lfloor \frac{F}{2} \right\rfloor. \end{aligned}$$

Finally, consider (iii). Then $f_\ell = a_\ell \oplus a_{\ell+1} \oplus \cdots \oplus a_{n-1}$ for $\ell = n-2, n-3, \ldots, 0$. Thus $F = A \oplus \lfloor \frac{A}{2} \rfloor \oplus \lfloor \frac{A}{2^2} \rfloor \oplus \cdots \oplus \lfloor \frac{A}{2^{n-1}} \rfloor$. □

Theorem 4.2(ii) gives a one-to-one correspondence between A and F; for convenience, let A_F denote the alternating configuration corresponding to F. When $F = k$,

$$A_k = k \oplus \left\lfloor \frac{k}{2} \right\rfloor.$$

Lemma 4.3. *If $N \equiv 2 \pmod 4$ and the given GSEN is set by alternating configuration A_k, then the forward control tags of even-numbered inputs are k and the forward control tags of odd-numbered inputs are $2^n - 1 - k$.*

Proof. This lemma follows from Lemma 4.1, $A_k = k \oplus \lfloor \frac{k}{2} \rfloor$, and Theorem 4.2(i). □

Now we prove a theorem, which is the foundation of our algorithms.

Theorem 4.4. *Suppose $N \equiv 2 \pmod 4$. Then the N alternating configurations $A_0, A_1, \ldots, A_{N-1}$ ensure that every input i can get to every output j; in other words, $A_0, A_1, \ldots, A_{N-1}$ can fulfill an all-to-all communication in a GSEN.*

Proof. Let i be an arbitrary input. For $k = 0, 1, \ldots, N-1$, let j_k be the destination of i when the network configuration is set according to A_k. First consider the case that i is even. By Lemmas 3.3 and 4.3, $j_k = (i \cdot 2^n + k) \mod N$. Since $A_0, A_1, \ldots, A_{N-1}$ ensure that k varies from 0 to $N-1$ and $j_k = (i \cdot 2^n + k) \mod N$, it follows that i can get to every output. Now consider the case that i is odd. By Lemmas 3.3 and 4.3, $j_k = (i \cdot 2^n + 2^n - 1 - k) \mod N$. Since $A_0, A_1, \ldots, A_{N-1}$ ensure that k varies from 0 to $N-1$ and $j_k = (i \cdot 2^n + 2^n - 1 - k) \mod N$, it follows that i can get to every output. □

For example, the 10 alternating configurations $A_0 = 0, A_1 = 1, A_2 = 3, A_3 = 2, A_4 = 6, A_5 = 7, A_6 = 5, A_7 = 4, A_8 = 12, A_9 = 13$ can fulfill an all-to-all communication in a 10×10 GSEN. Note that $A_0, A_1, \ldots, A_{N-1}$ are not the only way to fulfill an all-to-all communication in a GSEN. In fact, any N consecutive integers in $0, 1, \ldots, 2^n - 1$ can fulfill an all-to-all communication.

The purpose of this paper is to propose an optimal all-to-all personalized exchange algorithm for GSENs. However, since there is no all-to-all broadcast algorithm for GSENs, we will also propose one. Therefore, in the following, three algorithms will be proposed. The first algorithm fulfills all-to-all broadcast in GSENs. The second algorithm gives a preprocessing of the third algorithm. And the third algorithm fulfills all-to-all personalized exchange in GSENs.

Algorithm 1 : an algorithm to fulfill all-to-all broadcast in a GSEN with $N \equiv 2 \pmod 4$

```
1: for each processor i (0 ≤ i < N) do in parallel
2:   Processor i prepares a broadcast message;
3:   for k = 0 to N − 1 do in sequential
4:     Equip the broadcast message of processor i with the forward control tag k if i is even and 2^n − 1 − k if i is odd;
5:     Transmit the message;
6:   endfor
7: endfor
```

The correctness of Algorithm 1 follows from Lemma 4.3 and Theorem 4.4. The communication delay of Algorithm 1 is $O(N + n)$ since each of the N processors can receive its first message in n rounds and receive the remaining $N-1$ messages in $N-1$ rounds. By Lemma 3.2, Algorithm 1 is optimal.

All-to-all personalized exchange is much more complicated than all-to-all broadcast. In all-to-all personalized exchange, a source has to prepare a personalized message for each of its N destinations. Therefore, before a message is sent out, the source of the message has to know which output will be its current destination so that a personalized message can be prepared. Algorithm 2 is designed to overcome this difficulty. This algorithm constructs a matrix called *destination matrix* $D = (d_{i,k})$ so that $d_{i,k} = j$ if and only if the message sent out from processor i at round k arrives processor j.

The following theorem proves the correctness and gives the time complexity of Algorithm 2.

Algorithm 2 : an algorithm to construct the destination matrix $D = (d_{i,k})$ for a GSEN with $N \equiv 2 \pmod 4$

```
1: n ← ⌈log2 N⌉;
2: power ← 2^n;
3: for each processor i (0 ≤ i < N) do in sequential
4:   if i is even then m ← (i · power) mod N; else m ← ((i + 1) · power − 1) mod N; endif
5:   for k = 0 to N − 1 do in sequential
6:     if i is even then d_{i,k} ← (m + k) mod N; else d_{i,k} ← (m − k) mod N; endif
7:   endfor
8: endfor
```

Theorem 4.5. *Algorithm 2 constructs a matrix $D = (d_{i,k})$ so that $d_{i,k} = j$ if and only if the message sent out from processor i at round k arrives processor j. Moreover, it takes $O(N^2)$ time.*

Proof. To prove the correctness of Algorithm 2, it suffices to show that the message sent out from processor i at round k (see Algorithm 3 for round k) arrives processor $(m + k) \bmod N$ if i is even and arrive processor $(m - k) \bmod N$ if i is odd. Note that in Algorithm 3, we will use $A_0, A_1, \ldots, A_{N-1}$ to fulfill all-to-all personalized exchange. By Lemma 4.3, A_k contributes an even-numbered processor i the forward control tag k and it contributes an odd-numbered processor i the forward control tag $2^n - 1 - k$. Suppose i is even. Then at round k, the message sent out from processor i will be equipped with the forward control tag k; by Lemma 3.3, the destination is

$$j = (i \cdot 2^n + k) \bmod N = (i \cdot \text{power} + k) \bmod N = (m + k) \bmod N.$$

Now suppose i is odd. By Lemma 4.3, the message sent out from processor i will be equipped with the forward control tag $2^n - 1 - k$; by Lemma 3.3, the destination is

$$j = (i \cdot 2^n + 2^n - 1 - k) \bmod N = ((i + 1) \cdot \text{power} - 1 - k) \bmod N = (m - k) \bmod N.$$

It is not difficult to see that Algorithm 2 takes $O(N^2)$ time. We have this theorem. □

Consider the GSEN in Fig. 2 for an example of Algorithm 2. Then the matrix D constructed is:

$$D = \begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 1 & 0 & 9 & 8 & 7 & 6 & 5 & 4 & 3 & 2 \\ 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 0 & 1 \\ 3 & 2 & 1 & 0 & 9 & 8 & 7 & 6 & 5 & 4 \\ 4 & 5 & 6 & 7 & 8 & 9 & 0 & 1 & 2 & 3 \\ 5 & 4 & 3 & 2 & 1 & 0 & 9 & 8 & 7 & 6 \\ 6 & 7 & 8 & 9 & 0 & 1 & 2 & 3 & 4 & 5 \\ 7 & 6 & 5 & 4 & 3 & 2 & 1 & 0 & 9 & 8 \\ 8 & 9 & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 9 & 8 & 7 & 6 & 5 & 4 & 3 & 2 & 1 & 0 \end{bmatrix}.$$

Note that the matrix D needs to be constructed only once and therefore can be viewed as one of the system parameters. Thus the time complexity of Algorithm 2 is *not* included in the communication delay. Now we are ready to propose our all-to-all personalized exchange algorithm; see Algorithm 3.

Algorithm 3 : an algorithm to fulfill all-to-all personalized exchange in a GSEN with $N \equiv 2 \pmod 4$

```
1: for each processor i (0 ≤ i < N) do in parallel
2:   for k = 0 to N − 1 do in sequential //comment: round k
3:     Processor i prepares a personalized message for processor d_{i,k};
4:     Equip the personalized message with the forward control tag k if i is even and 2^n − 1 − k if i is odd;
5:     Transmit the message;
6:   endfor
7: endfor
```

The following theorem proves the correctness and gives the time complexity of Algorithm 3.

Theorem 4.6. *Algorithm 3 fulfills all-to-all personalized exchange in a GSEN with $N \equiv 2 \pmod 4$. Moreover, it takes $O(N + n)$ time.*

Proof. Algorithm 3 prepares a personalized message according to the matrix D, which is constructed by Algorithm 2. Thus, by Theorems 4.4 and 4.5, Algorithm 3 fulfills all-to-all personalized exchange for GSENs with $N \equiv 2 \pmod 4$. This algorithm takes $O(N + n)$ time since each of the N processors can receive its first personalized message in n rounds and receive the remaining $N - 1$ personalized messages in $N - 1$ rounds. □

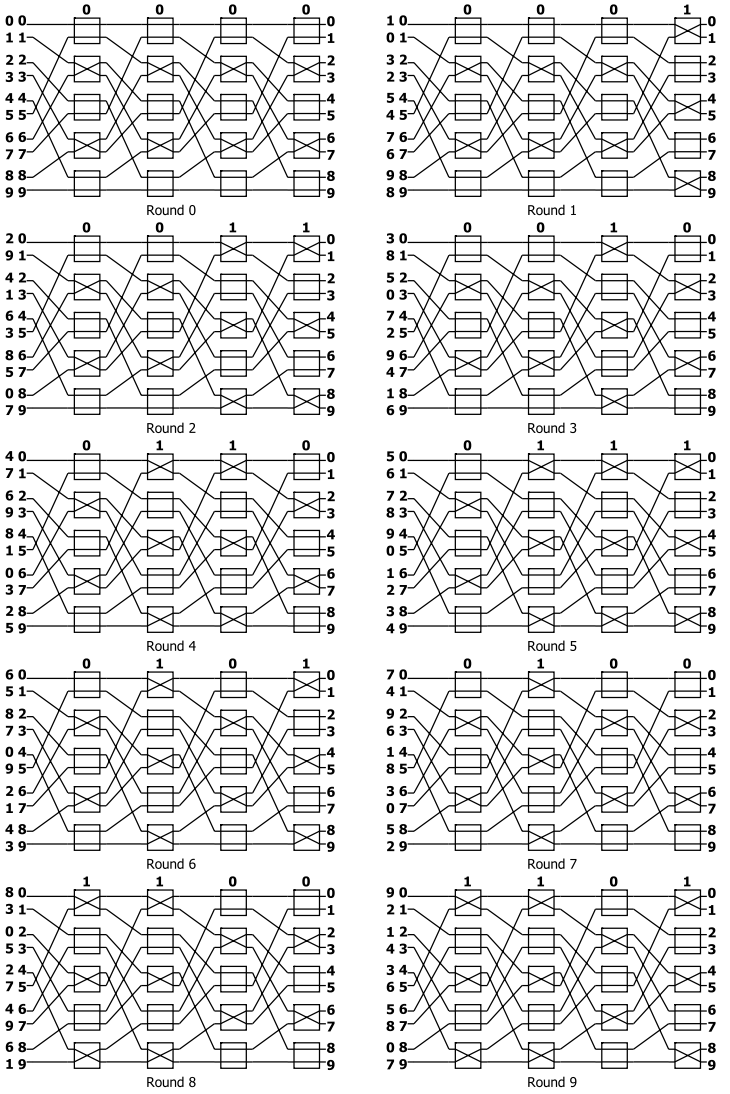


Fig. 9. An example of Algorithm 3.

By Lemma 3.2 and Theorem 4.6, we have the following corollary.

Corollary 4.7. *Algorithm 3 is optimal.*

Fig. 9 shows how Algorithm 3 fulfills all-to-all personalized exchange for the GSEN in Fig. 2. Take round 2 in Fig. 9 for an example. The 0-1 bits 0011 above stages 0, 1, 2, 3 denote the alternating configuration for round 2, which is $(0011)_2 = 3 = A_2$. The numbers on the left-hand side denote the destinations of personalized messages. Thus, at round 2, processor 0 sends a personalized message to processor 2, processor 1 sends a personalized message to processor 9, processor 2 sends a personalized message to processor 4, . . . , and processor 9 sends a personalized message to processor 7. Recall that $A_0 = 0, A_1 = 1, A_2 = 3, A_3 = 2, A_4 = 6, A_5 = 7, A_6 = 5, A_7 = 4, A_8 = 12, A_9 = 13$ can fulfill an all-to-all communication in a 10×10 GSEN. The 10 alternating configurations and the destinations of the messages are shown on the left-hand side of the GSEN for rounds 0, 1, . . . , 9 in Fig. 9.

Note that it is possible to combine Algorithms 2 and 3 and to avoid the construction of matrix D. See Algorithm 4 below. Now we end this section by proving the following theorem.

Theorem 4.8. $N = \mathcal{R}(N) < \mathcal{R}_{sc}(N) = 2^n$ if $N \equiv 2 \pmod 4$.

Proof. Since Algorithm 3 can fulfill all-to-all personalized exchange by using N network configurations, namely, $A_0, A_1, \ldots, A_{N-1}$, we have $\mathcal{R}(N) \leq N$. By Theorem 3.8 and by the fact that $\mathcal{R}(N) \leq N$ for $N \equiv 2 \pmod 4$, we have this theorem. □

Algorithm 4 : yet another algorithm to fulfill all-to-all personalized exchange in a GSEN with $N \equiv 2 \pmod 4$

```
1: for each processor i (0 ≤ i < N) do in parallel
2:   n ← ⌈log2 N⌉;
3:   power ← 2^n;
4:   if i is even then m ← (i · power) mod N; else m ← ((i + 1) · power − 1) mod N; endif
5:   for k = 0 to N − 1 do in sequential //comment: round k
6:     if i is even then j ← (m + k) mod N; else j ← (m − k) mod N; endif
7:     Processor i prepares a personalized message for processor j;
8:     Equip the personalized message with the forward control tag k if i is even and 2^n − 1 − k if i is odd;
9:     Transmit the message;
10:  endfor
11: endfor
```

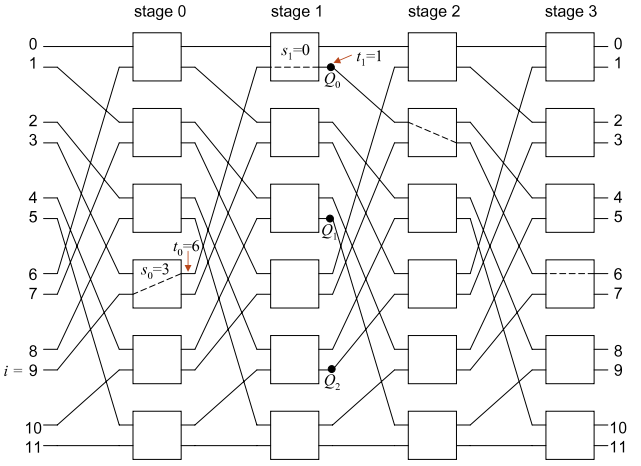


Fig. 10. A 12 × 12 GSEN, switches s_0 and s_1, terminals t_0 and t_1, and $\mathcal{Q} = \{Q_0, Q_1, Q_2\}$.

5. The value of $\mathcal{R}(N)$ when $N \equiv 0 \pmod 4$

The purpose of this section is to obtain $\mathcal{R}(N)$ for all $N \equiv 0 \pmod 4$. Recall that each stage of a GSEN consists of the perfect shuffle on N terminals followed by $N/2$ switches, the N terminals are numbered $0, 1, \ldots, N-1$, and the perfect shuffle operation on the N terminals is the permutation π defined by $\pi(i) = (2 \cdot i + \lfloor \frac{2 \cdot i}{N} \rfloor) \bmod N, \quad 0 \le i < N$. We first have a lemma.

Lemma 5.1. *Suppose $k \ge 2$, $N \equiv 0 \pmod{2^k}$, and $N \not\equiv 0 \pmod{2^{k+1}}$. Let i be an arbitrary input of a given $N \times N$ GSEN. If the forward control tag $F = f_{n-1}2^{n-1} + f_{n-2}2^{n-2} + \cdots + f_0 2^0$ used by i starts with $f_{n-1} = 0$ and $f_{n-t} = 1$ (for $t = 2, 3, \ldots, k$), then the terminal reached by i immediately after stage $k-1$ is $(i2^k + 2^{k-1} - 1) \bmod N$.*

Proof. Each stage of a GSEN has $N/2$ switches; we suppose these $N/2$ switches are labeled $0, 1, \ldots, N/2-1$. Consider the path $P(i, F)$ and the switches and terminals on the path. Let s_ℓ be the label of the switch at stage ℓ reached by $P(i, F)$. Let t_ℓ be the terminal immediately after stages ℓ that is reached by $P(i, F)$. See Fig. 10 for an illustration of the $N = 12$ and $k = 2$ case. By the perfect shuffle operation, $s_0 = i \bmod N/2$. Since $f_{n-1} = 0$, we have $t_0 = 2s_0 = (2i) \bmod N$. Again, by the perfect shuffle operation, $s_1 = t_0 \bmod N/2 = (2i) \bmod N/2$. Since $f_{n-2} = 1$, we have $t_1 = 2s_1 + 1 = (4i + 1) \bmod N = (i2^2 + 2^1 - 1) \bmod N$. In general, we assume $\ell \ge 1$. Then we have $s_\ell = t_{\ell-1} \bmod N/2$ and $t_\ell = 2s_\ell + f_{n-1-\ell}$. Continuing in this way, we have

$$t_{k-1} = (i2^k + f_{n-1}2^{k-1} + \cdots + f_{n-k}2^0) \bmod N = (i2^k + 2^{k-2} + \cdots + 2^0) \bmod N = (i2^k + 2^{k-1} - 1) \bmod N.$$

Hence this lemma holds. □

For $r = 0, 1, \ldots, \frac{N}{2^k} - 1$, let Q_r denote the terminal $r2^k + 2^{k-1} - 1$ immediately after stage $k-1$ and see Fig. 10 for an illustration of the $N = 12$ and $k = 2$ case. Let $\mathcal{Q} = \{Q_r \mid r = 0, 1, \ldots, \frac{N}{2^k} - 1\}$. We say a routing path *passes through* $\mathcal{Q}$ if it passes through one of the terminals in $\mathcal{Q}$.

Lemma 5.2. *Suppose $k \ge 2$, $N \equiv 0 \pmod{2^k}$, $N \not\equiv 0 \pmod{2^{k+1}}$, and consider an $N \times N$ GSEN. A routing path passes through $\mathcal{Q}$ if and only if the forward control tag $F = f_{n-1}2^{n-1} + f_{n-2}2^{n-2} + \cdots + f_0 2^0$ used by this path starts with $f_{n-1} = 0$ and $f_{n-t} = 1$ (for $t = 2, 3, \ldots, k$).*

Proof. Assume that the given routing path is from input i. Then this routing path is the path $P(i, F)$.
(Necessity) First suppose $P(i, F)$ passes through the terminal Q_r in $\mathcal{Q}$. Then by the perfect shuffle operation, we have

$$(i2^k + f_{n-1}2^{k-1} + \cdots + f_{n-k}) \bmod N = r2^k + 2^{k-1} - 1.$$

Since $2^k | N$, we can take modulo 2^k for both sides of the above equation and obtain

$$(i2^k + f_{n-1}2^{k-1} + \cdots + f_{n-k} \bmod N) \bmod 2^k = (r2^k + 2^{k-1} - 1) \bmod 2^k = 2^{k-1} - 1,$$

which implies $f_{n-1}2^{k-1} + f_{n-2}2^{k-2} + \cdots + f_{n-k} = 2^{k-1} - 1$, i.e., $f_{n-1} = 0$ and $f_{n-t} = 1$ (for $t = 2, 3, \ldots, k$).
(Sufficiency) Suppose the forward control tag F starts with $f_{n-1} = 0$ and $f_{n-t} = 1$ (for $t = 2, 3, \ldots, k$). Then by Lemma 5.1, the terminal reached by i immediately after stage $k-1$ will be $(i2^k + 2^{k-1} - 1) \bmod N$, which is $Q_{i \bmod \frac{N}{2^k}}$. Therefore $P(i, F)$ passes through $\mathcal{Q}$. □

Recall that $2^{n-1} < N \leq 2^n$. The following lemma requires N to satisfy $2^{n-1} + 2^{n-k} \leq N \leq 2^n$.

Lemma 5.3. *Suppose $k \geq 2$, $N \equiv 0 \pmod{2^k}$, $N \not\equiv 0 \pmod{2^{k+1}}$, and consider an $N \times N$ GSEN. If $2^{n-1} + 2^{n-k} \leq N \leq 2^n$ and the forward control tag $F = f_{n-1}2^{n-1} + f_{n-2}2^{n-2} + \cdots + f_0 2^0$ used by a path starts with $f_{n-1} = 0$ and $f_{n-t} = 1$ (for $t = 2, 3, \ldots, k$), then this path is a unique path.*

Proof. Note that if F starts with $f_{n-1} = 0$ and $f_{n-t} = 1$ (for $t = 2, 3, \ldots, k$), then $2^{n-1} - 2^{n-k} \leq F < 2^{n-1}$. Assume that the given routing path is from input i. Then this routing path is the path $P(i, F)$. By Lemma 3.5, $P(i, F)$ is a unique path if and only if $2^n - N \leq F < N$. Since

$$2^n - N \leq 2^n - 2^{n-1} - 2^{n-k} = 2^{n-1} - 2^{n-k} \leq F < 2^{n-1} < N,$$

$P(i, F)$ is a unique path for each $2^{n-1} - 2^{n-k} \leq F < 2^{n-1}$. Hence this lemma holds. □

Now we are ready to propose our result for $\mathcal{R}(N)$ with $N \equiv 0 \pmod 4$.

Theorem 5.4. *$\mathcal{R}(N) = \mathcal{R}_{sc}(N) = 2^n$ if $k \geq 2$, $N \equiv 0 \pmod{2^k}$, $N \not\equiv 0 \pmod{2^{k+1}}$, and $2^{n-1} + 2^{n-k} \leq N \leq 2^n$.*

Proof. Assume $k \geq 2$, $N \equiv 0 \pmod{2^k}$, $N \not\equiv 0 \pmod{2^{k+1}}$, and $2^{n-1} + 2^{n-k} \leq N \leq 2^n$. By Theorem 3.8, it suffices to prove that $\mathcal{R}(N) \geq 2^n$. In any all-to-all communication of a GSEN, a total of N^2 routing paths have to be established. Let i be an arbitrary input and let $F = f_{n-1}2^{n-1} + f_{n-2}2^{n-2} + \cdots + f_0 2^0$ be an arbitrary forward control tag such that F starts with $f_{n-1} = 0$ and $f_{n-t} = 1$ (for $t = 2, 3, \ldots, k$). Since F starts with $f_{n-1} = 0$ and $f_{n-t} = 1$ (for $t = 2, 3, \ldots, k$), we have $2^{n-1} - 2^{n-k} \leq F < 2^{n-1}$ and there are a total of 2^{n-k} such F's. By Lemma 5.3, $P(i, F)$ is a unique path. Since $2^{n-1} - 2^{n-k} \leq F < 2^{n-1}$, the number of such unique paths $P(i, F)$ is $N \cdot 2^{n-k}$. Let $\mathcal{U}$ denote the set of these $N \cdot 2^{n-k}$ unique paths. Then, in any all-to-all communication, all of the paths in $\mathcal{U}$ must appear. By Lemma 5.2, all of the paths in $\mathcal{U}$ will pass through $\mathcal{Q}$. Recall that given a network configuration, a permutation between the inputs and outputs can be obtained. Therefore, given a network configuration, N routing paths can be established. By Lemma 5.1, any network configuration can establish only $N/2^k$ routing paths in $\mathcal{U}$. Therefore $\mathcal{R}(N) \geq \frac{N \cdot 2^{n-k}}{N/2^k} = 2^n$. □

By Theorem 5.4, $\mathcal{R}(12) = 16$, $\mathcal{R}(24) = 32$, $\mathcal{R}(28) = 32$, $\mathcal{R}(40) = 64$, $\mathcal{R}(80) = 128$, and $\mathcal{R}(144) = 256$. The first $\mathcal{R}(N)$ that cannot be determined by Theorems 4.8 and 5.4 is $\mathcal{R}(20)$; we will determine it after introducing a variation of the alternating stage control technique; we call it *doubly alternating stage control*, meaning that the states of the switches of a stage *alternate between two straight states and two cross states*. The network configuration obtained by doubly alternating stage control is called a *doubly alternating configuration* and it can be represented by the number

$$A' = a'_{n-1}2^{n-1} + a'_{n-2}2^{n-2} + \cdots + a'_0 2^0$$

as follows. Let a'_ℓ denote the states of the switches at stage $n-1-\ell$ such that

- $a'_\ell = 0$ means the states are 0, 0, 1, 1, 0, 0, 1, 1, and so on.
- $a'_\ell = 1$ means the states are 1, 1, 0, 0, 1, 1, 0, 0, and so on.

Obviously, $0 \leq A' < 2^n$. Now we are ready to determine $\mathcal{R}(20)$.

Theorem 5.5. *$\mathcal{R}(20) = 24$.*

Proof. We first prove that $\mathcal{R}(20) \geq 24$. In any all-to-all communication of a 20×20 GSEN, a total of $20^2 = 400$ routing paths have to be established. To prove $\mathcal{R}(20) \geq 24$, we claim that 400 routing paths are not sufficient to fulfill an all-to-all communication in a 20×20 GSEN and at least $400 + 80 = 480$ routing paths have to be established in order to fulfill an all-to-all communication. If this claim is true, then since a network configuration can establish only 20 routing paths, we have $\mathcal{R}(20) \geq \frac{480}{20} = 24$. Now we prove this claim.

Let i be an arbitrary input and let $F = f_{n-1}2^{n-1} + f_{n-2}2^{n-2} + \cdots + f_0 2^0$ be an arbitrary forward control tag. By Lemma 3.5, $P(i, F)$ is a unique path if and only if $12 \leq F \leq 19$. Hence each input i contributes 8 unique paths $P(i, 12)$, $P(i, 13), \ldots, P(i, 19)$. Thus there are a total of 160 unique paths; we illustrate all of these 160 unique paths in Fig. 11. In this proof, states of switches at stage 2 play an important role. Denote the 10 switches at stage 2 by $S_0, S_1, \ldots, S_9$. Now we

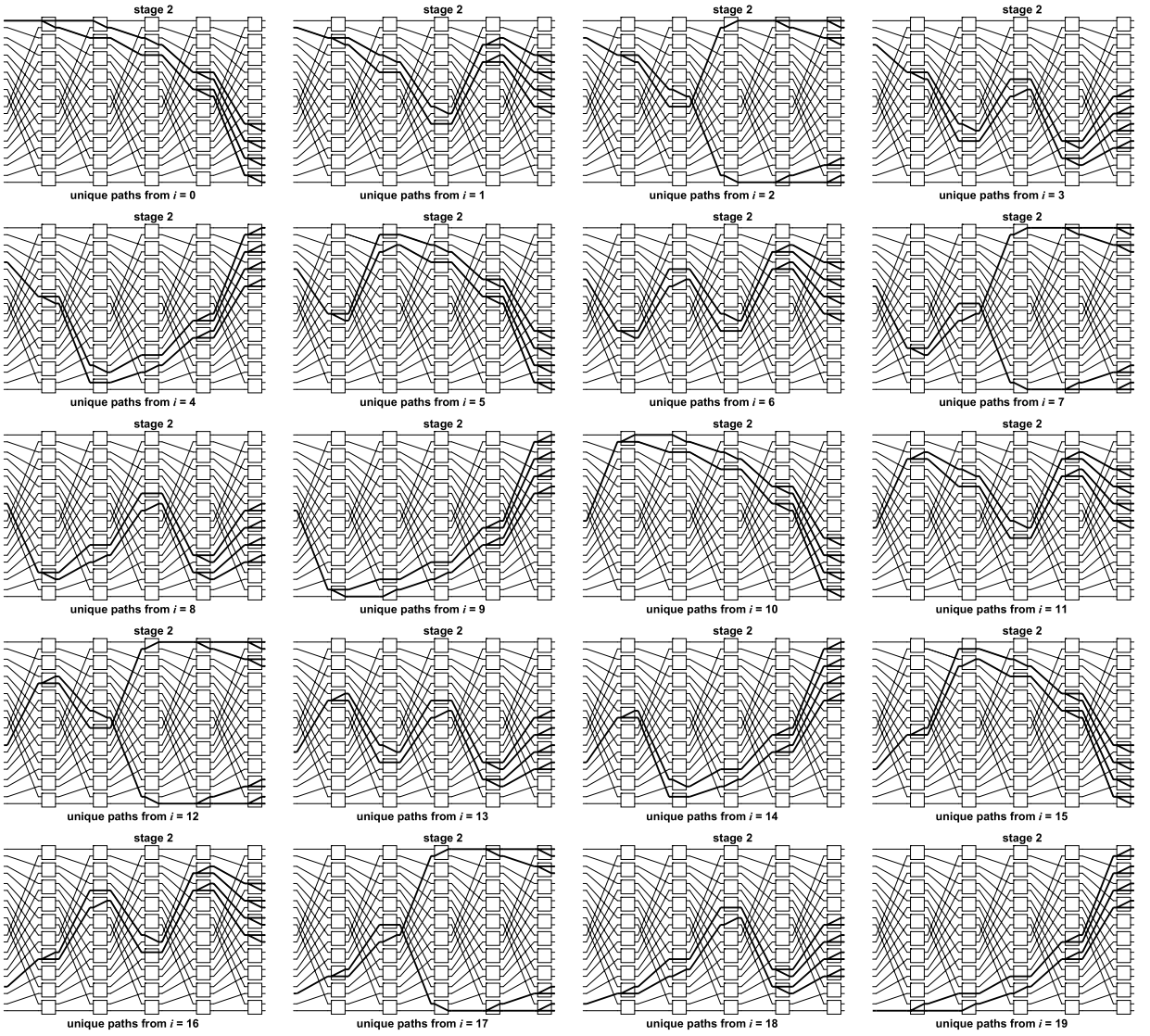


Fig. 11. The 160 unique paths in a 20 × 20 GSEN; each input i contributes 8 unique paths.

define types 00, 01, 10, and 11, according to the connection inside a switch at stage 2 as follows. A path is said to be *of type* xy, where x,y ∈ {0,1}, if the connection inside the switch (passed by the path) at stage 2 is from subport x to subport y. The following two facts can be observed from Fig. 11.

Fact 1: All of the unique paths passing through S_0, S_4, and S_8 are of type 10, through S_1, S_5, and S_9 are of type 01, through S_2 and S_6 are of type 00, and through S_3 and S_7 are of type 11. (See Fig. 12(a).)

Fact 2: Each switch at stage 2 has exactly 16 unique paths passing through it. More precisely, let $\mathcal{U}_i$ denote the set of all 16 unique paths passing through S_i. Then

$$\mathcal{U}_0 = \{P(i,F) \mid i = 2, 7, 12, 17 \text{ and } F = 16, 17, 18, 19\},\ \mathcal{U}_1 = \{P(i,F) \mid i = 0, 5, 10, 15 \text{ and } F = 12, 13, 14, 15\},$$
$$\mathcal{U}_2 = \{P(i,F) \mid i = 0, 5, 10, 15 \text{ and } F = 16, 17, 18, 19\},\ \mathcal{U}_3 = \{P(i,F) \mid i = 3, 8, 13, 18 \text{ and } F = 12, 13, 14, 15\},$$
$$\mathcal{U}_4 = \{P(i,F) \mid i = 3, 8, 13, 18 \text{ and } F = 16, 17, 18, 19\},\ \mathcal{U}_5 = \{P(i,F) \mid i = 1, 6, 11, 16 \text{ and } F = 12, 13, 14, 15\},$$
$$\mathcal{U}_6 = \{P(i,F) \mid i = 1, 6, 11, 16 \text{ and } F = 16, 17, 18, 19\},\ \mathcal{U}_7 = \{P(i,F) \mid i = 4, 9, 14, 19 \text{ and } F = 12, 13, 14, 15\},$$
$$\mathcal{U}_8 = \{P(i,F) \mid i = 4, 9, 14, 19 \text{ and } F = 16, 17, 18, 19\},\ \mathcal{U}_9 = \{P(i,F) \mid i = 2, 7, 12, 17 \text{ and } F = 12, 13, 14, 15\}.$$

By Fact 1, in a network configuration, switch S_0 has to be set to cross to let a unique path in $\mathcal{U}_0$ passing through it. Let $\mathcal{N}_0$ denote the set of paths of passing through S_0 which are of type 01; see Fig. 12(b). Also by Fact 1, in a network configuration, switch S_3 has to be set to straight to let a unique path in $\mathcal{U}_3$ passing through it. Let $\mathcal{N}_3$ denote the set of paths passing through S_3 which are of type 00; see Fig. 12(c). Let $I \times J$*-requests* denote the set of all (i, j)-requests with $i \in I$ and $j \in J$. It

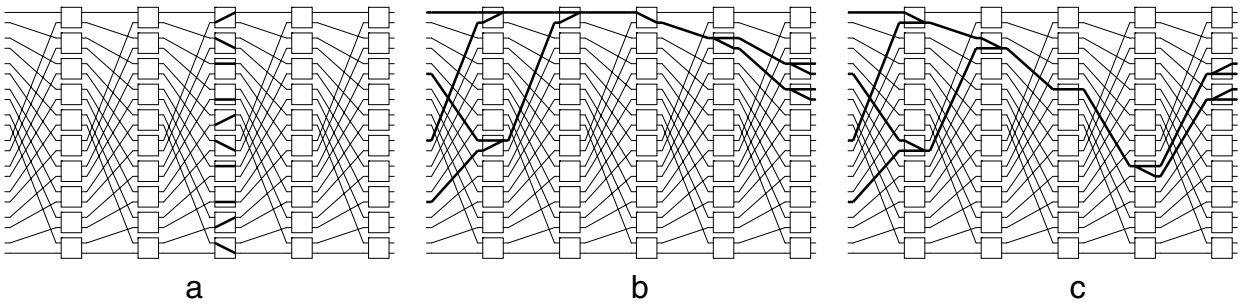


Fig. 12. (a) The setting of switches at stage 2 when unique paths pass through them. (b) The set of paths in $\mathcal{N}_0$. These paths fulfill $\{0, 5, 10, 15\} \times \{4, 5, 6, 7\}$-requests. (c) The set of paths in $\mathcal{N}_3$. These paths also fulfill $\{0, 5, 10, 15\} \times \{4, 5, 6, 7\}$-requests.

can be observed from Fig. 12(b)(c) that $\mathcal{N}_0$ and $\mathcal{N}_3$ both fulfill $\{0, 5, 10, 15\} \times \{4, 5, 6, 7\}$-requests. Thus when the 32 unique paths in $\mathcal{U}_0 \cup \mathcal{U}_3$ are fulfilled, the 32 paths in $\mathcal{N}_0 \cup \mathcal{N}_3$ are also established; however, $\mathcal{N}_0 \cup \mathcal{N}_3$ fulfills at most 16 routing requests and at least 16 routing requests are repeated. The same situation also occurs when the 32 unique paths in $\mathcal{U}_1 \cup \mathcal{U}_8$, in $\mathcal{U}_2 \cup \mathcal{U}_5$, in $\mathcal{U}_4 \cup \mathcal{U}_7$, and in $\mathcal{U}_6 \cup \mathcal{U}_9$ are established. From the above, a total of $16 \cdot 5 = 80$ routing requests are repeated. Hence to fulfill an all-to-all communication in a 20×20 GSEN, at least $400 + 80 = 480$ routing requests have to be fulfilled, i.e., 480 routing paths have to be established.

Now we prove $\mathcal{R}(20) \leq 24$ by showing that an all-to-all communication in a 20×20 GSEN can be fulfilled in 24 network configuration. A 20×20 GSEN has 32 doubly alternating configurations. Consider these 32 doubly alternating configurations. It is not difficult to check that $A' = 0$ and $A' = 17$ obtain the same permutation and hence only one of them is needed in an all-to-all communication. Each of the following pairs of doubly alternating configurations also obtain the same permutation and hence only one in each pair is needed in an all-to-all communication: $A' = 1$ and $A' = 16$, $A' = 2$ and $A' = 19$, $A' = 3$ and $A' = 18$, $A' = 8$ and $A' = 25$, $A' = 9$ and $A' = 24$, $A' = 10$ and $A' = 27$, and $A' = 11$ and $A' = 26$. By removing one doubly alternating configuration from each of the above eight pairs, we have a set $\mathcal{A}'$ containing 24 doubly alternating configurations; in particular, we can choose $\mathcal{A}' = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 20, 21, 22, 23, 28, 29, 30, 31\}$. In Fig. 13, we show all the permutations obtained by applying the network configurations in $\mathcal{A}'$. It is not difficult to see that $\mathcal{A}'$ fulfills an all-to-all communication in a 20×20 GSEN. Hence $\mathcal{R}(20) \leq 24$. □

6. Concluding remarks

The shuffle-exchange network has been proposed as a popular architecture for MINs. The generalized shuffle-exchange networks (GSEN) is a generalization of the shuffle-exchange network. We follow the convention used in [1,2,10] that an $N \times N$ GSEN has exactly $\lceil \log_2 N \rceil$ stages. Based on this convention, we define $n = \lceil \log_2 N \rceil$ and we have $2^{n-1} < N \leq 2^n$.

In this paper we consider the all-to-all personalized exchange problem in GSENs. Since a GSEN does not have the unique path property, previous algorithms [9,13] cannot be used. To our knowledge, no one has studied all-to-all personalized exchange in MINs which do not have the unique path property and do not satisfy $N = 2^n$. An optimal algorithm and several bounds on $\mathcal{R}(N)$ and $\mathcal{R}_{sc}(N)$ have been proposed in this paper; recall that $\mathcal{R}(N)$ is the minimum number of network configurations required to fulfill all-to-all communication in an $N \times N$ GSEN and $\mathcal{R}_{sc}(N)$ is the minimum number of network configurations required to fulfill all-to-all communication in an $N \times N$ GSEN when the stage control technique is assumed. In Theorem 3.8, we have proven $N \leq \mathcal{R}(N) \leq \mathcal{R}_{sc}(N) = 2^n$. In Theorem 4.8, we have proven $N = \mathcal{R}(N) < \mathcal{R}_{sc}(N) = 2^n$ if $N \equiv 2 \pmod 4$. In Theorem 5.4, we have proven $\mathcal{R}(N) = \mathcal{R}_{sc}(N) = 2^n$ if $k \geq 2, N \equiv 0 \pmod{2^k}, N \not\equiv 0 \pmod{2^{k+1}}$, and $2^{n-1} + 2^{n-k} \leq N \leq 2^n$. In Theorem 5.5, we have proven $\mathcal{R}(20) = 24$.

Before closing this paper, we list $\mathcal{R}(N)$ and $\mathcal{R}_{sc}(N)$ for $N = 4, 6, \ldots, 128$ in Fig. 14. We conjecture that when $N \equiv 4 \pmod 8$, the best way to reduce the number of network configurations used in an all-to-all communication in a GSEN is to use doubly alternating stage control. One can examine $\mathcal{R}(36) \leq 40$ and $\mathcal{R}(44) \leq 48$ by the aid of a computer. We also conjecture that when $N \equiv 8 \pmod{16}$, the best way to reduce the number of network configurations used in an all-to-all communication in a GSEN is to use *quadruply alternating stage control*, meaning that the states of the switches of a stage *alternate between four straight states and four cross states*. The network configuration obtained by quadruply alternating stage control is called a *quadruply alternating configuration* and it can be represented by the number

$$A'' = a''_{n-1}2^{n-1} + a''_{n-2}2^{n-2} + \cdots + a''_0 2^0$$

as follows. Let a''_ℓ denote the states of the switches at stage $n-1-\ell$ such that

- $a''_\ell = 0$ means the states are 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, and so on.
- $a''_\ell = 1$ means the states are 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, and so on.

Obviously, $0 \leq A'' < 2^n$.

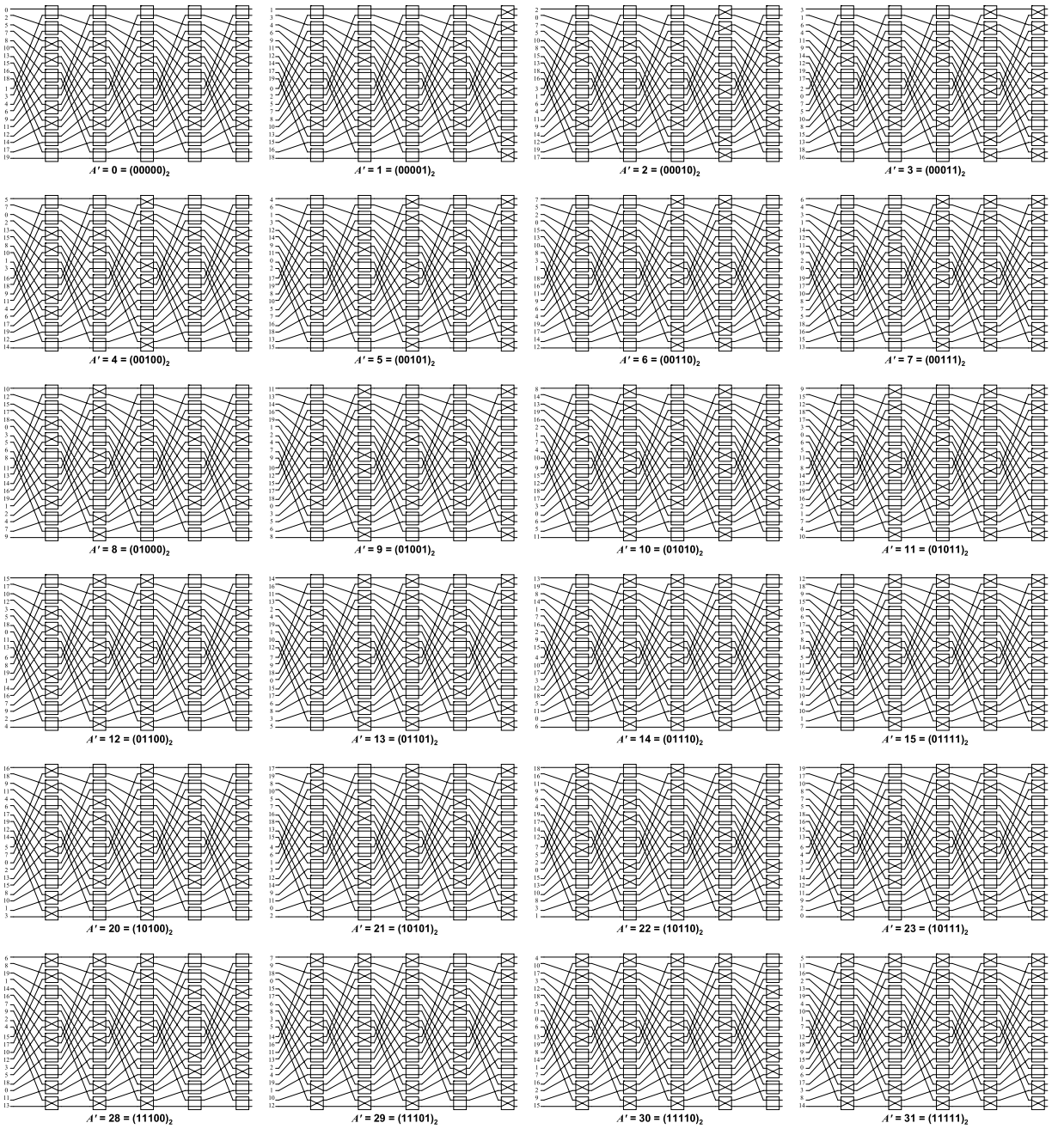


Fig. 13. Fulfill an all-to-all communication in a 20×20 GSEN by using the 24 network configurations in $\mathcal{A}' = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 20, 21, 22, 23, 28, 29, 30, 31\}$. The destinations of the messages for each network configuration are shown on the left-hand side of the GSEN.

Let $\mathcal{A}'$ denote a set of doubly alternating configurations and let $\mathcal{A}''$ denote a set of quadruply alternating configurations. The following results are obtained by the aid of a computer.

- $\mathcal{R}(36) \leq 40$; by using $\mathcal{A}' = \{0 \sim 3,\ 8 \sim 19,\ 24 \sim 35,\ 40 \sim 43,\ 48 \sim 51, 56 \sim 59\}$.
- $\mathcal{R}(44) \leq 48$; by using $\mathcal{A}' = \{0 \sim 3,\ 8 \sim 19,\ 24 \sim 35,\ 40 \sim 51,\ 56 \sim 63\}$.
- $\mathcal{R}(68) \leq 72$; by using $\mathcal{A}' = \{0 \sim 11,\ 16 \sim 43,\ 48 \sim 63,\ 68 \sim 71,\ 80 \sim 83,\ 100 \sim 104,\ 112 \sim 115\}$.
- $\mathcal{R}(72) \leq 96$; by using $\mathcal{A}'' = \{0 \sim 63,\ 72 \sim 79,\ 88 \sim 95,\ 104 \sim 111,\ 120 \sim 127\}$.
- $\mathcal{R}(76) \leq 88$; by using $\mathcal{A}' = \{0 \sim 7,\ 12 \sim 39,\ 44 \sim 67,\ 80 \sim 91,\ 96 \sim 99,\ 112 \sim 123\}$.
- $\mathcal{R}(84) \leq 96$; by using $\mathcal{A}' = \{0 \sim 11,\ 16 \sim 43,\ 48 \sim 63,\ 68 \sim 71,\ 80 \sim 95,\ 100 \sim 103,\ 112 \sim 127\}$.
- $\mathcal{R}(92) \leq 112$; by using $\mathcal{A}' = \{0 \sim 7,\ 12 \sim 39,\ 44 \sim 71,\ 76 \sim 103,\ 108 \sim 127\}$.

N	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34	36	38	40	42	44
$R(N)$	4	6	8	10	16	14	16	18	24	22	32	26	32	30	32	34	≤ 40	38	64	42	≤ 48
$Rsc(N)$	4	8	8	16	16	16	16	32	32	32	32	32	32	32	32	64	64	64	64	64	64

N	46	48	50	52	54	56	58	60	62	64	66	68	70	72	74	76	78	80	82	84	86
$R(N)$	46	64	50	64	54	64	58	64	62	64	66	≤ 72	70	≤ 96	74	≤ 88	78	128	82	≤ 96	86
$Rsc(N)$	64	64	64	64	64	64	64	64	64	64	128	128	128	128	128	128	128	128	128	128	128

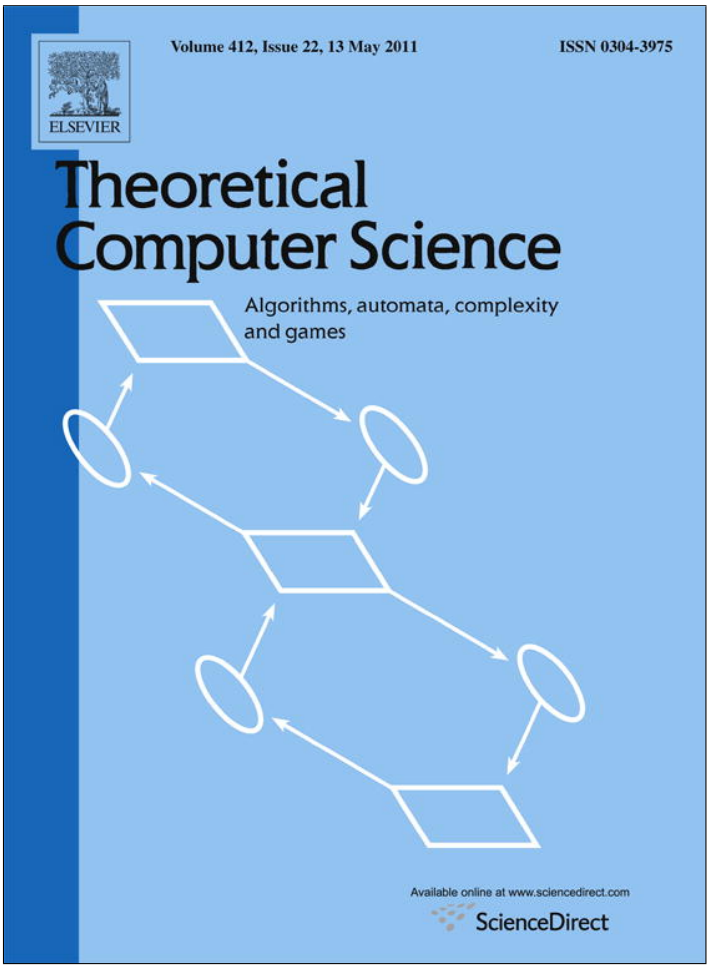
N	88	90	92	94	96	98	100	102	104	106	108	110	112	114	116	118	120	122	124	126	128
$R(N)$	128	90	≤ 112	94	128	98	128	102	128	106	128	110	128	114	128	118	128	122	128	126	128
$Rsc(N)$	128	128	128	128	128	128	128	128	128	128	128	128	128	128	128	128	128	128	128	128	128

Fig. 14. Known results of $\mathcal{R}(N)$ and $\mathcal{R}_{sc}(N)$ for $N = 4, 6, \ldots, 128$.

Although we know that $\mathcal{R}(36) \leq 40$, we are unable to prove that $\mathcal{R}(36) \geq 40$. Several open problems can be found in Fig. 14. In particular, we conjecture $\mathcal{R}(36) = 40$, $\mathcal{R}(44) = 48$. Determining $\mathcal{R}(N)$ for all N such that $N \equiv 0 \pmod 4$ is still an open problem.

References

[1] Z. Chen, Z. Liu, Z. Qiu, Bidirectional shuffle-exchange network and tag-based routing algorithm, IEEE Commun. Lett. 7 (3) (2003) 121–123.
[2] C. Chen, J.K. Lou, An efficient tag-based routing algorithm for the backward network of a bidirectional general shuffle-exchange network, IEEE Commun. Lett. 10 (4) (2006) 296–298.
[3] M. Gerla, E. Leonardi, F. Neri, P. Palnati, Routing in the bidirectional shufflenet, IEEE/ACM Trans. Netw. 9 (1) (2001) 91–103.
[4] C.P. Kuruskal, A unified theory of interconnection network structure, Theoret. Comput. Sci. 48 (1986) 75–94.
[5] J.K. Lan, W.Y. Chou, C. Chen, Efficient routing algorithms for the bidirectional general shuffle-exchange network, Discrete Math. Algorithms Appl. 1 (2) (2009) 267–281.
[6] D.H. Lawrie, Access and alignment of data in an array processor, IEEE Trans. Comput. C- 24 (12) (1975) 1145–1155.
[7] S.C. Liew, On the stability if shuffle-exchange and bidirectional shuffle-exchange deflection networkA, IEEE/ACM Trans. Netw. 5 (1) (1997) 87–94.
[8] V.W. Liu, C. Chen, R.B. Chen, Optimal all-to-all personalized exchange in d-nary banyan multistage interconnection networks, J. Comb. Optim. 14 (2007) 131–142.
[9] A. Massini, All-to-all personalized communication on multistage interconnection networks, Discrete Appl. Math. 128 (2) (2003) 435–446.
[10] K. Padmanabhan, Design and analysis of even-sized binary shuffle-exchange networks for multiprocessors, IEEE Trans. Parallel Distrib. Syst. 2 (4) (1991) 385–397.
[11] C. Qiao, L. Zhou, Scheduling switch disjoint connections in stage-controlled photonic banyans, IEEE Trans. Commun. 47 (1) (1999) 139–148.
[12] R. Ramaswami, Multi-wavelength lightwave networks for computer communication, IEEE Commun. Mag. 31 (2) (1993) 78–88.
[13] Y. Yang, J. Wang, Optimal all-to-all personalized exchange in self-routable multistage networks, IEEE Trans. Parallel Distrib. Syst. 11 (3) (2000) 261–274.
[14] Y. Yang, J. Wang, Optimal all-to-all personalized exchange in a class of optical multistage networks, IEEE Trans. Parallel Distrib. Syst. 12 (9) (2001) 567–582.



Theoretical Computer Science 412 (2011) 2237–2252

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Constructing independent spanning trees for locally twisted cubes☆

Yi-Jiun Liu, James K. Lan, Well Y. Chou, Chiuyuan Chen *

Department of Applied Mathematics, National Chiao Tung University, Hsinchu 300, Taiwan

ARTICLE INFO

Article history:
Received 11 December 2009
Accepted 27 December 2010
Communicated by D.-Z. Du

Keywords:
Independent spanning trees
Data broadcasting
Design and analysis of algorithms
Locally twisted cubes
Hypercubes
Hypercube variants
Parallel algorithm

ABSTRACT

The independent spanning trees (ISTs) problem attempts to construct a set of pairwise independent spanning trees and it has numerous applications in networks such as data broadcasting, scattering and reliable communication protocols. The well-known ISTs conjecture, Vertex/Edge Conjecture, states that any n-connected/n-edge-connected graph has n vertex-ISTs/edge-ISTs rooted at an arbitrary vertex r. It has been shown that the Vertex Conjecture implies the Edge Conjecture. In this paper, we consider the independent spanning trees problem on the n-dimensional locally twisted cube LTQ_n. The very recent algorithm proposed by Hsieh and Tu (2009) [12] is designed to construct n edge-ISTs rooted at vertex 0 for LTQ_n. However, we find out that LTQ_n is not vertex-transitive when $n \geq 4$; therefore Hsieh and Tu's result does not solve the Edge Conjecture for LTQ_n. In this paper, we propose an algorithm for constructing n vertex-ISTs for LTQ_n; consequently, we confirm the Vertex Conjecture (and hence also the Edge Conjecture) for LTQ_n.

1. Introduction

Two spanning trees in a graph G are said to be vertex/edge independent if they are rooted at the same vertex r and for each vertex v of G, $v \neq r$, the paths from r to v in two trees are vertex/edge disjoint except the two end vertices. A set of spanning trees of G are said to be vertex/edge independent if they are pairwise vertex/edge independent. The vertex/edge independent spanning trees (ISTs) problem attempts to construct a set of pairwise vertex/edge independent spanning trees and it has has applications such as data broadcasting, scattering and reliable communication protocols. For example, a rooted spanning tree in the underlying graph of a network can be viewed as a broadcasting scheme for data communication and fault-tolerance can be achieved by sending n copies of the message along the n independent spanning trees rooted at the source node [1]. For other applications, see [3] for the multi-node broadcasting problem, [21] for one-to-all broadcasting, and [2] for n-channel graphs, reliable broadcasting and secure message distribution.

The independent spanning trees problem has been widely studied in the last two decades. Two well-known conjectures on this problem are raised by Zehavi and Itai [27]: (refer to [4] or [23] for graph terminologies)

Conjecture 1.1 (*Vertex Conjecture*). *Any n-connected graph has n vertex-ISTs rooted at an arbitrary vertex r.*

Conjecture 1.2 (*Edge Conjecture*). *Any n-edge-connected graph has n edge-ISTs rooted at an arbitrary vertex r.*

Zehavi and Itai [27] also raised the question: It would be interesting to show that either the Vertex Conjecture implies the Edge Conjecture, or vice versa. Later, Khuller and Schieber [16] successfully proved that the Vertex Conjecture implies the Edge Conjecture, i.e., if any n-connected graph has n vertex-ISTs, then any n-edge-connected graph has n edge-ISTs. Khuller

☆ This research was partially supported by the National Science Council of the Republic of China under the grants grant NSC97-2628-M-009-006-MY3.
* Corresponding author. Tel.: +886 3 5731767.
E-mail addresses: cychen@mail.nctu.edu.tw, cychen@cc.nctu.edu.tw (C. Chen).

0304-3975/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2010.12.061

Table 1
The connectivity, edge-connectivity and diameters of Q_n and its variants.

Topology	$\kappa(G)$	$\lambda(G)$	Diameter	
Q_n	n	n	n	
LTQ_n	n	n	$\lceil (n+1)/2 \rceil$ $\lceil (n+3)/2 \rceil$	if $n < 5$ if $n \geq 5$
TQ_n	n	n	$\lceil (n+1)/2 \rceil$	
MQ_n	n	n	$\lceil (n+2)/2 \rceil$ $\lceil (n+1)/2 \rceil$	in 0-MQ_n for $n \geq 4$ in 1-MQ_n for $n \geq 1$

and Schieber's proof also works for the directed graphs. For the directed case, Edmonds [7] solved the Edge Conjecture. Khuller and Schieber [16] pointed out that the Vertex Conjecture for directed graphs is the strongest conjecture since it implies all the other conjectures.

The vertex and the edge conjectures have been confirmed only for $n \leq 4$. In particular, in [15], Itai and Rodeh proposed a linear-time algorithm for constructing two edge-ISTs for a 2-edge-connected graph; they also solved the Vertex Conjecture for $n = 2$. In [27], Zehavi and Itai solved the Vertex Conjecture for $n = 3$, but they did not proposed an algorithm for constructing three vertex-ISTs. In [6], Cheriyan and Maheshwari proposed an $O(|V(G)|^2)$-time algorithm for constructing three vertex-ISTs in a 3-connected graph. In [5], Curran et al. proposed an $O(|V(G)|^3)$-time algorithm for constructing four vertex-ISTs in a 4-connected graph. When $n \geq 5$, both the vertex and the edge conjectures are still open. It has been proven that the Vertex/Edge Conjecture holds for several restricted classes of graphs or digraphs, such as planar graphs [9,10,17,18], maximal planar graphs [19], product graphs [20], chordal rings [14,24], de Bruijn and Kautz digraphs [8,11], and hypercubes [22,26]. Note that the development of algorithms for constructing vertex-ISTs tends toward pursuing two research goals: One is to design efficient construction schemes (for example, [14,17,19,24] proposed linear-time algorithms) and the other is to reduce the heights of vertex-ISTs (for example, [11,22,24] proposed the idea of height improvements).

The hypercube (Q_n) is one of the most popular interconnection network topologies due to its simple structure and ease of implementation. Several commercial machines with hypercube topology have been built and a huge amount of research work, both theoretical and practical, has been done on various aspects of the hypercube. However, it has been shown that the hypercube does not achieve the smallest possible diameter for its resources. Therefore, many variants of the hypercube have been proposed. The most well-known variants are locally twisted cubes (LTQ_n), twisted cubes (TQ_n), crossed cubes (CQ_n) and Möbius cubes (MQ_n). A concise comparison including the connectivity, edge-connectivity and diameters of Q_n and its variants is shown in Table 1. Clearly, one advantage of LTQ_n over Q_n is that the diameter of LTQ_n is only about half of that of Q_n.

Before going further, we now briefly review results of the vertex-ISTs problem for Q_n. It is well known that Q_n is n-connected. Since Q_n is a product graph, the algorithm proposed by Obokata et al. [20] can be used to construct n vertex-ISTs for Q_n. As to the construction of the height-reduced vertex-ISTs on Q_n, Tang et al. [22] modified the algorithm in [20] and proposed an $O(n2^n)$-time algorithm for constructing an optimal set (in the sense of smallest average path lengths) of n vertex-ISTs for Q_n. It was pointed out by Yang et al. [26] that the algorithms in [20,22] are designed by a recursive fashion and such a construction forbids the possibility that the algorithm could be parallelized; Yang et al. [26] therefore proposed a parallel construction for an optimal set of n vertex-ISTs for Q_n.

The purpose of this paper is to confirm the Vertex Conjecture for the n-dimensional locally twisted cube LTQ_n. The very recent algorithm proposed by Hsieh and Tu [12] is designed to construct n edge-ISTs rooted at vertex 0 for LTQ_n. However, we find out that LTQ_n is not vertex-transitive whenever $n \geq 4$ (see Section 2). Therefore, Hsieh and Tu did not solve the Edge Conjecture for LTQ_n. In this paper, we will propose an algorithm for constructing n vertex-ISTs rooted at an arbitrary vertex of LTQ_n. Therefore, we will confirm the Vertex Conjecture for LTQ_n. Since vertex-ISTs are edge-ISTs, we also confirm the Edge Conjecture for LTQ_n.

In the remaining discussion, we will simply use ISTs to denote vertex-ISTs unless otherwise specified. This paper is organized as follows. In Section 2, we give definitions and notations used in the paper. In Section 3, we present an algorithm to construct n ISTs rooted at an arbitrary vertex of LTQ_n. In Section 4, we prove the correctness of our algorithm. Concluding remarks are given in the last section.

2. Preliminaries

All graphs in this paper are simple undirected graphs. Let G be a graph with vertex set $V(G)$ and the edge set $E(G)$. Let $x, y \in V(G)$. A path from x to y is denoted as x, y-path. The *distance* between two vertices x and y, denoted by $d(x, y)$, is the length of a shortest x, y-path. Two x, y-paths P and Q are *edge-disjoint* if $E(P) \cap E(Q) = \emptyset$. Two x, y-paths P and Q are *internally vertex-disjoint* if $V(P) \cap V(Q) = \{x, y\}$. A subgraph T of G is a *spanning tree* if T is a tree and $V(T) = V(G)$. Two spanning trees T and T' of G are *vertex-independent/edge-independent* if T and T' are rooted at the same vertex, say r, and

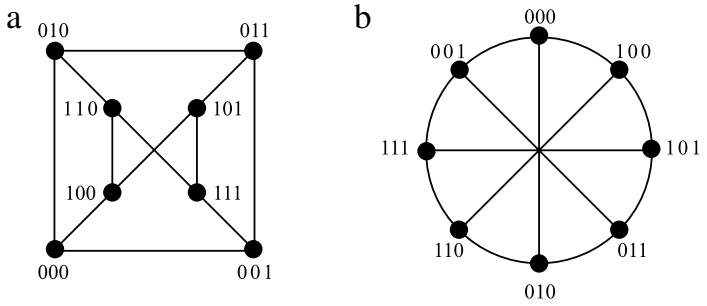


Fig. 1. (a) LTQ_3. (b) A symmetric drawing of LTQ_3.

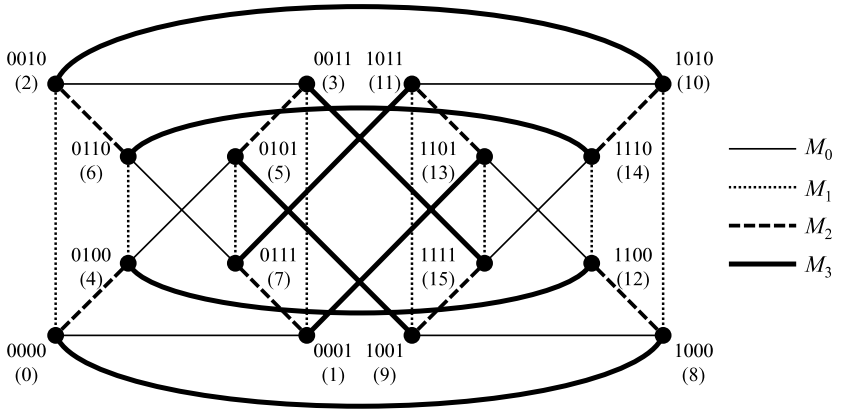


Fig. 2. LTQ_4 and its perfect matchings $\{M_0, M_1, M_2, M_3\}$.

for each $v \in V(G)$, $v \neq r$, the r, v-path in T and the r, v-path in T' are (internally) vertex-disjoint/edge-disjoint. A set of spanning trees of G are *vertex-independent/edge-independent* if they are pairwise vertex-independent/edge-independent.

2.1. The locally twisted cube

The *n-dimensional locally twisted cube* LTQ_n ($n \geq 2$), proposed first by Yang et al. [25], has 2^n vertices. Each vertex is an n-string on $\{0, 1\}$, i.e., a binary string of length n. The LTQ_n is defined recursively as follows.

Definition 1 ([25]). 1. LTQ_2 is the graph consisting of four vertices labeled with 00, 01, 10, and 11, respectively, and connected by the four edges (00, 01) (00, 10), (01, 11), and (10, 11).

2. LTQ_n ($n \geq 3$) is built from two disjoint copies of LTQ_{n-1}'s as follows: Let $0LTQ_{n-1}$ (respectively, $1LTQ_{n-1}$) denote the graph obtained by prefixing the label of each vertex in one copy of LTQ_{n-1} with 0 (respectively, 1). Connect each vertex $0x_{n-2}x_{n-3}\ldots x_0$ of $0LTQ_{n-1}$ to the vertex $1(x_{n-2} \oplus x_0)x_{n-3}\ldots x_0$ of $1LTQ_{n-1}$ with an edge, where "$\oplus$" represents the XOR operation, or equivalently, the modulo 2 addition.

Figs. 1 and 2 illustrate LTQ_3 and LTQ_4, respectively. Yang et al. [25] also mentioned that the locally twisted cube can be equivalently defined by the following non-recursive fashion.

Definition 2 ([25]). Let $x = x_{n-1}x_{n-2}\ldots x_0$ and $y = y_{n-1}y_{n-2}\ldots y_0$ be two vertices of LTQ_n ($n \geq 2$). Then vertices x and y are adjacent if and only if one of the following conditions are satisfied.

1. There is an integer $2 \leq k \leq n-1$ such that
   (a) $x_k = \bar{y}_k$ ($\bar{y}_k$ is the complement of y_k in $\{0, 1\}$)
   (b) $x_{k-1} = y_{k-1} \oplus x_0$
   (c) all the remaining bits of x and y are identical.
2. There is an integer $0 \leq k \leq 1$ such that x and y only differ in the kth bit.

From Definition 2, LTQ_n is obviously an n-regular graph, and the labels of any two adjacent vertices of LTQ_n differ in at most two consecutive bits. Note that in the remaining part of this paper, the label of a vertex in LTQ_n is presented in binary representation and decimal representation interchangeably when there is no ambiguity.

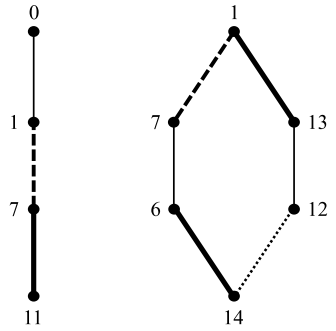


Fig. 3. The in-vertex-transitivity of LTQ_4.

2.2. The neighbor information and the perfect matchings of the locally twisted cube

From Definition 2, the n neighbors of an arbitrary vertex $x = x_{n-1}x_{n-2}\ldots x_0$ of LTQ_n is given by

$$\begin{aligned} f_0(x) &= x_{n-1}x_{n-2}x_{n-3}\ldots x_2x_1\bar{x}_0,\\ f_1(x) &= x_{n-1}x_{n-2}x_{n-3}\ldots x_2\bar{x}_1x_0,\\ f_2(x) &= x_{n-1}x_{n-2}x_{n-3}\ldots \bar{x}_2\ (x_1\oplus x_0)\ x_0,\\ \vdots &= \qquad\qquad\qquad \vdots\\ f_{n-2}(x) &= x_{n-1}\bar{x}_{n-2}\ (x_{n-3}\oplus x_0)x_{n-4}\ \ldots x_1x_0,\\ f_{n-1}(x) &= \bar{x}_{n-1}\ (x_{n-2}\oplus x_0)\ x_{n-3}\ldots x_2x_1x_0, \end{aligned} \tag{1}$$

where $f_k(x)$, $0 \le k \le n-1$, is called the *kth dimensional neighbor* of x; see also Lemma 4 in [13]. By (1), the n neighbors of vertices 0 and 1 can be determined as follows.

Lemma 2.1. *The n neighbors of vertex* 0 *in* LTQ_n *is given by*

$$f_k(0) = 2^k,$$

for $k = 0, 1, \ldots, n-1$. *The n neighbors of vertex* 1 *in* LTQ_n *is given by*

$$f_k(1) = \begin{cases} 0 & \text{if } k = 0,\\ 3 & \text{if } k = 1,\\ 2^k + 2^{k-1} + 1 & \text{if } 2 \le k \le n-1. \end{cases}$$

Given a graph $G = (V, E)$, a *matching* M of G is a set of pairwise non-adjacent edges of G. A *perfect matching* is a matching that saturates all the vertices; in other words, every vertex in the graph is incident to exactly one edge in the matching. From Eq. (1), for all vertices x of LTQ_n and for all $0 \le k \le n-1$, we have

$$f_k(f_k(x)) = x. \tag{2}$$

Therefore, for a fixed k, the set of edges connecting a vertex and its k-th dimensional neighbor forms a perfect matching of LTQ_n. More precisely,

$$M_k = \{(x, f_k(x) \mid x \in V(LTQ_n)\}$$

is a perfect matching of LTQ_n. See Fig. 2 for an illustration.

2.3. The even–odd-vertex-transitivity of the locally twisted cube

A graph is vertex-transitive if for every pair of vertices u and v, there is an automorphism that maps u to v. Intuitively, a vertex-transitive network looks the same from every node. The vertex-transitive property is advantageous to the design and simulation of some algorithms. It is not difficult to see that LTQ_2 and LTQ_3 are vertex-transitive; see Fig. 1. However, in the following, we will show that LTQ_n is not vertex-transitive when $n \ge 4$.

Theorem 2.2. *The locally twisted cube* LTQ_n *is not vertex-transitive for* $n \ge 4$.

Proof. For $n = 4$, let $N_k(r)$ denote the set $N_k(r) = \{x \in V(LTQ_n) \mid d(x, r) = k\}$. Consider the set $\Omega(r) = \{x \in N_2(r) \mid N_1(x) \cap N_1(r) = 1 \text{ and } N_1(x) \cap N_3(r) = 1\}$. Then $\Omega(0) = \{7\}$, but $\Omega(1) = \{6, 12\}$; see Fig. 3 for an illustration. Therefore LTQ_4 is not vertex-transitive.

Now consider LTQ_n with $n \ge 5$. It is well-known that vertices 0 and $2^n - 2$ are at the farthest distance of LTQ_n and $d(2^n - 2, 0) = \lceil \frac{n+3}{2} \rceil$. In the following, we prove that LTQ_n is not vertex-transitive by showing the following claim.

Claim 2.3. *For an arbitrary vertex $x \in V(LTQ_n)$, $n \geq 5$, the distance $d(x, 1) \leq \lceil \frac{n+1}{2} \rceil$.*

Proof of Claim 2.3. Before showing the claim, some notations are introduced first. Let $x = x_{n-1}x_{n-2}\ldots x_0$. Scanning the bits of x from x_{n-1} to x_1 (notice that we ignore the bit x_0). Suppose there are a total of m bits equal to 1 and a total of k disjoint pairs of consecutive bits equal to "11", we denoted it by "11"-bits. A bit x_i, $1 \leq i \leq n-1$, is said to be *isolated* if after removing the k disjoint pairs of "11"-bits of x, we have $x_i = 1$. For example, consider $x = 111011$ in LTQ_6. Then $m = 4$, $k = 1$ and x_1, x_3 are isolated. Clearly, $0 \leq k \leq \lfloor \frac{m}{2} \rfloor$ holds.

It should be noticed that if $m < \lceil \frac{n-1}{2} \rceil$, then there exists a trivial path from x to 1: (i) If $x_0 = 0$, then corrects all $x_i = 1$ bits, $1 \leq i \leq n-1$, to 0, and then corrects x_0 to 1; (ii) If $x_0 = 1$, then corrects x_0 to 0. Then corrects all $x_i = 1$ bits, $1 \leq i \leq n-1$, to 0, and then correct x_0 to 1. Clearly, both paths have length at most $m + 2 \leq \lceil \frac{n+1}{2} \rceil$. In the following, we assume $m \geq \lceil \frac{n-1}{2} \rceil$. Therefore,

$$m - \left\lceil \frac{n-1}{2} \right\rceil \leq k \leq \left\lfloor \frac{m}{2} \right\rfloor$$

holds. There are two cases.

Case 1: $x_0 = 0$. A path from x to 1 can be found as follows: Step 1: Remove all the isolated bits of x. Step 2: Correct x_0 to 1. Step 3: Match all "11"-bits. Clearly, Steps 1, 2 and 3 take $m - 2k$, 1 and k steps, respectively. The total number of steps is

$$m - k + 1 \leq m - \left(m - \left\lceil \frac{n-1}{2} \right\rceil \right) + 1 = \left\lceil \frac{n+1}{2} \right\rceil.$$

For example, consider $x = 11101010$ in LTQ_8. We have $m = 5$, $k = 1$ and x_1, x_3, x_5 are isolated bits. A path from x to 1 is built as follows: $11101010 \xrightarrow{\text{Step 1}} 11001010 \xrightarrow{\text{Step 1}} 11000010 \xrightarrow{\text{Step 1}} 11000000 \xrightarrow{\text{Step 2}} 11000001 \xrightarrow{\text{Step 3}} 00000001$.

Case 2: $x_0 = 1$. We further divide this case into two subcases:

Subcase 2.1: $m + 1 - \lceil \frac{n-1}{2} \rceil \leq k \leq \lfloor \frac{m}{2} \rfloor$. Then a path from x to 1 can be found as follows: Step 1: Correct x_0 to 0. Step 2: Remove all the isolated bits of x. Step 3: Correct x_0 to 1. Step 4: Match all "11"-bits. Clearly, Steps 1, 2, 3 and 4 take 1, $m - 2k$, 1 and k steps, respectively. Thus the total number of steps is

$$m - k + 2 \leq \left\lceil \frac{n+1}{2} \right\rceil.$$

For example, consider $x = 11011011$ in LTQ_8. We have $m = 5$, $k = 2$ and x_1 is a isolated bit. A path from x to 1 is built as follows: $11011011 \xrightarrow{\text{Step 1}} 11011010 \xrightarrow{\text{Step 2}} 11011000 \xrightarrow{\text{Step 3}} 11011001 \xrightarrow{\text{Step 4}} 00011001 \xrightarrow{\text{Step 4}} 00000001$.

Subcase 2.2: $k = m - \lceil \frac{n-1}{2} \rceil$. In this case, all bits $x_{n-1}, x_{n-3}, \ldots, x_1$ must equal to 1 if n is even; either all bits $x_{n-2}, x_{n-3}, \ldots, x_1$ or all bits $x_{n-1}, x_{n-3}, \ldots, x_2$ must equal to 1 if n is odd. Thus a path from x to 1 can be found by bitwise correcting the bits to 0 (by scanning the bits from x_{n-1} to x_1). Since it takes one step to correct an isolated bit and one step to correct a "11"-bits, the total step is

$$(m - 2k) + k = \left\lceil \frac{n-1}{2} \right\rceil.$$

For example, consider $x = 10111011$ in LTQ_8. We have $m = 5$, $k = 1$. A path from x to 1 is built as follows: $10111011 \xrightarrow{\text{isolated}} 01111011 \xrightarrow{\text{"11"-bits}} 00011011 \xrightarrow{\text{"11"-bits}} 00000011 \xrightarrow{\text{isolated}} 00000001$. □

From the above discussion, we have $d(x, 1) \leq \lceil \frac{n+1}{2} \rceil$. As a result, LTQ_n is not vertex-transitive for $n \geq 4$. □

Although LTQ_n fails to be vertex-transitive for $n \geq 4$, it does satisfy the *even–odd-vertex-transitive* property: for every pair of vertices $x = x_{n-1}x_{n-2}\ldots x_0$, $y = y_{n-1}y_{n-2}\ldots y_0$ with the same parity, i.e., $x_0 = y_0$, there is an automorphism ψ that maps x to y. In other words, in LTQ_n, all even-numbered vertices are symmetric and all odd-numbered vertices are symmetric. By using this property, we may pay our attention of constructing ISTs to use vertex 0 and vertex 1 as the common root without loss of generality.

Theorem 2.4. *The locally twisted cube LTQ_n satisfies the even–odd-vertex-transitive property.*

Proof. It suffices to prove that there exists an automorphism which maps v ($\neq 0$) to 0 (resp., v ($\neq 1$) to 1), whenever v is an even-numbered (resp., odd-numbered) vertex. For two n-bits binary strings x and y, let $x \oplus y$ denote the bitwise XOR (modulo 2) of x and y. Let $v = v_{n-1}v_{n-2}\ldots v_0 \in V(LTQ_n)$.

Suppose v is an even-numbered vertex. For $x = x_{n-1}x_{n-2}\ldots x_0 \in V(LTQ_n)$, define a function ψ_0 as follows:

$$\psi_0(x) = v \oplus x.$$

It is not difficult to see that ψ_0 is a bijection from $V(LTQ_n)$ to $V(LTQ_n)$. Now we verify that ψ_0 preserves the adjacency. Consider any edge $(x, f_k(x)) \in E(LTQ_n)$. Since $v_0 = 0$, we have

$$\psi_0(x) = (v_{n-1} \oplus x_{n-1})\ (v_{n-2} \oplus x_{n-2})\ \ldots (v_{k+1} \oplus x_{k+1})\ (v_k \oplus x_k)\ (v_{k-1} \oplus x_{k-1})\ \ldots\ (v_1 \oplus x_1)\ x_0.$$

Algorithm 1 CONSTRUCT_IST

Input: All vertices of LTQ_n and root r.
Output: n ISTs $T_0, T_1, \ldots, T_{n-1}$ rooted at r.

```
1:  for i = 0 to n − 1 do in parallel        ▷ construct T_i simultaneously
2:      child_of_the_root ← f_i(r)
3:      V(T_i) ← {child_of_the_root}
4:      for t = 1 to n do                    ▷ outer for-loop
5:          S ← ∅;
6:          for each vertex v ∈ V(T_i) do    ▷ inner for-loop
7:              u ← f_{(i+t) mod n}(v)
8:              E(T_i) ← E(T_i) ∪ {(v, u)}   ▷ set the parent of vertex u as v in T_i
9:              S ← S ∪ {u}
10:         end for
11:         V(T_i) ← V(T_i) ∪ S
12:     end for
13: end for
```

Also,

$$\psi_0(f_k(x)) = \begin{cases} (v_{n-1}\oplus x_{n-1})\ (v_{n-2}\oplus x_{n-2})\ \ldots\ (v_1\oplus u_1)\ \bar{x}_0 & \text{if } k = 0, \\ (v_{n-1}\oplus x_{n-1})\ (v_{n-2}\oplus x_{n-2})\ \ldots\ (v_2\oplus u_2)\ (v_1\oplus \bar{x}_1)\ x_0 & \text{if } k = 1, \end{cases}$$

and for $2 \le k \le n-1$,

$$\psi_0(f_k(x)) = (v_{n-1}\oplus x_{n-1})\ (v_{n-2}\oplus x_{n-2})\ \ldots\ (v_{k+1}\oplus x_{k+1})\ (v_k\oplus \bar{x}_k)\ (v_{k-1}\oplus x_{k-1}\oplus x_0)\ (v_{k-2}\oplus x_{k-2})\ \ldots\ (v_1\oplus x_1)\ x_0.$$

Since $v_k \oplus \bar{x}_k = \overline{v_k \oplus x_k}$ no matter $v_k = x_k$ or $v_k \neq x_k$, we have

$$\psi_0(f_k(x)) = f_k(\psi_0(x))$$

and hence $(\psi_0(x), \psi_0(f_k(x))) \in E(LTQ_n)$.

Similar arguments can be applied to the case of v being an odd-numbered vertex, except that the bijection function from $V(LTQ_n)$ to $V(LTQ_n)$ is replaced by

$$\psi_1(x) = v \oplus x \oplus 1. \quad \square$$

3. The algorithm

We now present an algorithm, called CONSTRUCT_IST, for constructing n ISTs $T_0, T_1, \ldots, T_{n-1}$ rooted at an arbitrary vertex r for the locally twisted cube LTQ_n in Algorithm 1. For convenience, call the for-loop in lines 4–12 of this algorithm the "outer for-loop" and call the for-loop in lines 6–10 the "inner for-loop". This algorithm constructs $T_0, T_1, \ldots, T_{n-1}$ simultaneously and it works as follows. Since LTQ_n is n-regular, the n neighbors of the root r must be the unique child of the root r in $T_0, T_1, \ldots, T_{n-1}$, respectively. In this algorithm, the unique child of the root r in T_i is set as $f_i(r)$. Thus, initially $V(T_i) = \{f_i(r)\}$. At the tth iteration of the outer for-loop, each vertex v in $V(T_i)$ is connected to a new vertex $u = f_{(i+t) \bmod n}(v)$ by using the edges in perfect matching $M_{(i+t) \bmod n}$, and the edge (v, u) is added to T_i (i.e., the parent of u is set as v in T_i). After n iterations of the outer for-loop, T_i is constructed.

Example 1. We now demonstrate how Algorithm CONSTRUCT_IST constructs T_2 rooted at vertex 1 in LTQ_4. In line 2 of the algorithm, the unique child of the root 1 is set as $f_2(1) = 7$. Thus $V(T_2) = \{7\}$. Now consider the outer for-loop. For $t = 1$, each vertex in $V(T_2)$ is connected to a new vertex by using the edges in M_3; thus the edge (7, 11) is added to T_2; so S becomes {11} and $V(T_2)$ becomes {7, 11}. For $t = 2$, each vertex in $V(T_2)$ is connected to a new vertex by using the edges in M_0; thus the edges (7, 6) and (11, 10) are added to T_2; so S becomes {6, 10} and $V(T_2)$ becomes {7, 11, 6, 10}. For $t = 3$, each vertex in $V(T_2)$ is connected to a new vertex by using the edges in M_1; thus the edges (7, 5), (11, 9), (6, 4) and (10, 8) are added to T_2; so S becomes {5, 9, 4, 8} and $V(T_2)$ becomes {7, 11, 6, 10, 5, 9, 4, 8}. Finally, for $t = 4$, each vertex in $V(T_2)$ is connected to a new vertex by using the edges in M_2; thus the edges (7, 1), (11, 13), (6, 2), (10, 14), (5, 3), (9, 15), (4, 0) and (8, 12) are added to T_2; so S becomes {1, 13, 2, 14, 3, 15, 0, 12} and $V(T_2)$ becomes {7, 11, 6, 10, 5, 9, 4, 8, 1, 13, 2, 14, 3, 15, 0, 12}. See Fig. 4 for an illustration.

4. Correctness

The purpose of this section is to prove that $T_0, T_1, \ldots, T_{n-1}$ generated by Algorithm Construct_IST are n ISTs rooted at an arbitrary vertex r for LTQ_n. To this end, some notations are first introduced in Section 4.1. We show that $T_0, T_1, \ldots, T_{n-1}$ are n spanning trees of LTQ_n in Section 4.2. The vertex-independency of $T_0, T_1, \ldots, T_{n-1}$ is shown in Section 4.3.

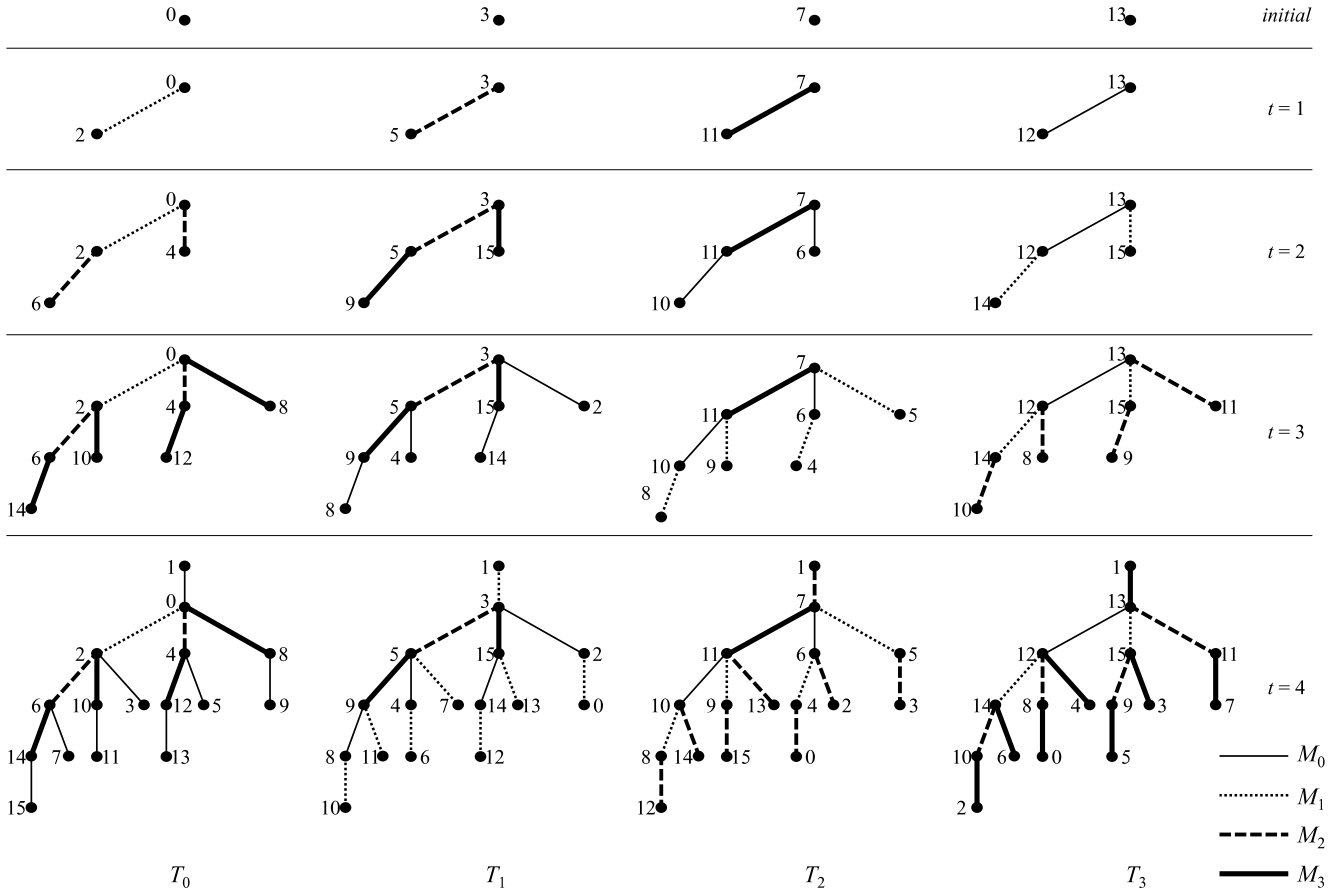


Fig. 4. Four ISTs rooted at vertex 1 in LTQ_4 constructed by Algorithm CONSTRUCT_IST.

4.1. The notations

Definition 3. For $V' \subseteq V(LTQ_n)$, define $f_i(V')$ to be

$$f_i(V') = \{f_i(v) \mid v \in V'\}.$$

Definition 4. For a fixed integer i, $0 \leq i \leq n-1$, define O_i^n to be the *ordered* set

$$O_i^n = \{i, (i-1) \bmod n, (i-2) \bmod n, \ldots, (i-n+1) \bmod n\}.$$

Notice that O_i^n can be obtained by arranging $0, 1, \ldots, n-1$ around a circle, starting from the number i and picking up these n numbers counterclockwise. For example, $O_0^4 = \{0, 3, 2, 1\}$, $O_1^4 = \{1, 0, 3, 2\}$ and $O_3^4 = \{3, 2, 1, 0\}$.

Definition 5. The *Hamming distance* between two vertices $x, y \in V(LTQ_n)$, denoted by $Ham(x, y)$, is the number of positions at which the corresponding symbols are different. More precisely, $Ham(x, y) = |\{i \mid x_i \neq y_i, 0 \leq i \leq n-1\}|$. For two fixed vertices $x, y \in V(LTQ_n)$, suppose $Ham(x, y) = m$. Define $H_i(x, y)$ to be an *ordered* set consisting of the indices of the m different bits, listed according to the order given by O_i^n.

Definition 6. For two fixed vertices $x, y \in V(LTQ_n)$, suppose $H_i(x, y) = \{c_{m-1}, c_{m-2}, \ldots, c_0\}$ with $m \geq 2$ and $H_i(x, y) \neq O_i^n$. We say that j *is between* c_u *and* c_{u-1} *for some* $0 \leq u \leq m-1$ *with respect to* O_i^n if $j \notin H_i(x, y)$ and when $0, 1, \ldots, n-1$ are arranged on a circle, the location of j on the circle is between c_u and c_{u-1}.

For example, consider LTQ_4. Suppose $v = 12$. Then $H_0(v, 0) = \{3, 2\}$, $H_1(v, 3) = \{1, 0, 3, 2\}$, $H_2(v, 7) = \{1, 0, 3\}$ and $H_3(v, 13) = \{0\}$. Since $1 \notin H_0(v, 0)$, 1 is between $c_u = 3$, $c_{u-1} = 2$; $0 \notin H_0(v, 0)$, 0 is between $c_u = 2$, $c_{u-1} = 3$.

Definition 7. For two vertices $x, y \in V(LTQ_n)$, define $\Pi_i(x, y)$ to be the *ordered* set consisting of all the indices of perfect matchings used in the x, y-path in T_i, $0 \leq i \leq n-1$, listed according to the order from x to y.

For example, consider T_2 rooted at vertex 1 of LTQ_4 in Fig. 4. Suppose $v = 12$. Then $\Pi_2(v, 7) = \{2, 1, 0, 3\}$. Moreover, the path from v to 7 is

$$1100 \xrightarrow{M_2} 1000 \xrightarrow{M_1} 1010 \xrightarrow{M_0} 1011 \xrightarrow{M_3} 0111.$$

Definition 8. Define $I(a, b)$, where $a \geq b$, to be the sequence such that

$$I(a, b) = \begin{cases} a, a-1, \ldots, b+1 & \text{if } a > b, \\ a & \text{if } a = b. \end{cases}$$

4.2. The spanning trees

Throughout this subsection, let $T_0, T_1, \ldots, T_{n-1}$ be the output of Algorithm Construct_IST. The purpose of this subsection is to prove that $T_0, T_1, \ldots, T_{n-1}$ are n spanning trees rooted at r. By Theorem 2.4, we assume $r = 0$ and $r = 1$ as the common roots without loss of generality. To prove that T_i, $0 \leq i \leq n-1$, is a spanning tree rooted at r, we prove the following *loop invariant*:

Loop invariant: At the start of the tth iteration of the outer for-loop, T_i is connected, $|V(T_i)| = 2^{t-1}$ and $|E(T_i)| = |V(T_i)| - 1$.

The loop invariant is trivial true prior to the first loop iteration since in line 3, Algorithm Construct_IST sets $V(T_i) = \{f_i(r)\}$. Hence T_i is connected, $|V(T_i)| = 2^0$ and $|E(T_i)| = |V(T_i)| - 1$. We now prove that if the loop invariant is true before the tth iteration of the outer for-loop, then it remains true before the next iteration. Algorithm Construct_IST first resets S to be empty in line 5. For each vertex v in $V(T_i)$, Algorithm Construct_IST adds the edge (v, u) to T_i in line 8, where $u = f_{(i+t) \bmod n}(v)$, by using the edges in $M_{(i+t) \bmod n}$, and adds u to S in line 9. Since each newly generated edge is incident to a vertex in $V(T_i)$, T_i remains to be connected. Now we claim that

Claim 4.1. $V(T_i) \cap S = \emptyset$.

If Claim 4.1 is true, then at the end of the inner for-loop, the newly generated edges between $V(T_i)$ and S clearly form a matching that saturates $V(T_i)$ and S. Thus $|V(T_i)| = |S|$. Consequently, after the tth iteration of the outer for-loop, T_i is connected, $|V(T_i)| = 2^{t-1} + 2^{t-1} = 2^t$ and $|E(T_i)| = 2^{t-1} - 1 + 2^{t-1} = 2^t - 1 = |V(T_i)| - 1$. When the outer for-loop terminates, $t = n + 1$. Therefore, T_i is connected, $|V(T_i)| = 2^n$ and $|E(T_i)| = |V(T_i)| - 1$. Also, at the end of the $(t = n)$th iteration of the outer for-loop, Algorithm Construct_IST adds the edge $(r, f_i(r))$ to T_i. Therefore T_i is a spanning tree rooted at r of LTQ_n. In the following, we prove that Claim 4.1 is true for $r = 0$ and $r = 1$. We first consider the case of $r = 0$.

Lemma 4.2. *Claim 4.1 is true for $r = 0$.*

Proof. Consider the tth iteration of the outer for-loop. Set $k = (i + t) \bmod n$ for easy writing. Let $v \in V(T_i)$ and $u \in S$. If $t \in \{1, 2, \ldots, n-1\}$, then $(v_k, u_k) = (0, 1)$. If $t = n$, then we have $(v_i, u_i) = (1, 0)$. Therefore $V(T_i) \cap S = \emptyset$. □

Lemma 4.3. *Claim 4.1 is true for $r = 1$.*

Proof. Consider T_i, $0 \leq i \leq n-1$. Set $k = (i + t) \bmod n$ for easy writing. Let $v \in V(T_i)$ and $u \in S$.

Case 1: $i = 0$. If $t \in \{1, 2, \ldots, n-1\}$, then $(v_k, u_k) = (0, 1)$. If $t = n$, then $(v_i, u_i) = (1, 0)$. Therefore $V(T_i) \cap S = \emptyset$.

Case 2: $i = n - 1$. If $t \in \{1, 2, \ldots, n-2\}$, then $(v_k, u_k) = (0, 1)$. If $t = n - 1$, then we have $(v_{n-2}, u_{n-2}) = (1, 0)$. If $t = n$, then we have $(v_i, u_i) = (1, 0)$. Therefore $V(T_i) \cap S = \emptyset$.

Case 3: $i \in \{1, 2, \ldots, n-2\}$. We further divide this case into two subcases.

Subcase 3.1: $t \in \{1, 2, \ldots, n-2\}$. The proof of this case is the same as Case 2.

Subcase 3.2: $t = n$. By the loop invariant, T_i induces a tree before the tth iteration of the outer for-loop. Partition $V(T_i)$ into V_0 and V_1 as follows:

$$V_0 = \{\text{all the vertices in the subtree rooted at } f_{i+1}(f_i(1))\} \quad \text{and} \quad V_1 = V(T_i) \setminus V_0.$$

See Fig. 5 for an illustration.

By (1) and by Lemma 4.6, we have: (i) the ith bit of all the vertices in V_0 is 0 and hence the ith bit of all the vertices in $f_i(V_0)$ is 1, and (ii) the ith bit of all the vertices in V_1 is 1 and hence the ith bit of all the vertices in $f_i(V_1)$ is 0. Notice that

$$S = f_i(V_0) \cup f_i(V_1).$$

Therefore, to prove Claim 4.1, it suffices to prove that

$$V_0 \cap f_i(V_1) = \emptyset \quad \text{and} \quad V_1 \cap f_i(V_0) = \emptyset. \tag{3}$$

If $i = n - 2$, then the $(n-1)$-bit of all the vertices in V_0 and $f_{n-2}(V_0)$ is 1; however, the $(n-1)$-bit of all the vertices in V_1 and $f_{n-2}(V_1)$ is 0. Thus when $i = n - 2$, $V_0 \cap f_{n-2}(V_1) = \emptyset$ and $V_1 \cap f_{n-2}(V_0) = \emptyset$. Now suppose $i \in \{1, 2, \ldots, n-3\}$. Partition V_0 into $V_{0,0}$ and $V_{0,1}$ such that

$$V_{0,0} = \{\text{all the vertices in the subtree rooted at } f_{i+2}(f_{i+1}(f_i(1)))\} \quad \text{and} \quad V_{0,1} = V_0 \setminus V_{0,0}.$$

Partition V_1 into $V_{1,0}$ and $V_{1,1}$ such that

$$V_{1,0} = \{\text{all the vertices in the subtree rooted at } f_{i+2}(f_i(1))\} \quad \text{and} \quad V_{1,1} = V_1 \setminus V_{1,0}.$$

By (1) and Lemma 4.6, the pair of the $(i+1)$th and the ith bit of all the vertices in $V_{0,0}$ and $f_i(V_{1,1})$ is (0, 0); in $f_i(V_{0,0})$ and $V_{1,1}$ is (0, 1); in $V_{0,1}$ and $f_i(V_{1,0})$ is (1, 0) and in $f_i(V_{0,1})$ and $V_{1,0}$ is (1, 1). Thus to prove (3), it suffices to prove that

$$V_{0,0} \cap f_i(V_{1,1}) = \emptyset, \qquad V_{1,1} \cap f_i(V_{0,0}) = \emptyset, \qquad V_{1,0} \cap f_i(V_{0,1}) = \emptyset \quad \text{and} \quad V_{0,1} \cap f_i(V_{1,0}) = \emptyset. \tag{4}$$

For $v = v_{n-1}, v_{n-1}, \ldots, v_0 \in V(LTQ_n)$ with $v \neq 0$, let q be the largest index of v such that $v_q = 1$. If $v = 0$, then let $q = -1$. By (1) and Lemma 4.6, we have Table 2.

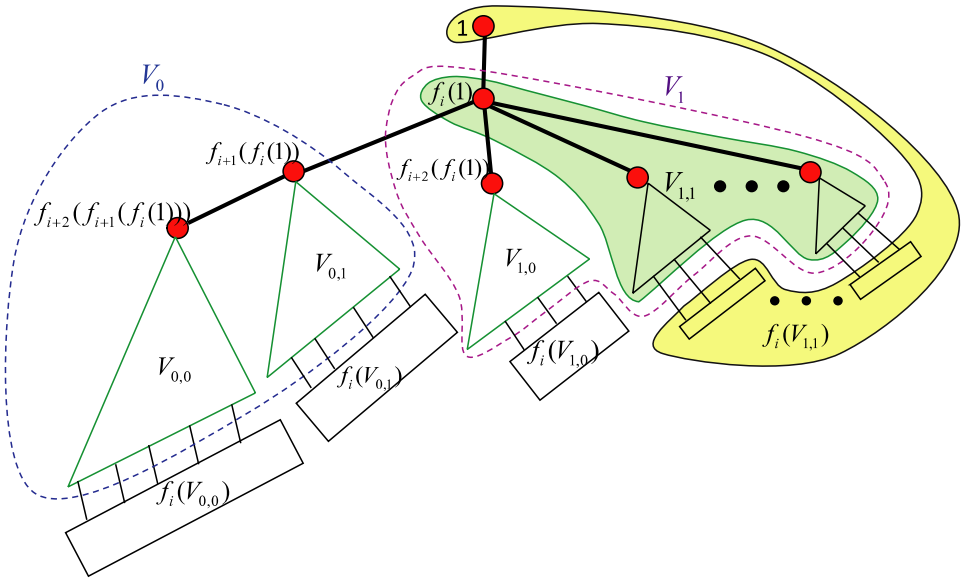


Fig. 5. An illustration for the proof of Lemma 4.3.

Table 2
The value of q for every vertex in the given set.

$V_{0,0} \cup f_i(V_{0,0})$	$V_{1,1} \cup f_i(V_{1,1})$	$V_{1,0} \cup f_i(V_{1,0})$	$V_{0,1} \cup f_i(V_{0,1})$
$q \geq i+2$	$q \leq i+1$ or $q \geq i+3$	$q \geq i+3$	$q = i+1$ or $q \geq i+3$

We first prove that $V_{0,0} \cap f_i(V_{1,1}) = \emptyset$ and $V_{1,1} \cap f_i(V_{0,0}) = \emptyset$. By Table 2, each vertex in $V_{1,1} \cap f_i(V_{1,1})$ with $q \leq i+1$ does not belong to $V_{0,0} \cup f_i(V_{0,0})$ since every vertex in $V_{0,0} \cup f_i(V_{0,0})$ has $q \geq i+2$. Also, each vertex in $V_{0,0} \cup f_i(V_{0,0})$ with $q = i+2$ does not belong to $V_{1,1} \cap f_i(V_{1,1})$ since each vertex in $V_{1,1} \cap f_i(V_{1,1})$ has $q \neq i+2$. Thus, we may focus on vertices with $q = i+3$ or $q > i+3$. Note that each vertex in $V_{0,0} \cup f_i(V_{0,0})$ with $q = i+3$ has its $(i+2)$th bit to be 0; however, from Table 2, we know that each vertex in $f_i(V_{1,1}) \cup V_{1,1}$ with $q \geq i+3$ has its $(i+2)$th bit to be 1. Therefore, each vertex in $V_{0,0} \cup f_i(V_{0,0})$ with $q = i+3$ does not belong to $V_{1,1} \cup f_i(V_{1,1})$. It remains to consider the vertices with $q > i+3$. For each $x \in V_{0,0} \cup f_i(V_{0,0})$, the bit string of x formed by x_q to x_{i+2} is in

$$L_0 = \{\underbrace{1\overbrace{00\cdots0}^{q-i-2\ 0\text{'s}}}_{q-i-1\ \text{bits}}, \underbrace{1\overbrace{00\cdots0}^{q-i-4\ 0\text{'s}}11}_{q-i-1\ \text{bits}}, \underbrace{1\overbrace{00\cdots0}^{q-i-5\ 0\text{'s}}101}_{q-i-1\ \text{bits}}, \underbrace{1\overbrace{00\cdots0}^{q-i-6\ 0\text{'s}}1001}_{q-i-1\ \text{bits}}, \ldots, \underbrace{101\overbrace{00\cdots0}^{q-i-5\ 0\text{'s}}1}_{q-i-1\ \text{bits}}, \underbrace{11\overbrace{00\cdots0}^{q-i-4\ 0\text{'s}}1}_{q-i-1\ \text{bits}}\}.$$

However, for each $y \in V_{1,1} \cup f_i(V_{1,1})$, the bit string of y formed by y_q to y_{i+2} is in

$$L_1 = \{\underbrace{1\overbrace{00\cdots0}^{q-i-3\ 0\text{'s}}1}_{q-i-1\ \text{bits}}, \underbrace{1\overbrace{00\cdots0}^{q-i-4\ 0\text{'s}}10}_{q-i-1\ \text{bits}}, \underbrace{1\overbrace{00\cdots0}^{q-i-5\ 0\text{'s}}100}_{q-i-1\ \text{bits}}, \underbrace{1\overbrace{00\cdots0}^{q-i-6\ 0\text{'s}}1000}_{q-i-1\ \text{bits}}, \ldots, \underbrace{101\overbrace{00\cdots0}^{q-i-4\ 0\text{'s}}}_{q-i-1\ \text{bits}}, \underbrace{11\overbrace{00\cdots0}^{q-i-3\ 0\text{'s}}}_{q-i-1\ \text{bits}}\}.$$

It is not difficult to check that $L_0 \cap L_1 = \emptyset$. Hence we have $V_{0,0} \cap f_i(V_{1,1}) = \emptyset$ and $V_{1,1} \cap f_i(V_{0,0}) = \emptyset$.

Similar arguments can show that $V_{0,1} \cap f_i(V_{1,0}) = \emptyset$ and $V_{1,0} \cap f_i(V_{0,1}) = \emptyset$, except that $V_{0,0} \cup f_i(V_{0,0})$ is replaced by $V_{1,0} \cup f_i(V_{1,0})$ and $V_{1,1} \cup f_i(V_{1,1})$ is replaced by $V_{0,1} \cup f_i(V_{0,1})$. From the above discussion, we have (4) and hence have (3). Therefore $V(T_i) \cap S = \emptyset$. □

By Theorem 2.4 and Lemmas 4.2 and 4.3, we have the following result.

Lemma 4.4. *$T_0, T_1, \ldots, T_{n-1}$ are n spanning trees rooted at r for LTQ_n.*

4.3. The vertex-independency of the n spanning trees

In this subsection, we show that $T_0, T_1, \ldots, T_{n-1}$ generated by Algorithm Construct_IST are vertex-independent trees rooted at an arbitrary vertex r for LTQ_n. By Theorem 2.4, without loss of generality, we may assume $r = 0$ and $r = 1$ as the common roots. To this end, we need to show that for any i, j with $0 \leq i < j \leq n-1$ and for each $v(\neq r) \in V(LTQ_n)$, the r, v-path in T_i and the r, v-path in T_j are internally vertex-disjoint. Recall that the child of the root in T_i and T_j are $f_i(r)$ and $f_j(r)$, respectively. In the following, we further assume $v \notin \{r, f_i(r), f_j(r)\}$ since if $v \in \{r, f_i(r), f_j(r)\}$, then the r, v-path in T_i and the r, v-path in T_j are clearly internally vertex-disjoint. Let $parent_i(v)$ (resp., $parent_j(v)$) be the parent of vertex v in T_i

(resp., T_j). Let P_1 (resp., P_2) be the $parent_i(v), f_i(r)$-path (resp., $parent_j(v), f_j(r)$-path) in T_i (resp., T_j). Since $f_i(r) \neq f_j(r)$, the r, v-path in T_i and the r, v-path in T_j are internally vertex-disjoint if and only if $V(P_1) \cap V(P_2) = \emptyset$. We prove T_i and T_j are vertex-independent by showing the following claim:

Claim 4.5. $V(P_1) \cap V(P_2) = \emptyset$.

Before proving Claim 4.5, we need a lemma.

Lemma 4.6. *T_i, $0 \leq i \leq n-1$, constructed by Algorithm Construct_IST has the property that for each $v \in V(LTQ_n) \setminus \{r, f_i(r)\}$, the path from v to $f_i(r)$ in T_i uses each perfect matching in $\{M_0, M_1, \ldots, M_{n-1}\}$ at most once.*

Proof. It follows from the fact that $f_{(i+t) \bmod n}$ used in the for-loop between the inner for-loop are distinct when the outer for-loop iterates from $t = 1$ to $t = n$. □

We first consider the case of $r = 0$.

Lemma 4.7. *$T_0, T_1, \ldots, T_{n-1}$ are n vertex-independent trees rooted at $r = 0$ for LTQ_n.*

Proof. To prove Claim 4.5, we first describe the path from v to the child of the root in T_i when $r = 0$. For any $v \in V(T_i) \setminus \{0, f_i(0)\}$, the $v, f_i(0)$-path in T_i can be determined by $\Pi_i(v, f_i(0))$. In addition, $\Pi_i(v, f_i(0))$ can be determined by $H_i(v, f_i(0))$ as follows. Suppose $v = v_{n-1}v_{n-2}\ldots v_0$ and $H_i(v, f_i(0)) = \{c_{m-1}, c_{m-2}, \ldots, c_0\}$. If $v_0 = 0$, then $\Pi_i(v, f_i(0))$ can be determined by

$$\Pi_i(v, f_i(0)) = \begin{cases} H_i(v, f_i(0)) & \text{if } i \neq 0, \\ \{c_{m-1} = 0, I(c_{m-2}, c_{m-3}), \ldots, I(c_3, c_2), I(c_1, c_0)\} & \text{if } i = 0 \text{ and } m-1 \text{ is even}, \\ \{c_{m-1} = 0, I(c_{m-2}, c_{m-3}), \ldots, I(c_2, c_1), I(c_0, 0)\} & \text{if } i = 0 \text{ and } m-1 \text{ is odd}. \end{cases} \tag{5}$$

If $v_0 = 1$ and $i \neq 0$, then $H_i(v, f_i(0))$ must contain 0; in this case, we assume $c_e = 0$ for some e. Thus if $v_0 = 1$, $\Pi_i(v, f_i(0))$ can be determined by

$$\Pi_i(v, f_i(0)) = \begin{cases} \{I(c_{m-1}, c_{m-2}), I(c_{m-3}, c_{m-4}), \ldots, I(c_1, c_0)\} & \text{if } i = 0 \text{ and } m \text{ is even}, \\ \{I(c_{m-1}, c_{m-2}), I(c_{m-3}, c_{m-4}), \ldots, I(c_2, c_1), I(c_0, 0)\} & \text{if } i = 0 \text{ and } m \text{ is odd}, \\ \{I(c_{m-1}, c_{m-2}), I(c_{m-3}, c_{m-4}), \ldots, I(c_{e+2}, c_{e+1}), c_e, c_{e-1}, \ldots, c_0\} & \text{if } i \neq 0 \text{ and } m-e \text{ is odd}, \\ \{I(c_{m-1}, c_{m-2}), I(c_{m-3}, c_{m-4}), \ldots, I(c_{e+1}, 0), c_e, c_{e-1}, \ldots, c_0\} & \text{if } i \neq 0 \text{ and } m-e \text{ is even}. \end{cases} \tag{6}$$

Now we show that Claim 4.5 is true for $r = 0$. Suppose not, then there exists a vertex a $(\neq v) \in V(P_1) \cap V(P_2)$. Suppose

$$H_i(v, f_i(0)) = H_i(v, 2^i) = \{c_{m-1}, c_{m-2}, \ldots, c_0\}. \tag{7}$$

There are four cases.

Case 1: $v_i = 1$ and $v_j = 1$. Then there must exist u such that $c_u = j$. Thus

$$H_j(v, f_j(0)) = H_j(v, 2^j) = \{c_{u-1}, c_{u-2}, \ldots, c_0, i, c_{m-1}, c_{m-2}, \ldots, c_{u+1}\}. \tag{8}$$

By (5)–(7), c_{m-1} is the first element in $\Pi_i(v, 2^i)$. Let $x \in V(P_1)$. Then the (c_{m-1})th bit of x is $v_{c_{m-1}}$ only when (i) $(c_{m-1}+1) \in \Pi_i(v, 2^i)$, and (ii) $c_{m-1} + 1 \geq 2$, and (iii) there exists $q = q_{n-1}q_{n-2}\ldots q_0 \in V(P_1)$ such that $x = f_{c_{m-1}+1}(q)$ and $q_0 = 1$. We now prove that (i)–(iii) will not occur simultaneously; hence for all $x \in V(P_1)$, the (c_{m-1})th bit of x is $\overline{v}_{c_{m-1}}$. If $|H_i(v, 2^i)| = 1$, then (i) cannot occur. Suppose $|H_i(v, 2^i)| \geq 2$ and both (i) and (iii) occur; that is, there exists $q = q_{n-1}q_{n-2}\ldots q_0 \in V(P_1)$ such that $x = f_{c_{m-1}+1}(q)$ and $q_0 = 1$. By (7), $c_{m-1} + 1$ is the last element in $\Pi_i(v, 2^i)$. Since $q_0 = 1$, $I(c_0, 0) \subseteq \Pi_i(v, 2^i)$. By Lemma 4.6 and by the fact that $I(c_0, 0) = \{c_0, c_0 - 1, \ldots, 1\}$, we have $c_{m-1} + 1 = 1$; thus (ii) does not occur and consequently the (c_{m-1})th bit of all the vertices in $V(P_1)$ is $\overline{v}_{c_{m-1}}$. Since $v_i = 1$, the ith bit of all the vertices in $V(P_1)$ is 1. By (5) and (6) and (8), the (c_{m-1})th bit of those vertices in $V(P_2)$ with the ith bit being 1 is $v_{c_{m-1}}$. Thus no such a exists and Claim 4.5 is true.

Case 2: $v_i = 0$ and $v_j = 0$. Then $c_{m-1} = i$. If $|H_i(v, 2^i)| = 1$, then $H_i(v, 2^i) = \{i\}$, which implies that $v = 0$; this contradicts to the assumption that $v \neq 0$. Thus $|H_i(v, 2^i)| \geq 2$ and there must exist u such that j is between c_u and c_{u-1} with respect to O_i^n. Thus

$$H_j(v, 2^j) = \begin{cases} \{j, c_{m-2}, c_{m-3}, \ldots, c_{u+1}, c_{u=0}\} & \text{if } j = i+1, \\ \{j, c_{u-1}, c_{u-2}, \ldots, c_0, c_{m-2}, c_{m-3}, \ldots, c_{u+1}\} & \text{if otherwise}. \end{cases} \tag{9}$$

By (5)–(7), the ith bit of all vertices in $V(P_1)$ is 1. By (5) and (6) and (9), the jth bit of all the vertices in $V(P_2)$ is 1. The ith bit and the jth bit of a are both 1. If $I(c_u, c_{u-1}) \nsubseteq \Pi_i(v, 2^i)$, each vertex in $V(P_1)$ has its jth bit to be 0. If (i) $j \neq i+1$ and $I(c_0, c_{m-2}) \nsubseteq \Pi_j(v, 2^j)$, or if (ii) $j = i+1$ and $v_0 \neq 1$, then each vertex in $V(P_2)$ has its ith bit to be 0. Thus the existence of a implies that $I(c_u, c_{u-1}) \subseteq \Pi_i(v, 2^i)$ and $I(c_0, c_{m-2}) \subseteq \Pi_j(v, 2^j)$. Note that $I(c_u, c_{u-1}) \subseteq \Pi_i(v, 2^i)$ implies that $i = 0$ and

hence $v_0 = 0$ (since case 2 requires $v_i = 0$). However, $I(c_0, c_{m-2}) \subseteq \Pi_j(v, 2^j)$ implies $v_0 = 1$, which contradicts to $v_0 = 0$. Thus no such a exists and Claim 4.5 is true.

Case 3: $v_i = 0$ and $v_j = 1$. Then $c_{m-1} = i$ and there must exist u such that $c_u = j$. If $|H_i(v, 2^i)| = 1$, then $H_j(v, 2^i) = \emptyset$. This implies that $v = 2^j$, which contradicts to the assumption that $v \neq 2^j$. Thus

$$H_j(v, 2^j) = \{c_{u-1}, c_{u-2}, \ldots, c_0, c_{m-2}, c_{m-3}, \ldots, c_{u+1}\}. \tag{10}$$

By (5)–(7), the ith bit of all vertices in $V(P_1)$ is 1. The ith bit of a is 1. If $I(c_0, c_{m-2}) \nsubseteq \Pi_j(v, 2^j)$, each vertex in $V(P_2)$ has its ith bit to be 0. Thus the existence of a implies that $I(c_0, c_{m-2}) \subseteq \Pi_j(v, 2^j)$, which further implies $v_0 = 1$. Since $I(c_0, c_{m-2}) \subseteq \Pi_j(v, 2^j)$, $V(P_2)$ has only one vertex $x = x_{n-1}x_{n-2}\ldots x_0$ such that $x_i = 1$ and $x = f_{i+1}(q)$ for some $q \in V(P_2)$. The existence of a implies that $x = a$. Since $v_0 = 1$, $\Pi_i(v, 2^i)$ starts with $I(i, c_{m-2})$, i.e., $\Pi_i(v, 2^i)$ is of the form $\{I(i, c_{m-2}), \ldots\}$. By (6), c_{m-3} is the first element after $I(i, c_{m-2})$ in $\Pi_i(v, 2^i)$. Recall that $\Pi_i(v, 2^i)$ is an ordered set of all the indices of perfecting matchings used in the $v, 2^i$-path in T_i listed according to the order from v to 2^i. Thus the first vertex in $V(P_1)$ can be obtained by applying the first perfect matching obtained from the first element in $\Pi_i(v, 2^i)$ to v, the second vertex in $V(P_1)$ can be obtained by applying the second perfect matching obtained from the second element in $\Pi_i(v, 2^i)$ to the first vertex in $V(P_1)$, and so on. Thus we can partition $V(P_1)$ into $V_{1,1}$ and $V_{1,2}$ such that $V_{1,1}$ consists of those vertices in $V(P_1)$ before $f_{c_{m-3}}$ is applied and $V_{1,2} = V(P_1) - V_{1,1}$. Let $y = y_{n-1}y_{n-2}\ldots y_0$ be an arbitrary vertex in $V_{1,1}$. Then $Ham(y_iy_{i-1}\ldots y_{c_{m-2}}, v_iv_{i-1}\ldots v_{c_{m-2}}) = 2$. However, $Ham(x_ix_{i-1}\ldots x_{c_{m-2}}, v_iv_{i-1}\ldots v_{c_{m-2}}) = 0$. Thus $x \notin V_{1,1}$. On the other hand, $x_{c_{m-3}} = v_{c_{m-3}}$ but the (c_{m-3})th bit of all the vertices in $V_{1,2}$ is $\overline{v}_{c_{m-3}}$; thus $x \notin V_{1,2}$. Since $x \notin V_{1,1}$ and $x \notin V_{1,2}$, we have $x \notin V(P_1)$. Since $x = a$, it follows that $a \notin V(P_1)$. Thus no such a exists and Claim 4.5 is true.

Case 4: $v_i = 1$ and $v_j = 0$. Then there must exist u such that j is between c_u and c_{u-1} with respect to O_i^n. Thus

$$H_j(v,2^j) = \begin{cases} \{j, i, c_{m-1}, c_{m-2}, \ldots, c_{u=0}\} & \text{if } i \text{ is between } c_0 \text{ and } c_{m-1} \text{ with respect to } O_i^n, \\ \{j, c_{u-1}, c_{u-2}, \ldots, c_0, i, c_{m-1}, c_{m-2}, \ldots, c_u\} & \text{if otherwise.} \end{cases} \tag{11}$$

By (5), (6) and (11), the jth bit of all vertices in $V(P_2)$ is 1. Since $v_i = 1$, the ith bit of all the vertices in $V(P_1)$ is 1. The ith bit and the jth bit of a are both 1. By (11), we have two subcases.

Subcase 4.1: i is between c_0 and c_{m-1} with respect to O_i^n. Then $V(P_2)$ has only one vertex $f_j(v)$ with its ith bit and jth bit both being 1. By (5)–(7), c_{m-1} is the first element in $\Pi_i(v, 2^i)$. Thus the (c_{m-1})th bit of those vertices in $V(P_1)$ with the jth bit being 1 is $\overline{v}_{c_{m-1}}$. However, by (5), (6) and (11), the (c_{m-1})th bit of $f_j(v)$ is $v_{c_{m-1}}$. Thus no such a exists and Claim 4.5 is true.

Subcase 4.2: i is not between c_0 and c_{m-1} with respect to O_i^n. By (5), (6) and (11), the ith bit of all the vertices in $V(P_1)$ is 1. If $|H_i(v, 2^i)| = 1$, then $H_i(v, 2^i) = \{c_0\}$; since $v_j = 0$, we have $c_0 \neq j$, which implies that each vertex in $V(P_1)$ has its jth bit to be 0 and consequently no such a exists and Claim 4.5 is true. Now suppose $|H_i(v, 2^i)| \geq 2$. Then when $I(c_u, c_{u-1}) \nsubseteq \Pi_i(v, 2^i)$, each vertex in $V(P_1)$ has its jth bit to be 0. Thus the existence of a implies that $I(c_u, c_{u-1}) \subseteq \Pi_i(v, 2^i)$. Since $I(c_u, c_{u-1}) \subseteq \Pi_i(v, 2^i)$, $V(P_1)$ has only one vertex $x = x_{n-1}x_{n-2}\ldots x_0$ such that $x_j = 1$ and $x = f_{j+1}(q)$ for some $q \in V(P_1)$. The existence of a implies that $x = a$. By (5), (6) and (11), the (c_{m-1})th bit of those vertices in $V(P_2)$ with the ith bit being 1 is $v_{c_{m-1}}$. However, the $x_{c_{m-1}} = \overline{v}_{c_{m-1}}$. So if $x \in V(P_1)$, then $x \notin V(P_2)$. Thus no such a exists and Claim 4.5 is true. □

From the above discussion, Claim 4.5 is true and therefore $T_0, T_1, \ldots, T_{n-1}$ are vertex-independent rooted at $r = 0$ of LTQ_n. □

Now we consider the case of $r = 1$.

Lemma 4.8. *$T_0, T_1, \ldots, T_{n-1}$ are n vertex-independent trees rooted at $r = 1$ for LTQ_n.*

Proof. To prove Claim 4.5, we first describe the path from v to the child of the root in T_i when $r = 1$. For any $v \in V(T_i) \setminus \{1, f_i(1)\}$, the $v, f_i(1)$-path in T_i can be determined by $\Pi_i(v, f_i(1))$. Furthermore, $\Pi_i(v, f_i(1))$ can be determined by the ordered set $H_i(v, f_i(1))$ as follows. Suppose $v = v_{n-1}v_{n-2}\ldots v_0$ and $H_i(v, f_i(1)) = \{c_{m-1}, c_{m-2}, \ldots, c_0\}$. Let c_{e-1} be the first (from bit c_{m-1} to c_0) member in $H_i(v, f_i(1))$ that is larger than i. If $i = 0$, $\Pi_i(v, f_i(1))$ can be determined by

$$\Pi_0(v, f_0(1)) = H_0(v, f_0(1)). \tag{12}$$

If $i \neq 0$ and $v_0 = 0$, we have $c_e = 0$ for some e. Thus $\Pi_i(v, f_i(1))$ can be determined by

$$\Pi_i(v, f_i(1)) = \begin{cases} \{c_{m-1}, c_{m-2}, \ldots, c_e, I(c_{e-1},c_{e-2}), I(c_{e-3},c_{e-4}), \ldots, I(c_1, c_0)\} & \text{if } e \text{ is even,} \\ \{c_{m-2}, c_{m-3}, \ldots, c_e, I(c_{e-1},c_{e-2}), I(c_{e-3},c_{e-4}), \ldots, I(c_0, i)\} & \text{if } e \text{ is odd and } c_{m-1} = i, \\ \{i, c_{m-1}, c_{m-2}, \ldots, c_e, I(c_{e-1},c_{e-2}), I(c_{e-3},c_{e-4}), \ldots, I(c_0, i)\} & \text{if } e \text{ is odd and } c_{m-1} \neq i. \end{cases} \tag{13}$$

When $i \neq 0$ and $v_0 = 1$, in order to obtain $\Pi_i(v, f_i(1))$ from $H_i(v, f_i(1))$, the following notations are introduced. Define H_i^1 to be the sequence

$$H_i^1 = \begin{cases} c_{m-1}, c_{m-2}, \ldots, c_e & \text{if } |H_i^2| \text{ is even,} \\ i, c_{m-1}, c_{m-2}, \ldots, c_e & \text{if } |H_i^2| \text{ is odd and } c_{m-1} \neq i \\ c_{m-2}, c_{m-3}, \ldots, c_e & \text{if } |H_i^2| \text{ is odd and } c_{m-1} = i, \end{cases}$$

and define H_i^2 to be the sequence

$$H_i^2 = c_{e-1}, c_{e-2}, \ldots, c_0.$$

Define $\zeta_i(v, f_i(1))$ to be the sequence

$$\zeta_i(v, f_i(1)) = \begin{cases} H_i^1, H_i^2 & \text{if } |H_i^1| \text{ is even and } |H_i^2| \text{ is even,} \\ H_i^1, H_i^2, i & \text{if } |H_i^1| \text{ is even and } |H_i^2| \text{ is odd,} \\ H_i^1, 0, H_i^2 & \text{if } |H_i^1| \text{ is odd and } |H_i^2| \text{ is even,} \\ H_i^1, 0, H_i^2, i & \text{if } |H_i^1| \text{ is odd and } |H_i^2| \text{ is odd.} \end{cases} \tag{14}$$

Suppose

$$\zeta_i(v, f_i(1)) = \varsigma_u, \varsigma_{u-1}, \ldots, \varsigma_0.$$

Thus if $i \neq 0$ and $v_0 = 1$, $\Pi_i(v, f_i(1))$ can be determined by

$$\Pi_i(v, f_i(1)) = \{I(\varsigma_u, \varsigma_{u-1}), I(\varsigma_{u-2}, \varsigma_{u-3}), \ldots, I(\varsigma_1, \varsigma_0), \}. \tag{15}$$

Now we show that Claim 4.5 is true for $r = 1$. Suppose not, then there exists a vertex a $(\neq v) \in V(P_1) \cap V(P_2)$. Suppose

$$H_i(v, f_i(1)) = \{c_{m-1}, c_{m-2}, \ldots, c_0\}. \tag{16}$$

There are four cases.

Case 1: $0 = i < j \leq n - 1$. The proof of this case is divided into two parts, depending on $v_0 = 1$ or $v_0 = 0$. Suppose $v_0 = 1$. Then $0 \notin H_j(v, f_j(1))$. Thus the 0th bit of all the vertices in $V(P_2)$ is 1. By (12) and (16), 0 is the first element in $H_0(v, f_0(1))$; this implies that the 0th bit of all the vertices in $V(P_1)$ is 0. Thus no such a exists. In the following, we assume $v_0 = 0$. Then $0 \notin H_0(v, f_0(1))$. The 0th bit of all the vertices in $V(P_1)$ is 0; this implies that the 0th bit of a is 0. There are two possibilities: $j = 1$ or $j > 1$.

Subcase 1.1: $j = 1$. Note that either $1 \in \Pi_1(v, f_1(1))$ or $1 \notin \Pi_1(v, f_1(1))$. If $1 \notin \Pi_1(v, f_1(1))$, then 0 is the first element in $\Pi_1(v, f_1(1))$. This implies that the 0th bit of all the vertices in $V(P_2)$ is 1. Thus no such a exists. If $1 \in \Pi_1(v, f_1(1))$, then 1 and 0 are the first element and the second element in $\Pi_1(v, f_1(1))$, respectively. Thus the 0th bit of all the vertices in $V(P_2) \setminus \{f_1(v)\}$ is 1. The existence of a implies that $f_1(v) = a$.

If $v_1 = 0$, then $1 \notin H_0(v, f_0(1))$. This implies that the 1st bit of all the vertices in $V(P_1)$ is 0. However, it is obvious that the 1st bit of $f_1(v)$ is 1. Therefore $f_1(v) \notin V(P_1)$. Thus no such a exists. Now suppose $v_1 = 1$. Since $1 \in \Pi_1(v, f_1(1))$, there must exist some $k > 1$ such that $v_k = 1$; this implies that $c_{m-1} > 1$. By (12) and (16), the (c_{m-1})th bit of all the vertices in $V(P_1)$ is $\overline{v}_{c_{m-1}}$. However, the (c_{m-1})th bit of $f_1(v)$ is $v_{c_{m-1}}$. Therefore $f_1(v) \notin V(P_1)$. Thus no such a exists and $V(P_1) \cap V(P_2) = \emptyset$.

Subcase 1.2: $j > 1$. By (12), (13) and (16), we have: c_{m-1} is the first element in $H_i(v, f_i(1))$, $c_{m-1} \in H_j(v, f_j(1))$, $0 \in H_j(v, f_j(1))$, and c_{m-1} appears after 0 in the ordered set $H_j(v, f_j(1))$. Thus the (c_{m-1})th bit of all the vertices in $V(P_1)$ is $\overline{v}_{c_{m-1}}$. However, the (c_{m-1})th bit of those vertices with the 0th bit being 0 in $V(P_2)$ is $v_{c_{m-1}}$. Thus no such a exists.

From the above discussion, Claim 4.5 is true for *Case* 1.

Case 2: $1 = i < j \leq n - 1$. The proof of this case is divided into two parts, depending on $v_0 = 0$ or $v_0 = 1$.

Subcase 2.1: $v_0 = 0$. Then it is not difficult to see (by comparing the jth and the 0th bits of $f_j(v)$ and all the vertices in $V(P_1)$) that $f_j(v) \notin V(P_1)$. Thus a can not be $f_j(v)$. It remains to consider those vertices in $V(P_2) \setminus f_j(v)$. The remaining proof is further divided into two parts, depending on $v_{j-1} = 0$ or $v_{j-1} = 1$.

Subcase 2.1.1: $v_{j-1} = 0$. Since $v_0 = 0$ and $v_{j-1} = 0, j - 1 \in \Pi_j(v, f_j(v))$. Since $v_0 = 0$ and $j - 1 \in \Pi_j(v, f_j(v))$, the $(j - 1)$th bit of all the vertices in $V(P_2) \setminus f_j(v)$ is 1. However, the $(j - 1)$th bit of all the vertices in $V(P_1)$ is 0. Thus no such a exists and Claim 4.5 is true.

Subcase 2.1.2: $v_{j-1} = 1$. We claim that: the bits from v_{j-2} to v_2 are all 0, i.e., $v_{j-2} = v_{j-3} = \cdots = v_2 = 0$. Suppose this claim is not true and let k be the largest number between $j - 2$ and 2 (inclusive) such that $v_k = 1$. By (13) and (16), the $(j - 1)$th and the kth bits of all the vertices in $V(P_2) \setminus f_j(v)$ is 1 and 0, respectively. However, the $(j - 1)$th bit of those vertices in $V(P_1)$ with kth bit being 0 is 0. Thus $v_{j-2} = v_{j-3} = \cdots = v_2 = 0$. So the 1st bit of all the vertices in $V(P_1)$ is 1 and the 1st bit of all the vertices in $V(P_2) \setminus f_j(v)$ is 0. Thus no such a exists and Claim 4.5 is true.

Subcase 2.2: $v_0 = 1$. The proof of this part is further divided into six parts as follows.

Subcase 2.2.1: $j = 2$, $v_1 = 1$ and $v_2 = 1$. Since $v_0 = 1$ and $v_1 = 1$ and $v_2 = 1$,

$$H_j(v, f_j(1)) = (c_{m-1}, c_{m-2}, \ldots, c_1).$$

Suppose m is even. Then by (14) and (15),

$$\Pi_i(v, f_i(1)) = \{I(c_{m-1}, c_{m-2}), \ldots, I(c_1, c_0 = 2)\}$$

and

$$\Pi_j(v, f_j(1)) = \{I(2, 0), I(c_{m-1}, c_{m-2}), \ldots, I(c_1, 2)\}.$$

Thus, the 2nd bit of all the vertices in $V(P_1)$ are 1. However, the 2nd bit of all the vertices in $V(P_2)$ are 0. Thus no such a exists. Suppose m is odd. Then by (14) and (15),

$$\Pi_i(v, f_i(1)) = \{1, I(c_{m-1}, c_{m-2}), \ldots, I(c_0, 1)\}$$

and

$$\Pi_j(v, f_j(1)) = \{I(c_{m-1}, c_{m-2}), \ldots, I(c_2, c_1)\}.$$

Hence the 1st bit of all the vertices in $V(P_1)$ is 0. However, the 1st bit of all the vertices in $V(P_2)$ is 1. Thus no such a exists.

Subcase 2.2.2: $j = 2$, $v_1 = 0$ and $v_2 = 1$. Since $v_0 = 1$ and $v_1 = 0$ and $v_2 = 1$, we have $c_{m-1} = 1$, $c_0 = 2$ and

$$H_j(v, f_j(1)) = \{c_{m-1}, c_{m-2}, \ldots, c_1\}.$$

Suppose $m - 1$ is odd. Then by (14) and (15),

$$\Pi_i(v, f_i(1)) = \{I(c_{m-2}, c_{m-3}), \ldots, I(c_0, 1)\}$$

and

$$\Pi_j(v, f_j(1)) = \{1, I(c_{m-2}, c_{m-3}), \ldots, I(c_2, c_1)\}.$$

Thus, the 1st bit of all vertices in $V(P_1)$ are 0. However, the 1st bit of all vertices in $V(P_2)$ is 1. Thus no such a exists. Suppose $m - 1$ is even. Then by (14) and (15),

$$\Pi_i(v, f_i(1)) = \{1, I(c_{m-2}, c_{m-3}), \ldots, I(c_1, c_0)\}$$

and

$$\Pi_j(v, f_j(1)) = \{2, I(c_{m-2}, c_{m-3}), \ldots, I(c_1, 2)\}.$$

Thus, the 2nd bit of all vertices in $V(P_1)$ are 1. However, the 2nd bit of all vertices in $V(P_2)$ is 0. Thus no such a exists.

Subcase 2.2.3: $j = 2$, $v_1 = 1$ and $v_2 = 0$ (resp., $v_1 = 0$ and $v_2 = 0$). Then

$$H_j(v, f_j(1)) = \{2, c_{m-1}, c_{m-2}, \ldots, c_0\}.$$

Suppose m (resp., $m - 1$) is even. Then by (14) and (15), the 2nd bit of all vertices in $V(P_1)$ is 0. However, the 2nd bit of all vertices in $V(P_2)$ is 1. Suppose m (resp., $m - 1$) is odd. Then by (14) and (15), the 1st bit of all vertices in $V(P_1)$ is 0. However, the 1st bit of all vertices in $V(P_2)$ is 1. Thus no such a exists.

Subcase 2.2.4: $j \neq 2$ and $v_{j-1} = 0$. Then the $(j-1)$th bit of all the vertices in $V(P_1)$ are 0. However, the $(j-1)$th bit of all the vertices in $V(P_2)$ are 1. Thus no such a exists.

Subcase 2.2.5: $j \neq 2$, $v_{j-1} = 1$ and at least one of the bits in $v_{j-2}v_{j-3}\ldots v_2$ is 1. Then there exists q such that $v_q = 1$ and q is the largest number between $j - 2$ and 2 (inclusive).

Subcase 2.2.5.1: Suppose $I(j, q) \nsubseteq \Pi_j(v, f_j(1))$. Then the qth and the $(j-1)$th bit of all the vertices in $V(P_2)$ are 0 and 1, respectively; however, the $(j-1)$th bit of those vertices in $V(P_1)$ with the qth bit being 0 is 0. Thus no such a exists.

Subcase 2.2.5.2: Suppose $I(j, q) \subseteq \Pi_j(v, f_j(1))$. Then we partition $V(P_2)$ into $V_{2,1}$ and $V_{2,2}$ such that

$$V_{2,1} = \{\text{all the vertices in } V(P_2) \text{ before the perfect matching } M_q \text{ is applied}\} \text{ and } V_{2,2} = V(P_2) \setminus V_{2,1}.$$

Consider the vertices in $V_{2,1}$. Suppose $v_j = 0$. Since $j \in I(j, q)$, we can compare the jth bit of all vertices in $V(P_1)$ and in $V_{2,1}$ to see that no such a exists. Suppose $v_j = 1$. Then the number of bits in $v_{n-1}v_{n-2}\ldots v_{j+1}$ that are 1 is odd, i.e., $|H_j^2|$ is odd. This implies that $c_{m-1} \neq j$. Since $c_{m-1} \neq j$, by comparing the c_{m-1}th bit of all the vertices in $V(P_1)$ and in $V_{2,1}$, we know that $V(P_1) \cap V_{2,1} = \emptyset$. Consider the vertices in $V_{2,2}$. Then the qth and the $(j-1)$th bit of all the vertices in $V_{2,2}$ are 0 and 1, respectively. However, the $(j-1)$th bit of those vertices in $V(P_1)$ with the qth bit being 0 is 0. Hence $V(P_1) \cap V_{2,2} = \emptyset$. Since $V(P_1) \cap V_{2,1} = \emptyset$ and $V(P_1) \cap V_{2,2} = \emptyset$, no such a exists.

Subcase 2.2.6: $j \neq 2$, $v_{j-1} = 1$ and all the bits in $v_{j-2}v_{j-3}\ldots v_2$ are 0 (i.e., $v_{j-2} = v_{j-3} = \cdots = v_2 = 0$). For convenience, let $t(w_1, w_2)$ denote the number of bits in $v_{w_1}v_{w_1-1}\ldots v_{w_2}$ that are 1. There are three possibilities.

Subcase 2.2.6.1: Suppose $t(n-1, i+1)$ is even. Then $t(n-1, j)$ is odd. Thus the ith bit of all the vertices in $V(P_2)$ is 0. However, the ith bit of all the vertices in $V(P_1)$ is 1. Thus no such a exists.

Subcase 2.2.6.2: Suppose $t(n-1, i+1)$ is odd and $v_j = 0$. Then $t(n-1, j+1)$ is even. Thus the jth bit of all the vertices in $V(P_2)$ is 1. However, the jth bit of all the vertices in $V(P_1)$ is 0. Thus no such a exists.

Subcase 2.2.6.3: Suppose $t(n-1, i+1)$ is odd and $v_j = 1$. Then $t(n-1, j+1)$ is odd. Thus the ith bit of all the vertices in $V(P_1)$ is 0 and the jth bit of all the vertices in $V(P_2)$ is 0. Then the. ith and the jth bit of a are 0. By (15), the $(j-1)$th bit of all the vertices in $V(P_2)$ with the ith and the jth bit be 0 is 1. However, only the vertex $2^{j-1}+1$ in $V(P_1)$ with the $(j-1)$th bit is 1, and the ith and the jth bit are 0. The existence of a implies $a = 2^{j-1}+1$. Since $t(n-1, j+1)$ is odd, there exists $v_k = 1$, where $k > j$. Then it is easy to find that $a \notin V(P_2)$ by comparing the $k$thBthe jth and the ith bit of a and all vertices in $V(P_2)$. Thus no such a exists.

From the above discussion, Claim 4.5 is true for *Case* 2.

Case 3: $3 \leq i+1 = j \leq n-1$. By (12)–(16), we have the following results. Suppose $t(n-1, i+1)$ is odd. Then the ith bit of all vertices in $V(P_1)$ is 0 and $j \notin \Pi_j(v, f_j(1))$; however, the ith bit of all the vertices in $V(P_2)$ is 1. Suppose $t(n-1, i+1)$ is even and $v_j = 0$. Then the jth bit of all the vertices in $V(P_2)$ is 1; however, the jth bit of all the vertices in $V(P_1)$ is 0. Suppose $t(n-1, i+1)$ is even and $v_j = 1$. Then the jth bit of all the vertices in $V(P_2)$ is 0; however, the jth bit of all the vertices in $V(P_1)$ is 1. Thus no such a exists and Claim 4.5 is true.

Case 4: $3 \leq i+1 < j \leq n-1$. We divide the proof into three parts, depending on the values of v_{j-1} and v_{i-1}.

Subcase 4.1: $v_{j-1} = 0$. Then if $j \in \Pi_i(v, f_i(1))$, then $V(P_1)$ has only one vertex (say, vertex x) with its $(j-1)$th bit being 1. By comparing from the jth to the $(i-1)$th bits of x with the jth to the $(i-1)$th bits of each vertex in $V(P_2)$, we have $x \notin V(P_2)$. If $j \in \Pi_j(v, f_j(1))$, then $f_j(v)$ is the unique vertex in $V(P_2)$ with its $(j-1)$th bit being 0. By comparing from the jth to the $(i-1)$th bits of $f_j(v)$ with the jth to the $(i-1)$th bits of each vertex in $V(P_1)$, we have $f_j(v) \notin V(P_1)$. Then by (12)–(16), the $(j-1)$th bit of all the vertices in $V(P_1) \setminus \{x\}$ is 0; however, the $(j-1)$th bit of all the vertices in $V(P_2) \setminus f_j(v)$ is 1. Thus no such a exists.

Subcase 4.2: $v_{i-1} = 0$. Then we can use similar arguments to prove that no such a exists.

Subcase 4.3: $v_{i-1} = 1$ and $v_{j-1} = 1$. By (12)–(15), we have following the results. When $i \in H_i(v, f_i(1))$ and $v_0 = 1$, $V(P_1)$ has only one vertex $f_i(v)$ with the $(i-1)$th bit being 0. It is easy to find $f_i(v) \notin V(P_2)$ by comparing those bits from the $(j-1)$th to the $(i-1)$th of $f_i(v)$ with each vertex in $V(P_2)$. And since the $(i-1)$th bit of all the vertices in $V(P_1) \setminus f_i(v)$ is 1, the existence of a implies that the $(i-1)$th bit of a must be 1.

Partition $V(P_2)$ into two $V_{2,1}$ and $V_{2,2}$ such that

$$V_{2,1} = \{\text{all the vertices in } V(P_2) \text{ before the perfect matching } M_i \text{ is applied}\} \text{ and } V_{2,2} = V(P_2) \setminus V_{2,1}.$$

Thus the $(i-1)$th bit of all the vertices in $V_{2,1}$ is 1, and if a exists, then $a \in V_{2,1}$. We now claim that:

Claim 4.9. *If a exists, then $v_{j-2} = v_{j-3} = \cdots = v_{i+1} = 0$.*

Proof of Claim 4.9. Suppose this claim is not true. Then let q be the largest index between $j-2$ and $i+1$ (inclusive) such that $v_q = 1$. Let $y = y_{n-1}y_{n-2}\ldots y_0$ be an arbitrary vertex in $V_{2,1} \setminus \{f_j(v)\}$. Note that $f_j(v) \in V_{2,1}$ only when $j \in \Pi_j(v, f_j(1))$. Also note that $q \in \Pi_j(v, f_j(1))$. Moreover, if $j \in H_j(v, f_j(1))$, then q is the first element after j in $H_j(v, f_j(1))$; if $j \notin H_j(v, f_j(1))$, then q is the first element in $H_j(v, f_j(1))$. Since q exists, by (13)–(15), the bits $y_{j-2}y_{j-3}\ldots y_{i+1}$ will be different from the bits $v_{j-2}v_{j-3}\ldots v_{i+1}$. However, let $x = x_{n-1}x_{n-2}\ldots x_0$ be an arbitrary vertex in $V(P_1)$. Then the bits $x_{j-2}x_{j-3}\ldots x_{i+1}$ are identical to the bits $v_{j-2}v_{j-3}\ldots v_{i+1}$. Thus every vertex in $V_{2,1} \setminus \{f_j(v)\}$ is not in $V(P_1)$. Although $f_j(v) \in V_{2,1}$, $f_j(v)$ is not in $V(P_1)$ (this can be observed by comparing the jth bit and from the $(j-2)$th to the $(i+1)$th bits of all the vertices in $V(P_1)$ with jth bit and the bits from the $(j-2)$th to the $(i+1)$th bits of $f_j(v)$). Thus $V(P_1) \cap V_{2,1} = \emptyset$. Since if a exists, then $a \in V_{2,1}$. Thus a does not exists and we have this claim. □

By Claim 4.9, in the remaining proof, we assume $v_{i-1} = 1$, $v_{j-1} = 1$ and $v_{j-2} = v_{j-3} = \cdots = v_{i+1} = 0$. For convenience, let t denote the number of bits in $v_{n-1}v_{n-2}\ldots v_{j+1}$ that are 1. We further divided the proof into four subcases.

Subcase 4.3.1: $v_i = 1$ and $v_j = 1$. Suppose t is even. Then the first member in $\Pi_j(v, f_j(1))$ is i. However, $i \notin \Pi_i(v, f_i(1))$. Thus no such a exists and $V(P_1) \cap V(P_2) = \emptyset$. Suppose t is odd. Then $j \in \Pi_j(v, f_j(1))$ and $I(j-1, i) \subset \Pi_i(v, f_i(1))$. Thus the jth bit of all the vertices in $V(P_2)$ is 0. Partition $V(P_1)$ into $V_{1,1}$ and $V_{1,2}$ such that

$$V_{1,1} = \{\text{all the vertices in } V(P_1) \text{ before the perfect matching } M_{(j+1) \bmod n} \text{ is applied}\} \text{ and } V_{1,2} = V(P_1) \setminus V_{1,1}.$$

Thus the jth bit of all vertices in $V_{1,1}$ is 1 and the jth bit of all vertices in $V_{1,2}$ is 0. By the fact that the jth bit of all the vertices in $V(P_2)$ is 0, to prove $V(P_1) \cap V(P_2) = \emptyset$, it suffices to prove $V_{1,2} \cap V(P_2) = \emptyset$. If $v_0 = 1$, then the $(j-1)$th bit of all the vertices in $V(P_2) \setminus f_j(v)$ is 1; however, the $(j-1)$th bit of all the vertices in $V_{1,2}$ is 0. Since the ith bit of is 1, but the ith bit of

all the vertices in $V_{1,2}$ is 0, $f_j(v) \notin V_{1,2}$. If $v_0 = 0$, then the $(j-1)$th bit of all the vertices in $V(P_2)$ is 1, and the $(j-1)$th bit of all the vertices in $V_{1,2} \setminus \{z = 2^{j-1} + 2^{i-1} + 1\}$ is 0. Since t is odd, there exists $v_k = 1$ for some $k > j$. Thus $z \notin V(P_2)$ by comparing the kth bit of them. Therefore, no such a exists in this case.
Subcase 4.3.2: $v_i = 0$ and $v_j = 0$. Suppose t is even. Then the jth bit of all the vertices in $V(P_2)$ is 1. However, the jth bit of all the vertices in $V(P_1)$ is 0. Suppose t is odd. Then the number of bits in $v_{n-1}v_{n-2}\ldots v_{i+1}$ that are 1 is even; this implies that i is the first member in $\Pi_i(v, f_i(1))$. Thus the ith bit of all the vertices in $V(P_2)$ is 0. However, the ith bit of all the vertices in $V(P_1)$ is 1. Thus no such a exists.

Subcase 4.3.3: $v_i = 0$ and $v_j = 1$. Suppose t is even. Then the first member in $\Pi_j(v, f_j(1))$ is $i-1$ and the first member in $\Pi_i(v, f_i(1))$ is i. So the ith bit of all the vertices in $V(P_2)$ is 0; however, the ith bit of all the vertices in $V(P_1)$ is 1. Suppose t is odd. Define q to be the index of the leftmost nonzero bit of v. Then $q > j$. Thus the $(i-1)$th bit of all the vertices in $V(P_2) \setminus \{f_j(v)\}$ is 0; however, the $(i-1)$th bit of all the vertices in $V(P_1)$ is 1. By comparing the jth and the qth bits of $f_j(v)$ with the jth and the qth bits of every vertex in $V(P_1)$, we have $f_j(v) \notin V(P_1)$. Thus no such a exists.

Subcase 4.3.4: $v_i = 1$ and $v_j = 0$. If the number of those bits from v_{n-1} to v_{j+1} being 1 is even, then the jth bit of all the vertices in $V(P_2)$ is 1; however the jth bit of all the vertices in $V(P_1)$ is 0. If the number of those bits from v_{n-1} to v_{j+1} being 1 is odd, then the number of bits in $v_{n-1}v_{n-2}\ldots v_{i+1}$ that are 1 is even. Thus i is the first member of $\Pi_j(v, f_j(1))$; but $i \notin \Pi_i(v, f_j(1))$, which implies that the ith bit of all the vertices in $V(P_2)$ is 0 but the ith bit of all the vertices in $V(P_1)$ is 1. So Claim 4.5 is true for this case.

As a result, Claim 4.5 is true for *Case* 4. From the above discussion, Claim 4.5 is true for all the cases, and therefore $T_0, T_1, \ldots, T_{n-1}$ are vertex-independent rooted at $r = 0$ of LTQ_n. □

By Theorem 2.4 and Lemmas 4.7 and 4.8, we have the following result.

Theorem 4.10. *$T_0, T_1, \ldots, T_{n-1}$ are n vertex-ISTs rooted at r for LTQ_n.*

5. Concluding remarks

The independent spanning trees (ISTs) problem attempts to construct a set of pairwise independent spanning trees and it has numerous applications in networks such as data broadcasting, scattering and reliable communication protocols. The well-known ISTs conjecture, Vertex/Edge Conjecture, states that any n-connected/n-edge-connected graph has n vertex-ISTs/edge-ISTs rooted at an arbitrary vertex r. Both the Vertex and Edge Conjectures are still open on general graphs for $n \geq 5$.

In this paper, we consider the ISTs problem on the n-dimensional locally twisted cube LTQ_n. The very recent algorithm proposed by Hsieh and Tu [12] is designed to construct n edge-ISTs rooted at vertex 0 for LTQ_n. However, we find that LTQ_n is not vertex-transitive when $n \geq 4$ and therefore Hsieh and Tu's result does not solve the Edge Conjecture for LTQ_n. In this paper, we present an algorithm to construct n vertex-independent spanning trees rooted at an arbitrary vertex for LTQ_n. To the best of our knowledge, this is the first result to confirm the Vertex Conjecture for the locally twisted cubes. In addition, it is also interesting to confirm whether the Vertex Conjecture is true for other hypercube variants.

References

[1] F. Bao, Y. Funyu, Y. Hamada, Y. Igarashi, Reliable broadcasting and secure distributing in channel networks, IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences E81A (1998) 796–806.
[2] F. Bao, Y. Igarashi, S.R. Ohring, Reliable broadcasting in product networks, Graph-theoretic Concepts in Computer Science 1517 (1998) 310–323.
[3] Y.S. Chen, C.Y. Chiang, C.Y. Chen, Multi-node broadcasting in all-ported 3-D wormhole-routed torus using an aggregation-then-distribution strategy, Journal of System Architecture 50 (2004) 575–589.
[4] G. Chartrand, L. Lensniak, Graph and Digraphs, Wadsworth, Monterey, CA, 1981.
[5] S. Curran, O. Lee, X. Yu, Finding four independent trees, SIAM Journal on Computing 35 (2006) 1023–1058.
[6] J. Cheriyan, S.N. Maheshwari, Finding nonseparating induced cycles and independent spanning trees in 3-connected graphs, Journal of Algorithms 9 (1988) 507–537.
[7] J. Edmonds, Edge-disjoint branchings, in: R. Rustin (Ed.), Combinatorial Algorithms, in: Courant Inst. Sci. Symp., vol. 9, Algorithmics Press, New York, 1973, pp. 91–96.
[8] Z. Ge, S.L. Hakimi, Disjoint rooted spanning trees with small depths in de Bruijn and Kautz graphs, SIAM Journal on Computing 26 (1997) 79–92.
[9] A. Huck, Independent trees in planar graphs, Graphs and Combinatorics 5 (1999) 29–77.
[10] A. Huck, Independent trees in graphs, Graphs and Combinatorics 10 (1994) 29–45.
[11] T. Hasunuma, H. Nagamochi, Independent spanning trees with small depths in iterated line digraphs, Discrete Applied Mathematics 110 (2001) 189–211.
[12] S.Y. Hsieh, C.J. Tu, Constructing edge-disjoint spanning trees in locally twisted cubes, Theoretical Computer Science 410 (2009) 8–10.
[13] K.S. Hu, S.S. Yeoh, C.Y. Chen, L.H. Hsu, Node-pancyclicity and edge-pancyclicity of hypercube variants, Information Processing Letters 102 (1) (2007) 1–7.
[14] Y. Iwasaki, Y. Kajiwara, K. Obokata, Y. Igarashi, Independent spanning trees of chordal rings, Information Processing Letters 69 (1999) 155–160.
[15] A. Itai, M. Rodeh, The multi-tree approach to reliability in distributed networks, Information and Computation 79 (1988) 43–59.
[16] S. Khuller, B. Schieber, On independent spanning-trees, Information Processing Letters 42 (1992) 321–323.
[17] K. Miura, D. Takahashi, S. Nakano, T. Nishizeki, A linear-time algorithm to find four independent spanning trees in four-connected planar graphs, International Journal of Foundations of Computer Science 10 (1999) 195–210.
[18] K. Miura, D. Takahashi, S. Nakano, T. Nishizeki, A linear-time algorithm to find four independent spanning trees in four-connected planar graphs, Discrete Applied Mathematics 83 (1998) 3–20.

[19] S. Nagai, S. Nakano, A linear-time algorithm to find independent spanning trees in maximal planar graphs, IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences E84A (2001) 1102–1109. Also appears in: Proceedings of 26th Workshop on Graph-Theoretic Concepts in Computer Science, WG 2000, in: LNCS 1928, Springer, 2000, pp. 290–301.
[20] K. Obokata, Y. Iwasaki, F. Bao, Y. Igarashi, Independent spanning trees of product graphs, Lecture Notes in Computer Science 197 (1996) 338–351. See also: K. Obokata, Y. Iwasaki, F. Bao, Y. Igarashi, Independent spanning trees of product graphs and their construction, in: IEICE Trans. Fundamentals of Electronics,Communications and Computer Sciences, E79-A, pp. 1894–1903, 1996.
[21] Y.C. Tseng, S.Y. Wang, C.W. Ho, Efficient broadcasting in wormhole-routed multicomputers: a network-partitioning approach, IEEE Transaction on Parallel and Distributed Systems 10 (1999) 44–61.
[22] S.M. Tang, Y.L. Wang, Y.H. Leu, Optimal independent spanning trees on hypercubes, Journal of Information Science and Engineering 20 (2004) 143–155.
[23] D.B. West, Introduction to Graph Theory, 2nd ed., Prentice Hall, Upper Saddle River, NJ, 2001.
[24] J.S. Yang, J.M. Chang, S.M. Tang, Y.L. Wang, Reducing the height of independent spanning trees in chordal rings, IEEE Transactions on Parallel and Distributed Systems 18 (2007) 644–657.
[25] X. Yang, D.J. Evans, G.M. Megson, The locally twisted cubes, International Journal of Computer Mathematics 82 (2005) 401–413.
[26] J.S. Yang, S.M. Tang, J.M. Chang, Y.L. Wang, Parallel construction of optimal independent spanning trees on hypercubes, Parallel Computing 33 (2007) 73–79.
[27] A. Zwhavi, A. Itai, Three tree-paths, Journal of Graph Theory 13 (1989) 175–188.