Noname manuscript No.

A parallel algorithm for constructing multiple independent spanning trees in bubble-sort networks

Shih-Shun Kao · Ralf Klasing · Ling-Ju Hung · Chia-Wei Lee · Sun-Yuan Hsieh

Received: date / Accepted: date

A preliminary version of this paper was published in *Proceedings of the 15th International Conference on Algorithmic Aspects in Information and Management* (*AAIM 2021*), *Lecture Notes in Computer Science 13153, pp. 252-264, 2021.*

Parts of the work were supported by the LaBRI under the "Projets émergents" program, the ANR project TEMPOGRAL (ANR-22-CE48-0001), and National Science and Technology Council of Taiwan under grants NSTC 111-2222-E-141-001. This study has been carried out in the frame of the "Investments for the future" Programme IdEx Bordeaux - SysNum (ANR-10-IDEX-03-02).

Shih-Shun Kao
CNRS, LaBRI, Université de Bordeaux, 351 Cours de la Libération, 33405 Talence, France
Department of Computer Science and Information Engineering, National Cheng Kung University, No. 1, University Road, Tainan, Taiwan
E-mail: shih-shun.kao@labri.fr

Ralf Klasing
CNRS, LaBRI, Université de Bordeaux, 351 Cours de la Libération, 33405 Talence, France
E-mail: ralf.klasing@labri.fr

Ling-Ju Hung
Department of Creative Technologies and Product Design, National Taipei University of Business, No.100, Sec. 1, Fulong Road, Taoyuan, Taiwan
E-mail: ljhung@ntub.edu.tw

Chia-Wei Lee
Department of Computer Science and Information Engineering, National Taitung University, No. 369, Sec. 2, University Road, Taitung, Taiwan
E-mail: cwlee@nttu.edu.tw

Sun-Yuan Hsieh
Department of Computer Science and Information Engineering, National Cheng Kung University, No. 1, University Road, Tainan, Taiwan
Department of Computer Science and Information Engineering, National Chi Nan University, No. 1, University Road, Nantou, Taiwan
Institute of Information Science, Academia Sinica, No. 128, Sec. 2, Academia Road, Taipei, 11529, Taiwan
E-mail: hsiehsy@mail.ncku.edu.tw

Abstract The use of multiple independent spanning trees (ISTs) for data broadcasting in networks provides a number of advantages, including the increase of fault-tolerance and secure message distribution. Thus, the designs of multiple ISTs on several classes of networks have been widely investigated. Kao *et al.* [*Journal of Combinatorial Optimization* 38 (2019) 972-986] proposed an algorithm to construct independent spanning trees in bubble-sort networks. The algorithm is executed in a recursive function and thus is hard to parallelize. In this paper, we focus on the problem of constructing ISTs in bubble-sort networks B_n and present a non-recursive algorithm. Our approach can be fully parallelized, i.e., every vertex can determine its parent in each spanning tree in constant time. This solves the open problem from the paper by Kao *et al.* Furthermore, we show that the total time complexity $\mathcal{O}(n \cdot n!)$ of our algorithm is asymptotically optimal, where n is the dimension of B_n and $n!$ is the number of vertices of the network.

Keywords Independent spanning trees · Bubble-sort networks · Interconnection networks.

1 Introduction

Fault-tolerant transmission and secure message distribution are the challenges in designing interconnected networks in a reliable communication network. Providing disjoint paths between each pair of vertices is the practical way to satisfy the above requirements through multi-path routing. Consequently, we can resume data transmission if the current transmission path is disconnected, allowing us to resume the data transmission via a disjoint backup path. Fault-tolerant communication is improved in [3,17]. A fault-free network can also be used to ensure message security through disjoint paths as shown in [3,34]. The source node is capable of dividing a message into various packets and transmitting them each via separate paths to the destination. Accordingly, each node in the network receives only one packet with the exception of the destination node that receives all the packets.

A network of interconnections is usually represented by a simple undirected graph $G = (V, E)$, where the vertex set $V(G)$ and the edge set $E(G)$ represent the set of processors and the set of communication links, respectively. A *spanning tree* T in G is a connected acyclic subgraph of G such that $V(T) = V(G)$. A pair of spanning trees rooted at the same vertex, say r, are called independent spanning trees (ISTs) if, for any vertex $v \in V(G) \setminus \{r\}$, the paths from v to r in any two trees share no common edge and no common vertex except for v and r. Multiple ISTs provide reliable communication in a network, and therefore, the provision of multiple ISTs is sufficient for providing a reliable network.

Over the past three decades, researchers have studied ISTs. According to Zehavi and Itai [45], there could be k ISTs rooted at any vertex of a k-connected graphs. Since that time, this conjecture has been verified only for k-connected graph with $k \leq 4$ (see [13,14,17]). Because k-connected graphs

for $k \geq 5$ have not yet been solved, the follow-up research has focused on constructing ISTs on specific networks, e.g., the construction of ISTs on recursive circulant graphs [38,39], generalized recursive circulant graphs [12], pancake networks [8], burnt pancake networks [44], (n, k)-star graphs [15], alternating group networks [16], transposition networks [23], torus networks [29], augmented cubes [10,31], line graph of the hypercube [9] and some variations of hypercubes [4,24,30,34,41]. In particular, special topics related to ISTs include the research on construction of ISTs in parallel [5–7,36,41,43] and reducing the height of the ISTs [35,37,40]. A survey on constructing ISTs in networks can be found in [11].

Since Cayley graphs have been used extensively to design the topologies of interconnection networks, construction of ISTs on Cayley graphs is significative. The bubble-sort network of dimension n is denoted by B_n. B_n is one of the most attractive subclasses of Cayley graphs generated by the transposition graph which is specified by an n-path as its transposition graph [1,18]. However, whether there exists a parallel algorithm to construct ISTs in Cayley graphs and bubble sort networks is still open in the literature [18]. Notice that in the case of $n \geq 4$, B_n is vertex-transitive but not edge-transitive [21]. The connectivity of B_n is $n-1$ and its diameter is $n(n-1)/2$ [1,28]. Analysis of hamiltonian laceability, pancyclicity, the container problem and node-to-node disjoint paths in B_n are obtained in [2,19,27,28,1], respectively. Researchers have also investigated fault tolerance, diagnosability, and reliability in bubble-sort networks in [20,32,42].

For the construction of ISTs on bubble-sort networks, Kao *et al.* [18] proposed an algorithm to construct $n-1$ ISTs of B_n and showed that the algorithm has optimal amortized efficiency for multiple trees construction. In particular, every vertex can determine its parent in each spanning tree in constant amortized time. The algorithm is executed in a recursive function and thus is hard to parallelize. In this paper, we present a parallel algorithm to construct $n-1$ ISTs in bubble-sort networks B_n. Our approach can be fully parallelized, i.e., every vertex can determine its parent in each spanning tree in constant time. This solves the open problem from [18]. Furthermore, we show that the total time complexity $\mathcal{O}(n \cdot n!)$ of our algorithm is asymptotically optimal, where n is the dimension of B_n and $n!$ is the number of vertices of the network.

The rest of this paper is organized as follows. In Section 2, we introduce the bubble-sort network and some notations. In Section 3, we introduce the algorithm for constructing independent spanning trees of B_n. In Section 4, we first show the correctness of our algorithm and give the complexity analysis. Then, we analyze the height of the ISTs. Finally, conclusions and future works are given in Section 5.

2 Preliminaries

Let Σ_n be the set of all permutations on $\{1, 2, \ldots, n\}$. For a permutation $p \in \Sigma_n$ and an integer $i \in \{1, 2, \ldots, n\}$, we use the following notations. The

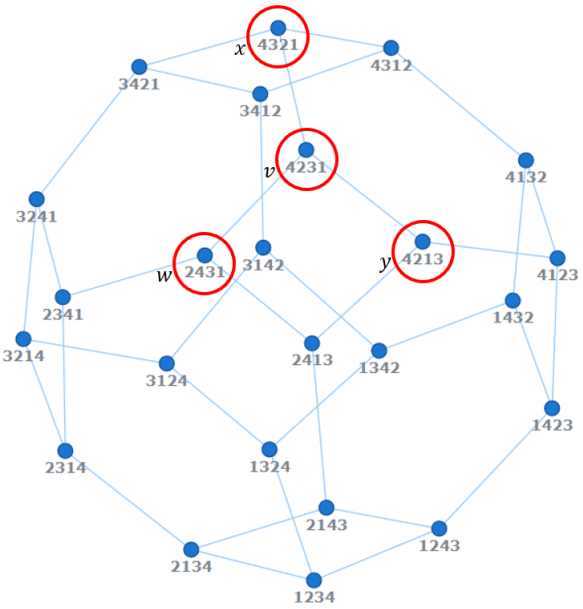

Fig. 1 The bubble-sort network B_4. The vertex $v = 4231$ has three adjacent vertices $w = 2431$, $x = 4321$ and $y = 4213$.

symbol at the ith position of p is denoted by p_i, and the position where the symbol i appears in p is denoted by $p^{-1}(i)$. For $i \in \{1, \ldots, n-1\}$, let $p\langle i \rangle = p_1 p_2 \cdots p_{i-1} p_{i+1} p_i p_{i+2} \cdots p_n$ be the permutation of Σ_n obtained from p by swapping two consecutive symbols at positions i and $i+1$. Hence, $p\langle p^{-1}(i) \rangle$ is the permutation obtained from p by swapping symbol i and its immediately succeeding symbol.

Definition 1 [1] The *bubble-sort network* of dimension n, denoted by B_n, is the undirected graph consisting of the vertex set $V(B_n) = \Sigma_n$ and the edge set $E(B_n) = \{(v, v\langle i \rangle) \colon v \in \Sigma_n, 1 \leqslant i \leqslant n-1\}$.

For example, Fig. 1 depicts B_4. A symbol i is said to be at the *right position* of p if $p_i = i$, and for $p \neq 12 \cdots n$ the position of the first symbol i from the right which is not in the right position is denoted by $r(p)$.

For any graph G, the *distance* from the vertex u to the vertex v in G, denoted by $d(u, v)$, is the length of the shortest path from u to v. The *diameter* of a graph G is $\max_{u,v \in V(G)} d(u, v)$ [33]. The diameter of a graph G is denoted by $D(G)$. B_n has diameter $D(B_n) = n(n-1)/2$ [1,28].

3 Constructing ISTs on B_n

In this section, we present an algorithm for constructing $n-1$ ISTs of B_n. Since B_n is vertex-transitive, without loss of generality, we may choose the

identity $\mathbf{1}_n = 12\cdots n$ as the common root r of all ISTs. Also, since B_n has connectivity $n-1$, the root in every spanning tree has a unique child. For $1 \leqslant t \leqslant n-1$, if the root of a spanning constructed by Algorithm 1 takes $\mathbf{1}_n\langle t\rangle$ as its unique child, then this spanning tree of B_n is denoted by T_t^n.

The case $n = 3$. This case has been solved in [18]. The two paths $T_1^3 = (123, 213, 231, 321, 312, 132)$ and $T_2^3 = (123, 132, 312, 321, 231, 213)$ are ISTs of B_3 that take $\mathbf{1}_3 = 123$ as the common root r. The solution is shown in Fig. 2.

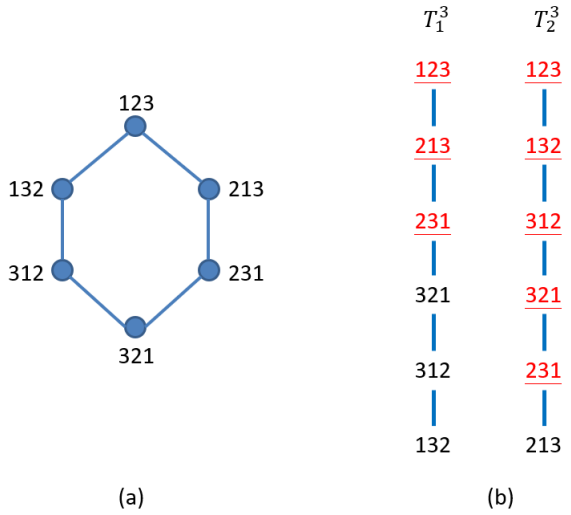

Fig. 2 (a) The bubble-sort network B_3. (b) A set of two ISTs of B_3. The paths in T_1^3 and T_2^3 from the vertex $v = 231$ to the root $r = 123$ are underlined.

The case $n \geqslant 4$. In general, for B_n with $n \geqslant 4$, the construction of the ISTs $T_1^n, \ldots, T_{n-1}^n$ of B_n can be accomplished by Algorithm 1 to determine the parent of each vertex (except the root) in every spanning tree.

The main idea of the algorithm is as follows. In T_t^n for $t \in \{1, 2, \ldots, n-2\}$ some paths are from the leaf vertex x with $x_n \in \{1, 2, \ldots, n-1\} \setminus \{t\}$ to the vertex y with $y_n = t$. Then, these paths are from the vertex y with $y_n = t$ to the root r. Moreover, there are some paths from the leaf vertex z with $z_n = n-1$ to the root r. Furthermore, there are some paths from the leaf vertex u with $u_n = n$ to the root r. In T_{n-1}^n some paths are from the leaf vertex v with $v_n = n$ to the vertex u with $u_n = n-1$. Then, these paths are from the vertex u with $u_n = n-1$ to the root r. Moreover, there are some paths from the leaf vertex v with $v_n \in \{1, 2, \ldots, n\} \setminus \{n-1\}$ to the vertex u with $u_n = n-1$. Then, these paths are from the vertex u with $u_n = n-1$ to the root r.

Algorithm 1: Parent1(v, t, n)

Input : v: the vertex $v = v_1 \dots v_n \in V(B_n) \setminus \{\mathbf{1}_n\}$
t: the t-th tree T_t^n in IST
n: the dimension of B_n
Output: p: $p =$ Parent1(v, t, n) the parent of v in T_t^n
if $v_n = n$ **then**
(1) **if** $t \neq n-1$ **then** $p =$ FindPosition(v)
(2) **else** $p =$ Swap(v, v_{n-1})
end
else
if $v_n = n-1$ and $v_{n-1} = n$ and Swap$(v, n) \neq \mathbf{1}_n$ **then**
(3) **if** $t = 1$ **then** $p =$ Swap(v, n)
(4) **else** $p =$ Swap$(v, t-1)$
end
else
(5) **if** $v_n = t$ **then** $p =$ Swap(v, n)
(6) **else** $p =$ Swap(v, t)
end
end
return p

Function FindPosition(v)

Input : v: the vertex $v = v_1 \cdots v_n$ in B_n
Output: p: the vertex adjacent to v in B_n
(1.1) **if** $t = 2$ and Swap$(v, t) = \mathbf{1}_n$ **then** $p =$ Swap$(v, t-1)$
(1.2) **else if** $v_{n-1} \in \{t, n-1\}$ **then** $j = r(v), p =$ Swap(v, j)
(1.3) **else** $p =$ Swap(v, t)
return p

Function Swap$(v,\ x)$

Input : v: the vertex $v = v_1 \cdots v_n$ in B_n
x: the symbol in the vertex $v_1 \cdots v_n$
Output: p: the vertex adjacent to v in B_n
$i = v^{-1}(x), p = v\langle i \rangle$
return p

Note that in a pre-processing stage, each vertex $v = v_1 v_2 \cdots v_n$ $(v \neq \mathbf{1}_n)$ computes its inverse permutation, i.e., $v^{-1}(1)v^{-1}(2) \cdots v^{-1}(n)$, and the position of the first symbol i from the right which is not in the right position, i.e., $r(v)$. For example, we consider $v = 4231$. The position of the first symbol from the right which is not in the right position is 4. Thus, $r(v) = 4$. The position where the symbol 4 appears in v is denoted by $v^{-1}(4)$. Thus, $v^{-1}(4) = 1$. This can be done efficiently in $\mathcal{O}(n)$ time for each vertex. Algorithm 1 uses two functions FindPosition(v) and Swap(v, x). The function FindPosition(v) has three conditions: (1) if $t = 2$ and Swap$(v, t) = \mathbf{1}_n$ then calls the Swap$(v, t-1)$ function; (2) if ($t \neq 2$ or Swap$(v, t) \neq \mathbf{1}_n$) and $v_{n-1} = t$ or $n-1$ then finds the rightmost symbol j in v which is not in the right position, and then calls the Swap(v, j) function; (3) if ($t \neq 2$ or Swap$(v, t) \neq \mathbf{1}_n$) and $v_{n-1} \neq t$ or $n-1$

Table 1 The parent of every vertex $v \in V(B_4) \setminus \{\mathbf{1}_4\}$ in T_t^4 for $t \in \{1,2,3\}$ calculated by Algorithm 1

v	t	v_4	Rule	p	v	t	v_4	Rule	p
1234	-	-	-	-	3124	1	4	(1.3)	3214
						2		(1.2)	1324
						3		(2)	3142
1243	1	3	(6)	2143	3142	1	2	(6)	3412
	2		(6)	1423		2		(5)	3124
	3		(5)	1234		3		(6)	1342
1324	1	4	(1.3)	3124	3214	1	4	(1.2)	2314
	2		(1.2)	1234		2		(1.3)	3124
	3		(2)	1342		3		(2)	3241
1342	1	2	(6)	3142	3241	1	1	(5)	3214
	2		(5)	1324		2		(6)	3421
	3		(6)	1432		3		(6)	2341
1423	1	3	(6)	4123	3412	1	2	(6)	3421
	2		(6)	1432		2		(5)	3142
	3		(5)	1243		3		(6)	4312
1432	1	2	(6)	4132	3421	1	1	(5)	3241
	2		(5)	1342		2		(6)	3412
	3		(6)	1423		3		(6)	4321
2134	1	4	(1.2)	1234	4123	1	3	(6)	4213
	2		(1.1)	2314		2		(6)	4132
	3		(2)	2143		3		(5)	1423
2143	1	3	(3)	2134	4132	1	2	(6)	4312
	2		(4)	2413		2		(5)	1432
	3		(4)	1243		3		(6)	4123
2314	1	4	(1.2)	2134	4213	1	3	(6)	4231
	2		(1.3)	3214		2		(6)	4123
	3		(2)	2341		3		(5)	2413
2341	1	1	(5)	2314	4231	1	1	(5)	2431
	2		(6)	3241		2		(6)	4321
	3		(6)	2431		3		(6)	4213
2413	1	3	(6)	2431	4312	1	2	(6)	4321
	2		(6)	4213		2		(5)	3412
	3		(5)	2143		3		(6)	4132
2431	1	1	(5)	2341	4321	1	1	(5)	3421
	2		(6)	4231		2		(6)	4312
	3		(6)	2413		3		(6)	4231

then calls the Swap(v,t) function. The function Swap(v,x) swaps the symbol x in v in its position i with the symbol in position $i+1$. Since we have the pre-processing stage, the two functions FindPosition(v) and Swap(v,x) can be calculated in constant time.

Table 1 shows the parent of every vertex $v \in V(B_4) \setminus \{\mathbf{1}_4\}$ in T_t^4 for $t \in \{1,2,3\}$ calculated by Algorithm 1. In this table, the column 'Rule' indicates which rule is used for computing the parent p. For example, we consider $v = 3214$ and $t = 3$. Since $v_4 = 4$, it follows from Rule (2) that $p = \text{Swap}(v, v_{4-1}) = 3241$. Also, we consider $v = 4231$ and $t = 1$. Since $v_4 = 1$, it follows from Rule (5) that $p = \text{Swap}(v, 4) = 2431$. In Fig. 3, the corresponding three ISTs rooted at vertex $\mathbf{1}_4$ for B_4 are shown.

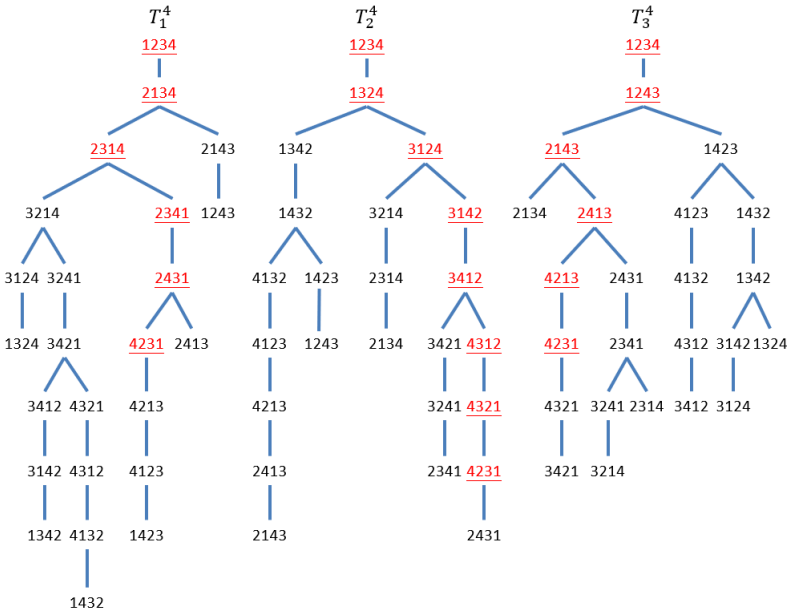

Fig. 3 The three ISTs of B_4 calculated by Algorithm 1. The paths in T_1^4 to T_3^4 from the vertex $v = 4231$ to the root $r = 1234$ are underlined.

4 Correctness and complexity analysis

In this section, we first show the complexity and the correctness of Algorithm 1. Then, we analyze the height of the ISTs constructed by Algorithm 1.

Theorem 1 *For $n \geqslant 4$, $T_1^n, T_2^n, \ldots, T_{n-1}^n$ constructed by Algorithm 1 are $n-1$ ISTs of B_n. Moreover, every vertex can determine its parent in each spanning tree in constant time. The total time complexity of Algorithm 1 is $\mathcal{O}(n \cdot n!)$ which is asymptotically optimal.*

Proof. We first show the complexity of Algorithm 1. There are $n-1$ ISTs, each IST contains $n!$ vertices, hence the lower bound on the time complexity is $\Omega(n \cdot n!)$. Obviously, in Algorithm 1 each vertex can determine its parent in each spanning tree in constant time. Hence, the total time complexity of Algorithm 1 is $\mathcal{O}(n \cdot n!)$ which is asymptotically optimal.

We now show the correctness of Algorithm 1. For a tree T and $u, u' \in T$, we use $T(u, u')$ to denote the unique path joining u and u' in T. Suppose that $n \geqslant 4$. Let $r = \mathbf{1}_n (= 12 \cdots n)$. The proof is by showing that for any vertex $v \in V(B_n) \setminus \{r\}$,

(a) for $n \geqslant 4$, $t \in \{1, \cdots, n-1\}$, Algorithm 1 constructs a unique path from the vertex $v = v_1 \ldots v_n \in V(B_n) \setminus \{\mathbf{1}_n\}$ to the root r of T_t^n, and thereby showing that for $n \geqslant 4$, every T_t^n is a spanning tree for $t \in \{1, \cdots, n-1\}$,

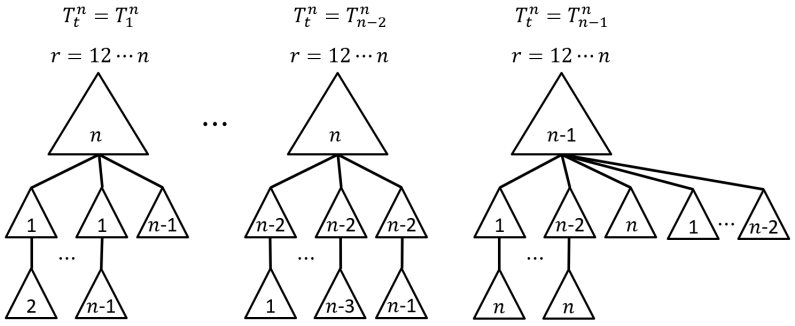

Fig. 4 The structure of the $n-1$ ISTs of B_n. For $t = 1, 2, \ldots, n-1$, in each of the subfigures a triangle with a label $k \in \{1, 2, \ldots, n\}$ inside corresponds to a subtree in T_t^n such that each vertex in the subtree has the last symbol k.

(b) the two paths from v to r in any two trees of $T_1^n, T_2^n, \ldots, T_{n-1}^n$ share no common edge and no common vertex except for v and r, and thereby proving the independence.

The structure of the $n-1$ ISTs of B_n is shown in Fig. 4. According to the last symbol of $v = v_1 \cdots v_n$, we consider the following three cases: Case A: $v_n = n$, Case B: $v_n = n-1$ and Case C: $v_n = j$ for $j \in \{1, 2, ..., n-2\}$. In Case A: $v_n = n$, we consider the following two subcases: Case A.1: $t \neq n-1$ and Case A.2: $t = n-1$. In Case A.1: $t \neq n-1$, we consider the following two subcases: Case A.1.1: $t \neq 2$ or $\text{Swap}(v, t) \neq \mathbf{1}_n$ and Case A.1.2: $t = 2$ and $\text{Swap}(v, t) = \mathbf{1}_n$. In Case A.1.1: $t \neq 2$ or $\text{Swap}(v, t) \neq \mathbf{1}_n$, we consider the following two subcases: Case A.1.1.1: $v_{n-1} \in \{t, n-1\}$ and Case A.1.1.2: $v_{n-1} \notin \{t, n-1\}$. In Case B: $v_n = n-1$, we consider the following two subcases: Case B.1: $v_{n-1} \neq n$ or $\text{Swap}(v, n) = \mathbf{1}_n$ and Case B.2: $v_{n-1} = n$ and $\text{Swap}(v, n) \neq \mathbf{1}_n$. In Case B.1: $v_{n-1} \neq n$ or $\text{Swap}(v, n) = \mathbf{1}_n$, we consider the following two subcases: Case B.1.1: $v_n = t$ and Case B.1.2: $v_n \neq t$. In Case B.2: $v_{n-1} = n$ and $\text{Swap}(v, n) \neq \mathbf{1}_n$, we consider the following two subcases: Case B.2.1: $t \neq 1$ and Case B.2.2: $t = 1$. In Case C: $v_n = j$ for $j \in \{1, 2, ..., n-2\}$, we consider the following two subcases: Case C.1: $v_n = t$ and Case C.2: $v_n \neq t$. The Rule (1.1) to Rule (6) are shown in Algorithm 1 and the two functions FindPosition(v) and Swap(v, x).

Case A: $v_n = n$.

Case A.1: $t \neq n-1$.

Case A.1.1: $t \neq 2$ or $\text{Swap}(v, t) \neq \mathbf{1}_n$.

Case A.1.1.1: $v_{n-1} \in \{t, n-1\}$.

By Rule (1.2), Algorithm 1 finds the rightmost symbol j in v which is not in the right position, and then calls the Swap(v, j) function. We can obtain the parent of a vertex by using Rule (1.2) repeatedly until all the symbols are in the right position. Thus, we obtain the path $T_t^n(v, r)$. Firstly, v has the symbol t at position $n-1$ and the symbol n at the right position. Then, for each k from $n-1$ down to 2, vertices along the path keep the symbol t at position k until a vertex with the symbol t at position $k-1$ and the symbol k at the right position is reached (e.g., consider the path $T_1^6(543216, \mathbf{1}_6) = (453\underline{21}6, 435\underline{21}6, 432\underline{51}6, 432\underline{15}6, 342\underline{15}6, 324\underline{15}6, 32\underline{14}56, 2\underline{31}456, 2\underline{13}456, \underline{12}3456)$, where the symbols with underscore are at positions $k-1$ and k. Hence, Algorithm 1 constructs a unique path from v to r. It also guarantees that $T_t^n(v, r)$ and $T_{t'}^n(v, r)$ are vertex-disjoint for $t \neq 2, t' \in \{1, 2, \ldots, n-2\}$ with $t \neq t'$. For an illustration of the paths $T_1^n(v, r)$, see Fig. 5(a).

Case A.1.1.2: $v_{n-1} \notin \{t, n-1\}$.

By Rule (1.3), the vertex p calculated is Swap(v, t). We can obtain the parent of a vertex by using Rule (1.3) repeatedly until a vertex z with $z_{n-1} = t$ and $z_n = n$ is reached. Thus, we obtain the path $T_t^n(v, z)$. The vertices along the path swap the symbol t to the position $n-1$. Note that every internal vertex in the path $T_t^n(v, z)$ takes n as the last symbol. For distinct paths, since exchanges of symbols start at different positions and are operated in a sequence, they can only share a common vertex v. From Case A.1.1.1, Algorithm 1 constructs a path $T_t^n(z, r)$ such that every internal vertex has the last symbol n. Hence, Algorithm 1 constructs a unique path from v to r. It also guarantees that $T_t^n(v, r)$ and $T_{t'}^n(v, r)$ are vertex-disjoint for $t \neq 2, t' \in \{1, 2, \ldots, n-2\}$ with $t \neq t'$. For an illustration of the paths $T_{n-2}^n(v, r)$, see Fig. 5(b).

Case A.1.2: $t = 2$ and Swap$(v, t) = \mathbf{1}_n$.

By Rule (1.1), the vertex p calculated is Swap$(v, t-1)$. We showed in Case A.1.1.2 that Algorithm 1 constructs a path $T_2^n(p, z) \cup T_2^n(z, r)$ such that every vertex in these paths takes n as the last symbol. Hence, Algorithm 1 constructs a unique path from v to r. It also guarantees that $T_2^n(v, r)$ and $T_{t'}^n(v, r)$ are vertex-disjoint for $t' \in \{1, 3, \ldots, n-1\}$.

Case A.2: $t = n-1$.

By Rule (2), the vertex p calculated is Swap(v, v_{n-1}). Thus, we obtain the path $T_t^n(v, p)$. Then, by Rule (5), the vertex p calculated is Swap(v, n). We can obtain the parent of a vertex by using Rule (5) repeatedly until the root r is reached. Thus, we obtain the path $T_t^n(p, r)$. Hence, Algorithm 1 constructs a path $T_{n-1}^n(p, r)$ such that $T_{n-1}^n(v, p) \cup T_{n-1}^n(p, r)$ contains internal vertices with the last symbol k for $k \in \{1, 2, \ldots, n-1\}$. The vertices along the path swap the symbol from $n-1$ to 1 to the right position until the vertex to the root r.

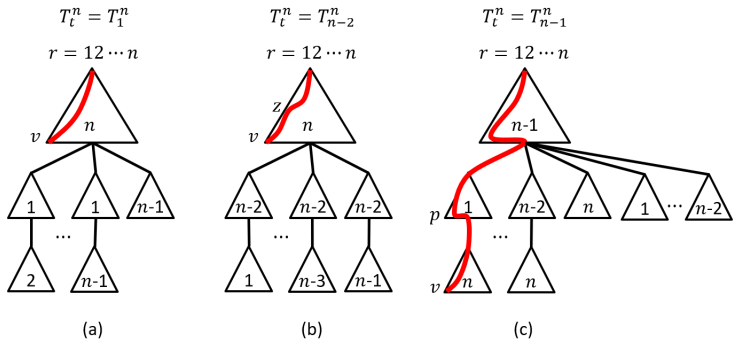

Fig. 5 An illustration of the paths described in the proof of Case A of Theorem 1. (a) The red(bold) lines indicate the paths $T_1^n(v,r)$ described in the proof of Case A.1.1.1 of Theorem 1. (b) The red(bold) lines indicate the paths $T_{n-2}^n(v,r)$ described in the proof of Case A.1.1.2 of Theorem 1. (c) The red(bold) lines indicate the paths $T_{n-1}^n(v,r)$ described in the proof of Case A.2 of Theorem 1.

Hence, Algorithm 1 constructs a unique path from v to r. Since every vertex in the path $T_{t'}^n(v,r)$ for $t' \in \{1,2,\ldots,n-2\}$ has the last symbol n, this guarantees that $T_{n-1}^n(v,r)$ and $T_{t'}^n(v,r)$ are vertex-disjoint for $t' \in \{1,2,\ldots,n-2\}$. For an illustration of the paths $T_{n-1}^n(v,r)$, see Fig. 5(c).

Case B: $v_n = n-1$.

Case B.1: $v_{n-1} \neq n$ or $\mathrm{Swap}(v,n) = \mathbf{1}_n$.

Case B.1.1: $v_n = t$.

By Rule (5), the vertex p calculated is $\mathrm{Swap}(v,n)$. We can obtain the parent of a vertex by using Rule (5) repeatedly until a vertex x with $x_{n-1} = t$ and $x_n = n$ is reached. Thus, we obtain the path $T_t^n(v,x)$. Note that every internal vertex in the path $T_t^n(v,x)$ takes t as the last symbol. For distinct paths, since exchanges of symbols start at different positions and are operated in a sequence, they can only share a common vertex v. From Case A.1.1.1, Algorithm 1 constructs a path $T_t^n(x,r)$ such that every vertex in these paths takes n as the last symbol. Hence, Algorithm 1 constructs a unique path from v to r. This also shows that $T_t^n(v,r)$ and $T_{t'}^n(v,r)$ for $t,t' \in \{1,2,\ldots,n-1\}$ are vertex-disjoint with $t \neq t'$. For an illustration of the paths $T_1^n(v,r)$, see Fig. 6(a).

Case B.1.2: $v_n \neq t$.

By Rule (6), the vertex p calculated is $\mathrm{Swap}(v,t)$. We can obtain the parent of a vertex by using Rule (6) repeatedly until a vertex u with $u_{n-1} = j$ and $u_n = t$ is reached. Thus, we obtain the path $T_t^n(v,u)$. Note that every

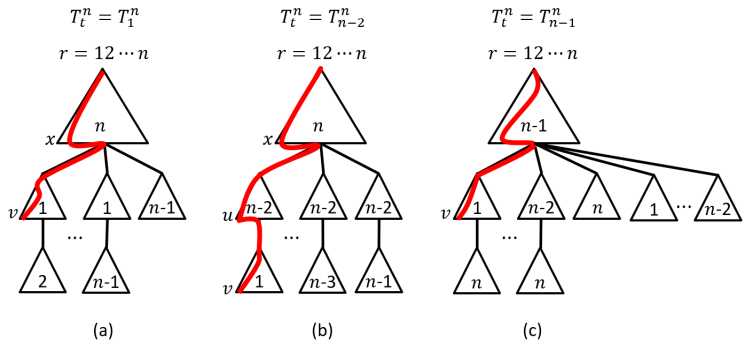

Fig. 6 An illustration of the paths described in the proof of Case B.1 and Case C of Theorem 1. (a) The red(bold) lines indicate the paths $T_1^n(v,r)$ described in the proof of Case B.1.1 and C.1 of Theorem 1. (b) The red(bold) lines indicate the paths $T_{n-2}^n(v,r)$ described in the proof of Case B.1.2 and C.2 of Theorem 1. (c) The red(bold) lines indicate the paths $T_{n-1}^n(v,r)$ described in the proof of Case B.1.2 and C.2 of Theorem 1.

internal vertex in the path $T_t^n(v,u)$ takes j as the last symbol. For distinct paths, since exchanges of symbols start at different positions and are operated in a sequence, they can only share a common vertex v. If $t \neq n-1$, from Case B.1.1, Algorithm 1 constructs a path $T_t^n(u,r)$ such that every internal vertex has the last symbol t or n. For an illustration of the paths $T_{n-2}^n(v,r)$, see Fig. 6(b). If $t = n-1$, Algorithm 1 constructs a path $T_{n-1}^n(v,r)$ such that every internal vertex in these paths takes $n-1$ as the last symbol. Hence, Algorithm 1 constructs a unique path from v to r. This also shows that $T_t^n(v,r)$ and $T_{t'}^n(v,r)$ for $t,t' \in \{1,2,\ldots,n-1\}$ are vertex-disjoint with $t \neq t'$. For an illustration of the paths $T_{n-1}^n(v,r)$, see Fig. 6(c).

Case B.2: $v_{n-1} = n$ and $\mathrm{Swap}(v,n) \neq \mathbf{1}_n$.

Case B.2.1: $t \neq 1$.

By Rule (4), the vertex p calculated is $\mathrm{Swap}(v,t-1)$. We can obtain the parent of a vertex by using Rule (4) repeatedly until a vertex u with $u_{n-1} \neq n$ is reached. Thus, we obtain the path $T_t^n(v,u)$. Note that every internal vertex in the path $T_t^n(v,u)$ takes $n-1$ as the last symbol. For distinct paths, since exchanges of symbols start at different positions and are operated in a sequence, they can only share a common vertex v. If $t \neq n-1$, we showed in Case B.1.2 that Algorithm 1 constructs a path $T_t^n(u,r)$ such that every vertex in these paths takes t or n as the last symbol. For an illustration of the paths $T_{n-2}^n(v,r)$, see Fig. 7(b). If $t = n-1$, Algorithm 1 constructs a path $T_{n-1}^n(v,r)$ such that every vertex in these paths takes $n-1$ as the last symbol. Hence, Algorithm 1 constructs a unique path from v to r. This also shows that $T_t^n(v,r)$ and $T_{t'}^n(v,r)$ for $t \neq 1, t' \in \{1,\ldots,n-1\}$ are vertex-disjoint with

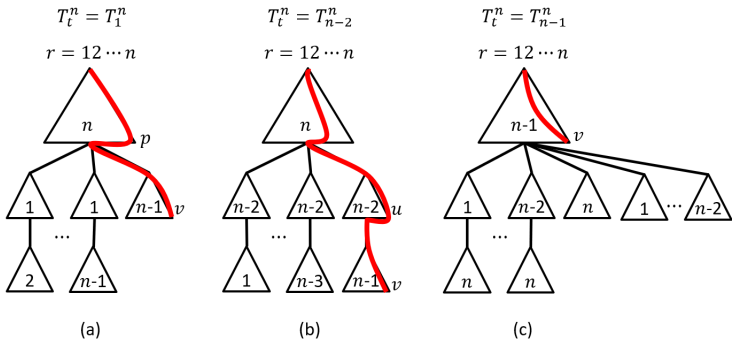

Fig. 7 An illustration of the paths described in the proof of Case B.2 of Theorem 1. (a) The red(bold) lines indicate the paths $T_1^n(v,r)$ described in the proof of Case B.2.2 of Theorem 1. (b) The red(bold) lines indicate the paths $T_{n-2}^n(v,r)$ described in the proof of Case B.2.1 of Theorem 1. (c) The red(bold) lines indicate the paths $T_{n-1}^n(v,r)$ described in the proof of Case B.2.1 of Theorem 1.

$t \neq t'$. For an illustration of the paths $T_{n-1}^n(v,r)$, see Fig. 7(c).

Case B.2.2: $t = 1$.

By Rule (3), the vertex $p = p_1 \dots p_n$ calculated is Swap(v,n). Thus, $p_n = v_{n-1} = n$. From Case A.1.1.1, Algorithm 1 constructs a path $T_1^n(p,r)$ such that every internal vertex has the last symbol n. We showed in Case B.2.1 that Algorithm 1 constructs a path $T_t^n(v,r)$ for $t \in \{2,\dots,n-1\}$ such that every vertex in these paths takes $n-1$, t or n as the last symbol. Hence, Algorithm 1 constructs a unique path from v to r. This also shows that $T_1^n(v,r)$ and $T_{t'}^n(v,r)$ for $t' \in \{2,\dots,n-1\}$ are vertex-disjoint. For an illustration of the paths $T_1^n(v,r)$, see Fig. 7(a).

Case C: $v_n = j$ for $j \in \{1,2,...,n-2\}$.

Case C.1: $v_n = t$.

By Rule (5), the vertex p calculated is Swap(v,n). We can obtain the parent of a vertex by using Rule (5) repeatedly until a vertex x with $x_{n-1} = t$ and $x_n = n$ is reached. Thus, we obtain the path $T_t^n(v,x)$. Note that every internal vertex in the path $T_t^n(v,x)$ takes t as the last symbol. For distinct paths, since exchanges of symbols start at different positions and are operated in a sequence, they can only share a common vertex v. From Case A.1.1.1, Algorithm 1 constructs a path $T_t^n(x,r)$ such that every vertex in these paths takes n as the last symbol. Hence, Algorithm 1 constructs a unique path from v to r. This also shows that $T_t^n(v,r)$ and $T_{t'}^n(v,r)$ for $t,t' \in \{1,2,\dots,n-1\}$ are vertex-disjoint with $t \neq t'$. For an illustration of the paths $T_1^n(v,r)$, see

Fig. 6(a).

Case C.2: $v_n \neq t$.

By Rule (6), the vertex p calculated is Swap(v, t). We can obtain the parent of a vertex by using Rule (6) repeatedly until a vertex u with $u_{n-1} = j$ and $u_n = t$ is reached. Thus, we obtain the path $T_t^n(v, u)$. Note that every internal vertex in the path $T_t^n(v, u)$ takes j as the last symbol. For distinct paths, since exchanges of symbols start at different positions and are operated in a sequence, they can only share a common vertex v. If $t \neq n-1$, from Case B.2.1, Algorithm 1 constructs a path $T_t^n(u, r)$ such that every internal vertex has the last symbol t or n. For an illustration of the paths $T_{n-2}^n(v, r)$, see Fig. 6(b). If $t = n-1$, Algorithm 1 constructs a path $T_{n-1}^n(v, r)$ such that every internal vertex in these paths takes $n-1$ as the last symbol. Hence, Algorithm 1 constructs a unique path from v to r. This also shows that $T_t^n(v, r)$ and $T_{t'}^n(v, r)$ for $t, t' \in \{1, 2, \ldots, n-1\}$ are vertex-disjoint with $t \neq t'$. For an illustration of the paths $T_{n-1}^n(v, r)$, see Fig. 6(c).

This completes the proof of Theorem 1. □

We now analyze the height of the ISTs constructed by Algorithm 1. The *height* of a rooted tree T, denoted by $h(T)$, is the number of edges from the root to a farthest leaf. We define $H_n = \max_{1 \leqslant t \leqslant n-1} h(T_t^n)$ to analyze the height of our constructed ISTs for B_n.

Theorem 2 *For the bubble-sort network B_n, $n \geq 3$, Algorithm 1 constructs $n-1$ ISTs of B_n with height at most $D(B_n) + n - 1$.*

Proof. From Algorithm 1, the path from the vertex v with $v_n = 2$ to the vertex u with $u_n = 1$ has at most $n-1$ edges, and the path from the vertex u with $u_n = 1$ to the vertex x with $x_n = n$ has at most $n-1$ edges. Moreover, the path from the vertex w with $w_n = n$ to the vertex x with $x_n = n$ and $x_{n-1} = t$ has at most $n-1$ edges, and the path from the vertex x with $x_n = n$ and $x_{n-1} = t$ to the vertex y with $y_n = n$ and $y_{n-1} = n-1$ has at most $n-2$ edges, and the path from the vertex y with $y_n = n$ and $y_{n-1} = n-1$ to the vertex z with $z_n = n$, $z_{n-1} = n-1$ and $z_{n-2} = n-2$ has at most $n-3$ edges. Since $(n-2) + (n-3) + \cdots + 1 = (n-1)(n-2)/2$, the path from the vertex x with $x_n = n$ and $x_{n-1} = t$ to the root r has at most $(n-1)(n-2)/2$ edges. Hence, the path from the vertex v with $v_n = 2$ to the root r has at most $(n-1)(n-2)/2 + (n-1) + (n-1) = n(n+1)/2 - 1$ edges.

Therefore, $H_n \leq n(n+1)/2 - 1$. As the diameter of B_n is $n(n-1)/2$ [1, 28], the height of the constructed ISTs in B_n is at most $D(B_n) + n - 1$.

This completes the proof of Theorem 2. □

5 Conclusion

In this paper, we have proposed an algorithm for constructing $n-1$ ISTs rooted at an arbitrary vertex of the bubble-sort network B_n. Our approach can be fully parallelized, i.e., every vertex can determine its parent in each spanning tree in constant time. Furthermore, we show that the total time complexity $\mathcal{O}(n \cdot n!)$ of our algorithm is asymptotically optimal, where n is the dimension of B_n and $n!$ is the number of vertices of the network.

Since B_n is a regular graph with connectivity $n-1$, the number of constructed ISTs is the maximum possible. For future work, a problem remaining open from our work is whether our algorithm can be extended to the (n,k)-bubble-sort graph [26,46,47] which is a generalization of bubble-sort networks. Moreover, the butterfly graph [22,25] has good structural symmetries, is regular of degree 4, and the recursive construction properties are similar to bubble-sort networks. Thus, it is of interest to study the construction of ISTs on butterfly graphs.

References

1. S. B. Akers and B. Krishnamurty, "A group theoretic model for symmetric interconnection networks," *IEEE Transactions on Computers*, vol. 38, no. 4, pp. 555–566, 1989.
2. T. Araki and Y. Kikuchi, "Hamiltonian laceability of bubble-sort graphs with edge faults", *Information Sciences*, vol. 177, no. 13, pp, 2679–2691, 2007.
3. F. Bao, Y. Funyu, Y. Hamada, and Y. Igarashi, "Reliable broadcasting and secure distributing in channel networks," in: *Proc. of 3rd International Symposium on Parallel Architectures, Algorithms and Networks*, ISPAN'97, Taipei, December 1997, pp. 472–478.
4. J.-M. Chang, J.-D. Wang, J.-S. Yang, and K.-J. Pai, "A comment on independent spanning trees in crossed cubes," *Information Processing Letters*, vol. 114, no. 12, pp. 734–739, 2014.
5. J.-M. Chang, T.-J. Yang, and J.-S. Yang, "A parallel algorithm for constructing independent spanning trees in twisted cubes," *Discrete Applied Mathematics*, vol. 219, pp. 74–82, 2017.
6. Y.-H. Chang, J.-S. Yang, J.-M. Chang, and Y.-L. Wang, "A fast parallel algorithm for constructing independent spanning trees on parity cubes," *Applied Mathematics and Computation*, vol. 268, pp. 489–495, 2015.
7. Y.-H. Chang, J.-S. Yang, S.-Y. Hsieh, J.-M. Chang, and Y.-L. Wang, "Construction independent spanning trees on locally twisted cubes in parallel," *Journal of Combinatorial Optimization*, vol. 33, no. 3, pp. 956–967, 2017.
8. D.-W. Cheng, C.-T. Chan, and S.-Y. Hsieh, "Constructing independent spanning trees on pancake networks," *IEEE Access*, vol. 8, pp. 3427–3433, 2020.
9. B. Cheng, J. Fan, C.-K. Lin, X. Jia, and X. Li, "Constructing node-independent spanning trees on the line graph of the hypercube by an independent forest scheme," *Journal of Parallel and Distributed Computing*, vol. 134, pp. 104-115, 2019.
10. B. Cheng, J. Fan, Q. Lyu, C.-K. Lin, X. Li, and G. Chen, "Constructing node-independent spanning trees in augmented cubes," *Fundamenta Informaticae*, vol. 176, pp. 103-128, 2020.
11. B. Cheng, D. Wang and J. Fan, "Independent spanning trees in networks - A survey," *ACM Computing Surveys*, accepted. (Published: 04 April 2023) DOI: https://doi.org/10.1145/3591110
12. D.-W. Cheng, K.-H. Yao, and S.-Y. Hsieh, "Constructing independent spanning trees on generalized recursive circulant graphs," *IEEE Access*, vol. 9, pp. 74028–74037, 2021.

13. J. Cheriyan and S.N. Maheshwari, "Finding nonseparating induced cycles and independent spanning trees in 3-connected graphs," *Journal of Algorithms*, vol. 9, no. 4, pp. 507–537, 1988.
14. S. Curran, O. Lee, and X. Yu, "Finding four independent trees," *SIAM Journal on Computing*, vol. 35, no. 5, pp. 1023–1058, 2006.
15. J.-F. Huang, E. Cheng, and S.-Y. Hsieh, "Two algorithms for constructing independent spanning trees in (n, k)-star graphs," *IEEE Access*, vol. 8, pp. 175932–175947, 2020.
16. J.-F. Huang, S.-S. Kao, S.-Y. Hsieh, and R. Klasing, "Top-down construction of independent spanning trees in alternating group networks," *IEEE Access*, vol. 8, pp. 112333–112347, 2020.
17. A. Itai and M. Rodeh, "The multi-tree approach to reliability in distributed networks," *Information and Computation*, vol. 79, no. 1, pp. 43–59, 1988.
18. S.-S. Kao, K.-J. Pai, S.-Y. Hsieh, R.-Y. Wu, and J.-M. Chang, "Amortized efficiency of constructing multiple independent spanning trees on bubble-sort networks," *Journal of Combinatorial Optimization*, vol. 38, no. 3, pp. 972–986, 2019.
19. Y. Kikuchi and T. Araki, "Edge-bipancyclicity and edge-fault-tolerant bipancyclicity of bubble-sort graphs", *Information Processing Letter*, vol. 100, no. 2, pp. 52–59, 2006.
20. T.-L. Kung and C.-N. Hung, "Estimating the subsystem reliability of bubblesort networks", *Theoretical Computer Science*, vol. 670, pp. 45–55, 2017.
21. S. Lakshmivarahan, J. Jwo, and S. K. Dhall, "Symmetry in interconnection networks based on Cayley graphs of permutation groups: A survey," *Parallel Computing*, vol. 19, no. 4, pp. 361–407, 1993.
22. F.T. Leighton, "Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes," *Morgan Kaufmann Publishers*, 1992.
23. C.-F. Lin, J.-F. Huang, and S.-Y. Hsieh, "Constructing independent spanning trees on transposition networks," *IEEE Access*, vol. 8, pp. 147122–147132, 2020.
24. J.-C. Lin, J.-S. Yang, C.-C. Hsu, and J.-M. Chang, "Independent spanning trees vs. edge-disjoint spanning trees in locally twisted cubes," *Information Processing Letters*, vol. 110, no. 10, pp. 414–419, 2010.
25. L.-H. Liu, J.-E. Chen, S.-Q. Chen, and W.-J. Jia, "An new representation for interconnection network structures," *Journal of Central South University of Technology*, vol. 9, no. 1, pp. 47–53, 2002.
26. N. Shawash, *"Relationships among popular interconnection networks and their common generalization,"* Ph.D. thesis, Oakland University, 2008.
27. Y. Suzuki and K. Kaneko, "An algorithm for disjoint paths in bubble-sort graphs", *Systems and Computers in Japan*, vol. 37, no. 27–32, 2006.
28. Y. Suzuki and K. Kaneko, "The container problem in bubble-sort graphs," *IEICE Transactions on Information and Systems*, vol. 91, no. 4, pp. 1003–1009, 2008.
29. S.-M. Tang, J.-S. Yang, Y.-L. Wang, and J.-M. Chang, "Independent spanning trees on multidimensional torus networks," *IEEE Transactions on Computers*, vol. 59, no. 1, pp. 93–102, 2010.
30. Y. Wang, J. Fan, G. Zhou, and X. Jia, "Independent spanning trees on twisted cubes," *Journal of Parallel and Distributed Computing* vol. 72, no. 1, pp. 58–69, 2012.
31. Y. Wang, H. Shen, and J. Fan, "Edge-independent spanning trees in augmented cubes," *Theoretical Computer Science* vol. 670, pp. 23–32, 2017.
32. M. Wang, Y. Lin, and S. Wang, "The 2-good-neighbor diagnosability of Cayley graphs generated by transposition trees under the PMC model and MM* model", *Theoretical Computer Science*, vol. 628, pp. 92–100, 2016.
33. D. B. West, *"Introduction to graph theory,"* 2nd Edition, Prentice Hall, 2001.
34. J.-S. Yang, H.-C. Chan, and J.-M. Chang, "Broadcasting secure messages via optimal independent spanning trees in folded hypercubes," *Discrete Applied Mathematics*, vol. 159, no. 12, pp. 1254–1263, 2011.
35. J.-S. Yang and J.-M. Chang, "Optimal independent spanning trees on Cartesian product of hybrid graphs," *Computer Journal*, vol. 57, no. 1, pp. 93–99, 2014.
36. J.-S. Yang, J.-M. Chang, K.-J. Pai, and H.-C. Chan, "Parallel construction of independent spanning trees on enhanced hypercubes," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 11, pp. 3090–3098, 2015.

37. J.-S. Yang, J.-M. Chang, S.-M. Tang, and Y.-L. Wang, "Reducing the height of independent spanning trees in chordal rings," *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, no. 5, pp. 644–657, 2007.
38. J.-S. Yang, J.-M. Chang, S.-M. Tang, and Y.-L. Wang, "On the independent spanning trees of recursive circulant graphs $G(cd^m, d)$ with $d > 2$," *Theoretical Computer Science*, vol. 410, no. 21-23, pp. 2001–2010, 2009.
39. J.-S. Yang, J.-M. Chang, S.-M. Tang, and Y.-L. Wang, "Constructing multiple independent spanning trees on recursive circulant graphs $G(2^m, 2)$," *International Journal of Foundations of Computer Science*, vol. 21, no. 1, pp. 73–90, 2010.
40. J.-S. Yang, S.-S. Luo, and J.-M. Chang, "Pruning longer branches of independent spanning trees on folded hyper-stars," *Computer Journal*, vol. 58, no. 11, pp. 2972–2981, 2015.
41. J.-S. Yang, S.-M. Tang, J.-M. Chang, and Y.-L. Wang, "Parallel construction of optimal independent spanning trees on hypercubes," *Parallel Computing*, vol. 33, no. 1, pp. 73–79, 2007.
42. Y. Yang and S. Wang, J. Li, "Subnetwork preclusion for bubble-sort graph networks", *Information Processing Letters*, vol. 115, no. 11, pp. 817–821, 2015.
43. J.-S. Yang, M.-R. Wu, J.-M. Chang, and Y.-H. Chang, "A fully parallelized scheme of constructing independent spanning trees on Möbius cubes," *Journal of Supercomputing*, vol. 71, no. 1, pp. 894–908, 2015.
44. Y.-C. Yang, S.-S. Kao, R. Klasing, S.-Y. Hsieh, H.-H. Chou, and J.-M. Chang, "The construction of multiple independent spanning trees on burnt pancake networks," *IEEE Access*, vol. 9, pp. 16679–16691, 2021.
45. A. Zehavi and A. Itai, "Three tree-paths," *Journal of Graph Theory*, vol. 13, no. 2, pp. 175–188, 1989.
46. S.-L. Zhao and R.-X Hao, "The generalized connectivity of (n, k)-bubble-sort graphs," *The Computer Journal*, vol. 62, no. 9, pp. 1277–1283, 2019.
47. S.-L. Zhao and R.-X Hao, "The fault tolerance of (n, k)-bubble-sort networks," *Discrete Applied Mathematics*, vol. 285, pp. 204–211, 2020.

.....

Shih-Shun Kao received the B.S. degree from the Institute of Information and Decision Sciences, National Taipei University of Business, Taiwan, in 2018. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Information Engineering, National Cheng Kung University, Taiwan, and the CNRS, LaBRI, University of Bordeaux, France. His research interests include graph theory, interconnection networks, and algorithms.

.....

Ralf Klasing received the PhD degree from the University of Paderborn in 1995. From 1995 to 1997, he was an Assistant Professor at the University of Kiel. From 1997 to 1998, he was a Research Fellow at the University of Warwick. From 1998 to 2000, he was an Assistant Professor at RWTH Aachen. From 2000 to 2002, he was a Lecturer at King's College London. In 2002, he joined the CNRS as a permanent researcher. From 2002 to 2005, he was affiliated to the laboratory I3S in Sophia Antipolis. Currently, he is affiliated to the laboratory LaBRI in Bordeaux. In 2009, he received the HDR degree from the University Bordeaux 1. In 2010, he was promoted to Senior Researcher (DR CNRS). From 2010 to 2015, he was the Head of the Combinatorics and Algorithms team of the LaBRI.

His research interests are in "Design and Analysis of Algorithms" and in "Complexity". More particularly, his research focuses on (1) *Distributed algorithms*, (2) *Approximation algorithms for combinatorially hard problems*, (3) *Algorithmic methods for telecommunication*, (4) *Communication algorithms in networks*. He has co-authored three book chapters, two Springer Monographs, and has published 55 papers in refereed international journals and 55 papers in refereed international conferences with proceedings. He has published 6 invited papers in international conferences. He has given 14 invited talks at international conferences, and 1 invited lecture series. He has edited 5 LNCS volumes, and 3 special issues of the international journals *ACM Journal of Experimental Algorithmics*, *Journal of Computer and System Sciences*, and *Journal of Information Science and Engineering*.

He is Managing Editor of the 2 international journals *Algorithmica* and *Journal of Interconnection Networks*. He is a member of the Editorial Board of the 11 international journals *Algorithms*, *Computing and Informatics*, *Discrete Applied Mathematics*, *Fundamenta Informaticae*, *International Journal of Computer Mathematics: Computer Systems Theory*, *Journal of Computer and System Sciences*, *Journal of Parallel and Distributed Computing*, *Networks*, *Parallel Processing Letters*, *Theoretical Computer Science*, and *Wireless Networks*. He is a member of the Steering Committee of IWOCA. He was a member of the Steering Committee of SIROCCO. He was the Conference Co-Chair of I-SPAN 2018, and the Chair of the Program Commmittees of BWIC 2011, SEA 2012, and ALGOSENSORS 2016 (Track on *Distributed and Mobile Networks*). He acted as the Co-Chair of the Program Commmittees of SSS 2013 (Track on *Ad-hoc*, *Sensors*, *Mobile Agents and Robot Networks*), FCT 2017, I-SPAN 2017, IWOCA 2020, ALGOSENSORS 2021, and as the Chair of the best paper award committee of CIAC 2017. He has acted as a member of the Program Committees of 51 well-acknowledged international conferences, including ICALP, STACS, MFCS, FCT, ISAAC, IWOCA, WAOA, COCOON, OPODIS, SIROCCO and SOFSEM.

.....

.....

Ling-Ju Hung received the bachelor's degree in applied mathematics from National Chung Hsing University, Taiwan, in 2001 and the MS and PhD degrees in computer science and information engineering from National Chung Cheng University, Taiwan, in 2003 and 2012, respectively. She became a postdoctoral research fellow in the Department of Computer Science and Information Engineering, Hung Kuang University, Taiwan during 2012–2015 and in the Department of Computer Science and Information Engineering, National Cheng Kung University, Taiwan during 2015-2018. In 2018, Dr. Hung applied her algorithm background on solving industrial issues. She joined AROBOT as a senior manager to lead the Department of Algorithms. She led the algorithm team in AROBOT to build a speech recognition engine and won an excellent industrial system award in Formosa Speech Recognition Challenge in 2018. She is currently an Assistant Professor in the Department of Creative Technologies and Product Design, National Taipei University of Business, Taiwan. She has served as a guest editor of international journals including Algorithmica, Theoretical Computer Science, and Journal of Combinatorial Optimization. Her research interests include the design and analysis of fixed-parameter algorithms, exact algorithms, approximation algorithms, and graph theory.

.....

Sun-Yuan Hsieh received the PhD degree in computer science from National Taiwan University, Taipei, Taiwan, in June 1998. He then served the compulsory two-year military service. From August 2000 to January 2002, he was an Assistant Professor at the Department of Computer Science and Information Engineering, National Chi Nan University. In February 2002, he joined the Department of Computer Science and Information Engineering, National Cheng Kung University, and now he is a Chair Professor. He received the 2007 K. T. Lee Research Award, President's Citation Award (American Biographical Institute) in 2007, the Engineering Professor Award of Chinese Institute of Engineers (Kaohsiung Branch) in 2008, National Science Council’s Outstanding Research Award in 2009, IEEE Outstanding Technical Achievement Award (IEEE Tainan Section) in 2011, Outstanding Electronic Engineering Professor Award of Chinese Institute of Electrical Engineers in 2013, and Outstanding Engineering Professor Award of Chinese Institute of Engineers in 2014. He is Fellow of the British Computer Society (BCS) and Fellow of Institution of Engineering and Technology (IET).

Dr. Hsieh is also an experienced editor with editorial services to a number of journals, including serving as associate editors of IEEE ACCESS, IEEE Transactions on Reliability, Theoretical Computer Science (Elsevier), Discrete Applied Mathematics (Elsevier), Journal of Supercomputing (Springer), International Journal of Computer Mathematics (Taylor & Francis Group), Parallel Processing Letters (World Scientific), Discrete Mathematics, Algorithms and Applications (World Scientific), Fundamental Informaticae (Polish Mathematical Society), and Journal of Interconnection Networks (World Scientific). In addition, he has served on organization committee and/or program committee of several dozens international conferences in computer science and computer engineering. His current research interests include design and analysis of algorithms, fault-tolerant computing, bioinformatics, parallel and distributed computing, and algorithmic graph theory.

.....

.....

Chia-Wei Lee received the BS and MS degrees in the Department of Computer Science and Information Engineering from National Chi Nan University, Taiwan, in 2003 and 2005, respectively, and the PhD degree in the Department of Computer Science and Information Engineering, National Cheng Kung University, Tainan, Taiwan, in November 2009. He then served the compulsory 11-month military service. From December 2011 to July 2017, he was an Assistant Research Fellow at the Department of Computer Science and Information Engineering, National Cheng Kung University. In 2017, he joined the Department of Computer Science and Information Engineering, National Taitung University, and now he is an Associate Professor. His current research interests include graph theory and algorithms, interconnection networks, and system-level diagnosis.

.....

Chia-Wei_Lee

Ling-Ju_Hung

Ralf_Klasing

Shih-Shun_Kao

Sun-Yuan_Hsieh

Declaration of interests

☒ The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

☐ The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: